TP 7 et 8

Pour ces TP 7 et 8, il s'agit de commencer à communiquer via des programmes Python, entre l'ordinateur et le robot Thymio.

La connexion avec le robot s'effectue par de la programmation *événementielle* : votre programme doit récupérer les événements produits par le robot, envoyer des événements au robot.

Voici un petit programme :

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import pythymio

sendableEvents = pythymio.customEvents('colors', 'circle') # liste d'événements à envoyer
receivableEvents = ["buttons","prox"] # liste d'événements à recevoir

with pythymio.thymio(receivableEvents, sendableEvents) as Thym:
   def dispatch(evt_id, evt_name, evt_args): # fonction de réception d'événements
      Thym.send_event('become.yellow') # envoi d'un événement (de couleur)
      if evt_name == "fwd.prox": # toutes les 0.1 sec
         print(evt_args[2]) # réception d'un événement:
               # ici, valeur indiquant la proximité d'un obstacle devant le thymio
               # 0 = pas d'obstacle, 4000 = obstacle proche

    # lancement de l'écoute des événements
    Thym.loop(dispatch)
```

Evénements possibles pour le Thymio

Un événement est donné par un identifiant (`evt_id` ci-dessus), un nom (`evt_name` ci-dessus), un tableau d'arguments (`evt_args` ci-dessus).

Dans un programme python, il faut déclarer le tableau des noms d'événements que le programme peut recevoir du robot (en général, les valeurs de capteurs sur le robot) et le tableau des noms d'événements que le programme peut envoyer au robot (des sons, des couleurs, des ordres sur les roues, ...). Ces deux tableaux sont les paramètres de la méthode `pythymio.thymio`. Un nom d'événement est une chaîne de caractères.

Il y a deux manières de définir ces tableaux d'événements :

- soit en déclarant directement le tableau à partir des noms d'événements reconnus par le robot Thymio : c'est le cas de `receivableEvents` ci-dessus. Aller sur la page `https://www.thymio.org/fr:thymioapi` pour les détails sur les événements à recevoir ou à envoyer.
- soit en utilisant la méthode `pythymio.customEvents()` qui prend une liste de chaînes de caractères. Ces arguments sont des alias pour des tableaux de noms d'événements reconnus par le Thymio : c'est le cas de `sendableEvents` ci-dessus. La liste des alias est donnée ci-dessous, elle pourrait être complétée pour les prochains TP :
  - `buttons` pour les noms de boutons : "button.backward", "button.left", "button.center", "button.forward", "button.right"
  - `prox` pour les valeurs de distance correspondant aux noms "prox.*[*]"
  - `acc` pour les valeurs de distance correspondant aux noms "acc[*]"

— `temperature` pour la valeur de température
— `motor` pour les valeurs liées aux deux roues (noms "motor.right.*" et "motor.left.*")

Pour envoyer un événement au Thymio : `Thym.send_event(<NOM_EVT>)` où `<NOM_EVT>` est un nom d'événement. La réception des événements se fait par la fonction `dispatch`.

Préparatifs

Créer un répertoire dans lequel vous allez écrire vos codes. Copiez d'abord dans ce répertoire le fichier `pythymio.py` qui transformera vos commandes python en commandes reconnaissables par le langage du robot.

Pour chaque question :

— Ecrivez un programme Python dans un fichier texte (par exemple dans un fichier `Atemperature.py` pour la 1ère question).
— Quand vous êtes sûr de votre code, demandez au chargé de TP un robot Thymio et effectuez les opérations suivantes :

1. Connecter le Thymio sur un port USB.
2. Dans un terminal, lancer la commande `asebamedulla "ser:device=/dev/ttyACM0"` (attention : c'est ACM0 avec un zéro !). Il n'y aura pas d'affichage : c'est normal. Votre Thymio est maintenant connecté et prêt à communiquer par programme avec l'ordinateur. N'interrompez pas ce programme !
3. Lancer l'exécution de votre programme dans un autre terminal (par exemple `python Atemperature.py`).

A : Température

Pour votre premier programme avec Thymio, nous allons exploiter le capteur de température. Les évènements n'iront que dans un sens, du Thymio vers votre programme. Créer un programme Python. Pour communiquer avec le Thymio, il vous faut importer le paquetage : `\import pythymio`, et créer un bloc comme ci-dessus. Pour cet exercice, choisissez juste `["temperature"]` comme liste d'événements à recevoir, vous pouvez mettre `sendableEvents` à `[]`. Dans cette instruction, Thym est un surnom choisi librement pour parler du Thymio.
À l'intérieur de ce bloc il faut créer comme dans l'exemple ci-dessus une fonction qui traitera les événements en provenance du Thymio. Nous attachons aussi cette fonction à la boucle événementielle du Thymio en utilisant l'instruction : `Thym.loop(dispatch)` (si vous gardez `dispatch` comme nom de fonction, vous pouvez appeler votre fonction autrement). Cette fonction va être appelée à chaque événement du Thymio jusqu'à ce qu'on décide d'arrêter le programme avec Ctrl-C. Cette fonction reçoit en entrée trois arguments : un identifiant d'événement (que nous n'utiliserons pas), le nom de l'événement prefixé par `fwd.`, et une liste contenant les arguments de l'événement. Dans le cas de l'événement `fwd.temperature` nous recevrons une liste avec une seule valeur, la température actuelle en degrés Kelvin. Cet événement est déclenché à une fréquence fixe de 1Hz, ce qui est parfait pour effectuer un affichage. Affichez la température lue par le Thymio à chaque réception de l'événement `fwd.temperature`.

B : Filtrage par événement

La liste evts1 ne contient qu'un seul événement. Que se passe t'il si vous ajoutez à cette liste d'autres événements, par exemple `button.center` ? Faites en sorte que seul l'événement `fwd.temperature` déclenche un affichage. Affichez également `bouton du milieu` à chaque fois que est reçu l'événement `fwd.button.center`. Quel est le problème avec l'événement `button.center` ?

C : Accéléromètre

Ecrivez un programme pour n'écouter que l'événement interne `acc` avec lequel nous recevrons la liste des trois valeurs lues par l'accéléromètre. Cet événement est déclenché à une fréquence fixe de 16Hz. C'est un peu trop rapide pour un affichage. Nous allons donc utiliser un compte à rebours pour ne faire l'affichage des valeurs que

toutes les secondes. Il vous faudra utiliser un compteur `delay` de façon à attendre 16 fois que l'événement ait été déclenché avant un nouvel affichage. Vous aurez besoin d'une variable globale, que la fonction `dispatch` devra pouvoir modifier. Vous rappelez vous comment faire ?
Comme cette situation est appelée à se reproduire, plutôt que de multiplier les variables glo- bales, créez en une seule qui contiendra tout l'état du programme entre deux appels à `dispatch`. Pour cela créez un dictionnaire `state` et utilisez des clés différentes pour les différentes valeurs que vous voulez stocker. Par exemple, `state["delay"]` pourra contenir un entier valant initialement 15, decrémenté à chaque événement `fwd.acc` et qui, lorsqu'il atteindra zéro, déclenchera l'affichage puis se remettra à 15.
Quelle est la valeur l'attraction terrestre selon Thymio ?

D : Quitter

L'événement interne `buttons` a une fréquence de 100Hz. Attention, il y a de quoi bien occuper votre machine si vous faites un affichage à chaque réception. Cet événement retourne une liste de 5 valeurs, 0 ou 1, qui encode le fait que les boutons sont pressés ou non. Utilisez la technique du délai employée précédemment pour ne faire un affichage que tous les 80 centièmes de seconde. Trouvez à quoi correspond chaque bouton, puis faites en sorte de quitter la boucle événementielle avec `Thym.stop()` lorsque on appuie simultanément sur le bouton gauche et le bouton droit.

E : Jeu Simon

Le jeu Simon consiste à observer une séquence de boutons tirée au hasard puis reproduire la même séquence. La difficulté progresse en augmentant la longueur de la séquence. Le but du jeu est d'atteindre la plus haute difficulté possible. Pour que le jeu soit agréable la séquence de bouton s'accompagne de sons et de couleurs, un son et une couleur différente par bouton. Le bouton central n'entrera pas dans la composition des séquences.

1. Commencez par écrire une fonction qui prend en entrée un entier et retourne une séquence de lettres tirées parmi U, D, L et R (up/haut, down/bas, left/gauche, right/droite) de longueur l'entier reçu en paramètre. Testez votre fonction sur un programme ne nécessitant pas Thymio.
2. Vous allez utiliser Thymio pour montrer une séquence de boutons, en allumant des leds en direction des boutons (haut, bas, gauche, droite). Pour cela vous aurez besoin de créer des événements qui iront du programme au Thymio. Les événements qui vous intéressent sont : `circle.right`, `circle.left`, `circle.front`, `circle.back` et `circle.off`. Pour envoyer un événement au Thymio, par exemple `circle.off`, vous pouvez utiliser `Thym.send_event("circle.off")`. Vous pouvez ajouter du son et de la couleur à l'aide des événements : `chord.C3`, `chord.E3`, `chord.G3`, `chord.B3`, `sound.bad`, `sound.good`, `become.yellow`, `become.violet`, `become.green`, `become.blue`, `become.red`, `become.blank`. Associez à chaque lettre U, D, L et R un bouton, une couleur, un son. Utilisez la fonction précédente pour générer une suite de lettres parmi U, D, L et R, puis faites "jouer" bouton-couleur-son associées à ces lettres.