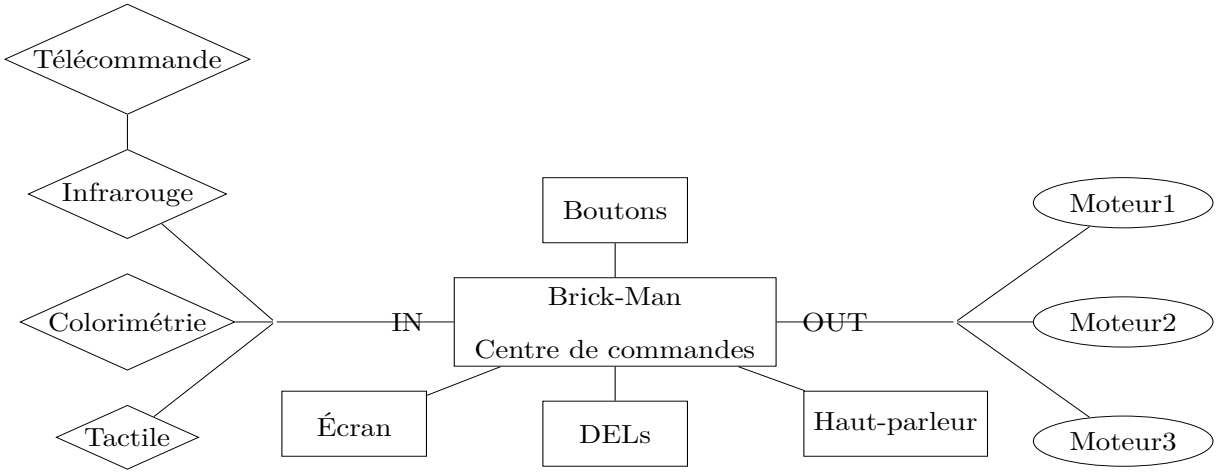
Ce TP prolonge l'introduction au robot faite dans le TP 7.
Les périphériques du robot sont élargies. On rappelle le lien [https://github.com/rhempel/ev3dev-lang-python].

I Le robot et ses périphéries

L'ensemble des périphéries du robot se compose d'un centre de commandes (le brick-man), d'entrées (ici des capteurs, une télécommande (via le capteur infrarouge) ou des boutons), de sorties (ici des moteurs, des DELs, un écran ou un haut-parleur). Certains périphériques sont intrasèquement reliés au brick-man (entourés par des rectangles), d'autres se relient par un câble.

On propose le schéma récapitulatif suivant :



II La programmation du robot sous python

Concernant la connexion au robot, se reporter au TP 7.

On vous rappelle qu'il faut avancer PAR ÉTAPES :
◇ tester une seule nouvelle fonctionnalité à la fois
◇ une fois testée et validée, SAUVEGARDER le fichier dans une COPIE numérotant l'avancement.

Votre fichier `.py` commencera par :

```
#!/usr/bin/env python3
import time, random
import ev3dev.ev3 as ev3    # la librairie du brick-man
```

D'une manière générale, pour appliquer une fonction a un élément du robot (entrée, sortie ou brick-man), on doit attacher cette fonction à l'élément (*e.g.* : la fonction `speak` fonctionne avec l'haut-parleur ici nommé `hp`. Cela donnera `hp.speak('Hello world')`).

rq : si vous voulez connaître toutes les fonctions possibles, il faut principalement lire le fichier `core.py` de la librairie `ev3dev` (∼3 000 lignes).

1 Haut-parleur

Le haut-parleur fonction à partir du brick-man sans connectique.

```
hp = ev3.Sound()            # defini la variable hp comme un haut-parleur
hp.tone([ (659, 460), (784, 340) ]).wait()   # fait des bips
hp.speak('Hello world')    # parle et dit 'Hello world'
hp.play('file.wav')        # joue le fichier file.wav
```

Exo 1 : Dans un fichier `test-01.py`, programmer le robot et faites-lui dire un message.

2 Moteurs

Les moteurs doivent être connectés au brick-man dont on doit préciser le port de sortie. Le petit moteur a une vitesse maximale de 1 560 tandis que les gros moteur de 1 050. La vitesse peut être négative ou positive tandis que la position peut être définie de manière absolue ou de manière relative à la position du moment.

```
mm = ev3.MediumMotor('outA')   # defini un petit moteur en sortie 'outA'
md = ev3.LargeMotor('outB')    # defini un moteur
md.stop()              # arret du moteur
md.count_per_rot       # renvoie le nombre du tacho par tour
md.position            # renvoie la position actuelle
md.position = 0        # defini la position actuelle comme 0
md.run_to_abs_pos(speed_sp=900, position_sp=1080)      # rotation absolue
md.run_to_rel_pos(speed_sp=1200, position_sp=-200)     # rotation relative
md.run_timed(time_sp=3000, speed_sp=500)   # rotation temporelle (en ms)
md.run_forever(speed_sp=-800)              # rotation infinie
```

Exo 2 : Dans un fichier `test-02.py`, programmer le robot afin qu'il avance tout droit durant 3 secondes puis qu'il s'arrête, qu'il tourne d'un quart de tour, s'arrête et dit un message.

3 Senseur tactile

```
ts = ev3.TouchSensor('in1')   # defini un senseur tactile a l'entree 'in1'
ts.is_pressed                 # renvoie 1 si le senseur est presse
```

Exo 3 : Dans un fichier `test-03.py`, programmer le robot afin qu'il ne fasse rien tant que le senseur tactile n'a pas été pressé.

4 Senseur infrarouge

Le senseur infrarouge possède deux modes : l'un ('IR-PROX') qui permet de quantifier la proximité du senseur avec un objet en face (selon les essais, la détection maximale est d'environ 700 mm) ; l'autre ('IR-REMOTE') qui permet de prendre en compte la télécommande.

```
irs = ev3.InfraredSensor('in2') # defini un senseur infrarouge a l'entree 'in2'
irs.proximity                   # renvoie un pourcentage de proximite (100%=70cm)
irs.mode = 'IR-PROX'            # met le senseur en mode proximite
irs.mode = 'IR-REMOTE'          # met le senseur en mode telecommande
```

Exo 4 : Dans un fichier `test-04.py`, programmer le robot afin qu'une fois qu'il avance, il s'arrête et fait un quart de tour s'il se rapproche à moins de 300mm d'un objet.

5 Télécommande

La télécommande possède 4 bande d'émission (4 channels). On définit alors une variable télécommande selon chacun des channels en la rattachant au senseur infrarouge préalablement défini.

Chaque channel possède 5 boutons (`red_up, red_down, blue_up, blue_down, beacon`).

Afin de pouvoir assigner une action lorqu'un bouton de la télécommande est appuyé ou relâché, le fonctionnement de la télécommande est légèrement différent du reste. Une fonction d'action (ici `tourne()`) est assignée au bouton voulu. Puis une fonction `on_press()` va être définie avec les actions souhaitées à la pression du bouton.

```
rc1 = ev3.RemoteControl(sensor=irs, channel=1)   # defini une telecommande selon le
↪  channel 1 rattachee au senseur irs
def tourne(motor, speed):
.   def on_press(state):
      if state:  # tourne si le bouton est presse
         motor.run_forever(speed_sp=speed)
      else:      # stop si le bouton est relache
         motor.stop()
   return on_press
rc1.on_blue_up = tourne(md, 900)   # ou md est une variable moteur
# utilisation dans le corps du programme :
rc1.process()
```

Exo 5 : Dans un fichier `test-05.py`, programmer le robot afin que l'utilisateur puisse commander sa motricité (marche avant, marche arrière, tourner).

6 Senseur colorimétrique

Le senseur colorimétrique est capable de percevoir une luminosité, de reconnaître une couleur ou de renvoyer le code RGB d'une couleur. Il possède également plusieurs mode mais en première utilisation le senseur gère automatique le mode dans lequel il doit se mettre.

Pour la reconnaissance des couleurs, le senseur doit être relativement proche du support (∼10mm). Le code couleur est un chiffre de 0 à 7 (aucune, noir, bleu, vert, jaune, rouge, blanc, marron). Lorsque le code RGB est renvoyé, 3 valeurs comprises entre 0 et 1020 sont renvoyées.

```
cs = ev3.ColorSensor('in3')      # defini 1 senseur colorimetrique a l'entree 'in3'
cs.reflected_light_intensity     # renvoie une couleur rouge
cs.ambient_light_intensity       # renvoie une couleur bleue
cs.color                         # renvoie le code couleur de 0 a 7
cs.raw                           # renvoie le code RGB
```

7 DELs du brick-man

Il existe 2 DELs : 'LEFT' et 'RIGHT' dont on peut définir les couleurs : BLACK (éteinte), AMBER (orange), RED ou GREEN. Les attributs 'LEFT', 'RIGHT', 'BLACK' etc. doivent être reliés à l'élément 'ev3.Leds' soit explicitement par 'ev3.Leds.RIGHT' soit comme ci-dessous.

```
led = ev3.Leds()                                  # defini la variable led comme des DELs
led.set_color(group=led.LEFT,color=led.BLACK) # defini la couleur de la DEL de gauche
```

8 Écran du brick-man

Pour dessiner sur l'écran, on utilise les fonctions de la librairie PIL. Pour une collection des fonctions possibles, on peut se reporter au site [http://www.effbot.org/imagingbook/imagedraw.htm].

```
ec = ev3.Screen()                # defini la variable ec comme un ecran
ec.clear()                       # efface l'ecran
ec.draw.text((70,5),'Bienvenue')# ecrit 'Bienvenue' a partir du point (70;5)
ec.draw.arc((1,2,9,7),-21,0)     # dessine un arc de cercle dans le rectangle (1;2)-(9;7)
↪  de -21 degres a 0 degre.
ec.draw.line((10,50,90,90))      # dessine le segment (10;50)-(90;90)
ec.update()                      # affiche le dessin
```

9 Boutons du brick-man

Il y a 5 boutons sur le brick-man, nommés `up, down, right, left, enter, backspace`. Ce qui est possible pour le bouton `up` fonctionne pour les autres.

```
bt = ev3.Button()       # defini la variable bt comme des boutons
bt.any()                # renvoie vrai si un des boutons est presse
bt.buttons_pressed      # renvoie la liste des boutons presses
bt.up                   # renvoie vrai si le bouton up est presse
```

rq : ce qui a été fait pour les boutons de la télécommande est possible pour les boutons du brick-man via la fonction `.on_up()`.

10 Autres

Enfin, voici quelques fontions pour utiliser l'heure (attente / chronomètre) ou effectuer un choix aléatoire.

```
t = time.time()               # renvoie l'heure machine
time.sleep(0.1)               # attente de 100 ms
random.seed( time.time() )    # reinitialise le generateur aleatoire
random.choice( (-10,0,70) )   # choix au hasard entre les valeurs -10, 0 et 70
```