|  |
| --- |
| #!/usr/bin/env python3 |
|  |  |
|  | ##Librerias## |
|  | from ev3dev.ev3 import \* |
|  | from time import sleep |
|  | import zmq |
|  |  |
|  | ##Definiendo los componentes a usar## |
|  | md = MediumMotor ('outA') ## Motor mediano |
|  | ml = LargeMotor ('outB') ## Motor izquierdo |
|  | mr = LargeMotor ('outC') ## Motor derecho |
|  |  |
|  | ir = InfraredSensor(); assert ir.connected, "Por favor, conecte el InfraredSensor" |
|  | ts = TouchSensor(); assert ts.connected, "Por favor, conecte el TouchSensor" |
|  | cl = ColorSensor(); assert cl.connected, "Por favor, conecte el ColorSensor" |
|  |  |
|  | ##Modo de uso de los sensores## |
|  | ir.mode = 'IR-PROX' |
|  | cl.mode = 'COL-AMBIENT' |
|  |  |
|  | ##Crendo y definiendo los puertos de coneccion con el servidor## |
|  | context = zmq.Context() |
|  | s = context.socket(zmq.REQ) |
|  | s1 = context.socket(zmq.REQ) |
|  | s2 = context.socket(zmq.REQ) |
|  | s3 = context.socket(zmq.REQ) |
|  | s.connect("tcp://192.168.1.90:7525") |
|  | s1.connect("tcp://192.168.1.90:6763") |
|  | s2.connect("tcp://192.168.1.90:6625") |
|  | s3.connect("tcp://192.168.1.90:5376") |
|  |  |
|  | ##Limite de velocidad de los motores## |
|  | def lim\_speed(speed): |
|  | if speed > 750: |
|  | speed = 750 |
|  | elif speed < 450: |
|  | speed = 450 |
|  | return speed |
|  |  |
|  | ##Definicion de funciones para movimientos## |
|  | def ori(): |
|  | md.run\_to\_rel\_pos(position\_sp=175, speed\_sp=350, stop\_action="hold") ##Motor que gira 90 a la derecha |
|  | md.wait\_while('runnig') |
|  | sleep(1) |
|  |  |
|  | def muestra(): |
|  | md.run\_to\_rel\_pos(position\_sp=-87.5, speed\_sp=350, stop\_action="hold") ##Motor que gira 90 a la derecha |
|  | md.wait\_while('runnig') |
|  | sleep(1) |
|  |  |
|  | def vuelta(a,b,c): |
|  | ml.run\_timed(time\_sp= a, speed\_sp= b, stop\_action='brake') |
|  | mr.run\_timed(time\_sp= a, speed\_sp= c, stop\_action='brake') |
|  | ml.wait\_while('runnig') |
|  | mr.wait\_while('runnig') |
|  |  |
|  | ##Definicion de guardado de datos en unarchivo.tx## |
|  | def datos(): |
|  | archi=open('datos/dato.txt', 'a') |
|  | archi.write('Fecha y Hora: '+ time.strftime("%x ")+ time.strftime("%X")+ '\n') |
|  | archi.close() |
|  |  |
|  | def gdatos(cm,lum,an): |
|  | archi=open('datos/dato.txt', 'a') |
|  | archi.write('Distancia avanzada: '+ str(cm) +'cm = '+ str(cm \* 0.01) + 'm = '+ str(cm \* 3.28084) +'ft'+'\n') |
|  | archi.write('Detecion de luz ambiental: ' + str(lum) +'%'+'\n') |
|  | archi.write('El angulo de giro fue :'+ str(an) +'\n') |
|  | archi.close() |
|  |  |
|  | ##Datos de las ganancias## |
|  | X\_REF = 120 |
|  | Y\_REF = 150 |
|  | KP = 0.4 |
|  | KI = 0.02 |
|  | KD = 0.05 |
|  | GAIN = 10 |
|  |  |
|  | ##Datos iniciales## |
|  | ml.position = 0 |
|  | mr.position = 0 |
|  |  |
|  | integral\_x = 0 |
|  | derivative\_x = 0 |
|  | last\_dx = 0 |
|  | integral\_y = 0 |
|  | derivative\_y = 0 |
|  | last\_dy = 0 |
|  |  |
|  | print("Conectado!!") |
|  | ##Entrando al ciclo while## |
|  | while not ts.value(): |
|  |  |
|  | v = ((ir.value()) \* 0.7) ##convercion a cm de la distacia detectada |
|  | ###variables para dar vuelta en el mismo eje### |
|  | z,an = 0,0 |
|  | a,b,c = 0,0,0 |
|  |  |
|  | s.send\_string(str(v)) |
|  | vi=s.recv() |
|  |  |
|  | ##Primera condicion por arriba de un valor de v el robot se movera## |
|  | if float(vi) > 35: |
|  |  |
|  | lum = cl.value() |
|  | s1.send\_string(str(lum)) |
|  | lumi=s1.recv() |
|  |  |
|  | md.stop() |
|  | ##Funcion del PID para que el motor llegue a su velocidad optima## |
|  | x = ir.value() |
|  | y = ir.value() |
|  | dx = X\_REF - x |
|  | integral\_x = integral\_x + dx |
|  | derivative\_x = dx - last\_dx |
|  | speed\_x = KP\*dx + KI\*integral\_x + KD\*derivative\_x |
|  | dy = Y\_REF - y |
|  | integral\_y = integral\_y + dy |
|  | derivative\_y = dy - last\_dy |
|  | speed\_y = KP\*dy + KI\*integral\_y + KD\*derivative\_y |
|  |  |
|  | ###condicion para evitar el inclinamiento al desplasarce### |
|  | if ml.position == mr.position: |
|  | ml.run\_forever(speed\_sp = lim\_speed(GAIN \* (speed\_y + speed\_x))) |
|  | mr.run\_forever(speed\_sp = lim\_speed(GAIN \* (speed\_y + speed\_x))) |
|  | elif ml.position > mr.position: |
|  | mr.run\_forever(speed\_sp = lim\_speed(GAIN \* (speed\_y + speed\_x)) + .5) |
|  | ml.run\_forever(speed\_sp = lim\_speed(GAIN \* (speed\_y + speed\_x)) - .5) |
|  | else: |
|  | ml.run\_forever(speed\_sp = lim\_speed(GAIN \* (speed\_y + speed\_x)) + .5) |
|  | mr.run\_forever(speed\_sp = lim\_speed(GAIN \* (speed\_y + speed\_x)) - .5) |
|  |  |
|  | last\_dx = dx |
|  | last\_dy = dy |
|  | ##Aqui termina el PID## |
|  |  |
|  | ##Para determinar la distacia que recorre el robot se toma como parametro la posicion de los motorres## |
|  | pos =(((ml.position)+(mr.position))/2) ##Posicion absoluta de los dos motores |
|  |  |
|  | s2.send\_string(str(pos)) |
|  | cm=s2.recv() |
|  | cm = float(cm) |
|  |  |
|  | sleep(1) |
|  |  |
|  | ##Segunda condicion para la toma de decicion## |
|  | else: |
|  |  |
|  | ###Se detienen los motores grandes por estar en "run\_forever"### |
|  | ml.stop() |
|  | mr.stop() |
|  | integral\_x = 0 |
|  | integral\_y = 0 |
|  | ##Funcion de ori() para la orientacion inicial del muestreo## |
|  | ##Funcion de muestra() para tomar muestra cada 45 grados del giro de la cabeza## |
|  | ##Funcion de vuelta() para crear el giro del robot## |
|  | ori() |
|  | x = (ir.value(),1500,350,-350) |
|  | sleep(1) |
|  |  |
|  | muestra() |
|  | x1= (ir.value(),750,350,-350) |
|  | sleep(1) |
|  |  |
|  | muestra() |
|  |  |
|  | muestra() |
|  | y1 = (ir.value(),750,-350,350) |
|  | sleep(1) |
|  |  |
|  | muestra() |
|  | y = (ir.value(),1500,-350,350) |
|  | sleep(1) |
|  |  |
|  | z = (x,x1,y1,y) |
|  | z,a,b,c = max(z) |
|  |  |
|  | vuelta(a,b,c) |
|  |  |
|  |  |
|  | if b > c: |
|  | an = a \* 0.06 |
|  | else: |
|  | an = a \* -0.06 |
|  |  |
|  | s3.send\_string(str(an)) |
|  | ani=s3.recv() |
|  | Sound.speak('Angle of rotation of' + str(ani)).wait() |
|  |  |
|  | #angulo(a,b,c) |
|  | datos() |
|  | gdatos(cm,lum,an) |
|  |  |
|  | ###Comando que formatea la posicion de los motores para volver a comensar un nuevo trayecto## |
|  | ml.position = 0 |
|  | mr.position = 0 |
|  | ori() |
|  |  |
|  | ##Generacion de paros completos de los motores## |
|  | ml.stop(stop\_action="hold") |
|  | mr.stop(stop\_action="hold") |
|  | md.stop(stop\_action="hold") |
|  |  |
|  | ml.stop(stop\_action="brake") |
|  | mr.stop(stop\_action="brake") |
|  | md.stop(stop\_action="brake") |
|  |  |
|  | ml.stop(stop\_action="coast") |
|  | mr.stop(stop\_action="coast") |
|  | md.stop(stop\_action="coast") |
|  |  |
|  | ml.stop() |
|  | mr.stop() |
|  | md.stop() |
|  | print ("Fin") |
|  | Sound.speak('Closed program').wait() |