SCIENCES NUMERIQUES ET TECHNOLOGIE
2de – L'informatique embarquée.

FICHE DE TRAVAIL

Activité 1

Après avoir visionné la vidéo ci-dessous, vous établirez une liste des différents capteurs présents dans une voiture autonome. Pour chaque capteur, vous expliquerez, en quelques lignes, son principe de fonctionnement et son intérêt.

Expliquez en quelques lignes ce qu'est le Deep Learning (vous pourrez faire des recherches sur le web) et comment il est utilisé dans le cas de la voiture autonome.

Qu'est-ce que "la chaîne de traitement d'une voiture autonome" (il est possible, et même souhaitable, de faire un schéma) ?

Activité 2

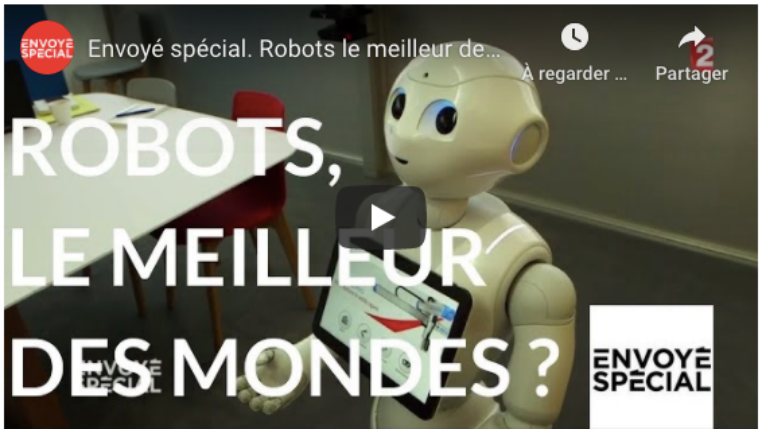
Toutes les réponses que vous donnerez devront être rédigées à l'aide d'un traitement de texte

Exercice 1 :

Selon vous, qu'est-ce qu'un robot ? Vous rédigerez une première réponse sans faire de recherche sur le Web. Dans un deuxième temps, vous répondrez à la même question après avoir fait des recherches sur le Web. Il faudra bien distinguer votre première et votre deuxième réponse.

Exercice 2 :

L'introduction des robots sur le marché du travail pose question. Après avoir visionné la vidéo ci-dessous, vous rédigerez un texte qui exposera les arguments en faveur et en défaveur de cette introduction.



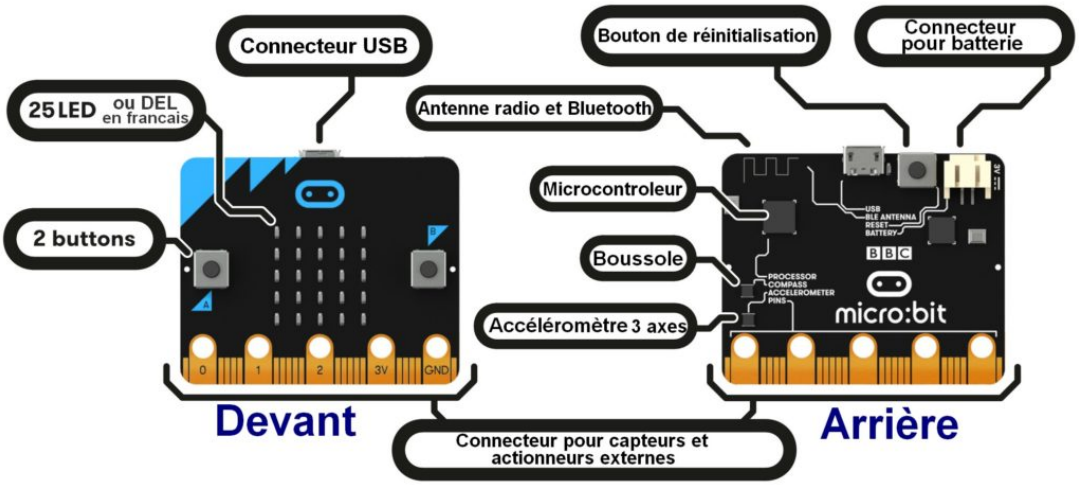
Activité 3

Prise en main de la micro:bit et de mu-editor.

Le ***BBC micro:bit*** est un ordinateur à carte unique basé sur le processeur ARM conçu par la BBC pour l'initiation à la programmation de système embarqué.

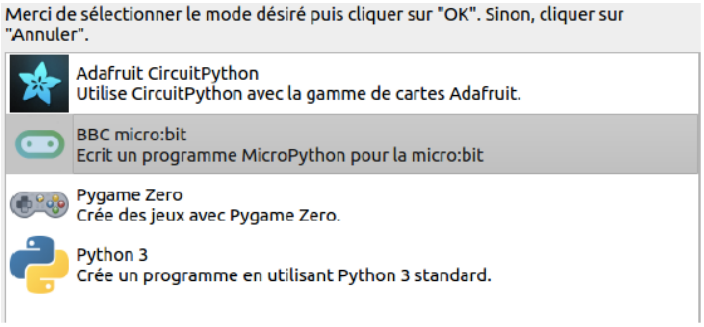
Un système embarqué est défini comme un système électronique et informatique autonome, souvent temps réel, spécialisé dans une tâche bien précise. Le terme désigne aussi bien le matériel informatique que le logiciel utilisé. Ses ressources sont généralement limitées. (wikipedia)

Le guide de présentation en ligne est disponible sur https://microbit.org/fr/guide/



Différents langages sont disponibles pour programmer cette carte. Nous nous limiterons ici au langage Python.

- Créer un dossier MicroBit
- Brancher votre micro:bit avec un cable usb
- Télécharger et installer mu-editor. Puis lancer le programme et choisissez d'écrire un programme pour micro:bit



- Créer un nouveau programme nommé prog1.py, copier le code ci-dessous puis enregistrer le dossier que vous venez de créer.

```
from microbit import *

display.show("Hello World")
```

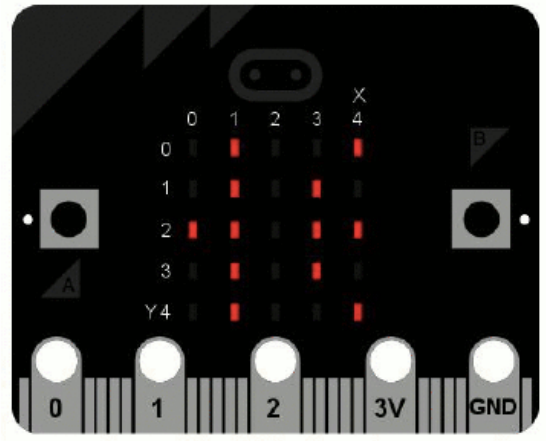
- Pour le téléverser dans la micro:bit, vous avez juste à cliquer sur l'icone 

Vous devriez voir le texte « Hello World » défiler sur les LED.

Activité 4

Les LED

Il y a 25 LED sur la afce avant du micro:bit qui peuvent être repérées par des coordonnées. (voir schéma ci-desous)



Pour afficher une led, il suffit d'utiliser la fonction : **microbit.display.set_pixel**(*x, y, value*)

- **x,y** sont les coordonnées de la led
- **value** définie la luminosité de la led (entre 0 et 9)
  0 : la led est éteinte 9 : la luminosité est maximum.

Exercice 1:

1) Analyser et expliquer ce qui fait de cet algorithme suivant :

```
from microbit import *

display.clear()

for i in range(5):
    display.set_pixel(i, 2, 9)
    sleep(1000)
    display.set_pixel(i, 2, 0)
```

2) Compléter le programme nommé led.py afin que lorsque le pixel arrive à droite, il revienne en arrière.

Activité 5

Les différents capteurs.

Vous aurez souvent besoin que votre programme soit dans l'attente que quelque chose se produise. Pour ce faire, vous devrez le faire « boucler » sur un morceau de code qui définit comment réagir à certains événements attendus comme l'appui sur un bouton.

Les boutons :

Si nous voulons que la micro:bit réagisse aux événement « *appui sur un bouton* » nous devrions le mettre dans une boucle infinie et vérifier si le bouton is_pressed.

- Créer un programme nommé bouton.py
- Recopier et téléverser le programme suivant :

```
from microbit import *

while True:
    if button_a.is_pressed():
        display.show(Image.HAPPY)
    else:
        display.show(Image.SAD)
```

Expliquer ce que fait ce programme :

Exercice 2 :

Ecrire un programme nommé bouton2.py qui affiche votre prénom si vous appuyez sur le bouton A, qui affiche votre nom si vous appuyer sur le bouton B et votre date de naissance si vous appuyer sur les boutons A et B en même temps.

Exercice 3 :

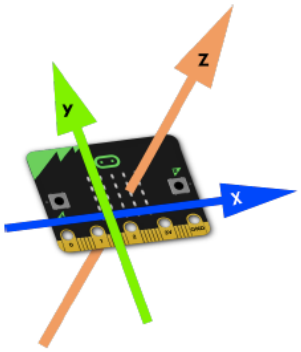
Ecrire un programme nommé lancerde.py qui affiche un entier au hasard entre 1 et 6 lorsqu'on appuie sur le bouton A. (Simulation d'un lancer de dé)

Aide : Importer la bibliothèque random et utiliser la fonction random.randint(1, 6)

L'accéléromètre :

Le BBC micro:bit est munit 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.



Il y a une fonction pour chaque axe qui renvoie un nombre positif ou négatif qui indique une mesure en milli-g. Lorsque la lecture est de 0, vous êtes « aligné » selon cet axe.

Nous allons créer un « niveau à bulle » très simple qui utilise la fonction get_x pour mesurer l'alignement de l'appareil selon l'axe des X :

- Créer un programme nommé niveaubulle.py
- Recopier et téléverser le programme suivant :

```
from microbit import *

while True :
        lecture = accelerometer.get_x()
        if lecture > 20 :
                display.show("D")
        elif lecture < -20 :
                display.show("G")
        else :
                display.show("-")
```

1) Que représente « lecture » ?

2) Quand le micro:bit affiche-t-il « - » ?

Exercice 4 :

La fonction accelerometer.was_gesture('shake') permet de savoir si vous avez secoué la micro:bit. En utilisant cette fonction, céer un programme nommé lancerde2.py qui simule un lacer de dé. (lorsque vous secouez le miro:bit, il affiche un nombre compris entre 1 et 6).

Exercice 5 :

Ecrire un programme nommé accelo_pixel.py qui permet de faire déplacer un point sur l'écran en fonction de l'inclinaison de la micro:bit

Température :

La micro:bit possède un capteur de température que vous pouvez utiliser dans vos programmes en utlisant la fonction temperature().

Exercice 6 :

Ecrire en programme nommé temperature.py qui affiche la température lorsqu'on appuie sur le bouton A.

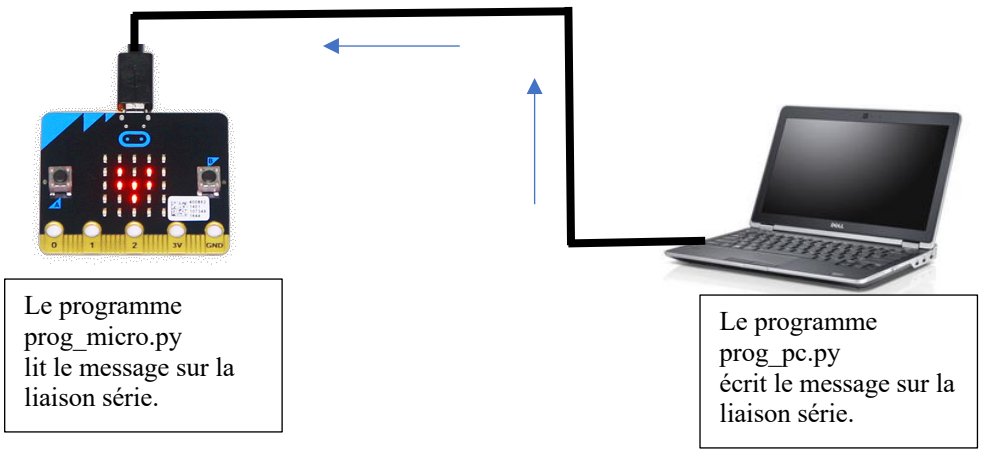
Activité 6

Interface Homme Machine, IHM

Nous allons créer une interface homme-machine. C'est-à-dire à dire des boutons sur l'ordinateur qui vont permettre de commander la micro:bit
Pour cela, on va utiliser la connexion série usb qui relie la micro:bit à l'ordinateur.

Nous allons donc créer deux programmes :

- Un programme qui tourne sur le PC avec une interface graphique qui envoie (écrit) les instructions sur la liaison série usb à la mico:bit
- Un second programme qui tourne sur la micro:bit qui lit les instruction sur la liaison série usb et qui les exécute.



L'interface est la suivante :
On crée une boite de dialogue pour enter les coordonnées d'un pixel à allumer.

Le programme prog_micro.py

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

- **uart(baudrate=9600)** permet d'initialiser la vitesse de transmission à 9600 bauds.
- **uart.any()** permet de savoir s'il y a un message à lire.
- **uart.read()** permet de lire le message envoyé sur la liaison série usb.
  Attention, les messages transmis doivent être en bytes, c'est pourquoi, ils sont convertis en chaîne de caractères puis en entiers.
- Ensuite : si le message envoyé est 11, on allume la led à la position 1,1.

A vous de créer le programme :

- Ouvrir Thonny puis créer un programme nommé prog_micro.py
- Recopier le programme suivant et l'enregistrer dans votre dossier MicroBit, puis téléverser le dans la micro:bit.

```
from microbit import *

uart.init(baudrate=9600)

while True :
    if uart.any():
        msg_bytes=uart.read()
        msg = str(msg_bytes, "utf-8")
        x=int(msg[0])
        y=int(msg[1])
        display.set_pixel(x,y,9)
        sleep(1000)
        display.clear()
```

Le programme prog_pc.py

Récupérer le fichier prog_pc.py que l'on donne ci-dessous puis testez-le.

- **serie = Serial(port)** permet d'ouvrir une connexion série sur le port de l'ordinateu.
- **serie.baudrate** permet de fixer la vitesse de transmission à la même valeur que celle choisie sur la carte. Comme sur la carte les messages transmis sont nécessairement en bytes, ce qui nécessite une conversion entre chaîne de caractères (str) et bytes avant transmission ou après réception.

```
from tkinter import *

#port série à déterminer
port = "/dev/cu.usbmodem1422"
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)
```

La suite du programme est la réalisation de l'interface graphique. On utilise la bibliothèque Tkinter.

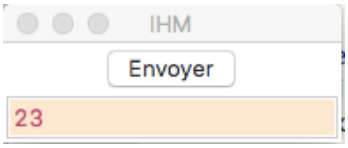
```
# Fenêtre principale
ma_fenetre = Tk()
ma_fenetre.title("IHM")

# 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")
# Positionnement du champ
champ_message.pack()

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

Exécuter le programme, lorsque vous entrer les valeurs 23, la diode à la position 2,3 devrait s'allumer.



Exercice 7 :

Modifier les deux programmes précédents sous les noms prog2_micro.py et prog2_pc.py afin de créer une interface graphique avec deux boutons « allumer » et « éteindre » qui permettent d'allumer et éteindre la LED centrale.