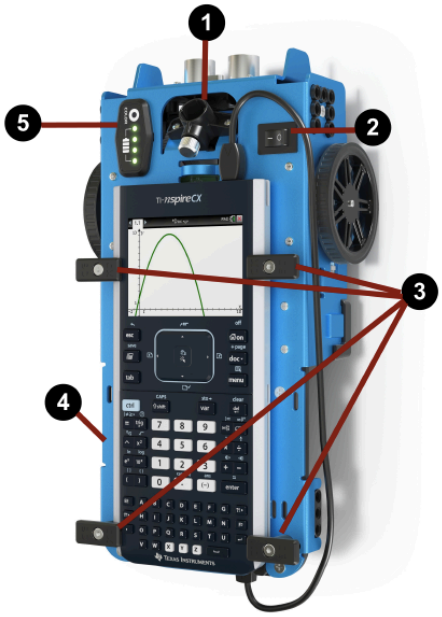
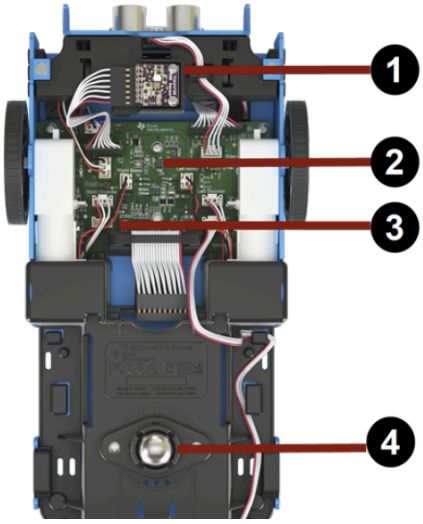
TP prise en main du ROVER

Explorons un peu ce Rover avant de le programmer :



1 : Support de crayon.
2 : Bouton Marche/Arrêt.
3 : Supports de calculatrice (**Attention : bien les soulever avant de les tourner !**)
4 : Plate-forme pour la calculatrice.
5 : Panneau DEL indicateur de batterie.



1 : Capteur de couleurs
2 : Gyroscope : mesure ou maintient l'orientation.
3 : Port I2C.
4 : Roulement à bille.

Le capteur à ultrasons pour mesurer la distance avec des obstacles par exemple.

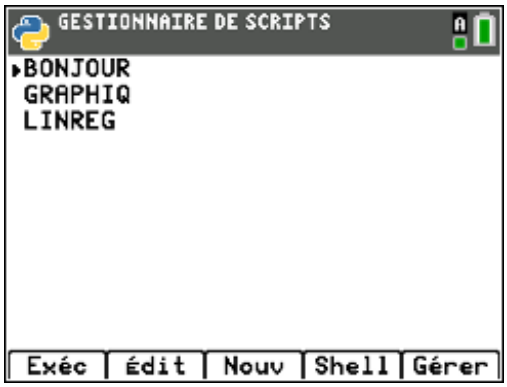
Capteur de luminosité « BRIGHTNESS ».
Port I2C pour connecter de HUB au Rover.
Port mini-USB B pour connecter la calculatrice au HUB.

Pour la connexion :

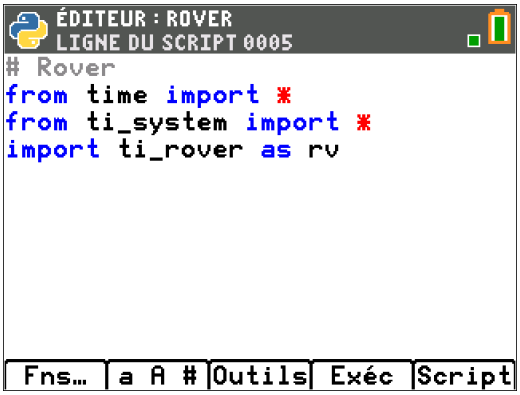
1. Prenez le Hub, et placer la nappe à 20 broches dans l'emplacement BREADBOARD.
ATTENTION : le fil rouge doit être du côté alimentation 5V !
2. Placer le Rover sur le dos, puis glissez **délicatement** le Hub dans l'emplacement situé à l'arrière du Rover (jusqu'au clic). Récupérer la nappe.
3. Brancher la nappe **délicatement** à la carte du Rover (un indicateur vous empêche de vous tromper de sens).
4. Connecter le câble I2C à la prise I2C du Hub et sur l'emplacement I2C 2 de la carte du Rover (m'appeler pour cette phase).
5. Placer le Rover sur ses roues et mettez en place la calculatrice.
Attention : Avant de faire tourner les supports de calculatrice, penser à les soulever.
Une fois la calculatrice en place, faire tourner les supports en plaçant le côté CE côté calculatrice.
6. Connecter la calculatrice au Hub en utilisant le câble data B (le connecteur B au port B du Hub et le connecteur A à la calculatrice).

Pour tous les tests de programme avec le Rover, posez-le sur le sol dans un endroit dégagé d'objets qui pourraient l'endommager.

Pour la création de votre programme, aller dans prgm puis choisissez `2 : Python App`.
Avec les menus du bas, vous pouvez exécuter votre code, le modifier, en créer un nouveau par exemple.



Attention, lorsque vous créer un nouveau programme, au moment de taper son NOM pensez à sélectionner dans type :7 :Rover.
Vous devez avoir cet écran :

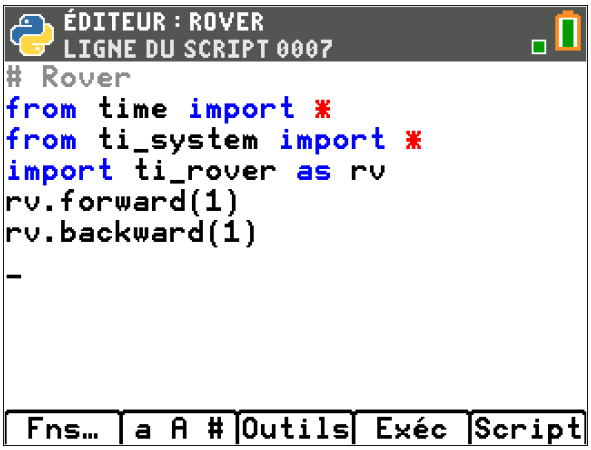


Voyons quelques commandes de base :

Avancer et reculer :
Les commandes se situent dans Fns... puis Modul puis `ti_rover` puis Conduire

L'unité par défaut est le décimètre !

Voici le code :



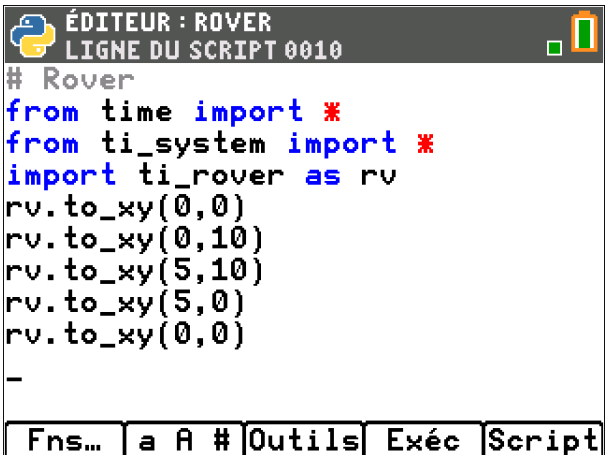
Taper-le, puis exécutez-le **avec le Rover au sol**.

Maintenant, à l'aide des commandes
`rv.forward(distance)`, `left(angle)`, `right(angle)` faîtes

faire à votre Rover un parcours rectangulaire de 1m par 50 cm.
Vous pouvez optimiser votre programme à l'aide du boucle For (`for i in range(début,fin)`).

Une autre façon de faire :
On peut aussi faire déplacer le Rover dans un repère orthonormé avec pour unité 1dm.

Je vous propose ce script :



Pour aller plus loin :
On peut faire un programme qui permette que le Rover suive un chemin polygonal dont on précise le nombre de côtés et la longueur de ceux-ci et qui permet de plus de suivre le chemin du Rover sur la calculatrice.

Voici un exemple de code à tester :

```
from time import *
from ti_system import *
import ti_rover as rv
import ti_plotlib as plt

def polygone(l,c):
  plt.cls()
  plt.axes("on")
  plt.pen("medium", "solid")
  plt.color(255, 0, 0)
  plt.grid(1, 1, "dot")
  plt.title("polygone regulier a " + str(c) + " cotes")
  x = 0
  y = 0
  b = 360 / c
  for i in range(c):
    rv.forward(l)
    rv.right(b)
    rv.wait_until_done()
    X = rv.waypoint_x()
    Y = rv.waypoint_y()
    plt.line(x, y, X, Y, "arrow")
    x = X
    y = Y
  plt.show_plot()
  rv.disconnect_rv()
```

Quelques remarques sur le code :
`plt.cls()` : permet d'effacer l'écran.
plt.axes`("on"):` permet l'affichage des axes.
`plt.pen("medium", "solid"):` permet de modifier le style du tracé.
`lt.color(255, 0, 0) :` permet de choisir la couleur du tracé.

`plt.grid(1, 1, "dot"):` permet l'affichage d'une grille avec des points
`plt.title("polygone regulier a " + str(c) + " cotes"):` permet l'affichage d'un titre au graphique. Le `str(c)` permet de convertir le paramètre c en caractère.