from vcScript import \*

from vcHelpers.Robot import \*

from vcHelpers.Robot2 import \*

import socket

import time

import errno

import os

import vcMatrix

local\_ip = '127.0.0.1'

robot\_ip = '127.0.0.1'

recv\_port = 30001

send\_port = 30002

SIZE = 250

app = getApplication()

comp = getComponent()

robot = comp

conectado = False

err = "";

robotnum = 1;

def OnStart():

#Executes before sim clock starts running when play is pressed

global robot, exe

global local\_ip, robot\_ip, recv\_port, send\_port, my\_socket, conectado

global SIZE, robotnum

conectado = False

#Connection vars

local\_ip = comp.getProperty('LocalIP').Value

robot\_ip = comp.getProperty('RobotIP').Value

recv\_port = comp.getProperty('RecvPort').Value

send\_port = comp.getProperty('SendPort').Value

intentoConexion = 5

while (intentoConexion > 0):

print "ROBOT\_%i: Intento de conexion nº: %i" %(robotnum,6-intentoConexion)

try:

my\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

my\_socket.settimeout(1)

my\_socket.connect((robot\_ip,recv\_port))

print "ROBOT\_%i: Éxito en la conexion" %(robotnum)

break

except socket.error, exc:

print "ROBOT\_%i: Error de conexion : %s" %(robotnum,exc)

if (intentoConexion == 1):

return

finally:

intentoConexion = intentoConexion - 1

if robot:

exe = robot.findBehavioursByType(VC\_ROBOTEXECUTOR)[0]

else:

exe = None

conectado = True

def OnRun():

#Executes after sim clock starts running

global robot, exe, my\_socket, conectado, err, robotnum

#Init receiving socket

if not conectado:

print 'ROBOT\_%i: No conectado' %(robotnum)

print 'ROBOT\_%i: Cerrando el programa' %(robotnum)

return

if not robot:

print 'ROBOT\_%i: No robot associated to TCP link' %(robotnum)

print 'ROBOT\_%i: Cerrando el programa' %(robotnum)

return

print ('ROBOT\_%i: Receiving packets from local IP {0}, port {1}\n'.format(local\_ip, recv\_port)) %(robotnum)

print ('ROBOT\_%i: Sending packets to IP {0}, port {1}\n'.format(robot\_ip, send\_port)) %(robotnum)

delay(0.001)

try:

#Send dummy package so robot picks up IP

my\_socket.send('DUMMY ')#, (robot\_ip, send\_port))

except:

pass

#delay(5)

program = exe.Program

mainroutine = program.MainRoutine

EjecucionRutina(mainroutine)

print('ROBOT\_%i: Cerrando el programa') %(robotnum)

def ParamPos(posMatrix):

# Funcion encargada de formatear matrices de posicion en strings

# Parametros de posicion

P = posMatrix.P

X = P.X

Y = P.Y

Z = P.Z

# Parametros de orientacion

cuaternio = posMatrix.getQuaternion()

QS = cuaternio.X

Qi = cuaternio.Y

Qj = cuaternio.Z

Qk = cuaternio.W

# Creacion del mensaje

punto = "X"+"{:.2f}".format(X)

punto = punto + "Y"+"{:.2f}".format(Y)

punto = punto + "Z"+"{:.2f}".format(Z)

punto = punto + "QS"+"{:.2f}".format(QS)

punto = punto + "Qi"+"{:.2f}".format(Qi)

punto = punto + "Qj"+"{:.2f}".format(Qj)

punto = punto + "Qk"+"{:.2f}".format(Qk)

return punto

def EnvioRobot(data):

# Funcion encargada de enviar datos al robot

global robot\_ip, send\_port

global my\_socket

data = data + " "

#print(data)

my\_socket.sendto(data, (robot\_ip, send\_port))

pass

def Confirmacion():

# Funcion encargada de realizar la confirmacion con el robot

contador = 1

while True:

if (contador%10)==0:

EnvioRobot("RTY")

print("RTY")

return False

elif (contador%5)==0:

EnvioRobot("CNF")

try:

(data, addr) = my\_socket.recvfrom(SIZE)

#print(data)

continuar = (data == "AccionRealizada")

if continuar:

break

except:

pass

delay(0.5)

contador = contador + 1

return True

def EjecucionRutina(rutina):

# Funcion encargada de interpretar las rutinas programadas y controlar el robot

global exe, robotnum

instrucciones = rutina.Statements

nInstrucciones = len(instrucciones)

# En caso de que la rutina posea variables, las almacenamos para el correcto

# funcionamiento de las instrucciones while, if,...

try:

propiedades = rutina.Properties

for i in range(1,len(propiedades)):

var = propiedades[i]

globals()[var.Name] = var.Value

except:

pass

# Lectura de cada instruccion de la rutina y actuacion acorde

i = 0

for i in range (i,nInstrucciones):

# Bucle while que permite reintentar una instruccion si falla (mediante continue)

while True:

try:

instruccion = instrucciones[i]

tipo = instruccion.Type

#print(tipo)

if (tipo == "PtpMotion" or tipo == "LinMotion"): #Movimiento

#Obtencion de datos de Wobj

base = instruccion.Base

wobj = base.PositionMatrix

#Obtencion de datos de Posicion Objetivo

posicion = instruccion.Positions[0]

posicion = posicion.PositionInReference

#Creacion de mensajes con la informacion

wobjData = "WOB"+ParamPos(wobj)

posicionData = "POS"+ParamPos(posicion)

if (tipo == "PtpMotion"):

movData = "MOVM1" #MoveJ

else:

movData = "MOVM0" #MoveL

#print(tipo)

#Envio de mensajes

EnvioRobot(movData)

if not(Confirmacion()): # Si no se realiza la confirmacion, se reintenta

delay(1)

continue

EnvioRobot(wobjData)

if not(Confirmacion()):

delay(1)

continue

EnvioRobot(posicionData)

if not(Confirmacion()):

delay(1)

continue

elif (tipo == "SetBin"): #Señal

valor = instruccion.OutputValue

puerto = instruccion.OutputPort

#print(instruccion.OutputPort)

if (puerto == 1):

#Si el puerto de la señal es el 1 (herramienta), se indica al robot

sigData = "SIG"+str(valor\*1)

#print(sigData)

EnvioRobot(sigData)

elif (tipo == "Call"): #Llamada a subrutina

subrutina = instruccion.Routine

#print(subrutina)

EjecucionRutina(subrutina)

elif (tipo == "Delay"): #Retraso

delay(instruccion.Delay)

elif (tipo == "While"): #Bucle While

condicion = instruccion.Condition

scope = instruccion.Scope

while eval(condicion):

control = EjecucionRutina(scope)

if control == "Break":

break

elif control == "Continue":

continue

elif (tipo =="IfElse"): #Instruccion if/else

condicion = instruccion.Condition

if eval(condicion):

EjecucionRutina(instruccion.ThenScope)

else:

EjecucionRutina(instruccion.ElseScope)

elif (tipo == "Break" or tipo == "Continue"): #Break o continue (para bucles while)

return tipo

# Se ejecuta la instruccion en la simulacion

exe.callStatement(instruccion,True)

# Para aquellas instrucciones que se realicen de manera fisica en el robot,

# esperamos una confirmacion

if ((tipo == "SetBin" and puerto == 1) or tipo =="PtpMotion" or tipo=="LinMotion"):

#print "Ejecutando accion"

if not(Confirmacion()):

delay(1)

continue

# En el caso del resto de señales (todas menos la de la herramienta), introducimos

# un pequeño delay para mejorar el funcionamiento

elif (tipo == "SetBin"):

delay(1)

# En el caso de que se produzca un error de socket, se procede al reinicio del programa

except socket.error as error:

if error.errno == 10053:

print("ROBOT\_%i: Error "+str(error.errno)) %(robotnum)

print("ROBOT\_%i: Intentando reconectar...") %(robotnum)

OnStart() # reconexion

continue # se reintenta la rutina

else:

print(os.strerror(error.errno))

print("Cancelando operacion")

return # A falta de un proceso de correcion de errores más depurado, cerramos el programa

delay(0.01)

break

#Poll interval

delay(0.01)

print "Tarea completada"

return