




## FISE 1A - UE 2.2 - Boucle Capteur Actionneur

1. Quels fichiers modifier ? `qualif1.py` et `control.py`
2. Quels fichiers à ne pas modifier ? `rob1a_v02.py`
3. Quels fichiers à modifier éventuellement (si filtrage) ? `filt.py`

rb : objet robot  1 seul robot  capteurs  
ctrl : objet contrôle : toutes les fonctions de contrôle  actionneurs  
(voir sujet BE)

```
1 import robia_v02 as robia # get robot simulator
2 import control # robot control functions
3
4 if __name__ == "__main__":
5     rb = robia.Rob1A() # create a robot (instance of Rob1A class)
6     ctrl = control.RobotControl() # create a robot controller
7
8     # to do :
9     # straight move until the front sonar gives a distance
10    # smaller than 0.3 m
11
12    rb.full_end() # clean end of the robot's simulation
```

mission  
à réaliser

définir les paramètres → vitesse  
→ distance d'arrêt

```
1 import robia_v02 as robia # get robot simulator
2 import control # robot control functions
3
4 if __name__ == "__main__":
5     rb = robia.Rob1A() # create a robot (instance of Rob1A class)
6     ctrl = control.RobotControl() # create a robot controller
7
8     spd = 120
9     dist_obstacle = 0.25
10
11
12
13     rb.full_end() # clean end of the robot's simulation
```

définir la fonction de contrôle  
et passer les paramètres utiles

```
1 import robia_v02 as robia # get robot simulator
2 import control # robot control functions
3
4 if __name__ == "__main__":
5     rb = robia.RobiaA() # create a robot (instance of RobiaA class)
6     ctrl = control.RobotControl() # create a robot controller
7
8     spd = 120
9     dist_obstacle = 0.4
10    ctrl.go_straight_stop_on_obstacle (rb, spd, dist_obstacle)
11
12
13    rb.full_end() # clean end of the robot's simulation
```

robot  
vitesse  
distance d'arrêt

définir la fonction de contrôle :

aller tout droit jusqu'à un obstacle

```
1 import time
2
3 class RobotControl:
4     def __init__(self):
5         self.distBetweenWheels = 0.12
6         self.nTicksPerRevol = 512
7         self.wheelDiameter = 0.06
8
9
10     def go_straight_stop_on_obstacle (self, rb, spd, dist_obstacle):
11         loop_iteration_time = 0.2 # 5 Hz
```

constantes utiles  
pour les odomètres

définir le temp d'itération

200ms (5Hz) : 5 commandes par secondes

cela ne sert à rien de commander plus vite!

nouvelle mesure  
Sonar  
toutes les 200ms

```
1 import time
2
3 class RobotControl:
4     def __init__(self):
5         self.distBetweenWheels = 0.12
6         self.nTicksPerRevol = 512
7         self.wheelDiameter = 0.06
8
9
10    def go_straight_stop_on_obstacle (self,rb,spd,dist_obstacle):
11        loop_iteration_time = 0.2 # 5 Hz
12        rb.set_speed(spd,spd)
```

on fait avancer le robot  
en ligne droite  
 $spd_l = spd_r = spd$

```
1 import time
2
3 class RobotControl:
4     def __init__(self):
5         self.distBetweenWheels = 0.12
6         self.nTicksPerRevol = 512
7         self.wheelDiameter = 0.06
8
9
10    def go_straight_stop_on_obstacle (self,rb,spd,dist_obstacle):
11        loop_iteration_time = 0.2 # 5 Hz
12        rb.set_speed(spd,spd)
13        while True:
14            t0_loop = time.time()
```

\_\_\_\_\_ o boucle de contrôle infinie  
↳ mesure du temps en début d'itération



```
1 import time
2
3 class RobotControl:
4     def __init__(self):
5         self.distBetweenWheels = 0.12
6         self.nTicksPerRevol = 512
7         self.wheelDiameter = 0.06
8
9
10    def go_straight_stop_on_obstacle (self, rb, spd, dist_obstacle):
11        loop_iteration_time = 0.2 # 5 Hz
12        rb.set_speed(spd, spd)
13        while True:
14            t0_loop = time.time()
15
16            dist_front = rb.get_sonar('front')
17            if dist_front > 0 and dist_front < dist_obstacle:
18                rb.set_speed(0,0)
19                break
```

mesure non filtrée  
(raw data)  
du Sonar Frontal

arrêt des moteurs

Sortie de la boucle infinie

Si mur à moins de dist\_obstacle

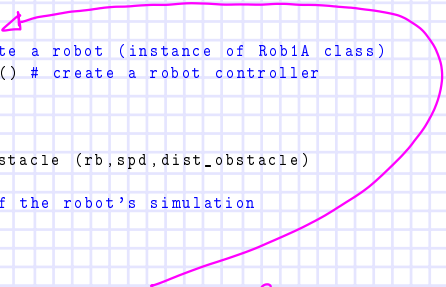
```
1 import time
2
3 class RobotControl:
4     def __init__(self):
5         self.distBetweenWheels = 0.12
6         self.nTicksPerRevol = 512
7         self.wheelDiameter = 0.06
8
9
10    def go_straight_stop_on_obstacle (self,rb,spd,dist_obstacle):
11        loop_iteration_time = 0.2 # 5 Hz
12        rb.set_speed(spd,spd)
13        while True:
14            t0_loop = time.time()
15
16            dist_front = rb.get_sonar('front')
17            if dist_front > 0 and dist_front < dist_obstacle:
18                rb.set_speed(0,0)
19                break
20
21            t1_loop = time.time()
22            dt_loop = t1_loop - t0_loop
23            dt_sleep = loop_iteration_time - dt_loop
24            if dt_sleep > 0:
25                time.sleep(dt_sleep)
```

*Annotations manuscrites :*

- Sur la ligne 23, une flèche pointe de `dt_sleep` vers le commentaire "attente pour que l'itération dure 200 ms".
- Sur la ligne 25, une flèche pointe de `time.sleep(dt_sleep)` vers le commentaire "attente pour que l'itération dure 200 ms".
- À droite, une note en vert : "calcul du temps d'exécution du code" avec une accolade englobant les lignes 21 à 23.
- En bas à gauche, une note en rose : "si  $dt\_sleep < 0 \rightarrow pb !!$  trop de calculs !!".

- ▶ Rendre sous MOODLE dans le devoir Qualif1 : `qualif1.py` et `control.py`
- ▶ Rendre sous MOODLE dans le devoir Qualif1 `filt.py` si vous avez utilisé des filtres (en principe non)
- ▶ Attendre la publication des résultats dans les tableaux MOODLE

```
1 import rob1a_v02 as rob1a # get robot simulator
2 import control # robot control functions
3
4 if __name__ == "__main__":
5     pseudo = "Lightning_Buzz"
6     rb = rob1a.Rob1A() # create a robot (instance of Rob1A class)
7     ctrl = control.RobotControl() # create a robot controller
8
9     spd = 120
10    dist_obstacle = 0.25
11    ctrl.go_straight_stop_on_obstacle (rb,spd,dist_obstacle)
12
13    rb.full_end() # clean end of the robot's simulation
```



définition d'un pseudo pour affichage des  
résultats sur ROOBLE

1. Quels fichiers modifier ? `qualif2.py` et `control.py`
2. Quels fichiers à ne pas modifier ? `rob1a_v02.py`
3. Quels fichiers à modifier éventuellement (si filtrage) ? `filt.py`

1. Ecrire une fonction de contrôle qui recherche la direction ou aller (sonar ne détecte pas de murs proches)
2. Ecrire une fonction de contrôle qui fait tourner le robot de 90 degrés (à droite ou à gauche) avec la boussole ou les odomètres
3. Réutiliser la fonction de déplacement en ligne droite de qualif1