```python
from easygopigo3 import EasyGoPiGo3
import socket
import time
import os
from math import *
import threading
from threading import Thread, Lock, Event

#pour les sockets et la communication avec le serveur
Adresse = "192.168.0.2"
Port = 12345


#initialisation des coordonnées initiales utilisées par la suite pour optimiser la fonction orientation()
xnext,ynext=0,0

#Ici on definit un prefixe plus court que EasyGoPiGo3 pour notre programme
gpg = EasyGoPiGo3()

#premièrement on initialise le sensor de distance du robot et le moteur pour faire tourner le capteur de distance
#initialisation de la variable problem_capt si probème de détection du capteur
distance = gpg.init_distance_sensor()

servo=gpg.init_servo("SERVO1")
servo.reset_servo()

#ensuite on va definir pour notre programme une distance de sécurité qui si elle est dépassée fera s'arreter le robot
safe_distance=120

#on fixe la vitesse du robot
gpg.set_speed(400)


########Programmes récupération données du routeur#######

def GetTargetPosition():
    serveur = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
38      serveur.connect((Adresse,Port))
39      GTP = bytes("GTP","utf-8")
40      serveur.send(GTP)
41
42      f = serveur.recv(100)
43      f = f.decode()
44
45      f = f.split()
46      f[0] =f[0][0:len(f[0])-1]
47      f[0] = float(f[0])
48      f[1] = float(f[1])
49      return f
50
51  def GetBluePosition():
52      serveur = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
53      serveur.connect((Adresse,Port))
54      GBP = bytes("GBP","utf-8")
55      serveur.send(GBP)
56
57      f = serveur.recv(100)
58      f = f.decode()
59
60      f = f.split()
61      f[0] =f[0][0:len(f[0])-1]
62      f[0] = float(f[0])
63      f[1] = float(f[1])
64      return f
65
66  def GetGreenPosition():
67      serveur = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
68      serveur.connect((Adresse,Port))
69      GGP = bytes("GGP","utf-8")
70      serveur.send(GGP)
71
72      f = serveur.recv(100)
73      f = f.decode()
74
75      f = f.split()
76      f[0] =f[0][0:len(f[0])-1]
77      f[0] = float(f[0])
78      f[1] = float(f[1])
79      return f
```

```
80
81  def GetStarted():
82      serveur = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
83      serveur.connect((Adresse,Port))
84      GBS = bytes("GBS","utf-8")
85      serveur.send(GBS)
86
87      f = serveur.recv(100)
88      f = f.decode()
89
90      return int(f)
91
92
93
94  def orientation():
95      global xnext,ynext
96      #ici si c les premiers pas du robot on calcule une première
        ↪  fois les deux vecteurs pour le calcul
97      #sinon on utilise la position précédente pour éviter de
        ↪  réavancer de 10 cm à chaque fois et ainsi perdre du temps
98      if xnext==0 and ynext==0:
99          x0,y0=GetBluePosition()[0],GetBluePosition()[1]
100         gpg.drive_cm(10)
101         x1,y1=GetBluePosition()[0],GetBluePosition()[1]
102         xt,yt=GetTargetPosition()[0],GetTargetPosition()[1]
103     else:
104         x0,y0=xnext,ynext
105         x1,y1=GetBluePosition()[0],GetBluePosition()[1]
106         xt,yt=GetTargetPosition()[0],GetTargetPosition()[1]
107
108     v1,v2=(x1-x0),(y1-y0)
109     w1,w2=(xt-x1),(yt-y1)
110     teta2_rad=atan2((w2*v1-w1*v2),(w1*v1+w2*v2))#angle entre le
        ↪  vecteur v(vecteur robot) vers le vecteur w (vecteur cible)
111     teta2=((180/pi)*teta2_rad)
112
113     xnext,ynext=x1,y1
114     gpg.turn_degrees(teta2*1.02)#on prend en compte les
        ↪  imprécisions mécaniques (moteurs) en appliquant un coeff
        ↪  multiplicatuer de 1,02
115
116
```

```
117 #######Programme mode threading#########
118
119 event=Event()
120
121 def jeu():
122     global xtcible, ytcible
123     gpg.set_eye_color((0,0,255))#allumer notre robot avec du bleu
        ↪ pour la détection via la webcam
124     gpg.open_eyes()
125     if GetStarted()==1:
126
            ↪ xtcible,ytcible=GetTargetPosition()[0],GetTargetPosition()[1]
127         print(cible.is_alive(),
            ↪ evitement.is_alive(),ringthread.is_alive())
128         if not cible.is_alive() or not evitement.is_alive() or not
            ↪ ringthread.is_alive():#si il y a un problème avec un
            ↪ des threads on passe alors en mode itératif avec la
            ↪ fonction de secours
129             if not evitement.is_alive():#si le problème concerne le
                ↪ capteur de distance
130                 probleme_capt=1
131             jeu_secours()
132         while True:
133             if event.is_set():
134                 break
135             orientation()
136             gpg.drive_cm(40)
137
138
139
140 xtcible,ytcible=GetTargetPosition()[0],GetTargetPosition()[1]
141
142 def changer_cible(x=xtcible,y=ytcible):
143     global xtcible,ytcible,xnext,ynext
144     time.sleep(0.1)
145     if GetStarted()==1:
146         time.sleep(0.15)
147         xt2,yt2=GetTargetPosition()[0],GetTargetPosition()[1]
148         time.sleep(0.05)
149         if xt2!=x or yt2!=y:#on vérifie à chaque instant que la
            ↪ cible n'a pas bougé d'endroit
150             print("cible")
```

```
151             event.set()
152             time.sleep(0.3)
153             event.clear()
154             with lock:
155                 gpg.stop()
156 
                    ↪  xtcible,ytcible=GetTargetPosition()[0],GetTargetPosition()[1]
157                 xnext,ynext=0,0
158                 jouer=threading.Thread(target=jeu)
159                 jouer.start()
160             changer_cible(xtcible,ytcible)
161         else:
162             time.sleep (0.2)
163             changer_cible(xt2,yt2)
164 
165 
166 #initialisation de la variable problem_capt si problème de
    ↪  détection du capteur
167 probleme_capt=0
168 
169 def manoeuvre_evitement():
170     global probleme_capt, xnext,ynext
171     if GetStarted()==1:
172         dist=distance.read_mm()
173 
174         #si le capteur ne fonctionne pas on s'en passe
175         if dist==type(None) and (probleme_capt==0):
176             probleme_capt=1
177             print("problem capt")
178 
179         xb,yb=GetBluePosition()[0],GetBluePosition()[1]
180         xg,yg=GetGreenPosition()[0],GetGreenPosition()[1]
181 
182     #on veut éviter les collisions entre robots.
183         if -120+xb<xg<120+xb and -120+yb<yg<120+yb and
            ↪  robot_presence==1:
184             print("autre robot")
185             event.set()
186             time.sleep(0.3)
187             with lock:
188                 gpg.drive_cm(-5)
189                 time.sleep(0.5)
```

```
190                 gpg.turn_degrees(90)
191                 xnext,ynext=0,0
192                 time.sleep(0.5)
193                 event.clear()
194                 jouer=threading.Thread(target=jeu)
195                 jouer.start()
196             manoeuvre_evitement()
197 
198     #si on est proche d'un obstacle
199         elif dist<safe_distance and probleme_capt==0:
200             print('passé par là')
201             gpg.stop()
202             event.set()
203             time.sleep(0.2)
204             with lock:
205                 gpg.stop()
206                 servo.rotate_servo(50)#on fait bouger le capteur de
                    ↪  distance à droite
207                 time.sleep(0.2)
208                 alpha=distance.read_mm()
209                 time.sleep(0.3)
210                 servo.rotate_servo(130)#on fait bouger le capteur
                    ↪  de distance à gauche
211                 time.sleep(0.2)
212                 beta=distance.read_mm()
213                 time.sleep(0.3)
214                 servo.rotate_servo(90)#on remet le capteur à sa
                    ↪  position initiale
215                 time.sleep(0.5)
216                 servo.reset_servo()
217                 if beta>alpha: #on choisit de trourner vers
                    ↪  l'obstacle le plus loin
218                     gpg.turn_degrees(-90)
219                     gpg.drive_cm(30)
220                 else:
221                     gpg.turn_degrees(90)
222                     gpg.drive_cm(30)
223 
224                 time.sleep(0.5)
225                 event.clear()
226                 jouer=threading.Thread(target=jeu)
227                 jouer.start()
```

```
228             manoeuvre_evitement()
229         else:
230             time.sleep(0.2)
231             manoeuvre_evitement()
232 
233 
234 def ring():#On fait en sorte que le robot ne sorte pas du ring
235     if GetSarted()==1:
236         xb,yb=GetBluePosition()[0],GetBluePosition()[1]
237         if xb>1180 or yb >750 or xb<46 or yb<46:#utilisation de
        ↪  mesures faites sur le ring projeté sur le sol
238             print("ring")
239             event.set()
240             time.sleep(0.3)
241             with lock:
242                 gpg.turn_degrees(180)
243                 time.sleep(0.15)
244                 gpg.drive_cm(40)
245                 time.sleep(0.5)
246                 event.clear()
247                 jouer=threading.Thread(target=jeu)
248                 jouer.start()
249             ring()
250         else:
251             time.sleep(0.2)
252             ring()
253 
254 ########Programme de secours##############
255 
256 def jeu_secours():
257     global probleme_capt,xnext,ynext
258     print("fonctions de secours")
259     xnext,ynext=0,0
260     event.set()
261     with lock:
262         while GetStarted()==1:
263             orientation()
264             gpg.drive_cm(5)
265             if probleme_capt==0:
266                 dist = distance.read_mm()
267                 if dist < safe_distance:
268                     evitement_objets_version_secours()
```

```
269
270
271 def evitement_objets_version_secours():
272         gpg.stop()
273         servo.rotate_servo(50)
274         time.sleep(0.2)
275         alpha=distance.read_mm()
276         time.sleep(0.3)
277         servo.rotate_servo(130)
278         time.sleep(0.2)
279         beta=distance.read_mm()
280         time.sleep(0.3)
281         servo.rotate_servo(90)
282         time.sleep(0.5)
283         print('passé par là')
284         servo.reset_servo()
285         if beta > alpha:
286             gpg.turn_degrees(-90)
287             gpg.drive_cm(10)
288         else:
289             gpg.turn_degrees(90)
290             gpg.drive_cm(10)
291
292
293 robot_presence=int(input("Eviter les collisions avec le robot
    ↪  adverse? Si oui tapez 1 sinon 0"))
294
295 lock=Lock()
296
297 def Wait_start():#on attend que le programme nous dises de démarrer
298     while True:
299         if GetStarted()==1:
300             break
301 Wait_start()
302
303 cible=threading.Thread(target=changer_cible)
304 cible.start()
305
306 evitement=threading.Thread(target= manoeuvre_evitement)
307 evitement.start()
308
309 ringthread=threading.Thread(target=ring)
```

```
310 ringthread.start()
311
312 jouer=threading.Thread(target=jeu)
313 jouer.start()
314
315
```