Nous allons utiliser nos connaissances en langage `python` afin de programmer le robot Mindstorms EV3 de LEGO®. Pour cela nous utiliserons une bibliothèque libre appelée `ev3dev` [http://www.ev3dev.org/]. Seule une partie des possibilités du robot seront présentées.

Notre but sera de faire effectuer au robot un parcours et qu'il s'arrête momentannément si un obstacle est rencontré.

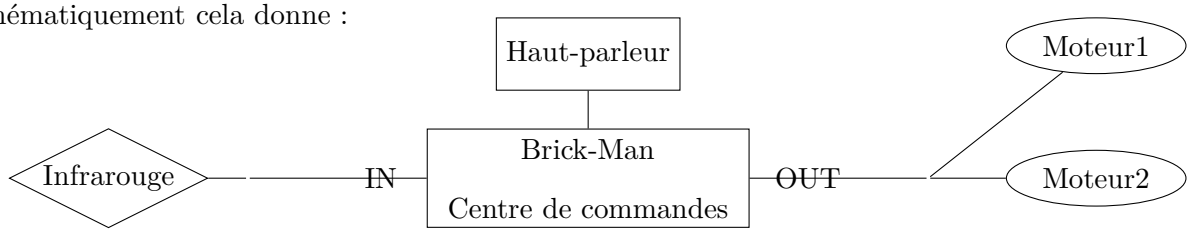
⚠ afin d'éviter les chutes, le robot roulera sur le sol. ⚠

I Le robot et ses périphéries

Le robot est composé :

- ◇ d'un centre de commandes : le brick-man
- ◇ d'entrées : le capteur infra-rouge (qui permet de connaître la distance à un obstacle)
- ◇ de sorties : haut-parleur (intégré au brick-man), deux moteurs (reliés au brick-man par des câbles)

Schématiquement cela donne :



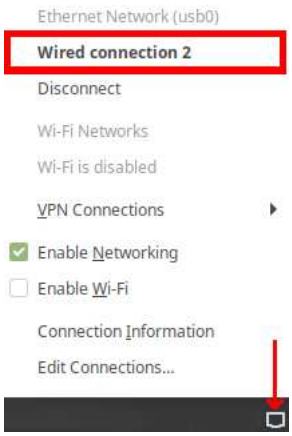
II Connexion au robot

Pour ce TP, nous devons travailler depuis un ordinateur sous Linux (un autre système d'exploitation que Windows). Pyzo est également installé sur ces machines.

1. commencer par créer un dossier à votre nom (appelé par la suite **MonDossier**) sur le bureau (`Desktop`). Vous y enregistrerez vos fichiers `Python` pour le robot. Lorsque vous voudrez tester un fichier sur le robot, il faudra l'y transférer. Le fichier `.py` sera transmis via le protocole ssh au travers d'un câble USB connecté au robot.

2. allumer le robot EV3 (touche du milieu), vérifier que dans `Wireless and Network/Tethering/Gadget` est coché.

3. le connecter au PC et vérifier cette connection (*cf.* ci-contre)



4. depuis un terminal, aller au répertoire contenant vos programmes (à faire une fois)
e.g. > `cd /home/mathsISN/Desktop/MonDossier`

5. rendre exécutable votre fichier .py (à faire une fois)
e.g. > `chmod 755 MonLogin-Robot.py`

6. vérifier que le fichier est éxécutable (le nom du fichier doit s'afficher en vert)
e.g. > `ls`

7. envoyer le fichier exécutable sur le robot
e.g. > `scp NomDuFichier-Robot.py robot@ev3dev.local:/home/robot/`

8. sur l'EV3 dans `File Browser`, lancer votre programme

Exo 1 : Télécharger le fichier ISN-chap06-Robot-01.py et le renommer `MonNom-Robot.py`
Faire les étapes nécessaires afin de le tester sur le robot EV3.

III Programmation du robot

Avancer PAR ÉTAPES (et pour chacune d'elles, sauvegarder le travail dans un fichier différent) :

- ◇ tester une seule nouvelle fonctionnalité à la fois
- ◇ une fois testée et validée, SAUVEGARDER une copie du fichier puis continuer à travailler.

Par exemple, nous avons validé 2 fonctionnalités, et travaillons sur une 3ème fonctionnalité sur le fichier `MonNom-Robot.py`. Une fois la 3ème fonctionnalité validée, on copie le fichier de travail `MonNom-Robot.py` dans `MonNom-Robot-03.py`.

Étudions le contenu du programme `ISN-chap06-Robot-01.py`.

```
#!/usr/bin/env python3
import time, random
import ev3dev.ev3 as ev3       # la librairie du brick-man

## definitions et initialisation
hp = ev3.Sound()               # haut parleur du brickman
md = ev3.LargeMotor('outA')    # moteur en sortie A appele md
md.stop_action = 'brake'       # pour que le moteur s'arrete en fin d'action

## programme principal
hp.speak('Ready').wait()
md.run_to_rel_pos(position_sp=360, speed_sp=900)
md.wait_while('running')       # attente tant que le moteur fonctionne
hp.speak('Finish').wait()
```

Le début de tout vos fichiers commenceront par les 3 premières lignes qui permettent d'importer des librairies indispensables au TP.

Chaque entrée / sortie se fait grâce à une variable qui doit être initialisée au fois au début du programme.

1 Haut-parleur

Le haut-parleur fonctionne à partir du brick-man sans connectique.

```
# definition et initialisation
hp = ev3.Sound()

# programme principal
hp.speak('Ready').wait()
```

On nomme `hp` l'élément haut-parleur du robot EV3. On appelle ensuite la fonction `speak()` du haut-parleur. Le texte énoncé est donné sous la forme d'une chaîne de caractères (en anglais). La mention facultative `wait()` oblige le robot à avoir fini de parler avant d'entreprendre une autre action.

Exo 2 : Après avoir sauvegardé votre fichier de travail, modifier les paroles du robot et tester votre programme.

2 Moteurs

Les moteurs doivent être connectés au brick-man et leur port de sortie doit être précisé lors de la définition. Les gros moteurs ont une vitesse absolue maximale de 1 050 et selon le sens de rotation, la vitesse peut être négative ou positive.

```
# definition et initialisation
md = ev3.LargeMotor('outA')    # defini un moteur en sortie 'outA'
md.stop_action = 'brake'       # pour que le moteur s'arrete en fin d'action

# programme principal
md.run_to_rel_pos(position_sp=360, speed_sp=900)
md.wait_while('running')       # attente tant que le moteur fonctionne
```

La ligne `md.run_to_rel_pos(position_sp=360, speed_sp=900)` signifie que le moteur `md` va effectuer une rotation de 360 degrés à la vitesse de 900.

Il existe d'autres fonctions permettant d'arrêter ou de faire tourner le moteur, en particulier :

```
md.stop()                                  # arret du moteur
md.run_timed(time_sp=3000, speed_sp=500)   # rotation temporelle (en ms)
md.run_forever(speed_sp=-500)              # rotation infinie
```

Exo 3 : Après avoir sauvegarder votre fichier, programmer le robot afin qu'il avance tout droit durant 3 secondes, s'arrête et tourne d'un quart de tour sur lui-même.

3 Capteur infra-rouge

Le capteur (ou senseur) infrarouge en mode `'IR-PROX'` permet de quantifier la proximité du senseur avec un objet en face (selon les essais, la détection maximale est d'environ 700 mm).

```
# definition et initialisation
irs = ev3.InfraredSensor('in2') # defini un senseur IR a l'entree 'in2'
irs.mode = 'IR-PROX'     # met le senseur en mode detection de distance

# programme principal
if irs.proximity < 25 :
    hp.speak('too close').wait()
```

`irs.proximity` renvoie un pourcentage de proximité (où 100% = 700mm).

Exo 4 : Après sauvegarde, programmer le robot afin qu'il avance jusqu'à ce qu'il soit à moins de 350mm d'un objet.

IV Application

Votre objectif est de concevoir d'un robot "tondeuse" ou "aspirateur". Ces robots programmés pour être actifs pendant une durée donnée et se déplacer en ligne droite jusqu'à rencontrer un obstacle. À ce moment, ils changent de direction de manière aléatoire puis continue tout droit.

Pour vous aider,

◇
```
choix = random.choice((90,270)) # choix aleatoire parmi 90 ou 270
```

La fonction `random.choice((a,b,c))` renvoie au hasard l'une des valeurs `a`, `b` ou `c`.

◇ le fichier ISN-chap06-Robot-02.py fait avancer le robot tout droit pendant 10 secondes.

```
#!/usr/bin/env python3
import time, random
import ev3dev.ev3 as ev3

## definitions et initialisation
hp = ev3.Sound()                    # haut parleur du brickman
md = ev3.LargeMotor('outA')         # moteur A
md.stop_action = 'brake'
mg = ev3.LargeMotor('outB')         # moteur B
mg.stop_action = 'brake'
irs = ev3.InfraredSensor()          # senseur infrarouge
irs.mode = 'IR-PROX'

## programme principal
hp.speak('Ready').wait()

t0 = time.time()                    # heure de depart
while time.time()-t0<10 :           # 10s d'utilisation
    hp.speak('Go').wait()
    md.run_to_rel_pos(position_sp=360, speed_sp=500)
    mg.run_to_rel_pos(position_sp=360, speed_sp=500)
    md.wait_while('running')

hp.speak('Finish').wait()
```

Exo 5 : faites un slalom simple et programmer votre robot afin qu'il le parcours et s'arrête en cas d'obstacle non-prévu.