Prácticas con *RobUALab*

Programación de un brazo robot I

Carlos A. Jara Bravo (carlos.jara@ua.es)
Santiago T. Puente Méndez (santiago.puente@ua.es)
Jorge Pomares Baeza (jpomares@ua.es)
Grupo de **I**nnovación **E**ducativa en **A**utomática

Protocolo de tareas: programación de un brazo robot I

A continuación se enumeran las distintas tareas que constituyen la práctica sobre programación de un brazo robot industrial:

Tareas a realizar

- Tareas de introducción a la simulación. Se realizarán las tareas A a G del punto 4 del guión de la práctica.
- Tareas de simulación. Se realizarán las tareas H e I del punto 4 del guión de la práctica.
- Tareas en modo remoto. Se realizarán las tareas J y K del punto 4 del guión de la práctica.

Manual de programación

1. Introducción

RobUALab.Ejs posee una herramienta para la programación del entorno virtual utilizando lenguaje Java. Mediante ella, podemos diseñar y ejecutar distintas tareas a partir de una serie de líneas de código. El entorno de programación se puede ejecutar pulsando el control *Programación* situado en la parte superior la ventana principal del applet.

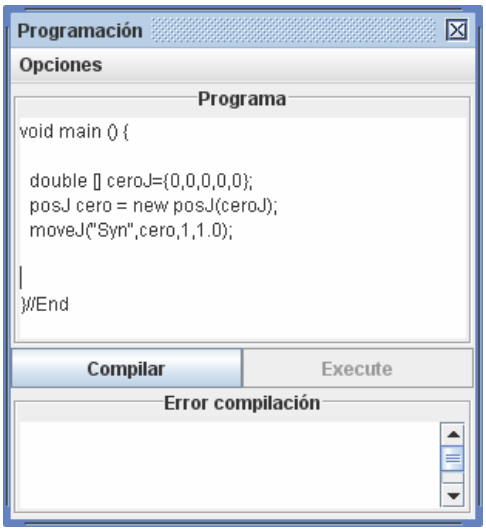


Figura 1. Entorno de programación de RobUALab.ejs

El entorno se basa en un sencillo editor de texto donde se encuentra el programa principal de la aplicación, denominado *main()*. Dentro de este método, el usuario debe introducir las líneas de código Java de la tarea que desea ejecutar en el entorno virtual. Posteriormente, para la compilación y ejecución del programa, el entorno de programación posee los controles *Compilar* y *Ejecutar*. En el caso de existir un error en la tarea programada, el entorno de programación posee una salida de texto donde aparecerá el error junto con la línea/s del programa que lo han producido para que el usuario las pueda corregir.

2. El lenguaje de programación Java

El lenguaje utilizado por la consola de programación del entorno RobUALab es el lenguaje Java. Por lo tanto, el usuario podrá utilizar cualquier método o tipo de variable de este lenguaje. Por ejemplo, se podrá declarar e inicializar cualquier tipo de variable primitiva (*int*, *float*, *double*, *boolean*) o arrays dentro de la función *main()*.

Las directivas de importación de clases y paquetes "*import*" no se pueden utilizar en la programación. La primera línea del programa debe ser "*void main (){*" y la última "*}//End*" siendo válido todo el código Java existente entre ambas líneas.

Una característica que también se debe tener en cuenta es que en el cuadro de programación deben usarse caracteres anglosajones ya que la utilización de otro tipo de caracteres puede dar lugar a errores en la compilación del código.

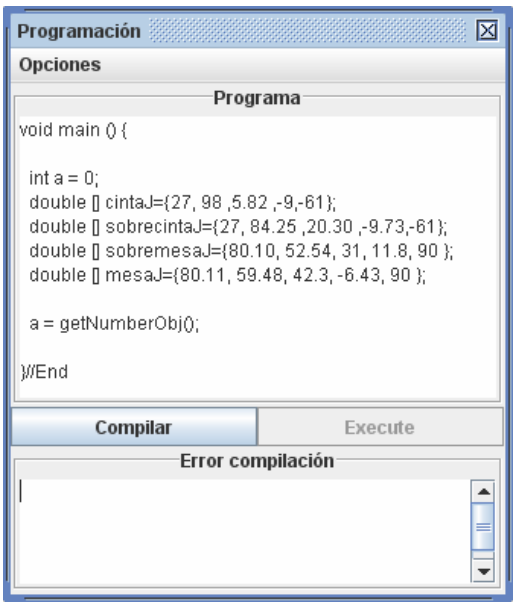


Figura 2. Uso del lenguaje Java dentro del entorno RobUALab.ejs

3. Métodos y clases para el movimiento del robot

El entorno de programación posee una serie de clases y métodos desarrollados para la programación del movimiento robot. Las dos clases principales existentes son *posJ* y *posC*, que definen un objeto posición en el espacio articular o cartesiano, respectivamente. Ambos poseen los mismos constructores y métodos, que a continuación se exponen (se explicará mediante la clase *posJ*):

- Constructores:

```
public posJ()
```
Crea un objeto *posJ* para un robot de 5 grados de libertad.

```
public posJ(int dimP)
```
Crea un objeto *posJ* para un robot de **dimP** grados de libertad.

```
public posJ(double[] valuesP)
```
Crea un objeto *posJ* a partir de un array de valores tipo **double**.

```
public posJ(double valP1,double... valuePn)
```
Crea un objeto *posJ* a partir de una serie de valores tipo **double**.

- Métodos:

```
public boolean setValue(double[] valuesP)
```
Establece los valores de posición en el objeto *posJ* a partir de un array de valores tipo **double**.

```
public boolean setValue(double value, int pos)
```
Establece el valor **value** en la posición **pos** del objeto *posJ*.

```
public int getDimension()
```
Toma el valor del tamaño del objeto *posJ*.

```
public double getValue(int pos)
```
Obtiene el valor de la posición **pos** del objeto *posJ*.

```
public double[] getValues()
```
Obtiene el array de valores del objeto *posJ*.

Estas dos clases se utilizan para poder programar movimientos en el robot. El método principal para la programación un movimiento articular en el robot es *moveJ*, que posee la siguiente declaración:

moveJ (String tipo, posJ pos, int mano, double... params)

- Parámetro **tipo**: este parámetro *String* indica el tipo de trayectoria a ejecutar. Las posibles trayectorias son:
  - Tipo "Syn": trayectoria síncrona.
  - Tipo "DSyn": trayectoria asíncrona.
  - Tipo "S3": trayectoria interpolador o spline cúbico.
  - Tipo "S5": trayectoria interpolador o spline quíntico.
  - Tipo "434": trayectoria interpolador 434.
- Parámetro **pos**: variable tipo *posJ* que especifica el punto final de la trayectoria.
- Parámetro **mano**: variable tipo *int* que especifica que se abra o cierre la pinza al principio de la trayectoria. El valor de 0 abre la pinza y el valor de 1 cierra la pinza.
- Parámetro **params**: especifica el tiempo de la trayectoria en segundos. Para las trayectorias *Syn* (síncrona), *S3* (spline cúbico) y *S5* (spline quíntico), el parámetro **params** es un valor o variable tipo *double*. Para la trayectoria *DSyn*, **params** es un array de tamaño 5 tipo *double* con el valor de tiempo de cada una de las articulaciones del robot. Y finalmente, para la trayectoria *434*, **params** especifica el valor del tiempo de aceleración y deceleración de la trayectoria en dos variables tipo *double*.

Existen otras funciones que también se ocupan de la programación de las trayectorias del robot:

- Función *home_robot(int mano)*: lleva al robot a la posición de inicio. El parámetro *mano*, se comporta de una manera similar que en la función *moveJ* explicada anteriormente.
- Función *set_speed(double[] v)*: configura la velocidad máxima del robot. El array *v* debe tener un tamaño de 5 y especifica el valor de la velocidad en rad/s.
- Función *open()* y *close()*: abre y cierra la pinza del robot.
- Función *belt()*: pone en funcionamiento la cinta transportadora del entorno hasta encontrar un objeto virtual.

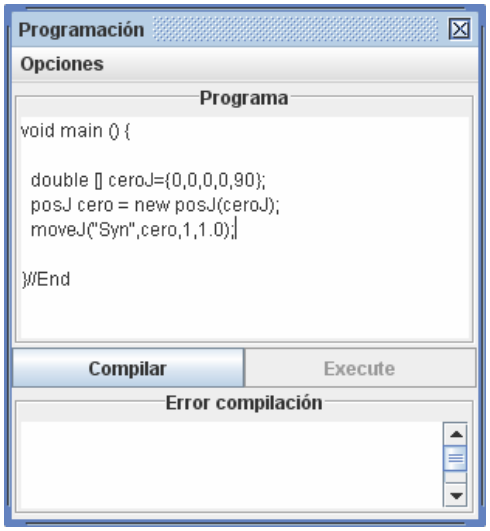


Figura 3. Ejemplo de programación

4. Métodos para obtener información del entorno virtual

La aplicación permite otras funciones para obtener información del entorno virtual. A continuación las funciones que el usuario puede utilizar en el entorno de programación:

```
public double[] getPosObj(String name)
```
Obtiene el valor de la posición del objeto del entorno virtual en coordenadas del mundo cuyo nombre es el pasado a la función.
```
public double[] getPosObj(int number)
```
Obtiene el valor de la posición del objeto del entorno virtual en coordenadas del mundo cuyo número es el pasado a la función.
```
public double[] getPosRobot()
```
Obtiene el valor de la posición del robot en coordenadas del mundo.
```
public int getNumberObj()
```
Obtiene el número de objetos existentes en el entorno virtual.
```
public posC getDirect(posJ pos)
```
Obtiene un objeto *posC* con las coordenadas (X,Y,Z,Roll,Pitch,Yaw) del extremo final del robot en coordenadas de mundo a partir del objeto *posJ* pasado a la función. En el caso de no obtener solución, devuelve **null.**

`public posJ getInverse(posC pos)`
Obtiene un objeto *posJ* con las coordenadas articulares (q1,q2,q3,q4,q5) a partir del objeto *posC* pasado a la función. En el caso de no obtener solución, devuelve **null.**

5. Ejemplo de programación

A continuación se muestra un programa que ejecuta un proceso que pone en marcha la cinta transportadora y ordena al robot a coger el objeto que la cinta transportadora ha detectado. Para las trayectorias, se ha utilizado el planificador "434". Para ello, antes de compilar el programa debe existir un objeto en la cinta transportadora.

```
void main () {
 //Vectores con las posiciones articulares
 double[] p1={26,77,28,-9,-63};
 double[] p2={26,96,12,-12,-63};
 double[] p3={26,64,26,4,-63};
 double[] p4={95,54,26,4,-63};
 double[] p5={95,65,26,4,-63};
 //Objetos posicion
 posJ pos1 = new posJ(p1);
 posJ pos2 = new posJ(p2);
 posJ pos3 = new posJ(p3);
 posJ pos4 = new posJ(p4);
 posJ pos5 = new posJ(p5);
 //Pone en funcionamiento la cinta hasta detectar un objeto
 belt();
 //Trayectorias del robot
 moveJ("434",pos1,-1,0.1,0.1);
 moveJ("434",pos2,0,0.1,0.1);
 moveJ("434",pos3,1,0.1,0.1);
 moveJ("434",pos4,-1,0.1,0.1);
 moveJ("434",pos5,-1,0.1,0.1);
 home_robot(0);
}//End
```

Programación de un brazo robot I: guión de prácticas

1. Introducción

Para llevar a cabo la práctica es conveniente leer detenidamente todos los apartados e ir realizando todas aquellas tareas que se proponen.

Las tareas de la práctica se realizarán con la ayuda del simulador del robot Scorbot ER-IX descrito en la sección "Interfaz", que se utilizará por una parte como herramienta para realizar los cálculos cinemáticas, la programación, y las simulaciones para comprobar visualmente los resultados obtenidos, y por otra parte para evaluar los resultados sobre el robot real mediante operación remota.

Las primeras tareas de la práctica se realizarán de forma teórica y en *modo simulación*, previas a la utilización del robot real, y posteriormente se realizarán las comprobaciones sobre el robot real en *modo remoto*. Al finalizar las tareas teóricas y de simulación, el alumno deberá enviar un informe al profesor con las experiencias realizadas, en el que también deben incluirse las tareas teóricas. Después, puede proceder a realizar las tareas de modo remoto, tras lo cual deberá enviar otro informe. El alumno deberá contrastar los resultados obtenidos en modo simulación con los resultados obtenidos en modo remoto.

2. Objetivos de la práctica

En esta práctica se pretende que el alumno se familiarice con la programación de un robot industrial para realizar tareas sencillas. Así, se abordará el posicionamiento del robot y la manipulación de objetos. Todo ello trabajando sobre el entorno virtual RobUALab y el robot Scorbot ER-IX.

3. Descripción de la práctica

El alumno debe utilizar el entorno RobUALab dentro del proyecto AutomatL@bs (http://lab.dia.uned.es/automatlab/) para familiarizarse con un brazo robot industrial de cinco grados de libertad. Además de conocer las distintas opciones que ofrece en entorno: movimientos, introducción de objetos en el entorno, manipulación de objetos, programación de movimientos mediante código, etc.

Para todo ello se dispone, de un manual en la web sobre el entorno y otro manual sobre el lenguaje de programación del robot.

En el caso de querer introducir objetos en el plano inclinado se debe realizar mediante un fichero, en el que las coordenadas de posición y orientación de un objeto rectangular son: "1,76 0,85 0,175 {1.57,0,0,1}", los tres primeros valores indican la posición del objeto en el mundo en coordenadas globales del sistema y los valores entre llaves la orientación, de

manera que la orientación se expresa como un giro, el primer valor, y los tres siguientes representan el vector unitario que define el eje de giro.

Por ejemplo: si deseamos introducir tres objetos rectangulares, de tamaño 60x30x30mm, en el plano inclinado, uno rojo, otro verde y otro azul, se deberá crear un fichero con las siguientes líneas:

```
Object1  1,76  0,85  0,175  {1.57,0,0,1}  255,0,0  Rectangle
Object2  1,76  0,85  0,175  {1.57,0,0,1}  0,255,0  Rectangle
Object3  1,76  0,85  0,175  {1.57,0,0,1}  0,0,255  Rectangle
```

4. Tareas a realizar

4.1. Tareas de introducción a la simulación

Las tareas de este apartado tienen como objetivo la familiarización con el simulador, con ayuda del manual que describe el interfaz de usuario del mismo. No se requiere entregar informe o memoria sobre estas tareas.

A. Entrar en el entorno eMersion con los datos de usuario y contraseña asignados. Practicar para mover el robot en su entorno usando primero los controles de movimientos articulares, y luego los controles para mover el extremo del robot en coordenadas cartesianas. Observar cómo se corresponden ambos movimientos, esto es, la cinemática directa e inversa del robot.

B. Probar a introducir trayectorias, usando distintos interpoladores de valores articulares. Será necesario especificar una posición de destino a la que llegar, el tiempo para el movimiento y la velocidad.

C. Observar cómo es la evolución de posición y de velocidad para las diferentes articulaciones cuando se ejecutan las trayectorias. Observar también la evolución en el espacio cartesiano (opción Control del menú).

D. Concatenar varias trayectorias, usando la opción Save con cada una, y ejecutar la lista completa.

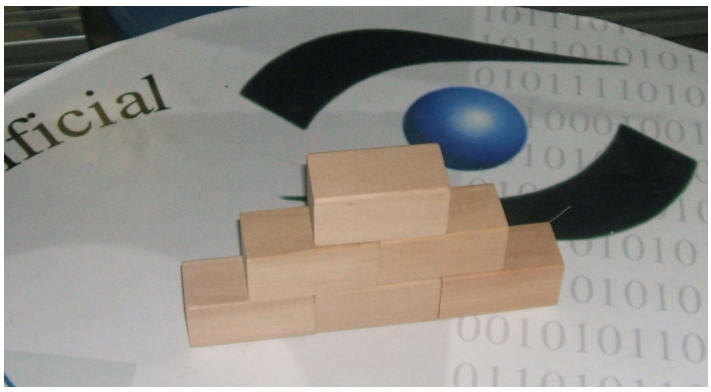
E. Probar a introducir objetos en el entorno y manipularlos con el robot.

F. Almacenar objetos en un fichero y recuperarlo.

G. Realizar un programa sencillo para ver como se comporta la programación.

4.2. Tareas de simulación

El alumno debe realizar el código de programación necesario para que el robot recoja las piezas existentes en plano inclinado y las posiciones sobre la mesa formando un muro, tal y como muestra la siguiente figura:



Para ello el robot colocará la primera pieza en la posición que se desee de la mesa, a continuación la siguiente junto a esta, a derecha o izquierda según se prefiera, la siguiente la pondrá encima de estas, en el medio de las dos. La siguiente pieza se colocará a la derecha o a la izquierda de las dos piezas de la base, la siguiente en el segundo nivel, ocupando la mitad de la pieza del nivel base que se ha colocado previamente. La siguiente en un tercer nivel, en el medio de las dos piezas del nivel segundo. Y así de manera similar mientras queden piezas disponibles en el plano inclinado. La posición de colocación de cada pieza en la estructura se debe calcular en el momento de manipular dicha pieza, no se permite el cálculo previo de todas las posiciones, hay que suponer que el número total de piezas existentes no es conocido a priori para el cálculo de posiciones.

Hay que indicar, que debido a que el robot coge la pieza en el plano inclinado por los extremos, se deberá utilizar un punto en la mesa para dejar la pieza, girar la pinza y volver a coger la pieza antes de colocarla en la estructura, ya que sino la manipulación será imposible, por colisionar la pinza del robot con una la pieza puesta anteriormente.

El alumno entregará una memoria con el código resultante.

H. *Programación de la tarea*. El alumno realizar la programación de la tarea.

I. *Simular la tarea*. Introducir el código en el simulador para que este realice la trayectoria deseada, para ello se deberán introducir en el plano inclinado varias piezas con las que pueda realizar la estructura.

4.3. Tareas en modo remoto

J. Introducir de nuevo el código realizado, como en el apartado I del punto 4.2. Conectar con el robot remoto y ejecutar la trayectoria.

K. Visualizar y analizar los valores reales del movimiento del robot real. Comparar el resultado de la tarea simulada con la real.