SNT - Informatique embarquée et objets connectés

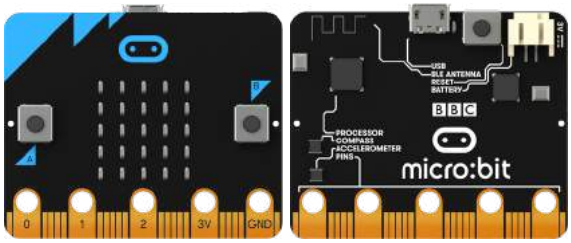
Activité élève BBC micro:bit et python

thomas.basso@ac-polynesie.pf

1. Présentation de la carte BBC micro:bit

BBC micro:bit est une carte à microcontrôleur (https://fr.wikipedia.org/wiki/Microcontr%C3%B4leur) conçue en 2015 au Royaume-Uni pour développer l'apprentissage de l'algorithmique et de la programmation. Pourvu de capteurs et d'actionneurs, 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**.

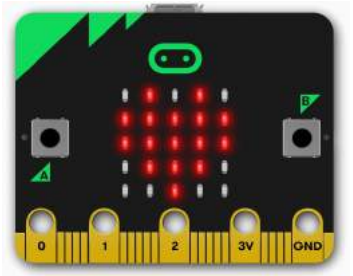


La carte micro:bit dispose des spécificités techniques (https://microbit.org/fr/guide/features/) suivantes :

- 25 LEDs programmables individuellement
- 2 boutons programmables
- Broches de connexion
- Capteurs de lumière et de température
- Capteurs de mouvements (accéléromètre et boussole)
- Communication sans fil, via Radio et Bluetooth
- Interface USB

2. Découverte des fonctionnalités

2.1 Commandes de base de l'afficheur, matrice de 5x5 LEDs voir vidéo explicative (en anglais) (https://youtu.be/qqBmvHD5bCw)



LED signifie Light Emitting Diode, Diode électroluminescente. La carte micro:bit en dispose de 25, toutes programmables individuellement, ce qui permet d'afficher du texte, des nombres et des images.

2.1.1 Afficher un texte "défilant" `display.scroll(string, delay=400)`

Nous allons commencer par afficher quelques informations sur l'afficheur.

Entrée [ ]:

```
from microbit import *
display.scroll("SNT")
```

La première ligne de ce programme importe la bibliothèque de fonctions micro:bit. La deuxième ligne fait défiler un message à l'écran. Cela n'arrive qu'une seule fois.

Exercice 1: Modifier le programme précédent pour qu'il affiche le texte de ton choix.

La vitesse de défilement peut être ralentie ou accélérée à l'aide du paramètre `delay` . Plus le nombre est grand, plus le défilement est lent.

Entrée [ ]:

```
from microbit import *
display.scroll("IL ETAIT UNE FOIS...", delay=20)
```

Exercice 2: Le défilement de la phrase du programme ci-dessus est trop rapide pour pouvoir la lire correctement. Modifie la valeur du paramètre `delay` pour qu'on puisse la lire facilement.

2.1.2 Afficher une "image" `display.show(image)`

Exécuter le programme suivant:

Entrée [ ]:

```
from microbit import *
display.show(Image.SAD)
```

Exercice 3: On constate que la carte est un peu triste. Modifier le programme précédent pour lui redonner de la joie.

Aide: ci-dessous la liste des images intégrées:

Image.HEART Image.HEART_SMALL Image.HAPPY Image.SMILE Image.SAD Image.CONFUSED Image.ANGRY Image.ASLEEP Image.SURPRISED Image.SILLY Image.FABULOUS Image.MEH Image.YES Image.NO Image.CLOCK12, Image.CLOCK11, Image.CLOCK10, Image.CLOCK9, Image.CLOCK8, Image.CLOCK7, Image.CLOCK6, Image.CLOCK5, Image.CLOCK4, Image.CLOCK3, Image.CLOCK2, Image.CLOCK1 Image.ARROW_N, Image.ARROW_NE, Image.ARROW_E, Image.ARROW_SE, Image.ARROW_S, Image.ARROW_SW, Image.ARROW_W, Image.ARROW_NW Image.TRIANGLE Image.TRIANGLE_LEFT Image.CHESSBOARD Image.DIAMOND Image.DIAMOND_SMALL Image.SQUARE Image.SQUARE_SMALL Image.RABBIT Image.COW Image.MUSIC_CROTCHET Image.MUSIC_QUAVER Image.MUSIC_QUAVERS Image.PITCHFORK Image.XMAS Image.PACMAN Image.TARGET Image.TSHIRT Image.ROLLERSKATE Image.DUCK Image.HOUSE Image.TORTOISE Image.BUTTERFLY Image.STICKFIGURE Image.GHOST Image.SWORD Image.GIRAFFE Image.SKULL Image.UMBRELLA Image.SNAKE

Prolongement: essayer plusieurs images intégrées.

Créer sa propre image

Chaque pixel LED sur l'affichage physique peut prendre une parmi dix valeurs. Si un pixel prend la valeur 0 c'est qu'il est éteint. Litéralement, il a une luminosité de zéro. En revanche, s'il prend la valeur 9 il est à la luminosité maximale. Les valeurs de 1 à 8 représentent des niveaux de luminosité entre éteint (0) et « au maximum » (9).

Entrée [ ]:

```
from microbit import *

bateau = Image("05050:"
               "05050:"
               "05050:"
               "99999:"
               "09990")

display.show(bateau)
```

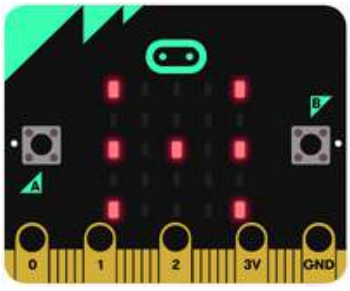
Comment dessiner une image? Chaque ligne de l'affichage physique est représentée par une ligne de nombres se terminant par : et entourée de guillemets doubles " . Chaque nombre indique une luminosité. Il y a cinq lignes de cinq nombres donc il est possible de spécifier la luminosité individuelle de chacune des cinq LED sur chacune des cinq lignes sur l'affichage physique. C'est ainsi que l'on crée une image.

Exercice 4: Le programme précédent crée et affiche l'image d'un bateau à deux mâts. Le modifier (en le recopiant) ci-dessous pour obtenir un bateau à un seul mât.

Entrée [ ]:

```
# Ecrire votre programme ici
```

Exercice 5: Rédiger ci-dessous le programme qui affiche l'image suivante:



Entrée [ ]:

```
# Ecrire votre programme ici
```

Remarque: on peux aussi écrire les images en une seule ligne:

Entrée [ ]:

```
from microbit import *

bateau = Image("05050:05050:05050:99999:09990")

display.show(bateau)
```

2.1.3 Les pixels (`display.set_pixel(x, y, val)`)

Vous pouvez régler la luminosité des pixels de l'affichage individuellement de 0 (désactivé) à 9 (luminosité maximale). Pour des informations sur les coordonnées de l'affichage, voir le guide pour matrice à LED (https://microbit.org/guide/hardware/leds/).

Exécuter le programme suivant:

Entrée [ ]:

```
from microbit import *
display.set_pixel(1, 4, 9)
```

Exercice 6: Recopier le programme précédent ci-dessous et le modifier pour allumer la LED du centre de la matrice.

Entrée [ ]:

```
# Ecrire votre programme ici

```

2.2 Boucle `while`

Le programme suivant utilise une boucle `while` pour faire clignoter le pixel central de manière répétée sur l'écran. La boucle `while` se répète tant que la condition spécifiée est vraie (`True`). Dans ce cas, nous avons dit que la condition est vraie. Cela crée une *boucle infinie*. Le code qui doit être répété est en retrait (c'est une "indentation" du texte).

L'instruction de veille `sleep()` provoque la pause du micro:bit pendant un nombre défini de millisecondes choisi entre parenthèses.

L'instruction `display.clear()` éteint l'affichage.

Exécuter le programme ci-dessous:

Entrée [ ]:

```
from microbit import *
while True:
    display.set_pixel(2, 2, 9)
    sleep(500)
    display.clear()
    sleep(500)
```

Avec un peu d'aléatoire (voir documentation sur le hasard (https://microbit-

micropython.readthedocs.io/fr/latest/tutorials/random.html))

Dans le programme suivant que vous exécuterez, on importe le module `random` de MicroPython et on l'utilise pour afficher un pixel au hasard sur la matrice.

Entrée [ ]:

```
from microbit import *
import random
n=random.randint(0,4)
p=random.randint(0,4)
display.set_pixel(n, p, 9)
```

Tester le programme précédent plusieurs fois de suite. Pour cela, redémarrer la micro:bit en appuyant sur le bouton `RESET` situé à l'arrière de la carte.

Exercice 7: Ecrire un programme ci-dessous qui allume successivement et indéfiniment des pixels au hasard à l'écran (aide: utiliser une boucle infinie).

Entrée [ ]:

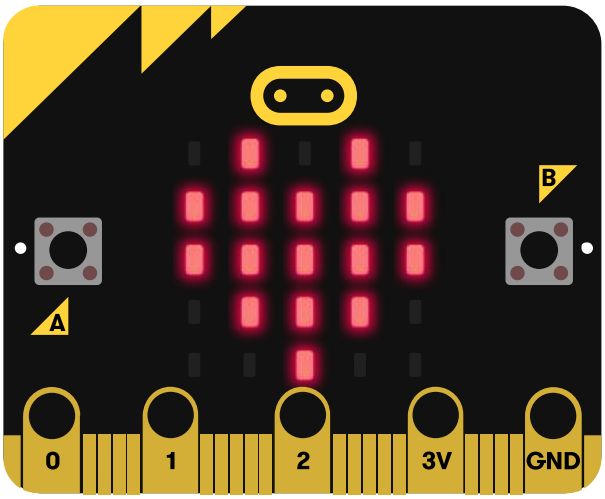
```
# Ecrire votre programme ici
```

Exercice 8: Ecrire ci-dessous un programme qui fait clignoter un coeur indéfiniment (voir illustration ci-dessous).

Entrée [5]:

```
# Illustration du résultat recherché
from IPython.display import IFrame
IFrame('https://makecode.microbit.org/---run?id=_RDdU5qgo0Cf7', width=300, height=300)
```

Out[5]:



Entrée [ ]:

```
# Ecrire votre programme ici

```

Créer une animation

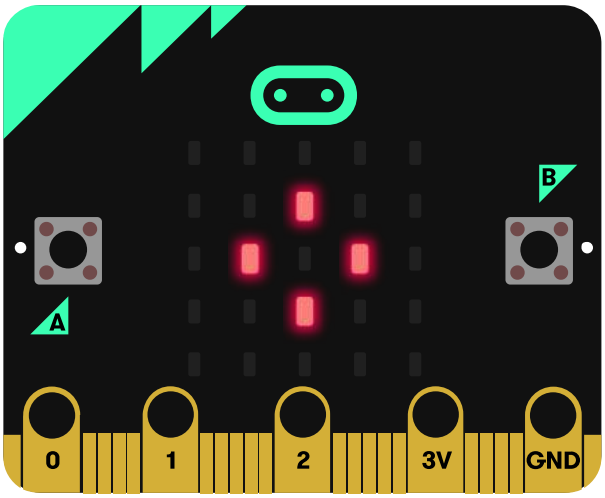
En affichant plusieurs images successives, on peut réaliser une animation.

Exercice 9: Ecrire un programme qui réalise l'animation suivante:

Entrée [6]:

```
# Illustration du résultat recherché
from IPython.display import IFrame
IFrame('https://makecode.microbit.org/---run?id=_4xi7Ct2DzWXK', width=300, height=300)
```

Out[6]:



Entrée [ ]:

```
# Ecrire votre programme ici

```

2.3 Boucle `for`

Le programme suivant utilise une boucle `for` pour faire défiler un pixel sur une ligne. Exécutez-le.

Entrée [ ]:

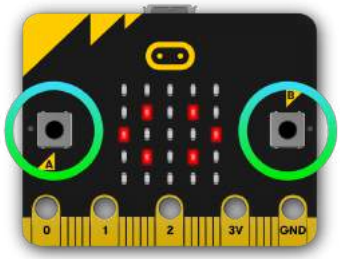
```
from microbit import *
while True:
    for i in range(5):
        display.set_pixel(i,0,9)
        sleep(200)
        display.clear()
```

Exercice 10 ***(la double boucle)*****:** S'inspirer du programme précédent pour réaliser un programme ci-dessous qui fait défiler un pixel sur tout l'écran.

Entrée [ ]:

```
# Ecrire votre programme ici

```

2.4 Les entrées boutons A, B et A+B - programmation événementielle (vidéo explicative) (https://youtu.be/t_Qujjd_38o)



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

Exemples avec le boutton A:

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

Exemple : Essayer le programme suivant qui fait défiler le texte "SNT" indéfiniment. On introduit l'**instruction conditionnelle** `if` qui va tester si le bouton A a été pressé (pendant le défilement du texte ou pendant la pause), auquel cas le programme s'arrête en exécutant la commande `break` .

Entrée [ ]:

```
from microbit import *
while True:
    display.scroll("SNT")
    sleep(200)
    if button_a.was_pressed():
        break
```

Instructions conditionnelles `if`, `elif`, `else`

Voici comment se structure une instruction conditionnelle. Selon la situation, il n'est pas forcément nécessaire d'utiliser `elif` ou `else` .

Entrée [ ]:

```
if quelque chose est vrai (``True``):
    # fais un truc
elif autre chose est vrai (``True``):
    # fais un autre truc
else:
    # fais encore autre chose.
```

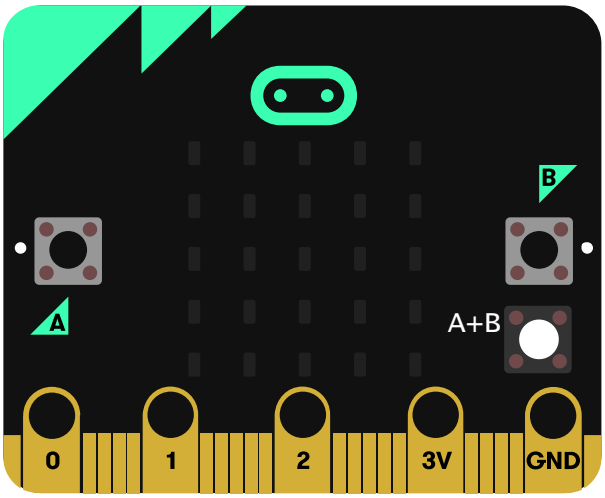
Exercice 11 : *Pierre feuille ciseaux!* Compléter le programme suivant dans lequel une pression simultanée sur les boutons A et B affichera une image de ciseaux. Sinon si, une pression sur le bouton A affichera une image de pierre. Sinon si, une pression sur le bouton B affichera une image de feuille. Il faudra créer vous-même l'image de la *pierre* et de la *feuille* avec un temps d'affichage de 0,5 seconde.

Voici pour exemple une illustration du résultat recherché: dans le simulateur suivant, clique sur le bouton A, le bouton B et le bouton qui simule la pression simultanée des boutons A et B.

Entrée [7]:

```
# Illustration du résultat recherché
from IPython.display import IFrame
IFrame('https://makecode.microbit.org/---run?id=_LYvH1C4LK2CT', width=300, height=300)
```

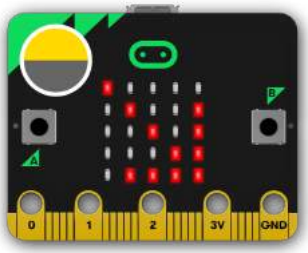
Out[7]:



Entrée [ ]:

```
# Ecrire votre programme ici

from microbit import *

pierre =

feuille =

ciseaux = Image("99009:"
                "99090:"
                "00900:"
                "99090:"
                "99009:")

while True:
    if button_a.is_pressed() and button_b.is_pressed():
        display.show(ciseaux)
        sleep(500)
    elif button_a.is_pressed():
        display.show(pierre)
        sleep(500)
    elif button_b.is_pressed():
        display.show(feuille)
        sleep(500)
    display.clear()
    sleep(100)
```

2.5 Capteur de lumière [(vidéo) (https://youtu.be/TKhCr-dQMBY)](https://youtu.be/TKhCr-dQMBY)



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.

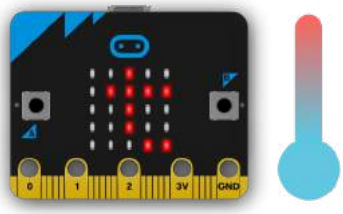
Exercice 12: Compléter le programme ci-dessous qui affiche une image de lune si on baisse la luminosité (en recouvrant la carte avec sa main par exemple) et un soleil sinon.

Entrée [ ]:

```
# Ecrire votre programme ici

from microbit import *

soleil = Image("90909:"
               "09990:"
               "99999:"
               "09990:"
               "90909:")

lune = Image("00999:"
             "09990:"
             "09900:"
             "09990:"
             "00999:")

while True:
    if display.read_light_level()>"remplir ici":
        display.show(soleil)
    else:
        display.show("remplir ici")
    sleep(10)
```

Prolongement: créer un programme qui affiche le niveau de luminosité et le tester avec la LED d'un téléphone portable ou une lampe-torche par exemple. Plus la luminosité sera élevée, plus il y aura de LEDs affichées sur la matrice.

2.6 Capteur de température (vidéo) (https://youtu.be/_T4N8O9xsMA)



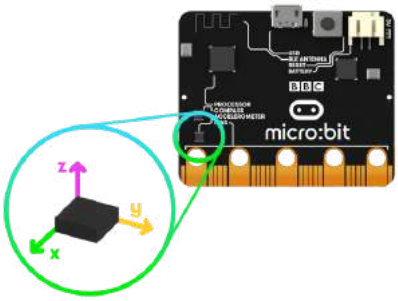
Le micro:bit n'a pas un capteur de température dédié. Au lieu de cela, la température fournie est en fait la température de la puce de silicium du processeur principal. Comme le processeur chauffe peu en fonctionnement (c'est un processeur ARM à grande efficacité), sa température est une bonne approximation de la température ambiante. L'instruction `temperature()` renvoie la température de la carte micro:bit en degrés Celsius.

Exercice 13: Ecrire un programme qui affiche la température (aide: on pourra utiliser l'instruction `display.scroll()` ; revoir le point 2.1.1).

Entrée [ ]:

```
# Ecrire votre programme ici

```

2.7 Accéléromètre (vidéo) (https://youtu.be/byngcwjO51U)



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

Si tu t'es déjà demandé comment un téléphone portable sait dans quel sens afficher les images sur son écran, c'est parce qu'il utilise un accéléromètre. Les manettes de jeux contiennent aussi des accéléromètres pour t'aider à tourner et à te déplacer dans les jeux.

La carte micro:bit est munie d'un accéléromètre. Il 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.

Dans l'exemple suivant à essayer, 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"

Entrée [ ]:

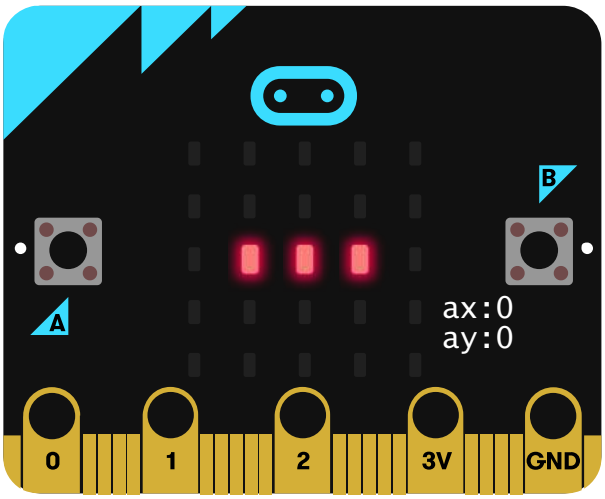
```
#Exemple
from microbit import *

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

Exercice 14: Compléter le programme suivant pour obtenir le résultat illustré dans le simulateur (survoler le simulateur avec la souris pour observer le résultat souhaité).

Entrée [8]:

```
# Illustration du résultat recherché
from IPython.display import IFrame
IFrame('https://makecode.microbit.org/---run?id=_d1HLyAa03Xdh', width=300, height=300)
```

Out[8]:



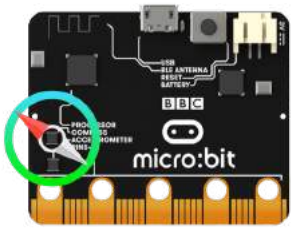
Entrée [ ]:

```
# Ecrire votre programme ici
from microbit import *

while True:
    abscisse = accelerometer.get_x()
    ordonnee = accelerometer.get_y()
    if abscisse > 500:
        display.show(Image.ARROW_E)
    elif abscisse < -500:
        display.show(Image.ARROW_W)
    elif ordonnee > 500:
        display.show(Image.ARROW_S)
    elif "remplir ici":
        "remplir ici"
    else:
        display.show("-")
```

Prolongement (*secouer les dés!*): Ecrire un programme qui simule un dé en affichant une face au hasard lorsque la micro:bit est secouée. On pourra utiliser l'instruction `accelerometer.is_gesture(shake)` qui teste si la carte est secouée. Plus d'informations sur les gestes ici (https://microbit-micropython.readthedocs.io/fr/latest/tutorials/gestures.html).

Pour le plaisir: Essayer le jeu d'évitement d'obstacles *"Simple Slalom"* utilisant l'accéléromètre. À récupérer sur cette page (https://microbit-micropython.readthedocs.io/fr/latest/accelerometer.html).

2.8 Boussole



La boussole détecte le champ magnétique de la Terre, nous permettant de savoir quelle direction la micro:bit indique. La boussole doit être étalonnée avant de pouvoir être utilisée. Pour cela, on utilise `compass.calibrate()` qui exécute un petit jeu: au départ, micro:bit fait défiler "Tilt to fill screen". Ensuite, incliner micro:bit pour déplacer le point au centre de l'écran autour jusqu'à ce que vous ayez rempli la totalité de l'écran.

La fonction `compass.heading()` donne le cap de la boussole sous la forme d'un entier compris entre 0 et 360, représentant l'angle en degrés, dans le sens des aiguilles d'une montre, avec le nord égal à 0.

Exercice 15: Compléter le programme suivant qui indique le Nord.

Entrée [ ]:

```
# Ecrire votre programme ici

from microbit import *

compass.calibrate()

while True:
    if compass.heading() < "remplir ici" or compass.heading() > "remplir ici":
        display.show(Image.ARROW_N)
    else:
        display.show(Image.DIAMOND_SMALL)
```

Prolongement: Améliorer le programme pour que le micro:bit indique "N", "S", "E" et "O" en fonction de l'orientation de la boussole.

Autre prolongement: fabriquer une station météo qui détermine la direction du vent.

Autre prolongement: étudier l'intensité du champ magnétique autour du périphérique (en utilisant la fonction `compass.get_field_strength()`). Plus d'informations sur les fonctions "boussole" ici (https://microbit-micropython.readthedocs.io/en/latest/compass.html).

3. IHM (Interface homme-machine)

Une IHM est un dispositif avec lequel l'humain peut envoyer ou récupérer de l'information avec l'objet connecté.

Notre IHM consistera en une fenêtre permettant l'acquisition de données provenant de la carte micro:bit et permettant également de transmettre des informations à la carte.

3.1 La carte transmet sa température (programme "PRG1")

Le programme suivant appelé ***"PRG1"*** affiche la température sur la matrice mais surtout **envoie la valeur de la variable "temp" et donc sa température à l'ordinateur via l'instruction `print(temp)`** .

Entrée [ ]:

```
# PRG1 à flasher dans la micro:bit (cliquer sur "Flasher")
# Programme à flasher avec le logiciel Mu en mode "BBC micro:bit"

from microbit import *     # Importe la bibliothèque microbit

while True:     # Boucle infinie
    temp=temperature()     # Initialise la variable temp à la valeur de la température
    display.scroll(temp)   # Affiche la température sur le microbit
    print(temp)            # Transmet la valeur de la variable à l'ordinateur
```

Cliquer sur le bouton `Flasher` du logiciel Mu pour téléverser et exécuter le programme dans la micro:bit (le logiciel Mu doit être en mode "BBC micro:bit", voir 2.2.2).

3.2 L'ordinateur reçoit et affiche la température

> Pour créer l'IHM, nous allons utiliser la bibliothèque Python ***tkinter*** qui va nous permettre de **construire une fenêtre pour interagir avec la micro:bit**.

> Nous utiliserons également la bibliothèque ***serial*** pour **gérer les entrées/sorties à travers les ports utilisés par le système**.

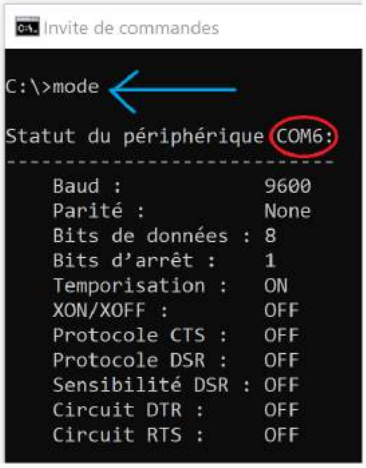
3.2.1 Identifier le port série (*port COM*) dédié à la micro:bit

Pour récupérer la valeur de la température, on va utiliser le port série (port USB de l'ordinateur). Il faut trouver le numéro de port sur lequel la micro:bit est branchée pour pouvoir communiquer.

Sur Windows:

Ouvrir l'invite de commande en tapant **cmd** dans la barre de recherche windows.

Taper **Mode** pour avoir la liste des périphériques branchés à l'ordinateur



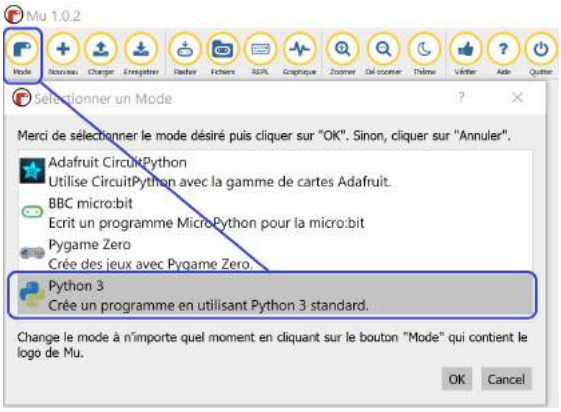
(COM1 et COM 2 sont parfois la souris et le clavier)

Sur MacOS:

Ouvrir le Terminal et saisir la commande: "ls /dev/tty.usb*

Votre port devrait apparaitre sous la forme: "/dev/tty.usbmodem14102"

3.2.2 Mettre le logiciel Mu en mode "Python 3"



3.2.3 Exécuter le programme créant l'IHM ci-dessous (programme "PRG2")

Entrée [ ]:

```
# PRG2
# Programme à exécuter avec le logiciel Mu en mode "Python 3" (cliquer sur "Lancer")

import serial                # Importe la bibliothèque serial
from tkinter import *        # Importe la bibliothèque tkinter
port = "votre port"          # "COM6" par exemple pour Windows ou "/dev/tty.usbmodem14102" s
ser = serial.Serial(port)    # Déclaration de la variable ser qui ouvre le port série
ser.baudrate = 115200        # Vitesse de transmission

data = ser.readline()            # On lit sur le port série ce qui est envoyé par la micro:b
ma_fenetre = Tk()                # On crée une fenêtre qui sert d'interface avec la carte
ma_fenetre.geometry("400x300")   # Fixe la taille de la fenêtre
ma_fenetre.title("IHM")          # Donne un titre à la fenêtre
affich_temp = Label(ma_fenetre, text=data, font=("arial", 70), fg="green")  # Crée le texte
affich_temp.pack()               # Positionne ce texte dans notre fenêtre
ma_fenetre.mainloop()            # Pour que la fenêtre reste ouverte
```

Une fenêtre s'ouvre affichant la température lue par la carte.

Exercice 16: Recopier le programme précédent ci-dessous puis modifier les paramètres de la fenêtre IHM (modifier les dimensions, le titre, la police de caractères, la taille de la police, la couleur de la police...). Le tester.

Entrée [ ]:

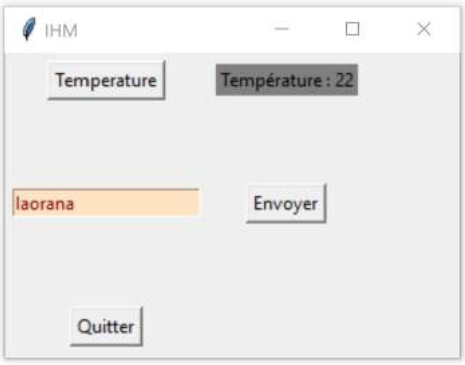
```
# Ecrire votre nouveau programme PRG2 ici
# Programme à exécuter avec le logiciel Mu en mode "Python 3" (cliquer sur "Lancer")
```

Exercice 17: Modifier le programme "PRG1" (côté carte) pour que la fenêtre affiche cette fois **le niveau de luminosité** (de 0 à 255) lu par la carte (voir le point "2.5 Capteur de lumière"). Il n'est pas nécessaire de modifier le PRG2 côté ordinateur.

Entrée [ ]:

```
# Ecrire votre nouveau "PRG1" ici
# Programme à flasher avec le logiciel Mu en mode "BBC micro:bit"
```

3.3 Réalisation d'une IHM plus complexe



Nous allons ajouter un **bouton pour récupérer la température** fournie par la carte et un **champ de saisie de texte** pour qu'il soit affiché sur la carte. Pour cela, nous allons utiliser les programmes *PRG1* et *PRG2* suivants.

PRG1 à flasher dans la carte micro:bit

Entrée [ ]:

```
# PRG1 à flasher dans la micro:bit (cliquer sur "Flasher")
# Programme à flasher avec le logiciel Mu en mode "BBC micro:bit"

from microbit import *

uart.init(baudrate=115200)              # Vitesse de transfert

display.scroll('Pret')                  # La carte affiche "pret"

while True:
    if uart.any():                      # Si la carte reçoit une transmission de l'ordinat
        message = uart.read()           # On stocke la valeur dans la variable "message"
        if message == b'temperature':   # Si la valeur est "temperature",
            temp = temperature()
            print(temp)                 # La carte envoie la valeur de la température à l'
            display.scroll(temp)        # La carte affiche la température sur sa matrice
        elif message == b'pacman':      # Sinon si le message reçu est "pacman"
            display.show(Image.PACMAN)  # La carte affiche l'image de Pacman
        else:                           # Sinon enfin
            textedefil = str(message, 'UTF-8')     # Transforme le message en chaîne de car
            display.scroll(textedefil, delay=80)   # La carte affiche le message reçu sous
    sleep(1000)
```

PRG2 à lancer dans l'ordinateur

Entrée [ ]:

```
# PRG2
# Programme à exécuter avec le logiciel Mu en mode "Python 3" (cliquer sur "Lancer")

import serial, time

from tkinter import *

port = "votre port"         # "COM6" par exemple pour Windows ou "/dev/tty.usbmodem14102" s
baud = 115200
s = serial.Serial(port)
s.baudrate = baud


def demande_temperature():           # Fonction pour demander la température à la carte mic
    s.write(b'temperature')
    data = s.readline()
    data = int(data[0:4])
    print(data)
    texte_temperature.set("Température : "+str(data))

def envoie_message():                # Fonction pour envoyer un message à faire défiler sur
    message_bytes = bytes(texte_message.get(), 'utf-8')
    s.write(message_bytes)

def quitter():         # Fonction pour fermer la fenêtre d'interface
    s.close()
    ma_fenetre.destroy()


# Fenêtre principale
ma_fenetre = Tk()
ma_fenetre.title("IHM")
ma_fenetre.geometry("300x200")

# Création d'un bouton pour recuperer la temperature
button_temperature = Button(ma_fenetre, text="Temperature", command=demande_temperature)
button_temperature.grid(row=1, column=0, padx=5, pady=5)

# Creation d'un label pour afficher la temperature
texte_temperature = StringVar()
texte_temperature.set("Température : ")
label_temperature = Label(ma_fenetre, textvariable=texte_temperature , bg="grey")
label_temperature.grid(row=1, column=1, padx=5, pady=5)

# Création d'un bouton pour envoyer un message
button_message = Button(ma_fenetre, text ="Envoyer", command=envoie_message)
button_message.grid(row=2, column=1, padx=5, pady=50)

# Création d'un champ de saisie d'un message
texte_message = StringVar()
texte_message.set('Iaorana')
champ_message = Entry(ma_fenetre, textvariable=texte_message, bg="bisque", fg="maroon", wid
champ_message.focus_set()
champ_message.grid(row=2, column=0, padx=5, pady=5)

# Création d'un bouton Quitter
button_quitter = Button(ma_fenetre, text="Quitter", command=quitter)
button_quitter.grid(row=3,column=0, padx=5, pady=5)
```

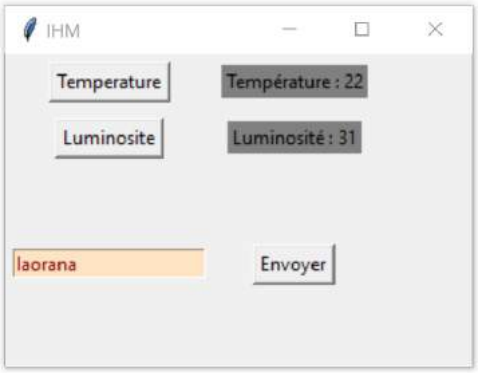
```
ma_fenetre.mainloop()
```

Vérifier le bon fonctionnement des programmes précédents en testant l'interface.

Question: Que se passe-t-il lorsqu'on envoie le texte "pacman" via le champ de saisie de l'interface?

Exercice 18: Ajouter un bouton "Luminosité" à l'IHM qui récupère l'intensité de la luminosité lue par le capteur de luminosité de la carte. On pourra prendre exemple sur le bouton "Température".

Résultat attendu:



Entrée [ ]:

```
# PRG1 à flasher dans la micro:bit (cliquer sur "Flasher")
# Programme à flasher avec le logiciel Mu en mode "BBC micro:bit"
```

Entrée [ ]:

```
# PRG2
# Programme à exécuter avec le logiciel Mu en mode "Python 3" (cliquer sur "Lancer")
```

The End ----------

---------- thomas.basso@ac-polynesie.pf