# 如何開始使用:

## 加入檔案

URCoordinates.cs

UrSocketControl.cs

## using namespace

using UrRobot.Socket;

using UrRobot.Coordinates;

## 程式執行

先執行PC端的server才用UR端連線，若使用Client模式，只要UR手臂有開機就可以連線了。

# 類別介紹

## Unit

## Angle

## URCoordinates

## URJoint

# 函式介紹

## 伺服器建立

### void startServer(string ip, int port)

開啟server讓UR手臂可以連線，ip是自己的ip，port是要看UR，預設是888。

### void stopServer()

關閉伺服器，UR端會直接中斷。

## 客戶端連線

客戶端的Class名稱UrSocketControl.Client，設變數的時候注意一下。

目前是使用port:30002。

### bool ClientConnect(string IP)

IP是UR的IP，請去UR的settings/network去找。

### void Client\_RTDE()

持續讀取UR的內容，目前會取得URCoordinates ClientPos和URCoordinates ClientForce，讓使用者在之後可以getPosition(), getForce(), getFilterForce()。

備註:這裡已經用執行續處理了，因為不能想要的時刻才讀取，封包會錯位。

### URCoordinates getPosition()

取得Client RTDE裡的position。

### URCoordinates getForce()

取得Client RTDE裡的Force。

注意:ClientForce目前還是使用URCoordinates的單位，所以是M，但實際上未確定單位(可能是N)。

### URCoordinates getFilterForce()

取得Client RTDE裡的Force。這裡用FIFO把30個資料做平均。

注意:ClientForce目前還是使用URCoordinates的單位，所以是M，但實際上未確定單位(可能是N)。

## 手臂控制

### void Stop()

停止手臂(須等到目前動作結束)。

### bool getPosition(ref URCoordinates pos)

拿取手臂座標，手臂必須在mode.stop或是mode.pservoj

### void goGripper(int pos, int force = 0, int speed = 0, bool wait = true)

夾爪夾放，pos介於0~255，255為全部夾緊，在UR控制板上可以做夾放校正。Wait如果是false則直接進行下一個動作，不會等夾取完。

### void goPosition(URCoordinates pos)

移動到座標點[pos]，使用mode.pmovep，有時UR會算錯Rx Ry Rz，所以必須要用goPosition2才能解決。

### void goPosition2(URCoordinates pos)

移動到座標點[pos]，使用mode.pmovej，會使用內建逆向運動學[1]

備註:此一函式是利用UR指令的movej，所以移動速度比較快，也解決一些movep會到不正確的姿態問題，但是因為經過內建的逆向運動學[1]，所以會出現問題。

### void goJoint(URJoint joint)

手臂移動到姿態[joint]

### void goTrack(URJoint joint)

此函式會開始Track mode會不斷追蹤你給的姿態或座標，在追蹤過程中可任意更換姿態或座標，也可以使用其他函式。

### void goTrack(URCoordinates pos)

此函式會開始Track mode會不斷追蹤你給的姿態或座標，在追蹤過程中可任意更換姿態或座標，也可以使用其他函式。會使用內建逆向運動學[1]。

### void stopTrack()

結束Track mode。

### void goRelativePosition(URCoordinates pos)

移動相對於現在的座標。

### void goRelativePosition(Unit x = null, Unit y = null, Unit z = null, Angle Rx = null, Angle Ry = null, Angle Rz = null)

移動相對於現在的座標，此函式目的為讓使用者方便給個別軸向的值。

### void goRelativeJoint(Angle j1 = null, Angle j2 = null, Angle j3 = null, Angle j4 = null, Angle j5 = null, Angle j6 = null)

移動相對於現在的姿態，此函式目的為讓使用者方便給個別軸向的值。

### void goFunction(string info)

### bool goFile(string file = "")

file就是讀取路徑的路徑(有專門的路徑格式[2])

## 力回授

## 委派(delegate)使用

### event ServerState stateChange

### ControlFunction dynamicFunction

先委派你的function

UrSocketControl.dynamicFunction = new UrSocketControl.ControlFunction(*myFunction*)

定義你的function

bool *myFunction*(string msg)

{

if(msg = “something”)

{

//執行程式

}

}

註解:當.path檔執行到function(msg)的時候，msg會被傳到你定義的function裡並執行。

# 附錄

1.內建逆向運動學

內建的逆向運動學有時會出錯，最常見的狀況是:希望計算的座標點與目前的座標點太過接近，就會出現error:找不到。

2.路徑格式

路徑會存在”Path\\”裡面，為.path格式，可以用txt開啟編輯，有專門的寫法，未來計畫出編輯工具。