

OBJECTIF DU TP

L'objectif du TP est d'une part de se familiariser avec les robots EV3 et leur API, et d'autre part de développer les éléments de base qui serviront pour les prochains TP.



CONSIGNES

Les robots EV3 utilisés dans ce TP sont fournis en partie montés, il est **déconseillé de les démonter ou de les modifier**. Vous pouvez commencer par effectuer les branchements suivants :

- Port C : moteur droit
- Port B : moteur gauche
- Port 2 : capteur de distance
- Port 3 : capteur couleur

Vous trouverez deux configurations différentes de robots, une avec **chenilles / capteur IR**, et une avec **roues classiques / capteur ultrason**.

Votre code devra être clair, organisé et documenté. Vous respecterez au maximum les conventions de programmation du langage utilisé.

EV3 ET MICROPYTHON

MICROPYTHON

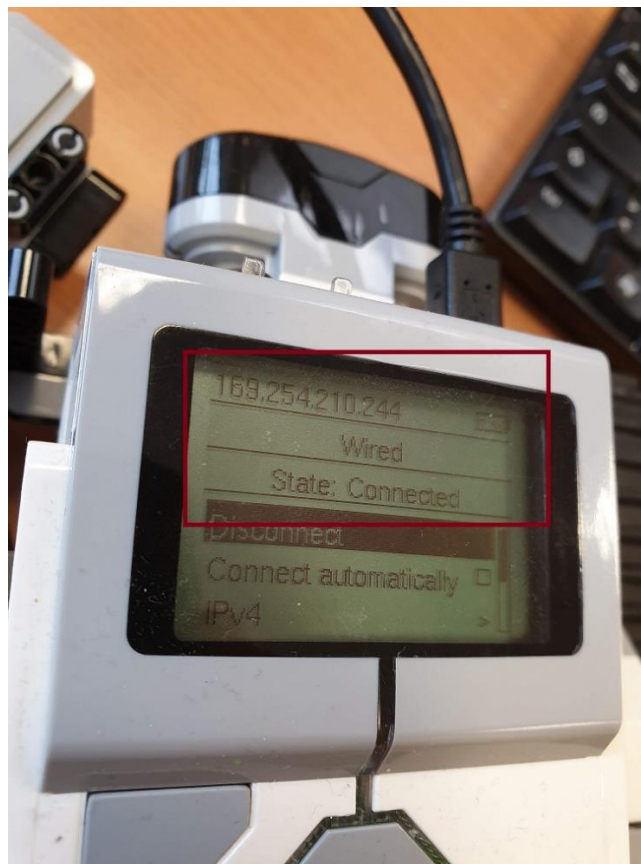
Les robots tournent sur un environnement **MicroPython** basé sur **ev3dev** et donnant accès à la librairie **Pybricks** exécutée sur python **3.4.0**. Attention, la dernière version de python étant **3.8.5**, certaines fonctionnalités accessibles sur votre ordinateur ne le seront pas sur les robots.

L'environnement MicroPython est déjà installé sur les cartes SD des robots, vous ne devez donc jamais flasher la carte SD.

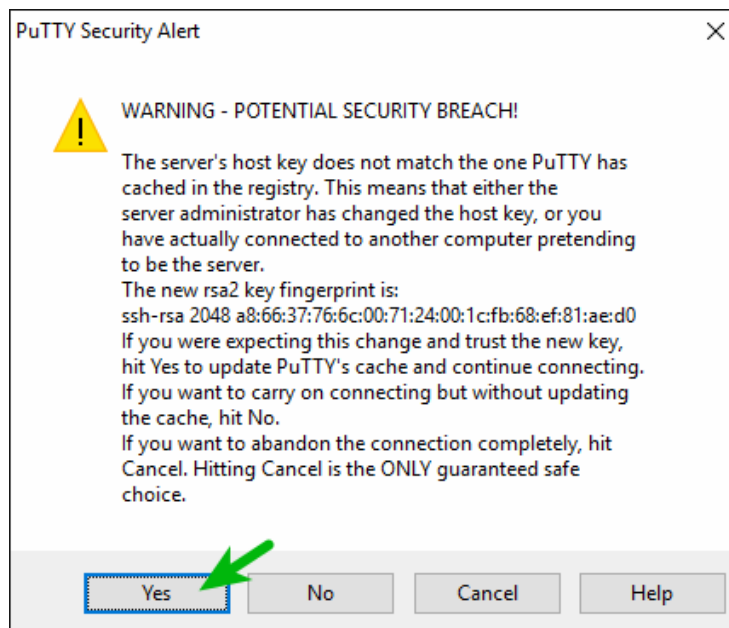
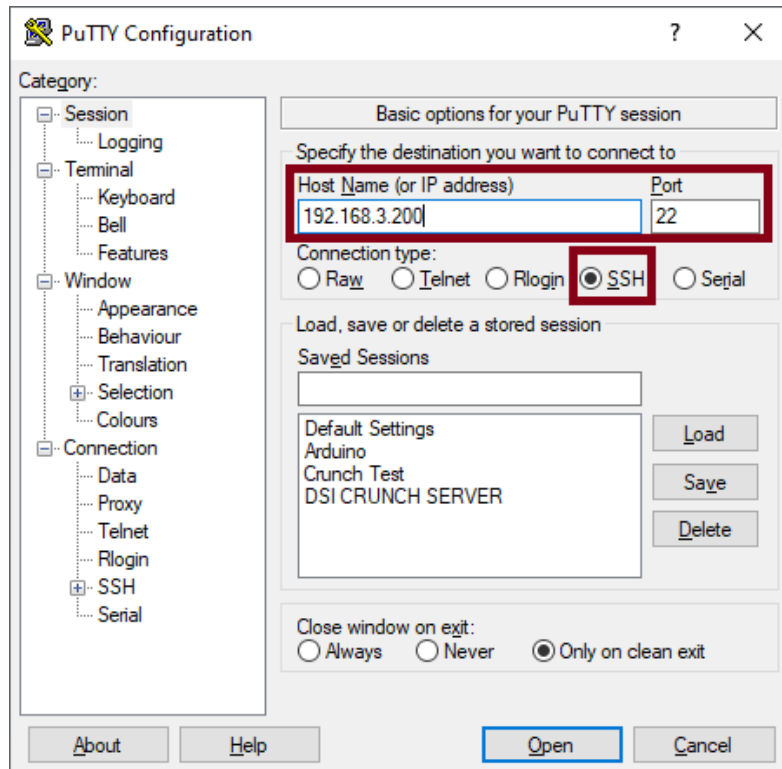
CONNEXION SSH

Pour commencer, vous allez tester la connexion avec le robot en SSH.

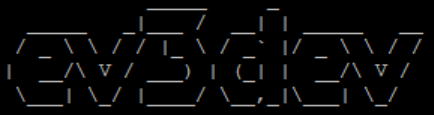
Reliez le robot à l'ordinateur grâce au câble usb. Une fois le robot connecté, celui-ci va fournir une adresse ip.



Lancez PuTTY et démarrez une connexion en utilisant l'adresse ip du robot.



Utilisez maintenant le login **robot** et le mot de passe **maker**.

```
robot@ev3dev: ~  
login as: robot  
robot@ev3dev's password:  
  
Debian Jessie on LEGO MINDSTORMS EV3!  
  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Mon Dec 28 20:53:52 2015 from freyr.lechnology.com  
robot@ev3dev:~$
```

Vous êtes maintenant connectés en SSH au robot.

CONFIGURATION WIFI

Il est maintenant conseillé de configurer le robot pour travailler en wifi de manière à éviter les désagréments de la connexion câblée usb. Vous pouvez connecter votre robot à votre smartphone ou à votre ordinateur avec un wifi en mode point d'accès.

Une fois connecté en SHH, il est possible de configurer la connexion au wifi de votre choix de la manière suivante :

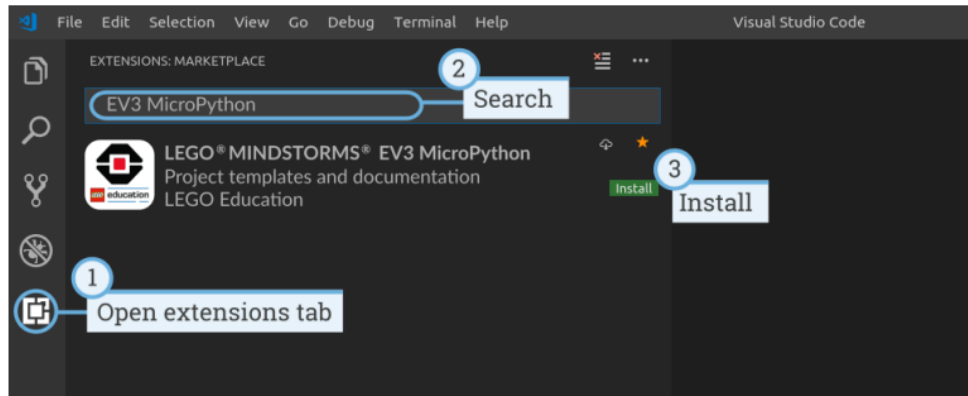
```
robot@ev3dev:~$ connmanctl
Error getting VPN connections: The name net.connman.vpn was not provided by any
connmanctl> enable wifi
Enabled wifi
connmanctl> scan wifi
Scan completed for wifi
connmanctl> services
*AO Wired                ethernet_b827ebbde13c_cable
                        wifi_e8de27077de3_hidden_managed_none
AH040444914              wifi_e8de27077de3_41483034303434393134_managed_psk
Frissie                  wifi_e8de27077de3_46726973736965_managed_psk
ruijgt gast              wifi_e8de27077de3_7275696a67742067617374_managed_psk
schuur                   wifi_e8de27077de3_736368757572_managed_psk
connmanctl> agent on
Agent registered
connmanctl> connect wifi_e8de27077de3_41      # You can use the TAB key at this point to autocomplete the name
connmanctl> connect wifi_e8de27077de3_41483034303434393134_managed_psk
Agent RequestInput wifi_e8de27077de3_41483034303434393134_managed_psk
  Passphrase = [ Type=psk, Requirement=mandatory ]
Passphrase? *****
Connected wifi_e8de27077de3_41483034303434393134_managed_psk
connmanctl> quit
```

Une fois le robot connecté en wifi, vous pouvez déconnecter le câble usb.

VISUAL STUDIO CODE

Visual Studio Code va permettre de coder en **python** et d'envoyer les programmes sur les robots.

Après avoir lancé Visual Studio Code vous pourrez installer l'extension **EV3 MicroPython** nécessaire au développement (voir ci-dessous).



Vous pourrez également vérifier la présence des extensions **Python for VSCode** et **ev3dev-browser**.

DOCUMENTATION

Vous pouvez vous référer à la documentation fournie avec ce sujet ***Getting Started EV3 MicroPython Lego.pdf***, que vous pourrez également trouver ici <https://education.lego.com/en-us/support/mindstorms-ev3/python-for-ev3>.

Attention : la partie installation est déjà effectuée, vous pouvez donc commencer directement au **chapitre 2**.

PREMIER PROGRAMME

Pour commencer vous aller faire un simple programme permettant d'afficher sur l'écran du robot la version Python (voir ci-dessous).

```
#!/usr/bin/env pybricks-micropython

from pybricks.tools import wait
import sys

print(sys.version)
wait(5000)
```

Remarque : tous vos programmes devront commencer par le shebang **#!/usr/bin/env pybricks-micropython** de manière à être dans l'environnement **pybricks**.

TRAVAIL DEMANDE

Vous allez développer plusieurs modules spécifiques sous forme de classes, et organisés dans plusieurs fichiers.

Remarque : les informations données par la suite ont pour but de vous aider à structurer votre base de travail qui servira tout au long du semestre, vous êtes libres de réinterpréter les indications, ainsi que d'ajouter toutes les classes et méthodes que vous jugerez nécessaires.

PILOTAGE DES MOTEURS



Créez une classe spécifique au contrôle du robot avec les méthodes de base suivantes :

- `void forward(speed)` : le robot avance selon la vitesse indiquée
- `void rotate(angle, aSpeed)` : le robot tourne sur lui-même selon l'angle et la vitesse angulaire donnés
- `void stop()` : le robot s'arrête

Remarque : à ce stade vous n'utiliserez que les méthodes dérivées de la classe **Motor**.

CAPTEUR DE DISTANCE



Créez une classe spécifique au capteur de distance avec au minimum une méthode **distance()** qui renvoie la distance actuelle mesurée. La classe doit pouvoir s'adapter au type du capteur, c'est-à-dire **IR** ou **Ultrason**.

CAPTEUR DE COULEUR



Créez une classe spécifique au capteur de couleur avec au minimum une méthode **color()** qui renvoie la couleur actuelle mesurée. Vous pouvez également commencer à faire des méthodes plus haut niveau permettant de détecter des zones au sol, identifiables par des couleurs distinctes.

ETAT DU ROBOT

Créez une classe spécifique permet de représenter l'état actuel du robot.

Vous pourrez notamment ajouter une variable **distance** ainsi qu'une variable **couleur** indiquant les dernières informations connues. Vous mettrez au fur et à mesure dans cette classe toutes les informations que vous jugerez utiles.

AFFICHEUR LCD



Créez une classe spécifique à l'affichage des informations via l'écran LCD.

Ajoutez une méthode **status()** permettant d'afficher l'état actuel du robot à partir de la classe précédente.

LOGS

Créez une classe spécifique permettant d'écrire des logs au format **.csv**.

Ajoutez une méthode **log()** permettant d'écrire dans un fichier l'état actuel du robot. Chaque ligne inscrite dans le fichier doit comporter une information temporelle de manière à pouvoir générer des courbes en ouvrant le fichier dans Excel.

SON ET LUMIERE

Créez une classe spécifique à l'utilisation du son et de la lumière sur la brique.



Pour les lumières il s'agit de tirer partie de la possibilité de contrôler l'affichage de couleur sur les LEDs gauche et droite. Vous pourrez donc créer des méthodes permettant de transmettre certaines informations, notamment grâce au clignotement.

PILOTAGE DES MOTEURS (HAUT-NIVEAU)

Testez la classe existante **DriveBase**. Vous pouvez ensuite intégrer cette classe à votre contrôleur de robot.

COMMUNICATION SANS-FIL

L'idée principale va être de connecter plusieurs robots en wifi à un point d'accès (ordinateur) de manière à faire partie d'un même réseau. L'ordinateur doit exécuter un serveur en python qui va attendre les connexions des clients (les robots). Une fois le réseau établi, les clients doivent pouvoir envoyer des messages au serveur et inversement. Un robot doit également pouvoir communiquer avec n'importe quel autre robot du réseau (en passant par le point d'accès). Chaque classe (serveur et client) aura donc au minimum une méthode **send()** et une méthode **receive()** permettant d'échanger des messages.

GESTION DES EVENEMENTS

Vous pourrez améliorer votre code de manière à intégrer une gestion des événements. Par exemple un événement pourrait être déclenché à chaque fois que votre module couleur détecte un changement de zone ou encore lorsqu'une limite de proximité est atteinte par le capteur de distance.