Session Hiver 2021

Technologie de systèmes ordinés (Électronique programmable et robotique)

247-4B6-GG
Réaliser un projet de logiciel appliqué à la robotique

Lab 1 :
Dépendances et comprendre le API

Enseigné par Rachid Benali

Laboratoires et programmation réalisés par
Félix Chenette-Stewart

Dépendances

Tous vos projets et scripts devront contenir le code suivant au début afin d'assurer le bon fonctionnement des librairies pour les AX-12.

```
from pyax12.connection import Connection
from pyax12.argparse_default import common_argument_parser
import pyax12.packet as pk
import RPi.GPIO as gpio
import time
```

Option de déroutage de paramètres et debugging

Utilisé principalement pour le debugging mais est aussi utilisé pour contourner certains paramètres par défaut. Il n'est pas nécessaire de l'utiliser si le projet fonctionne bien mais le code suivant doit être présent dans votre script pour assurer la présence des paramètres par défaut.

```
parser = common_argument_parser(desc=main.__doc__)
    args = parser.parse_args()
```

Connection au moteurs AX-12

```
serial_connection = Connection(port='/dev/ttyS0',
                               baudrate=1000000,
                               timeout=0.01,
                               rpi_gpio=18)
```

Identification du moteur AX-12

```
dynamixel_id = 0x10
```

Convertir son numéro de moteur en hexadécimal et rajouter 0x devant
Ex : moteur no 14 = E en hexadécimal
Votre ligne de code : dynamixel_id = 0x0E
*args.dynamixel_id est utilisé pour communiquer avec tous les moteurs AX-12.

API

```
def set_cw_angle_limit(self, dynamixel_id, angle_limit, degrees=False):
# et
def set_ccw_angle_limit(self, dynamixel_id, angle_limit, degrees=False):
```

Utilisation: Limite l'angle de rotation sens horaire(cw) et anti-horaire (ccw) à l'angle spécifié (angle limit).

:param int dynamixel_id: No. d'identification du AX-12 (0x00 à 0xFD).

:param int angle_limit: et :param bool degrees:
Si "Degrees'' = True, la valeur en degré (angle_limit) doit être entre -150 et 150.
Si ''Degrees'' = False, le moteur sera en rotation libre. Les valeurs possibles (angle_limit) seront de 0x00 à 0x3FF pour le sens horaire et de 0x3FF à 0x7FF pour le sens anti-horaire.

*Si on spécifie une limite de ''0'' degrés pour ''CW'' et ''CCW'', et que ''degrees''=False, les AX-12 seront en mode moteur ou ''Wheel mode'' et tourneront tant et aussi longtemps que la commande sera en exécution. Pour limiter le temps d'exécution, on peut utiliser time.sleep(*temps en secondes*) pour suspendre l'exécution du code. Par exemple, time.sleep(5) va suspendre les opérations pour 5 secondes. Pour quitter le mode moteur ou ''Wheel mode'' et revenir au mode stepper, il faut donner à angle_limit, une valeur supérieure à 0. Exemple, on peut faire la ligne de code suivante :

```
serial_connection.set_ccw_angle_limit(dynamixel_id, 1023, degrees=False)
```

Puisque CCW et CW n'ont pas tous les deux un angle_limit de 0, le AX-12 reviendras en mode stepper.

```
def set_speed(self, dynamixel_id, speed):
```

Utilisation: Détermine la Vitesse de rotation du AX-12

:param int dynamixel_id: No. d'identification du AX-12 (0x00 à 0xFD).

:param int speed: Vitesse de rotation. (0 à 1023) ou (0x00 à 0x3FF).

```
def goto(self, dynamixel_id, position, speed=None, degrees=False):
```

Utilisation: Fait tourner le moteur (dynamixel_id) jusqu'à un angle déterminé (position), à une certaine vitesse (speed).

:param int dynamixel_id: No. d'identification du AX-12 (0x00 à 0xFD)

:param int speed: Vitesse de rotation. (0 à 1023) ou (0x00 à 0x3FF).

:param int position: et :param bool degrees:
Si "Degrees'' = True, la valeur en degré (angle_limit) doit être entre -150 et 150.
Si ''Degrees'' = False, le moteur sera en rotation libre. Les valeurs possibles (angle_limit) seront de 0x00 à 0x3FF pour le sens horaire et de 0x3FF à 0x7FF pour le sens anti-horaire.

Exemple de code complet :

```
from pyax12.connection import Connection
from pyax12.argparse_default import common_argument_parser
import pyax12.packet as pk
import RPi.GPIO as gpio
import time

def main():

    parser = common_argument_parser(desc=main.__doc__)
    args = parser.parse_args()

    serial_connection = Connection(port='/dev/ttyS0',
                                   baudrate=1000000,
                                   timeout=0.01,
                                   rpi_gpio=18)



    dynamixel_id = args.dynamixel_id

    ##
    serial_connection.goto(dynamixel_id, 0, speed=200, degrees=True)
    time.sleep(1)
    serial_connection.goto(dynamixel_id, -150, speed=150, degrees=True)
    time.sleep(1)
    serial_connection.goto(dynamixel_id, 150, speed=300, degrees=True)
    time.sleep(1)
    serial_connection.goto(dynamixel_id, 0, speed=512, degrees=True)
    ##
    time.sleep(5)
    ###
    serial_connection.set_cw_angle_limit(dynamixel_id, 0, degrees=False)
    serial_connection.set_ccw_angle_limit(dynamixel_id, 0, degrees=False)
    serial_connection.set_speed(dynamixel_id, 512)
    time.sleep(2)
    serial_connection.set_speed(dynamixel_id, 0)
    serial_connection.set_ccw_angle_limit(dynamixel_id, 1023, degrees=False)
    serial_connection.goto(dynamixel_id, 0, speed=512, degrees=True)
    ###

    serial_connection.close()

if __name__ == '__main__':
    main()
```

Questions :
Dans un document séparé,

1. Expliquez le fonctionnement en détail du bloc de code entre les ‘’##’’.

2. Expliquez le fonctionnement en détail du bloc de code entre les ‘’###’’.

Ex : Pour “serial_connection.goto(dynamixel_id, 0, speed=50, degrees=True)”, le moteur AX-12 se déplacera à la position 0 à une vitesse de 50.