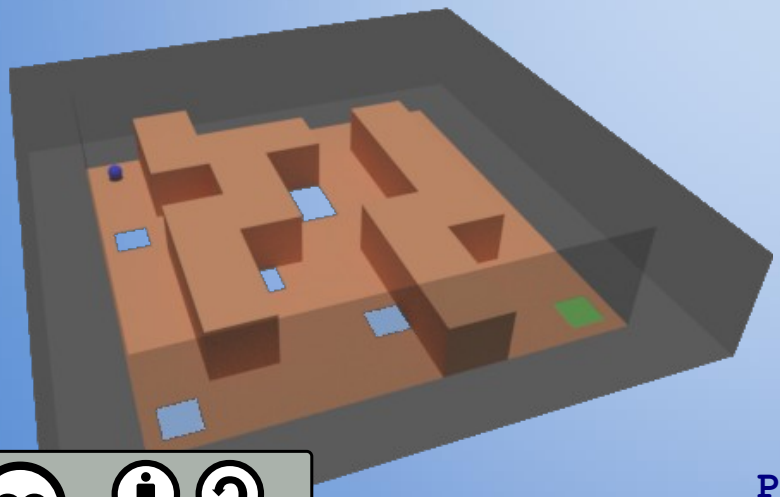
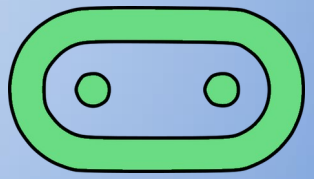
Labyrinthe à bille
Créer une scène 3D interactive

Tutoriel 5 :
Interfacer avec micro:bit





Philippe Roy <philippe.roy@ac-grenoble.fr>
https://forge.aeif.fr/blender-edutech/blender-edutech-tuto

Objectif

L'objectif de ce tutoriel est faire interagir les objets de la scène 3D (des objets virtuelles) à partir d'actions physiques mesurées par des capteurs. Les **capteurs** sont ici intégrés (onboard) ou reliés à une **carte micro:bit**. La carte communique avec l'ordinateur par une **liaison série**. La guidance de ce tutoriel a pour pré-requis la réalisation des deux tutoriels précédents (Tutoriel 1 : Ma première scène, Tutoriel 2 : Passage au Python).

Le tutoriel se décompose en 6 étapes :

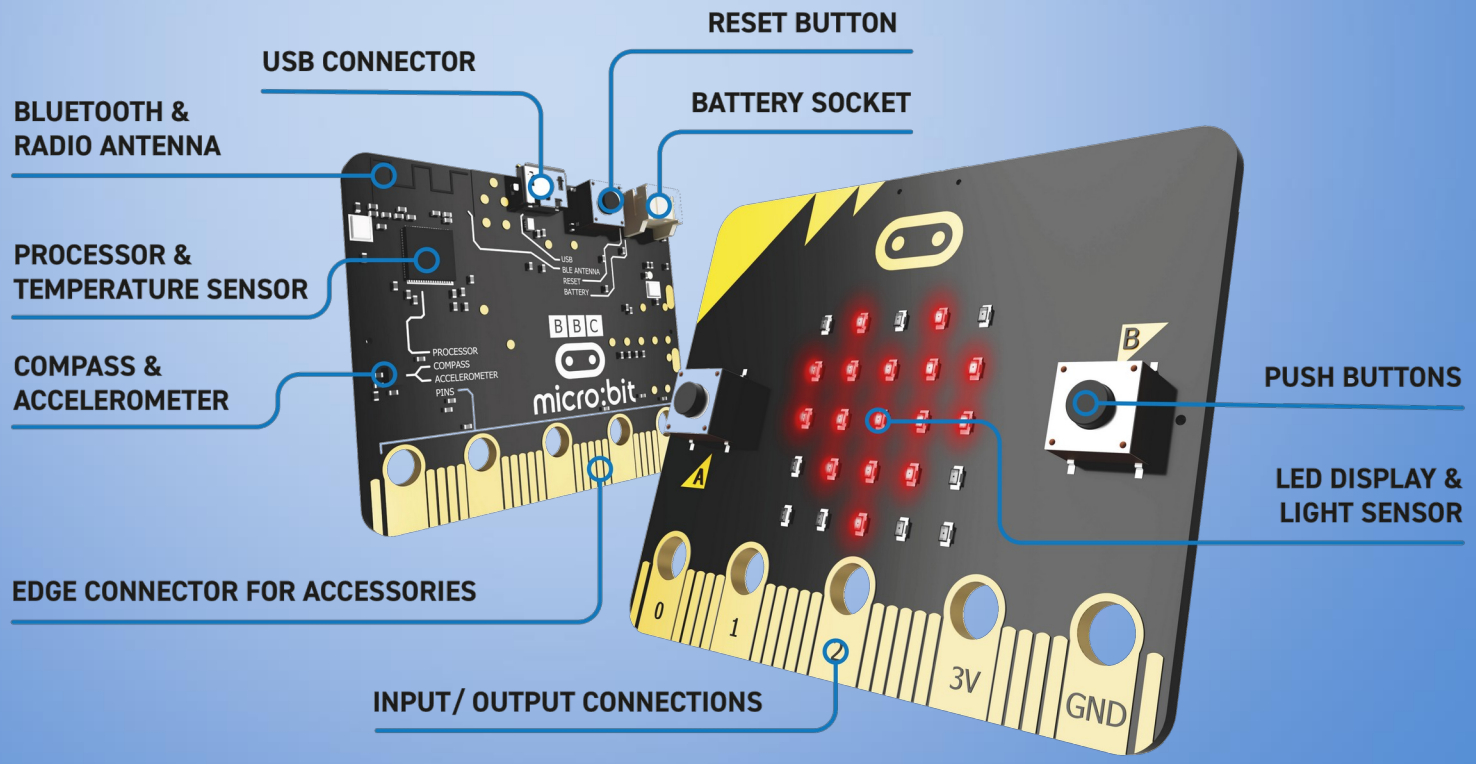
- 1. Programmer la carte micro:bit
- 2a. Installation de la bibliothèque pySerial sous GNU/Linux
- 2b. Installation de la bibliothèque pySerial sous Windows
- 3. Déplacer le plateau avec la centrale inertielle
- 4. Afficher la position de la bille sur la matrice de leds
- 5. Détecter automatiquement le micro-contrôleur
- 6. Inclure pySerial dans la distribution de l'éxécutable

1. Programmer la carte micro:bit

La carte **BBC micro:bit** est un microcontrôleur qui a plusieurs avantages :

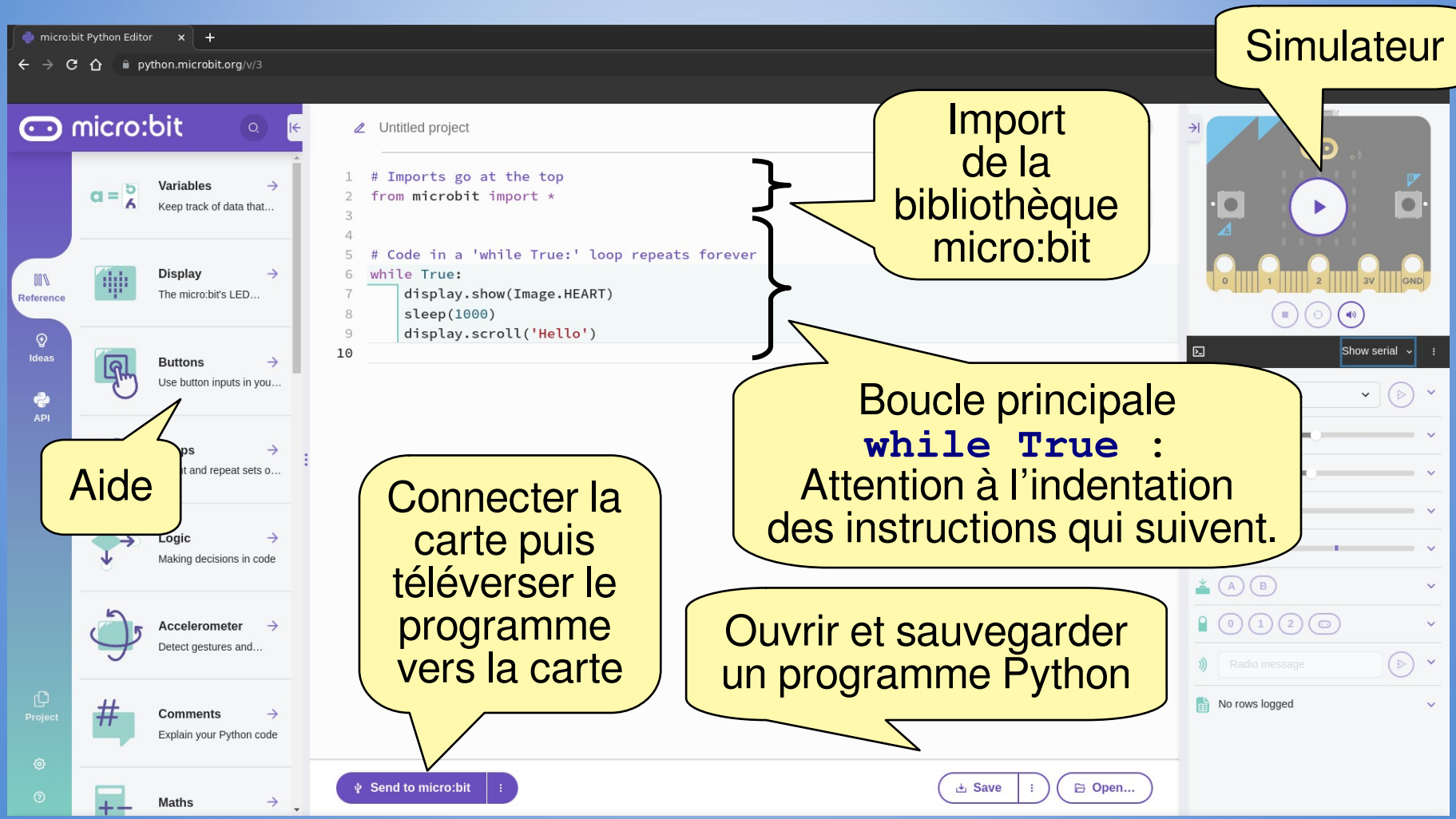
- il est open source (**https://github.com/microbit-foundation**),
- il se programme avec le langage Python,
- il est accompagné avec beaucoup de composants intégrés (onboard).

Comparativement avec les cartes Aduino, elle a moins de possibilités mais sa mise en œuvre est plus facile et plus rapide.



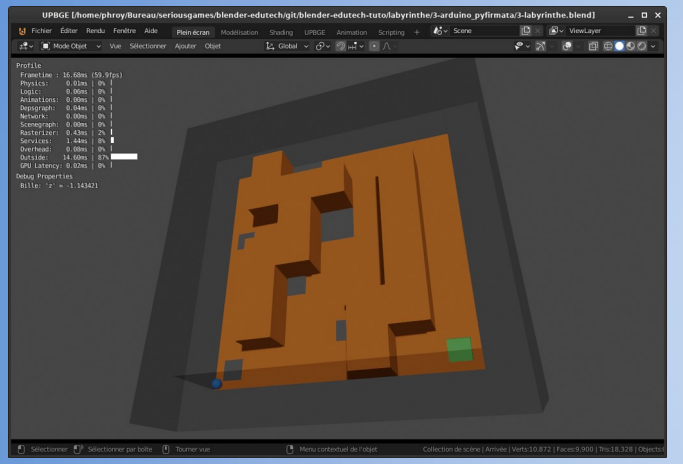
1. Programmer la carte micro:bit

L'édition du programme Python va se faire avec l'éditeur en ligne du site : **https://python.microbit.org/** avec le navigateur **Chromium** (open source) ou **Chrome** (Firefox n'est pas compatible avec WebUSB).



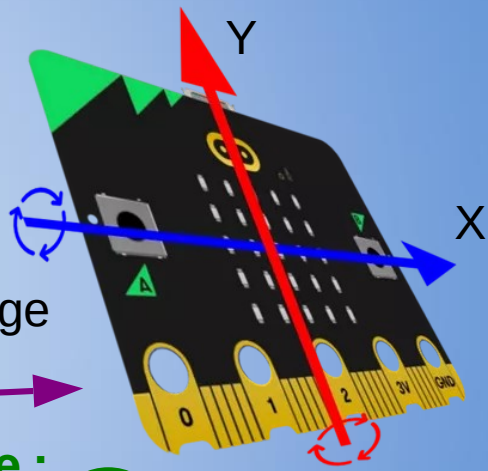
2. Programmer la carte Arduino avec la centrale inertielle (capteur IMU)

Nous allons utiliser la **liaison série** pour échanger des chaînes de caractère (message texte) entre la carte micro:bit et le module Python.



Format du message texte : ‘’roulis,tangage‘‘

Liaison série (USB)



Programme UPBGE : 5-labyrinthe.py

Programme carte : 5-labyrinthe-microbit.py

Le **programme UPBGE** va être à l’écoute du port série et lire les messages. Il appliquera les angles lues à l’inclinaison du plateau.

Le **programme de la carte** va acquérir les angles de roulis et de tangage du capteur et générer le message sur le port série.

1. Programmer la carte micro:bit

Le programme carte est le fichier « **5-labyrinthe-microbit.py** ».

```
from microbit import uart, sleep
from microbit import *

##############################################################################
# 5-labyrinthe-microbit.py
##############################################################################

##############################################################################
# Initialisation
##############################################################################

attente_image = Image("00000:00000:00300:00000:00000")
display.show(attente_image)  # Témoin de fonctionnement

uart.init(baudrate = 115200) # Initialisation du port série

##############################################################################
# Boucle principale
##############################################################################

while True:
    accel_x = accelerometer.get_x() # Roulis
    accel_y = accelerometer.get_y() # Tangage
    uart.write(str(accel_x)+","+str(accel_y)+"\n")
```

1 : Importer les bibliothèques

2 : Initialisation

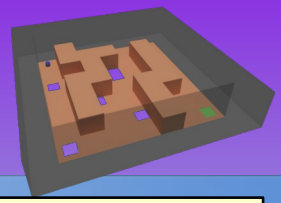
- Affiche le témoin de fonctionnement
- Ouvrir le port série

5-labyrinthe-microbit.py

3 : Boucle principale

- Lire les angles de roulis (axe x) et de tangage (axe y)
- générer le message texte pour
- le port série avec les deux angles

1. Programmer la carte micro:bit

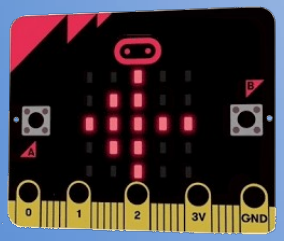


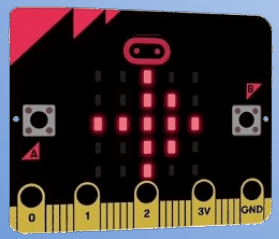
```
# Affichage de la l'inclinaison
if accel_x < -30: # Ouest
    if accel_y <-30:
        display.show(Image.ARROW_NW)
    elif accel_y >30:
        display.show(Image.ARROW_SW)
    else:
        display.show(Image.ARROW_W)
elif accel_x > 30: # Est
    if accel_y <-30:
        display.show(Image.ARROW_NE)
    elif accel_y >30:
        display.show(Image.ARROW_SE)
    else:
        display.show(Image.ARROW_E)
else: # Nord ou Sud
    if accel_y < -30 :
        display.show(Image.ARROW_N)
    elif accel_y > 30 :
        display.show(Image.ARROW_S)
    else: # Au centre
        display.show(attente_image)
sleep(100)
```

5-labyrinthe-microbit.py

4 : Boucle principale suite

- Afficher sur la matrice de leds de la carte les flèches nord, sud, est et ouest pour indiquer inclinaison de la carte





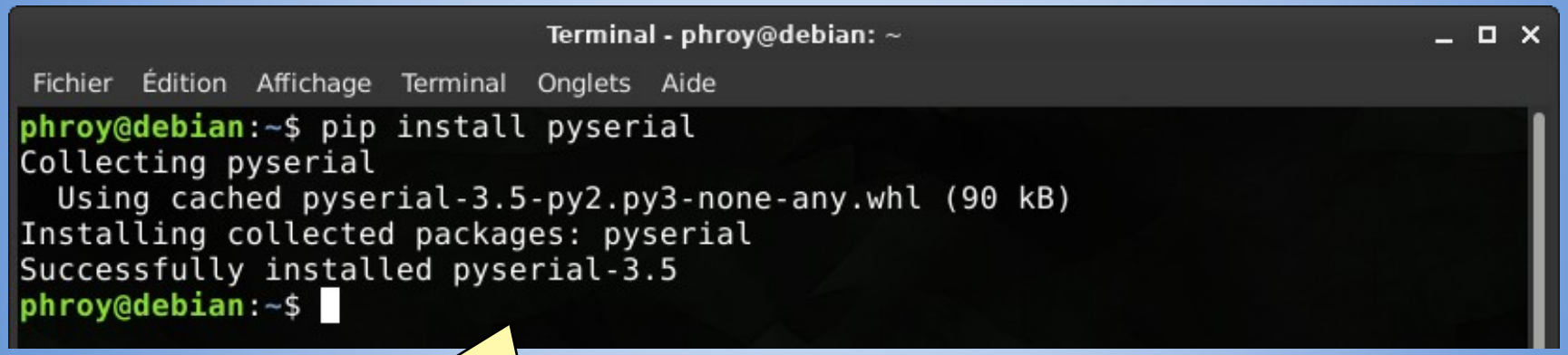
5 : Téléverser le programme vers la carte

6 : Tester la carte seule

2a. Installation de la bibliothèque pySerial sous GNU/Linux

La bibliothèque **pySerial** permet d'utiliser la **liaison série** dans un programme Python. Il faut donc installer la bibliothèque **pySerial**.

Généralement l'installateur de **bibliothèques Python** `pip` est déjà installé, sinon il faut utiliser le gestionnaire de paquet de la distribution pour l'installer.



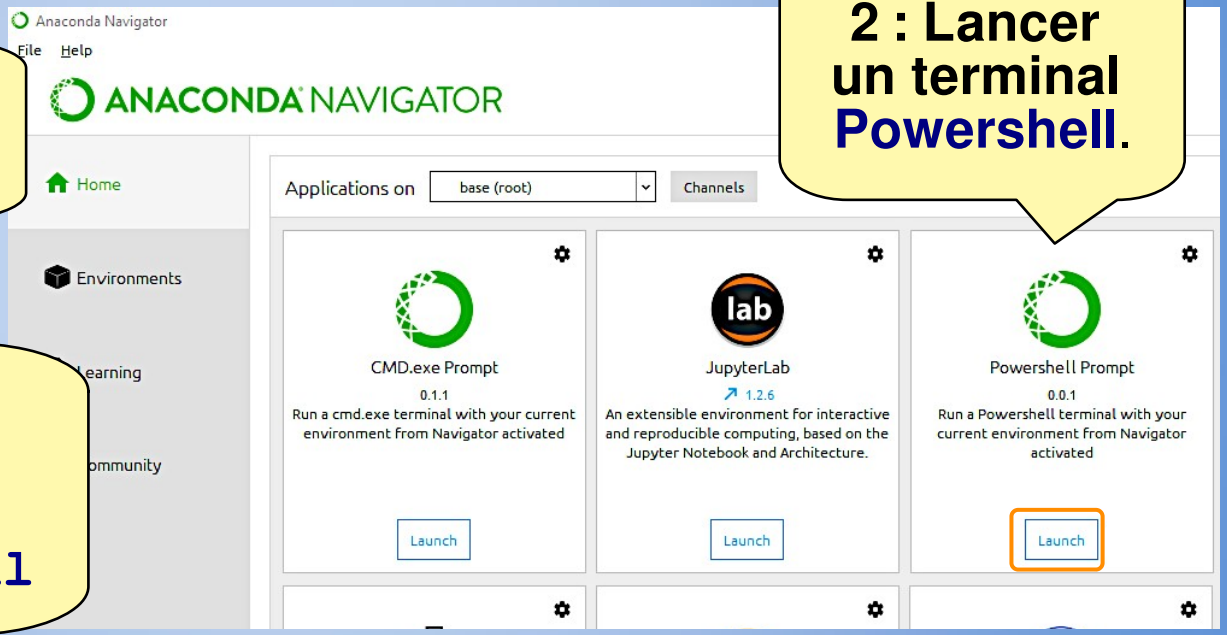
1 : Installer pySerial
Dans un **terminal**
exécuter la commande :
`pip install pyserial`

Si la bibliothèque pySerial n'est pas reconnue par UPBGE, il faut alors l'installer dans l'environnement local de UPBGE, pour cela voir l'**étape 6**.

2b. Installation de la bibliothèque pySerial sous Windows

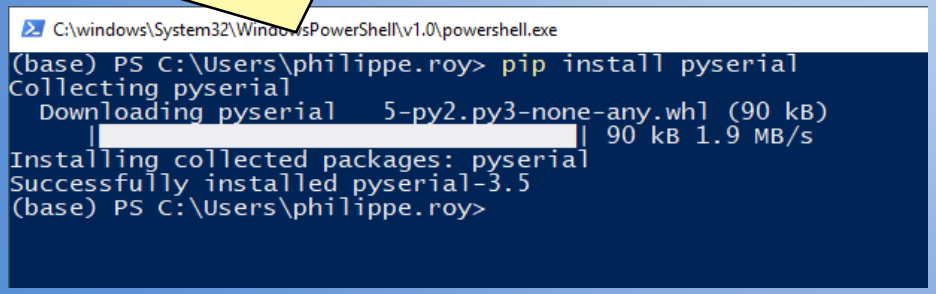
La bibliothèque **pySerial** permet d'utiliser la **liaison série** dans un programme Python. Il faut donc installer la bibliothèque **pySerial**. Sous Windows, l'installation de la distribution **Anaconda** est une solution simple et efficace de mettre en place un **environnement Python**.

1 : Lancer le programme Anaconda navigator



2 : Lancer un terminal Powershell.

3 : Installer pySerial
Dans le **terminal**
exécuter la commande :
`pip install pyserial`

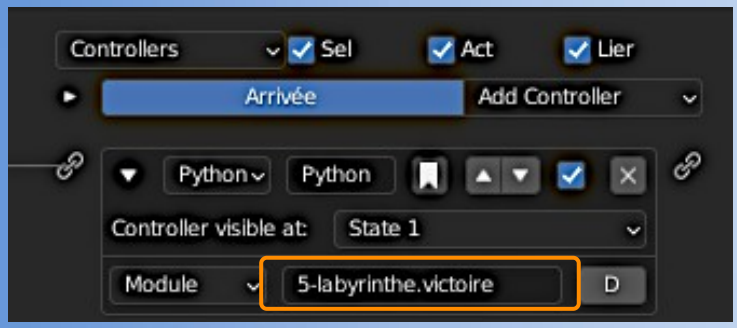


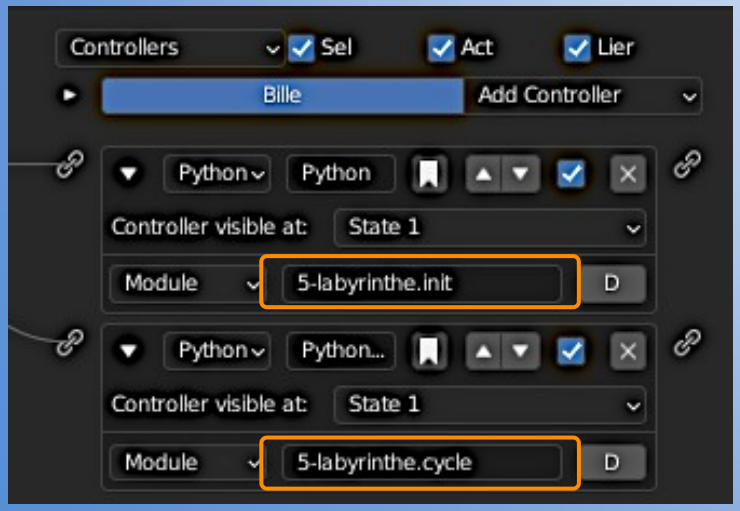
Si la bibliothèque pySerial n'est pas reconnue par UPBGE, il faut alors l'installer dans l'environnement local de UPBGE, pour cela voir l'**étape 6**.

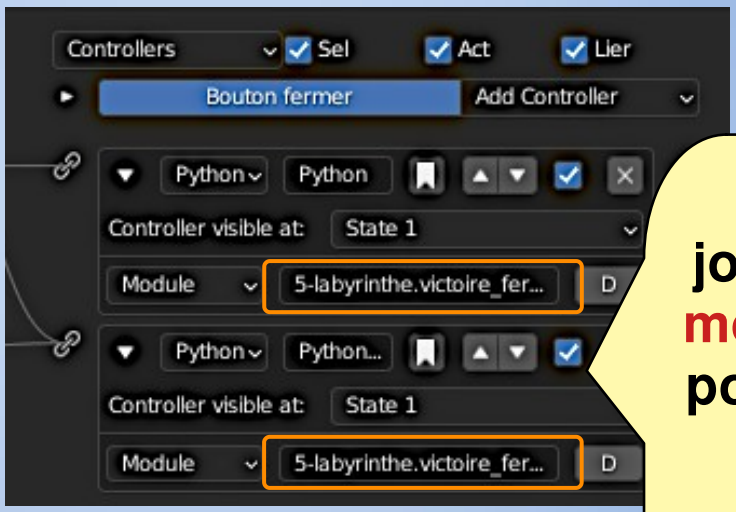
3. Déplacer le plateau avec la centrale inertielle

Les fichiers de départ de ce tutoriel sont les fichiers résultats du tutoriel 2. Il faut donc :

- copier et renommer « **2-labyrinthe.blend** » en « **5-labyrinthe.blend** »,
- copier et renommer « **2-labyrinthe.py**» en « **5-labyrinthe.py** ».



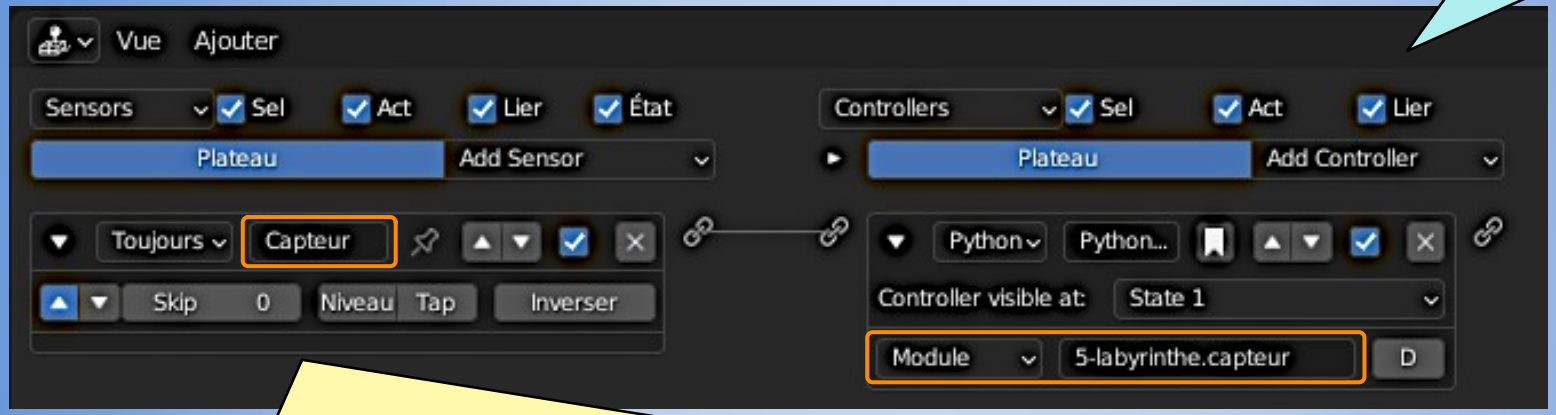




1 : Mettre à jour le nom des modules Python pour l'ensemble des briques logiques
Renommer les noms « **2-labyrinthe.*** » en « **5-labyrinthe.*** »

3. Déplacer le plateau avec la centrale inertielle

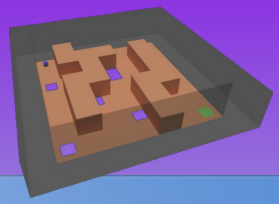
Briques logiques de **Plateau**



2 : Mettre UPBGE à l'écoute du capteur IMU

- **Renommer** le **Capteur Toujours** avec **Capteur** (ancien nom : « Clavier »)
- **renommer** le **Module Python** avec **5-labyrinthe.capteur** (ancien nom : « 2-labyrinthe.clavier »)

3. Déplacer le plateau avec la centrale inertielle



3 : Ajout l'importation de la bibliothèque

```
import serial
```

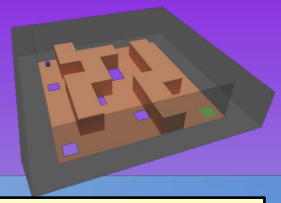
5-labyrinthe.py

```
import bge # Bibliothèque Blender Game Engi
import serial # Liaison série

##############################################################################
# 5-labyrinthe.py
##############################################################################

# Récupérer la scène 3D
scene = bge.logic.getCurrentScene()

# Constantes
JUST_ACTIVATED = bge.logic.KX_INPUT_JUST_ACTIVATED
JUST_RELEASED = bge.logic.KX_INPUT_JUST_RELEASED
ACTIVATE = bge.logic.KX_INPUT_ACTIVE

# Communication avec la carte Arduino
serial_baud = 115200
# serial_comm = serial.Serial('COM4',serial_baud, timeout=0.016) # Windows
serial_comm = serial.Serial('/dev/ttyACM0',serial_baud, timeout=0.016) # GNU/Linux
print (serial_comm)
```

4 : Créer la communication par la liaison série

- **Port** : **COM4**, **/dev/ttyACM0**, ...
- **Vitesse** : **115200** bauds
- **Timeout** : **0.016** s , c'est le **temps de cycle** de **UPBGE** (**60 fps**)

4. Déplacer le plateau avec la centrale inertielle



```
################################################################################
# Gestion de la centrale inertielle de la carte micro:bit
# Les valeurs du capteur sont transmises de 0 à 1024 (10 bits) où 1024 -> 90°.
# Les valeurs de l'inclinaison du plateau dans UPBGE sont à fournir en radian.
################################################################################

# Lecture du capteur IMU
def capteur(cont):
    obj = cont.owner # obj est l'objet associé au contrôleur donc 'Plateau'
    obj_bille = scene.objects['Bille']
    echelle = 0.2 # Facteur d'échelle entre la capteur et la scène 3D
    ecart=0.001 # Écart maxi sur la rotation

    # Touche ESC -> Quitter
    keyboard = bge.logic.keyboard
    if keyboard.inputs[bge.events.ESCKEY].status[0] == ACTIVATE:
        serial_comm.close()
        bge.logic.endGame()

    # Lecture de la liaison série : programme micro:bit : 5-labyrinthe-microbit.py
    serial_msg_in = str(serial_comm.readline())

    # Roulis/Roll(x) et Tangage/Pitch(y)
    if serial_msg_in.find(",") > 0:
        txt = serial_msg_in.split(',',2)
        x_txt = txt[0][2:]
        y_txt = txt[1][:-3]
        if x_txt != "" and y_txt != "": # Absence de valeur
            x=-(float(x_txt)/651.8)*echelle # 651.8 = 1024 / (pi/2)
            y=-(float(y_txt)/651.8)*echelle # 651.8 = 1024 / (pi/2)
            while abs(x-obj.worldOrientation.to_euler().x) > ecart :
                obj.applyRotation((x-obj.worldOrientation.to_euler().x, 0,
                                   -obj.worldOrientation.to_euler().z), False)
            while abs(y-obj.worldOrientation.to_euler().y) > ecart :
                obj.applyRotation((0, y-obj.worldOrientation.to_euler().y,
                                   -obj.worldOrientation.to_euler().z), False)
```

5 : Créer la fonction capteur

5-labyrinthe.py

4. Déplacer le plateau avec la centrale inertielle

```
##############################################################################
# Gestion du clavier
##############################################################################

# Flèches pour tourner le plateau sur les axes Y et Y avec une mise à 0 sur l'axe Z
def clavier(cont):
    obj = cont.owner # obj est l'objet associé au contrôleur donc 'Plateau'
    keyboard = bge.logic.keyboard
    resolution = 0.01

    # Flèche haut - Up arrow
    if keyboard.inputs[bge.events.UPARROWKEY].status[0] == ACTIVATE:
        obj.applyRotation((-resolution,0,-obj.worldOrientation.to_euler().z), False)

    # Flèche bas - Down arrow
    if keyboard.inputs[bge.events.DOWNARROWKEY].status[0] == ACTIVATE:
        obj.applyRotation((resolution,0,-obj.worldOrientation.to_euler().z), False)

    # Flèche gauche - Left arrow
    if keyboard.inputs[bge.events.LEFTARROWKEY].status[0] == ACTIVATE:
        obj.applyRotation((0, -resolution,-obj.worldOrientation.to_euler().z), False)

    # Flèche droit - Right arrow
    if keyboard.inputs[bge.events.RIGHTARROWKEY].status[0] == ACTIVATE:
        obj.applyRotation((0, resolution,-obj.worldOrientation.to_euler().z), False)
```

5-labyrinthe.py

6 : Supprimer la fonction clavier

7 : Tester le capteur [P]

```
Terminal - phroy@debian: /mnt/fe77fe04-4eb8-43
Fichier  Édition  Affichage  Terminal  Onglets  Aide
Blender Game Engine Started
Serial<id=0x7fca98354b50, open=True>(port='/dev/tty
timeout=0.016, xonxoff=False, rtscts=False, dsrdtr=False)
Chuuuu.....te
Blender Game Engine Finished
Debug: 1920, 1008
rcti: : xmin 0, xmax 1919, ymin 20, ymax 1027 (1919x1007)

Blender Game Engine Started
Serial<id=0x7fca982b4910, open=True>(port='/dev/ttyACM1', baudrate=115200, bytesize=8, parity='N', stopbits=1,
timeout=0.016, xonxoff=False, rtscts=False, dsrdtr=False)
Chuuuu.....te
Chuuuu.....te
^[Blender Game Engine Finished
```

4. Afficher la position de la bille sur la matrice de leds

La **matrice de leds** va nous permettre d'**afficher la position de la bille** avec les coordonnées en x et en y. Avec **une matrice 5x5**, la **valeur de x et de y** est de **0 à 4**. La liaison série va être utilisée dans **les deux sens** :

- Arduino → UPBGE : angle de roulis et de tangage (ça c'est déjà fait !),
- UPBGE → Arduino : position en x et en y.

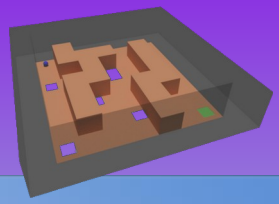
```
##############################################################################
# Boucle principale
##############################################################################

while True:
    accel_x = accelerometer.get_x() # Roulis
    accel_y = accelerometer.get_y() # Tangage
    uart.write(str(accel_x)+","+str(accel_y)+"\n")

    # UBGE -> micro:bit (lecture du message)
    msg_byte = uart.readline()
    if msg_byte:
        display.clear()
        msg_str = str(msg_byte, 'ascii')
        if "91" in msg_str: # Chute
            display.show(Image.SAD)
            sleep(500);
            uart.write("start\n")
        elif "92" in msg_str: # Victoire
            display.show(Image.HAPPY)
        else: # Position de la bille
            display.set_pixel(int(msg_str[0]), int(msg_str[1]), 9 )
```

5-labyrinthe-microbit.py

1 : Exécuter l'ordre associé au message entrant
(venant de UPBGE)

- Lire le message entrant
- si message = « 91» afficher un visage triste puis envoie « start »,
- si le message = « 92 » afficher un visage heureux,
- sinon afficher la position de la bille

4. Afficher la position de la bille sur la matrice de leds



5-labyrinthe-microbit.py

```
# Affichage de la l'inclinaison
if accel_x < -30: # Ouest
    if accel_y <-30:
        display.show(Image.ARROW_NW)
    elif accel_y >30:
        display.show(Image.ARROW_SW)
    else:
        display.show(Image.ARROW_W)
elif accel_x > 30: # Est
    if accel_y <-30:
        display.show(Image.ARROW_NE)
    elif accel_y >30:
        display.show(Image.ARROW_SE)
    else:
        display.show(Image.ARROW_E)
else: # Nord ou Sud
    if accel_y < -30 :
        display.show(Image.ARROW_N)
    elif accel_y > 30 :
        display.show(Image.ARROW_S)
    else: # Au centre
        display.show(attente_image)


sleep(100)
```

2 : Supprimer l'affichage des flèches sur la matrice des leds de la carte

3 : On garde le cadencement à 0,1 s

4. Afficher la position de la bille sur la matrice de leds

Au niveau du **programme Python**, il y a trois modifications :

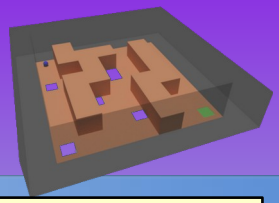
- **envoyer le message XY** (une valeur de 0 à 55) de la **position de la bille** par la liaison série,
- envoyer les messages « **91** » ou « **92** » en cas respectivement de **chute** ou de **victoire**,
- repositionner la bille au départ si nous recevons le message « **start** » de la carte Arduino.

5-labyrinthe.py

```
################################################################################
# Gestion de la centrale inertielle de la carte micro:bit
# Les valeurs du capteur sont transmises de 0 à 1024 (10 bits) où 1024 -> 90°.
# Les valeurs de l'inclinaison du plateau dans UPBGE sont à fournir en radian.
################################################################################

# Lecture du capteur IMU
def capteur(cont):
    obj = cont.owner # obj est l'objet associé au contrôleur donc 'Plateau'
    obj_bille = scene.objects['Bille']
    echelle = 0.2 # Facteur d'échelle entre la capteur et la scène 3D
    ecart=0.001 # Écart maxi sur la rotation

    # Touche ESC -> Quitter
    keyboard = bge.logic.keyboard
    if keyboard.inputs[bge.events.ESCKEY].status[0] == ACTIVATE:
        serial_comm.close()
        bge.logic.endGame()

    # Lecture de la liaison série : programme Arduino : 4-labyrinthe-imu.ino
    serial_msg_in = str(serial_comm.readline())

    # Mettre la bille à la position de départ avec une vitesse nulle
    if serial_msg_in.find("start") > 0:
        if obj_bille['victoire'] or obj_bille['chute']:
            depart()
    ...
```

4 : Remettre à la bille au départ si le message entrant est « start » (dans la fonction **capteur**)

5. Afficher la position de la bille sur la matrice de leds



5-labyrinthe.py

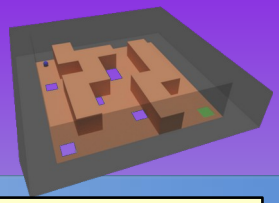
```
################################################################################
# Gameplay
################################################################################

...

# Cycle (boucle de contrôle de la bille)
def cycle(cont):
    obj = cont.owner # obj est l'objet associé au contrôleur donc 'Bille'
    obj['z']=obj.worldPosition.z # la propriété z est mis à jour avec la position globale en z de la bille

    # Écriture de la position de la bille sur la liaison série
    if obj['victoire'] == False and obj['chute'] == False:
        # obj['x'] = obj.worldPosition.x # de -3.5 à 3.5
        # obj['y'] = obj.worldPosition.y # de 3.5 à -3.5
        obj['Lx']=round((obj.worldPosition.x+3.5)*(4/7))  # de 0 à 4
        if obj['Lx']<0: obj['Lx']=0
        if obj['Lx']>4: obj['Lx']=4
        obj['Ly']=round((-obj.worldPosition.y+3.5)*(4/7))  # de 0 à 4
        if obj['Ly']<0: obj['Ly']=0
        if obj['Ly']>4:  obj['Ly']=4
        serial_msg_out = str(obj['Lx'])+str(obj['Ly'])+"\n"
        serial_comm.write(serial_msg_out.encode())

    # Si l'altitude de bille < -10 et pas de victoire -> chute
    if obj['z'] < -10 and obj['victoire'] == False:
        print ("Chuuuu.....te")
        depart() # Replacer la bille au départ

        # Afficher image de chute sur la matrice de leds
        serial_msg_out = "91\n"
        serial_comm.write(serial_msg_out.encode())
        obj['chute'] = True
```

5 : Envoyer le message texte des coordonnées en x et y de la bille (dans la fonction **cycle**)

6 : Envoyer le message « 91 » lors de la chute et supprimer les 2 lignes `print` et `depart()` (toujours dans la fonction **cycle**)

Le départ de la bille est maintenant provoqué par le message **« start »**.

5. Afficher la position de la bille sur la matrice de leds



```
##############################################################################
# Gameplay
##############################################################################

...

# Victoire (collision de la bille avec l'arrivée)
def victoire(cont):

    # Afficher image de victoire sur la matrice de leds
    serial_msg_out = "92\n"
    serial_comm.write(serial_msg_out.encode())
    scene.objects['Bille']['victoire'] = True

    # Animation du Panneau victoire
    scene.objects['Panneau victoire'].setVisible(True, True) # Afficher le panneau de la victoire
    scene.objects['Panneau victoire - plan'].restorePhysics() # Restaurer la physique du panneau cliquable
    start = 1
    end = 100
    layer = 0
    priority = 1
    blendin = 1.0
    mode = bge.logic.KX_ACTION_MODE_PLAY
    layerWeight = 0.0
    ipoFlags = 0
    speed = 1
    scene.objects['Panneau victoire'].playAction('Panneau victoireAction', start, end, layer,
                                                 priority, blendin, mode, layerWeight, ipoFlags, speed)
```

7 : Envoyer le message « 92 » lors de la victoire (dans la fonction **victoire**)

5-labyrinthe.py

8 : Tester la matrice de leds [P]

5. Détecter automatiquement le micro-contrôleur

Au début du programme il faut saisir le port sur lequel est branché la carte. Par exemple si le port est « COM4 » le code est : `serial_comm = serial.Serial('COM4',serial_baud,timeout=0.016)`. Or le port change souvent et afin d'éviter de retoucher le code on souhaite détecter automatiquement le port. Nous allons créer un module Python uniquement pour la détection du port : « **labyrinthe_carte.py** ».

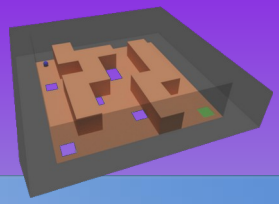
1 : Créer le fichier labyrinthe_carte.py

2 : Importation des bibliothèques

3 : Fonction de détection de la carte

```
import serial # Liaison série
from serial.tools.list_ports import comports # Détection du port automatique

##############################################################################
# labyrinthe_carte.py
##############################################################################

# Recherche automatique du port (microbit, Arduino Uno et Arduino Mega)
def autoget_port():
    # USB Vendor ID,  USB Product ID
    carte_dict={'microbit' :[3368, 516],
                'uno' :[9025, 67],
                'mega' :[9025, 66]}
    for com in comports(): # micro:bit
        if com.vid == carte_dict['microbit'][0] and com.pid == carte_dict['microbit'][1]:
            return [com.device,"micro:bit"]
    for com in comports(): # Arduino Uno
        if com.vid == carte_dict['uno'][0] and com.pid == carte_dict['uno'][1]:
            return [com.device,"Arduino Uno"]
    for com in comports(): # Arduino Mega
        if com.vid == carte_dict['mega'][0] and com.pid == carte_dict['mega'][1]:
            return [com.device,"Arduino Mega"]
    return [None,""]
```

5. Détecter automatiquement le micro-contrôleur



labyrinthe_carte.py

```
# Établir la communication avec la carte par la liaison série avec une vitesse
def init_serial(speed=115200):
    [port, carte_name] = autoget_port()
    if port is None:
        print("Carte Arduino/microbit introuvable")
        return None
    else:
        serial_comm = serial.Serial(port, speed, timeout=0.016)
        if serial_comm is None:
            print("Communication avec Carte Arduino/microbit impossible")
            return None
        else:
            print("Communication avec Carte Arduino/microbit établie sur "+port+" à la vitesse "
                  +str(speed)+" bauds")
            return serial_comm
```

4 : Fonction d'initialisation de la communication avec la carte par la liaison série

6. Détecter automatiquement le micro-contrôleur

Nous allons maintenant utiliser la fonction `init_serial()` dans **5-labyrinthe.py**.

```
import bge # Bibliothèque Blender Game Engine (BGE)
import pserial # Liaison série
import labyrinthe_carte # Liaison avec la carte

################################################################################
# 4-labyrinthe.py
################################################################################

# Récupérer la scène 3D
scene = bge.logic.getCurrentScene()

# Constantes
JUST_ACTIVATED = bge.logic.KX_INPUT_JUST_ACTIVATED
JUST_RELEASED = bge.logic.KX_INPUT_JUST_RELEASED
ACTIVATE = bge.logic.KX_INPUT_ACTIVE

# Communication avec la carte Arduino
serial_baud = 115200
# serial_comm = serial.Serial('COM4',serial_baud, timeout=0.016) # Windows
serial_comm = serial.Serial('/dev/ttyACM1',serial_baud, timeout=0.016) # GNU/Linux
print (serial_comm)

# Détection de la carte avec la liaison série
serial_comm = labyrinthe_carte.init_serial()
if serial_comm is None:
    bge.logic.endGame()
```

5 : Ajout l'importation de notre module `import labyrinthe_carte`

6 : Supprimer la création manuelle de l'objet serial_comm

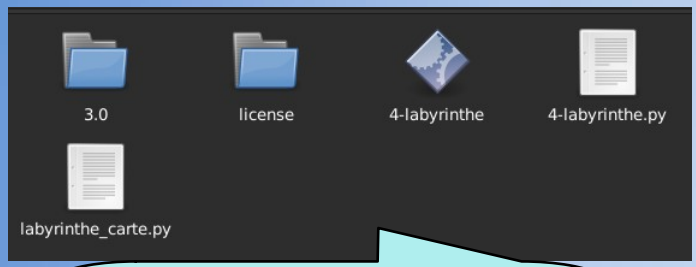
7 : Ajouter la création automatique de l'objet serial_comm

5-labyrinthe.py

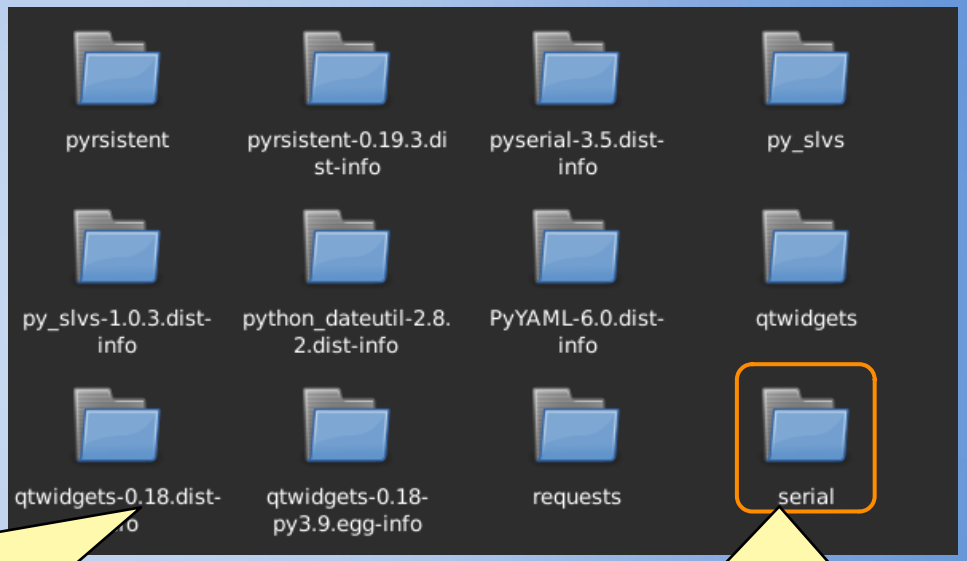
8 : Tester le détection automatique de la carte [P]

6. Inclure pySerial dans la distribution de l'exécutable

Pour pouvoir faire fonctionner l'**exécutable (game runtime)**, il faut que la bibliothèque **pySerial** soit présente dans l'**environnement local de l'exécutable**.



Répertoire de l'exécutable



1 : Aller dans le répertoire contenant les bibliothèques de l'environnement local de l'exécutable

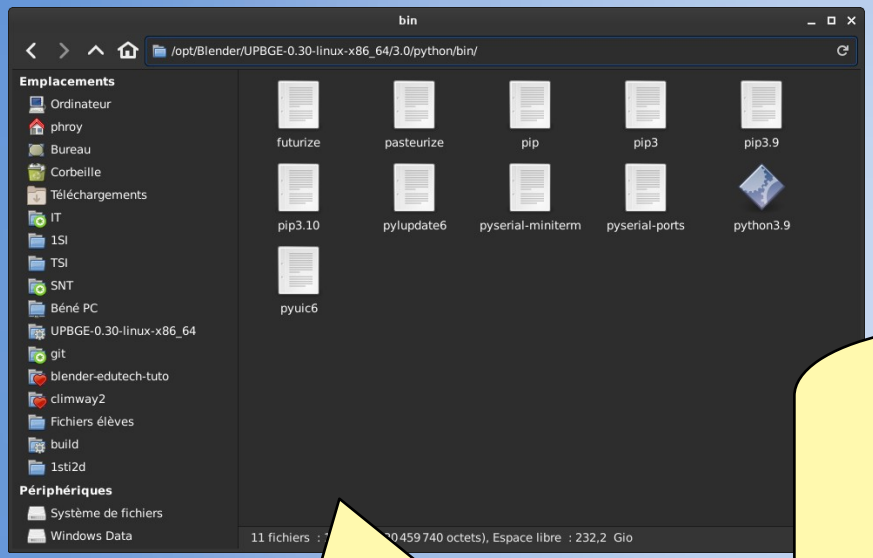
Chemin à partir du répertoire de l'exécutable :
3.0/python/lib/python3.9/site-packages/

2 : Vérifier la présence du répertoire serial.

- **Si c'est le cas, c'est bon cela devrait fonctionner**
- **Sinon voir la suite**

6. Inclure pySerial dans la distribution de l'exécutable

Si la bibliothèque **serial** n'est pas présente, il faut l'installer dans l'**environnement Python de UPBGE**.



4 : Télécharger le script d'installation du gestionnaire de paquet pip `get-pip.py` et copier le dans le répertoire des binaires Python de UPBGE

5 : Toujours dans le répertoire des binaires Python de UPBGE, ouvrir une console et installer le gestionnaire de paquet Pip :

GNU/Linux : `$ ./python3.9 get-pip.py`

Windows : `python.exe get-pip.py`

3 : Aller dans le répertoire des binaires Python de UPBGE

GNU/linux : UPBGE-0.30-linux-x86_64/3.0/python/bin/

Windows : UPBGE-0.30-windows-x86_64\3.0\python\bin

6. Inclure pySerial dans la distribution de l'exécutable

6 : Installer la bibliothèque pySerial

GNU/Linux :

- **avec la console ouverte**
- **toujours dans le répertoire des binaires Python de UPBGE,**
- `$ ./pip install pyserial`

Windows :

- **avec la console ouverte**
- **aller dans le répertoire 'Scripts' :**
  `cd C:\foo\UPBGE-0.30-windows-x86_64\3.0\python\Scripts`
- `pip.exe install pyserial`

7 : Générer un nouveau exécutable (game runtime), la bibliothèque dit être maintenant incluse dans l'environnement local de l'exécutable.

Bravo ! Vous êtes arrivé à l'issue de ce tutoriel.