Annexos

Sumari

Annex A. 02_main.py

```python
#!/usr/bin/env python3
import threading
import time
from ev3dev2.motor import LargeMotor, OUTPUT_A, OUTPUT_B, OUTPUT_C
import math
import os
os.system('setfont Lat15-TerminusBold14')

temps_inic = time.time() # instant en que s inicia el programa

intern= open("03_dades_moviment.txt","w")
f = open("01_valors.txt","r")

# calcula el valor on es produeix el pic d acceleracio
def valorumax(jerk):
        if 0 in [jerk[0],jerk[2]]:
            if jerk.index(0)==0:
                return -jerk[2]/jerk[1]
            if jerk.index(0)==2:
                return-jerk[1]/jerk[0]
        else:
            umes = (-jerk[1]+math.sqrt(jerk[1]**2-4*jerk[0]*jerk[2]))/(2*jerk[0])
            umenys = (-jerk[1]-math.sqrt(jerk[1]**2-4*jerk[0]*jerk[2]))/(2*jerk[0])
            if umes<1 and umes>0:
                return umes
            if umenys<1 and umenys>0:
                return umenys

# donats un valor de u i un polinomi, evalua el polinomi en u
def valorb(u,polx):
    grau=len(polx)-1
    suma = 0
    for i in polx:
        suma += i*u**grau
        grau -=1
    return suma

# retorna el signe d un vector
def signe(posini,posfin):
        if posfin == posini:
                return 0
        if posfin > posini:
                return 1
        else:
                return -1

# retorna l angle d un vector
def donam_angle(pos_inic,pos_final):

    x_signe = signe(pos_inic[0],pos_final[0])
    y_signe = signe(pos_inic[1],pos_final[1])

    if x_signe == 0:
        the = y_signe*math.pi/2
    elif y_signe == 0:
        the = math.pi/2 -x_signe*math.pi/2
    else:
        if x_signe == y_signe:
            the = math.atan(abs(pos_final[1]-pos_inic[1])/abs(pos_final[0]-
pos_inic[0])) +(math.pi/2 -x_signe*math.pi/2)
```

```
60          else:
61              the = math.atan(abs(pos_final[1]-pos_inic[1])/abs(pos_final[0]-
pos_inic[0])) +(math.pi + x_signe*math.pi/2)
62
63      return the
64
65  # Dades inicials
66
67  intern.write("Dades del moviment: \r\n")
68  intern.write("..................................... \r\n")
69  theta1 = 0      # inicialitza la vel. angular del motor 1
70  theta2 = 0      # inicialitza la vel. angular del motor 2
71  theta3 = 0      # inicialitza la vel. angular del motor 3
72  radi_roda = float(f.readline().replace("\n","").replace("
","").replace("diametre_roda=",""))/2 #llegeix la línia on s introdueix el diametre
73  long_eix = float(f.readline().replace("\n","").replace("
","").replace("long_eix=","")) # llegeix la línia on s introdueix longitud de l eix
74  alpha = 30*math.pi/180 # angle que formen els eixos de les rodes 2 i 3 amb l'eix 1':
pi/6
75  intern.write("radi_roda: "+str("{:.2f}".format(radi_roda))+' mm \r\n')
76  intern.write("longitud_eix: "+str("{:.2f}".format(long_eix))+' mm \r\n')
77  intern.write("..................................... \r\n")
78
79  thetamax = 175 # rpm maximes per construccio
80  thetamax_rad = thetamax*2*math.pi/60 # velocitat angular maxima en rad/s
81  velmax_centre = thetamax_rad*radi_roda # velocitat maxima del centre de la roda mm/s
82  conversio = 100/thetamax_rad # parametre per convertir velocitat en percentatge
83  xi_1 = 0.7 #
84  xi_2 = 1-xi_1
85  pol_vel1 = [3*(2*xi_1-1),4*(1-3*xi_1),6*xi_1,0,0]
86  pol_pos1 = [3*(2*xi_1-1)/5,(1-3*xi_1),2*xi_1,0,0,0]
87  pol_acc1 = [4*3*(2*xi_1-1),3*4*(1-3*xi_1),2*6*xi_1,0]
88  pol_jerk1 = [3*4*3*(2*xi_1-1),2*3*4*(1-3*xi_1),2*6*xi_1]
89  pol_vel2 = [3*(2*xi_2-1),4*(1-3*xi_2),6*xi_2,0,0]
90  pol_pos2 = [3*(2*xi_2-1)/5,(1-3*xi_2),2*xi_2,0,0,0]
91  pol_acc2 = [4*3*(2*xi_2-1),3*4*(1-3*xi_2),2*6*xi_2,0]
92  pol_jerk2 = [3*4*3*(2*xi_2-1),2*3*4*(1-3*xi_2),2*6*xi_2]
93
94  u_max = valorumax(pol_jerk1)
95  epsa_max = valorb(u_max,pol_acc1)
96  epsd_max = valorb(1,pol_pos1)
97
98  running = True
99  # Aquesta part modifica la class Thread.
100
101 class calibra(threading.Thread):
102
103         def __init__(self):
104                 self.m1 = LargeMotor(OUTPUT_A)
105                 self.m2 = LargeMotor(OUTPUT_B)
106                 self.m3 = LargeMotor(OUTPUT_C)
107
108                 threading.Thread.__init__(self)
109
110                 motors=[self.m1, self.m2, self.m3]
111
112                 for m in motors:
113                         m.reset()
114                         m.position = 0
115                         m.stop_action = 'brake'
```

```
116
117
118         def run(self):
119
120                 print("Engines running!")
121
122                 while running:
123
124                         self.m1.on(theta1*conversio)
125                         self.m2.on(theta2*conversio)
126                         self.m3.on(theta3*conversio)
127
128                 t11= time.time()
129                 self.m1.stop()
130                 t12= time.time()
131                 self.m2.stop()
132                 t23 = time.time()
133                 self.m3.stop()
134                 t33= time.time()
135                 intern.write("temps frenada m3: "+str("{:.3f}".format(t33-t23))+" s
\r\n")
136                 intern.write("temps frenada m2: "+str("{:.3f}".format(t23-t12))+" s
\r\n")
137                 intern.write("temps frenada m1: "+str("{:.3f}".format(t12-t11))+" s
\r\n")
138                 intern.write("temps total de frenada: "+str("{:.3f}".format(t33-
t11))+" s \r\n")
139                 intern.write("---------------------------------------- \r\n")
140
141 if __name__ == "__main__":
142
143         motor_thread = calibra()
144         motor_thread.setDaemon(True)
145         motor_thread.start()
146
147
148 # Cos del programa
149
150 linia = f.readline()
151 temps_control = time.time()
152 tram = 1
153
154 while linia != '':
155         temps_mov = 0
156         temps_act = time.time()
157         rest_time = 0.02
158         rotrans = False
159
160         if "%" in linia:
161                 pass
162
163         elif "[" in linia:
164                 rest_time =  float(linia.replace('\n','').replace(" ","").strip("
[]"))
165                 intern.write("atura motors "+str("{:.3f}".format(rest_time))+" s
\r\n")
166                 intern.write("..................................... \r\n")
167
168         else:
169                 temps_act = time.time()
```

```
170                 l = linia.replace('\n','').replace(" ","").split(";")
171                 vel_max = float(l[2])
172                 psi_max = float(l[3])
173                 mov_possible = False
174 
175                 if abs(vel_max) < velmax_centre and abs(psi_max*long_eix) <
velmax_centre:
176                         mov_possible = True
177 
178                 conf_inic = [float(vel) for vel in l[0].strip('()').split(',')]
179                 conf_final = [float(vel) for vel in l[1].strip('()').split(',')]
180                 desp_mov = math.sqrt((conf_final[0]-conf_inic[0])**2+(conf_final[1]-
conf_inic[1])**2)
181 
182                 if vel_max != 0:
183                         a_max = (velmax_centre-abs(vel_max))
184                         t_acc = epsa_max*abs(vel_max)/a_max
185                         s_max = epsd_max*abs(vel_max)*t_acc
186                         ang_max = epsd_max*abs(psi_max)*t_acc
187                         t_const = (desp_mov - 2*s_max)/abs(vel_max)
188 
189                 else:
190                         a_max = (velmax_centre-abs(psi_max)*long_eix)
191                         t_acc = epsa_max*abs(psi_max)/a_max
192                         ang_max = epsd_max*abs(psi_max)*t_acc
193                         s_max = epsd_max*abs(vel_max)*t_acc
194                         t_const = (conf_final[2]-conf_inic[2] -
2*ang_max)/abs(psi_max)
195 
196                 temps_mov = t_const + 2*t_acc
197 
198                 beta = donam_angle(conf_inic,conf_final)+conf_inic[2]
199 
200                 intern.write("DADES TRAM : "+str(tram)+" \r\n")
201                 intern.write("a_max: "+str("{:.3f}".format(a_max))+" mm/s^2 \r\n")
202                 intern.write("temps acceleracio: "+str("{:.3f}".format(t_acc))+" s
\r\n")
203                 intern.write("s_max: "+str("{:.3f}".format(s_max))+" mm \r\n")
204                 intern.write("desplacament total: "+str("{:.3f}".format(desp_mov))+"
mm \r\n")
205                 intern.write("angle beta: "+str("{:.3f}".format(beta))+" rad
("+str("{:.3f}".format(beta*180/math.pi))+" ) deg \r\n")
206                 intern.write("temps a v constant: "+str("{:.3f}".format(t_const))+"
s \r\n")
207                 intern.write("---------------------------- \r\n")
208 
209                 if mov_possible:
210                         temps_inic_mov = time.time()
211                         n = 0
212                         while time.time()-temps_inic_mov < t_acc:
213 
214                                 u_acc = (time.time()-temps_inic_mov)/t_acc
215                                 motors =
[vel_max*valorb(u_acc,pol_vel1)*math.cos(beta),vel_max*valorb(u_acc,pol_vel1)*math.s
in(beta),psi_max*valorb(u_acc,pol_vel1)]
216                                 theta1 = (motors[0] - motors[2]*long_eix)/radi_roda
217                                 theta2 = (-motors[0]*math.sin(alpha) +
motors[1]*math.cos(alpha)- motors[2]*long_eix)/radi_roda
218                                 theta3 = (-motors[0]*math.sin(alpha) -
motors[1]*math.cos(alpha) - motors[2]*long_eix)/radi_roda
```

```
219
220                                 intern.write("temps control: "+str("
{:.3f}".format(round(time.time()-temps_control,3)))+" s \r\n")
221                                 intern.write("vT= "+str("
{:.2f}".format(motors[0]))+" mm/s, vL= "+str("{:.2f}".format(motors[1]))+" mm/s,
psi= "+str("{0:.2f}".format(motors[2]))+" rad/s \r\n" )
222                                 intern.write("theta1: "+str("
{:.3f}".format(theta1))+" rad/s ("+str("{:.2f}".format(theta1*conversio))+" %)
\r\n")
223                                 intern.write("theta2: "+str("
{:.3f}".format(theta2))+" rad/s ("+str("{:.2f}".format(theta2*conversio))+" %)
\r\n")
224                                 intern.write("theta3: "+str("
{:.3f}".format(theta3))+" rad/s ("+str("{:.2f}".format(theta3*conversio))+" %)
\r\n")
225                                 intern.write("---------------------------- \r\n")
226                                 time.sleep(rest_time)
227
228
229                         motors =
[vel_max*valorb(1,pol_vel1)*math.cos(beta),vel_max*valorb(1,pol_vel1)*math.sin(beta)
,psi_max*valorb(1,pol_vel1)]
230
231                         while time.time()-temps_inic_mov < t_acc+t_const:
232
233                                 theta1 = (motors[0] - motors[2]*long_eix)/radi_roda
234                                 theta2 = (-motors[0]*math.sin(alpha) +
motors[1]*math.cos(alpha)- motors[2]*long_eix)/radi_roda
235                                 theta3 = (-motors[0]*math.sin(alpha) -
motors[1]*math.cos(alpha) - motors[2]*long_eix)/radi_roda
236
237                         intern.write("tram a velocitat constant durant "+str("
{:.3f}".format(t_const))+" s \r\n")
238                         intern.write("temps control: "+str("
{:.3f}".format(round(time.time()-temps_control,3)))+" s \r\n")
239                         intern.write("vT= "+str("{:.2f}".format(motors[0]))+" mm/s,
vL= "+str("{:.2f}".format(motors[1]))+" mm/s, psi= "+str("
{0:.2f}".format(motors[2]))+" rad/s \r\n" )
240                         intern.write("theta1: "+str("{:.3f}".format(theta1))+" rad/s
("+str("{:.2f}".format(theta1*conversio))+" %) \r\n")
241                         intern.write("theta2: "+str("{:.3f}".format(theta2))+" rad/s
("+str("{:.2f}".format(theta2*conversio))+" %) \r\n")
242                         intern.write("theta3: "+str("{:.3f}".format(theta3))+" rad/s
("+str("{:.2f}".format(theta3*conversio))+" %) \r\n")
243                         intern.write("---------------------------- \r\n")
244
245                         temps_inic_fre = time.time()
246
247                         while time.time()-temps_inic_fre < t_acc:
248
249                                 u_acc = (time.time()-temps_inic_fre)/t_acc
250                                 motors = [vel_max*(1-
valorb(u_acc,pol_vel2))*math.cos(beta),vel_max*(1-
valorb(u_acc,pol_vel2))*math.sin(beta),psi_max*(1-valorb(u_acc,pol_vel2))]
251                                 theta1 = (motors[0] - motors[2]*long_eix)/radi_roda
252                                 theta2 = (-motors[0]*math.sin(alpha) +
motors[1]*math.cos(alpha)- motors[2]*long_eix)/radi_roda
253                                 theta3 = (-motors[0]*math.sin(alpha) -
motors[1]*math.cos(alpha) - motors[2]*long_eix)/radi_roda
254
```

```
255                                 intern.write("temps control: "+str("
{:.3f}".format(round(time.time()-temps_control,3)))+" s \r\n")
256                                 intern.write("vT= "+str("
{:.2f}".format(motors[0]))+" mm/s, vL= "+str("{:.2f}".format(motors[1]))+" mm/s,
psi= "+str("{0:.2f}".format(motors[2]))+" rad/s \r\n" )
257                                 intern.write("theta1: "+str("
{:.3f}".format(theta1))+" rad/s ("+str("{:.2f}".format(theta1*conversio))+" %)
\r\n")
258                                 intern.write("theta2: "+str("
{:.3f}".format(theta2))+" rad/s ("+str("{:.2f}".format(theta2*conversio))+" %)
\r\n")
259                                 intern.write("theta3: "+str("
{:.3f}".format(theta3))+" rad/s ("+str("{:.2f}".format(theta3*conversio))+" %)
\r\n")
260                                 intern.write("---------------------------- \r\n")
261                                 time.sleep(rest_time)
262
263                         theta1 = 0
264                         theta2 = 0
265                         theta3 = 0
266
267                 else:
268                         theta1 = 0
269                         theta2 = 0
270                         theta3 = 0
271                         print("Velocitat massa elevada:")
272                         print("Moviment no permes")
273                         intern.write("Velocitat massa elevada: Moviment no permes
\r\n")
274
275                 tram +=1
276
277         if temps_mov == 0:
278                 intern.write("")
279
280         else:
281                 intern.write("---------------------------- \r\n")
282                 intern.write("temps control: "+str("
{:.3f}".format(round(time.time()-temps_control,3)))+" s \r\n")
283                 intern.write("..................................... \r\n")
284
285         linia = f.readline()
286
287 running = False
288 # tancament del programa
289 print(str("{:.4f}".format(round(time.time()-temps_inic,3)))+" s \r\n")
290 intern.write("temps total: "+str("{:.3f}".format(round(time.time()-temps_inic,3)))+"
s \r\n")
291 intern.write("trams de moviment: "+str(tram)+" \r\n" )
292 intern.write("..................................... \r\n")
293 time.sleep(1)
294 f.close()
295 intern.close()
```

Annex B. 04_rotatrans.py

```python
#!/usr/bin/env python3
import threading
import time
from ev3dev2.motor import LargeMotor, OUTPUT_A, OUTPUT_B, OUTPUT_C
import math
import os
os.system('setfont Lat15-TerminusBold14')

temps_inic = time.time()
intern= open("02_dades_mov.txt","w")

#-----------------------------------------------------------#
# Introducció de dades. IMPORTANT verificar

angle_total = 5*3.14159 # angle total a girar (rad)
angle_inicial = 0 #angle inicial (rad)
temps_mov = 10 #temps de durada del moviment (s)
dist = 52 #distancia del centre del robot al CIR (mm)
radi_roda = 58/2 #radi de la roda (mm)
long_eix = 52 #longitud des del centre de la roda fins al centre del xassis (mm)
alpha = math.pi/6
thetamax= 175 # rpm maximes per construccio
rampup = 500 #temps acceleracio en ms
rampdown = 500 #temps acceleracio en ms
#-----------------------------------------------------------#
theta1 = 0     # inicialitza la vel. angular del motor 1
theta2 = 0     # inicialitza la vel. angular del motor 2
theta3 = 0     # inicialitza la vel. angular del motor 3
thetamax_rad = thetamax*2*math.pi/60 # vel angular max rad/s
velmax_centre = thetamax_rad*radi_roda # velocitat maxima del centre de la roda mm/s
conversio = 100/thetamax_rad # per tal de tenir el valor de theta com a percentatge
resolucio = 100
pas = temps_mov/resolucio
temps_seguretat = 60
psi = (angle_total-angle_inicial)/temps_mov

intern.write("Dades del moviment: rotacio + translacio (1 GL) \r\n")
intern.write("..................................... \r\n")
intern.write("radi_roda: "+str("{0:.2f}".format(radi_roda))+' mm \r\n')
intern.write("longitud_eix: "+str("{0:.2f}".format(long_eix))+' mm \r\n')
intern.write("distancia al CIR: "+str("{0:.2f}".format(dist))+' mm \r\n')
intern.write("angle girat: "+str("{0:.2f}".format(angle_total*180/math.pi))+' deg
\r\n')
intern.write("velocitat agular: "+str("{0:.2f}".format(psi))+' rad/s \r\n')
intern.write("velocitat centre: "+str("{0:.2f}".format(psi*dist))+' mm/s \r\n')
intern.write("..................................... \r\n")

running = True

class rotatrans(threading.Thread):

    def __init__(self):

        self.m1 = LargeMotor(OUTPUT_A)
        self.m2 = LargeMotor(OUTPUT_B)
        self.m3 = LargeMotor(OUTPUT_C)
        threading.Thread.__init__(self)
        motors=[self.m1, self.m2, self.m3]
        n=0
        for m in motors:
```

```
            m.reset()
            m.position = 0
            m.ramp_down_sp = rampdown - n
            m.ramp_up_sp = rampup - n
            m.stop_action = 'brake'
            n += 1

    def run(self):

        print("Engines running!")

        while running:

            self.m1.on(theta1*conversio)
            self.m2.on(theta2*conversio)
            self.m3.on(theta3*conversio)

if __name__ == "__main__":

    motor_thread = rotatrans()
    motor_thread.setDaemon(True)
    motor_thread.start()


intern.write("velocitats generalitzades xassis (vL, vT i psi): \r\n")
temps_inic_mov = time.time()
t = 0 # inicialitza el parametre t
temps_calcul = 0

while time.time()-temps_inic_mov <= temps_mov:

    t = time.time() - temps_inic_mov
    temps_calc1 = time.time()
    theta1 = -psi*(long_eix+dist*math.cos(angle_inicial-psi*t))/radi_roda
    theta2 = -psi*(long_eix-dist*math.sin(math.pi/6 - angle_inicial +
psi*t))/radi_roda
    theta3 = -psi*(long_eix-dist*math.cos(math.pi/3 - angle_inicial +
psi*t))/radi_roda
    time.sleep(pas*2)
    vT = (theta1/(1+math.sin(alpha))-theta2*0.5/(1+math.sin(alpha)) -
theta3*0.5/(1+math.sin(alpha)) )*(radi_roda)
    vL = ( theta2*0.5/math.cos(alpha) - theta3*0.5/math.cos(alpha))*(radi_roda)
    psi_x = (theta1*math.sin(alpha)/(long_eix*
(1+math.sin(alpha)))+theta2*0.5/(long_eix*(1+math.sin(alpha)))+
theta3*0.5/(long_eix*(1+math.sin(alpha))))*(-radi_roda)

    if psi_x < 0:
        vo = - math.sqrt(vT**2+vL**2)

    else:
        vo = math.sqrt(vT**2+vL**2)

    intern.write("v_T: "+str("{0:.3f}".format(vT))+" mm/s, v_L: "+str("
{0:.3f}".format(vL))+" mm/s, psi: "+str("{0:.3f}".format(psi_x))+"rad/s \r\n")
    intern.write("theta1: "+str("{0:.3f}".format(theta1))+" rad/s ("+str("
{0:.3f}".format(theta1*conversio))+" %), theta2: "+str("{0:.3f}".format(theta2))+"
rad/s ("+str("{0:.3f}".format(theta2*conversio))+" %), theta3: "+str("
{0:.3f}".format(theta3))+" rad/s ("+str("{0:.3f}".format(theta3*conversio))+" %)
\r\n")
```

```
109     intern.write("-------------------------------------------------------------------
    ------------------------------- \r\n")
110 
111     temps_calc2 = time.time()
112     temps_calcul = temps_calcul +temps_calc2 - temps_calc1
113     temps_act = time.time()-temps_inic_mov
114 
115     if time.time()-temps_inic == temps_seguretat:
116         break
117 
118 running = False
119 print("temps total: "+str(time.time()-temps_inic)+" s")
120 intern.write("temps total: "+str("{0:.3f}".format(time.time()-temps_inic))+" s
    \r\n")
121 intern.write("------------------------------ \r\n")
122 intern.write("temps de calcul: "+str("{0:.3f}".format(temps_calcul))+" s \r\n")
123 intern.write("temps moviment: "+str("{0:.3f}".format(temps_mov))+"s ("+str("
    {0:.3f}".format(temps_act))+" s) \r\n")
124 intern.write(" ..................................................... \r\n")

125 time.sleep(1) # Para l accio fins que s acabi el Thread
126 
127 intern.close()
128 
```

Annex C. 01_valors.txt

```
% dades robot NOMES MODIFICAR NUMERO NO AFEGIR ITEMS EXTRA
diametre_roda = 58
long_eix = 52
%A partir d aquesta linia es pot escriure i esborrar
% Aixo es un comentari
% evitar accents, apostrofs, dieresis, etc. NO deixar linies en blanc
% entre dades.
% Si es volen introduir les velocitats generalitzades del xassis
% La introduccio de dades es fa seguint la seguent linia:
% (x1[mm],y1[mm],psi1[graus]);(x2[mm],y2[mm],psi2[graus]); velocitat [mm/s];psi
[rad/s]
% les velocitats maximes per construccio son:
% v_max = 16.755 mm/s, psi_max = 9.34 rad/s
% [] indica temps en segons durant el qual el programa no recalcula valors
% MOVIMENT: dues trajectories quadrades
(0,0,0);(0,400,0);200;0
(0,400,0);(400,400,0);200;0
(400,400,0);(400,0,0);200;0
(400,0,0);(0,0,0);200;0
[2]
(0,0,0);(0,0,3.14159);0;2
(0,0,0);(-400,0,0);200;0
(-400,0,0);(-400,-400,0);200;0
(-400,-400,0);(0,-400,0);200;0
(0,-400,0);(0,0,0);200;0
[2]
(0,0,0);(0,0,3.14159);0;-2
```