Structures de contrôle : boucles et conditions

Compétences

- Implanter des instructions conditionnelles simples et imbriquées.
- Différencier la boucle conditionnelle et inconditionelle

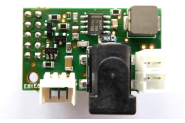
Contexte

Dans ce TP vous allez apprendre à configurer et à piloter un servomoteur Dynamixel ROBOTIS à l'aide des objets définis dans le module Python `pypot`. Il s'agit de visiter et comprendre certaines fonctionnalités bas niveau du module pypot.dynamixel. Toutes la documention nécessaire à l'utilisation de `pypot` se trouve dans les liens ci-dessous, il n'est pas nécessaire de s'y attarder dans l'immédiat.

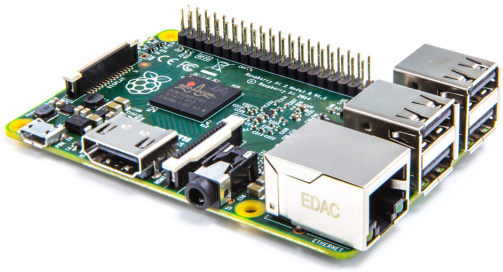
- API Pypot : http://poppy-project.github.io/pypot/api.html
- Dynamixel Low-level IO : https://poppy-project.github.io/pypot/dynamixel.html)
- Tutorial Pypot : http://poppy-project.github.io/pypot/tutorial.html

Matériel utilisé

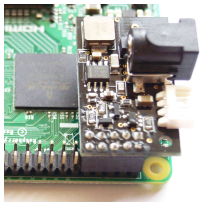
Servomoteur Dynamixel XL-320 La communication se fait par le biais d'une liaison 3 broches en TTL (Vcc, Gnd, data) encore appelée **liaison série asynchrone**, avec un débit maximum de 1Mbps. Les servomoteurs dynamixels sont reliés en série entre eux et avec une alimentation comprise entre 6 et 8V. Documentation XL-320



La **carte pixl** est conçue pour alimenter et piloter jusqu'à 6 moteurs Dynamixel compatibles. Elle se connecte directement sur les broches 1 à 10 de votre Raspberry Pi, et dispose d'un mode d'alimentation, par prise jack noire DC (7,5 V @ 2 A minimum)

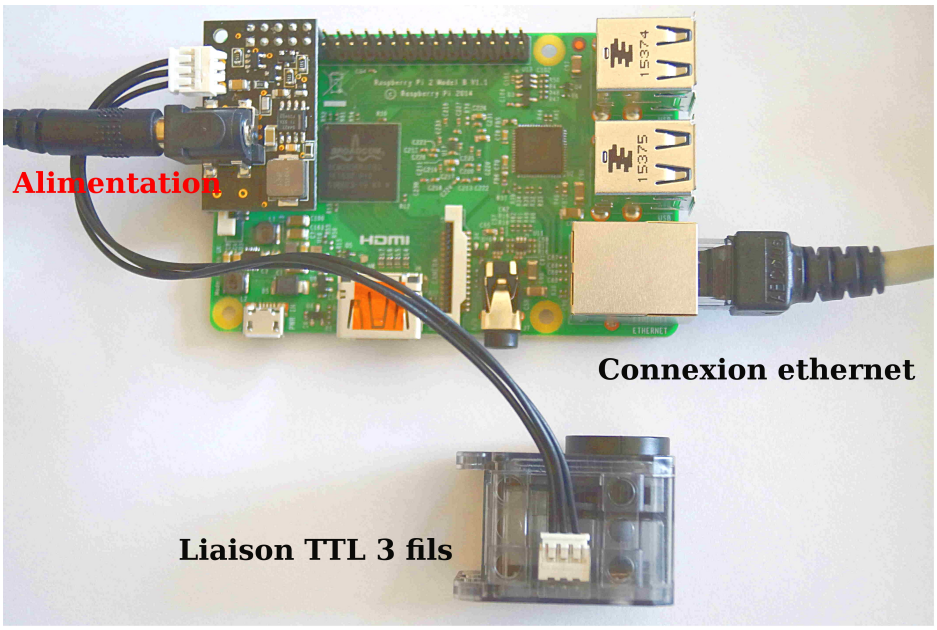


Le **Raspberry Pi** est un nano-ordinateur monocarte à processeur ARM conçu par le créateur de jeux vidéo David Braben, dans le cadre de sa fondation Raspberry Pi2.
Cet ordinateur, qui a la taille d'une carte de crédit, est destiné à encourager l'apprentissage de la programmation informatique ; il permet l'exécution de plusieurs variantes du système d'exploitation libre GNU/Linux et des logiciels compatibles. Il est fourni nu (carte mère seule, sans boîtier, alimentation, clavier, souris ni écran) dans l'objectif de diminuer les coûts et de permettre l'utilisation de matériel de récupération.



Association de la carte Pixl avec le Raspberry PI (RPI)

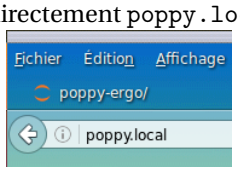
Branchement du servomoteur avec le PC



- Connecter le PC et la RPI à l'aide d'un cable ethernet (RJ45)
- Brancher le moteur sur la carte pixl à l'aide du cable TTL
- Alimenter l'ensemble avec une alimentation 7,5 V

La carte raspberry dispose d'une carte SD sur laquelle on a installé un système d'exploitation (OS : Operating System) basé sur une distribution Linux. Une fois l'OS lancé (le démarrage de l'OS est automatique au moment ou vous branchez l'alimenation de la carte) il est possible d'utiliser Anaconda (distribution Python libre). Cette distribution Python est directement accessible à travers une page d'accueil générée par un serveur Web installé sur la RPI, pour ce faire vous devez :

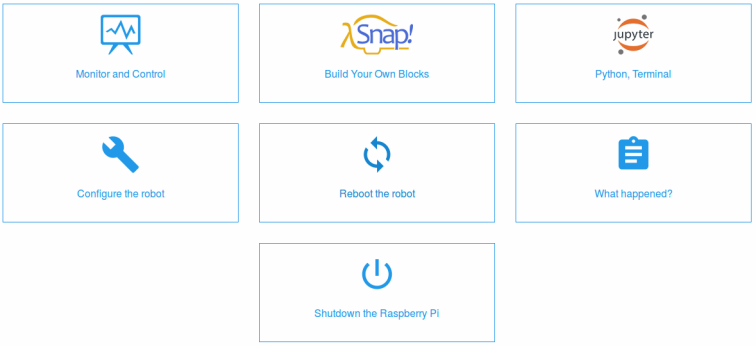
- Ouvrir un navigateur Web, par exemple firefox
- Vous connectez au serveur Web de la carte RPI en tapant l'adresse du serveur dans la barre d'url du navigateur : `HTTP://nom_du_serveur.local`, en général le nom du serveur est inscrit à coté de la carte, si ce n'est pas le cas alors vous pouvez taper : `http://poppy.local` ou directement `poppy.local`



- Une fois connecté vous devriez obtenir l'écran d'accueil. Il ne reste plus qu'à cliquer sur le bouton **Jupiter** pour obtenir un notebook.

Pour information les différents boutons permettent de controler une créature Poppy (un robot) dont nous reparlerons plus tard :

Image de la version v1.0



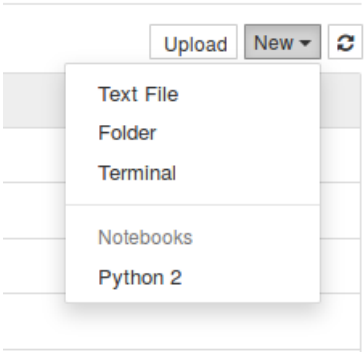
Monitor and Control pour commander le robot à partir d'une interface graphique.
Snap ! pour programmer le robot à partir du langage Snap !
Jupyter Pour programmer le robot ou les moteurs à partir de Python
Configure the robot pour modifier certaines propriétés du robot comme l'activation de la caméra.
Reboot the robot redémarrer le robot en cas de problème.
What happened ? fichier de logs
Shutdown the raspberry Pi pour éteindre le robot

Utiliser le servomoteur Dynamixel XL-320 à partir du notebook

Lancer un nouveau notebook

Si ce n'est pas encore fait, cliquer sur **Jupyter : python console**.
Pour ouvrir un nouveau notebook

- cliquer à droite de l'écran sur New
- choisir **Python 2**



Pour finir donner le nom `TP_XL-320` à votre notebook en cliquant sur **Untitled**

Créer un objet moteur de type : Dxl320IO (Dynamixel 320-IO)

Échanger des informations à travers le port série

pypot est la librairie qui permet de d'utiliser le servomoteur, le ligne 2 permet d'obtenir la version installée sur votre ordinateur.

```
1 import pypot
2 print("version de pypot installée :", pypot.__version__)
```

La connexion avec le servomoteur se fait grâce à la carte pixl branchée sur la carte RPI

```
1 import pypot.dynamixel
2 ports = pypot.dynamixel.get_available_ports()
3 if not ports:
4     print ('no serial port found!')
5 print('ports found:', ports)
```

Décrire ligne par ligne ce que fait le script ci-dessus, si votre système fonctionne correctement vous devriez observer sur la sortie standard (c'est à dire votre écran) :

- GNU/Linux : ['/dev/ttyACM...']
- Mac Os X : ['/dev/tty.usbmodem...']
- Windows : ['COM...']

Créer un objet de communication avec le(s) servomoteur(s)

L'API Pypot : Une API (Application Programmable Interface, traduisez *interface de programmation*) est un ensemble de fonctionnalités permettant d'accéder aux services d'une application, par l'intermédiaire d'un langage de programmation. Par exemple l'API Pypot explique comment créer un objet de communication avec un ou plusieurs moteurs montés en série à l'aide du constructeur suivant :
`Dxl320IO(port, baudrate=1000000, timeout=0.05, use_sync_read=False, error_handler_cls=None, convert=True)`
Le seul paramètre qui n'est pas renseigné par défaut est : `port`

1. À l'aide de l'API quel est le type attendu pour le paramètre `port` ?
2. Lors de la connexion au port série nous avons initalisé une variable `ports`, quel est son type ?
3. Pour accéder à la valeur référencée par cette variable nous écrirons : `ports[0]`, l'objet de communication peut ainsi être créé de la manière suivante :

```
xl_320 = pypot.dynamixel.Dxl320IO(ports[0])
```

Une fois l'objet instancié, il est important de fermer le port de communication avant d'en créer un autre avec la fonction : `xl_320.close()`. Il existe également l'attribut `xl_320.closed` (pas de parenthèse dans ce cas) qui renvoie **True** si la communication est ouverte et **False** sinon. Écrire une suite d'instructions qui permet de créer un nouvel objet moteur en fermant la connexion actuelle si elle existe déjà.

4. Vérifier le type de l'objet de communication avec le servomoteur.
5. Pour controler un servomoteur, il est nécessaire de connaître son identifiant. L'identifiant noté *id* est un nombre entier. Cette valeur sera ensuite passée en paramètre aux fonctions permettant d'agir sur le servomoteur. L'*id* peut être déterminé avec la fonction : `scan` Cette fonction n'est pas décrite dans l'API Dynamixel, à l'aide de la fonction `dir` puis `help` décrire le prototype de la fonction `scan`
6. Pour obtenir une liste de valeurs **ids** on peut utiliser la fonction `range(n)` ou $n \in \mathbb{N}$. Déterminer ainsi l'id de votre servomoteur

Lecture des données du servomoteur

Les informations relatives au(x) servomoteur(s) peuvent être obtenues à l'aide de fonctions ou d'attributs appelés **getters**.

- `get_moving_speed(ids)` renvoie la vitesse de rotation du ou des servomoteur(s).
  Par défaut la valeur du paramètre vitesse est à zéro, ce qui signifie que la vitesse est maximale. Vitesse à vide : 114 rpm (à 7,4V)
- `get_present_position(ids)` renvoie la position actuel du ou des servomoteur(s).
- `baudrate` renvoie la vitesse de transmission de la liaison série et correspond au nombre de bits transmis par seconde.

Vérifier les valeurs renvoyées par le servomoteur à l'aide des instructions suivantes (n'oubliez pas d'adapter l'id à celui de votre servomoteur) :

```
ids = [6]
print("get_present_speed:",xl_320.get_present_position(ids))
print("get_moving_speed:",xl_320.get_moving_speed(ids))
print("baudrate:",xl_320.baudrate)
```

Quelle différence syntaxique y a t-il entre l'utilisation d'une fonction et d'un attribut ?

Modifier les données du servomoteur

Dans cette partie, nous allons voir quelques une des fonctions **setters** de la classe Dxl320IO permettant de modifier les valeurs des paramètres du servomoteur :

- `change_id(value_for_id)` pour changer l'id d'un servomoteur
- `set_LED_color(value_for_id)` permet d'allumer les leds (rouge, verte, bleue) du servomoteur
  utile pour déterminer un mode de fonctionnement (rouge=danger)
- `switch_led_off(ids)` permet d'éteindre une led
- `set_goal_position(value_for_id)` permet de modifier la position angulaire du servomoteur
- `set_moving_speed(value_for_id)` permet de modifier la vitesse angulaire du servomoteur

1. Écrire un script permettant de faire clignoter la led rouge pendant 12 secondes avec une temporisation de clignotement de 1s (1s allumée, 1s éteinte). Pour cela on utilise Le module time qui permet de manipuler des temps de façon simple.

```
import time             #import du module temps
t0 = time.time()        #renvoie le temps écoulé en secondes depuis une date précise
print (t0-time.time()) #renvoie le temps écoulé depuis t0
```

En informatique l'idée retenue a été de représenter une date et une heure en fonction du nombre de secondes écoulées depuis une date précise. La plupart du temps, cette date est *l'Epoch Unix* (le mot epoch vient de l'anglais époque ou ère), le 1er janvier 1970 à 00 :00 :00.

La fonction `set_LED_color(value_for_id)` prend un paramètre de type dictionnaire. Par exemple pour modifier la couleur de la LED du servomoteur d'id=6 on écrira :

```
xl_320.set_LED_color({6:'green'}) #{clé:valeur}
```

2. Modifier le script pour alterner les couleurs vert, bleu, rouge, toujours avec la même temporisation.
3. Écrire un script qui permet de faire osciller, un nombre donné de fois, le servomoteur de - 50 degrés à + 50 degrés avec une vitesse de rotation de 50 degrés par seconde. La fonction `set_goal_position(value_for_id)` prend un paramètre de type dictionnaire. Par exemple pour indiquer une valeur cible au servomoteur d'id=6 on écrira :

```
xl_320.set_goal_position({6:50}) #{clé:valeur}
```

   Attention il faudra tenir compte des données constructeurs sur le servomoteur indiquant que la résolution de l'angle de rotation est de : $0,29^o$
4. Généraliser votre script pour que les oscillations s'effectuent en `angle_min` et `angle_max`. Lors des oscillations la LED verte est allumée sinon la LED est rouge.