

5 DEFINICIÓN DEL ALGORITMO DE CONTROL MEDIANTE EL PLC

5.1 FUNDAMENTOS DE LA TEORÍA DEL CONTROL DE BRAZOS ROBÓTICOS.

INTRODUCCIÓN.

La cinemática del robot estudia su movimiento con respecto a un sistema de referencia. Así, la cinemática se interesa por la descripción analítica del movimiento espacial del robot como una función del tiempo, y en particular por las relaciones entre la posición y la orientación del extremo final del robot con los valores que toman sus coordenadas articulares. Existen dos problemas fundamentales para resolver la cinemática del robot. El primero de ellos se conoce como el problema cinemático directo, y consiste en determinar cuál es la posición y orientación del extremo final del robot, con respecto a un sistema de coordenadas que se toma como referencia, conocidos los valores de las articulaciones y los parámetros geométricos de los elementos del robot. El segundo denominado problema cinemático inverso, resuelve la configuración que debe adoptar el robot dada una posición y orientación de su extremo final. En la Tabla 5.1 se muestra un diagrama del problema cinemático descrito.

Jaques Denavit y Richard S. Hartenberg propusieron un método sistemático para descubrir y representar la geometría espacial de los elementos de una cadena cinemática, y en particular de un robot, con respecto a un sistema de referencia fijo. Este método utiliza una matriz de transformación homogénea para describir la relación espacial entre dos elementos rígidos adyacentes, reduciéndose el problema cinemático directo a encontrar una matriz de transformación homogénea (4 X 4) que relacione la localización espacial del robot con respecto al sistema de coordenadas de su base.

Tabla 5.1. Diagrama entre cinemática directa e inversa.

Valor de las coordenadas articulares ($q_0, q_1, q_2, \dots, q_n$)	Cinemática directa \rightarrow \leftarrow Cinemática inversa	Posición y orientación del extremo del robot ($x, y, z, \alpha, \beta, \gamma$)
---	---	--

Por otra parte, la cinemática del robot trata también de encontrar las relaciones entre las velocidades del movimiento de las articulaciones y las del extremo. Esta relación viene dada por el modelo diferencial expresado mediante la matriz Jacobiana.

El movimiento relativo en las articulaciones resulta en el movimiento de los elementos que posicionan la mano del robot en una orientación deseada. En la mayoría de las aplicaciones de robótica, se está interesado en la descripción espacial del efector final del manipulador con respecto a un sistema de coordenadas de referencia fija.

La cinemática del brazo del robot trata con el estudio analítico de la geometría del movimiento de un robot con respecto a un sistema de coordenadas de referencia fijo como una función del tiempo sin considerar las fuerzas-momentos que originan dicho movimiento. Así pues, trata con la descripción analítica del desplazamiento espacial del robot como función del tiempo, en particular las relaciones entre variables espaciales de tipo de articulación y la posición y orientación del efector final del robot.

5.1.1 CINEMÁTICA DIRECTA.

EL PROBLEMA CINEMÁTICO DIRECTO.

Se utiliza fundamentalmente el álgebra vectorial y matricial para representar y describir la localización de un objeto en el espacio tridimensional con respecto a un sistema de referencia fijo. Dado que un robot se puede considerar como una cadena cinemática formada por objetos rígidos o eslabones unidos entre sí mediante articulaciones, se puede establecer un sistema de referencia fijo, situado en la base del robot, y describir la localización de cada uno de los eslabones con respecto a dicho sistema de referencia usando parámetros de transformación como se puede observar en la Figura 5.1. De esta forma, el problema cinemático directo se reduce a encontrar una matriz homogénea de transformación T que relacione la posición y orientación del extremo del robot respecto del sistema de referencia fijo situado en la base del mismo. Esta matriz T será función de las coordenadas articulares.

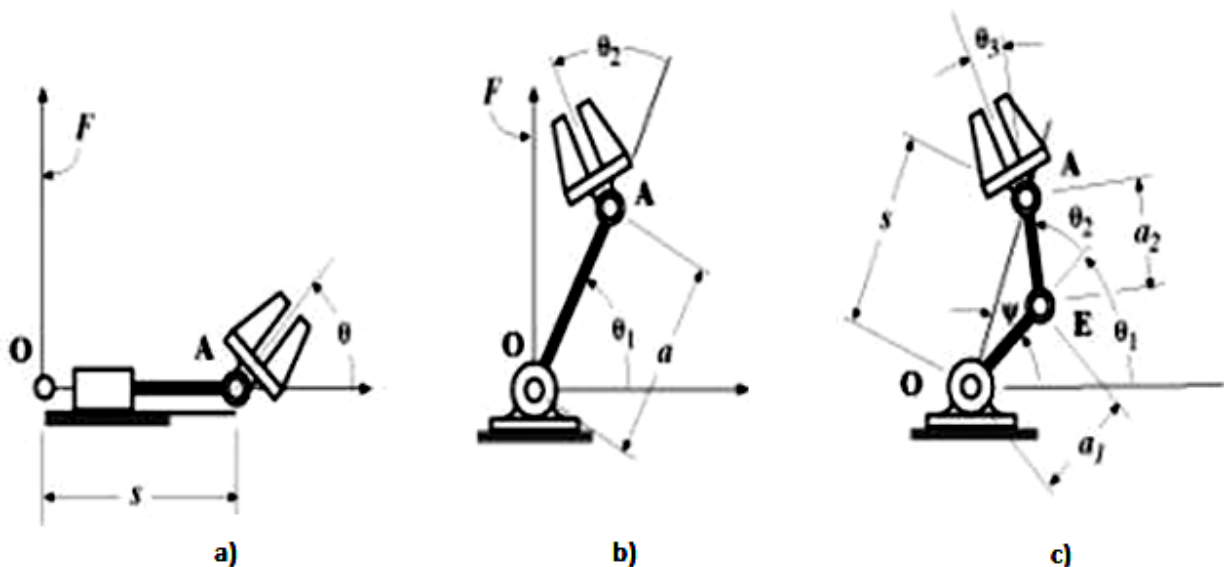


Figura 5.1. Articulaciones referidas a un sistema O situado en la base del robot y sus parámetros que los relacionan para: a) 1 articulación, b) 2 articulaciones, c) 3 articulaciones giratorias. (Barrientos, 2007)

SOLUCIÓN DEL PROBLEMA CINEMÁTICO DIRECTO MEDIANTE MATRICES DE TRANSFORMACIÓN HOMOGÉNEA.

La solución del problema cinemático directo consiste en encontrar las relaciones que permiten conocer la localización espacial del extremo del robot a partir de los valores de sus coordenadas articulares.

Para robots de más de 2 grados de libertad puede plantearse un método sistemático basado en la utilización de las matrices de transformación homogénea.

En general, un robot de n grados de libertad está formado por n eslabones unidos por n articulaciones, de forma que cada par articulación-eslabón constituye un grado de libertad. A cada eslabón se le puede asociar un sistema de referencia solidario a él y, utilizando las transformaciones homogéneas, es posible representar las rotaciones y traslaciones relativas entre los distintos eslabones que componen el robot.

Normalmente, la matriz de transformación homogénea que representa la posición y orientación relativa entre los sistemas asociados a dos eslabones consecutivos del robot se suele denominar ${}^i A_{i+1}$. Así pues, ${}^0 A_1$ describe la posición y orientación del sistema de referencia solidario al primer eslabón con respecto al sistema de referencia solidario a la base, ${}^1 A_2$ describe la posición y orientación del segundo eslabón respecto del primero, etc. Del mismo modo, denominando a las matrices resultantes del producto de las matrices ${}^i A_{i+1}$ con i desde 1 hasta k , se puede representar de forma total o parcial la cadena cinemática que forma el robot. Así, por ejemplo, la posición y orientación del sistema solidario con el segundo eslabón del robot con respecto al sistema de coordenadas de la base se puede expresar mediante la matriz ${}^0 A_2$:

$${}^0 A_2 = {}^0 A_1 \cdot {}^1 A_2$$

De manera análoga, la matriz ${}^0 A_3$ representa la localización del sistema del tercer eslabón:

$${}^0 A_3 = {}^0 A_1 \cdot {}^1 A_2 \cdot {}^2 A_3$$

Cuando se consideran todos los grados de libertad, a la matriz ${}^0 A_n$ se le suele denominar T . Así, dado un robot de seis grados de libertad, se tiene que la posición y orientación del eslabón final vendrá dada por la matriz T :

$$T = {}^0 A_6 = {}^0 A_1 \cdot {}^1 A_2 \cdot {}^2 A_3 \cdot {}^3 A_4 \cdot {}^4 A_5 \cdot {}^5 A_6$$

Aunque para descubrir la relación que existe entre dos elementos contiguos se puede hacer uso de cualquier sistema de referencia ligado a cada elemento, la forma habitual que se suele utilizar en robótica es la representación de Denavit-Hartenberg.

MÉTODO DE DENAVIT-HARTENBERG

En 1955, Denavit-Hartenberg propusieron un método matricial que permite establecer de manera sistemática un sistema de coordenadas (S_i) ligado a cada eslabón i de una cadena articulada, pudiéndose determinar a continuación las ecuaciones cinemáticas de la cadena completa.

Según la representación Denavit-Hartenberg, escogiendo adecuadamente los sistemas de coordenadas asociados para cada eslabón, será posible pasar de uno al siguiente mediante 4 transformaciones básicas que dependen exclusivamente de las características geométricas del eslabón.

Estas transformaciones básicas consisten en una sucesión de rotaciones y traslaciones que permitan relacionar el sistema de referencia del elemento i con el sistema del elemento $i-1$. Las transformaciones en cuestión son las siguientes:

1. Rotación alrededor del eje Z_{i-1} un ángulo θ_i .
2. Traslación a lo largo de Z_{i-1} una distancia d_i ; vector $d_i(0,0,d_i)$.

3. Traslación a lo largo de X_i una distancia a_i ; vector $a_i (a_i, 0, 0)$
4. Rotación alrededor del eje X_i , un ángulo α_i .

Dado que el producto de matrices no es conmutativo, las transformaciones se han de realizar en el orden indicado. De este modo se tiene que:

$${}^{i-1}A_i = T(z, \theta_i) \cdot T(0, 0, d_i) \cdot T(a_i, 0, 0) \cdot T(X, \alpha_i)$$

Y realizando el producto de matrices, respetando el orden de las matrices de transformación de la ecuación anterior, se tiene la matriz de transformación homogénea genérica:

$$A_i = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & 0 \\ s\theta_i & c\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_i & -s\alpha_i & 0 \\ 0 & s\alpha_i & c\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} c\theta_i & c\alpha_i s\theta_i & s\alpha_i s\theta_i & a_i c\theta_i \\ s\theta_i & c\alpha_i c\theta_i & -s\alpha_i c\theta_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

donde α_i, a_i , son los parámetros D-H del eslabón i dependen únicamente de las características geométricas de cada eslabón y de las articulaciones que le unen con el anterior y siguiente.

De este modo, basta con identificar dichos parámetros para obtener las matrices A y relacionar así todos y cada uno de los eslabones del robot. Como se ha indicado, para que la matriz A_i , relacione los sistemas S_{i-1} y S_i , es necesario que los sistemas se hayan escogido de acuerdo a unas determinadas normas. Éstas, junto con la definición de los 4 parámetros de Denavit-Hartenberg, conforman el siguiente algoritmo para la resolución del problema cinemático directo.

ALGORITMO DE DENAVIT- HARTENBERG PARA LA OBTENCIÓN DEL MODELO.

1. Numerar los eslabones comenzando con 1 (primer eslabón móvil de la cadena) y acabando con n (último eslabón móvil). Se numera como eslabón 0 a la base fija del robot.
2. Numerar cada articulación comenzando por 1 (la correspondiente al primer grado de libertad y acabando en n).
3. Localizar el eje de cada articulación. Si ésta es rotativa, el eje será su propio eje de giro. Si es prismática, será el eje a lo largo del cual se produce el desplazamiento.
4. Para i de 0 a $n - 1$, situar el eje Z_i , sobre el eje de la articulación $i + 1$.
5. Situar el origen del sistema de la base S_0 en cualquier punto del eje Z_0 . Los ejes X_0 e Y_0 se sitúan de modo que formen un sistema dextrógiro con Z_0 .

6. Para i de 1 a $n - 1$, situar el sistema S_i (solidario al eslabón i) en la intersección del eje Z_i con la línea normal común a Z_{i-1} y Z_i . Si ambos ejes se cortasen se situaría S_i en el punto de corte. Si fuesen paralelos S_i se situaría en la articulación $i + 1$.
7. Situar X_i en la línea normal común a Z_{i-1} y Z_i .
8. Situar Y_i de modo que forme un sistema dextrógiro con X_i y Z_i .
9. Situar el sistema S_n en el extremo del robot de modo que Z_n coincida con la dirección de Z_{n-1} y X_n sea normal a $Z_{n-1} \vee Z_n$.
10. Obtener θ_i como el ángulo que hay que girar en torno a Z_{i-1} para que X_{i-1} y X_i queden paralelos.
11. Obtener d_i como la distancia, medida a lo largo de Z_{i-1} , que habría que desplazar para que X_i y X_{i-1} quedasen alineados.
12. Obtener a_i como la distancia medida a lo largo de X_i (que ahora coincidiría con X_{i-1}) que habría que desplazar el nuevo S_{i-1} para que su origen coincidiese con S_i .
13. Obtener α_i como el ángulo que habría que girar sobre X_i (que ahora coincidiría con X_{i-1}), para que el nuevo S_{i-1} coincidiese totalmente con S_i .
14. Obtener las matrices de transformación ${}^{i-1}A_i$.
15. Obtener la matriz de transformación que relaciona el sistema de la base con el del extremo del robot

$$T = {}^0A_n = {}^0A_1 \cdot {}^1A_2 \dots {}^{n-1}A_n.$$
16. La matriz T define la orientación (submatriz de rotación) y posición (submatriz de traslación) del extremo referido a la base en función de las n coordenadas articulares.

Es el ángulo que forman los ejes X_{i-1} y X_i medido en un plano perpendicular al eje, utilizando la regla de la mano derecha. Se trata de un parámetro variable en articulaciones giratorias.

Es la distancia a lo largo del eje Z_{i-1} desde el origen del sistema de coordenadas $(i - 1)$ — hasta la intersección del eje Z_{i-1} con el eje Z_i . Se trata de un parámetro variable en articulaciones prismáticas.

Es a la distancia a lo largo del eje X_i que va desde la intersección del eje Z_{i-1} con el eje Z_i hasta el origen del sistema i -ésimo, en el caso de articulaciones giratorias. En el caso de articulaciones prismáticas, se calcula como la distancia más corta entre los ejes Z_{i-1} y Z_i .

Es el ángulo de separación del eje z_{i-1} y el eje z_i , medido en un plano perpendicular al eje z_{i-1} , utilizando la regla de la mano derecha.

Una vez obtenidos los parámetros DH, el cálculo de las relaciones entre los eslabones consecutivos del robot es inmediato, ya que vienen dadas por las matrices A, que se calculan según la expresión general. Las relaciones entre eslabones no consecutivos vienen dadas por las matrices T que se obtienen como producto de un conjunto de matrices A.

Tabla 5.2. Parámetros de Denavit-Hartenberg para el Scorbot-ER V Plus				
Art.	a_i	d_i	α_i	θ_i
1	0	l_1	90°	$\theta_1 (0^\circ)$
2	l_2	0	0	$\theta_2 (0^\circ)$
3	l_3	0	0	$\theta_3 (0^\circ)$
4	0	l_4	90°	$\theta_4 (90^\circ)$
5	0	l_5	0	$\theta_5 (0^\circ)$

5.1.2 CINEMÁTICA INVERSA.

El objetivo del problema cinemático inverso consiste en encontrar los valores que deben adoptar las coordenadas articulares del robot $\mathbf{q} = (q_1, q_2, \dots)$ para que su extremo se posicione y oriente según una determinada localización espacial.

Así, es posible abordar el problema cinemático directo de una manera sistemática, a partir de la utilización de matrices de transformación homogéneas e independientemente de la configuración del robot. Sin embargo no ocurre lo mismo con el problema cinemático inverso, siendo el procedimiento de obtención de las ecuaciones fuertemente dependiente de la configuración del robot.

Se han desarrollado algunos procedimientos genéricos susceptibles de ser programados, de modo que una computadora pueda, a partir del conocimiento de la cinemática del robot (con sus parámetros de DH, por ejemplo) obtener la n-upla de valores articulares que posicionan y orientan su extremo. El inconveniente de estos procedimientos es que se trata de métodos numéricos iterativos, cuya velocidad de convergencia e incluso su convergencia en sí no está siempre garantizada.

A la hora de resolver el problema cinemático inverso es mucho más adecuado encontrar una solución cerrada. Esto es, encontrar una relación matemática explícita de la forma:

$$q_k = F_k(x, y, z, \alpha, \beta, \gamma)$$

$$K = 1 \dots n \text{ (grados de libertad)}$$

Este tipo de solución presenta, entre otras, las siguientes ventajas:

- En muchas aplicaciones, el problema cinemático inverso ha de resolverse en tiempo real (por ejemplo, en el seguimiento de una determinada trayectoria). Una solución de tipo iterativo no garantiza tener la solución en el momento adecuado.

- Al contrario de lo que ocurría en el problema cinemático directo, con cierta frecuencia la solución del problema cinemático inverso no es única; existiendo diferentes n-uplas (q_1, \dots) que posicionan y orientan el extremo del robot del mismo modo. En estos casos una solución cerrada permite incluir determinadas reglas o restricciones que aseguren que la solución obtenida sea la más adecuada posible.

No obstante, a pesar de las dificultades comentadas, la mayor parte de los robots poseen cinemáticas relativamente simples que facilitan en cierta medida la solución de su problema cinemático inverso. Por ejemplo si se consideran solamente los tres primeros grados de libertad de muchos robots, estos tienen una estructura planar, lo que significa que los tres primeros elementos quedan contenidos en un plano. Esta circunstancia facilita la resolución del problema. Asimismo, en muchos robots se da la circunstancia de que los últimos tres grados de libertad, dedicados fundamentalmente a orientar el extremo del robot, correspondan a giros sobre los ejes que se cortan en un punto.

De nuevo, esta situación facilita el cálculo de la n-upla (q_1, \dots) correspondiente a la posición y orientación deseadas. Por lo tanto, para los casos citados y otros, es posible establecer ciertas pautas generales que permitan plantear y resolver el problema cinemático inverso de una manera sistemática.

Los métodos geométricos permiten tener normalmente los valores de las primeras variables articulares, que son las que consiguen posicionar el robot. Para ello utilizan relaciones trigonométricas y geométricas sobre los elementos del robot. Se suele recurrir a la solución de triángulos formados por los elementos y articulaciones del robot, como es mostrado en la Figura 5.2.

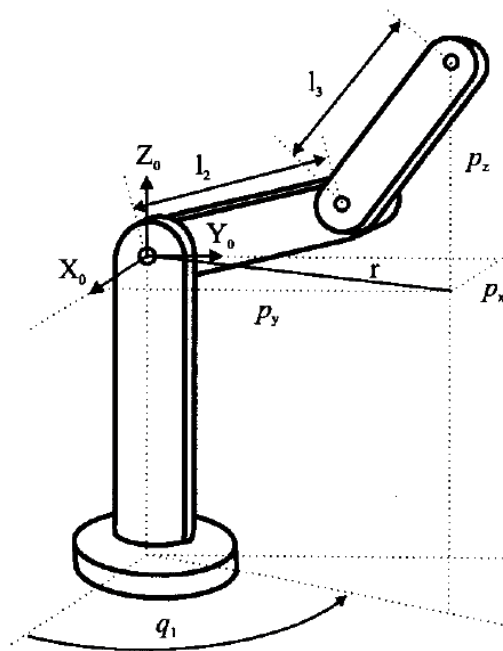


Figura 5.2. Parámetros para la cinemática inversa usando el método geométrico. (Barrientos, 2007)

Por último, si se consideran robots con capacidad de posicionar y orientar su extremo en el espacio, esto es, robots con 6 grados de libertad, el método de desacoplamiento cinemático permite, para determinados tipos de robots, resolver los primeros grados de libertad, dedicados al posicionamiento, de una manera independiente a la resolución de los últimos

grados de libertad, dedicados a la orientación. Cada uno de estos dos problemas simples podrá ser tratado y resuelto por cualquier procedimiento.

SOLUCIÓN 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 solamente 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 solución del problema cinemático inverso de un robot con 3 grados de libertad de rotación (estructura típica articular) como el que se muestra en la Figura 5.3. El dato de partida son las coordenadas (P_x, P_y) , referidas a (r, p_z) en las que se requiere posicionar su extremo.

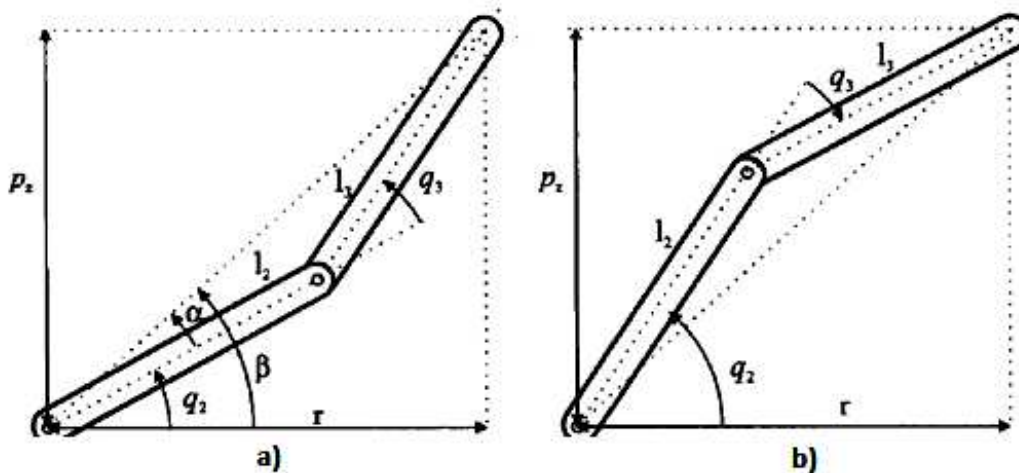


Figura 5.3. Configuración de parámetros para: a) codo arriba b) codo abajo. (Barrientos, 2007)

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_1 se obtiene inmediatamente como:

$$q_1 = \tan^{-1} \left(\frac{P_x}{P_y} \right)$$

Considerando ahora únicamente los dos elementos 2 y 3 que están situados en un plano y utilizando el teorema del coseno, se tendrá:

$$r^2 = (P_x)^2 + (P_y)^2$$

$$r^2 + (P_x)^2 = (l_2)^2 + (l_3)^2 + 2l_2l_3 \cos(q_2)$$

$$\cos(q_3) = \frac{(P_x)^2 + (P_y)^2 + (P_z)^2 - (l_2)^2 - (l_3)^2}{2l_2 l_3}$$

Esta expresión permite obtener q_1 en función del vector de posición del extremo P. No obstante, por motivos de ventajas computacionales, es más conveniente utilizar la expresión del arco tangente en lugar del arco seno.

Puesto que:

$$\text{sen}(q_3) = \pm\sqrt{1 - \cos^2 q_3}$$

se tendrá que:

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

con

$$\cos(q_3) = \frac{(P_x)^2 + (P_y)^2 + (P_z)^2 - (l_2)^2 - (l_3)^2}{2l_2 l_3}$$

Como se ve, existen dos posibles soluciones para q_3 según se tome el signo positivo o negativo de la raíz. Estas corresponden a las configuraciones de codo arriba y codo abajo del robot.

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

$$q_2 = \beta - \alpha$$

siendo:

$$\beta = \tan^{-1}\left(\frac{P_z}{r}\right) = \tan^{-1}\left(\frac{P_z}{\pm\sqrt{(P_x)^2 + (P_y)^2}}\right)$$

$$\alpha = \tan^{-1}\left(\frac{l_3 \sin q_3}{l_2 + l_3 \cos q_3}\right)$$

finalmente:

$$q_2 = \tan^{-1}\left(\frac{P_z}{\pm\sqrt{(P_x)^2 + (P_y)^2}}\right) - \tan^{-1}\left(\frac{l_3 \sin q_3}{l_2 + l_3 \cos q_3}\right)$$

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.

5.2 IMPLEMENTACIÓN EN LABVIEW

LabVIEW es un programa con ambiente de desarrollo muy parecido al lenguaje C o BASIC, que se utiliza hoy en día. Sin embargo LabVIEW a diferencia de estas aplicaciones, basadas en lenguajes de programación tipo texto, usa un lenguaje de programación G o gráfico, para crear programas en diagrama de bloques.

LabVIEW, como C o BASIC, es un sistema de programación de propósito general, con extensas bibliotecas de funciones para cualquier tipo de tarea de programación. También incluye bibliotecas para la adquisición de datos, análisis, presentación y almacenaje de datos. Otra característica de este software es su capacidad de establecer puntos de revisión, animar la ejecución de cada elemento del diagrama, para observar cómo se procesa cada uno de los elementos, lo cual da la posibilidad de depurar y aligerar el programa desarrollado.

Los programas desarrollados en LabVIEW son llamados instrumentos virtuales o VI, por sus siglas en inglés, ya que su apariencia y forma de operar puede imitar a los instrumentos actuales. Sin embargo, los VI son similares al funcionamiento de un lenguaje de programación convencional.

Un VI consiste en una interfaz, un diagrama de flujo de datos, el cual funge como código fuente, y un icono de conexiones el cual permite llamar dicho VI mediante otros instrumentos virtuales de mayor categoría. Específicamente un VI es estructurado de la siguiente manera:

- La interfaz con la cual interactúa el usuario es llamado Panel Frontal, por su similitud con un panel de operación de un instrumento físico. El panel frontal puede contener perillas, barras de desplazamiento, botones, así como otros controles e indicadores. La introducción de datos se hace mediante el mouse y teclado de la PC y la presentación de los resultados se observa en el monitor.
- El VI recibe las instrucciones de un diagrama de bloques, el cual se construye en lenguaje G. El diagrama de bloques también es el código fuente para el VI.
- Los VI son modulares y jerárquicos. Se pueden usar como programa principal, o como un subprograma dentro de otros programas. Un VI usado dentro de otro VI es llamado subVI. El icono y los conectores de un VI trabajan como una lista de parámetros gráficos, de esta forma otros VI pueden pasar datos a través de un subVI.

Con dichas características, LabVIEW promueve el concepto de programación modular, de esta forma una aplicación se divide en una serie de tareas, las cuales a su vez se pueden dividir, para que una aplicación complicada se convierta en una serie de subtareas más simples. De esta forma se crea un subVI para cada una de las subtareas y se combinan dichos VI en otro diagrama de bloques para lograr una tarea más general. Finalmente el VI de mayor nivel contiene todos los subVI los cuales representan las funciones de la aplicación. De esta forma la depuración de la programación dentro de cada subVI se logra con mayor facilidad.

5.2.1 USO DE LOS VI

La jerarquía en la creación de aplicaciones es muy importante, para ello, se debe empezar por un VI principal o de nivel alto y definir cuáles serán las entradas y salidas de la aplicación, posteriormente se construye una serie de subVIs que lleven a cabo las operaciones necesarias sobre los datos. Si un diagrama de bloques tiene un número excesivo de iconos, estos deben agruparse dentro de un VI de menor nivel, para mantener la simplicidad del diagrama de bloques principal. Este enfoque modular permite, como ya se mencionó, una depuración más sencilla del programa, así como una comprensión y mantenimiento más simple.

CONTROLES, E INDICADORES.

Un control es un objeto que se coloca en el panel frontal para el ingreso de datos en un VI. Un indicador es un objeto que desde el panel frontal muestra una salida. Los controles e indicadores en el lenguaje G se pueden ver como las entradas y salidas de un lenguaje de programación convencional.

TERMINALES

Las terminales son regiones en un VI o funciones a través de las cuales pasan los datos, son el análogo de los parámetros en lenguajes de programación basados en texto. Es importante conectar correctamente las terminales de una función o VI, tomando en cuenta el tipo de datos que se está recibiendo y enviando.

ESTRUCTURAS

Las estructuras son elementos de control de un programa. Se tienen 5 tipos: ciclos While, ciclos For, estructura Case, estructura Sequence y Formula Node.

Un ciclo While es una estructura que repite una sección de código mientras una condición se cumpla. Mientras que un ciclo For ejecuta una porción de código un número definido de veces.

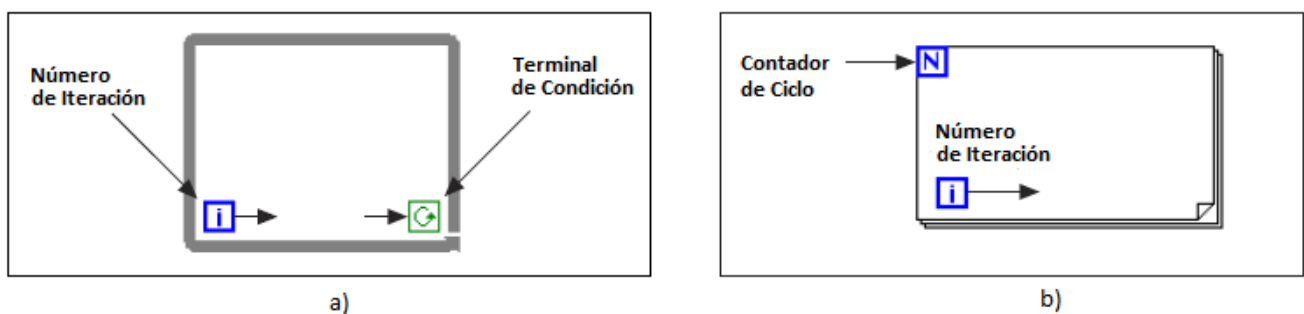


Figura 5.4. Estructuras cíclicas en LabVIEW. a) Estructura While b) Estructura For. (National Instruments, 1998)

Para ambos casos se tiene un tipo de registro llamado Shift Register o registro de corrimiento, el cual transfiere valores de un ciclo de la iteración hacia el siguiente, del lado derecho del recuadro de los ciclos aparece una terminal en la cual se conecta el dato que quiera ser transferido, y del lado izquierdo se conecta el dato que será recibido como se observa en la Figura 5.5, puede ser usado para obtener el dato de varias iteraciones atrás y con cualquier tipo de dato.

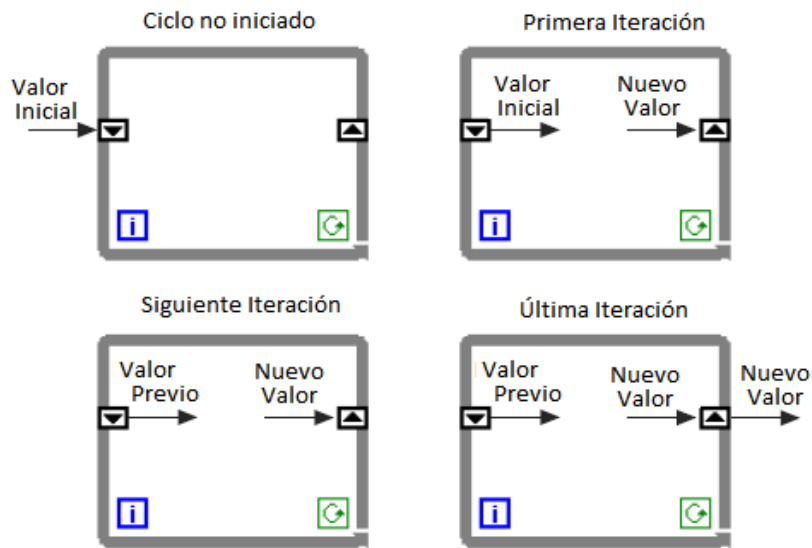


Figura 5.5. Seguimiento de los valores de Shift Register dentro de un ciclo While. (National Instruments, 1998)

Una estructura del tipo Case se muestra en la Figura 5.6, la cual tiene dos o más subdiagramas, o casos, de los cuales solamente uno es llevado a cabo cuando se ejecuta la estructura. Esto depende del valor de la variable conectada a la terminal de selección o selector. Cabe destacar que en lenguaje G, se debe incluir un caso predeterminado el cual se ejecuta en caso de que el selector no precise de algún otro caso.

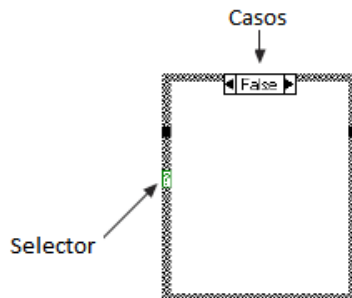


Figura 5.6. Estructura Case en LabVIEW. (National Instruments, 1998)

La estructura Sequence se puede ver en la Figura 5.7, luce como los marcos de un rollo de película, el cual ejecuta los diagramas de bloques de manera secuencial.

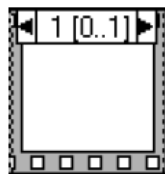


Figura 5.7. Estructura Sequence en LabVIEW. (National Instruments, 1998)

Por último un Formula Node como el que se muestra en la Figura 5.8, es una estructura en la cual se pueden ingresar fórmulas de tipo texto, directamente dentro del diagrama de bloques. Existe otra estructura muy parecida al Formula Node, llamada Math Script la cual se diferencia por usar lenguaje de Matlab para ingresar fórmulas.

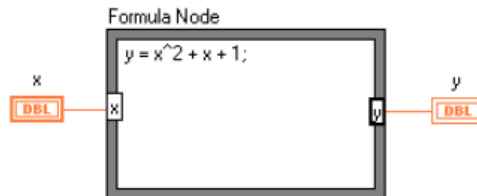
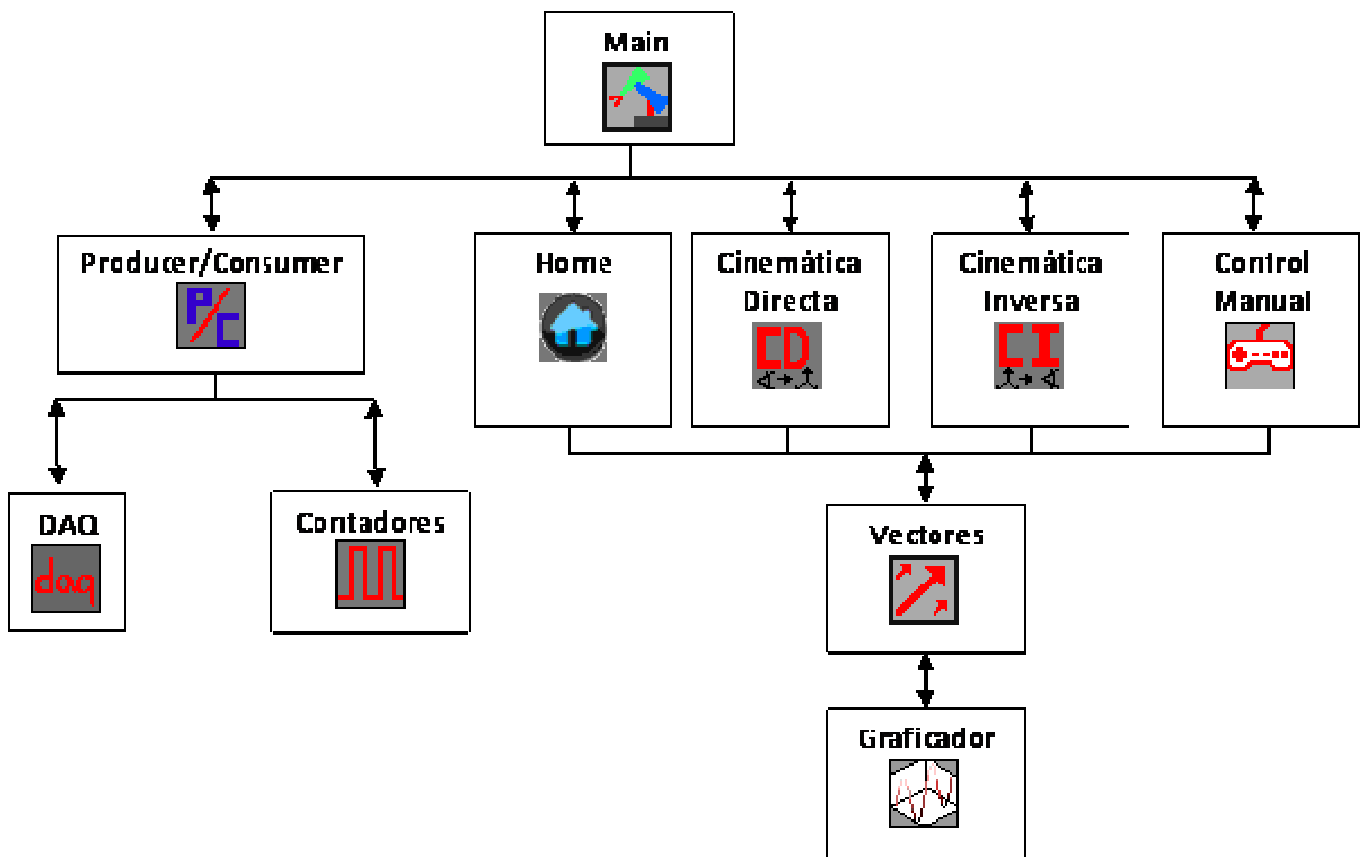


Figura 5.8. Formula Node en LabVIEW. (National Instruments, 1998)

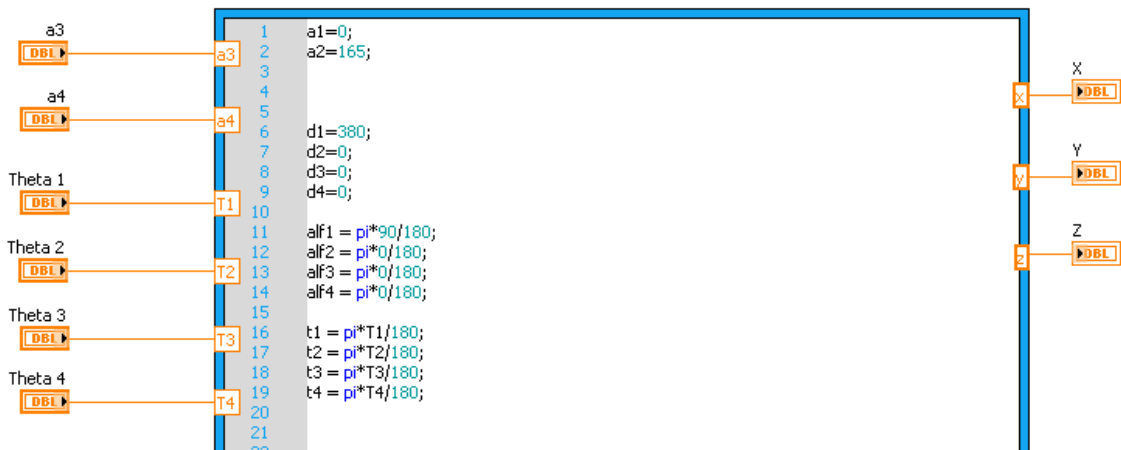
Para el uso del LabVIEW en la operación del robot, y con base en los fundamentos teóricos y las necesidades de lectura de encoder se siguió el siguiente esquema de programación e interconexión de subVI:



5.2.2 SUB VI CINEMÁTICA DIRECTA (CD)

Será necesario crear un VI completamente destinado a la resolución del problema Cinemático Directo, para el cual se utiliza una estructura Math Script que reciba los parámetros de la **Tabla 5.2** y devuelva las coordenadas de cada articulación, así como las del efector final.

Se comienza declarando las constantes y las variables del sistema



Las constantes no sólo serán las longitudes de las articulaciones (d [mm]), también están declarados los ángulos α en radianes.

Se puede notar que los ángulos $a3$ y $a4$ están asignados a 2 entradas, por lo que se podría pensar que son variables, lo cual no es así, ya que las entradas $a3$ y $a4$ sólo reciben o el valor de 0, o el de la longitud de las 2 últimas articulaciones (la razón de esta forma de declaración se explica en Tema 5.2.3 VI Vectores).

Como se puede notar entonces, $Theta 1$, $Theta 2$, $Theta 3$, y $Theta 4$ son las únicas variables y son los ángulos de cada una de las articulaciones.

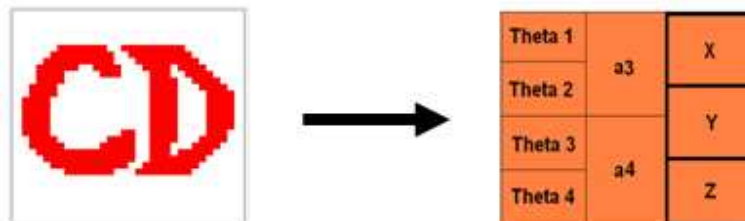
Con los valores de los parámetros de la **Tabla 5.2** ya declarados y valuando en la matriz de transformación genérica del capítulo 5.1.1, se obtienen las matrices de transformación A_i para cada articulación. Obsérvese que solamente se están usando 4 matrices, debido a que el quinto grado de libertad es el *roll* de la muñeca, el cual solo afecta a la orientación del efector, pero no a la posición. Ya con las 4 matrices, se puede obtener la Matriz de Transformación Homogénea de la cual se extraen los valores de la posición del último eslabona del robot.

```

20
21 A1=[cos(t1) -sin(t1)*cos(alf1) sin(t1)*sin(alf1) a1*cos(t1);
22 sin(t1) cos(t1)*cos(alf1) -cos(t1)*sin(alf1) a1*sin(t1);
23 0 sin(alf1) cos(alf1) d1;
24 0 0 0 1];
25
26 A2=[cos(t2) -sin(t2)*cos(alf2) sin(t2)*sin(alf2) a2*cos(t2);
27 sin(t2) cos(t2)*cos(alf2) -cos(t2)*sin(alf2) a2*sin(t2);
28 0 sin(alf2) cos(alf2) d2;
29 0 0 0 1];
30
31 A3=[cos(t3) -sin(t3)*cos(alf3) sin(t3)*sin(alf3) a3*cos(t3);
32 sin(t3) cos(t3)*cos(alf3) -cos(t3)*sin(alf3) a3*sin(t3);
33 0 sin(alf3) cos(alf3) d3;
34 0 0 0 1];
35
36 A4=[cos(t4) -sin(t4)*cos(alf4) sin(t4)*sin(alf4) a4*cos(t4);
37 sin(t4) cos(t4)*cos(alf4) -cos(t4)*sin(alf4) a4*sin(t4);
38 0 sin(alf4) cos(alf4) d4;
39 0 0 0 1];
40
41 T= A1*A2*A3*A4
42
43 x= T(1,4)
44 y= T(2,4)
45 z= T(3,4)

```

Por último, solamente se deben asignar los conectores a los controles e indicadores correspondientes para poder usarlo como SubVI desde el Main.

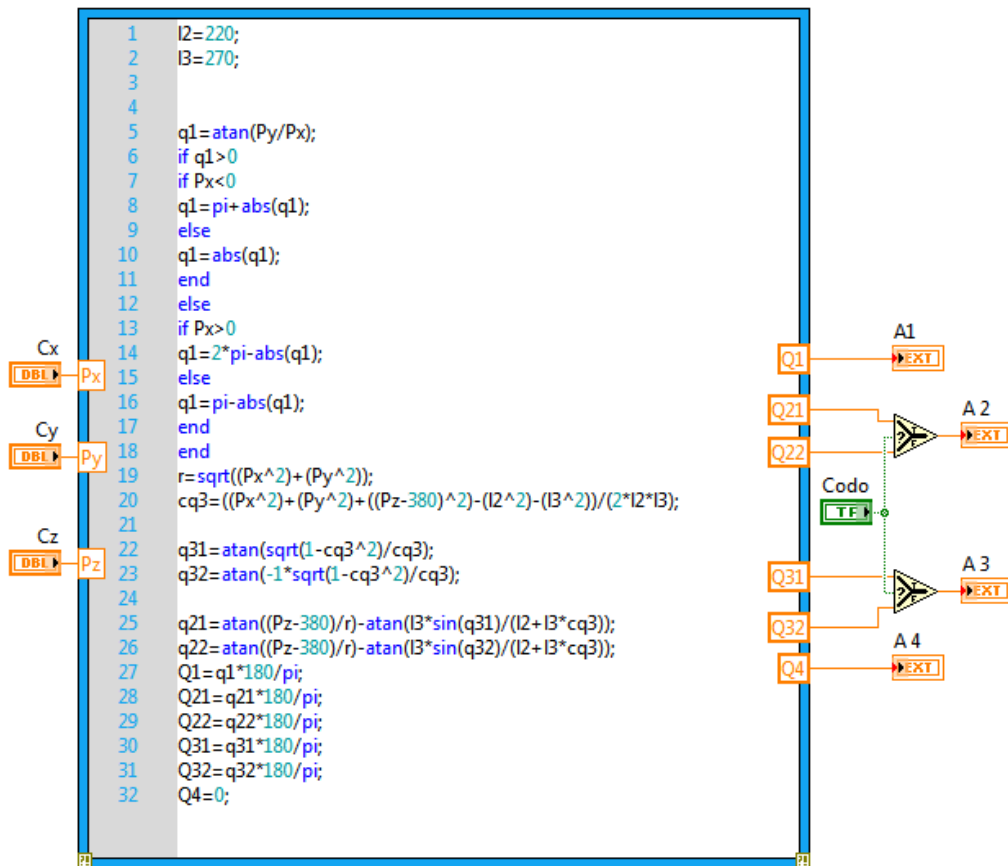


5.2.3 SUBVI CINEMÁTICA INVERSA

Será necesario otro SubVI específicamente para la Cinemática Inversa, en el cual se ejecute el algoritmo del método geométrico usando Math Script.

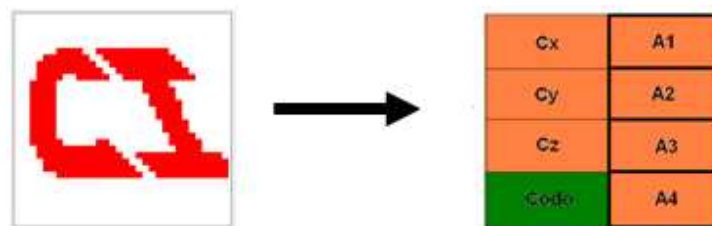
Es importante aclarar que los 2 grados de libertad de la muñeca no son contemplados para el cálculo, ya que igual que en la Cinemática Directa el *roll* de la muñeca no cambia la posición del efector.

El caso del *pitch* de la muñeca es diferente, debido a que el movimiento de dicha articulación si afecta la posición y no sólo a su orientación. Para evitar este problema se decidió utilizar únicamente 3 articulaciones (cadera, hombro, codo), haciendo 0 el ángulo entre la articulación 3 y 4 y sumando el largo del efector final al del eslabón entre las mismas.



Como se explicó anteriormente, las 2 posibilidades para Q2 y Q3 hace necesario poder elegir entre las configuraciones del codo, para ello es posible utilizar un valor booleano que defina los ángulos ya sea para codo arriba codo o abajo.

Finalmente se asignan los conectores para poder usarlo desde el Main.

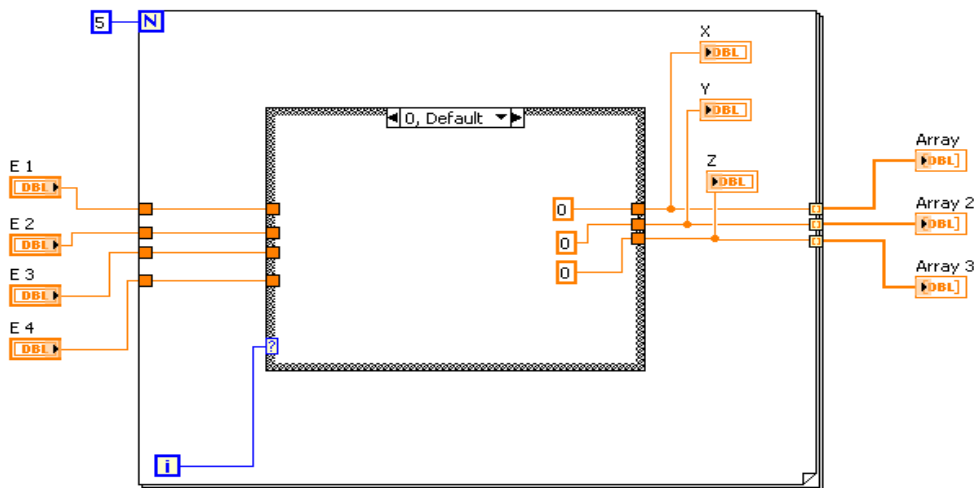


5.2.4 SUBVI VECTORES

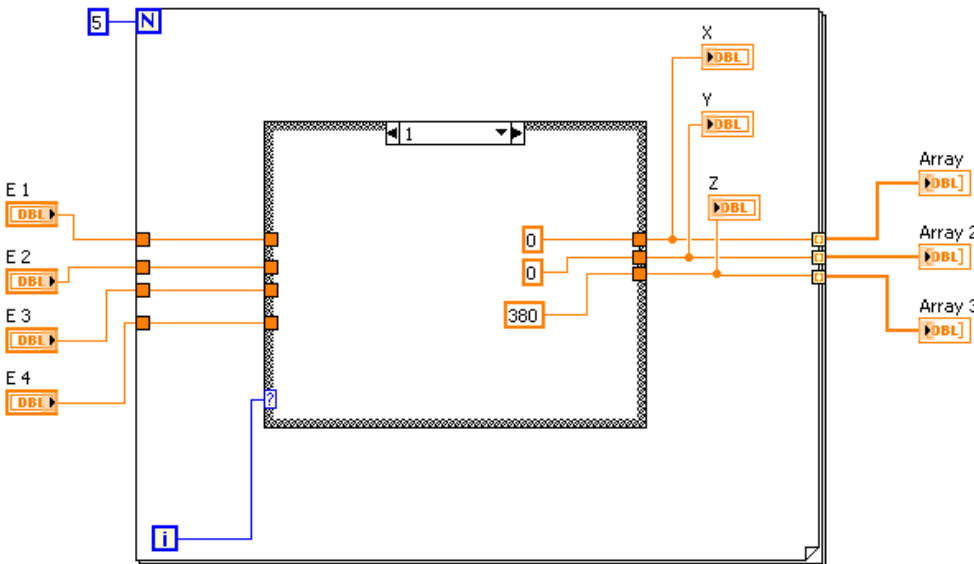
Ya que se tienen los SubVI básicos de control, podrán ser utilizados para crear una imagen 3D del brazo modelado. Para ello es necesario crear un nuevo SubVI que realice un ciclo que construya los 4 vectores a partir de la Cinemática Directa. Estos vectores corresponderán a cada eslabón del brazo.

El ciclo principal es un *for loop* de 5 iteraciones, el cual con cada iteración obtiene la posición de cada una de las articulaciones. De esta forma, llenando 3 arreglos de 1x5 con las posiciones desde el origen hasta el efector se puede simular el brazo en una gráfica 3D.

El primer punto siempre será el origen, por lo que en la primera iteración del *for* se envía de la *estructura case* a la primera casilla de cada arreglo un 0.



El segundo punto es el hombro y como se puede notar, este punto también es constante, ya que el giro de la cadera no afecta su posición. Por lo tanto la segunda iteración envía a la segunda casilla de los arreglos la posición (0, 0,380).

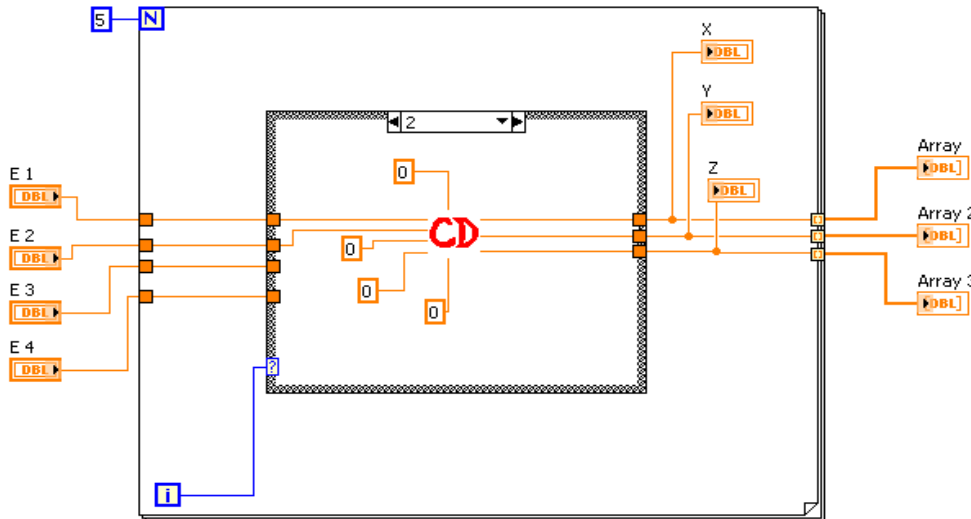


Para obtener la posición del codo es necesario llamar al SubVI de la Cinemática Directa. Si se analiza la CD se puede observar que Theta 1 y Theta 2 son las variables necesarias para obtener la posición del codo y, por lo tanto, serán conectadas directamente a E1 y E2; Theta 3 y Theta 4 no influyen en la posición del codo, por lo tanto se les asignará un 0.

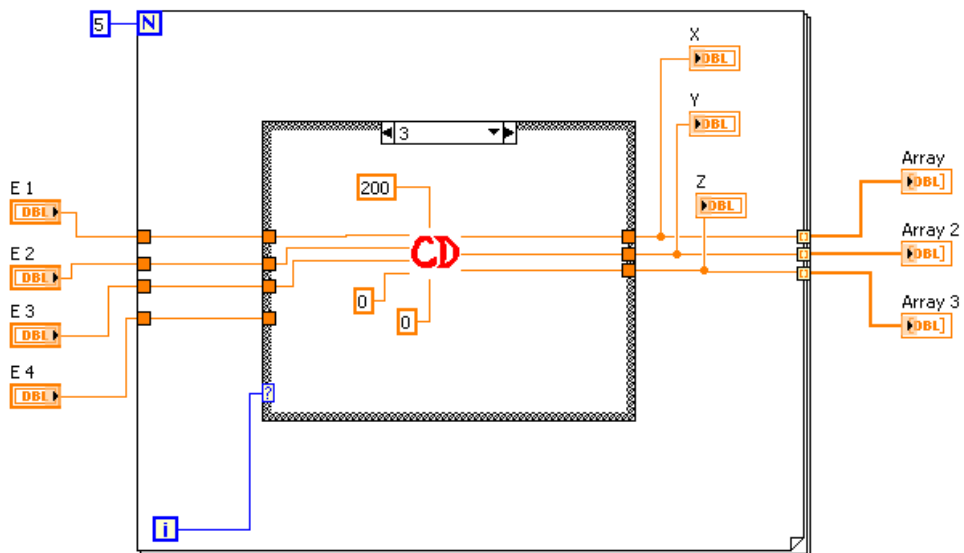
Nótese que en la CD se dejaron Alf3 y Alf4 conectadas a entradas, que se tendrán que asignar para este caso y ahora se puede entender el porqué de esta situación. La matriz de Transformación Homogénea (T) que se obtiene por la

multiplicación de las 4 matrices de rotación entrega la posición del efector final, pero al hacer Alf_3 y Alf_4 igual 0, puede considerarse que las matrices 2A_3 y 3A_4 serán matrices identidad y las coordenadas que la matriz T devolverá son las de la posición del codo.

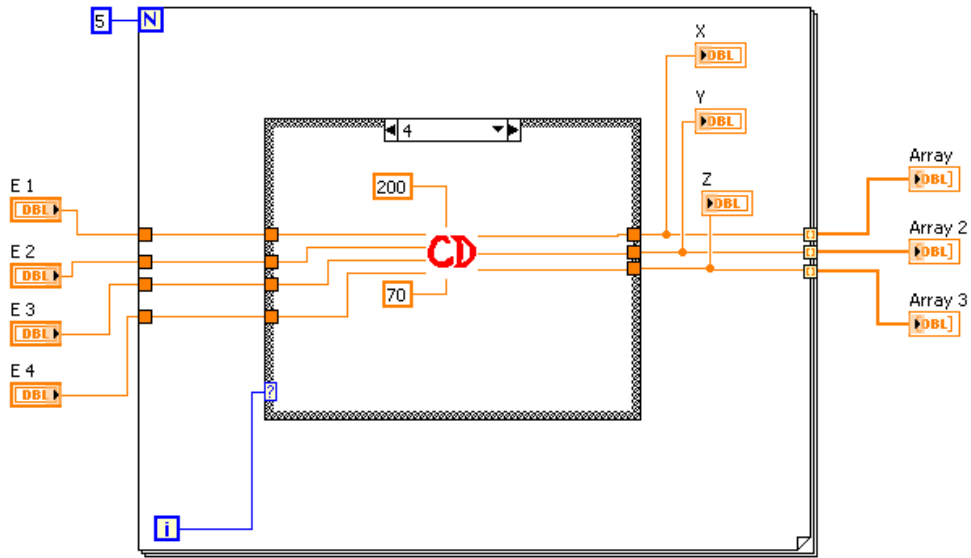
De esta manera se obtiene el valor de la tercera casilla de los arreglos



Con la cuarta iteración del ciclo *for* se busca obtener la posición de la muñeca, por lo cual de igual forma que en el caso anterior se hace uso de la CD, pero en este caso Theta 3 tomará el valor variable de E3, Alf_3 toma el valor de la longitud del eslabón correspondiente y Alf_4 y Theta 4 permanecen siendo 0. El resultado será almacenado en la cuarta casilla de los arreglos.

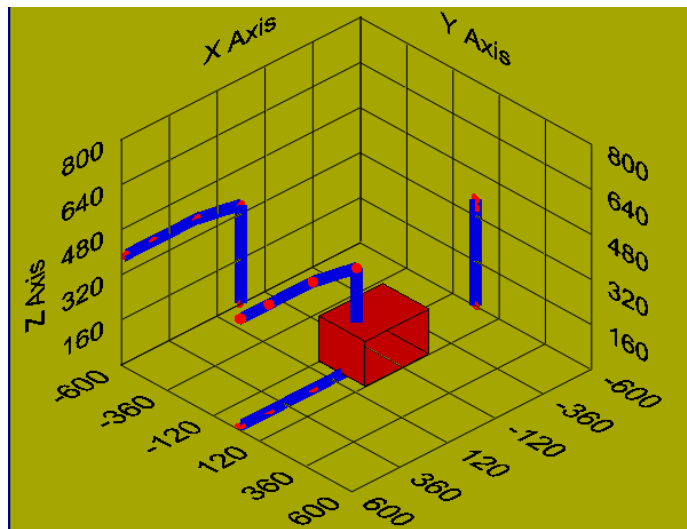


Con la última iteración se obtiene la posición del efector final, por lo cual los 4 valores de los controles E1, E2, E3, E4 serán conectados a su correspondiente ángulo Theta y Alf_3 y Alf_4 recibirán el valor de la longitud de su eslabón respectivo.

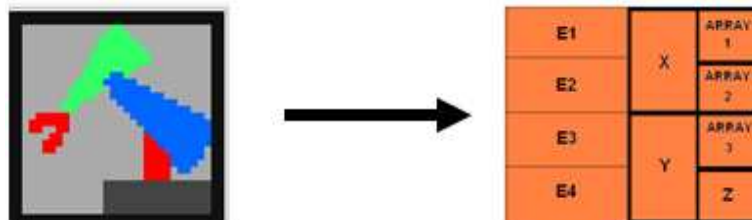


Ya que se tienen los 3 arreglos llenos y se puede ver que cada uno corresponde a un eje cartesiano (X, Y, Z), se observa que para cada conjunto de ángulos dados a las 4 articulaciones se obtienen los 3 arreglos que posicionan al robot.

Usando esta metodología se obtiene una imagen de vectores en 3D como se muestra en la siguiente figura.



Los conectores para este SubVI quedan como se muestra en la figura:

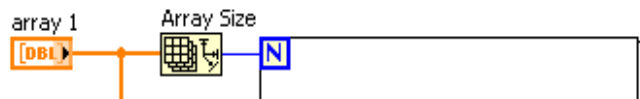


5.2.5 SUBVI TRAYECTORIA

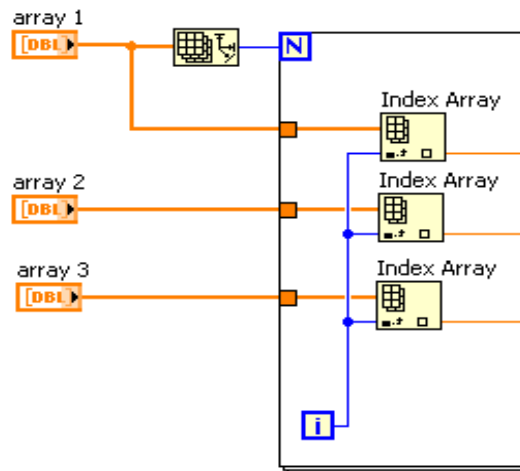
El objetivo principal de la simulación es permitir al usuario elegir los ángulos de las articulaciones o un punto dentro del espacio de trabajo del robot y obtener una imagen 3D del robot modelado basado en vectores. A partir de esto se puede crear un ciclo que permita enviar una serie de puntos que describan una trayectoria.

Se puede ver la necesidad de utilizar 3 arreglos de datos como entradas, cada uno con su coordenada cartesiana correspondiente (X, Y, Z). Estos arreglos deben ser de igual tamaño entre sí, pero el tamaño cambia con la trayectoria y con la exactitud con la que el robot se desplace sobre ella, ya que entre más puntos se listen, más fino será el movimiento.

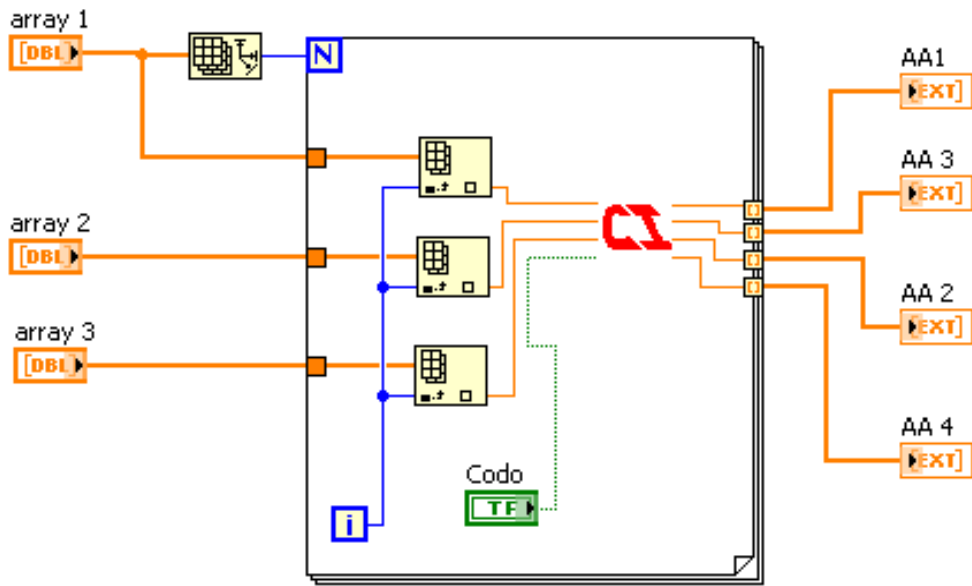
Para poder sacar todos los valores de los arreglos, sin importar de qué tamaño sean éstos se utiliza un ciclo *for* con el mismo número de iteraciones que de elementos tengan los arreglos. Para ello se utiliza el bloque Array Size, el cual recibe el arreglo y regresa el número de elementos que contiene.



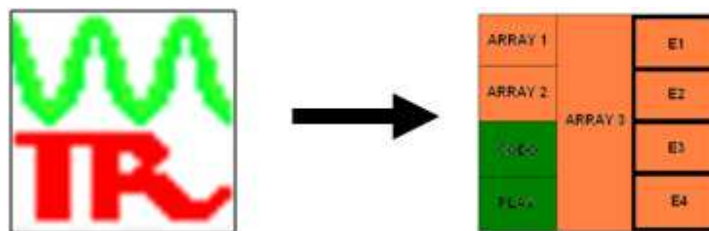
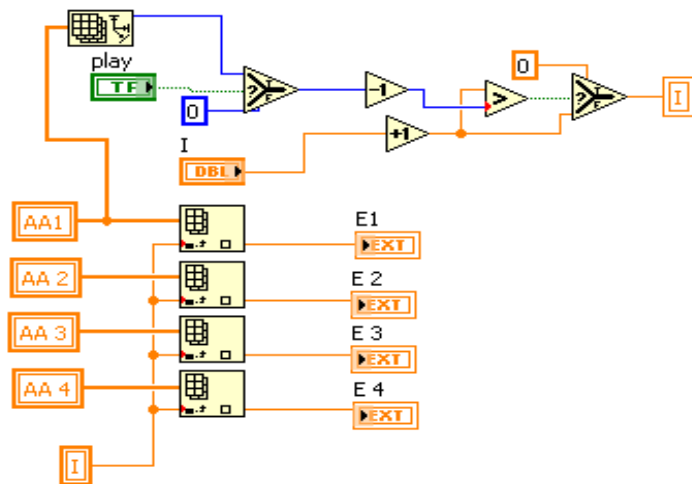
Ahora es necesario que con la primera iteración se obtenga la terna de elementos correspondiente a la primera coordenada, con la segunda iteración la segunda coordenada y así sucesivamente hasta el final de los arreglos. El bloque Index Array recibe el arreglo y regresa el elemento que se encuentra en el valor del índice, el cual está conectado a la cuenta de la iteración actual.



Como el SubVI vectores construye el robot a partir de la CD, es necesario que la terna de valores que definen cada punto, se convierta en un conjunto de los 4 ángulos que precise ese mismo punto. Es por ello que en cada iteración la terna de valores es enviada al SubVI de Cinemática Inversa, el cual devuelve conjuntos de 4 ángulos que al ser indexados en el borde del ciclo *for* se guardan en arreglos.

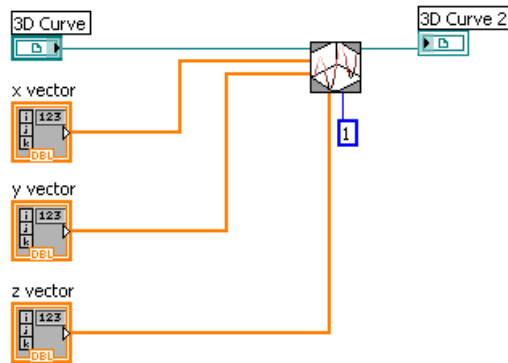


Hasta ahora solamente se ha pasado de arreglos de coordenadas cartesianas a arreglos de ángulos para las articulaciones, y es necesario enviar cada conjunto de ángulos al SubVI Graficador. Se empleará un ciclo similar al *for* pero que incremente una variable auxiliar desde 0 hasta el número de elementos del arreglo de ángulos. No se utiliza una estructura de LabVIEW ya que se necesita que este ciclo esté siempre activo mientras se esté ejecutando este VI, se podría utilizar un ciclo *while*, pero como no existe condición de término no regresaría al VI Main.



5.2.6 SUBVI GRAFICADOR

Este VI prácticamente es para mantener organizado y legible el proyecto, ya que su función simplemente es recibir los arreglos de los vectores y utilizarlos para construir la simulación del brazo dentro de una gráfica 3D



Siguiendo con las conexiones



Para mantener ordenado el proyecto se crearon otros 3 sub VI que se encargan de toda la adquisición de datos. El primero de ellos es llamado DAQ y contiene las líneas de adquisición de datos digitales y analógicos, el segundo es llamado Flancos y contiene la etapa de análisis, recibe los datos y realiza el conteo de pulsos; el tercero es un patrón de diseño llamado Producer / Consumer, el cual optimiza el tiempo de ejecución.

5.2.7 SUBVI DAQ

Este VI crea las líneas de adquisición de datos, tanto digitales como analógicos. En la Figura 5.9 se observa cómo se configura un puerto de entradas digitales, ya que se necesitan 6 líneas, una para cada microswitch.

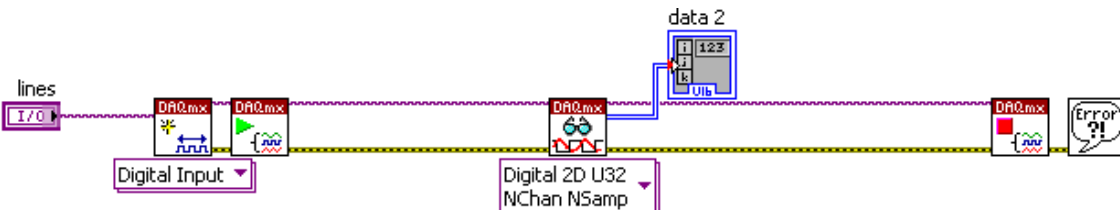


Figura 5.9. Diagrama de bloques para utilizar un puerto digital

Para la creación del puerto de entradas analógicas es necesario un bloque extra que configure el reloj y la tasa de muestreo de las líneas. En la siguiente imagen se observa la configuración utilizada:

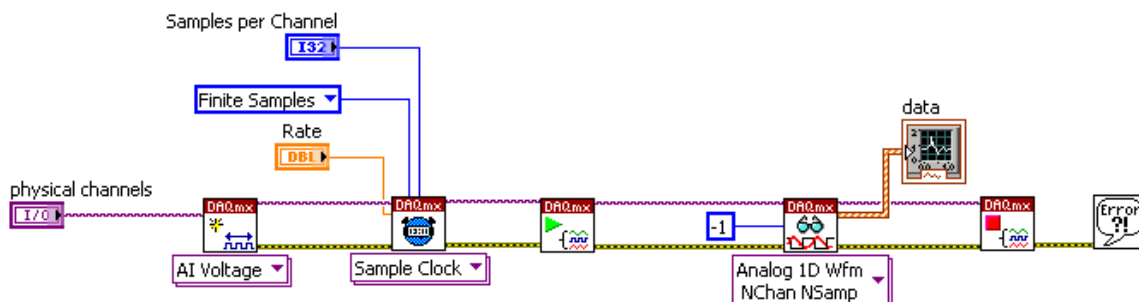
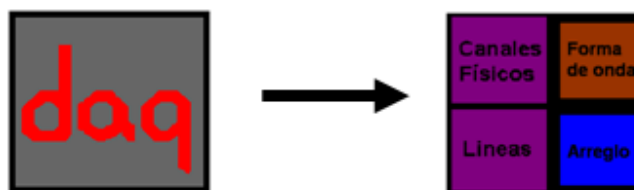


Figura 5.10. Diagrama de bloques para utilizar un puerto analógico

Las conexiones del Sub VI se muestran a continuación:



5.2.8 SUBVI FLANCOS

Este Sub VI recibe los datos de las lecturas tanto analógicas como digitales y las procesa para obtener los estados de los microswitches y el conteo de pulsos.

Primero se explicará cómo se analizaron las entradas digitales provenientes de los microswitches.

Al crear un puerto digital de 6 líneas, se puede obtener la lectura de dos diferentes formas: como arreglo decimal 2-D o como forma de onda digital. La segunda dificulta demasiado su análisis ya que entrega los datos de un modo que no pueden ser fácilmente utilizados por los bloques básicos de manejo de ondas digitales.

De forma contraria, el arreglo decimal 2-D, devuelve un número decimal que representa la conversión del número binario posible con los 6 bits cada uno relacionado a las 6 líneas. Esto quiere decir, que si no hay microswitches presionados, el número binario es 111111, y lo que se tendría en el arreglo es un 63. Si por ejemplo dos microswitches se accionan, la línea digital entregaría por decir 010111, lo cual en el arreglo sería 23.

Ya con la lectura en el arreglo, se puede saber mediante un pequeño análisis qué bits exactamente están activos, usando bloques para regresar el número decimal a binario y revisarlo bit por bit.

En la Figura 5.11 se puede observar cómo se revisa la casilla (0,0) del arreglo decimal que se recibe (data 2), posteriormente se hace la conversión de Decimal a Arreglo Binario y nuevamente pasa por un bloque Index Array que revisa las primeras 5 casillas del Arreglo Binario, para después comparar cada elemento con una constante booleana False, si el comparativo da como resultado True quiere decir que la lectura fue 0, por lo que el microswitch está activado, y por consecuencia la variable de la cuenta regresa a 0.

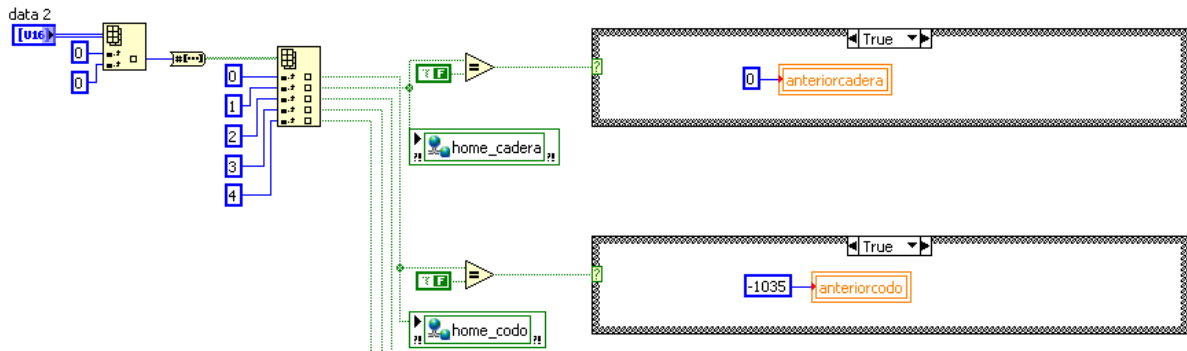


Figura 5.11. Análisis para los microswitches de la cadera y el codo

Esto se realiza para cada uno de los 6 microswitches para poder posicionar el brazo en la posición de Home. Cabe mencionar que cuando algunas de las articulaciones se encuentran en su posición de Home, la cuenta no regresa a 0, ya se estableció que si la cuenta es igual a 0, el ángulo también lo es, y para esas articulaciones la posición de Home no es a 0 grados.

Ahora se analizarán las señales de las entradas analógicas provenientes de los encoders. En la Figura 5.12 se puede observar el diagrama final para el conteo de pulsos.

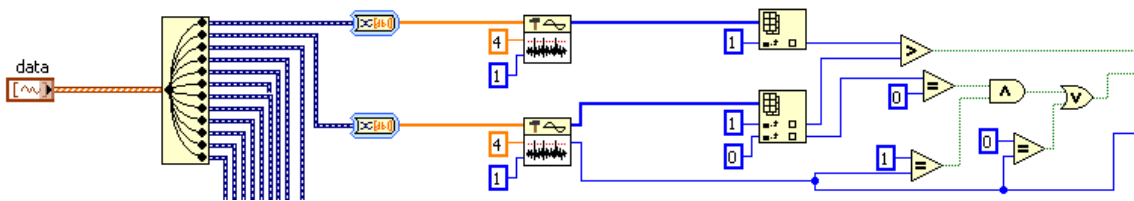


Figura 5.12. Análisis de las señales analógicas

La señal de tipo forma de onda (data) que se recibe de las líneas analógicas es necesario separarla, ya que se creó un puerto analógico con las 12 señales juntas, además de ser necesario convertir los datos de forma de onda a Arreglo 1-D numérico.

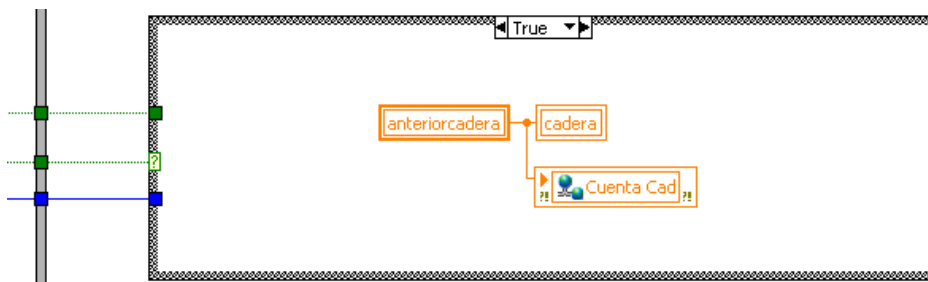
Ya que tenemos por separado las 2 señales de cada encoder, se debe realizar el conteo de pulsos bajo ciertas circunstancias, además de obtener la dirección de giro del motor a partir de las señales.

El bloque que realiza la detección, recibe el Arreglo numérico 1-D y revisa cada una de sus casillas buscando cambios de nivel con las condiciones establecidas. En este caso el voltaje mínimo para que el bloque cuente un pico válido es de 4 volts, y para tener un 0 lógico válido se necesita un voltaje máximo de 1 volt.

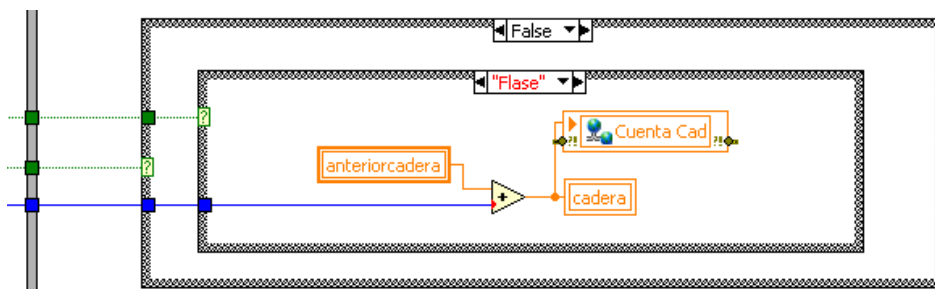
Sólo se llevará la cuenta de una de las dos señales, ya que ambas señales tendrán el mismo número de pulsos, pero es necesario saber en cuál de ambas señales se detectó antes el primer pico válido. Es por ello que se analiza el arreglo que se obtiene del bloque detector de picos, ya que dicho arreglo contiene los índices, o la localización de todos los picos válidos de la señal analógica. Debido a que la adquisición de datos es mediante muestreos, y no podemos saber si alguna señal antes del muestreo estaba en alto o en bajo, al hacer la comparación de la primera casilla de los Arreglos de Índices se puede obtener el sentido de giro equivocado. Siendo así, se compara la segunda casilla de cada arreglo y se obtiene la dirección.

El conteo de pulsos necesita tener algunas condiciones para que sea un pulso válido, nuevamente debido a la forma en que se tienen los datos del muestreo es posible que aunque el brazo no esté en movimiento la señal del encoder permanezca en alto y por lo tanto el bloque detector de picos estaría aumentando la cuenta de pulsos erróneamente, por lo cual debemos usar algunas compuertas lógicas para discriminar esos casos y obtener el conteo más exacto posible.

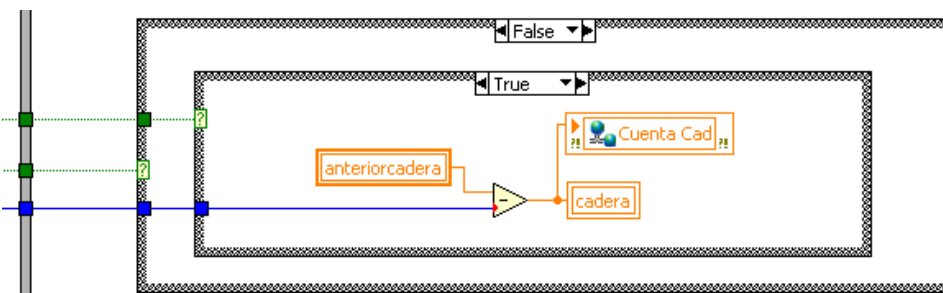
De la Figura 4 se observa que tenemos 3 cables con los resultados del análisis, los cuales se utilizarán para controlar dos estructuras case anidadas. De esta forma se tienen tres casos posibles: El primero será cuando las compuertas lógicas indiquen que no hay pulsos válidos durante el muestreo, dejando la cuenta actual igual a la cuenta anterior de la articulación, en la figura siguiente se observa dicho caso.



El segundo caso se presenta cuando sí existen pulsos válidos, y el sentido de giro es positivo. En este caso los pulsos serán sumados a la cuenta anterior.

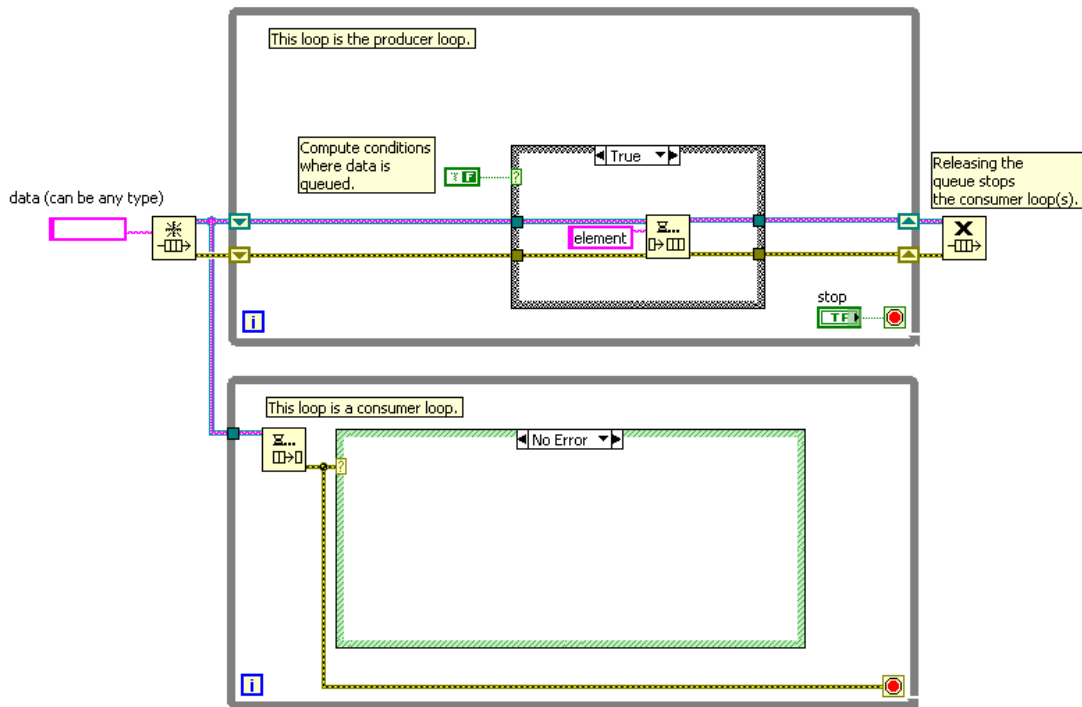


En la tercera opción el sentido de giro es contrario, para lo cual los pulsos válidos del muestreo son restados de la cuenta anterior.



El análisis de cada par de señales se realiza para cada articulación con diagramas de bloques iguales a los explicados anteriormente, con la diferencia de la variable global a la que se le asigna el conteo.

5.2.9 SUB VI PRODUCER / CONSUMER



El Patrón de Diseño Producer/Consumer está basado en el diseño Maestro/Esclavo, el cual está orientado a mejorar la forma en que se comparten los datos entre ciclos que se ejecutan a diferentes tasas de velocidad.

Este diseño es usado para desacoplar los procesos en dos categorías: los que producen los datos y los que consumen los datos. Las pilas o colas de datos serán usadas para compartir datos entre los ciclos funcionando como un buffer entre ellos.

También es comúnmente usado para procesos donde se adquieren varios grupos de datos que se deben procesar en el orden en que se adquirieron. Al usar pilas de datos dentro de un mismo ciclo, la fase de adquisición de datos llena las pilas para posteriormente ser analizadas, impidiéndonos obtener más datos en ese tiempo.

El modelo Productor / Consumidor permite al Productor adquirir datos para ser analizados en el Consumidor a su propia velocidad, permitiendo al Productor continuar apilando datos.

Para nuestra aplicación el ciclo Productor será el que adquiera las señales que provengan tanto de los encoder como de los microswitches. Debido a que la tarjeta NI USB-6255 posee canales digitales que tienen una velocidad máxima de 1 KHz. se conectan a éstos únicamente los microswitches, y los encoder se conectan a las entradas analógicas que permiten velocidades mucho mayores, como máximo 20 MHz.

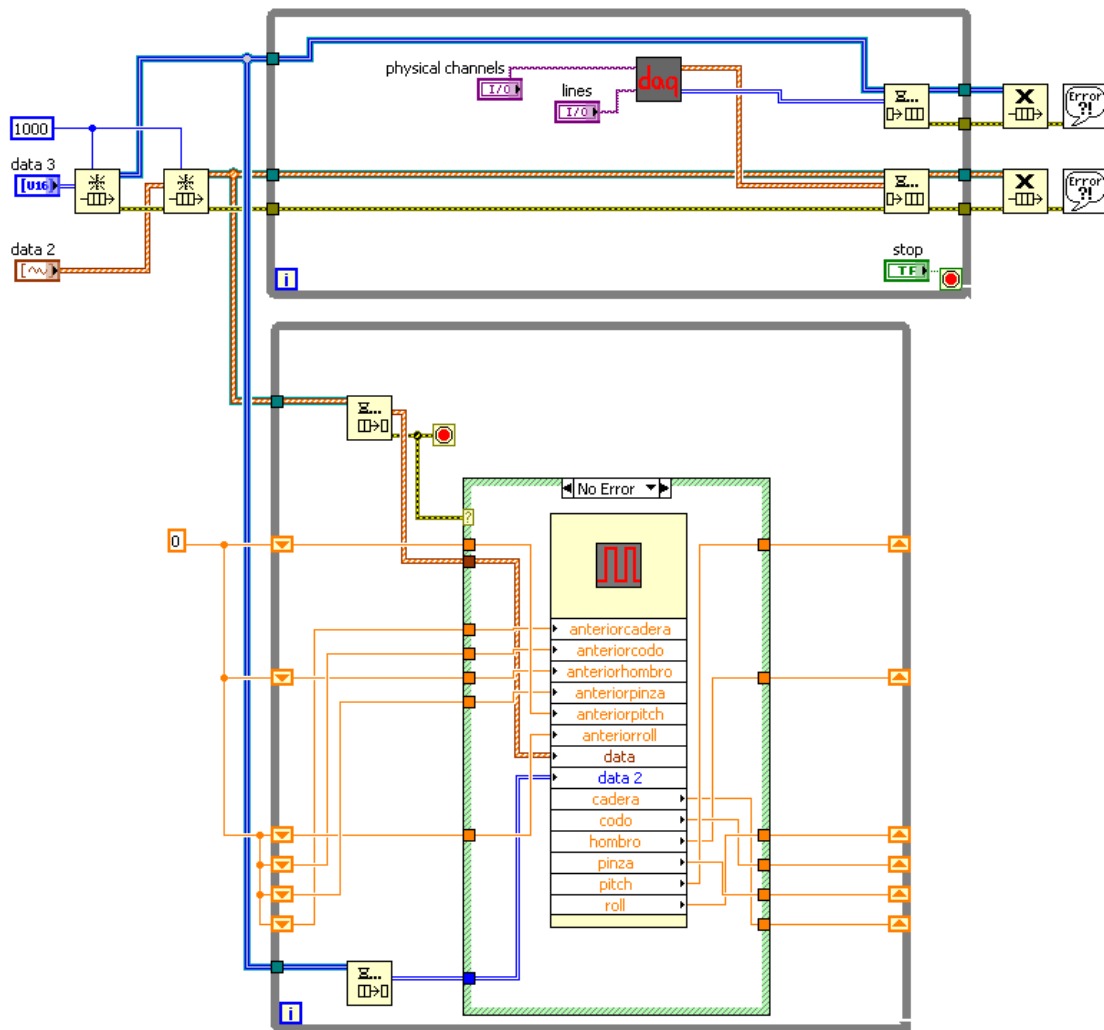


Figura 5.13. Estructura Productor-Consumidor para la lectura de encoder.

Como se puede observar en la Figura 5.13, se crean dos pilas con referencias de diferentes tipos de datos. La primera pila (data 3), está referida a un tipo de dato Arreglo 2-D, la cual almacena los datos de las líneas digitales. La segunda (data 2) almacenará datos de tipo forma de onda, provenientes de las líneas analógicas.

El ciclo while superior es el ciclo Productor, tiene constantes de canales físicos conectados a las entradas del SubVI DAQ, estas constantes contienen los nombres de las líneas como vienen especificadas en la **¡Error! No se encuentra el origen de la referencia..** A la salida de DAQ obtenemos los datos de la adquisición, cada salida está conectada a su pila correspondiente.

El ciclo Consumidor estará formado por el Sub VI Flancos, el cual recibe las pilas de datos de las adquisiciones. Las variables “anteriorarticulacion” y “articulacion” fueron creadas para mantener las cuentas actuales y las del ciclo anterior. Como se observa fue necesario usar shift registers, ya que después del conteo de pulsos de un ciclo la cuenta debe conservarse para ser sumada o restada a la cuenta del ciclo siguiente.

5.3 CONFIGURACIÓN Y PROGRAMACIÓN DEL PLC.

El software STEP 7 de Siemens provee una herramienta capaz de: crear programas en diferentes lenguajes de programación de PLC como son AWL, KOP y FUP; hacer los ajustes en la configuración necesarios para el uso del CPU y módulos específicos en que se esté trabajando; realizar ajustes en las comunicaciones por medio de la modificación del direccionamiento de los módulos; así mismo, su capacidad de simulación permite visualizar el funcionamiento de los programas sin la necesidad de cargarlo a un PLC real para así observar posibles fallas.

El S7-300 es un módulo de automatización, el cual se caracteriza por tener la capacidad de conexión de diferentes módulos de trabajo dependiendo de la tarea que vaya a desempeñar, así el S7-300 puede presentarse con un módulo central CPU, una fuente de alimentación con diferentes capacidades de amperaje, módulos de entradas y salidas tanto analógicas como digitales, módulos de funciones especializadas así como procesadores de comunicaciones para la creación de redes.

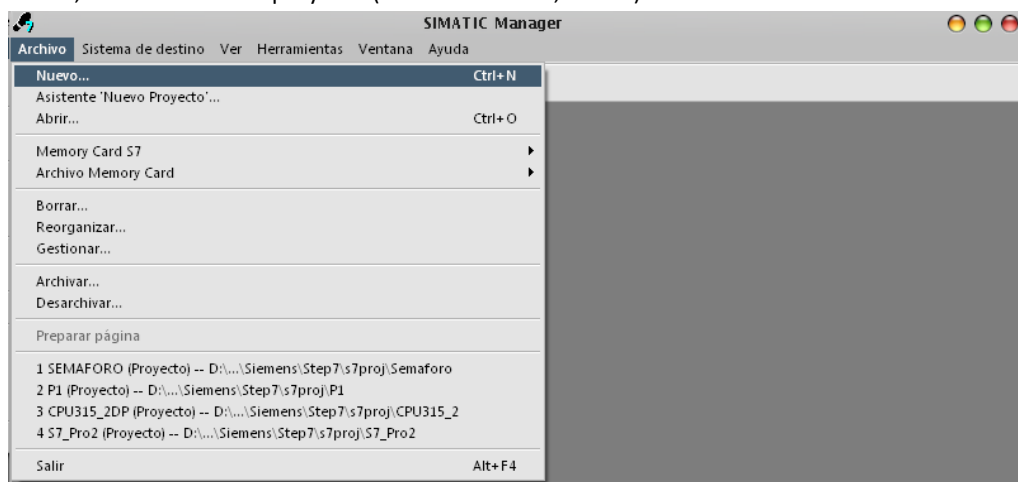
Para poder implementar un programa dentro del PLC es necesario configurar el hardware, definir el protocolo de comunicación entre la PC y el PLC y crear el bloque de programación mediante el cual el PLC encenderá o apagará las salidas de sus módulos. Todo ello está definido en los siguientes pasos.

5.3.1 USO DE SIMATIC MANAGER PARA LA CREACIÓN DE UN PROYECTO.

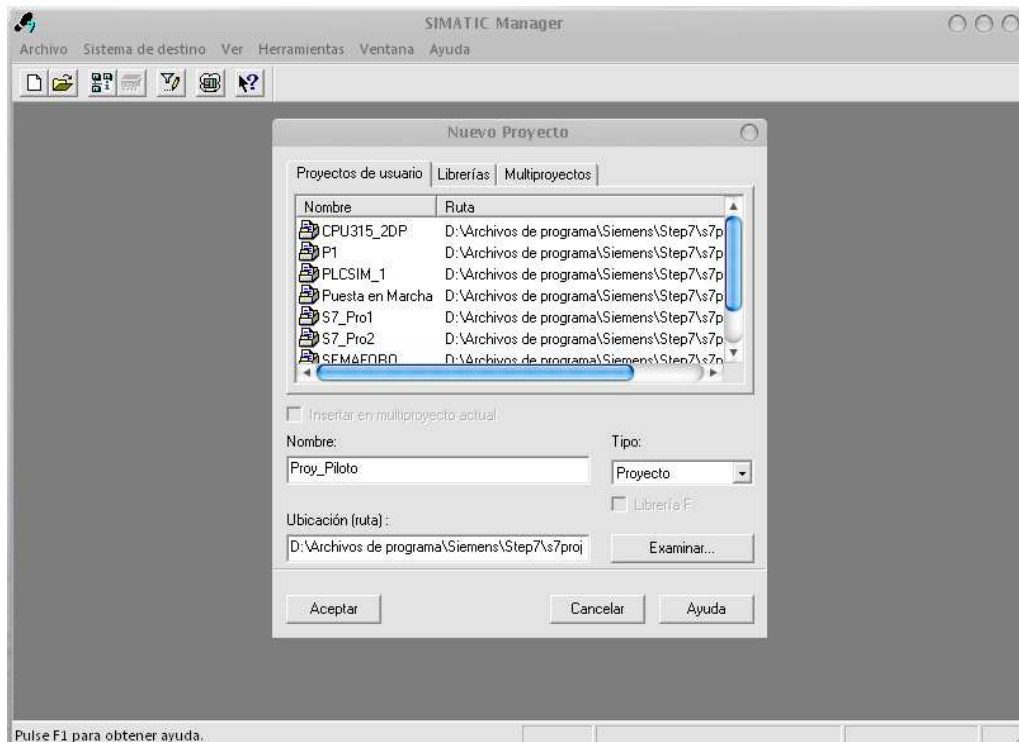
CREACIÓN DE UN PROYECTO



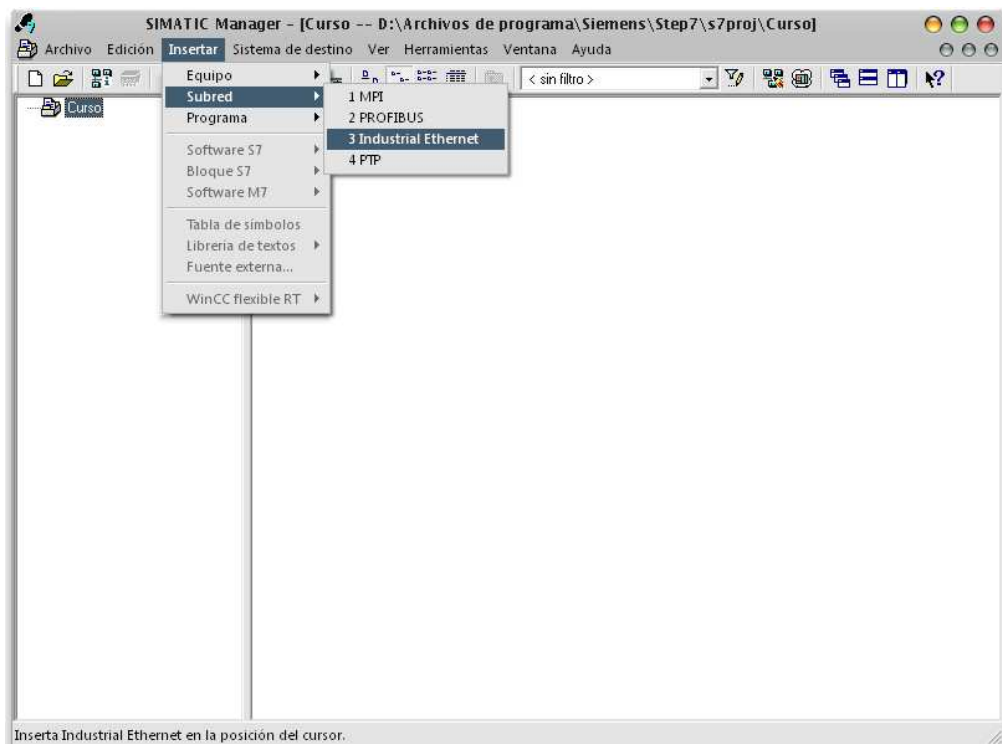
1. Abrir desde el escritorio el software SIMATIC Manager, desde aquí es posible llamar y configurar las herramientas de creación de programas, de configuración de módulos que contenga el PLC S7-300, comunicaciones, simulación, entre otras.
2. Una vez abierto, se crea un nuevo proyecto (Archivo→Nuevo, Ctrl+N)



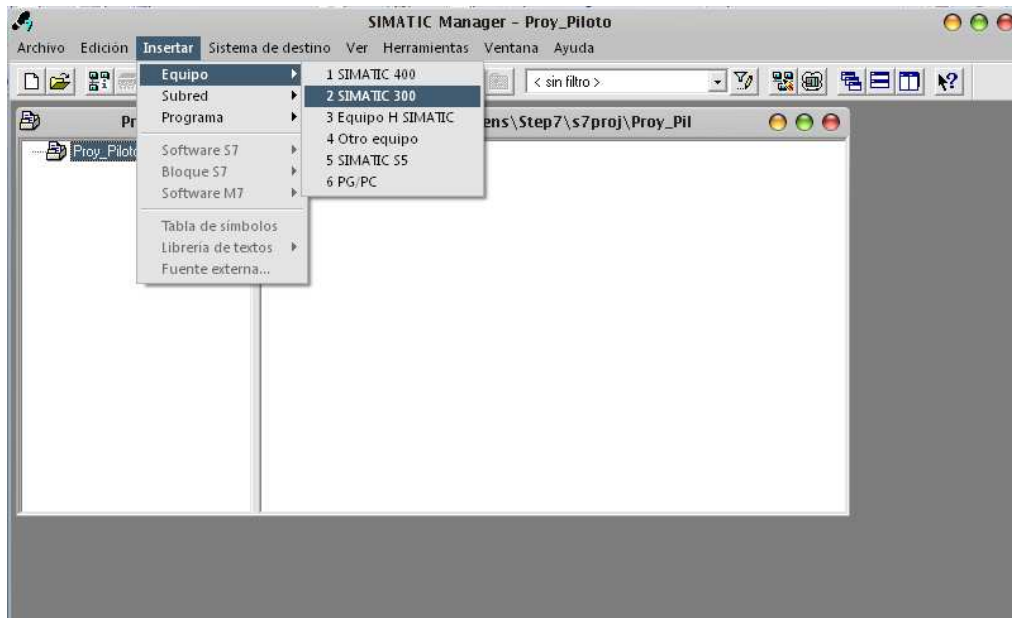
3. Se asigna un nombre de proyecto y se define una ruta o ubicación diferente para la creación del proyecto.



4. Crear una nueva red Ethernet (Insertar → Subred → Industrial Ethernet), para trabajar con este protocolo.

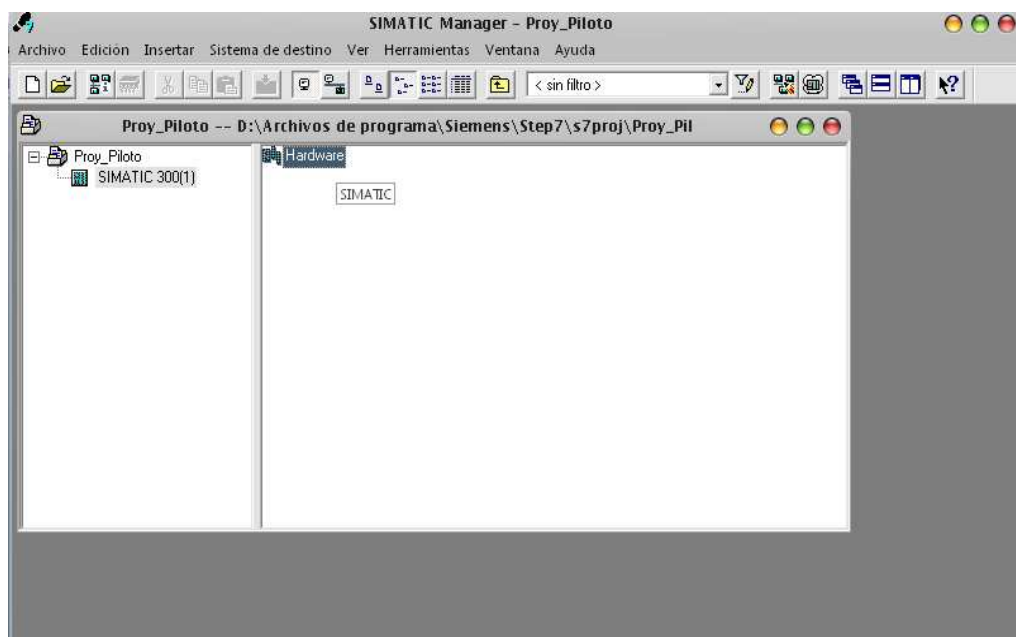



5. Insertar un equipo S7-300 (Insertar → Equipo → S7-300).

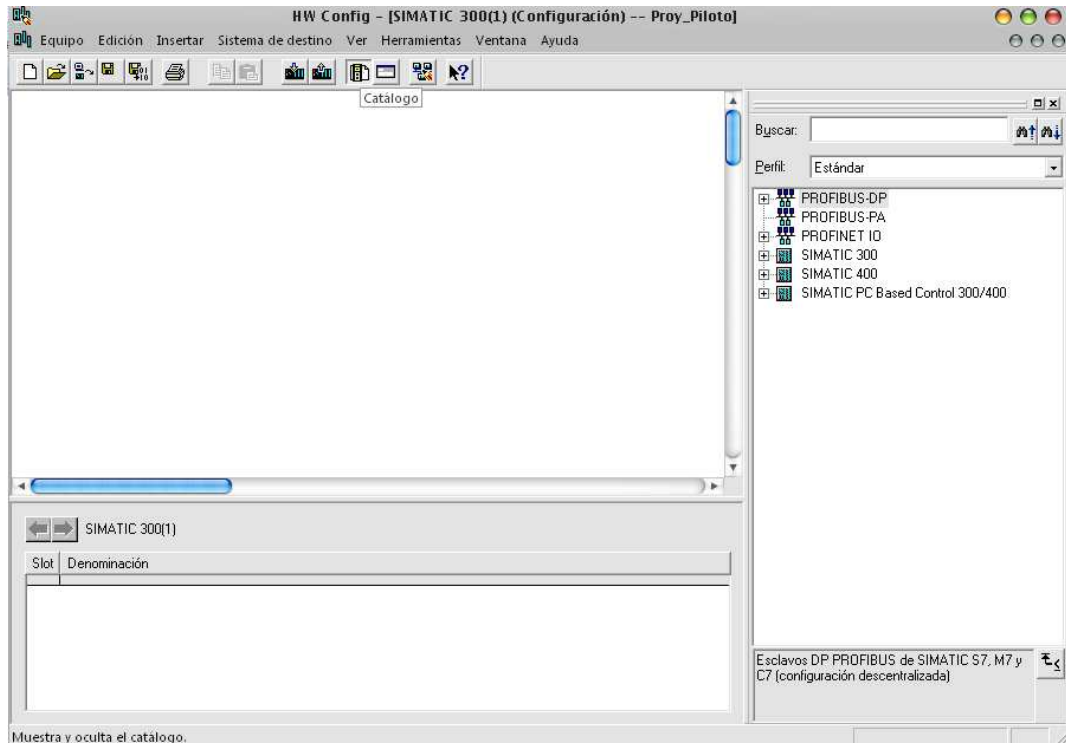


CONFIGURACION DEL HARDWARE

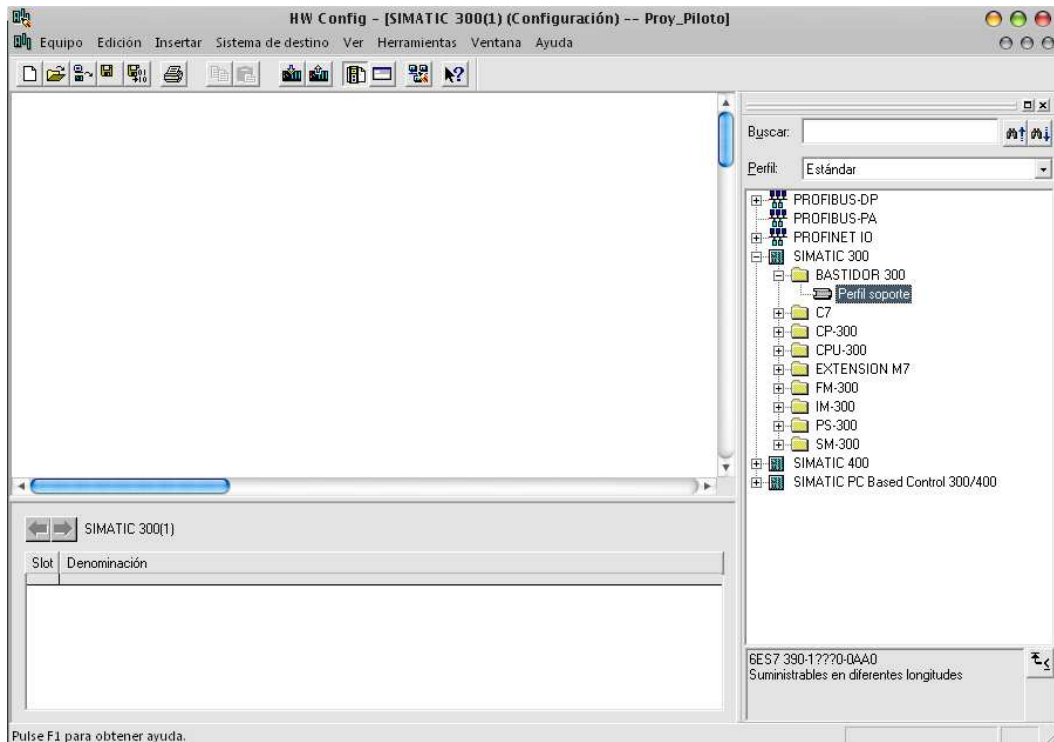
6. Una vez que es creado el PLC se muestra una ventana con la herramienta de configuración del hardware (doble clic en Hardware).



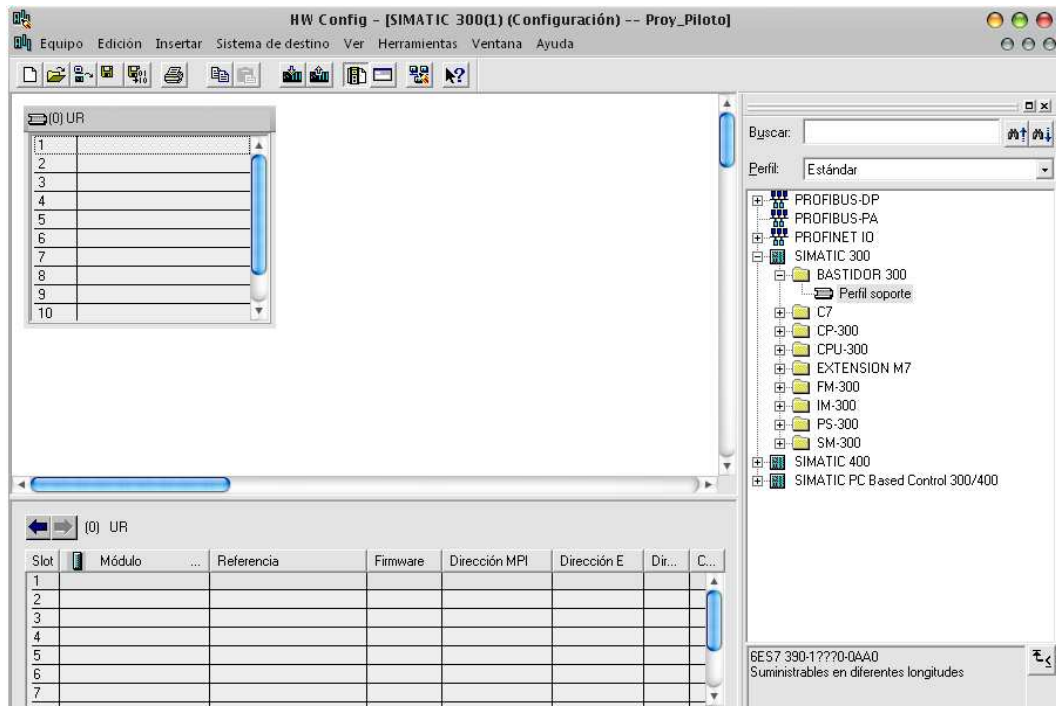
7. En la ventana de configuración del hardware, el icono del catálogo  debe estar activo, con lo cual se despliega la gama de módulos existentes para el S7-300 y S7-400, así como los PROFIBUS.



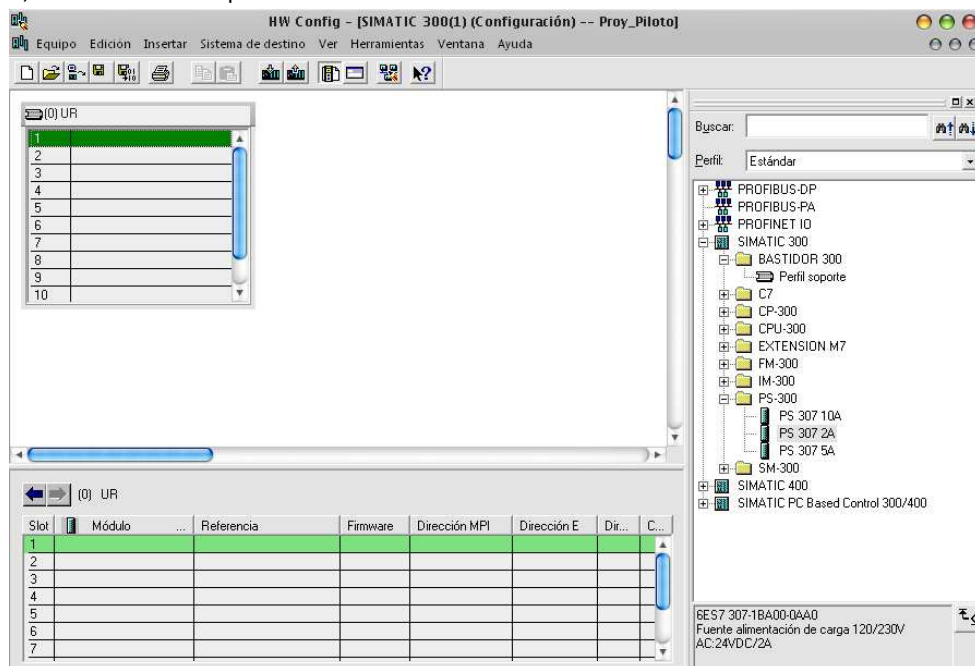
8. Hacer clic en Perfil de Soporte dentro del catálogo (SIMATIC 300→Bastidor 300→Perfil Soporte).



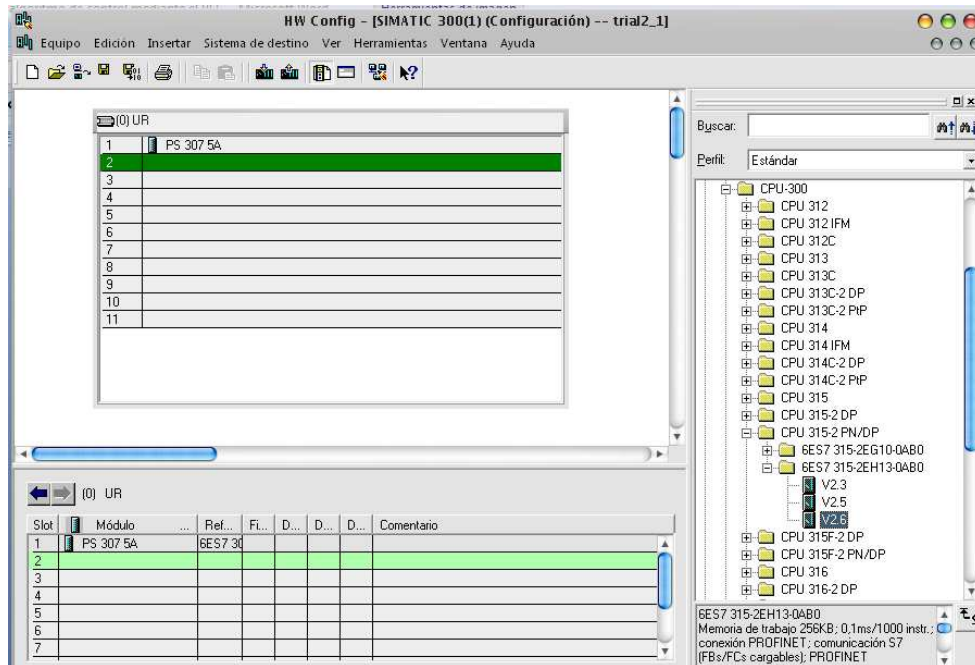
Se muestra una tabla en la cual se agregan los módulos.



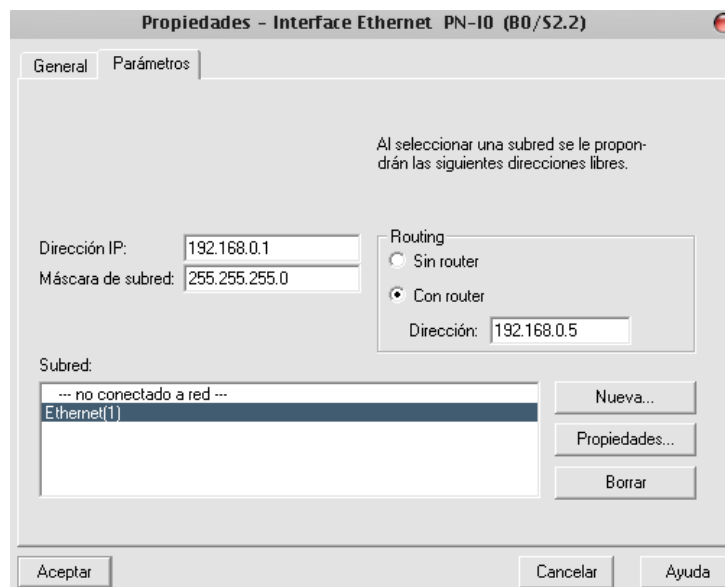
- Ahora es posible seleccionar y arrastrar hasta el bastidor los módulos, es importante señalar que la primera línea siempre está reservada para la fuente, la segunda para el CPU y la tercera para módulos de comunicación, de este modo a partir de la línea 4 se pueden agregar los módulos de entradas y salidas, de propósito específico, etc. Por lo tanto, primero se agrega la fuente (SIMATIC 300 → PS 300), seleccionando en este caso la PS-307 5A y se arrastra a la primera línea del bastidor. Cabe señalar que debajo del catálogo, una vez que seleccionado un módulo o componente, se muestra una explicación con las características de cada módulo.



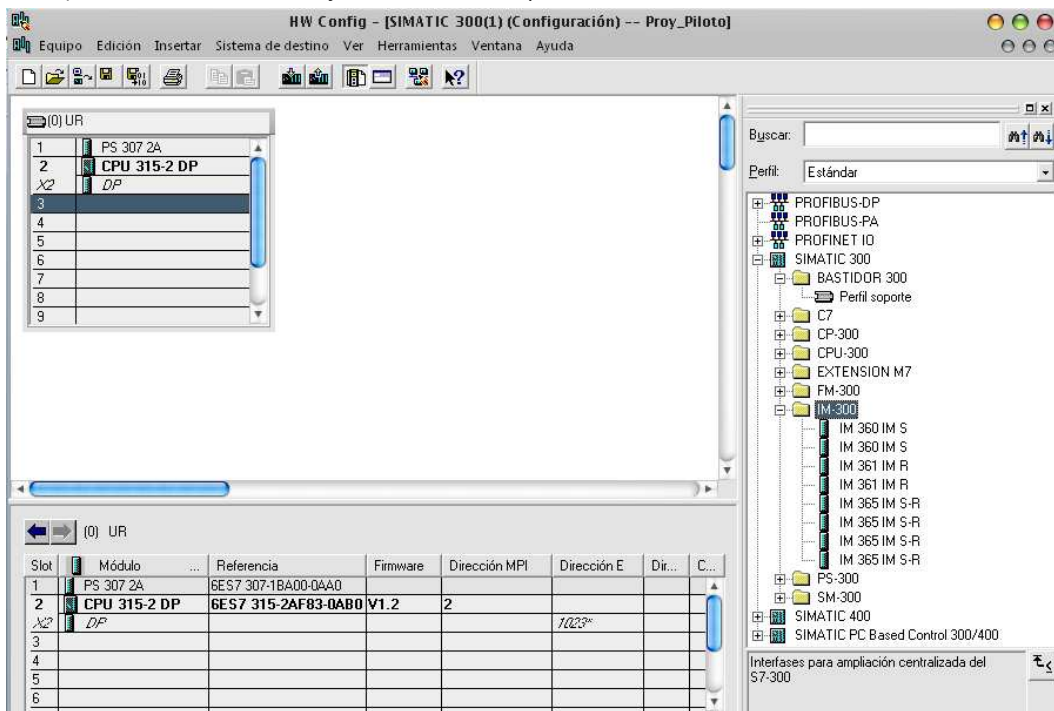
10. Posteriormente se selecciona el CPU en la segunda posición del bastidor, que en este caso será un 315F- PN/DP (SIMATIC 300→CPU 300→CPU 315- 2 DP→6ES7 315-2EH13-0AB0→V2.6)



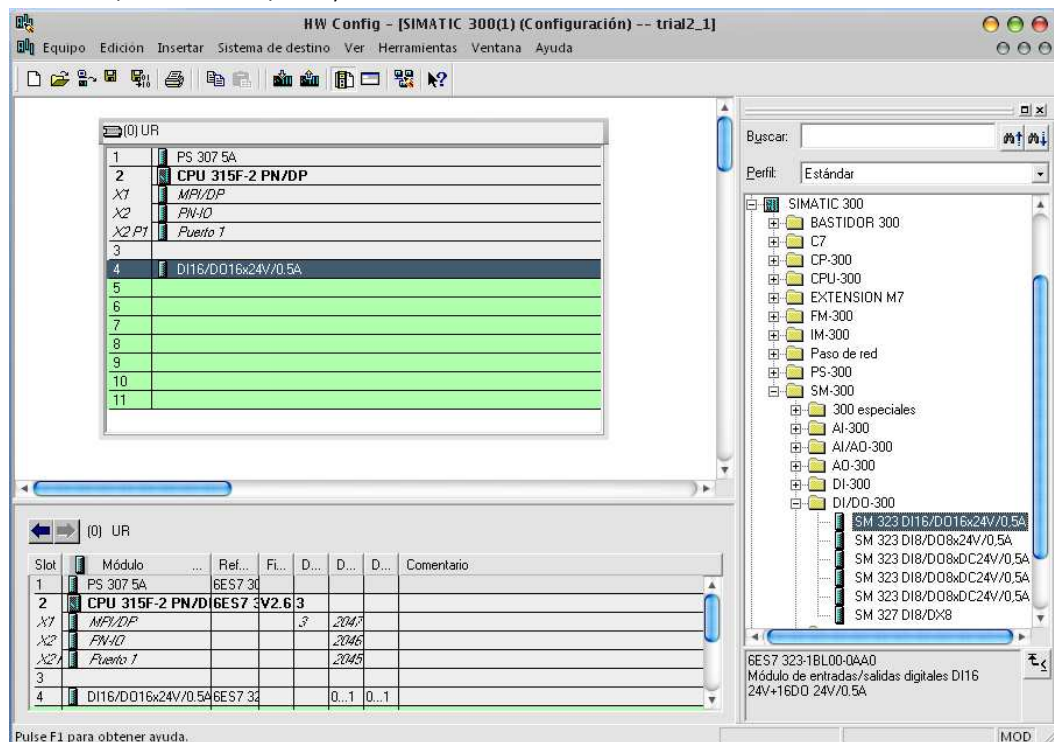
En la nueva ventana se ingresa la información para la configuración del puerto Ethernet, seleccionando la dirección que tiene el PLC dentro de la red, proporcionando además la dirección IP del ruteador que controla las comunicaciones entre los dispositivos, por último seleccionar la red Ethernet.



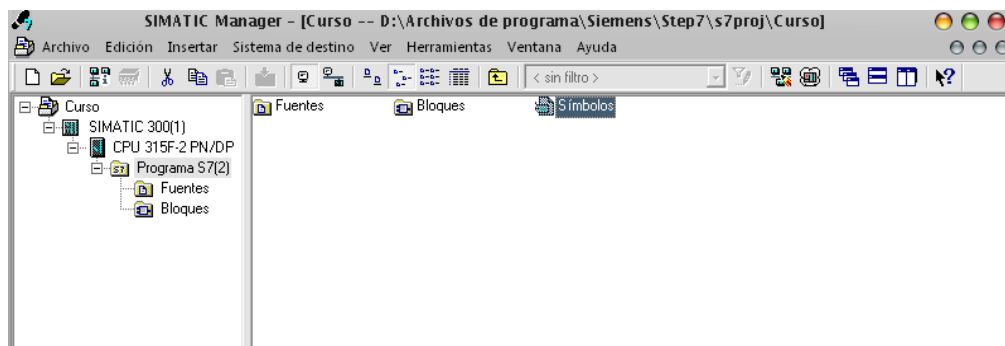
11. En caso de contar con un módulo de comunicación será agregado en la tercera línea del bastidor (SIMATIC 300→IM-300), en caso contrario se deja en blanco ese espacio.



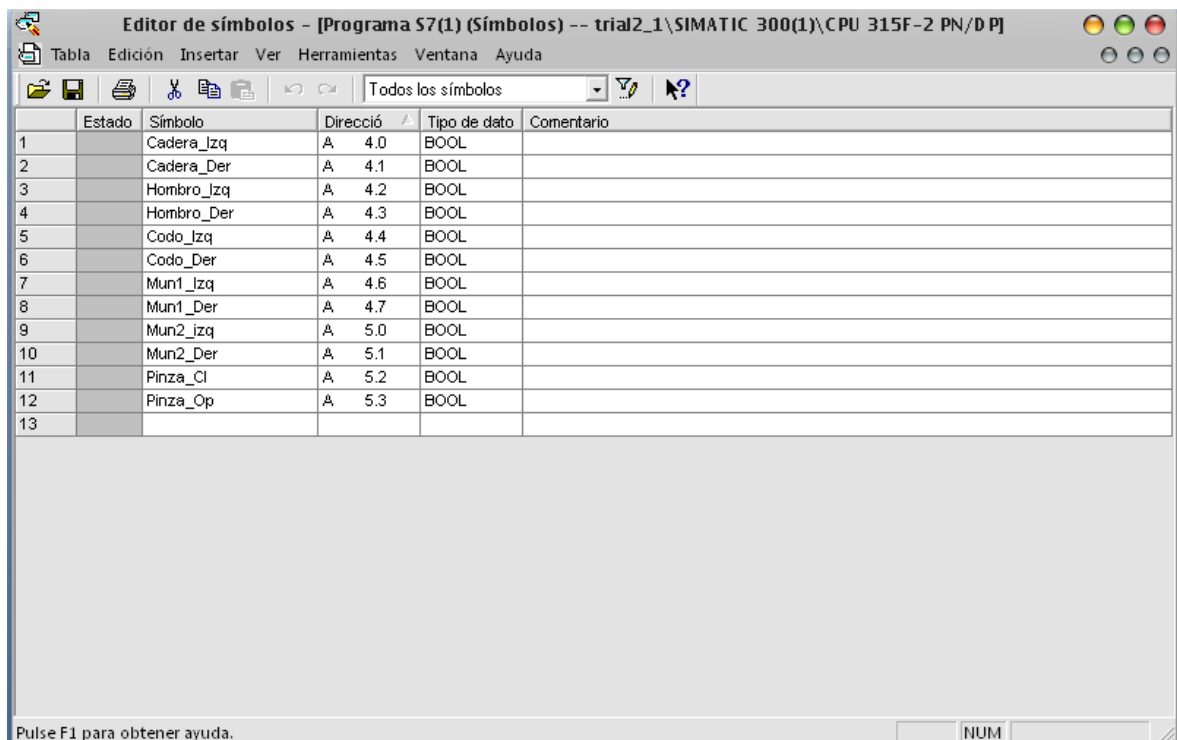
12. Posteriormente se agrega un módulo de entradas y salidas digitales en la cuarta línea del bastidor, en este caso un módulo de 16 entradas digitales y 16 salidas digitales a 24 VDC/0.5 A. (SIMATIC 300→SM300→DI/DO-300→SM323 DI16/DO16xDC24V/0.5 A)



13. Los módulos de entradas y salidas analógicas en caso de ser usados se agregarán de igual manera. (Para entradas analógicas SIMATIC 300→SM300→AI-300. Para salidas analógicas SIMATIC 300→SM300→AO-300). Las propiedades de cada módulo pueden ser modificadas haciendo clic derecho sobre el módulo dentro del bastidor.
14. Una vez definido el hardware de del PLC se guarda la configuración y se cierra la ventana de HW Config.
15. Para el control de las salidas, entradas, contadores, temporizadores, y demás componentes de un software de PLC, se pueden crear los símbolos de las variables para una mejor identificación. En el SIMATIC Manager abrir la tabla de símbolos del PLC (Proyecto→Simatic 300→ CPU 315 F-2 PN/DP→Programas S7→Símbolos)



16. En la nueva ventana se muestra el Editor de Símbolos, donde es posible agregar las variables y asociarlas a una dirección del PLC. Una vez creadas las variables, guardar y cerrar el Editor.



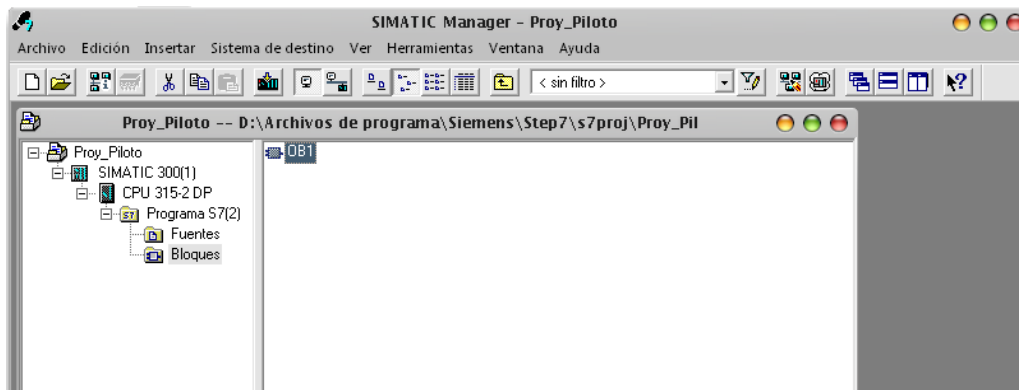
En la **Tabla 5.3** se muestran los tipos de datos que maneja el PLC.

Tabla 5.3. Tipo de Datos de un S7-300

Tipo y descripción	Tamaño en Bits	Formato-Opciones	Rango y notación numérica (Valores máximo y mínimo)	Ejemplo
BOOL (Bit)	1	Texto Booleano	TRUE/FALSE	TRUE
BYTE (Byte)	8	Número Hexadecimal	B#16#0 a B#16#FF	B#16#10
WORD (Palabra)	16	Número Binario	2#0 a 2#1111_1111_1111_1111	2#0001_0000_0000_0000
		Número Hexadecimal	W#16#0 a W#16#FFFF	W#16#1000
		BCD	C#0 a C#999	C#998
		Número Decimal sin signo	B#(0,0) a B#(255,255)	B#(10,20)
DWORD (Doble Palabra)	32	Número Binario	2#0 a 2#1111_1111_1111_1111_1111_1111_1111_1111	2#1000_0001_0001_1000_1011_1011_0111_1111
		Número Hexadecimal	DW#16#0000_0000 a DW#16#FFFF_FFFF	DW#16#00A2_1234
		Número Decimal sin signo	B#(0,0,0,0) a B#(255,255,255,255)	B#(1,14,100,120)
INT (Entero)	16	Número Decimal con signo	-32768 a 32767	1
DINT (Int,32 bit)	32	Número Decimal con signo	L#-2147483648 a L#2147483647	L#1
REAL (Número en coma flotante)	32	Número en coma flotante IEEE	Máximo: +/-3.402823e+38 Mínimo: +/-1.175495e-38	1.234567e+13
S5TIME (Tiempo Simatic)	16	Tiempo S7 en pasos de 10 ms	S5T#0H_0M_0S_10MS a S5T#2H_46M_30S_0MS and S5T#0H_0M_0S_0MS	S5T#0H_1M_0S_0MS S5TIME#1H_1M_0S_0MS
TIME (Tiempo IEC)	32	Tiempo IEC en pasos desde 1ms, entero con signo	-T#24D_20H_31M_23S_648MS a T#24D_20H_31M_23S_647MS	T#0D_1H_1M_0S_0MS TIME#0D_1H_1M_0S_0MS
DATE (Fecha IEC)	16	Fecha IEC en pasos de 1 día	D#1990-1-1 a D#2168-12-31	DATE#1994-3-15
TIME_OF_DAY (Fecha y Hora)	32	Tiempo en pasos de 1ms	TOD#0:0:0.0 a TOD#23:59:59.999	TIME_OF_DAY#1:10:3.3
CHAR (Carácter)	8	Caracteres ASCII	'A', 'B' etc.	'B'

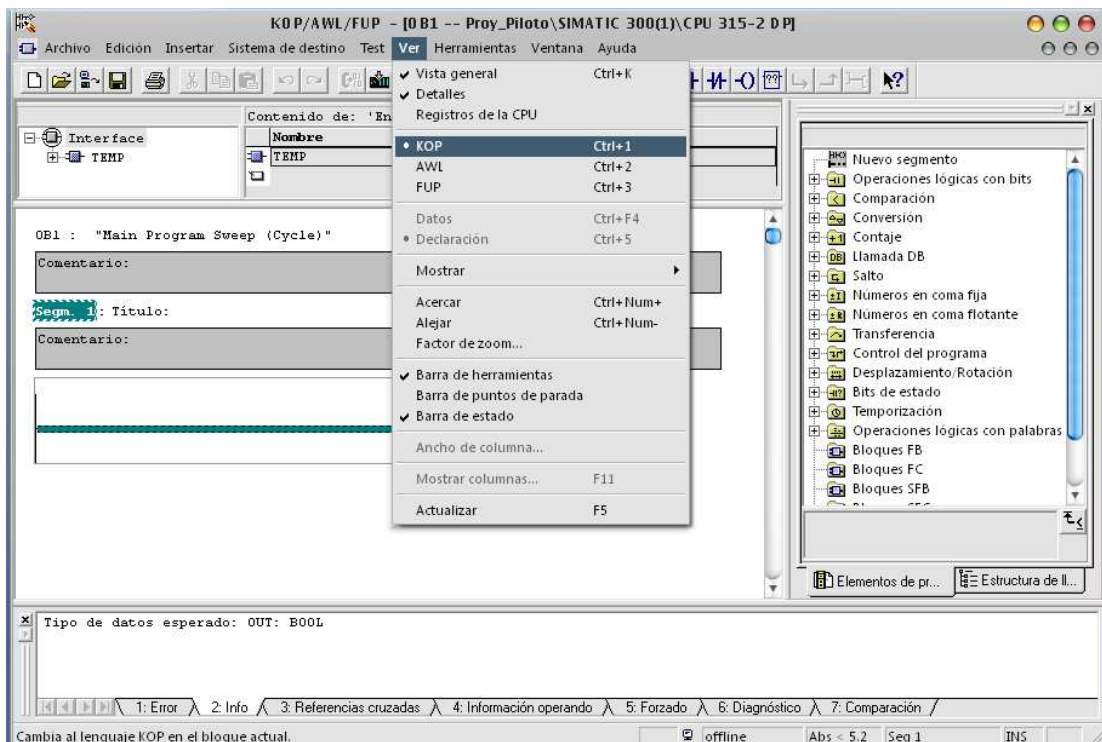
CREACIÓN DE UN PROGRAMA

- Una vez creado el proyecto y con el PLC configurado se puede escribir un bloque de programación. Para ello en el SIMATIC Manager dar doble clic en el OB1 dentro del menú del S7-300. (Proyecto→SIMATIC 300→CPU 315-2 DP→Programa S7→Bloques→OB1).




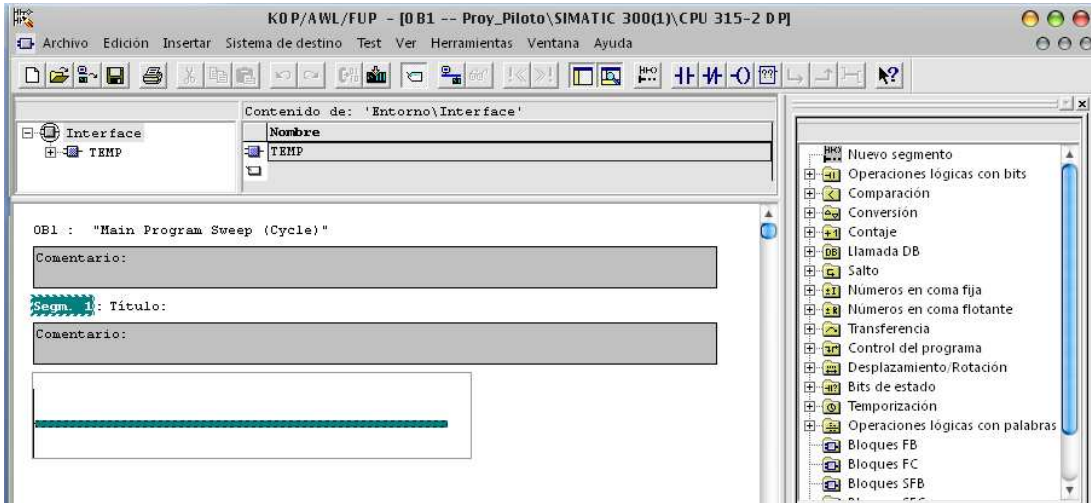
La nueva ventana muestra las propiedades del Bloque, por defecto se deja el bloque tal cual y se da clic en Aceptar, hecho esto se despliega una ventana con el Editor de Programas.


- Para comenzar a escribir, se debe seleccionar el lenguaje que será utilizado, dentro del Editor de Programas se selecciona ya sea en lenguaje KOP, AWL o FUP dentro del menú "Ver" en este caso es usado el lenguaje KOP (Ver→KOP)

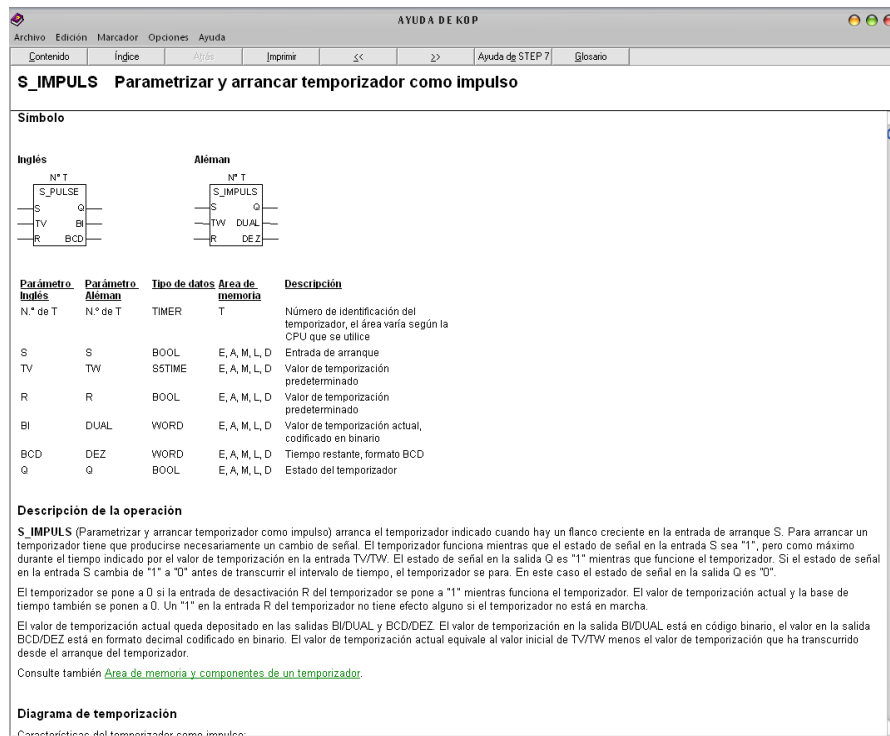


19. Después se selecciona el primer segmento para poder escribir la primera instrucción. Para agregar más segmentos

se da clic en el icono .



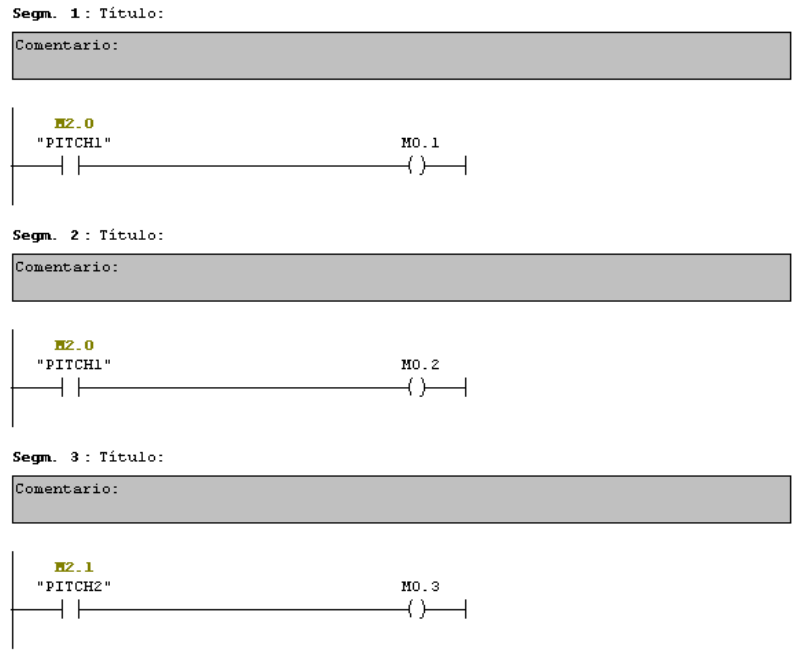
20. Con el icono de ayuda  dando clic sobre las funciones se despliega un menú que explica la simbología, la descripción de la forma de operación, diagrama de las funciones en caso de existir, palabras de estado y un ejemplo de su forma de uso.



21. Por último se guarda el programa , se depura y envía al PLC .

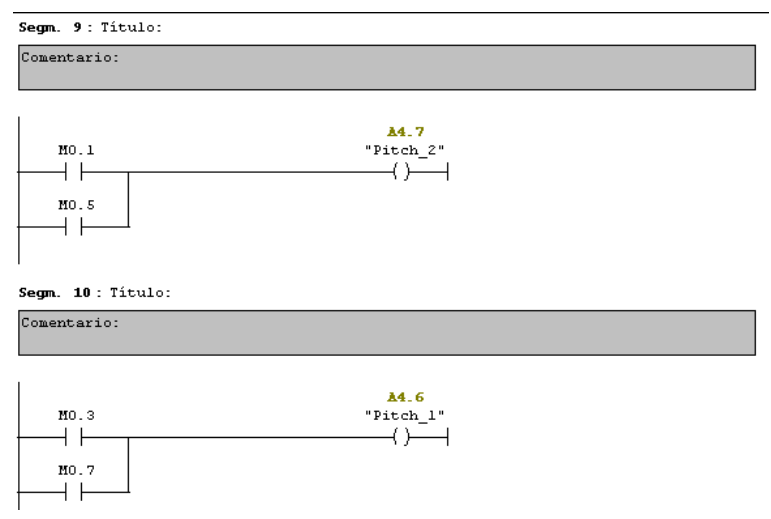
Para el giro del pitch o el roll, solamente es necesario establecer el sentido de giro de los dos motores encargados de estas tareas, esto es si giran hacia el mismo lado se tendrá movimiento de Pitch y si giran en sentido contrario habrá movimiento de Roll.

Para ello se creó una lógica dentro del OB1 del PLC, como se muestra a continuación

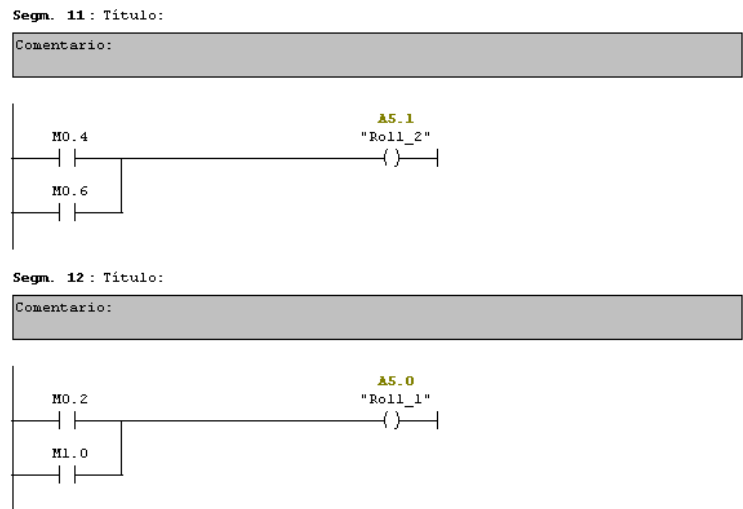


Definiendo Pitch1, Pitch2, Roll1 y Roll2 como las 4 salidas a encender, en diferentes combinaciones, y usando dos marcadores por cada salida como se muestra en la figura, se procede a encender las salidas usando dichos marcadores como se muestra a continuación.

Para el Pitch se tiene que



Mientras que para el Roll.



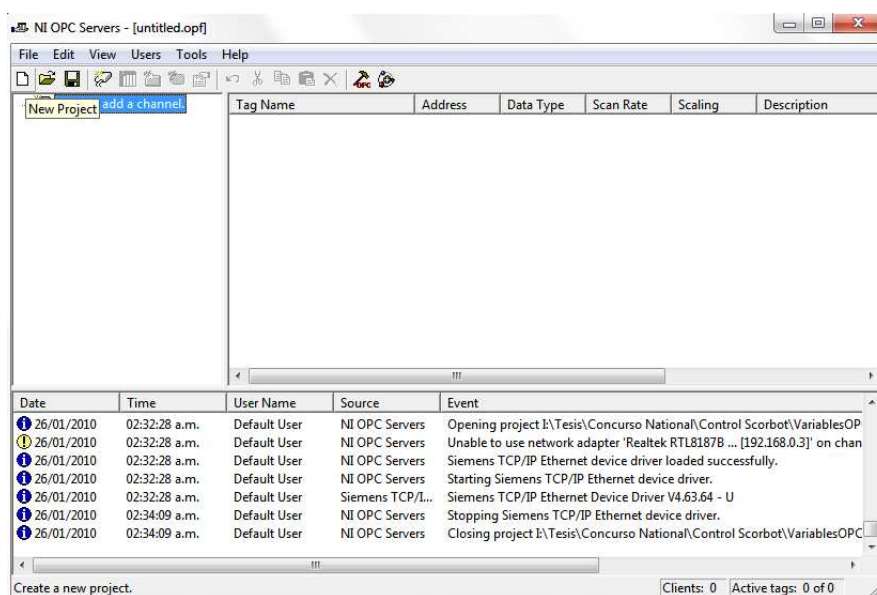
Con esto se logra crear las combinaciones de salidas que darán el movimiento necesario.


5.4 CONECTAR LABVIEW A UN PLC

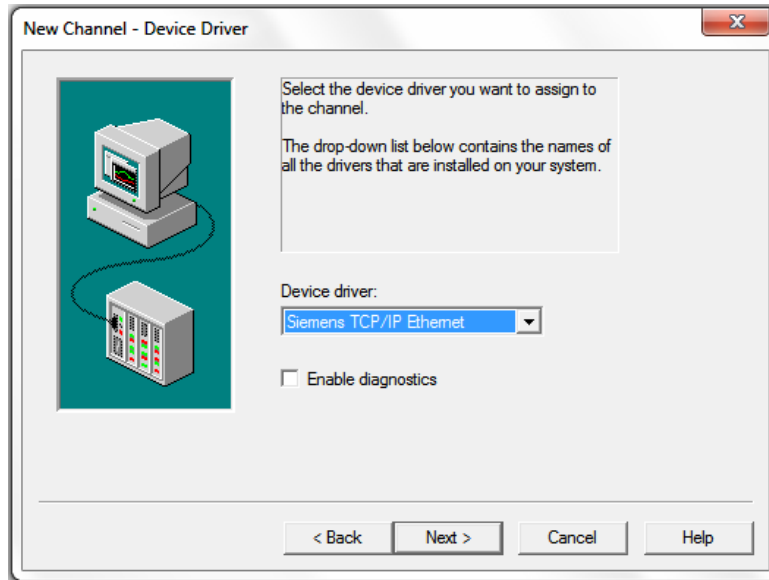
LabVIEW da la posibilidad de programar cualquier PLC en una amplia variedad de formas. OPC (OLE for Process Control) define el estándar para la comunicación de datos en tiempo real, entre los dispositivos de control y las interfaces HMI.

Para ello se debe hacer uso del módulo NI OPC Servers, con el cual se pueden crear, configurar, ver etiquetas que se asocian a las variables de entrada, salida, marcadores, temporizadores, contadores, etc. del PLC. Para dar de alta dichas etiquetas se siguen los siguientes pasos:


1. Se crea un nuevo proyecto dentro de NI OPC Servers.

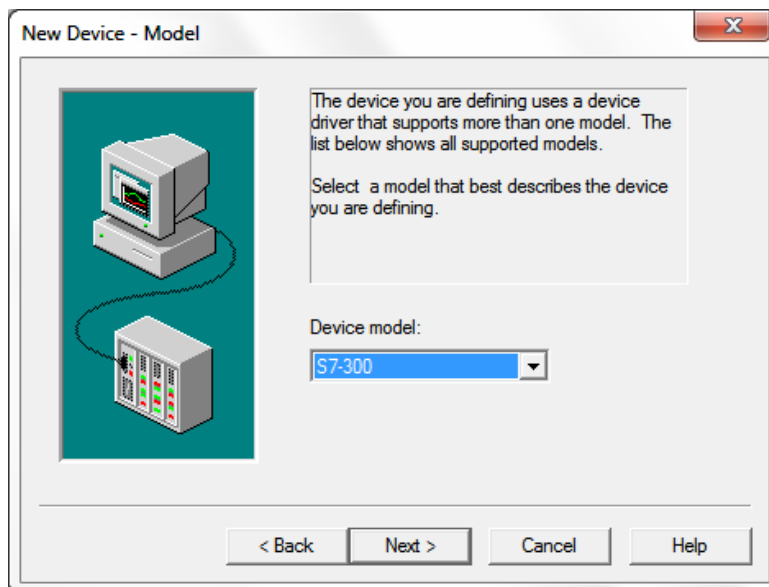



- Una vez creado, se agrega un canal , el cual contendrá la información acerca de la marca del PLC asociado, así como la forma de comunicación que se establecerá con el mismo. En este caso será un controlador Siemens TCP/IP Ethernet.

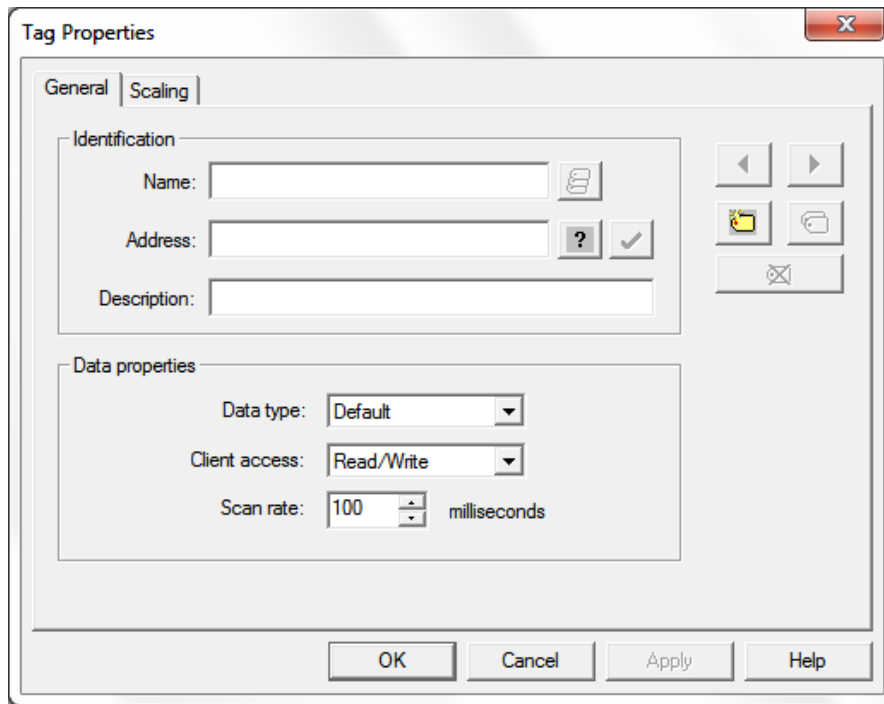


Posteriormente se selecciona el dispositivo presente en la PC por el cual se conecta al PLC, en este caso será una tarjeta de red inalámbrica con una dirección IP establecida. Se establece el tipo de escritura que se hará sobre las variables del PLC y se finaliza el asistente.

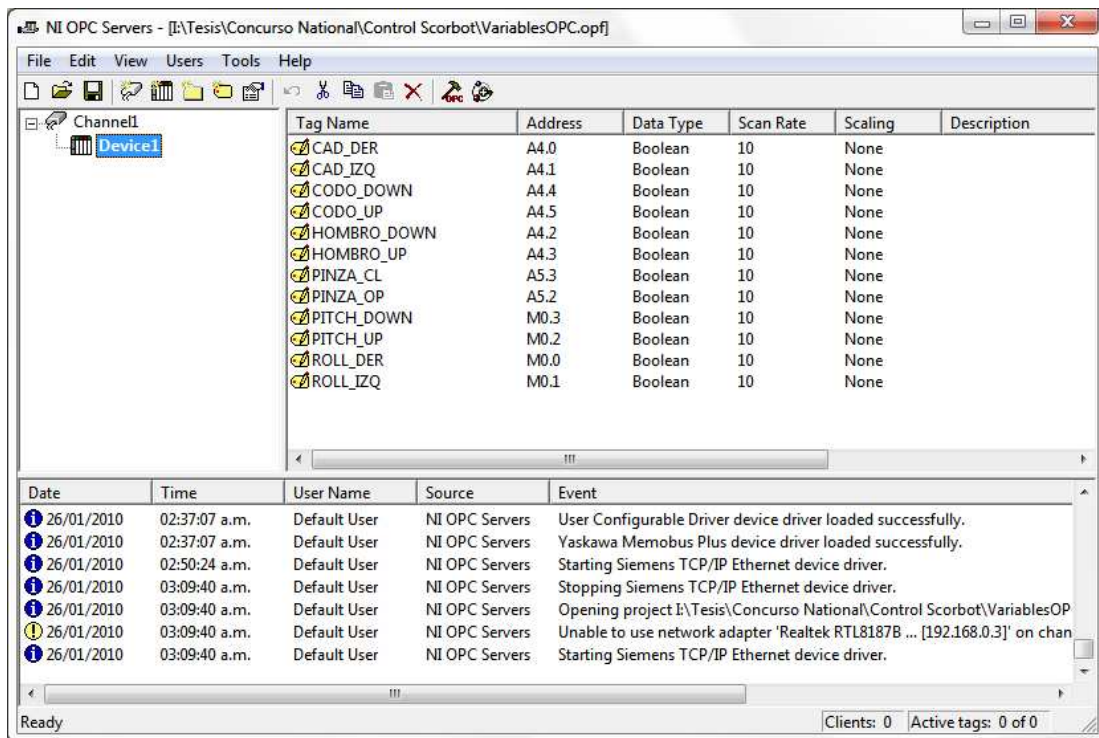
- Una vez que se tiene el canal, se procede a agregar un dispositivo , con lo cual se abrirá un nuevo asistente, dentro del cual se selecciona el PLC Siemens usado, en este caso el S7-300, así como su dirección IP (192.168.0.1), las demás configuraciones se dejan predeterminadas.



- Posteriormente se deben agregar las etiquetas , para las cuales se abrirá una nueva ventana en la cual se configurará un nombre para la etiqueta, la dirección con la cual está asociada dentro del PLC, así como el tipo de acceso que se tendrá (Escritura y/o Lectura) y el tiempo de muestreo.

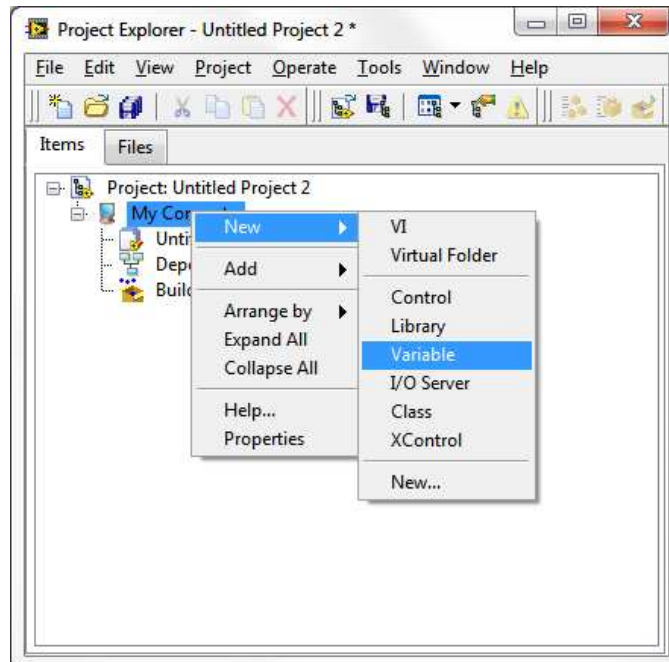


Se hace esto para cada de las variables quedando como se observa en la figura.

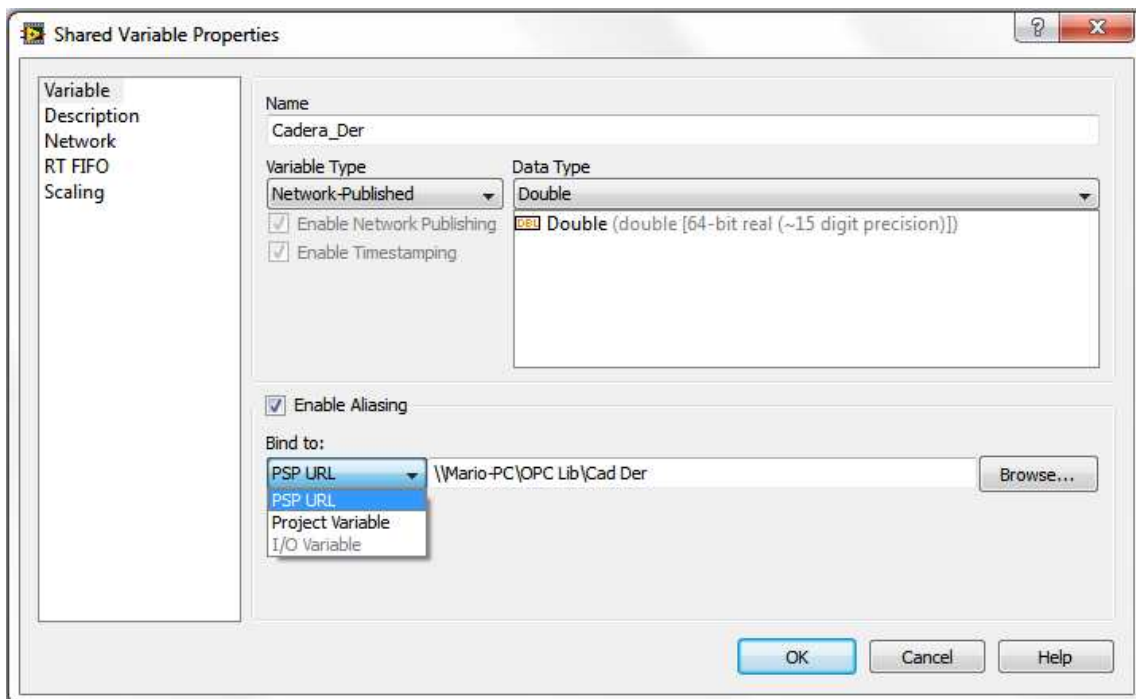


5.4.1 CONECTAR LABVIEW AL PLC USANDO UN SERVIDOR I/O.

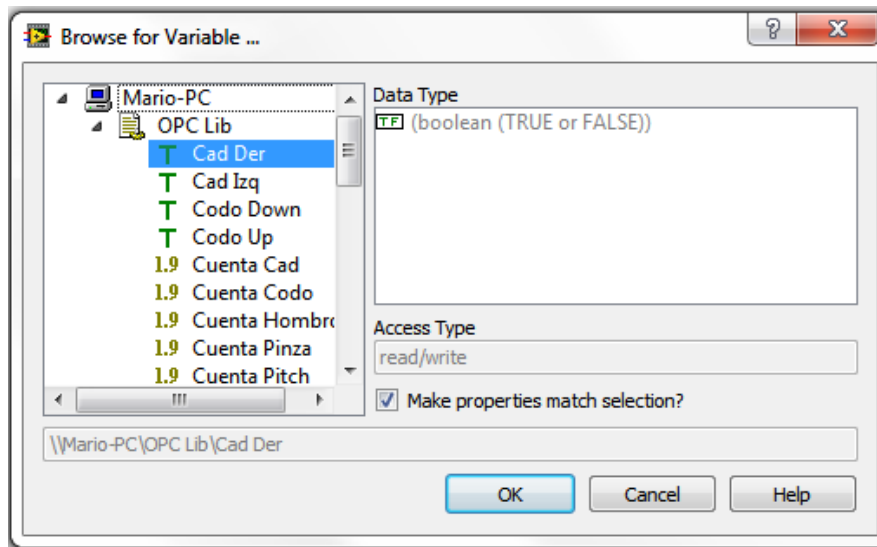
1. Para usar las etiquetas creadas anteriormente dentro de una aplicación, éstas deben ser agregadas al proyecto (New→Variable).



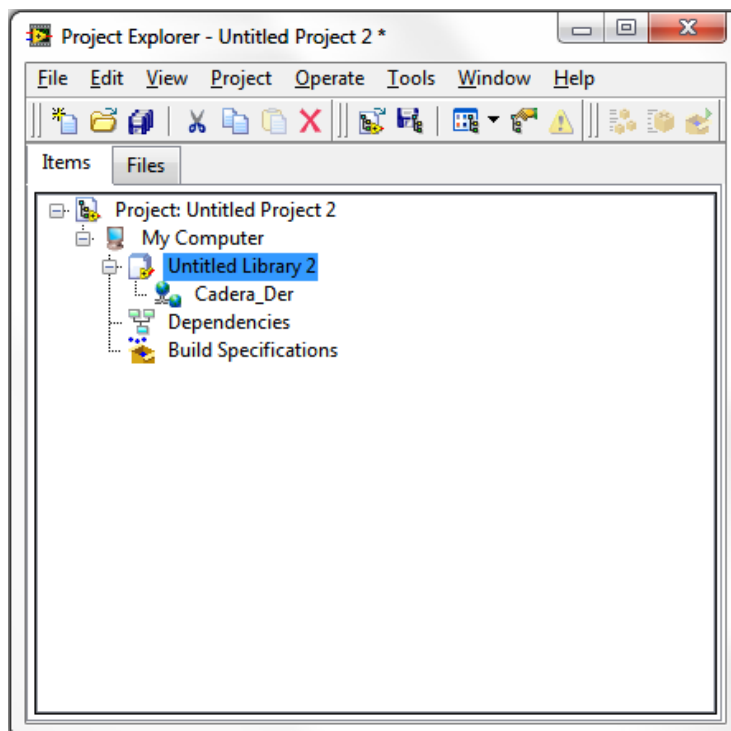
2. En la nueva ventana, se debe nombrar la variable, habilitar la casilla de "Enable Aliasing" y usar conexión tipo PSP URL.



3. Posteriormente establecer la dirección donde se encuentran las variables del servidor OPC.



Una vez hecho esto en la ventana principal del proyecto se observa la nueva variable asociada en este caso a la salida que habilita el giro de la cadera hacia la derecha.



Se hará lo mismo para cada variable usada, quedando como se muestra en la siguiente figura.

