UTN – FRBA

Robótica

Embedded + Compilador Robot RT1 Poseidón

(Tipo Scara de 3 grados de libertad)

Profesor: Hernán Giannetta

Alumnos:

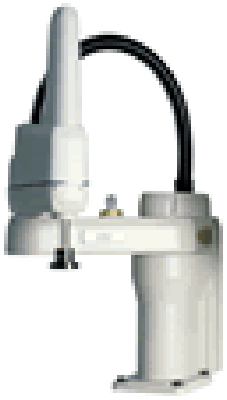
Martín Cabrera 111492-0
Christian Gutierrez 96985-5

1C/2009

Índice

Introducción. Características principales del sistema y del robot.

Configuración Scara



Esta configuración está especialmente diseñada para realizar tareas de montaje en un plano. Está constituida por dos articulaciones de rotación con respecto a dos ejes paralelos, y una de desplazamiento en sentido perpendicular al plano. El volumen de trabajo de este robot, es función de la longitud de los brazos que lo componen y de los límites de ángulos de giro impuestos por las características constructivas de las articulaciones.

Para llevar a cabo los cálculos y de esta forma asegurar el correcto funcionamiento del robot en cuanto a la cinemática y dinámica se refiere, se toma en consideración la siguiente teoría:

Modelo cinemático directo de los manipuladores.

La cinemática es la ciencia del movimiento que trata a éste sin importarle las fuerzas que lo causan. Dentro de la cinemática se estudia la posición, la velocidad, aceleración y todas las derivadas de las variables de posición de mayor orden con respecto al tiempo o cualquier otra variable. El estudio de la cinemática de los manipuladores se refiere a todas las propiedades geométricas y basadas en el tiempo del movimiento.
Los robots consisten en un conjunto de eslabones conectados mediante articulaciones que permiten el movimiento relativo entre los eslabones vecinos. El número de grados de libertad que un robot posee es el número de variables de posición independientes que deberían ser especificadas para localizar todas las partes del mecanismo. En el caso de los robots industriales el número de grados de libertad suele equivaler al número de articulaciones siempre y cuando cada articulación tenga un solo grado de libertad. Al final de la cadena de eslabones del robot se encuentra el órgano terminal. Dependiendo de la aplicación del robot, el órgano terminal puede ser una pinza, un soldador, un electroimán o un gripper como en este caso.
Generalmente se describe la posición del robot dando una descripción del marco de la herramienta, la cual esta unida al órgano terminal, relativo al marco de la base, el cual está a su vez unido a la base fija del robot.
El modelo cinemático directo es el problema geométrico que calcular la posición y orientación del efector final del robot. Dados una serie de ángulos entre las articulaciones, el problema cinemática directo calcula la posición y orientación del marco de referencia del efector final con respecto al marco de la base.

Modelo cinemático inverso del robot.

Dada la posición y orientación del efector final del robot, el problema cinemático inverso consiste en calcular todos los posibles conjuntos de ángulos entre las articulaciones que podrían usarse para obtener la posición y orientación deseada.
El problema cinemático inverso es más complicado que la cinemática directa ya que las ecuaciones no son lineales, sus soluciones no son siempre fáciles o incluso posibles en una forma cerrada. También surge la existencia de una o de diversas soluciones. La existencia o no de la solución lo define el

espacio de trabajo de un robot dado. La ausencia de una solución significa que el robot no puede alcanzar la posición y orientación deseada porque se encuentra fuera del espacio de trabajo del robot o fuera de los rangos permisibles de cada una de sus articulaciones.

Resolución del problema cinemático inverso por métodos geométricos

Como se ha indicado, este procedimiento es adecuado para robots de pocos grados de libertad o para el caso de que se consideren sólo los primeros grados de libertad, dedicados a posicionar el extremo.
El procedimiento en sí se basa en encontrar suficiente número de relaciones geométricas en las que intervendrán las coordenadas del extremo del robot, sus coordenadas articulares y las dimensiones físicas de sus elementos.
Para mostrar el procedimiento a seguir se va a aplicar el método a la resolución del problema cinemático inverso de un robot. Los datos de partida son las coordenadas (p_x, p_y, p_z) referidas a $[S_0]$ en las que se quiere posicionar su extremo.
Como se ve, este robot posee una estructura planar, quedando este plano definido por el ángulo de la primera variable articular q_1
El valor de q se obtiene inmediatamente como:

$$q_1 = arctg\left(\frac{p_y}{p_x}\right)$$

Considerando ahora únicamente los elementos 2 y 3 que están situados en un plano (ver fig. más abajo), y utilizando el teorema del coseno, se tendrá:

$$\left.\begin{aligned} r^2 &= p_x^2 + p_y^2 \\ r^2 &= p_z^2 = l_2^2 + l_3^2 + 2\,l_2\,l_3\cos q_3 \end{aligned}\right\} \Rightarrow$$

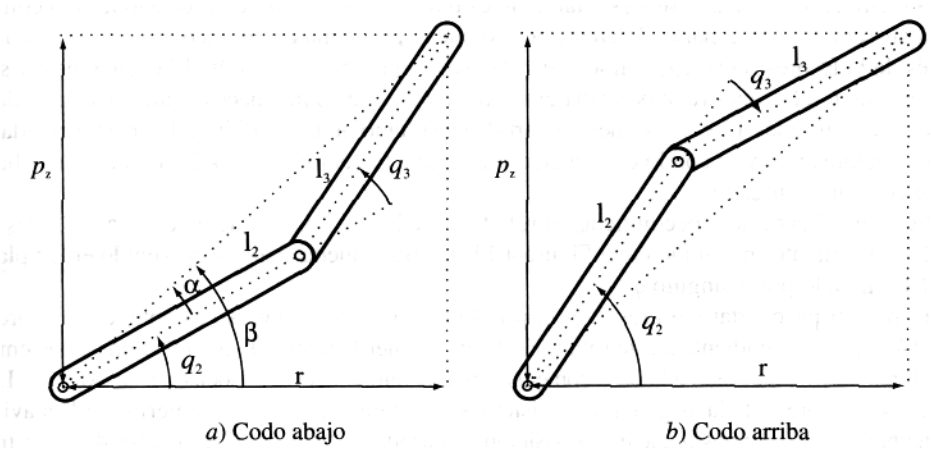
$$\cos q_3 = \frac{p_x^2 + p_y^2 + + p_z^2 - l_2^2 - l_3^2}{2\,l_2\,l_3}$$

Esta expresión permite obtener q_3 en función del vector de posición del extremo p. No obstante, y por motivos de ventajas computacionales, es más conveniente utilizar la expresión de la arcotangente en lugar del arcoseno.

$$\operatorname{sen} q_3 = \pm\sqrt{1-\cos^2 q_3}$$

$$q_3 = \operatorname{arctg}\left(\frac{\pm\sqrt{1-\cos^2 q_3}}{\cos q_3}\right)$$

Como se ve en la siguiente figura, donde se representaron los elementos 2 y 3, existen 2 posibles soluciones para q_3 según se tome el signo positivo o el signo negativo en la raíz. Éstas corresponden a las configuraciones de codo arriba y codo abajo del robot.



a) Codo abajo *b*) Codo arriba

El cálculo de q_2 se hace a partir de la diferencia entre β y α*:*

$$q_2 = \beta - \alpha$$

Siendo:

$$\beta = arctg\left(\frac{p_z}{r}\right) = arctg\left(\frac{p_z}{\pm\sqrt{p_x^2 + p_y^2}}\right)$$

$$\alpha = arctg\left(\frac{l_3 \text{ sen } q_3}{l_2 + l_3 \cos q_3}\right)$$

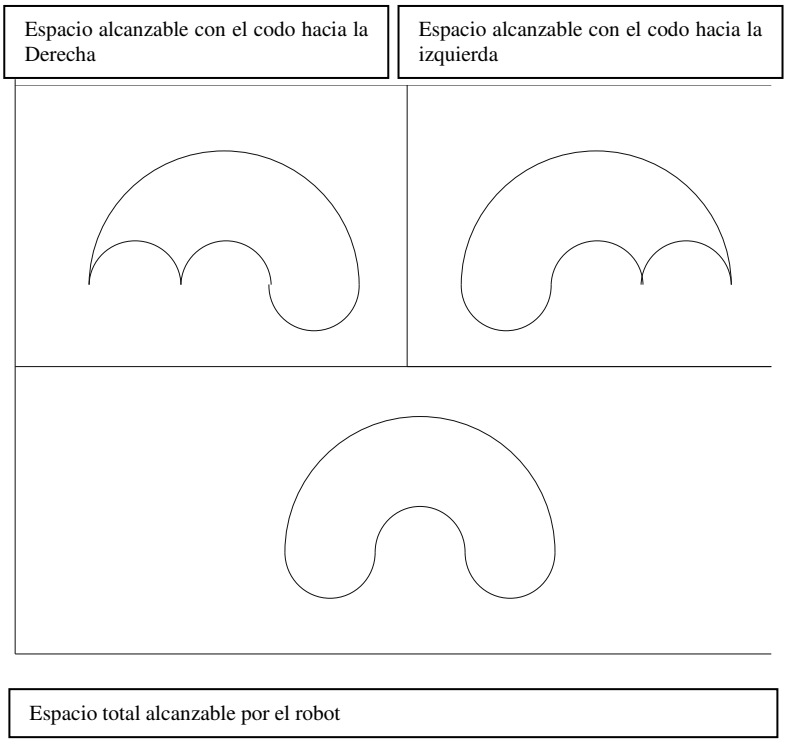
Y finalmente

$$q_2 = arctg\left(\frac{p_z}{\pm\sqrt{p_x^2 + p_y^2}}\right) - arctg\left(\frac{l_3 \text{ sen } q_3}{l_2 + l_3 \cos q_3}\right)$$

De nuevo los dos posibles valores según la elección del signo dan lugar a dos valores diferentes de q_2 correspondientes a las configuraciones codo arriba y abajo.
Las expresiones resuelven el problema cinemático inverso para el robot considerado.

Se obtiene así una nueva restricción en el espacio alcanzable para esta ecuación, puesto que la misma se hace infinito cuando q_1= 90º o -90º

Se obtiene el siguiente espacio alcanzable por el robot:



Velocidades, fuerzas estáticas y singularidades

Además de tratar problemas de posicionamiento estáticos debemos analizar los robots en movimiento.
Cuando se analiza la velocidad de un mecanismo es conveniente definir una matriz llamada el Jacobiano del manipulador. El jacobiano establece una aplicación entre las velocidades del espacio de articulaciones y las velocidades del espacio cartesiano. En ciertos puntos (llamados singularidades) está aplicación no es invertible.
Muchas veces los robots que no se mueven en el espacio, simplemente aplican una fuerza estática sobre alguna pieza o superficie de trabajo. En este caso se nos proporciona una fuerza de contacto y un momento de aquí que la matriz Jacobiana nos ayuda a encontrar la solución.

Se va a obtener la Jacobiana geométrica del Robot Scara . Las tres primeras filas de ésta relacionarán las componentes de la velocidad lineal *v* ; con las velocidades articulares, mientras que las tres últimas filas definirán la relación entre las componentes de la velocidad angular *w* y las articulares. De modo que

$$\begin{bmatrix} v_x \\ v_y \\ v_z \\ w_x \\ w_y \\ w_z \end{bmatrix} = \mathbf{J} \cdot \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} \mathbf{J}_v \\ \\ \mathbf{J}_w \end{bmatrix} \cdot \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}$$

La matriz T correspondiente al robot viene dada por:

$$\mathbf{T} = {}^{0}\mathbf{A}_4 = \begin{bmatrix} C_{124} & S_{124} & 0 & l_3C_{12} + l_2C_1 \\ S_{124} & -C_{124} & 0 & l_3S_{12} + l_2S_1 \\ 0 & 0 & -1 & -l_4 + q_3 + l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Siendo, por tanto,

$$\mathbf{p} = (p_x,\ p_y,\ p_z)$$

Con

$$p_x = l_3C_{12} + l_2C_1$$
$$p_y = l_3S_{12} + l_2S_1$$
$$p_z = l_1 + q_3$$

$$v_x = \frac{dp_x}{dt} = -\left(l_3S_{12} + l_2S_1\right)\dot{q}_1 - l_3S_{12}\dot{q}_2$$

$$v_y = \frac{dp_y}{dt} = \left(l_3C_{12} + l_2C_1\right)\dot{q}_1 + l_3C_{12}\dot{q}_2$$

$$v_z = \frac{dp_z}{dt} = \dot{q}_3$$

Por lo que J tomará la forma:

$$\mathbf{J}_v = \begin{bmatrix} -\left(l_3S_{12} + l_2S_1\right) & -l_3S_{12} & 0 & 0 \\ l_3C_{12} + l_2C_1 & l_3C_{12} & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

La submatriz J_v se obtendrá derivando la expresión de p con respecto de q_1 q_2 q_3 y q_4. Para obtener J_w se obtendrá la matriz antisimétrica Ω, a partir de la submatriz de rotación R. En este caso la submatriz de rotación R vale:

$$\mathbf{R} = \begin{bmatrix} C_{124} & S_{124} & 0 \\ S_{124} & -C_{124} & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

Su derivada respecto del tiempo será:

$$\dot{\mathbf{R}} = \frac{d\mathbf{R}}{dq_1}\dot{q}_1 + \frac{d\mathbf{R}}{dq_2}\dot{q}_2 + \frac{d\mathbf{R}}{dq_3}\dot{q}_3 + \frac{d\mathbf{R}}{dq_4}\dot{q}_4 =$$
$$= \begin{bmatrix} -S_{124} & C_{124} & 0 \\ C_{124} & S_{124} & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot (\dot{q}_1 + \dot{q}_2 + \dot{q}_4)$$

Por tanto valdrá:

$$\boldsymbol{\Omega} = \begin{bmatrix} 0 & -w_z & w_y \\ w_z & 0 & -w_x \\ -w_y & w_x & 0 \end{bmatrix} = \dot{\mathbf{R}} \cdot \mathbf{R}^{\mathbf{T}} = (\dot{q}_1 + \dot{q}_2 + \dot{q}_4) \cdot \begin{bmatrix} -S_{124} & C_{124} & 0 \\ C_{124} & S_{124} & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} C_{124} & S_{124} & 0 \\ S_{124} & -C_{124} & 0 \\ 0 & 0 & -1 \end{bmatrix}^T =$$

$$= (\dot{q}_1 + \dot{q}_2 + \dot{q}_4) \cdot \begin{bmatrix} -S_{124} & C_{124} & 0 \\ C_{124} & S_{124} & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} C_{124} & S_{124} & 0 \\ S_{124} & -C_{124} & 0 \\ 0 & 0 & -1 \end{bmatrix} =$$

$$= (\dot{q}_1 + \dot{q}_2 + \dot{q}_4) \cdot \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

De donde se obtiene que $w = (w_x, w_y, w_z)$ vale:

$$w_x = 0$$
$$w_y = 0$$
$$w_z = \dot{q}_1 + \dot{q}_2 + \dot{q}_4$$

Siendo, por tanto, la submatriz J_w

$$\mathbf{J}_w = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

Resultando que la Jacobiana geométrica para el robot SCARA toma la forma:

$$\mathbf{J}_v = \begin{bmatrix} \mathbf{J}_v \\ \mathbf{J}_w \end{bmatrix} = \begin{bmatrix} -(l_3 S_{12} + l_2 S_1) & -l_3 S_{12} & 0 & 0 \\ l_3 C_{12} + l_2 C_1 & l_3 C_{12} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

Con ella puede conocerse la velocidad lineal y angular del extremo del robot expresada en el sistema de coordenadas {S0}, según:

$$\begin{bmatrix} v_x \\ v_y \\ v_z \\ w_x \\ w_y \\ w_z \end{bmatrix} = \mathbf{J} \cdot \begin{bmatrix} q_1 \\ \\ q_2 \\ q_3 \\ \\ q_4 \end{bmatrix}$$

Dinámica

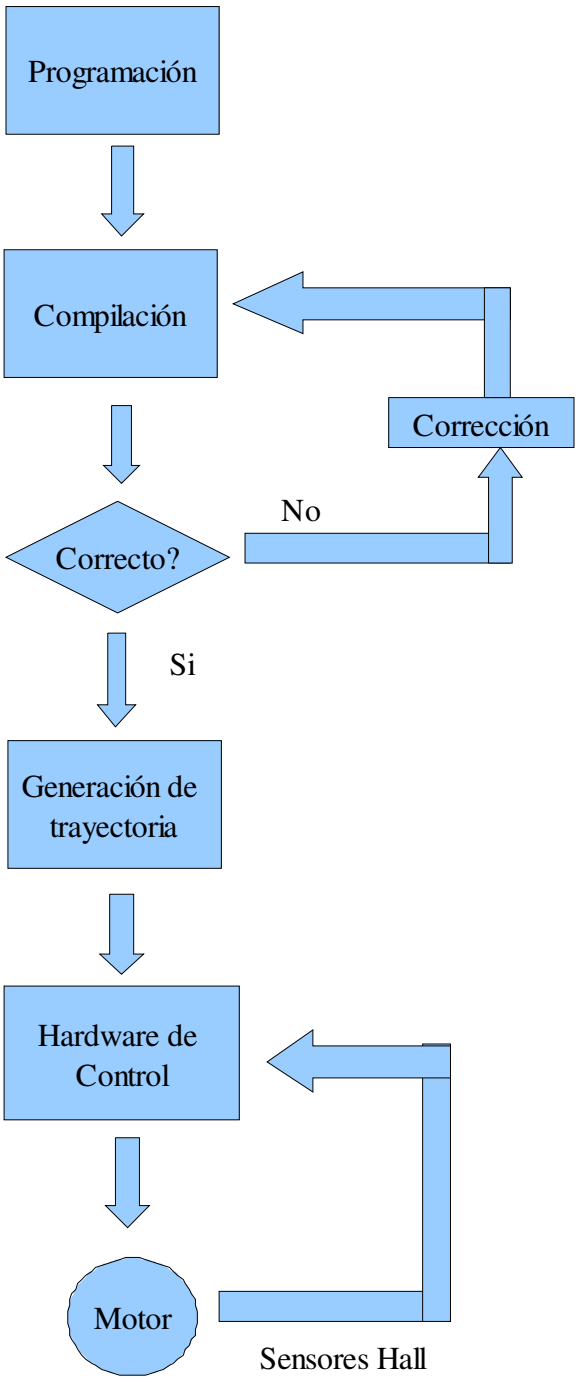
La dinámica es un campo de la física dedicado al estudio de las fuerzas requeridas para producir el movimiento. Para acelerar un robot desde el reposo y finalmente desacelerarlo hasta una completa posición de reposo, los actuadotes articulares (motores eléctricos, actuadotes hidráulicos y neumáticos) deben aplicar un conjunto complejo de funciones de par.

Un método para controlar que un robot siga un camino determinado consiste en calcular estas funciones de par usando las ecuaciones dinámicas del robot. Un segundo uso de las ecuaciones

dinámicas del movimiento es en la simulación. Reformulando las ecuaciones dinámicas de forma que la aceleración se calcule como una función del par actuador, es posible simular cómo un robot se movería bajo la aplicación de un conjunto de pares del actuador.

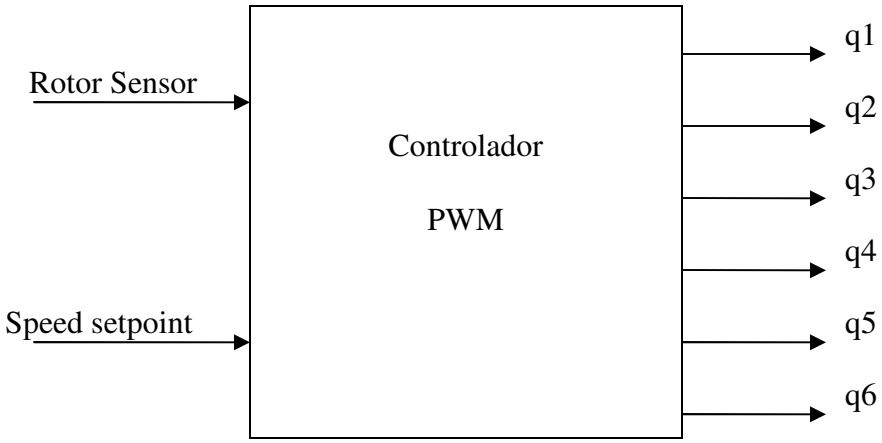
Circuito esquemático y diagrama en bloques

El concepto del sistema consta de los siguientes elementos:

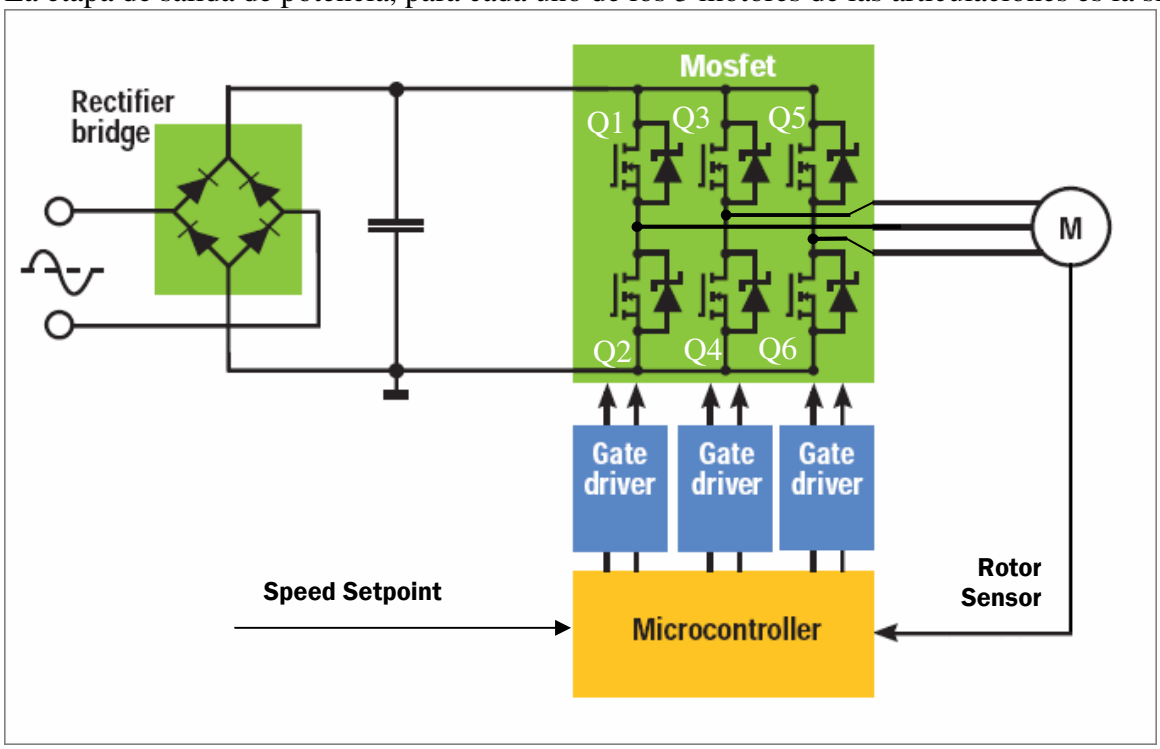


El programa compilador corre sobre una PC (Linux o Windows) y genera a partir del código en lenguaje RT, 4 archivos de salida con la información necesaria para mover al robot de la forma programada.
Uno de estos archivos es el que contiene la información a enviar al módulo *driver_pwm.vhd* desarrollado en el TPN2. Las consignas de los PWM (velocidad de los motores de cada articulación) están en el archivo, utilizando la base de tiempo del controlador. También desarrollamos una interfaz gráfica para simular (sólo bajo Linux).

El módulo *driver_pwm.vhd* recibe la consigna de velocidad de cada una de las articulaciones y en consecuencia genera la salida PWM de potencia, con la secuencia correspondiente al giro del motor.



La etapa de salida de potencia, para cada uno de los 3 motores de las articulaciones es la siguiente:



Introducción al lenguaje de programación del robot utilizado

A continuación se describe la sintaxis del lenguaje de programación desarrollado para el robot. El mismo fue creado pensando en la funcionalidad que le daríamos al robot y tomando ideas de los lenguajes ya existentes (como ABB Rapid y Epson RC+).

Formato de datos y reglas generales

`<velocidad>`
Parámetro de los comandos de movimiento que especifica el porcentaje de la velocidad máxima a la que se realizará el movimiento.
Formato:
`s<real>`
Es decir, una letra s seguida por un número real positivo.

`<posición>`
Parámetro que especifica una posición cartesiana (o un desplazamiento en coordenadas cartesianas)
Formatos posibles:
`<real> <real> <real>`
Cada número real es la posición o el desplazamiento en x, y, z respectivamente.
`p##`
Utiliza la posición previamente guardada en el punto preestablecido numero ##.
`inicio`
Utiliza las coordenadas de la posición de reposo (todo el brazo extendido).

`<articulación>`
Parámetro que especifica una posición de las articulaciones del robot (o un desplazamiento en las mismas)
Formatos posibles:
`<real> <real> <real>`
Cada número real es la posición o el desplazamiento en q1, q2, q3 respectivamente. Recuerde que en el Scara q1 y q2 son angulares (en grados) y q3 lineal.
`j##`
Utiliza la posición de articulaciones previamente guardada en el punto preestablecido numero ##.
`inicio`
Utiliza las posiciones de reposo de las articulaciones (todo el brazo extendido).

```
; Comentario
! Comentario
```
Los comentarios dentro del programa pueden ser escritos con cualquiera de los dos prefijos. Sólo se aceptan comentarios de toda la línea.

Comandos de movimiento

`inicio`
Mueve el robot a la posición de reposo (todo el brazo extendido) al 25% de velocidad. La posición de inicio para los valores por defecto es el punto $x = 75$, $y = 0$, $z = 0$.

`moveje[_rel] <articulación> <velocidad>`
Mueve el robot por la posición de las articulaciones, directamente. La velocidad se expresa como porcentaje de la velocidad máxima de cada articulación.

`mover[_rel] <posición>`
Mueve el robot rápidamente sin importar la trayectoria, cada articulación se mueve independientemente. Las articulaciones se mueven a su velocidad máxima.

`linea[_rel] <posición> <velocidad>`
Mueve el robot por una línea recta, desde la posición actual a la especificada.

`circ[_rel] <posición_1> <posición_2> <velocidad>`
Mueve el robot por una trayectoria circular, desde la posición actual a la posición_2, pasando por posición_1.
Nota: La trayectoria es circular en el plano x-y, la coordenada z se varía en forma lineal con el avance de la misma. El resultado es un movimiento elíptico.

`[_rel]`
El modificador opcional _rel hace que las posiciones especificadas en el comando sean interpretadas como relativas a la posición actual.

Comandos de asignación

`j## <articulación>`
Asignación de la posición preestablecido numero ## a la posición de las articulaciones especificada. Existen 100 posiciones, del j00 al j99.

`p## <posición>`
Asignación del punto preestablecido numero ## a la posición especificada. Existen 100 puntos, del p00 al p99.

Comandos de configuración

`set offset <posición>`
Establece el valor actual del offset. El valor del offset afecta todas las posiciones absolutas utilizadas en los comandos subsiguientes.

`set maxvel <real>`
Establece el valor actual de la velocidad máxima. Utiliza el valor absoluto del número real especificado.

`set maxacel <real> [<real>]`
Establece los valores de aceleración y desaceleración máximos actuales. En el caso de especificar un solo valor, lo utiliza tanto para la aceleración como para la desaceleración. Utiliza los valores absolutos de los números reales especificados.

`c#`
Establece la configuración del robot, donde # puede ser:
- `0` configuración tipo "brazo derecho".
- `1` configuración tipo "brazo izquierdo".

`t#`
Establece el tipo de trayectoria para los comandos siguientes, donde # puede ser:
- `0` Velocidad constante (Sólo para pruebas, ya que produciría aceleraciones instantáneas infinitas).
- `1` Perfil de velocidad trapezoidal. Utiliza la aceleración y la desaceleración máximas actuales.
- `2` Perfil de velocidad sigmoidal.

Comandos de visualización e información

`get offset`
Muestra en la consola el valor actual del offset.

`get maxvel`
Muestra en la consola el valor actual de la velocidad máxima.

`get maxacel`
Muestra en la consola los valores de aceleración y desaceleración máximos actuales.

`ctime`
Muestra en la consola el tiempo actual.

`cpos`
Muestra en la consola la posición actual.

`cjoint`
Muestra en la consola el valor actual de la posición de las articulaciones.

Comandos de flujo de programa

`fin`
Termina la ejecución del programa (ignora las líneas de programa siguientes).

Otros comandos

`do# <on | off>`
Enciende (on) o apaga (off) la salida digital numero #.

`espera <tiempo>`
Hace una demora del tiempo especificado, manteniendo la posición actual. El tiempo debe expresarse en segundos y puede ser un valor real positivo.

Comentarios sobre el desarrollo del software del compilador

Bison / Flex y gcc

Para la programación del compilador, utilizamos herramientas GNU, con Linux como sistema operativo. Igualmente comprobamos que todas ellas se pueden utilizar bajo Windows mediante alguna herramienta de emulación como *CygWin* o *MinGW*. Justamente bajo *MinGW* pudimos lograr compilar el programa en un ejecutable .EXE que corre en una consola de Windows.
Las herramientas elegidas fueron *Bison* como analizador sintáctico, *Flex* como analizador léxico y *gcc* como compilador de lenguaje C.
Bison produce, a partir de un archivo que describe la sintaxis del lenguaje, un archivo de código fuente en C que luego se utiliza en la compilación final.
Flex, de la misma manera, tomando un archivo que describe el léxico del lenguaje, genera un archivo de código fuente que luego es linkeado en el ejecutable final.
Estos dos archivos de código son compilados y linkeados junto con el resto de los fuentes del programa para generar un único ejecutable.

Makefile

Justamente por la complejidad del proceso de compilación, vimos la conveniencia de utilizar una herramienta llamada *make* que permite automatizar la compilación de proyectos como este.
Para ello se crea un archivo de texto llamado *Makefile* que contiene la descripción del proceso de compilación, con cada comando, lo que necesita como entrada y lo que genera como salida. En función de eso y usando las fechas de modificación de los archivos, *make* sabe qué pasos debe ejecutar para no volver a compilar todo si sólo se cambió un archivo fuente, y al mismo tiempo gestiona los llamados a *Bison* y *Flex* para que la compilación se lleve a cabo de manera correcta.

Estructura interna del soft

El programa del compilador consta de los siguientes archivos:
Makefile
Es el archivo necesario para generar el ejecutable utilizando la herramienta *make*.
robcomp.l
Es el archivo que describe el léxico del lenguaje, entrada de *Flex*.
robcomp.y
Es el archivo que describe la sintaxis del lenguaje, entrada de *Bison*.
robcomp.h
Es el archivo que contiene todas los prototipos, defines y tipos de datos utilizados en el programa, para ser incluído en el resto de los fuentes.
main.c
Es el archivo que contiene la función *main*, y otras accesorias, como las de inicialización y manejo de archivos de configuración.
cineinv.c
Es el archivo que contiene todas las funciones de cinemática, tanto directa como inversa.
gentray.c
Es el código fuente de las funciones de generación de trayectorias y perfiles de velocidad.

Carpeta *iniparser3.0b*
Es una librería utilizada para leer archivos de configuración .ini, que usamos para poder cambiar fácilmente los parámetros del robot y del controlador.

Chequeo de errores y warnings

El compilador verifica automáticamente la sintaxis, los parámetros incorrectos y que los comandos programados no sobrepasen la zona de trabajo del robot. De la misma forma, se verifica que en ningún momento se sobrepasen los valores máximos de velocidad y aceleración o desaceleración de cada articulación.
En el caso de manifestarse una de las mencionadas condiciones, se informa mediante la consola por medio de un mensaje descriptivo, encabezado por el número de línea en donde se encontró el error o donde se generó la advertencia.

Archivos generados

Los archivos generados por el compilador, contienen valores separados por tabulaciones y utilizan una línea por unidad de tiempo del controlador.

xyz.txt
Consta de las posiciones *x y z* del extremo del robot, en función del tiempo, para depuración y visualización en la simulación.

q123.txt
Incluye las posiciones de cada articulación (*q1*, *q2* y *q3*), para depuración y visualización en la simulación.

dq123.txt
Incluye las velocidades escaladas de cada articulación ($\Delta q1$, $\Delta q2$ y $\Delta q3$) y el valor del byte de salidas digitales. Este archivo está pensado para enviarlo o grabarlo en el firmware del controlador, ya que los valores de velocidad de articulación son las consignas de cada módulo driver de salida y el valor del byte de salidas digitales es el que debe escribirse en el puerto correspondiente del controlador.

do.txt
Contiene los valores de las salidas digitales, para depuración y visualización en la simulación.

Modo de uso del compilador RT

Para utilizar el compilador, ejecutar:
./robcomp < archivo_programa
donde *archivo_programa* es el nombre del archivo de texto que tiene los comandos. El "<" es para redireccionar el archivo a la stdin del compilador (si no, es posible escribirle los comandos a mano en la consola). La salida stdout es la interpretación de los comandos (se puede redirigir también a un archivo) y por stderr saca los warnings y errores (siempre por consola).

Opcionalmente se le puede pasar el nombre del archivo de configuración *.ini* que se desee utilizar, con la definición de los parámetros del robot y del controlador. Por ejemplo:
./robcomp archivo_configuracion < archivo_programa
Si el archivo de configuración especificado no existe, el compilador lo crea con los valores por defecto. Si no se le especifica qué archivo de configuración utilizar, automáticamente se utiliza el archivo *scara.ini*.

Parámetros por defecto

El compilador lee los parámetros constructivos del robot y del controlador desde el archivo de configuración especificado. De todas formas, nosotros definimos un conjunto de valores por defecto, que son los siguientes:

```
L1 = 40.0           ;Largo brazo 1
L2 = 35.0           ;Largo brazo 2
q1min = -89.0       ;Amplitud mínima de la articulación 1
q1MAX = 89.0        ;Amplitud máxima de la articulación 1
q2min = -160.0      ;Amplitud mínima de la articulación 2
q2MAX = 160.0       ;Amplitud máxima de la articulación 2
L3min = 0.0         ;Amplitud mínima del actuador lineal
L3MAX = 30.0        ;Amplitud máxima del actuador lineal
```

Ejemplos demostrativos de compilación, generación de errores y warnings

Los siguientes comandos son equivalentes, ambos mueven el robot hasta la posición de reposo al 25% de la velocidad:

```
inicio
moveje inicio s25
```

El siguiente comando da error ya que la coordenada 80 está fuera de la zona de trabajo del robot:

```
linea 80 0 40 s50

Linea 1: ERROR: El comando sale de la zona de trabajo.
```

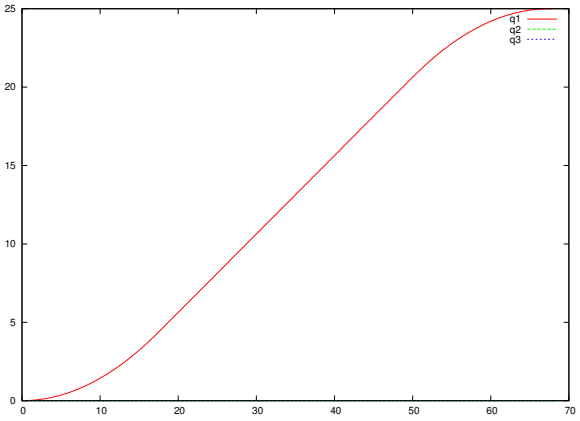
El siguiente comando genera una advertencia de sobrepaso de la velocidad máxima definida para la articulación:

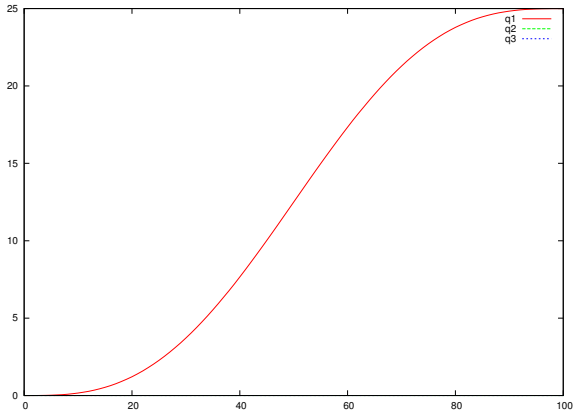
```
moveje_rel 25 0 0 s110

Linea 1: WARNING: Exceso de velocidad en Articulacion 1 (x=74.695
y=6.760 z=0.000 t=1.90)
```

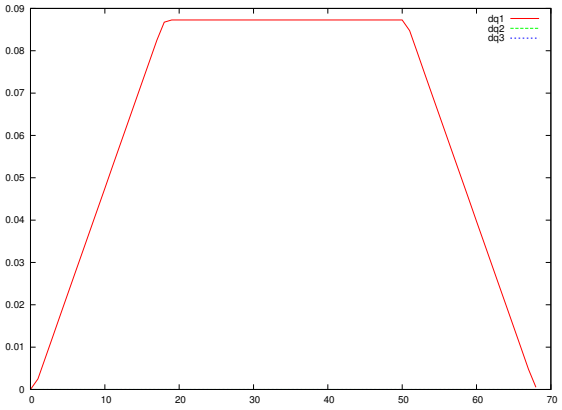
El siguiente comando sale de la zona de trabajo. Se indica la articulación que limitó el movimiento:

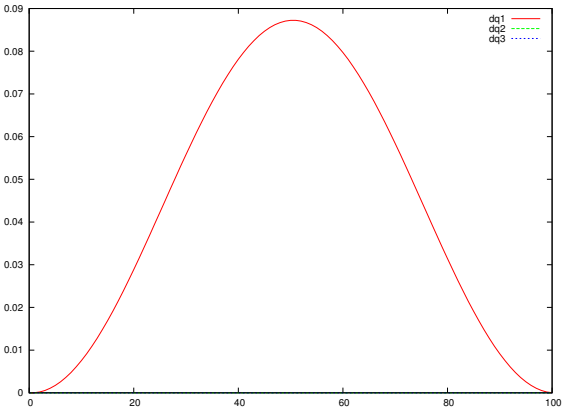
```
moveje 250 0 0 s80

Linea 1: WARNING: Articulacion 1 excede el maximo (x=1.561 y=74.984
z=0.000 t=23.00)
Linea 1: ERROR: El comando sale de la zona de trabajo.
```

Simulación del movimiento del robot como respuesta a programas en lenguaje RT

<table>
<tr><td colspan="2">Ejemplo de los distintos tipos de perfiles de velocidad:</td></tr>
<tr><td>Perfil trapezoidal:
<code>t1</code>
<code>moveje_rel 25 0 0 s100</code></td><td>Perfil sigmoidal:
<code>t2</code>
<code>moveje_rel 25 0 0 s100</code></td></tr>
<tr><td> 

Posición</td><td> 

Posición</td></tr>
<tr><td> 

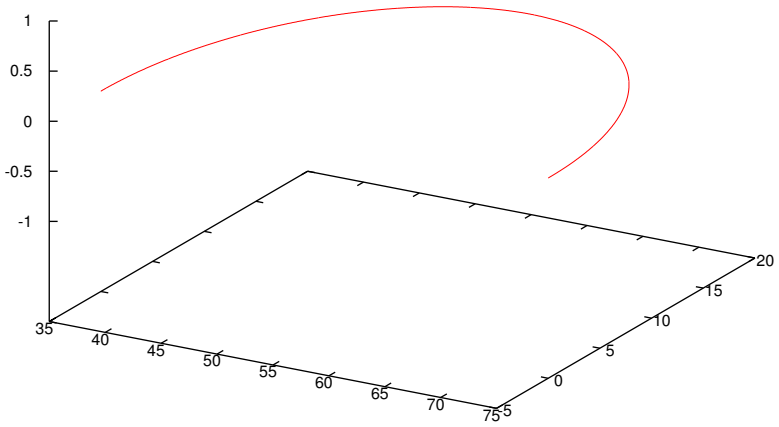
Velocidad</td><td> 

Velocidad</td></tr>
</table>

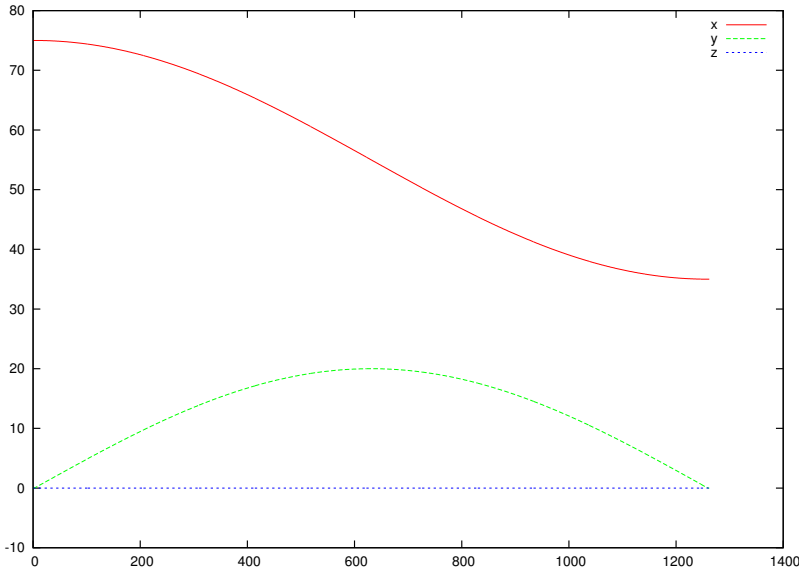
El siguiente comando mueve el robot por una trayectoria circular:

```
circ 55 20 0 35 0 0 s25
```

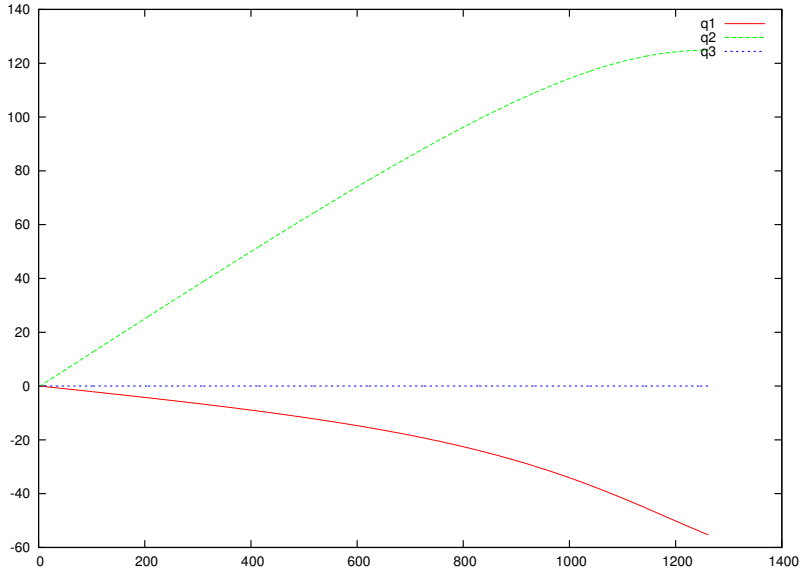
Vista de la trayectoria en el espacio



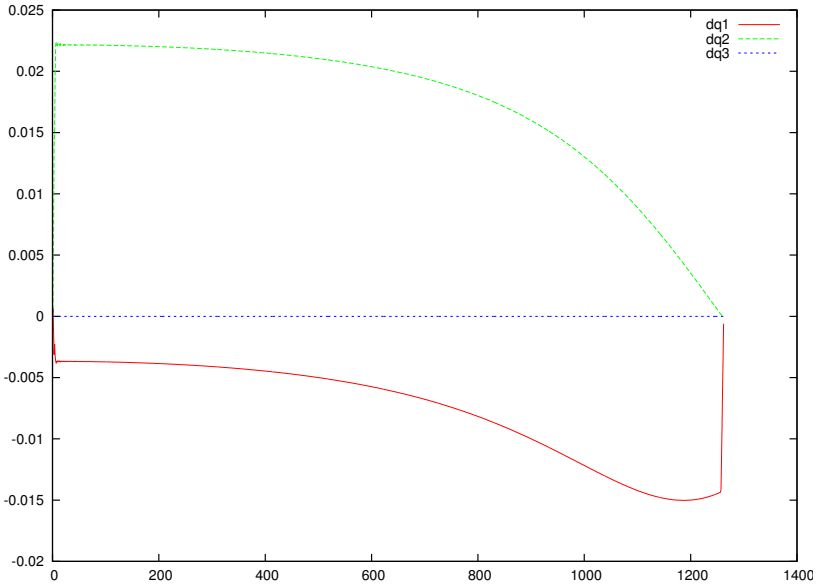
Coordenadas *x y z* en función del tiempo



Articulaciones q_1 q_2 q_3 en función del tiempo



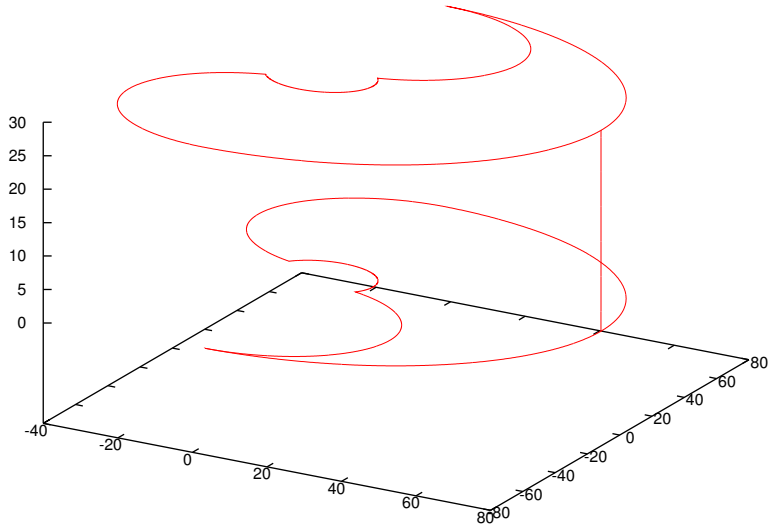
Velocidades de las articulaciones en función del tiempo



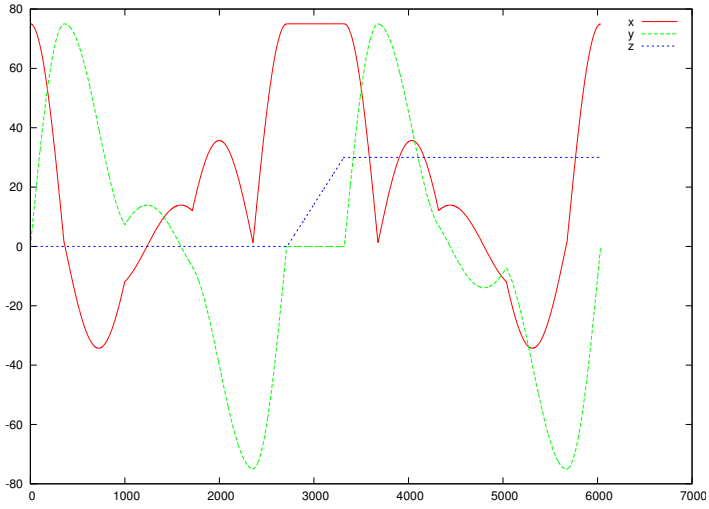
El siguiente programa lleva el extremo por todos los límites de las articulaciones, mostrando la zona de trabajo:

```
moveje 89 0 0 s50
moveje 89 160 0 s50
moveje -89 160 0 s50
moveje -89 0 0 s50
moveje 0 0 0 s50
moveje 0 0 30 s50
moveje 89 0 30 s50
moveje 89 -160 30 s50
moveje -89 -160 30 s50
moveje -89 0 30 s50
moveje 0 0 30 s50
fin
```

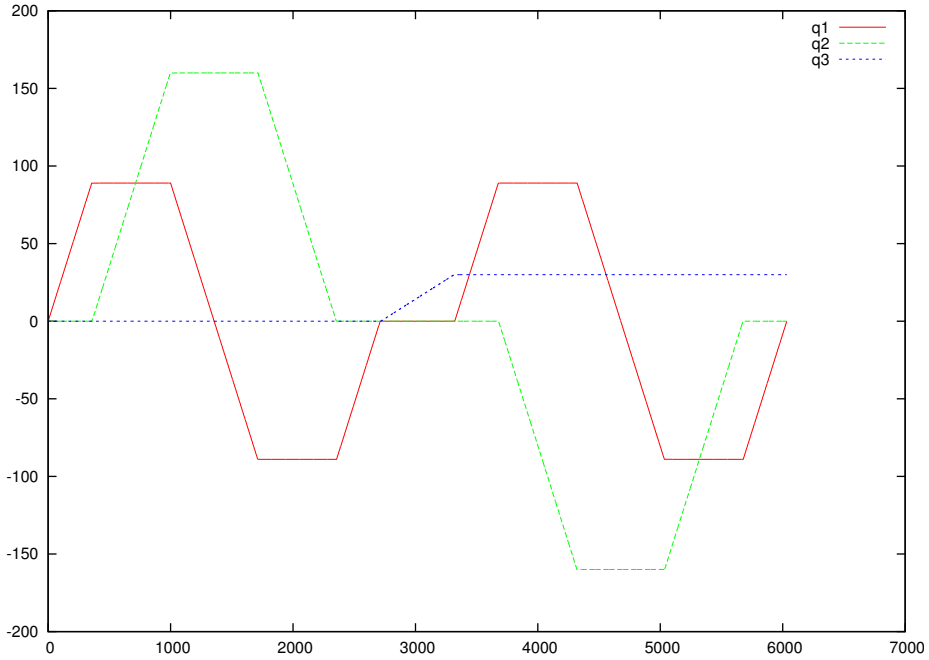
Vista de la trayectoria en el espacio



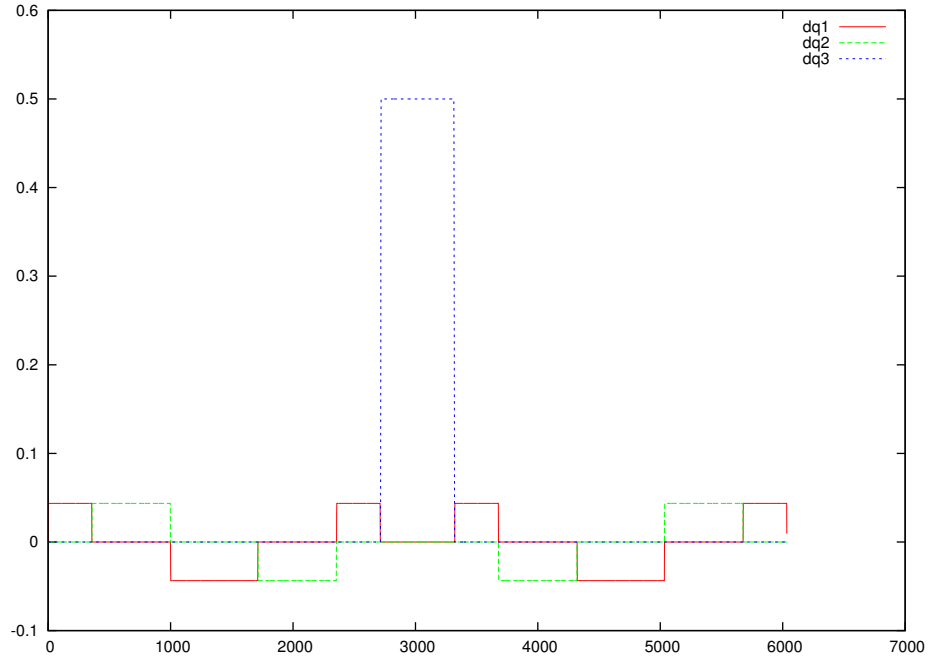
Coordenadas *x y z* en función del tiempo



Articulaciones q_1 q_2 q_3 en función del tiempo



Velocidades de las articulaciones en función del tiempo **(*)**

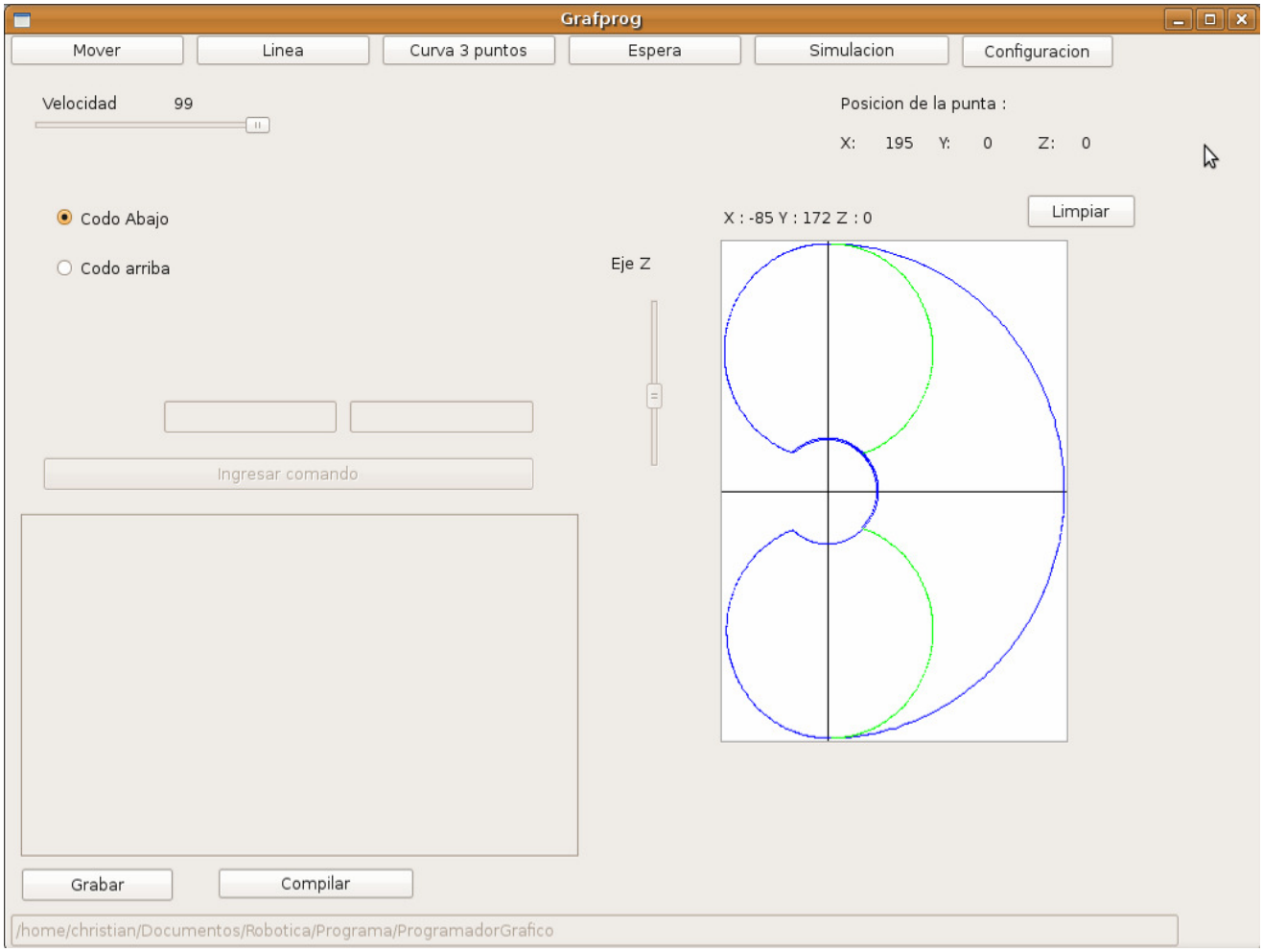


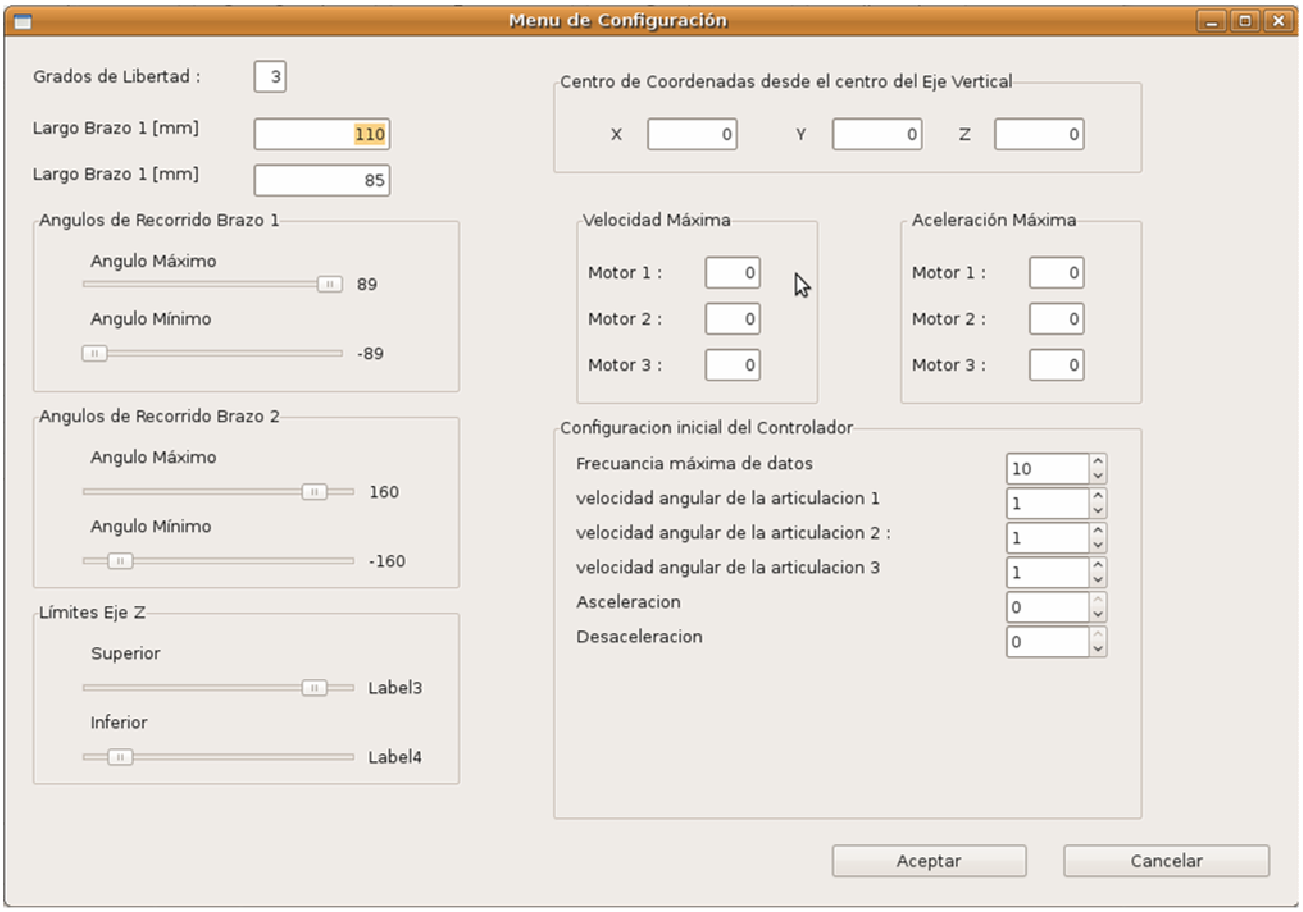
(*) Las velocidades son muy diferentes ya que se encuentran en distintas unidades (rad/s y cm/s, según sean rotaciones o desplazamientos)

Programador Gráfico

Diseñamos un software programador gráfico, para ayudar a escribir el programa en el lenguaje implementado, y también para simular los resultados y verlos gráficamente.

Este soft cuenta básicamente con 2 ventanas, la ventana principal, donde se escribe el programa y se simula el movimiento, y otra ventana de configuración, donde es posible editar los parámetros constructivos tanto del robot como del controlador (el archivo .ini que utiliza el compilador).





Conclusiones

Pudimos desarrollar el compilador de un lenguaje de programación para un robot Scara, así como también diseñar su controlador y simularlo exitosamente. Aún quedan muchas mejoras por realizar, como enriquecer los tipos de movimiento, mejorar el manejo de variables, y ampliar las posibilidades de controlar el flujo del programa (condiciones, saltos, bucles, etc.)

También encontramos durante el desarrollo las dificultades referentes a los excesos en las velocidades y aceleraciones de las articulaciones individuales, al requerir que el robot realice una trayectoria aparentemente no tan exigente. Este problema lo rodeamos estableciendo los controles durante la compilación. Sería deseable, como mejora futura, que el compilador pueda disminuir automáticamente las velocidades para mantener el funcionamiento estable durante toda la trayectoria.