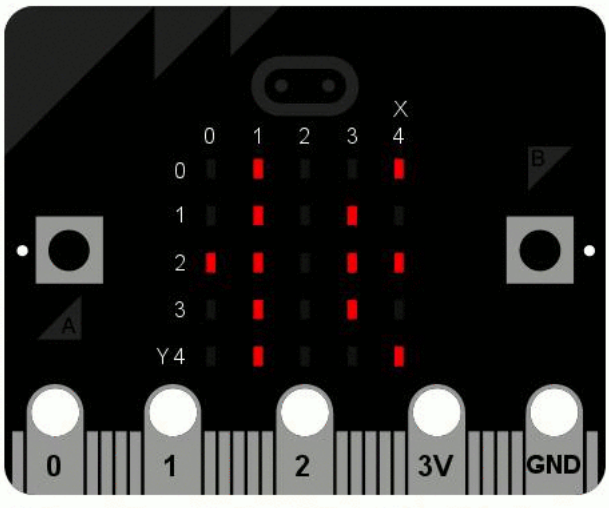
1 Interface Homme Machine

Définition 1

Une Interface Homme Machine ou IHM est un ensemble de dispositifs physique (boutons, curseurs) et logiciels (interface graphique) permettant d'échanger des informations avec une machine.

Exercice 1 *Manipulation de diodes et boucle `for`*

L'écran de la carte micro:bit est une grille ou matrice constituée de 25 diodes électroluminescentes ou LED. Chaque diode est repérée par ses coordonnées x et y variant entre 0 et 4, comme ci-dessous. De plus une diode peut être émettre un signal lumineux d'intensité variable entre 0, diode éteinte, et 9, intensité maximale.
En Python, l'instruction `display.set_pixel(x, y, v)` permet d'allumer le diode de coordonnées x et y avec l'intensité v.



Source : https://microbit.org/fr/guide/python/

1. Ouvrir le logiciel Mu en mode micro:bit, saisir le programme ci-dessous dans l'éditeur avant de l'enregistrer dans un dossier pertinent sous le nom exo1-microbit.py puis de le transférer sur la carte.

Programme 1

```
1 from microbit import *
2
3 for x in range(5):
4     display.set_pixel(x,x,9)
```

2. Décrire l'effet du programme sur la carte.

.....

Préciser le rôle de chaque instruction.

.....

.....

.....

3. Modifier le programme pour qu'il allume tous les pixels de coordonnées `(x,0)` avec $0 \leqslant x \leqslant 4$.
4. Modifier le programme pour que les diagonales de la matrice de diodes s'allument en forme de croix.

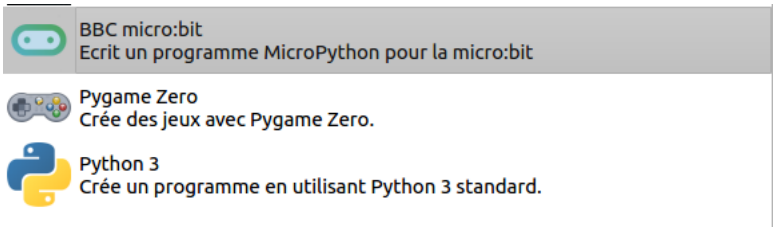
Exercice 2 *Commande d'un actionneur par une IHM*

L'objectif de cet exercice est de réaliser une IHM simple permettant d'envoyer un code de trois chiffres `xyv` à la carte pour qu'elle d'affiche (actionneur) la diode de coordonnées `(x,y)` avec l'intensité `v`.

Les systèmes embarqués sont de nos jours souvent connectés à un réseau local ou à l'Internet ce qui permet de communiquer avec eux à travers des IHM déportées sur le Web ou un smartphone. On va écrire deux programmes `Python` : un programme pour l'IHM s'exécutant uniquement sur l'ordinateur et un autre pour le système informatique embarqué qui sera transféré sur la carte.

Partie A : réalisation du système embarqué

Dans cette partie, on utilise `Mu` en mode `micro:bit`.



Ouvrir le logiciel `Mu` en mode `micro:bit`, saisir le programme ci-dessous dans l'éditeur avant de l'enregistrer dans un dossier pertinent sous le nom `exo2-microbit.py` puis de le transférer sur la carte.

Programme 2

```
1  from microbit import *
2
3  #initialisation de la vitesse de transmission en bauds
4  uart.init(baudrate=9600)
5
6
7  while True:
8      #s'il y a un message dans la file de réception
9      if uart.any():
10         #lecture du message qui est en bytes
```

```
        msg_bytes=uart.read()
        #conversion en chaîne de caractères en précisant l'encodage
        msg = str(msg_bytes, "utf-8")
        #conversion en entier du premier caractère du message (x)
        x = int(msg[0])
        #conversion en entier du second caractère du message (y)
        y = int(msg[1])
        #conversion en entier du troisième caractère du message (v)
        v = int(msg[2])
        display.set_pixel(x, y, v)
    sleep(1000)
    display.clear()
```

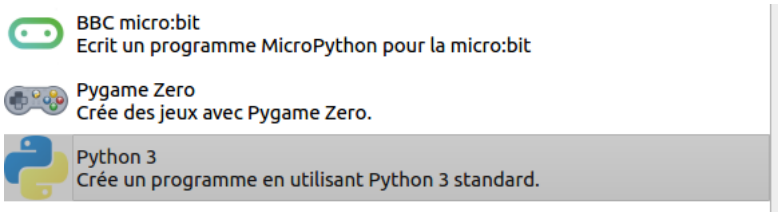
Méthode *Communication série côté micro:bit*

Le module uart permet de communiquer avec l'ordinateur connecté sur le port série USB.

- uart.init(baudrate=9600) est nécessaire pour fixer la vitesse de transmission en bauds.
- Pour lire un message sur l'entrée série, on vérifie s'il reste des messages à lire dans la file d'attente avec uart.any() puis on lit toute la file de messages avec uart.read() ou le message de tête jusqu'au prochain saut de ligne avec uart.readline(). On peut effacer les caractères invisibles de saut de ligne avec uart.readline().strip()
- Un message reçu msg_bytes est un flux d'octets, de type bytes en Python, avant de le traiter, il faut le convertir en chaîne de caractères avec str(msg_bytes, 'utf-8').
- Pour transmettre un message msg de type chaîne de caractères, on le convertit en bytes avec msg_bytes = bytes(msg, 'utf-8') puis on l'écrit sur la sortie série avec print(msg_bytes).

Partie B : réalisation de l'IHM

Dans cette partie, on utilise Mu en mode Python.



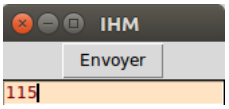
1. Récupérer le fichier exo2-pc.py dont on donne le code ci-dessous et l'enregistrer dans un dossier pertinent.

Programme 3

```
from serial import *
from tkinter import *

#port série à découvrir en ligne de commande avec mode
port = "COM4"
serie = Serial(port)
#même vitesse de transmission que sur la carte
serie.baudrate = 9600

def envoie_message():
    """Fonction d'envoi"""
    msg = texte_message.get()
    message_bytes = bytes(msg, "utf-8")
    serie.write(message_bytes)

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

# Création d'un bouton pour envoyer un message
button_message = Button(ma_fenetre, text ="Envoyer", command =
     envoie_message)
#positionnement du bouton
button_message.pack()

# Création d'un champ de saisie d'un message
texte_message = StringVar()
champ_message = Entry(ma_fenetre, textvariable= texte_message,
                      bg ="bisque", fg="maroon", width="20")
#on donne le focus au champ de saisie
champ_message.focus_set()
#positionnement du champ
champ_message.pack()

#boucle de capture d'événements
ma_fenetre.mainloop()
```

2. Exécuter le programme en cliquant sur l'icône `Lancer`, une fenêtre graphique comme ci-dessous s'affiche. Écrire 115 dans le champ de saisie puis cliquer sur le bouton `envoyer`, le pixel de coordonnées `(1,1)` devrait s'allumer avec une intensité moyenne.

Pour arrêter l'IHM, fermer la fenêtre graphique puis cliquer sur `Arrêter` dans la barre de menu de `Mu`.



3. Revenons sur le code de l'IHM.

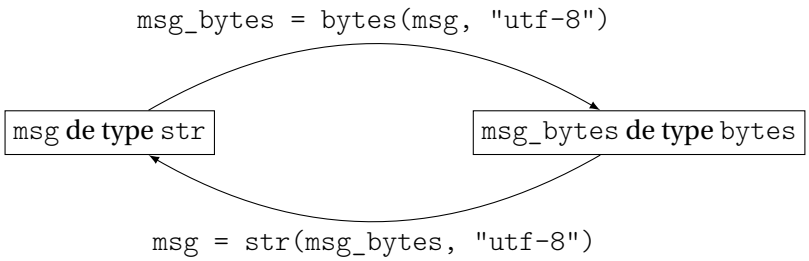
La réalisation de l'interface graphique est commentée dans le code, elle s'appuie sur le module tkinter qu'on charge avec `from tkinter import *`.

La communication série avec la carte `micro:bit` s'appuie sur le module `serial` qu'on charge avec `from serial import *`.

Méthode *Communication série côté* `ordinateur`

Le module `serial` permet de communiquer avec une carte connectée sur le port série USB.

- `serie = Serial(port)` permet d'ouvrir une connexion série sur le port de l'ordinateur où la carte est connectée. Sous Windows ce port peut être découvert avec la commande `mode` depuis la console lancée avec `cmd` dans la barre de recherche. Si l'accès n'est pas autorisé, il suffit de tester les différentes possibilités : `"COM3"`, `"COM4"` ou `"COM5"`.
- On commence par fixer la vitesse de transmission `serie.baudrate` à la même valeur que celle choisie sur la carte.
- Comme sur la carte les messages transmis par communication série sont nécessairement en `bytes`, ce qui nécessite une conversion entre chaîne de caractères de type `str` et `bytes` avant transmission ou après réception.



- On lit un message en `bytes` sur l'entrée série, avec `serie.read()` ou `serie.readline()` si on veut s'arrêter au prochain saut de ligne.
- On écrit un message en `bytes` sur l'entrée série, avec `serie.write(msg_bytes)`

Partie C : prolongements

1. Modifier le programme de la carte `exo2-microbit.py` et l'enregistrer sous le nom `exo2Part2-microbit.py`, pour que la carte allume pendant une seconde :

- la diagonale de diodes de coordonnées `(x, x)` avec $0 \leqslant x \leqslant 4$ si on transmet le message `"a"` par l'IHM ;
- l'autre diagonale de diodes si on transmet le message `"b"` par l'IHM ;
- une croix formée des deux diagonales si on transmet le message `"c"` par l'IHM.

Exercice 3 *Acquisition des données d'un système embarqué par une IHM*

L'objectif de cet exercice est de réaliser une IHM simple permettant d'acquérir la composante selon l'axe x de l'accélération mesurée par la capteur de la carte micro-bit et de l'afficher dans l'interface graphique d'une IHM.

Partie A : réalisation du système embarqué

Dans cette partie, on utilise Mu en mode micro:bit.

1. Dans un fichier exo3-microbit.py, compléter le programme ci-dessous pour qu'il envoie sur la sortie série la valeur capturée par accelerometer.get_x() si le caractère "c" est reçu sur l'entrée série.

Programme 4

```
from microbit import *

uart.init(baudrate=9600)

while True:
    if uart.any():
        #à  compléter
    sleep(1000)
```

2. Enregistrer puis transférer le programme sur la carte.

Partie B : réalisation de l'IHM

Dans cette partie, on utilise Mu en mode Python.

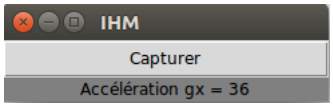
1. Récupérer le fichier exo3-pc-eleve.py dont le code est donné ci-dessous et l'enregistrer dans un dossier pertinent.

Programme 5

```
from serial import *
from tkinter import *

port = "COM4"
serie = Serial(port)
serie.baudrate = 9600

def capture():
    """Fonction de capture"""
    serie.write(bytes("c", "utf-8"))
    data = str(serie.readline().strip(), "utf-8")
    texte_gx.set("Accélération gx = " + data)

# Fenêtre principale
```

```
15 ma_fenetre = Tk()
16 ma_fenetre.title("IHM")
17
18 # Création d'un bouton pour envoyer un message
19 button_message = Button(ma_fenetre, text ="Capturer", command =
       capture)
20 #positionnement du bouton
21 button_message.pack(fill = X)
22
23 # Creation d'un label pour afficher la composante gx de l'accélé
       ration
24 texte_gx = StringVar()
25 texte_gx.set("Accélération gx = ")
26 label_gx = Label(ma_fenetre, textvariable = texte_gx , bg ="grey",
       width="30", justify = LEFT)
27 label_gx.pack(fill = X)
28
29 #boucle de capture d'événements
30 ma_fenetre.mainloop()
```

2. Exécuter le programme en cliquant sur l'icône `Lancer`, une fenêtre graphique comme ci-dessous s'affiche. Vérifier qu'un clic que le bouton `capturer` permet bien d'afficher la valeur souhaitée dans l'IHM. Incliner la carte pour faire varier cette valeur.



3. Décrire l'action déclenchée par l'appui sur le bouton `capturer`.

.....

.....

.....

.....

Partie C : prolongements

a. En mode `micro:bit`, modifier le programme de la carte `exo3-microbit.py` et l'enregistrer sous le nom `exo3Part2-microbit.py`, pour que la carte transmette sur sa sortie série un caractère différent selon l'angle `compass.heading()` avec le nord magnétique mesuré par son capteur boussole :

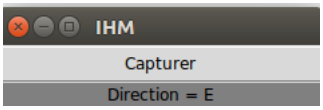
- `"N"` pour Nord si l'angle en degrés est supérieur à 315 ou inférieur à 45 ;
- `"E"` pour Est si l'angle est entre 45 et 135 ;
- `"S"` pour Sud si l'angle est entre 135 et 225 ;

- "O" pour Ouest sinon.

⚠ Ne pas oublier de calibrer la boussole avec `compass.calibrate()`

b. En mode Python, lancer l'IHM programmée dans `exo3-pc-eleve.py` et vérifier que les directions transmises par la carte sont bien affichées. On peut adapter l'affichage.

⚠ En cas d'erreur, relancer uniquement le programme de l'IHM.



2 Objet connecté

Définition 2

Un objet connecté est un système informatique embarqué disposant d'une connexion à un réseau local ou à L'Internet. Actuellement, il existe plus d'objets que d'humains connectés à Internet et leur nombre va augmenter fortement dans les prochaines années. On parle d'IOT pour Internet Of Things.

Exercice 4 *Transmission de données sur le Web*

L'objectif de cet exercice est d'illustrer la transmission de données d'un objet connecté, la carte micro:bit, vers une IHM en ligne, un canal public de visualisation sur le Web d'URL https://thingspeak.com/channels/846201. Le site Web thingspeak fournit des outils de visualisation et d'analyse de données transmises par des objets connectés. Il est aussi possible de récupérer des échantillons de données au format CSV.
Comme on ne dispose pas du module d'extension Wifi pour micro:bit, on utilise un programme sur l'ordinateur connecté pour servir de passerelle entre la carte et l'Internet .

1. Ouvrir Mu en mode micro:bit et compléter dans un fichier `exo4-microbit.py` le programme ci-dessous pour que toutes les 5 secondes il mesure la valeur de la composante en x de l'accélération, la fasse défiler sur l'écran puis la transmette sur la sortie série. Enregistrer et transférer le fichier sur la carte.

Programme 6

```
from microbit import *
uart.init(baudrate=9600)
while True:
    #à  compléter
```

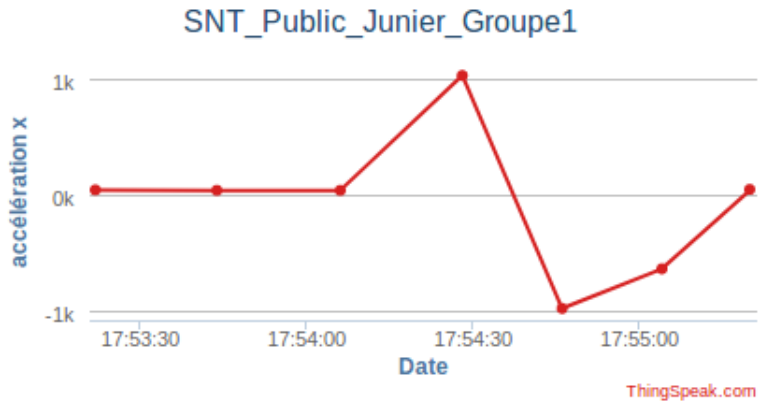
2. Récupérer l'un des 24 fichiers `exo4-pc-ChannelX-fieldY.py` et l'enregistrer dans un dossier pertinent.

Programme 7 *exo4-pc-ChannelX-fieldY.py*

```
from serial import *
from requests import *

serie = Serial("/dev/ttyACM0", 9600)
#Channel Groupe 1 : https://thingspeak.com/channels/846201
#Champ rempli : field2
apiKey = "DEMANDERPROFESSEUR"

while True:
    #récupération des données sur l'entrée série
    msg_bytes = serie.readline().strip()
    msg = str(msg_bytes, "utf-8")
    #transmission des données par requête HTTP
    r = get("https://api.thingspeak.com/update.json",
            params = {"field2" : msg , "key" : apiKey})
```

3. Passer Mu en mode Python et lancer l'exécution du programme exo4-pc-ChannelX-fieldY.py, qui va retransmettre les données acquises par la carte vers une interface Web de visualisation. Dans un navigateur Web, ouvrir la page d'URL https://thingspeak.com/channels/846201, le graphique des données est celui du du champ fieldX transmis comme paramètre dans l'URL (dernière ligne du programme ci-dessus).

4. Incliner la carte pour faire varier la valeur de la composante en x de l'accélération et observer si le graphique en ligne se met à jour.



5. Modifier le programme de la carte pour qu'il transmette les mesures de température du processeur.

Exercice 5 *Récupération de données sur le Web*

Un objet connecté peut transmettre mais aussi recevoir des informations sur le réseau : des mises à jour logicielles mais aussi des données.
Pour illustrer cet aspect, on va récupérer des données de trafic d'une autoroute du Massachussets , disponibles sur le canal public https://thingspeak.com/channels/38629 de thingspeak. Le champ field1

représente la densité de véhicules dans un sens et le champ field2 dans l'autre. Ces mesures, prises toutes les 15 secondes sur certains intervalles de temps fixés, sont transmises par des objets connectés : une caméra reliée à un nano-ordinateur Raspberry Pi où s'exécute un programme de traitement d'image pour détecter les véhicules. Plus de détail, sur cette page.

1. Ouvrir Mu en mode Python, récupérer le fichier exo5-pc.py, l'enregistrer dans un dossier pertinent puis lancer son exécution.

Programme 8 *exo5-pc.py*

```
from serial import *
from requests import *
import time

port = "COM4"
serie = Serial(port)
serie.baudrate = 9600

reponse = get("https://api.thingspeak.com/channels/38629/fields/1.
    json?results=50")
data_json = reponse.json()
#parcours en boucle des mesures
for releve in data_json['feeds']:
    # la mesure releve['field1'] est une chaine de caractères
    #de la forme '25.00000', on la convertit en décimal avec float
    #puis on tronque à l'entier avec int
    #avant de reconvertir en chaine avec un saut de lignes
    traffic = str(int(float(releve['field1']))) + '\r\n'
    serie.write(bytes(traffic, 'utf-8'))
    time.sleep(4) #attente de 4 secondes
```

2. Passer en mode micro:bit et compléter dans un fichier exo5-microbit.py le programme ci-dessous pour que toutes les secondes il lise la mesure transmise sur son entrée série puis la fasse défiler sur l'écran. Enregistrer et transférer le fichier sur la carte. Vérifier que l'écran affiche bien les données de traffic du field1 du canal public https://thingspeak.com/channels/38629.

Programme 9

```
from microbit import *

uart.init(baudrate=9600)
while True:
    if uart.any():
        #à  compléter
    display.clear()
    sleep(2000)
```