

Jumelage numérique de **Ropy** avec **Maqueen**



Présentation de Ropy et de son environnement de programmation



Ropy est un rover martien qui se commande grâce au langage **Python**. L'interface de programmation se décompose en 2 fenêtres : un éditeur de texte et le simulateur.



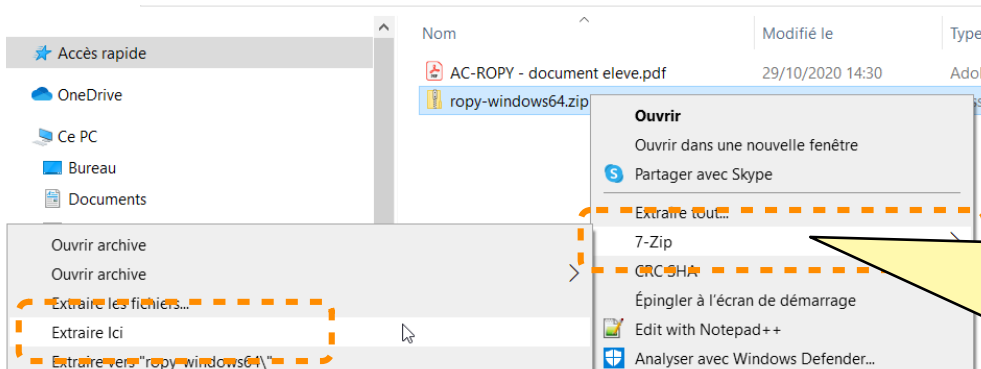
Le **simulateur** permet de **visualiser l'évolution du Rover**.

Un **éditeur de texte** (Notepad++, Spyder, Atom, Emacs, ...) pour **écrire le programme en Python**.

```
Spyder (Python 3.9)
Fichier  Édition  Recherche  Source  Exécution  Débuguer  Console  Projets  Outils  Affichage  Aide
/home/phroy/Bureau/SNT/2 - Python/Ropy/ropy-v2.0-linux64/rp_cmd.py
rp_cmd.py
1  import bge # Bibliothèque Blender Game Engine (UPBGE)
2  import time
3  from rp_lib import * # Bibliothèque Ropy
4
5  #####
6  # rp_cmd.py
7  # @title: Commandes pour le Rover Ropy
8  # @project: Ropy (Blender-EduTech)
9  #####
10
11  # Initialisation du niveau :
12  # Niveau 1 : Les premiers pas de Ropy
13  # Niveau 2 : Ma première fonction
14  # Niveau 3 : Sécuriser Ropy
15  # Niveau 4 : Partir au bout du monde
16  # Niveau 5 : Faire face à l'inconnu
17  # Niveau 6 : Se rendre utile
18  #####
19
20  #####
21  # Fonctions
22  #####
23
24  #####
25  # Commandes
26  #####
27
28
29  def commandes():
30
31  # Ecrire votre code ici
```

Éditer le programme avec Spyder

Ouvrir le fichier rp_cmd.py

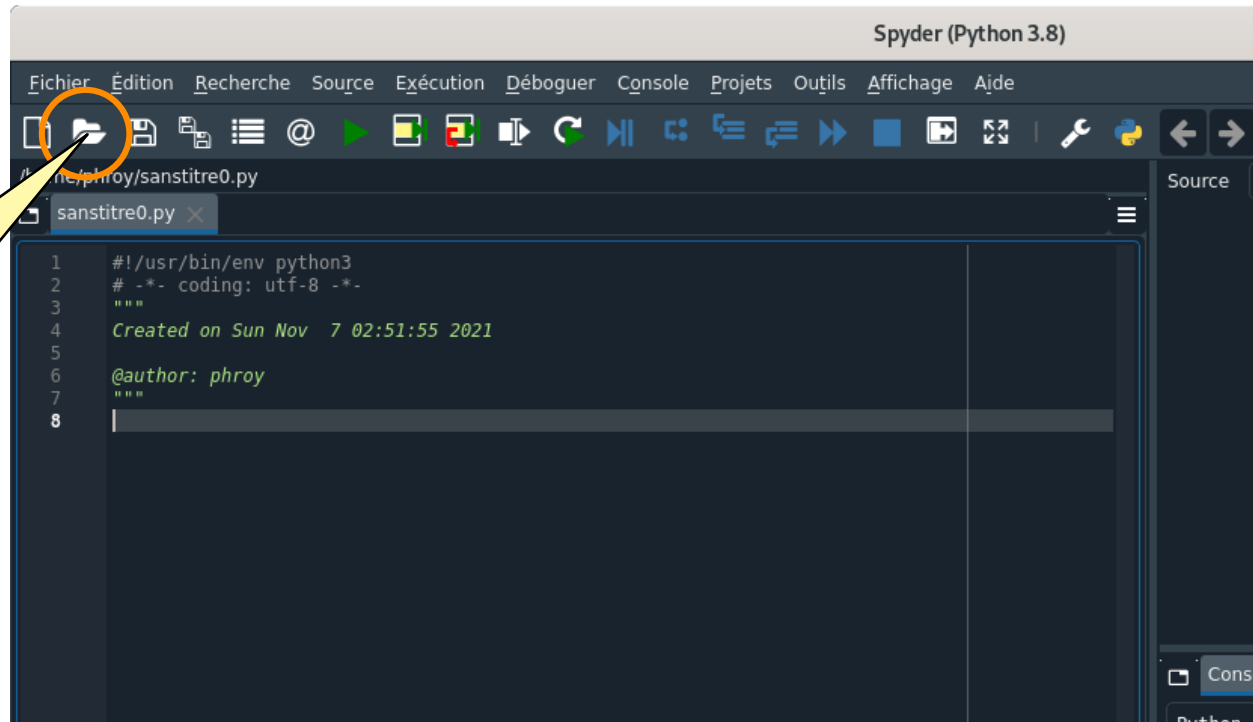


1 : Récupérer l'archive **ropy-windows64.zip** et la décompresser avec **7-Zip** dans votre répertoire. L'extraction va créer le répertoire **ropy**

2 : Lancer le Logiciel **Spyder**.



3: Ouvrir le fichier Python à éditer **rp_cmd.py** (Ropy commandes) présent dans le répertoire **ropy**.



Éditer le programme avec Spyder

Exécution du programme



5 : **Sauvegarder** le fichier

Attention !

Toujours sauvegarder le fichier avant son exécution avec le simulateur.

Le simulateur est le programme **ropy.exe**

Arrêter et réinitialiser

Afficher l'aide

Niveau actuel

6 : **Exécuter** le programme

Afficher l'objectif

Aller à la boutique

4 : **Écrire** le code Python

Afficher les tâches de la mission

```
Fichier  Édition  Recherche  Source  Exécution  Déboguer  Console
/home/phroy/Bu
rp_cmd.py x
2 import tim
3 from rp_li
4
5 #####
6 # rp_cmd.p
7 # @title: C
8 # @project: Ropy
9 #####
10
11 #####
12 # Initialisation du niveau :
13 # Niveau 1 : Les premiers pas de Ropy
14 # Niveau 2 : Ma première fonction
15 # Niveau 3 : Sécuriser Ropy
16 # Niveau 4 : Partir au bout du monde
17 # Niveau 5 : Faire face à l'inconnu
18 # Niveau 6 : Se rendre utile
19 #####
20
21 #####
22 # Fonctions
23 #####
24
25 #####
26 # Commandes
27 #####
28
29 def commandes():
30
31 rp_gauche()
32 rp_avancer()
33 rp_avancer()
34 rp_avancer()
35 rp_avancer()
36
37
38 rp_fin() # A garder
39
```



Contenu du fichier rp_cmd.py



Le fichier `rp_cmd.py` comporte 4 sections.

```
import bge # Bibliothèque Blender Game Engine (UPBGE)
import time
from rp_lib import * # Bibliothèque Ropy

#####
# rp_cmd.py
# @title: Commandes pour le Rover Ropy
# @project: Ropy (Blender-EduTech)
#####

#####
# Initialisation du niveau :
# Niveau 1 : Les premiers pas de Ropy
# Niveau 2 : Ma première fonction
# Niveau 3 : Sécuriser Ropy
# Niveau 4 : Partir au bout du monde
# Niveau 5 : Faire face à l'inconnu
# Niveau 6 : Se rendre utile
#####

#####
# Fonctions
#####

#####
# Commandes
#####

def commandes():
    ➔ rp_gauche()
    rp_avancer()
    rp_avancer()
    rp_avancer()
    rp_avancer()

    rp_fin() # A garder

#####
# En: Externals calls << DONT CHANGE THIS SECTION >>
# Fr: Appels externes << NE PAS MODIFIER CETTE SECTION >>
#####

if __name__=='start':
    thread_cmd_start(commandes)
if __name__=='stop':
    thread_cmd_stop()
```

Le code doit être indenté
(décalé sur la droite) avec
la touche Tab

} **Import des bibliothèques**
Ne pas modifier cette section

} **Fonctions** : section pour le
codage de **vos fonctions**

} **Commandes** : section pour le
codage des commandes du robot

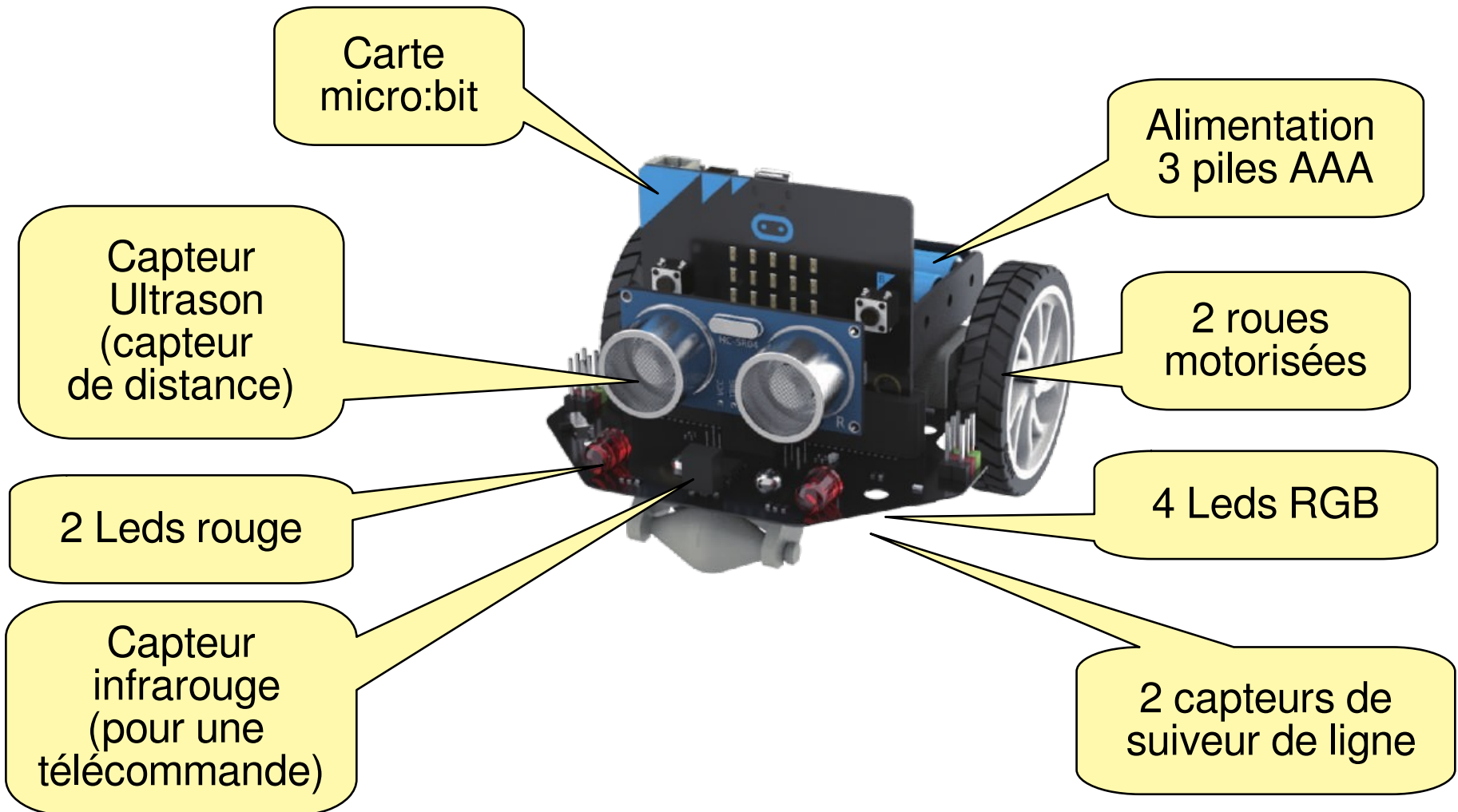
} **La commande `rp_fin()`**
est à conserver.

} **Appels du simulateur**
(Blender Game Engine)
Ne pas modifier cette section

Présentation du robot Maqueen



Maqueen est un robot mobile qui peut se piloter avec une carte **micro:bit**.



Éditer un programme Python pour les cartes micro:bit



L'édition du programme Python va se faire avec l'éditeur en ligne du site : <https://python.microbit.org/> avec le navigateur Chrome.

The screenshot shows the micro:bit Python Editor interface. On the left is a sidebar with various components like Variables, Display, Buttons, Logic, Accelerometer, Comments, and Maths. The main area contains a code editor with Python code for a micro:bit program. On the right is a simulator window showing a virtual micro:bit board with a play button and a serial terminal.

Aide

Connecter la carte puis téléverser le programme vers la carte

Import de la bibliothèque micro:bit

Boucle principale `While True :` Attention à l'indentation des instructions qui suivent.

Ouvrir et sauvegarder un programme Python

Simulateur

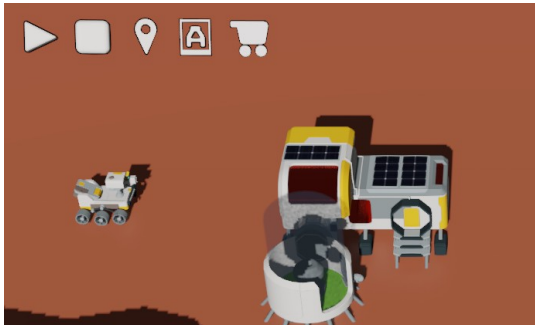
```
1 # Imports go at the top
2 from microbit import *
3
4
5 # Code in a 'while True:' loop repeats forever
6 while True:
7     display.show(Image.HEART)
8     sleep(1000)
9     display.scroll('Hello!')
10
```

Mission 1 : Mettre en place le jumeau réel Maqueen



Tout d'abords il faut mettre place le jumeau réel, c'est à dire il faut que le robot **Maqueen** soit à l'écoute des ordres émis par **Ropy** pour ensuite les exécuter.

Ropy



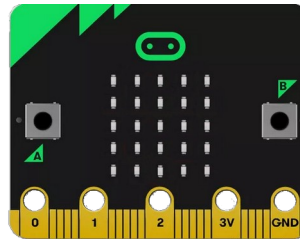
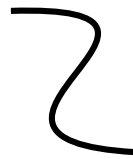
Dans le programme Python
rp_cmd.py,
écrire au tout début
l'établissement de la
communication série:

```
rp_jumeau('COM1', 115200)
```

COM1 est le port de
communication
(voir page suivante)

Liaison série (USB)

Carte micro:bit relais



Charger le programme
rp_maqueen-relay.py
(répertoire twins)
dans la carte.

Liaison radio



Carte micro:bit robot



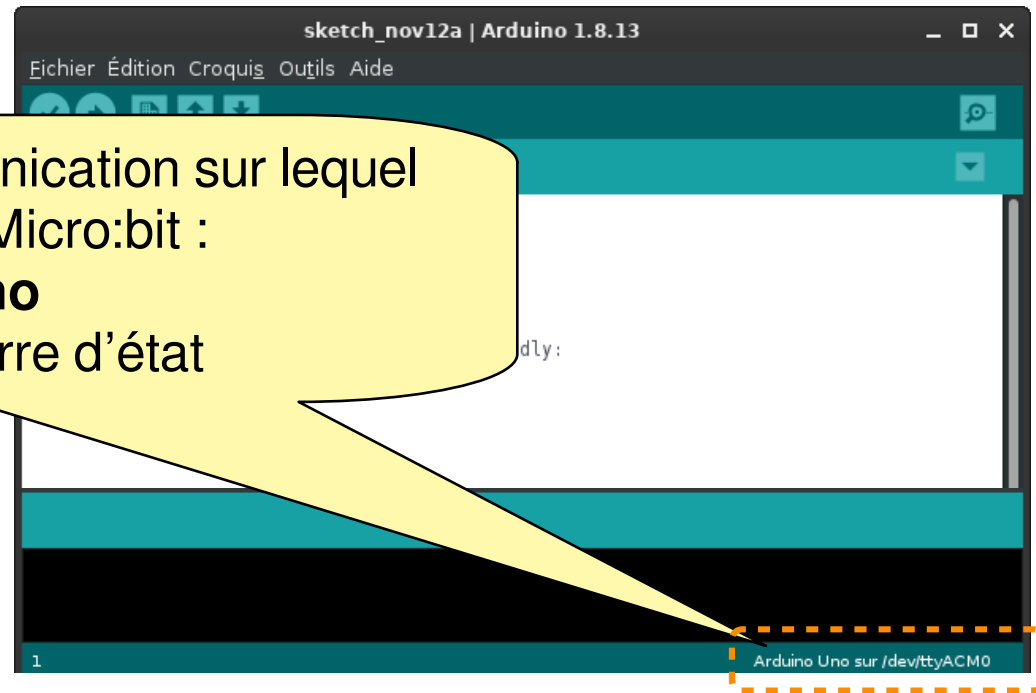
Charger le programme
rp_maqueen-robot.py
(répertoire twins)
dans la carte.

Mission 1 : Mettre en place le jumeau réel

Programme rp_cmd.py



- 1 : **Identifier** le port de communication sur lequel est branché la carte Micro:bit :
- lancer le **programme Arduino**
 - lire le **port** détecté dans la barre d'état



```
File Edit Search Source Run Debug Consoles Projects
/home/phroy/Bureau/seriousgames/blender-edutech/git/ropy/rp_c
rp_cmd.py* x
24
25 #####
26 # Commandes
27 #####
28
29 def commandes():
30
31     rp_jumeau('/dev/ttyACM0', 115200)
32
33     rp_avancer()
34
35     rp_fin() # A garder
36
```

- 2 : Dans le programme **rp_cmd.py** (Spyder) **coder** le jumelage et un mouvement simple :
- ```
rp_jumeau (' /dev/ttyACM0 ' , 115200)
rp_avancer ()
```

- 3 : **Tester** le programme en observant si le robot **Maqueen** exécute bien le mouvement programmé.

# Mission 2 : Pilotage manuel du robot Maqueen



En utilisant le **même protocole de communication lié au jumelage numérique**, on vous demande de piloter le robot à partir d'une carte micro:bit utilisée comme **télécommande**.

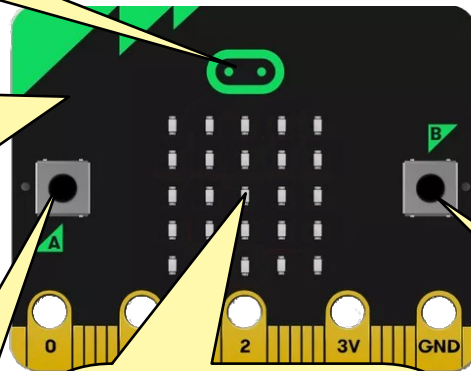
**Carte micro:bit  
télécommande**

**Liaison  
radio**

**Carte  
micro:bit  
robot**

**Marque** la case  
(carte v2)

- **Avance** si la carte est penché en avant
- **Recule** si la carte est penchée en arrière



**Affiche** le mouvement à exécuter

**Tourne** à gauche 90°

Créer le programme  
`rp_maqueen-tlcmd.py`

**Tourne** à droite 90°



Garder le programme  
`rp_maqueen-robot.py`  
(répertoire twins)  
dans la carte.

# Mission 3 : Pilotage manuel du rover Ropy



En gardant votre programme de la télécommande (précédemment réalisé), on vous demande de piloter manuellement le rover **Ropy** à partir de celle-ci.

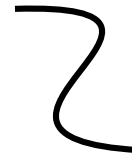
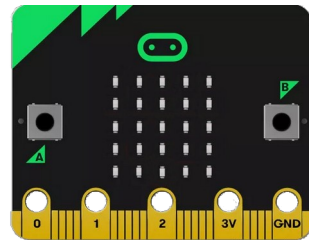
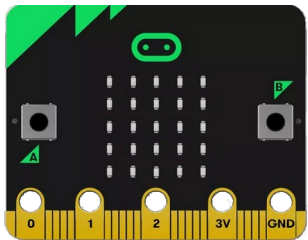
Carte  
micro:bit  
télécommande

Liaison  
radio

Carte  
micro:bit  
relais

Liaison  
Série  
(USB)

Ropy



Garder le programme  
**rp\_maqueen-tlcmd.py**

Modifier le programme  
**rp\_maqueen-relay.py**  
afin de placer le relais en  
écoute puis à transmettre les  
ordres reçus.

Éditer le programme Python **rp\_cmd.py**,  
afin de placer **Ropy** en écoute puis à  
exécuter les ordres demandés par la  
télécommande.