TP 11 et 12

RAPPEL : Aller sur la page `https://www.thymio.org/fr:thymioapi` pour les détails sur les événements à recevoir ou à envoyer. Pour activer le robot, dans une console, lancer la commande `asebamedulla "ser:device=/dev/ttyACM0"` (attention : c'est ACM0 avec un zéro !).

Ce TP a comme objectif de mettre au point des fonctions pour faire se déplacer le Thymio, et lui faire repérer les obstacles qui l'entourent.

A : Exemple de déplacement du Thymio

Le code suivant permet de faire se déplacer le robot.

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

with pythymio.thymio(["motor"],[]) as Thym:

    state = dict([])
    state["time"] = 0
    state["prog"] = dict([
        (90, (500, 500)),
        (140, (0, 0)),
        (180, (-500, 500)),
        (288, (0, 0)),
        (350, (500, 500)),
        (400, (0,0)),
        (450, (500, -500)),
        (558, (0, 0)),
        (600, (500, 500)),
        (650, (0, 0)),
        (700, (500, -500)),
        (808, (0, 0))
    ])
    state["end"] = max(state["prog"].keys()) + 10

    def dispatch(evtid, evt_name, evt_args):
        # https://www.thymio.org/en:thymioapi motor freq is 100Hz
        if evt_name == "fwd.motor": # every 0.01 sec
            if state["time"] in state["prog"]:
                (l,r) = state["prog"][(state["time"])]
                Thym.set('motor.left.target', [l])
                Thym.set('motor.right.target', [r])
            state["time"] += 1
            if state["time"] == state["end"]:
                Thym.stop()
```

```
    else: # What?
        print evt_name

# Now lets start the loopy thing
Thym.loop(dispatch)
print "Au revoir"
```

1. Testez le programme sur le Thymio.
2. Que signifie `(500,500)`, `(-500,500)`, `(500,-500)`, `(0,0)` ?
3. À quelle vitesse avance le robot lorsque la valeur roue gauche et droite est `200`, lorsque la valeur est `100` ? Pour cela, ne conservez du programme précédent que les 2 premières lignes de la valeur `"prog"` du dictionnaire, et testez selon les valeurs que vous mettez à la place de `(500,500)`. Attention, la vitesse de votre robot va dépendre de la charge de sa batterie.
4. Combien de temps faut-il pour que le robot tourne d'un quart lorsque la valeur roue gauche est `200` et roue droite est `-200` ? Pour cela, ne conservez du programme précédent que les lignes 3 et 4 de la valeur `"prog"` du dictionnaire, et testez selon le temps que vous laissez.
5. En modifiant le programme précédent, écrire un programme qui se déplace en suivant le bord d'un carré de 10cm de côté. Attention : tenez compte du fait que le Thymio a une certaine surface !

B : Fonctions de déplacement

La structure du programme dans la section précédente n'est pas satisfaisante : le plan intègre des temps absolus et non relatifs, il n'y a pas de modularité dans la structure du code. Cette section doit y remédier.

1. Créez une fonction `motorisation(l,r)` qui met la valeur de motorisation (la vitesse) de la roue gauche à `l` et de la roue droite à `r`.
2. Créez une fonction `arret()` qui arrête la motorisation.

On suppose maintenant que votre programme utilise un dictionnaire `state` dont une clé est `"vitesse"` déterminant une valeur de motorisation (on supposera que les valeurs de motorisations sont soit `0`, soit `state["vitesse"]`, soit `-state["vitesse"]`). En vous servant de la fonction `motorisation(l,r)` et de `state["vitesse"]` :

3. Créez une fonction `avancer()` qui fait avancer tout droit le robot.
4. Créez une fonction `tournerDroite()` qui fait tourner le robot (continûment).
5. Créez une fonction `tournerGauche()` qui fait tourner le robot (continûment).

On veut maintenant définir le plan d'action pour le robot, c'est-à-dire la suite d'actions que doit faire le robot. Concrètement un plan est une liste de paires de valeurs. Une paire `(d,a)` est composée d'une durée (un entier, durée en centièmes de seconde) et d'une action (un nom de fonction, par exemple `avancer`). Le plan sera le contenu de la clé `"plan"` du dictionnaire `state`.

6. Créez une fonction `plan()` qui initialise la clé `"plan"` pour que le robot avance d'une seconde puis tourne à gauche pendant 1 seconde puis avance d'une seconde.
7. Écrivez un programme et testez le plan précédent. Votre programme doit conserver la date de fin de l'action comme valeur d'une clé `"finAction"` du dictionnaire `state`, cette valeur est la somme de la date de déclenchement de l'action + la durée de l'action. Il doit appeler une fonction `dispatch()`. Cette fonction `dispatch()` doit récupérer les événements `"fwd.motor"` (qui arrivent tous les centièmes de seconde). Elle incrémente le temps à chaque événement, quand le temps arrive à la durée prévue, elle déclenche l'action suivante s'il y en a, sinon elle stoppe le Thymio.