[[1]](#footnote-1)

**Robot SCARA**

**Damián E. Alonso, Andrés L. Di Donato, Federico Morella**  
*Universidad Tecnológica Nacional – Facultad Regional Buenos Aires  
  
21/08/2012*

*Abstract*—El presente trabajo, presentado como trabajo práctico de la material Robótica, tiene por objetivo realizar un análisis completo del robot elegido: el SCARA. Se trabajará sobre diversos detalles, haciendo principalmente énfasis en el estudio de la cinemática y dinámica del robot, como así también en el desarrollo de un compilador que permita generar código para controlar nuestro robot.

*Index Terms*— Robótica, movimiento de robot, control de robot, compilador.

# Introducción

La palabra SCARA corresponde al acrónimo en inglés “*Selective Compliant Articulated Robot Arm”*, es decir “Brazo robótico articulado de obediencia selectiva”.

Fue diseñado en los años ochenta como una alternativa a los robots de ensamble ya existentes. Es muy apto para realizar perforaciones sobre el plano XY [1].

Una de las características fundamentales del robot consiste en su estructura de dos piezas enlazadas, de diseño similar a un brazo humano. De aquí proviene su calificativo de “articulado”. Esta característica permite al brazo extenderse hacia áreas confinadas, así como retraerse para volver a su posición inicial.

Lo anterior constituye una gran ventaja para el transporte de partes, su carga y su descarga (ver Fig. 1) en ámbitos donde el espacio es reducido, o existen obstáculos. Esto representa una gran ventaja frente a los robots cartesianos. Los SCARA pueden además operar más rápidamente que los cartesianos.



Fig. 1: Ejemplo de aplicación del robot SCARA.

# Cinemática directa

El objetivo del análisis cinemático que realizaremos a continuación es el de obtener una relación funcional matemática entre la localización del extremo del robot respecto de un sistema de referencia, y los valores asociados a cada una de sus articulaciones.

Posteriormente emplearemos las ecuaciones de cinemática directa para así poder graficar el espacio de trabajo de nuestro robot.

## Método utilizado

Utilizaremos como método el modelo basado en el producto de matrices de transformación homogéneas.

Consideraremos a nuestro robot como una cadena cinemática conformada por eslabones relacionados por articulaciones. Plantearemos como sistema de referencia la base del robot.

## Algoritmo de Denavit-Hartemberg

Utilizaremos el algoritmo de Denavit-Hartemberg [2] [3] para obtener el modelo cinemático directo.

En la Fig. 2 representamos geométricamente al robot y referenciamos el sistema de referencia (x0,y0,z0) así como los eslabones y las articulaciones.

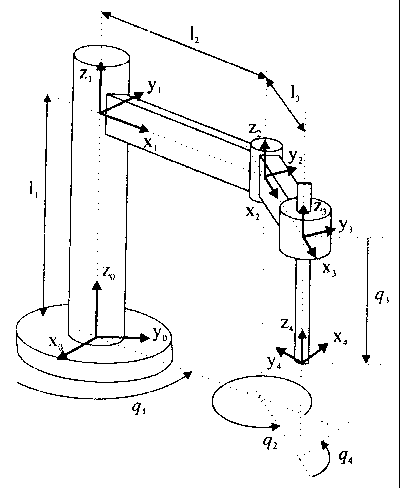


Fig. 2: Esquema de las coordenadas cartesianas adoptadas para el robot.

En base al sistema de referencia elegido, se busca obtener la matriz homogénea para poder dar solución al modelo cinemático directo del sistema. Empleando el Algoritmo de Denavit-Hartemberg, se deben analizar 4 movimientos:

1. Rotación en z.
2. Traslación en z.
3. Rotación en x.
4. Traslación en x

En base a ellos construimos la Tabla 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | θ  (Rotación en z) | d  (Traslación en z) | a  (Traslación en x) | α  (Rotación en x) |
| 0 | 90°+q1 | l1 | 0 | 0 |
| 1 | q2 | 0 | l2 | 0 |
| 2 | 0 | 0 | l3 | 0 |
| 3 | q4 | -q3 | 0 | 0 |

Tabla 1: representación de Denavit-Hartenberg del modelo cinemático

A partir de la tabla anterior, construimos las matrices de transformación homogénea. Utilizamos Matlab para simplificar la operatoria matemática y permitir posteriores simulaciones. El resultado obtenido se muestra en la matriz:

## Espacio de trabajo

Analizaremos el espacio de trabajo de nuestro robot SCARA en dos casos distintos: uno empleando un modelo meramente matemático sin restricciones en su movimiento, y otro más cercano a la realidad, imponiendo ciertas restricciones al movimiento de sus articulaciones.

### Espacio de trabajo sin restricciones.

Idealmente y si no considerásemos las restricciones mecánicas, el espacio de trabajo de un robot SCARA correspondería a un cilindro perfecto.

A los efectos de probar la validez de la matriz de transformación homogénea hallada en el apartado anterior, pasamos a graficar empleando MATLAB el espacio de trabajo del robot punto a punto, considerando que los ángulos pueden barrerse desde 0° a 360°.

Obtenemos así la siguiente representación:

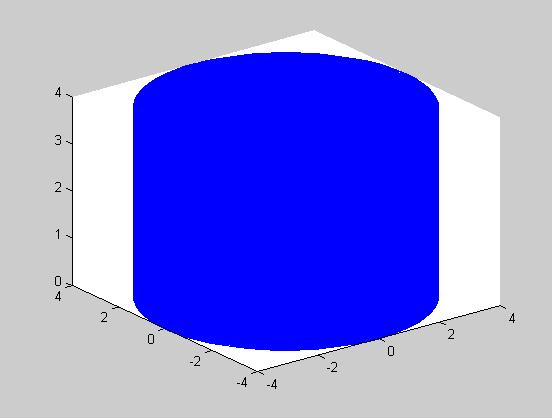


Fig. 3: Espacio de trabajo ideal del robot SCARA, sin considerar restricciones.

Para verificar la validez de las ecuaciones obtenidas, procedemos a repetir el proceso anterior, pero esta vez con menor cantidad de puntos.

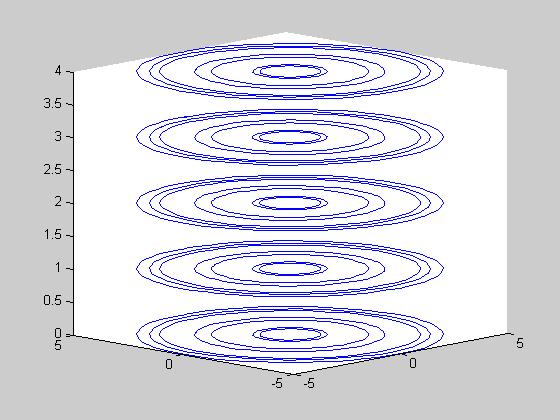


Fig. 4: Espacio de trabajo ideal del robot SCARA graficando menos puntos.

De la figura anterior podemos concluir que el robot se comporta en forma muy alineal. La gráfica anterior fue construida con incrementos constantes de los valores de las articulaciones, mientras que observamos saltos no-lineales en las circunferencias graficadas.

### Espacio de trabajo restringido.

Considerando la construcción física real del robot, acotamos los ángulos que las articulaciones podrían barrer en una representación más realista.

Rotación de q1: de 0° a 270°

Rotación de q2: de -135° a +135°

Repetimos entonces la representación del espacio de trabajo, obteniendo entonces las representaciones observadas en las Fig. 5 y Fig. 6:



Fig. 5: Espacio de trabajo restringido, vista 3D de frente.

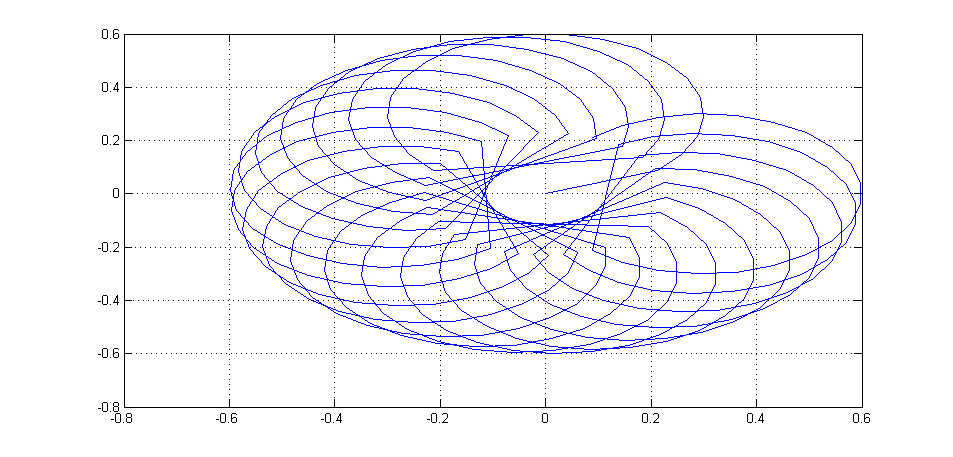


Fig. 6: Espacio de trabajo restringido, vista 2D superior.

Observamos entonces que la representación no corresponde a un cilindro completo sino a un sector cilíndrico.

## Implementación en un DSP

Repetimos los cálculos implementando las ecuaciones anteriores sobre un DSP, el Analog Devices Blackfin BF 533, para iguales rangos de variación de las articulaciones. Simulamos el comportamiento con Visual DSP++, entorno desarrollado por Analog Devices para dicha familia de procesadores. Luego efectuamos la simulación y exportamos los resultados a un archivo binario. Por último abrimos el archivo desde MATLAB para visualizar el resultado obtenido. El gráfico puede observarse en la Fig. 7. El mismo es coherente a los obtenidos con el scrpt de Matlab.

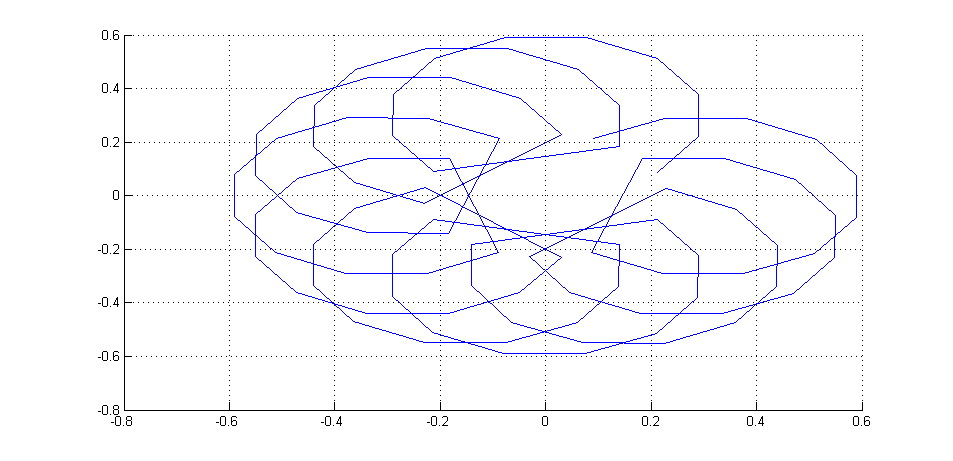


Fig. 7: Espacio de trabajo restringido obtenido por la simulación del DSP.

# Cinemática inversa

La cinemática inversa del manipulador, permite obtener las coordenadas articulares en función de las coordenadas xyz. El problema que surge a la hora de determinar las ecuaciones es que en muchos casos la solución no es única, y en efecto el SCARA presenta dos juegos de soluciones posibles para cada punto xyz válido. Éstas dos soluciones se deben al valor de q2, resultando una solución con 90°<q2<180° y otra con 0°<q2<90°. Para evitar dicha redundancia, fue necesario imponer una restricción de forma tal que 0°<q2<90°. Bajo éstas condiciones, las ecuaciones se determinan geométricamente:



Dado que en nuestro diseño A=B el triángulo ABP es isósceles por lo que:

Por lo tanto:

=

Para determinar el valor de q1 partimos de los ángulos internos de ABP:

Resultando:

En el SCARA, el eje z es independiente de los demás ejes, por lo que q3 se determina en forma directa:

# Análisis Dinámico

La dinámica es la parte de la física que describe la evolución en el tiempo de un sistema físico en relación a las causas que provocan los cambios de estado físico y/o estado de movimiento. En el caso de nuestro robot, el objetivo es determinar que torques, fuerzas y potencias deben realizar las articulaciones de nuestro robot al realizar una determinada trayectoria. Esto es también de gran interés para el dimensionamiento de los motores.

## Modelado físico del robot

Para comenzar con el análisis dinámico del motor, es necesario en primera instancia conocer los siguientes parámetros físicos[[2]](#footnote-2) de cada uno de los eslabones del robot:

* Momentos de inercia.
* Centro de gravedad.
* Masa.

Para tal fin se realizó un modelo del robot en el software T-Flex, el cual calcula el tensor de inercia de cada uno de los eslabones, y a partir de este, es posible obtener los parámetros físicos deseados. El conjunto realizado puede observarse en la Fig. 8.



Fig. 8: Modelo del robot realizado en T-Flex.

Los tensores de inercia contienen información acerca de la masa, los centros de gravedad y las inercias según los ejes de cada eslabón. Los valores obtenidos fueron los siguientes:

Se debe prestar atención a que la unidad de longitud utilizada por el T-Flex es milímetros, por lo que para trabajar en unidades del sistema internacional, es necesario multiplicar a las matrices anteriores por la siguiente matriz de escalamiento:

El centro de gravedad de cada eslabón según su propio sistema de referencia corresponde a la matriz I(3,1:3), y su valor para cada eslabón es:

La masa de cada eslabón corresponde al elemento I(4,4) de la matriz:

## ****Análisis dinámico****

El análisis dinámico se realizó mediante dos algoritmos distintos para luego comparar los resultados obtenidos.

Por un lado se realizó el análisis de Lagrange-Eüler el cual fue implementado mediante un Script de Matlab. Luego se realizó el modelo del robot utilizando el Robotic toolbox de Matlab y utilizando la misma trayectoria se obtuvieron las fuerzas aplicadas a cada eje. Ésta librería utiliza el algoritmo de Newton-Eüler el cual proporciona una mejor aproximación del valor real.

La trayectoria utilizada se observa en la Fig. 9, la cual se realiza en forma cíclica con un período de 1 segundo.

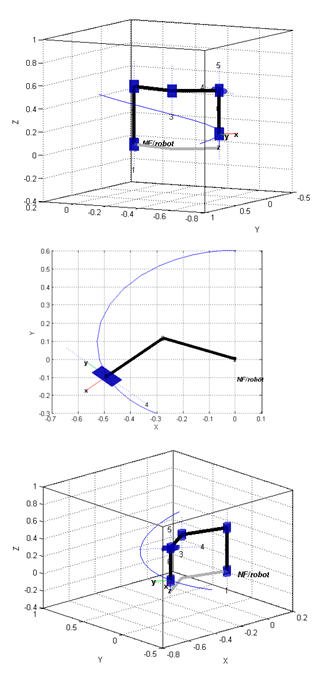


Fig. 9: Trayectoria utilizada para el análisis dinámico.

La Fig. 10 muestra el torque instantáneo y la potencia instantánea en cada eje calculados para ambos algoritmos.

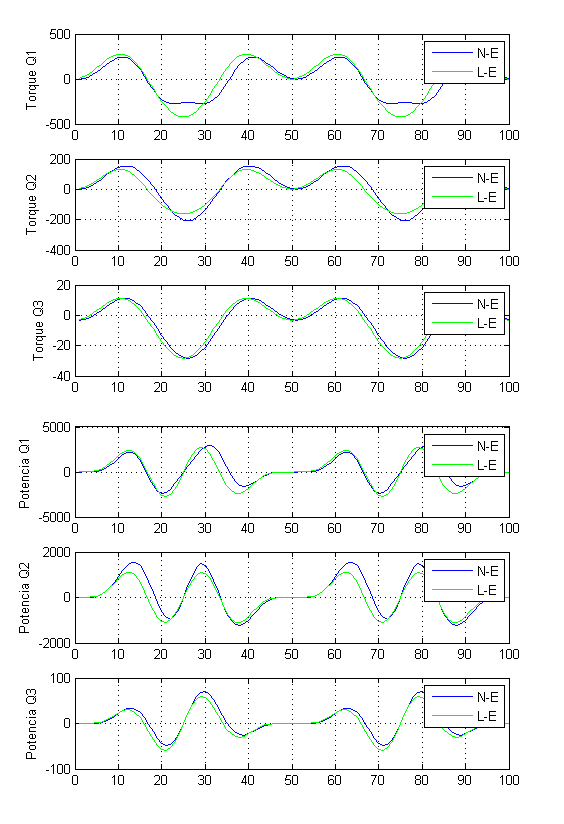


Fig. 10: Gráficos de toque y potencia instantánea comparando los 2 métodos usados.

Como puede observarse, existe una buena correlación entre ambos algoritmos.

En base a los resultados obtenidos se puede calcular los requerimientos para el dimensionamiento de los motores:

* Eje Q1
* Torque[Nm]=243.8
* Torque RMS[Nm]=187.3
* Potencia media[W]=1110
* Eje Q2
* Torque[Nm]=153.6
* Torque RMS[Nm]=114.3
* Potencia media[W]=641.5
* Eje Q3
* Fuerza[N]=11
* Fuerza RMS[N]=12.7
* Potencia media[W]=21

En función de éstos parámetros se pueden dimensionar los reductores y los motores a utilizar.

# Control de los motores

Elegimos para implementar nuestro robot, motores *brushless* (sin escobillas) de corriente continua con sensores de efecto Hall [4].

Para controlar la velocidad de los motores de nuestro robot, proponemos emplear un puente H trifásico, controlado por modulación de ancho de pulsos (PWM).

Replicamos el esquema circuital del mismo en la Fig. 11:

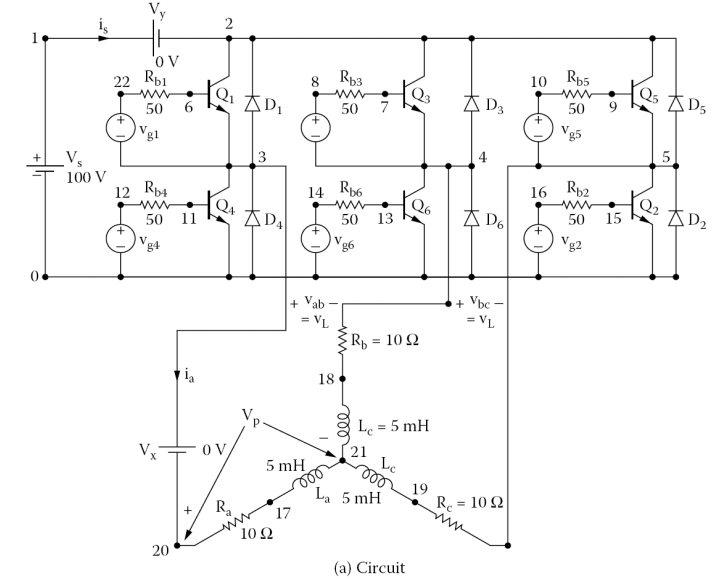


Fig. 11: Puente H trifásico para control de velocidad de motores.

Sin utilizar las señales de control, para garantizar el giro deberíamos aplicar la secuencia de la Fig. 12. Observamos que siempre habrá un par de transistores en estado activo y dos en reposo.

Con la Modulación PWM (Fig. 13), mientras que uno de los dos transistores del par conduce plenamente, el otro lo hará en forma alternada, a la frecuencia del PWM, que resulta bastante mayor que la frecuencia de giro y con un determinado ciclo de actividad.

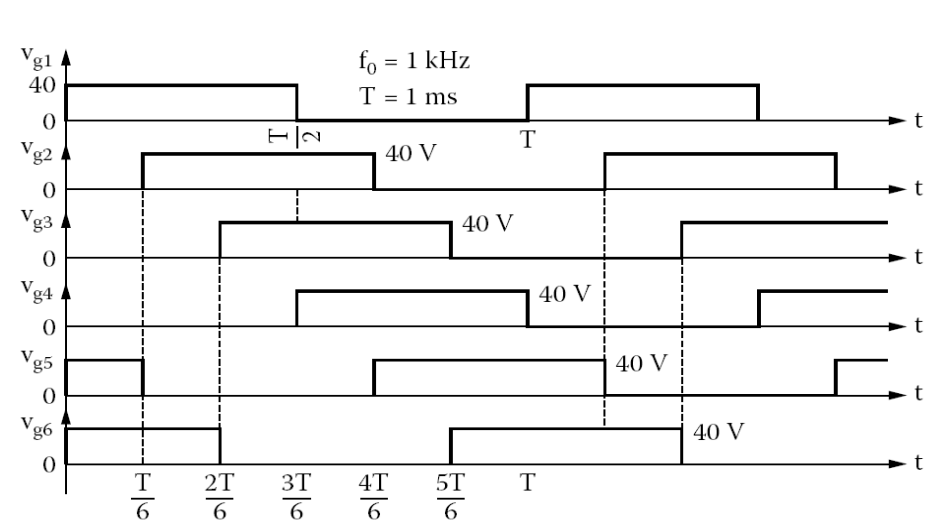


Fig. 12: Secuencia para giro del motor.

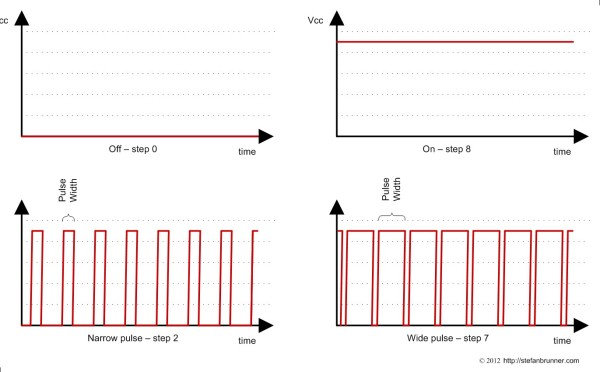


Fig. : Modulación PWM.

## Problemas de conmutación

Sabemos que la conmutación de transistores no es instantánea. En el esquema anterior, podría ocurrir que en los cambios de estado condujesen en forma simultánea dos transistores que interconecten +V con –V, ofreciendo un camino de baja resistencia eléctrica provocando un gran aumento de la corriente y destrucción de dichos transistores.

Esto puede ocurrir cada vez que se produce un cambio de estado, donde debería dejar de conducir en tiempo nulo un transistor y pasar a conducir el siguiente en tiempo nulo.

Por la tecnología de fabricación de dichos componentes y los efectos de almacenamiento de cargas, esto es irrealizable. La solución para evitar los problemas de cortocircuitos en la conmutación es agregar un “tiempo muerto” o “*dead time*” en cada uno de los cambios de estado evitando así el inconveniente antes explicado.

## Implementación del control

Se eligió como implementación la descripción en VHDL del control del dispositivo. Se eligió como plataforma una FPGA Cyclone II de Altera y se utilizó su entorno Quartus II así como el ModelSim.

Para ello, describimos en VHDL cuatro archivos diferentes:

1. Generación de las señales de estímulo del sistema, incluyendo las señales generadas por los sensores de efecto Hall del motor.
2. Generación de la señal de cambio de estado en función de la información brindada por los sensores Hall y generación de la señal de PWM.
3. Máquina de estados para seleccionar qué par de transistores deben activarse.
4. Generación de las señales a aplicar sobre cada uno de los seis transistores constituyentes del puente H.

Los cuatro archivos .vhd se adjuntan al siguiente trabajo.

Utilizando Quartus II, se generó para los archivos 2, 3 y 4 un bloque (“*model*”) con las entradas y salidas asociadas al mismo. Se lo incluye en la Fig. 14.

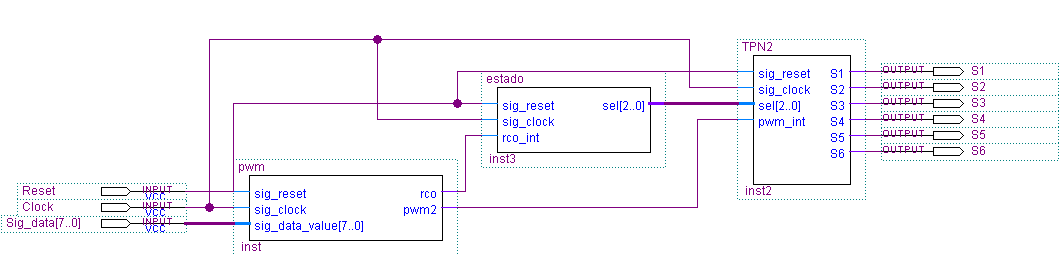


Fig. 14: Bloque del archivo generado.

Finalmente, con cada uno de los modelos generados, se creó un esquemático (“*Block diagram*”) interconectando cada bloque. De dicha interconexión se generó un único archivo .vhd al cual se le agregaron las señales estímulo.

Posteriormente se simuló empleando ModelSim dicho archivo obteniendo los resultados mostrados en la Fig. 15.

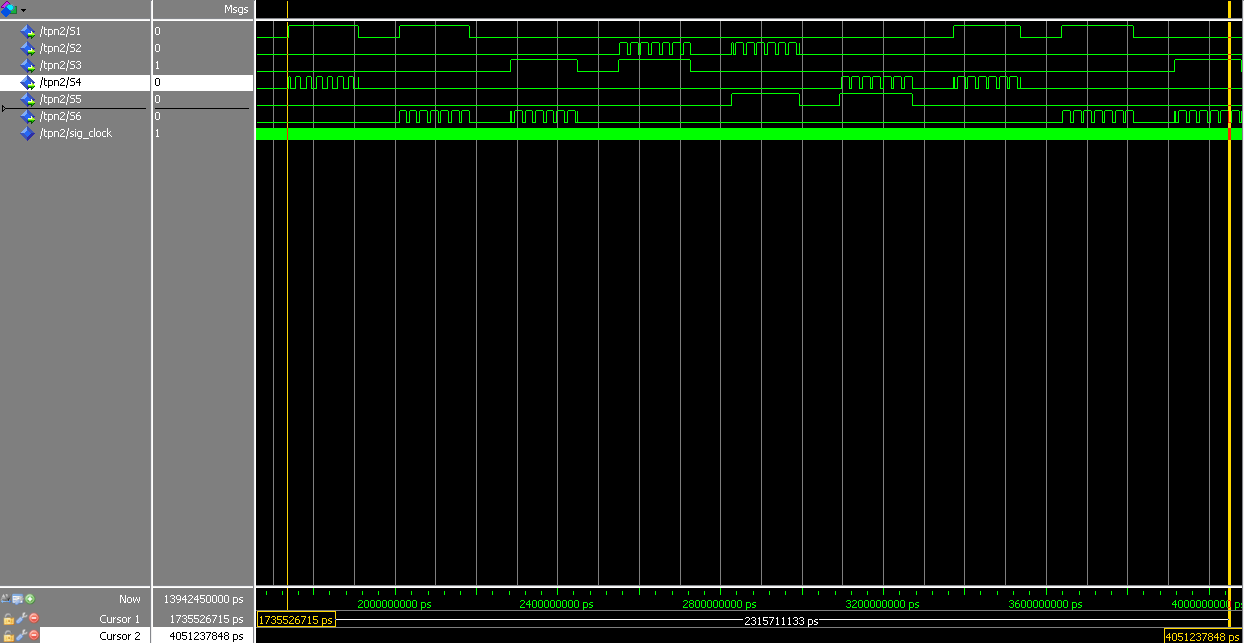


Fig. 15: Simulación del archivo en ModelSim.

Se verifica cómo se cumple el esquema de estados propuesto anteriormente. En la Fig. 16 observamos con más detalle la zona de transiciones.

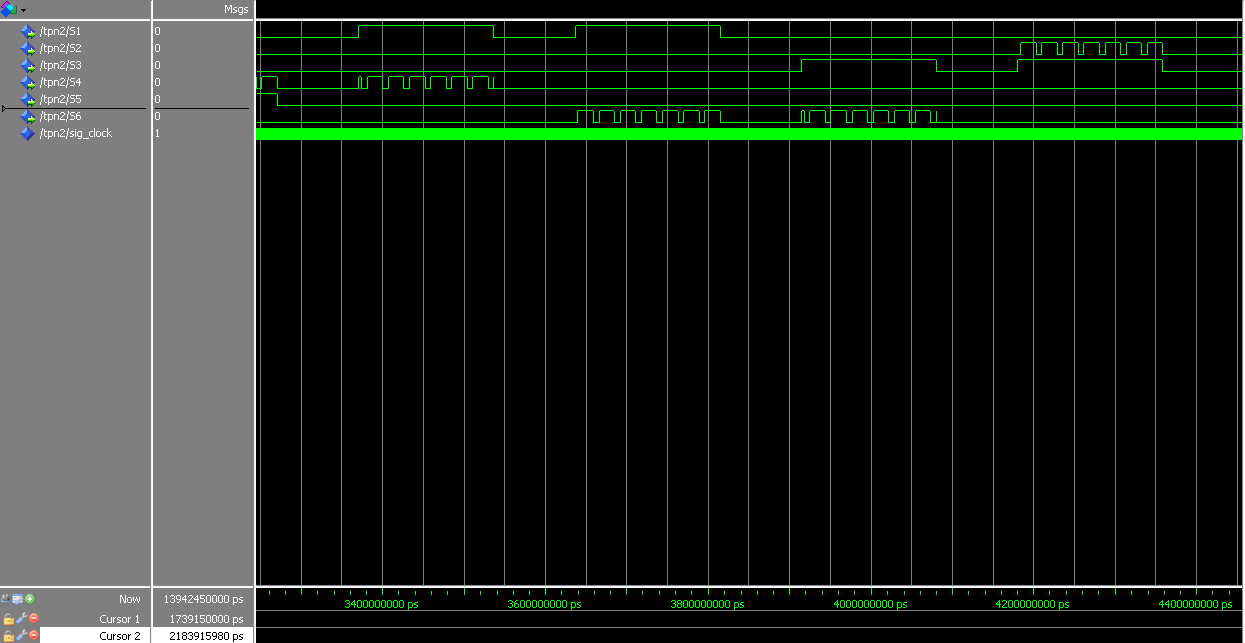


Fig. 16: Detalle de la zona de transiciones.

Observamos que por el tiempo muerto introducido, en ningún momento se produce la “superposición” de accionamiento de dos transistores de la misma rama. De esta manera evitamos los problemas mencionados. Los tiempos mostrados han sido exagerados a los efectos de poder mostrar gráficamente que la descripción VHDL empleada efectivamente soluciona el problema planteado.

# Compilador

ANTLR (“*ANother Tool for Language Recognition”,* en español "otra herramienta para reconocimiento de lenguajes") es una herramienta que opera sobre lenguajes, proporcionando un marco para construir reconocedores ([parsers](http://es.wikipedia.org/wiki/Parser" \o "Parser)), intérpretes, compiladores y traductores de lenguajes a partir de las descripciones gramaticales de los mismos (conteniendo acciones semánticas a realizarse en varios lenguajes de programación) [5]. El entorno de desarrollo utilizado es ANTLRWorks, el cual cuenta con interfaz gráfica que permite el desarrollo de gramáticas para la versión 3.0 o superior de ANTLR. Consiste en una aplicación independiente Java, que se puede ejecutar directamente desde un jar. Para tal fin es necesario contar con una máquina virtual de java.

Es software libre, lo que quiere decir que al descargarlo de la página oficial[[3]](#footnote-3) obtendremos tanto los ficheros compilados \*.class como el código fuente en forma de ficheros \*.java.

ANTLR es capaz de generar un analizador léxico, sintáctico o semántico en varios lenguajes (java, C++ y C#, por ejemplo) a partir de unos ficheros escritos en un lenguaje propio [6]. Dicho lenguaje es básicamente una serie de reglas EBNF y un conjunto de construcciones auxiliares.



Fig. 17: Esquema de funcionamiento de ANTLR

ANTLR genera analizadores pred-LL(k), y él mismo utiliza un analizador pred-LL(k) para leer los ficheros en los que están escritas las reglas EBNF. ANTLR admite acciones en sus reglas, además de otras prestaciones como paso de parámetros, devolución de valores o herencia de gramáticas.

Nuestro objetivo es generar un archivo de instrucciones (script) de Matlab a partir de un archivo de texto, el cual contiene instrucciones propias. Estas últimas han sido creadas tomando como referencia el set de instrucciones *Rapid* del compilador de ABB [7].

Para lograr esto se ha desarrollado un **archivo de gramática** (extensión *.g*), en el cual se indica como se conforma

Los ficheros con los que trabaja ANTLR tienen la terminación \*.g, y se los denomina **ficheros de especificación de gramáticas**. En este archivo se incluye la definición de todas las instrucciones a implementar, incluyendo la estructura que deben tener, los parámetros que debe recibir y lo que devuelve.

## Instrucciones desarolladas

A continuación se describen las instrucciones diseñadas en el compilador:

*1) pause*

Genera una pausa en el movimiento. Invoca a la función homónima de Matlab, por lo que se le pasa como parámetro el tiempo de pausa deseado en segundos. En la Fig. 18 se muestra la estructura de la instrucción.

*2) home*

Ejecuta las instrucciones necesarias para mover al robot a la posición definida como inicial.

*3) movel*

Desplazamiento del extremo del robot desde su posición actual hasta el punto indicado siguiendo una línea recta.

*4) movej*

Desplazamiento del extremo del robot hasta el punto indicado rápidamente, sin garantizar cuál es la trayectoria seguida (no hay coordinación de velocidad entre los distintos ejes del robot).

En la Fig. 18 se observa el esquema de cada instrucción, tal como lo muestra el compilador ANTLR.

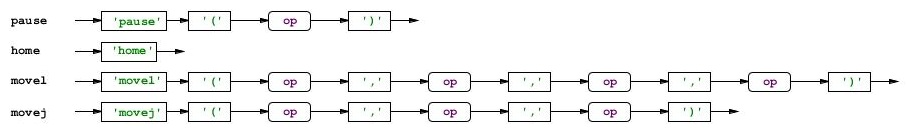


Fig. 18: Instrucciones desarrolladas para el compilador ANTLR

También ha sido necesario diseñar algunos *statements* utilizados conmúnmente en los lenguajes de programación, tales como:

* *if:* para manejo de sentencias condicionales.
* *for:* ciclo que se repiten una determinada canda de veces.
* *while*: ciclo que se repite mientras se cumpla una determinada condición.

Es importante destacar, que el compilador, al principio de cualquier script de Matlab que vaya a crear, incluirá primera una función de inicialización la cual, entre otras cosas, creará a la entidad del robot, definiéndole todos sus parámetros, y ubicando su extremo en la posición inicial.

Entre los archivos que se encuentran adjuntos a este informe, se incluyen el fichero de gramática, como así también un archivo de texto de ejemplo, junto con el script de Matlab generado a partir del mismo.

# Conclusiones

A lo largo del se curso se han realizado los pasos para estudiar un robot de manera completa. Se comenzó por definir el robot a utilizar, luego se estudió la cinemática (tanto directa como inversa) del mismo. Se realizaron scripts y funciones de Matlab para evaluar su comportamiento, para luego implementarlo en DSP. Tras la comparación de los mismos con los de Matlab, se puede decir que han sido óptimos. Después se estudió la dinámica (previo modelización del robot) y nuevamente se evaluó la misma mediante dos algoritmos distintos, obteniendo resultados lógicos y muy similares entre sí. Se implementó el control de los motores en FPGA, y se finalizó el trabajo con el desarrollo de algunas instrucciones en compilador.

Como puede observarse en este informe, se han cumplido con todos los objetivos planteados por la cátedra al comienzo del curso, buscando siempre verificar los resultados obtenidos, y en varios casos obtener una solución por distintos caminos para compararlos entre sí (en el caso de los análisis cinemático y dinámico).

# Archivos complementarios

Junto a este informe se entregan los archivos que complementan la información aquí brindada:

* Scripts y funciones de matlab tanto para el análisis cinemático como dinámico.
* Los archivos para la implementación de la cinemática en DSP (corren bajo el software VisualDSP++ de Analog Devices).
* Los archivos de Quartus y de simulación en ModelSim utilizados para el control de motores mediante FPGA.
* El modelo del robot en T-Flex.
* El fichero de gramática del compilador y un ejemplo de archivo de instrucciones junto con su correspondiente salida (script de Matlab).

# Bibliografía

|  |  |
| --- | --- |
| [1] | L. Westerland, The Extended Arm of Man, A History of the Industrial Robot., 2000. |
| [2] | J. Denavit and R. Hartenberg, Kinematic Synthesis of Linkages, New York: McGraw-Hill, 1964 . |
| [3] | Barrientos, Peñin, Balaguer and Aracil, Fundamentos de robótica, Mc Graw Hill, 2001. |
| [4] | D. W. Hart, Introduction to Power Electronics, Prentice-Hall, 1997. |
| [5] | Wikipedia, ANTLR, 2012. |
| [6] | E. J. G. Cota, Guía práctica de ANTLR 2.7.2, Sevilla, España, 2003. |
| [7] | ABB, Operating Manual: Introduction to Rapid (RobotWare 5.0), Västerås, Suecia, 2007. |

1. [↑](#footnote-ref-1)
2. Cada uno de los parámetros se calcula para cada eslabón de manera individual y aislada, respecto a su propio sistema de referencia. [↑](#footnote-ref-2)
3. Página web del ANTLR: *http://www.antlr.org* [↑](#footnote-ref-3)