UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

CONTROL DE UN BRAZO ROBOT DE CINCO GRADOS DE LIBERTAD MEDIANTE UN PLC

T E S I S

PARA OBTENER EL TÍTULO DE:

INGENIERO ELÉCTRICO ELECTRÓNICO

P R E S E N T A N:

GABRIEL TORRES MIRANDA

MARIO ALBERTO MATUS PÉREZ

DIRECTOR DE TESIS

M.F. GABRIEL HURTADO CHONG

Ciudad Universitaria, México, Mayo 2010

JURADO ASIGNADO:

PRESIDENTE:	**Profesor:**	M.I. RICARDO GARIBAY JIMÉNEZ
VOCAL:	**Profesor:**	M.F. GABRIEL HURTADO CHONG
SECRETARIO:	**Profesor:**	ING. GLORIA MATA HERNÁNDEZ
1er. SUPLENTE:	**Profesor:**	DR. JESÚS MANUEL DORADOR GONZÁLEZ
2° SUPLENTE:	**Profesor:**	ING. SERAFIN CASTAÑEDA CEDEÑO

SITIO DONDE SE DESARROLLÓ EL TEMA:

FACULTAD DE INGENIERÍA, UNAM

ASESOR DEL TEMA:

M.F. GABRIEL HURTADO CHONG
(nombre y firma)

SUPERVISOR TÉCNICO:

(nombre y firma)

SUSTENTANTE:

GABRIEL TORRES MIRANDA
(nombre y firma)

MARIO ALBERTO MATUS PÉREZ
(nombre y firma)

2 INTRODUCCIÓN

2.1 OBJETIVOS

Crear una solución viable para el reemplazo de los controladores originales de los robots Scorbot-ER V Plus, empleando un PLC y una interfaz HMI integrando LabVIEW y el módulo DSC.

Hacer un comparativo de la implementación de las funciones básicas para la manipulación del Scorbot, entre un PLC y un microcontrolador PIC.

Crear una interfaz gráfica la cual permita al usuario manipular los movimientos del robot así como observar una simulación de los mismos.

Plantear un apoyo académico para la enseñanza de Robots Industriales, diseñando una estructura de programa la cual permita aplicar la teoría y utilizar las aplicaciones necesarias para la simulación del robot, facilitando la posibilidad de aplicarlo para otro tipo de brazos.

Creación de manuales de apoyo para la enseñanza de PLC.

2.2 ALCANCES

El primer alcance previsto es la implementación de una interfaz electrónica la cual permita una fácil conexión entre el brazo robótico y los dispositivos que lo controlarán.

Presentar el uso de los microcontroladores PIC en la manipulación de los movimientos del robot mediante una simulación de su funcionamiento.

Diseñar una interfaz tanto en LabVIEW como en la HMI que contenga los controles para mover cada articulación del brazo de una forma manual, mediante la manipulación y monitoreo de las entradas y salidas del PLC.

Obtener una simulación 3D del brazo modelado en la cual se pueda aplicar la teoría del control de brazos robóticos específicamente de cinemática directa e inversa así como seguimientos de trayectorias.

Manual de uso del Simatic Manager para el S7-300 y la HMI OP 177B.

2.3 RESEÑA GENERAL

A lo largo del presente trabajo se plantean soluciones para operar un brazo robot de cinco grados de libertad por dos diferentes métodos; el primero se realizó usando un PLC Siemens S7-300 y una interfaz gráfica diseñada en LabVIEW utilizando una tarjeta Ni USB-6255 para conocer la posición del robot. El segundo se realizó con microcontroladores PIC, controlados desde una interfaz creada en Visual Basic, como se muestra en la Figura 2.1 .

Es indispensable comenzar detallando las aplicaciones y el propósito con el que son usados este tipo de Robots en la industria; para concluir si los dispositivos considerados para el control cumplen con los requerimientos. En el Capítulo 2 se muestra de manera general como está conformada una celda de producción, las clasificaciones de los brazos mecánicos y un breve preámbulo de los dispositivos de control.

Para realizar el proyecto se cuenta con un brazo robot de cinco grados de libertad Scorbot-ER V Plus, del cual se describen a fondo sus características físicas y de funcionamiento en el Capítulo 3. También se revisa la teoría de control de motores de corriente directa con encoder incremental, así como controladores PID y señales con modulación de ancho de pulso.

En el Capítulo 4 se explica el procedimiento para la creación de una interfaz electrónica que permita controlar el Scorbot el cual utiliza un conector DB50 para comunicarse con su controlador original. Se señala la función de cada uno de los pines de dicho conector y la manera en que son acopladas las señales que se reciben y se envíen a través de éste.

Posteriormente en el Capítulo 5, se describe la teoría alrededor del movimiento permitido por el brazo, en este capítulo, se basa todo el desarrollo de la interfaz gráfica diseñada en LabVIEW, ya que al comprender la cinemática del Scorbot y tras una breve introducción de las estructuras de programación utilizadas, es posible crear todos las subprogramas que serán el sustento de la gráfica 3D que simula al robot. También, como parte de esta unidad, se muestra como se establece una red entre un PLC S7-300 y una interfaz HMI Siemens, así como el monitoreo de esta desde LabVIEW usando el módulo DSC.

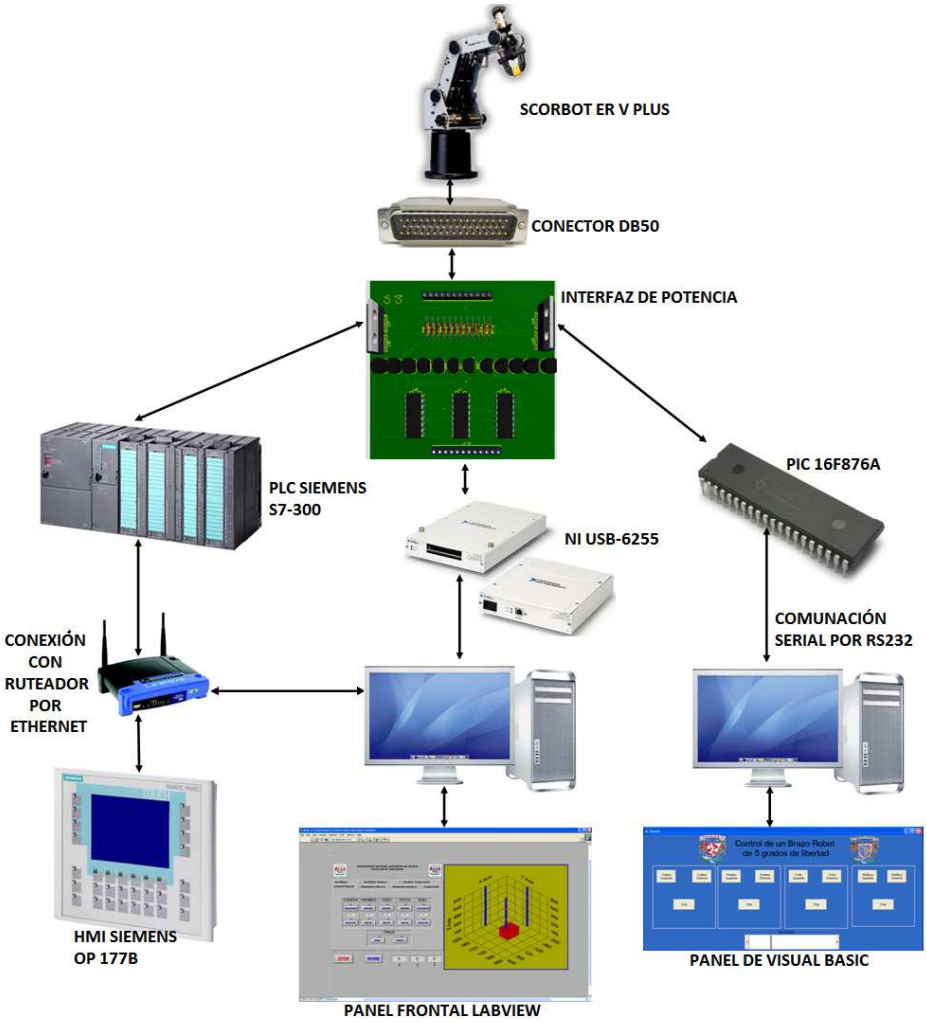


Figura 2.1. Diagrama de conexión de dispositivos para operar Scorbot-ER V Plus.

Después de haber configurado la red, se procede a explicar dentro del Capítulo 6 como usar los SubVI para crear una interfaz de usuario capaz de operar el Scorbot, utilizando la interfaz electrónica y el PLC. Asimismo, basados en las cinemática directa e inversa, es posible diseñar una simulación en 3D del brazo con la cual se pueda conocer su espacio de trabajo y probar trayectorias. Estas aplicaciones son muy útiles como apoyo académico, ya que con algunos conocimientos de LabVIEW es posible ajustarlas para diferentes tipos de robots.

Finalmente, en el Capítulo 7 se realiza un control basado en microcontroladores PIC creando una interfaz de usuario diseñada en Visual Basic, la cual por medio de puertos virtuales, se comunica con el software de desarrollo Proteus, donde están simulados los PIC, los motores y la interfaz electrónica necesaria. En este capítulo se plantean las dificultades que se presentaron durante ambos desarrollos, y la manera en que pueden enfrentarse.

2 ANTECEDENTES

2.1 AUTOMATIZACIÓN

Un sistema automatizado es un conjunto de dispositivos que trabajan juntos para ejecutar tareas o fabricar un producto o familia de productos. Los sistemas industriales automatizados pueden ser una máquina o un grupo de máquinas llamadas "celda". Los 4 tipos básicos de dispositivos en una celda son: de producción, de soporte, de control, y de retroalimentación.

2.1.1 DISPOSITIVOS DE PRODUCCIÓN

Se pueden incluir robots, máquinas de control numérico (CNC), máquinas de propósito específico (también llamadas Hard Automation Devices), etc. Los dispositivos de producción le agregan valor al producto. Estos realizan distintas etapas del proceso de manufactura como es el ensamblar, soldar, pintar y otras más que completan la tarea.

ROBOTS

Los robots son usados para diversas funciones dentro de la celda, en su mayoría de posicionamiento y de transporte de partes entre dos máquinas. Los robots son muy buenos para las tareas repetitivas, son muy rápidos y precisos. Para cada tarea, los diferentes tipos de robots (eléctricos neumáticos, o hidráulicos) tienen sus propias ventajas.

Los robots neumáticos son buenos para tareas de posicionamiento como mover partes entre otras máquinas. No son muy costosos, son rápidos y precisos, pero no son efectivos en tareas complejas y son muy limitadas en el número de posiciones en las que se pueden mover.

Por su parte los robots eléctricos son rápidos y más precisos, pero son más costosos que los neumáticos.

Los robots hidráulicos son muy buenos para aplicaciones pesadas. Son rápidos y se pueden mover suavemente, algo muy efectivo para aplicaciones de pintura. Son también muy usados en situaciones peligrosas donde chispazos eléctricos puedan causar explosiones.

MÁQUINAS CNC

Son máquinas cuyas acciones y movimiento son controladas por una computadora, la cual programa a la CNC con un códigos simples y números. Este tipo de máquinas incluyen tornos y molinos, máquinas de doblado de metales, máquinas láser o de descargas eléctricas para cortar metales, etc.

EQUIPO DE SOPORTE

Se incluyen sistemas automáticos de almacenamiento y recuperación (AS/RS), bandas transportadoras, y dispositivos de propósito específico, etc.

Los equipos AS/RS son usados para transportar materia prima, productos en proceso y productos terminados. El uso de computadoras asegura que el producto con más tiempo en el almacén sea usado primero. Un típico sistema AS/RS tiene

3 ejes de movimiento: el eje X que se mueve en toda la planta, el eje Y para movimientos verticales y el eje Z para moverse adentro y afuera y colocar el producto en el almacén.

Las bandas transportadoras son usadas para mover el producto entre celdas y procesos. Gracias a sensores muy simples o lectores de barras, la banda puede llevar el producto al lugar adecuado una vez identificado el tipo de producto presente.

Las máquinas de propósito específico son usualmente diseñadas para realizar una tarea en las que el uso de dispositivos más flexibles incrementaría el costo. Ejemplo de éstas son: las alimentadoras de partes uno a uno, las máquinas para alinear productos, paletizadoras, etc.

EQUIPOS DE CONTROL

Los Controladores de Lógica Programable (PLC) son los controladores más comunes en las celdas de producción, coordinan a todos los demás dispositivos, son el cerebro de la celda. Un PLC es una computadora especialmente diseñada para controlar e integrar otros dispositivos en el proceso y puede ser fácilmente manejada por técnicos[1]. Generalmente se programa en un lenguaje llamado lógica de escalera.

DISPOSITIVOS DE RETROALIMENTACIÓN

La retroalimentación se lleva a cabo a partir de sensores, estos son como los ojos y oídos de la celda. Proveen información de lo que está sucediendo al dispositivo de control.

Los sensores pueden ser tan simples como interruptores ON/OFF o más complejos como los de tipo analógico cuya salida es proporcional a la señal de entrada.

Los más usados son los que no tocan el objeto como son: foto sensores, sensores inductivos y capacitivos, ultrasónicos para medir la distancia al objeto, etc., ya que son más confiables y de lectura más rápida que los sensores mecánicos (interruptores).

Existen otros dispositivos de retroalimentación mucho más complejos, como son los lectores de código de barras o las cámaras y sistemas de visión.

2.2 BRAZOS MECÁNICOS

En 1948 Goertz del Argonne National Laboratory desarrolló, el primer telemanipulador, el cual era capaz de manipular elementos radioactivos sin riesgos para el operador, consistía en dos dispositivos mecánicos de los cuales el operador manipulaba el maestro y el esclavo reproducía los movimientos.

En 1954 sustituyen la transmisión mecánica, por una eléctrica y hacen uso de un servocontrol, con lo cual logran el primer telemanipulador con servocontrol bilateral. Después de este desarrollo hubo otros pioneros del tema, tal es el caso de Ralph Mosher, que en 1958 siendo parte del personal de General Electric, desarrolla Handy-Man el cual consistía en 2 brazos mecánicos teleoperados por un maestro del tipo exoesqueleto. Posteriormente la industria submarina, al igual que la industria espacial, se interesarían por esta tecnología.

[1] (Mandado Pérez, 2005)

La sustitución de un operador por un programa computacional, el cual se encarga de controlar el manipulador, dio el concepto de robot y de esta manera se empezó el diseño de estas máquinas, que en un principio mantenían configuraciones esférica y antropomórfica, debido a su gran utilidad para la manipulación de objetos.

En no más de 30 años el desarrollo de los robots los ha llevado a tomar un lugar en la mayoría de las áreas productivas, ya que los robots pueden realizar tareas repetitivas, tareas que conlleven un alto grado de precisión, o tareas que se deban realizar en lugares potencialmente peligrosas para un humano, y puede adaptarse inmediatamente a cualquier cambio en dichas tareas.

2.2.1 DEFINICIÓN Y CLASIFICACIÓN DEL ROBOT

Un robot, específicamente el robot industrial, "es una máquina de manipulación automática reprogramable y multifuncional con tres o más ejes que pueden posicionar y orientar materias, piezas, herramientas o dispositivos especiales para la ejecución de trabajos diversos en las diferentes etapas de la producción industrial, ya sea en una posición fija o en movimiento.[2]"

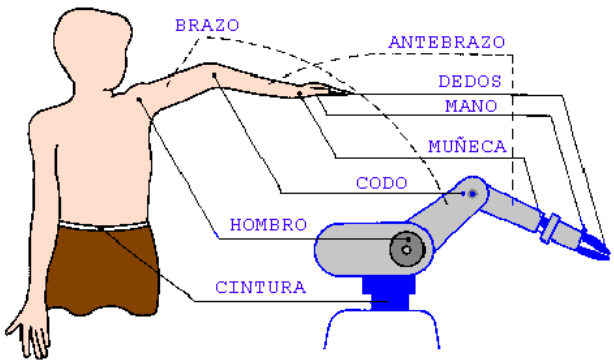
De igual manera, la Federación Internacional de Robótica estableció una clasificación para distinguir los tipos de robots.

Tabla 2.1 Clasificación de los robots industriales
Robot secuencial
Robot de trayectoria controlable
Robot adaptativo
Robot tele manipulado

2.2.2 ESTRUCTURA Y CARACTERÍSTICAS DE UN BRAZO MECÁNICO

ESTRUCTURA

Todo brazo robótico está constituido por eslabones unidos mediante articulaciones las cuales permiten el movimiento de dos eslabones consecutivos. Usualmente se hace una analogía con las extremidades superiores del cuerpo humano, de esta forma, como se observa en la Figura 2.1, cada eslabón tiene su homólogo con la estructura humana, por lo tanto para hacer referencia a los distintos elementos del brazo robótico se usan términos como cintura o cadera, hombro, codo, muñeca, etc.



[2] (Barrientos, 2007)

Figura 2.1. Analogía entre extremidad humana y brazo robótico industrial.

Para el movimiento de las articulaciones existe la posibilidad de deslizamiento o rotación, lo cual permite crear diferentes tipos de articulaciones con alguno de los movimientos mencionados o alguna mezcla de ambos como se muestra en la Figura 2.2, aunque las articulaciones más empleadas son las rotacionales y las prismáticas.

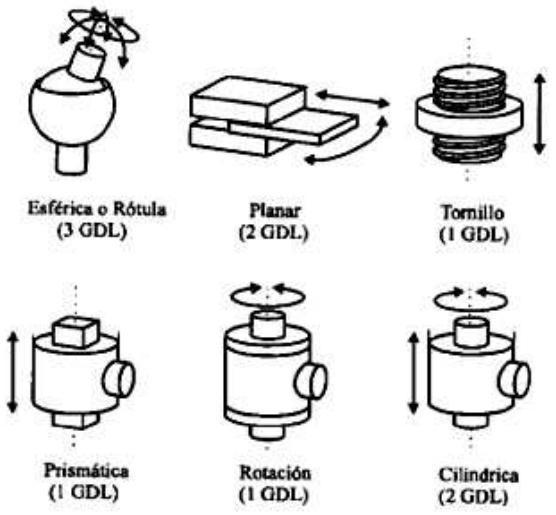


Figura 2.2. Tipos de articulaciones robóticas. (Barrientos, 2007)

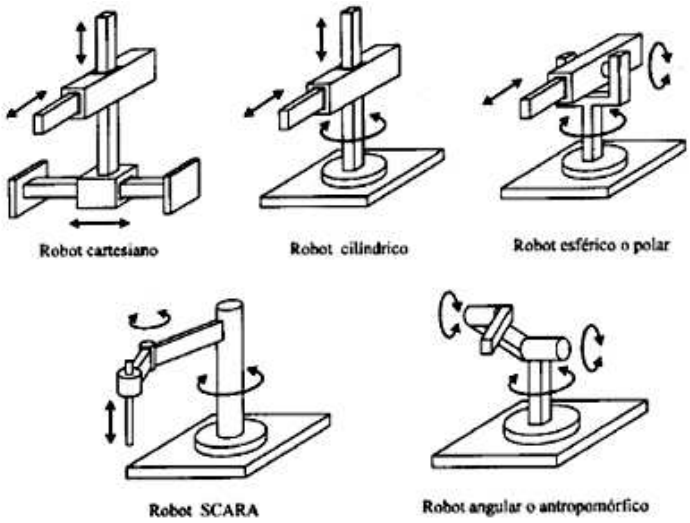


Figura 2.3. Configuraciones robóticas usadas en la industria. (Barrientos, 2007)

La mezcla de los diferentes tipos de articulaciones da lugar a diferentes configuraciones para los brazos robóticos, en la Figura 2.3 se muestran algunas de las configuraciones más usadas en la industria.

Con tales configuraciones se da lugar a los diferentes tipos de robot manipuladores, los cuales se clasifican a partir de su estructura cinemática, tomando en cuenta sus tres primeras articulaciones.

Antropomórficos (RRR). Sus tres articulaciones son de rotación.

Esféricos (RRP). Posee dos articulaciones de rotación y una prismática.

SCARA (RRP). Cuenta con dos articulaciones de rotación y una prismática.

Cilíndricos (RPP). Posee dos articulaciones primaticas y una rotacional.

Cartesianos (PPP). Posee tres articulaciones prismáticas.

ACTUADORES

Los elementos motrices que generan el movimiento de las articulaciones pueden ser eléctricos, hidráulicos o neumáticos. Los hidráulicos son los más empleados a la hora de manipular grandes cargas pero tienen un alto precio, mientras que los neumáticos dan una rápida respuesta a un bajo costo, por último los eléctricos cubren las necesidades de poco y mediano peso, además de que son muy empleados en la robótica por su alto control en la precisión de los movimientos. La **Tabla 2.2** muestra las características, ventajas y desventajas de cada uno de los tipos de actuadores mencionados.

Tabla 2.2. Características de los tipos de actuadores para robots			
	Neumático	**Hidráulico**	**Eléctrico**
Energía	Aire a presión	Aceite Mineral	Corriente Eléctrica
Opciones	Cilindros Motor de paletas Motor de pistón	Cilindros Motor de paletas Motor de pistones axiales	Corriente continua Corriente alterna Motor paso a paso
Ventajas	Precio Rapidez Sencillez Robustez	Rapidez Relación potencia-precio Autolubricantes Capacidad de carga Estabilidad frente a cargas estáticas	Precisión Fiabilidad Facilidad de control Sencilla instalación
Desventajas	Difícil de controlar Instalación Especial	Difícil Mantenimiento Instalación especial Fugas Precio	Potencia limitada

TRANSMISIONES

Son los elementos encargados de trasladar el movimiento desde los actuadores hasta las articulaciones, dentro de estos elementos se consideran los reductores acoplados a los motores. Debido a la importancia de querer reducir los momentos inerciales del manipulador, y de que los pares estáticos a vencer por los actuadores dependen directamente de las distancias de las masas al actuador, se intenta poner los actuadores cerca de la base del robot, lo cual lleva al uso de transmisiones en especial para las articulaciones más alejadas de la base del robot.

En la Tabla 2.3 se muestran las transmisiones para robots así como sus principales ventajas y desventajas.

Tabla 2.3. Sistemas de transmisión para robots			
Entrada-Salida	**Denominación**	**Ventajas**	**Desventajas**
Circular-Circular	Engranes Banda dentada Cadena Cable	Pares altos Grandes distancias Grandes distancias -	Holguras - Ruido Deformabilidad
Circular-Lineal	Tornillo sinfín Cremallera	Poca Holgura Holgura media	Rozamiento Rozamiento
Lineal-Circular	Cremallera	Holgura Media	Rozamiento

SENSORES

Para que un robot lleve a cabo las tareas establecidas, es necesario que en todo momento sepa la posición tanto de sus motores, como de su efector final. Esto se logra mediante los sensores internos que en todo caso van relacionados con el movimiento de los motores de cada articulación, en la **Tabla 2.4** se muestran los diferentes tipos de sensores.

Tabla 2.4. Tipos de sensores internos para robots.		
Presencia	Inductivo Capacitivo Efecto Hall Óptico Ultrasonido Contacto	
Posición	Analógicos	Potenciómetros Sincro Inductosyn LVDT
	Digitales	Encoder absoluto Encoder incremental
Velocidad	Tacogenerador	

EFECTOR FINAL

Los efectores finales son los encargados de llevar a cabo la tarea, en sí estos son los que tienen contacto directo con el área de trabajo y los elementos existentes en ella. Los más usados son los dispositivos de sujeción entre los cuales destacan las pinzas, ventosas, ganchos, imanes, etc. De igual manera se pueden acoplar otros tipos de herramientas como son pinza para soldar, soplete, cucharón, atornillador, fresadora, pistola de pintura, cañón laser, cañón de agua a presión, etc. En estos efectores finales es común usar un sensor que indique si está activa la herramienta, o si está abierta o cerrada en el caso de las pinzas, así como algunos otros sensores como pueden ser sistemas de visión, sensores de presión, etc.

CARACTERÍSTICAS

La estructura de la mayoría de los robots permite establecer ciertas características indistintas para todos los brazos robóticos, las cuales son explicadas a continuación:

1. **Cadena cinemática**. Es la estructura de los eslabones del robot.

2. **Grado de Libertad (GDL)**. Es el movimiento independiente que existe entre dos eslabones consecutivos del robot, por lo que dichos parámetros están relacionados con la suma de los grados de libertad existentes en cada articulación del robot. En la Figura 2.2 se observa el número de grados de libertad que tiene cada tipo de articulación.

   De este modo el número de grados de libertad indica la flexibilidad del efector final a la hora de posicionarse. A lo largo de la historia de los brazos robóticos se ha visto que son necesarios 6 grados de libertad (3 para posición y 3 de orientación) para alcanzar cualquier punto dentro del espacio de trabajo, dentro de la industria éste es el patrón seguido por la mayoría de los robots. Aunque también existen robots con un mayor número de grado los cuales son usados cuando existen obstáculos dentro de su área de trabajo. De igual forma se puede prescindir de ciertos GDL existiendo brazos de 4 ó 5 libertades cuando la tarea demanda movimientos más restringidos.

3. **Espacio de trabajo**. Está definido por el número de grados de libertad así como por las dimensiones y forma de los eslabones que constituyen al robot, considerando las limitantes del movimiento impuestas por el controlador. Para el cálculo del espacio de trabajo no se debe considerar la herramienta conectada a la

muñeca ya que en caso de ser cambiada es necesario volver a calcular dicho espacio. Un ejemplo de esto se muestra en la Figura 2.4 donde se observa el espacio de trabajo de un robot ROCCO.

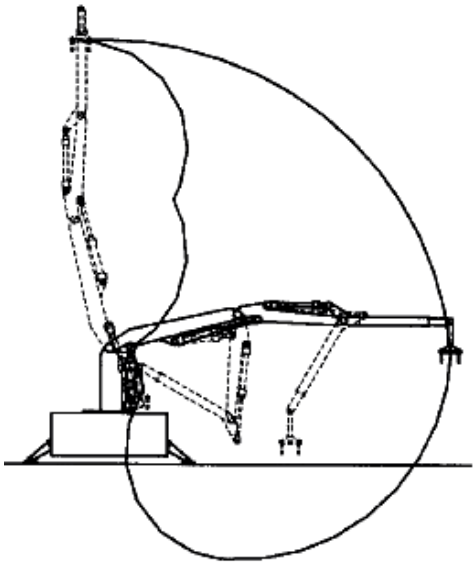


Figura 2.4. Espacio de trabajo de un robot ROCCO. (Barrientos, 2007)

REPETIBILIDAD, PRECISIÓN Y RESOLUCIÓN.

El gran desarrollo y uso de los robots en la industria se debe a su capacidad de posicionarse en el espacio indicado teniendo muy poco error, para el cálculo de dicho error se debe tener en cuenta los siguientes parámetros.

Precisión. Mide la distancia existente entre la posición deseada y la posición real del efector final del robot.

Repetibilidad. Es la capacidad del robot para llegar al mismo punto tantas veces como sea necesario, cabe mencionar que para este parámetro no se considera si el punto al que llega es el punto que se especificó, solamente se toma en cuenta la posición real y si ésta es consistente en cada movimiento.

Resolución. Mínimo movimiento del efector final que se puede efectuar desde el controlador.

En la Figura 2.5 se muestra gráficamente cada uno de los 3 parámetros.

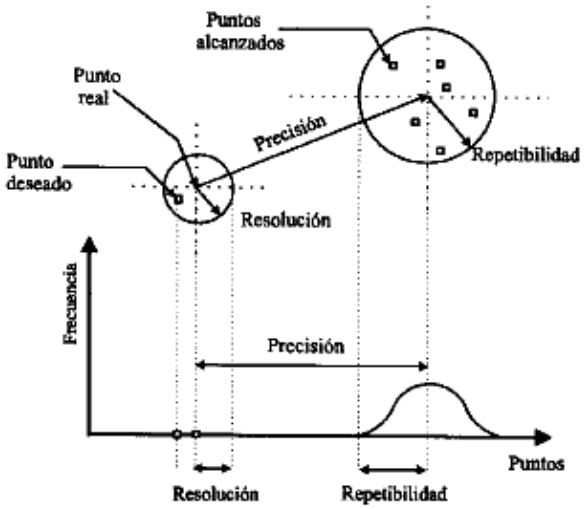


Figura 2.5. Representación gráfica de resolución, precisión y repetibilidad. (Barrientos, 2007)

Además se tienen los parámetros de velocidad y capacidad de carga los cuales son inversamente proporcionales. La velocidad es usada para el cálculo de los tiempos de ciclo, aunque muchas veces dicha velocidad no es alcanzada ya que el brazo hace movimientos cortos y rápidos.

Para la capacidad de carga se debe tomar en cuenta el peso de la herramienta y del objeto a mover, teniendo en cuenta que para un mayor peso la velocidad del manipulador disminuirá.

2.2.3 CONTROL

El objetivo del control se divide en dos áreas, el control cinemático y el control dinámico, así pues el control cinemático se lleva a cabo para establecer trayectorias que serán reproducidas por cada una de las articulaciones del robot para lograr en conjunto una trayectoria del manipulador con un fin en específico. Para la creación de dichas trayectorias se debe tomar en cuenta las limitantes físicas del robot, así como las posibles limitantes dentro de su área de trabajo. De igual manera se debe considerar el uso que se le dará a dicha trayectoria a cada momento para así poder establecer los instantes en los que se necesite mayor precisión o una fineza en cuanto a sus movimientos. Por otra parte, para el control dinámico su objetivo es, dadas las trayectorias establecidas por el control cinemático, seguirlas lo más fielmente posible, ya que por variables tales como son fricción e inercia, el movimiento del robot se ve afectado, esta diferencia entre el movimiento establecido y el movimiento real es la que se intenta minimizar usando el modelo dinámico del robot y las herramientas de la teoría de control.

2.3 CONTROLADORES DE LÓGICA PROGRAMABLE (PLC)

INTRODUCCIÓN

Los Controladores de Lógica Programable (PLC) o autómatas programables, "son una computadora industrial para la cual tanto su software como su hardware han sido adaptados para soportar las condiciones críticas de un ambiente industrial"[3]. Un PLC está compuesto en esencia por los mismos elementos que una PC, como los siguientes:

Consta de entradas las cuales pueden estar conectadas a varios tipos de dispositivos; también cuenta con salidas, estas dos partes así como la interfaz de programación que en muchos casos es una PC o un panel de programación, son las partes del PLC que interactúan con el humano, y mediante las cuales, la Unidad Central de Procesamiento (CPU) toma decisiones de lo que debe hacer.

Así, un PLC consta de:

- Módulos de entradas y salidas de señales discretas y analógicas así como módulos de entradas y salidas especiales. Estos módulos llevan conectados en las entradas sensores o interruptores, en las salidas arrancadores de motores, luces indicadoras ó válvulas y los módulos especiales llevan a cabo determinados procesos de la información que reciben, por ejemplo el conteo de pulsos eléctricos.
- Cuenta con módulo de procesamiento de comunicaciones para establecer una conexión con sistemas externos, tales como otros PLC, computadoras, interfaces hombre-máquina.
- Posee una unidad de memoria de acceso aleatorio (RAM), memoria E^2PROM y memoria flash.

CARACTERÍSTICAS

Para seleccionar un PLC es necesario tomar en cuenta ciertas características las cuales se indican a continuación.

UNIDAD CENTRAL

El CPU consta de uno o varios microprocesadores, de igual manera que una microcomputadora, así pues el CPU ejecuta el sistema operativo, controla la memoria, monitorea las entradas, evalúa su lógica y enciende las salidas apropiadas, además controla las comunicaciones con dispositivos externos. Dado el gran ruido eléctrico que se puede llegar a producir en un ambiente industrial, el cual puede ser generado por motores, cableados, máquinas de soldar y lámparas fluorescentes, los PLC están fuertemente aislados para evitar que se vean afectados por estas condiciones, y como medida precautoria el PLC crea rutinas de chequeo de la memoria para asegurarse de que la información grabada en el no ha sido modificada por dicho ruido, ésta sería la mayor diferencia entre un CPU de un PLC y un CPU de una computadora convencional

CAPACIDAD DE MEMORIA DE PROGRAMA/DATOS

Es el número de posiciones de las memorias de instrucción y de datos que es capaz de analizar seguidamente, esto es, la capacidad de analizar simultáneamente las entradas y salidas.

[3] (Mandado Pérez, 2005)

CAPACIDAD DE ENTRADAS Y SALIDAS DIGITALES

Se define como el número máximo de variables de entradas y salidas digitales que puede manejar un PLC. Siemens tiene una clasificación de sus autómatas programables, dejando como gama baja a la familia S7-200 los cuales tienen entre 6 y 24 entradas digitales y de 4 a 16 salidas digitales; la gama media comprende a la familia S7-300 las cuales poseen un número de variables de entradas/salidas entre 128 y 1024; por último, en la gama alta se tiene a la familia S7-400 en la cual se maneja un número de señales de variables de entrada/salida comprendido entre 32,768 y 131,056.

MODULARIDAD DE ENTRADAS Y SALIDAS

Es un concepto que hace referencia a la característica de un sistema que le permite ampliar sus capacidades al añadir módulos sin necesidad de modificar o anular los existentes, en el ámbito de los PLC esto se refiere a su capacidad de agregar módulos de entradas/salidas ya sean discretas o analógicas.

Usando este concepto se tiene una clasificación de tres tipos de autómatas:

- AUTÓMATAS PROGRAMABLES TOTALMENTE MODULARES. Son aquéllos cuya unidad central no tiene módulos fijos, sino que se conectan los módulos necesarios dependiendo de la aplicación.
- AUTÓMATAS PROGRAMABLES SEMI-MODULARES. Estos cuentan en su unidad central con un módulo de entradas y salidas fijo, pero prevén la posibilidad de agregar algún módulo para aumentar las capacidades del PLC.
- AUTÓMATAS PROGRAMABLES COMPACTOS O NO MODULARES. Son los PLC que en su unidad central contienen un número determinado y fijo de terminales, y sin capacidad de agregar módulos para aumentar sus capacidades.

CAPACIDAD DE INTERRUPCIÓN

Dado que los PLC llevan a cabo una lectura de sus entradas para el posterior manejo de salidas, esto limita bastante el uso de dichos dispositivos como máquinas en tiempo real, opto por usar las interrupciones que posee el microprocesador. Una interrupción se ejecuta cuando un suceso específico se presenta, con lo cual el PLC deja de realizar su tarea actual, para pasar a ejecutar alguna subrutina una vez terminada dicha subrutina el autómata sigue ejecutando la tarea que había dejado pendiente.

Existen varios tipos de interrupción, de las cuales las más usadas son:

- De reloj
- Temporizadas
- De contador
- De comunicación
- De terminales o bornes

INTERFAZ MÁQUINA –USUARIO

En algunos casos cuando se tiene un control de una máquina compleja, el operador debe proveer cierta información al PLC así como recibir el estado del proceso, o de algunas de sus variables, mientras se está ejecutando. En este caso se hace uso de una interfaz humano-máquina (HMI), entre las que destacan los paneles de operación (OP) los cuales cuentan con una pantalla o display y un teclado; así como los paneles táctiles (TP), los cuales cuentan con un display el cual posee sensores sensibles al tacto, con lo cual se elimina el teclado.

De esta forma un esquema completo de un PLC y sus componentes se muestra en la Figura 2.6.

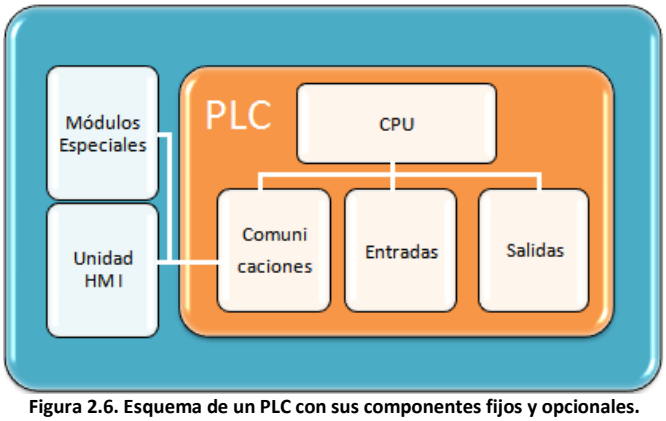


Figura 2.6. Esquema de un PLC con sus componentes fijos y opcionales.

LENGUAJE DE PROGRAMACIÓN

El lenguaje de programación típico de los autómatas programables se denomina lenguaje de escalera. En específico, Siemens desarrolló el sistema STEP7, dentro del cual se usan los siguientes lenguajes de programación

- **Lenguaje de lista de instrucciones (AWL)**. Consiste en un conjunto de códigos simbólicos, cada uno corresponde a una o más operaciones o instrucciones, este lenguaje se aproxima mucho al lenguaje ensamblador usado en los microprocesadores.

- **Lenguaje de esquema de contactos o diagrama de escalera (KOP)**. Recibe este nombre porque la tarea que debe realizar se representa gráficamente mediante un esquema de contactos. Este lenguaje es usado para:

    - Facilitar el cambio de un sistema de control realizado con relevadores
    - Hacer más fácil el diseño de sistemas sencillos de control.

  Se caracteriza por representar las variables de entradas y salidas mediante contactos normalmente abiertos o normalmente cerrados.

- **Lenguaje de diagrama de funciones (FUP).** Es un lenguaje simbólico en el cual las diferentes variables y operaciones se combinan mediantes símbolos de compuertas lógicas.

2.4 MICROCONTROLADORES

Recibe el nombre de controlador el dispositivo que se emplea para el gobierno de uno o varios procesos. Anteriormente los controladores se construían exclusivamente con componentes de lógica discreta, posteriormente se emplearon microprocesadores rodeados de chips de memoria y Entradas / Salidas sobre un circuito impreso.

Un microcontrolador es, en esencia, un circuito integrado de alta escala de integración que conjunta dentro del encapsulado las unidades principales de una computadora: Unidad Central de Procesos, memoria RAM, memoria ROM, líneas de E/S, puerto serial o paralelo, timer, convertidores analógico/digital (A/D) y digital/analógico (D/A) etc.

Una ventaja además de su cualidad de guardar y ejecutar programas que lo hacen versátil, es que al ser un sólo circuito es fácil de integrar a diseños electrónicos grandes.

2.4.1 ARQUITECTURA BÁSICA

Aunque inicialmente todos los microcontroladores adoptaron la arquitectura von Neumann, en el momento presente se impone la arquitectura Harvard. La arquitectura de von Neumann se caracteriza por disponer de una sola memoria principal donde se almacenan datos e instrucciones de forma indistinta. A dicha memoria se accede a través de un sistema de buses único (direcciones, datos y control).

La arquitectura Harvard, como se muestra en la Figura 2.7, dispone de dos memorias independientes una, que contiene sólo instrucciones y otra, sólo datos. Ambas disponen de sus respectivos sistemas de buses de acceso y es posible realizar operaciones de acceso (lectura o escritura) simultáneamente en ambas memorias.

Figura 2.7. La arquitectura Harvard. (Angulo Usategui, 2003)

2.4.2 ¿QUÉ MICROCONTROLADOR EMPLEAR?

A la hora de escoger el microcontrolador a emplear en un diseño concreto hay que tener en cuenta multitud de factores, como la documentación y herramientas de desarrollo disponibles así como su precio, la cantidad de fabricantes que lo producen y por supuesto las características del microcontrolador (tipo de memoria de programa, número de temporizadores, interrupciones, etc.):

- **Costos.** Para un fabricante que usa un microcontrolador en su producto, una diferencia de precio en el microcontrolador es importante (el consumidor debe pagar además el costo del circuito integrado, el de los otros componentes, el diseño del hardware y el desarrollo del software). Si se desea reducir el costo es necesario tener en cuenta las herramientas de apoyo con las que se cuentan: emuladores, simuladores, ensambladores, compiladores, etc.

- **Aplicación**. Antes de seleccionar un microcontrolador es imprescindible analizar los requisitos de la aplicación:

    - Procesamiento de datos**.** Puede ser necesario que el microcontrolador realice cálculos críticos en un tiempo limitado. En ese caso se debe seleccionar un dispositivo suficientemente rápido para ello. Por otro lado, se necesita tener en cuenta la precisión de los datos a manejar: si no es suficiente con un microcontrolador de 8 bits, puede ser necesario recurrir a microcontroladores de 16 ó 32 bits, o incluso a hardware de punto flotante. Una alternativa más barata y quizá suficiente es usar bibliotecas para manejar los datos de alta precisión.

    - Entrada/Salida. Para determinar las necesidades de Entrada/Salida del sistema es conveniente dibujar un diagrama de bloques del mismo, de tal forma que sea sencillo identificar la cantidad y tipo de señales a controlar. Una vez realizado este análisis puede ser necesario añadir periféricos de hardware externos o cambiar a otro microcontrolador más adecuado para el sistema.

    - Consumo. Algunos productos que incorporan microcontroladores están alimentados con baterías y su funcionamiento puede ser tan vital como activar una alarma antirrobo. Lo más conveniente en un caso como éste puede ser que el microcontrolador se encuentre en estado de bajo consumo pero que despierte ante la activación de una señal (una interrupción) y ejecute el programa adecuado para procesarla.

    - Memoria. Para detectar las necesidades de memoria de alguna aplicación se requiere separar en memoria volátil (RAM), memoria no volátil (ROM, EPROM, etc.) y memoria no volátil modificable (E^2PROM). Este último tipo de memoria puede ser útil para incluir información específica de la aplicación como un número de serie o parámetros de calibración.

      En cuanto a la cantidad de memoria necesaria puede ser imprescindible realizar una versión preliminar, aunque sea en pseudo-código, de la aplicación y a partir de ella realizar una estimación de cuánta memoria volátil y no volátil es necesaria y si es conveniente disponer de memoria no volátil modificable.

    - Ancho de palabra. El criterio de diseño debe ser seleccionar el microcontrolador de menor ancho de palabra que satisfaga los requerimientos de la aplicación. Usar un microcontrolador de 4 bits supondrá una reducción importante en los precios, mientras que uno de 8 bits puede ser el más adecuado si el ancho de los datos es de un byte. Los microcontroladores de 16 y 32 bits, debido a su elevado precio, deben reservarse para aplicaciones que requieran sus altas prestaciones (Entrada/Salida potente o espacio de direccionamiento muy elevado).

    - Diseño de la placa. La selección de un microcontrolador concreto condicionará el diseño de la placa de circuitos. Debe tenerse en cuenta que quizá usar un microcontrolador barato encarezca el resto de componentes del diseño.

3 DESCRIPCIÓN DE LAS CARACTERÍSTICAS DEL BRAZO

El sistema robótico está seccionado en dos partes: el brazo mecánico y su controlador electrónico.

3.1 CARACTERÍSTICAS MECÁNICAS DEL BRAZO

El brazo robot Scorbot-ER V Plus está construido como brazo vertical articulado, de cinco grados de libertad y una herramienta no intercambiable que en este caso es una pinza. Las articulaciones son todas de revolución excepto la pinza cuyo movimiento es prismático (apertura y cierre) como se muestra en la Figura 3.1.

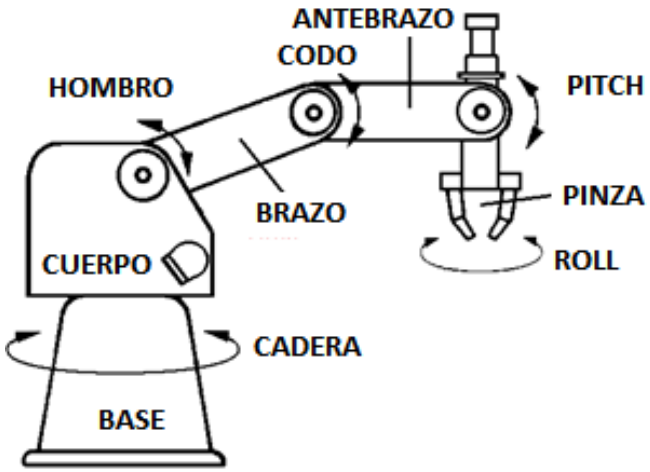


Figura 3.1. Articulaciones y eslabones del brazo robótico Scorbot-ER V Plus. (Intelitek, 2003)

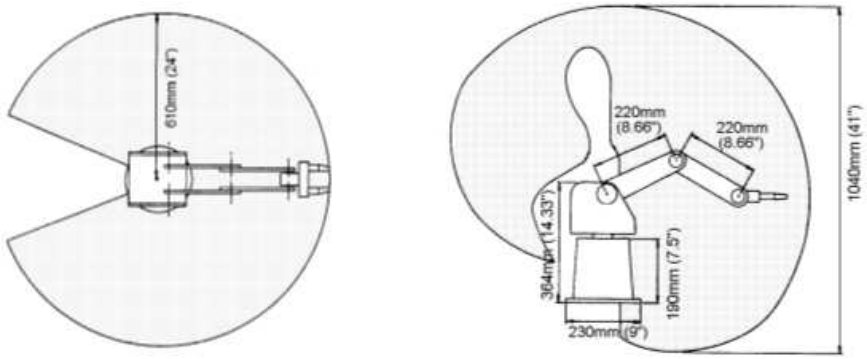
Las articulaciones están accionadas mediante motores, los cuales están acoplados indirectamente; esto es, el motor está montado lejos de las articulaciones y el movimiento del motor se transmite a través de bandas o engranes, lo que ayuda a que el peso de los motores quede sostenido por la base y no por cada una de las articulaciones, de igual forma permite variar la velocidad angular de cada articulación proporcionalmente a la velocidad del motor. En la Tabla 3.1 se observa el movimiento que realiza cada motor dentro del robot.

Tabla 3.1. Movimiento de cada articulación

# de Eje	Articulación	Movimiento	# de Motor
1	Cintura o Base	Rota el cuerpo	1
2	Hombro	Sube y baja brazo	2
3	Codo	Sube y baja antebrazo	3
4	Pitch	Sube y baja pinza	4+5
5	Roll	Rota pinza	4+5

ESPACIO DE TRABAJO.

La longitud de cada eslabón y la rotación determina el espacio de trabajo del robot, el cual puede ser observado en la Figura 3.2.

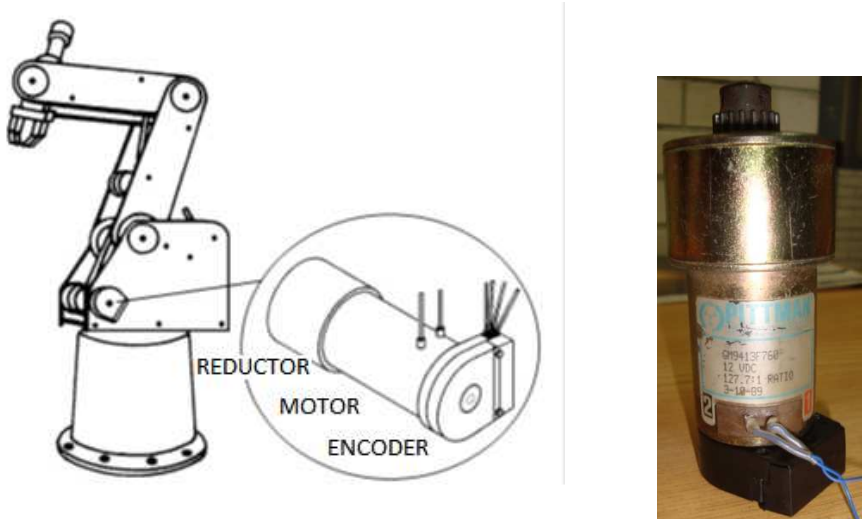


a) b)

Figura 3.2. Espacio de trabajo. a) Vista Superior, b) Vista Lateral. (Intelitek, 2003)

MOTORES Y TRANSMISIÓN

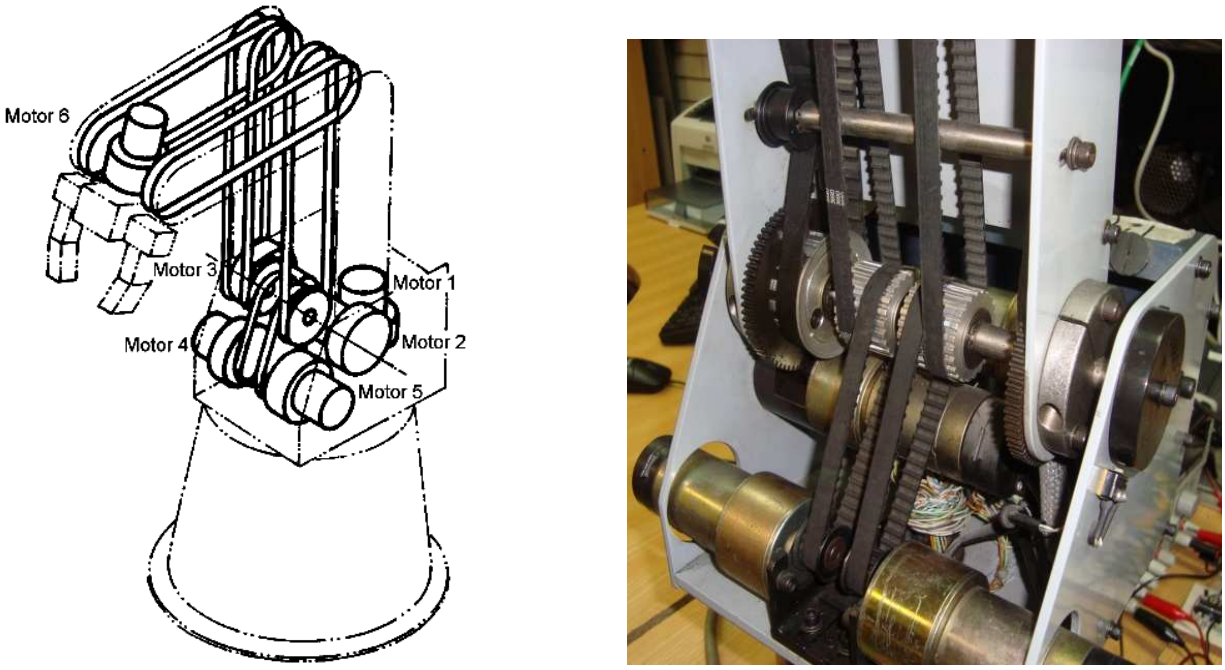
Las 5 articulaciones del robot y la pinza del efector son operadas por servomotores de corriente directa. La dirección de giro de cada articulación depende de la polaridad del voltaje de operación. Cada motor está acompañado por un encoder para control de lazo cerrado, como se observa en la Figura 3.3.



a) b)

Figura 3.3. a) Esquema de un motor de Scorbot. (Intelitek, 2003) b) Motor Pittman de 12 VDC.

Cada motor cuenta con diferentes tipos de transmisión, mientras que para la base y el hombro se usa una transmisión de engranajes dentados, para el codo se usan engranajes dentados y correas de regulación, para la muñeca se hace uso de correas de regulación y una unidad diferencial de engranajes dentados en el extremo del brazo, y en la pinza se transmite por medio de un tornillo de avance directamente acoplado al motor, en la Figura 3.4 se observa la disposición de dichas bandas. Cada articulación tiene un ángulo de giro limitado, los cuales son ilustrados en la Tabla 3.2.



a) b)

Figura 3.4. a) Disposición de las bandas de transmisión de Scorbot-ER V Plus. (Intelitek, 2003) b) Foto de las bandas de transmisión.

Tabla 3.2. Límite de giro en grados por articulación[1]	
Eslabón	**Limite [°]**
Cadera	310
Hombro	+130/-35
Codo	±130
Pitch	±130
Roll	Sin limite

Por otro lado la pinza tiene una apertura máxima de 65 [mm]

Todos los motores llevan un moto-reductor, para establecer un mayor torque, con lo cual el giro a la salida de la caja de engranes es menor al del motor, en la Tabla 3.3 se muestra la relación de transmisión de cada motor.

Tabla 3.3. Relación de transmisión de cada motor-reductor	
Motor	**Relación de transmisión (en vueltas)**
1,2,3	127.1:1
4,5	65.5:1
6 (Pinza)	19.5:1

[1] Especificaciones usando controlador original.

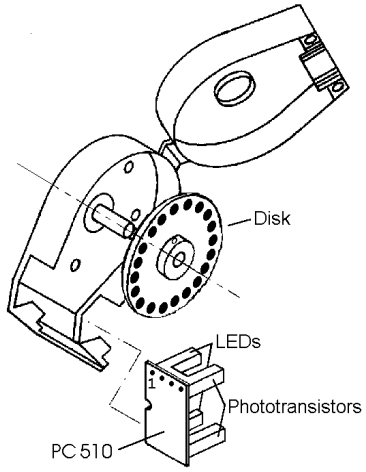
Con tales características de los motores y el controlador original se tienen las siguientes características mecánicas, ilustradas en la Tabla 3.4.

Tabla 3.4. Características mecánicas de Scorbot-ER V Plus[2]	
Carga Máxima	1 [kg] (incluyendo pinza)
Repetibilidad de la posición	0.5 [mm]
Peso total	11.5 [kg]
Velocidad Máxima	600 [mm/s]

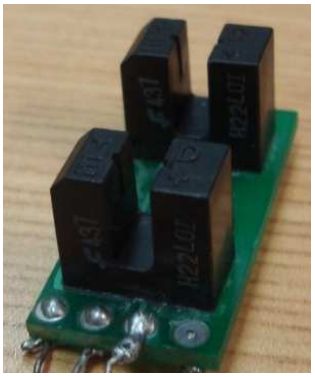
SENSORES

ENCODER

La localización y movimiento de cada eje está censada por un encoder óptico incremental, el cual está montado en la parte trasera del motor. Cuando se mueve el motor, el encoder genera una serie alternada de pulsos altos y bajos, el número de pulsos es proporcional al movimiento del eje, además la secuencia de pulsos indica el sentido de giro. En la Figura 3.5 se muestra uno de los tipos de encoder montado a los motores del manipulador.



a)



b)

c)

Figura 3.5. a) Encoder de Scorbot-ER V Plus. (Intelitek, 2003) b) Placa PC510. c) Encoder acoplado al robot.

[2] Especificaciones usando controlador original.

MICRO-INTERRUPTORES

El brazo mecánico cuenta con cinco micro-interruptores, uno por cada articulación, los cuales son usados para evitar choques entre las articulaciones así como el posicionamiento referencial (Home). Cuando todos los interruptores se encuentran activados indican que el brazo está ubicado en la posición de referencia o Home. Cuando el sistema es encendido, el robot debe ser enviado a dicha posición, mediante una rutina del software. En la Figura 3.6 se muestra un micro-interruptor de los montados en el robot.

Figura 3.6. Micro-interruptor de Scorbot-ER V Plus

PINZA

El Scorbot está provisto de una pinza mecánica la cual contiene cojines de agarre de goma, que pueden ser removidos para acoplar otro tipo de efectores finales como son los dispositivos de succión.

La muñeca tienen su movimiento basado en 3 engranes acoplados a los motores 4, 5 y 6, cuando se mueven los motores 4 y 5 en sentido contrario la muñeca se mueve hacia arriba o abajo, mientras que cuando se mueven en la misma dirección el movimiento que se realizará será horario o antihorario, de igual forma el motor 6 está acoplado directamente a la pinza mediante un tornillo sinfín el cual abre o cierra los dedos de la pinza.

3.2 CONTROL DE LOS ACTUADORES

MODULACIÓN DE ANCHO DE PULSO (PULSE WIDTH MODULATION PWM)

La modulación de ancho de pulso es una técnica para controlar circuitos analógicos con las salidas digitales de un procesador.

Una señal analógica tiene un cambio continuo de valor con una resolución infinita en tiempo y magnitud. Voltajes y corrientes analógicas son usadas para controlar dispositivos directamente como en el control de volumen en un estéreo; la perilla está conectada a una resistencia variable, conforme la perilla gira la resistencia aumenta o disminuye y por lo tanto la corriente que fluye se incrementa o decrementa. Intuitivamente podemos notar que en algunos casos no es muy económico o práctico, ya que los circuitos analógicos tienden a ser inexactos por varios factores como el ruido, por lo que al hacerlos más precisos se hacen más grandes y más caros.

Controlar sistemas analógicos digitalmente reduce drásticamente costos y consumo de energía. Además de que muchos microcontroladores incluyen controladores PWM lo cual hace aún más sencilla su implementación.

La PWM es una forma de digitalizar la señal analógica, usando las líneas digitales o de contadores de los procesadores. Con estos pines el ciclo de trabajo de la señal cuadrada es modulado para codificar un nivel específico de la señal analógica.

El *on-time* es el tiempo durante el cual la señal tiene un 1 lógico (5 volts) y la carga recibe alimentación de la fuente DC y el off-time es el tiempo donde la señal es un 0 lógico (0 Volts) y la carga no está alimentada, como se muestra en la Figura 3.7.

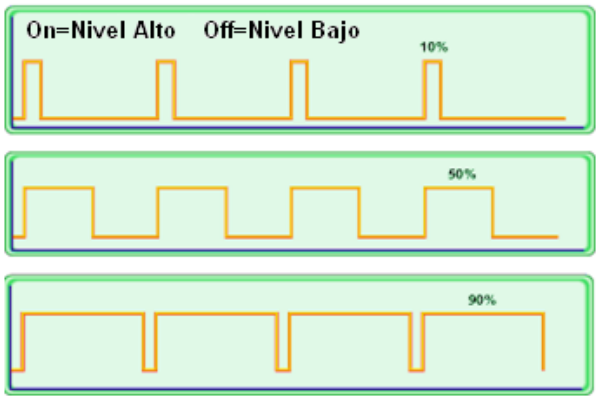


Figura 3.7. PWM con diferentes ciclos de trabajo. (Barr, 2001)

Algunas ventajas extras de la PWM es que no se requieren convertidores A/D o D/A para implementarla, además de que es casi inmune al ruido eléctrico, ya que éste solamente afectaría en el caso de que fuera un ruido muy fuerte que modificara el valor de 1 lógico a 0.

PARAMETROS DE CONTROL

En el controlador del sistema robótico de fábrica del Scorbot-ER V Plus, se tiene un sistema en lazo cerrado, donde el valor controlado es medido por cada uno de los encoders ópticos de los motores. La señal del encoder sirve como realimentación para el controlador, pudiendo corregir cualquier variación del valor deseado.

CONTROL PROPORCIONAL

El parámetro proporcional es la ganancia del sistema de control. Este valor determina el tiempo de reacción a los errores de posición.

Cuando existe un error en la posición, esto es si la posición actual del motor se encuentra fuera del valor requerido por n cuentas, el procesador multiplica el error por el parámetro proporcional y añade el producto al valor del convertidor digital-analógico el cual envía voltaje a los motores, reduciendo así el error.

El parámetro proporcional es el parámetro, dentro del sistema de control PID, que actúa más rápido en recudir el error en la posición, especialmente durante el movimiento. Además éste es el primer parámetro en responder al error en la posición cuando el robot es detenido en una posición específica.

Usando una componente proporcional grande, es más rápido la respuesta del sistema y la reducción del error. Pero a su vez un valor muy grande puede causar una oscilación permanente.

La principal desventaja del control proporcional es que por sí sólo no puede cancelar el error, pues una vez que reduce el error, no puede generar suficiente energía para superar la fricción en el sistema e impulsar al motor a su posición de destino.

Aun en estado estacionario, con carga, el valor controlado (señal de salida) siempre será diferente del valor deseado. El error en estado estacional puede ser reducido incrementando la ganancia, pero esto incrementa la oscilación y reduce la estabilidad.

CONTROL DIFERENCIAL

En el control diferencial, la salida controlada está en función de la velocidad de cambio en el error. Entre más rápido sea el cambio en el error, mas grande será la salida controlada. En otras palabras, el controlador es sensible a la pendiente de la señal de error.

El parámetro diferencial es responsable de reducir la velocidad del error. El sistema de control calcula la velocidad actual una vez por ciclo y compara con el valor deseado. Mientras el robot acelera (durante la primer parte de la ruta) el control diferencial actúa como un factor de impulso.

Mientras el robot desacelera (durante la segunda parte de la ruta) la componente diferencia actúa como un factor de frenado. Un buen ajuste en el valor diferencial propicia en un movimiento suave y limpio en toda la trayectoria. La falta de ajuste diferencial puede producir sobrepasos al final de la trayectoria. y un diferencial grande produce pequeñas vibraciones a lo largo de la trayectoria.

En este método de control, el controlador predice el valor del error tomando en cuenta la pendiente de la señal de error, y causa la corrección con antelación. Sin embargo, si el error es constante q invariable, el control diferencial no es capaz de reducir el error a cero.

CONTROL INTEGRAL

En el control integral, todos los errores que se han registrado cada ciclo se suman y el total se multiplica por el valor del parámetro integral.

En el control integral, la salida controlada, reduce la señal de error a cero a una taza proporcional al tamaño y duración del error. En otras palabras, a mayor error, ya sea en amplitud o tiempo, mayor será la salida controlada.

La principal ventaja del control integral es que el error en estado estacionario es siempre reducido a cero, ya que su valor aumenta cada ciclo, lo cual fortalece la capacidad del control para reaccionar y reducir el error. Sin embargo, usando un valor muy grande para el parámetro integral se puede propiciar sobrepasos, mientras que un valor muy pequeño puede prevenir la cancelación del error en estado estacionario.

A diferencia del parámetro proporcional, el integral tiene un efecto lento y es menos perceptible durante el movimiento. Sin embargo, cuando el motor hace un alto total y el parámetro proporcional ya no es capaz de reducir el error de estado estacionario, el parámetro integral tiene lugar y puede cancelar el error completamente.

4 DISEÑO DE LA INTERFAZ ELECTRÓNICA.

4.1 CONFIGURACIÓN DEL CONECTOR DB50.

El Scorbot viene provisto de fábrica de un conector DB50 el cual contiene el cableado hacia los elementos electrónicos del robot, en la tabla 4.1 se muestran los pines y su conexión con el robot así como su vínculo hacia la interfaz de potencia. En general se tienen 3 grupos de pines:

- Pines de alimentación de motores conectado al PLC mediante una interfaz de potencia la cual se muestra en la Figura 4.3.
- Pines de señales de los encoders.
- Pines de los microswitches.

Estos dos últimos grupos de pines están conectados a una tarjeta de adquisición de datos NI USB-6255 de National Instruments para su control con LabVIEW.

Tabla 4.1 Pines de conector DB50 de Scorbot-ER V Plus

Motores			
Eje	Motor	Número de Pin	Interfaz de Potencia
1	+	50	1Y (1)
	-	17	2Y (1)
2	+	49	4Y (1)
	-	16	3Y (1)
3	+	48	1Y (2)
	-	15	2Y (2)
4	+	47	3Y (2)
	-	14	4Y (2)
5	+	46	1Y (3)
	-	13	2Y (3)
Pinza	+	45	3Y (3)
	-	12	4Y (3)
Encoder			
Eje	Encoder	Número Pin	Conector en NI USB-6255 (Canales analógicos)
1	GND	33	-
	P_1	5	1
	V_{LED}	11	
	P_0	2	2
2	GND	32	-
	P_1	21	3
	V_{LED}	27	-
	P_0	1	4
3	GND	31	-
	P_1	4	5
	V_{LED}	10	-
	P_0	36	6
4	GND	30	-
	P_1	20	10
	V_{LED}	26	-
	P_0	35	11
5	GND	29	-
	P_1	3	12
	V_{LED}	9	-

	P_0	18	33
Tabla 4.2 (Cont.) Pines de conector DB50 de Scorbot-ER V Plus			
Pinza	GND P_1 V_{LED} P_0	28 19 25 34	- 34 - 35
Micro Switch			
Eje	Switch	Número Pin	Conector en NI USB-6255 (Canales digitales)
1	GND MS	33 23	- 97
2	GND MS	32 7	- 98
3	GND MS	31 24	- 99
4	GND MS	30 8	- 100
5	GND MS	29 6	- 101
Pinza	GND MS	28 22	- 102

Cabe mencionar que todos los pines que van referidos a tierra se conectaron a la tierra de la fuente utilizada.

4.2 SEÑALES DE CONTROL PARA MOTORES.

Para los motores se utiliza una circuito configurado como puente H, el cual se encuentra encapsulado en el integrado L293D del cual se puede ver su diagrama de pines en la Figura 4.1, dicho circuito contiene 4 circuitos para el manejo de cargas de potencia media, el cual es capaz de controlar corrientes de hasta 600 [mA] en cada circuito y un voltaje entre 4.5 y 36 [V]. El integrado permite formar, entonces, dos puentes H completos, con los que se puede realizar el manejo de dos motores. En este caso el manejo será bidireccional, con frenado rápido y con posibilidad de implementar fácilmente el control de velocidad.

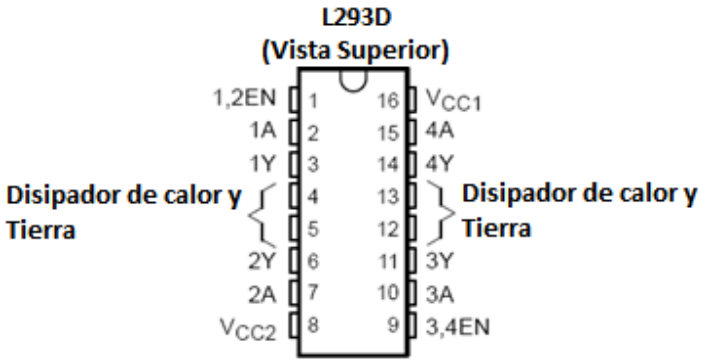
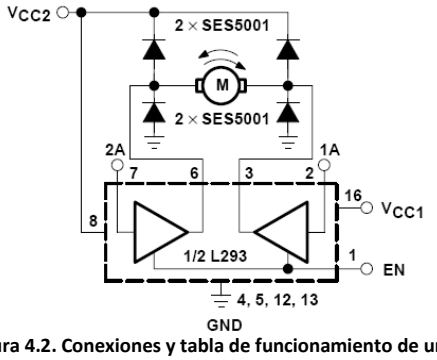


Figura 4.1. Diagrama de pines de puente H L293D. (Texas Instrument Inc., 2002)

Las salidas tienen un diseño que permite el manejo directo de cargas inductivas tales como relevadores, solenoides, motores de corriente continua y motores por pasos, ya que incorpora internamente los diodos de protección de contracorriente para cargas inductivas.

Las entradas son compatibles con niveles de lógica TTL. Para lograr esto, incluso cuando se manejen motores de voltajes no compatibles con los niveles TTL, el chip tiene pines de alimentación separados para la lógica (VCC2, que debe ser de 5V) y para la alimentación de la carga (VCC1).

Estos circuitos de salida se pueden habilitar en pares por medio de una señal TTL. Los circuitos de manejo de potencia 1 y 2 se habilitan con la señal 1,2EN y los circuitos 3 y 4 con la señal 3,4EN. Las entradas de habilitación permiten controlar de una manera muy sencilla el circuito, lo que permite tener una regulación de velocidad de los motores por medio de una modulación de ancho de pulso. En ese caso, las señales de habilitación en lugar de ser estáticas se controlarían por medio de pulsos de ancho variable. Las salidas actúan cuando su correspondiente señal de habilitación está en alto. En estas condiciones, las salidas están activas y su nivel varía en relación con las entradas. Cuando la señal de habilitación del par de circuitos de manejo está en bajo, las salidas están desconectadas y en un estado de alta impedancia. De esta forma cada integrado L293D controlará dos motores usando la configuración de la Figura 4.2.



EN	1A	2A	FUNCIÓN
H	L	H	Giro a la derecha
H	H	L	Giro a la izquierda
H	L	L	Detención rápida
H	H	H	Detención rápida
L	X	X	Detención rápida

L = bajo, H = alto, X = no afecta

Figura 4.2. Conexiones y tabla de funcionamiento de un integrado L293D para control bidireccional de un motor de DC.

Dado que el control del sentido de giro será habilitado por el PLC y este último entrega 24 VDC con una corriente de 20 [mA] a través de sus salidas del módulo digital, se debe hacer un acoplamiento entre el PLC y el integrado L293D, pues el integrado requiere más corriente para habilitar sus pines. Además se requiere bajar el voltaje de 24 a 5 [V], para ello se hace uso de un arreglo de resistencias como divisor de voltaje, y posteriormente se eleva la corriente usando un transistor NPN BC 547 para enlazarlo al integrado, como se muestra en la Figura 4.3.

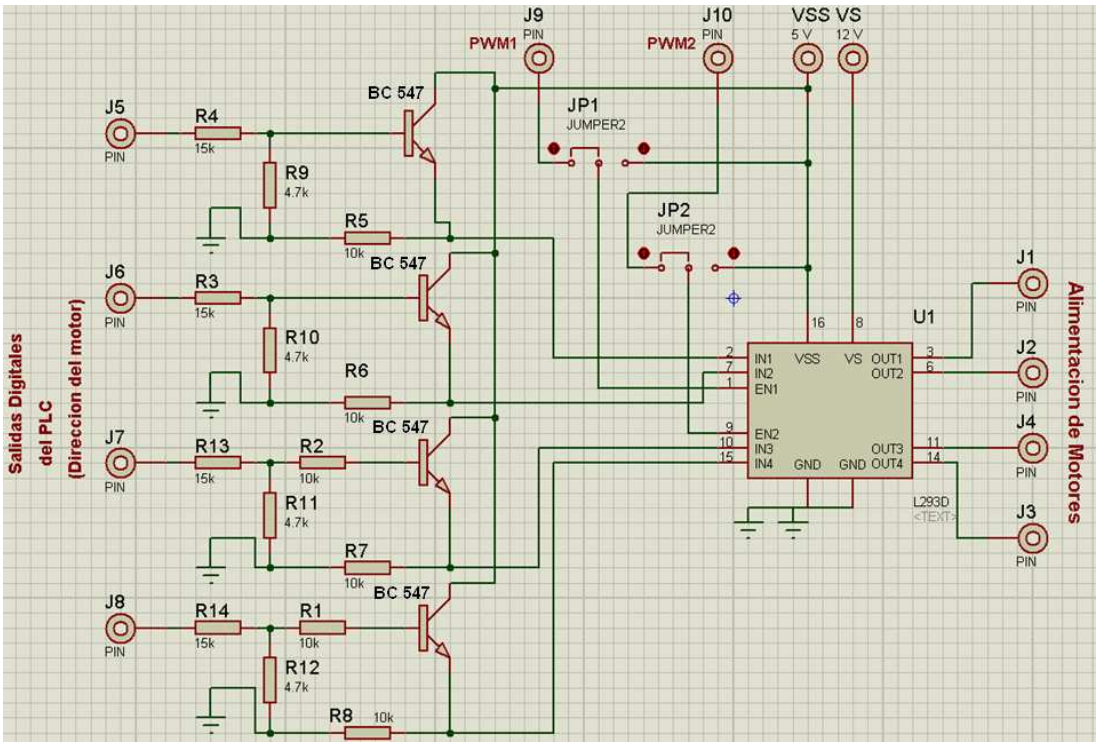


Figura 4.3. Circuito de control de dos motores mediante puente H conectado a salida digital de PLC.

4.3 SEÑALES DE RETROALIMENTACIÓN.

Para poder leer las señales de los fototransistores de cada encoder, se debe conectar el pin 2 a 5 [V] y el pin 1 a tierra como se muestra en la **Tabla 4.3**. Para cada uno de los fototransistores se debe realizar una conexión de colector abierto, es decir, se debe conectar una resistencia a VCC y el pin 3 y 4 se conecta a la otra terminal de la resistencia (ver Figura 4.4). De esta forma se obtienen las lecturas del pin 3 y 4, los cuales corresponden al par de fototransistores dentro de la placa PC510 contenida en el encoder. Se tomó en este caso una resistencia de 560 [Ω] pero lo importante es limitar la corriente, por lo que puede ser otro valor de resistencia.

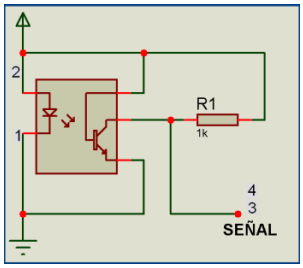


Figura 4.4. Conexión de los pines del integrado PC510.

En la Tabla 4.2 se enumeran los pines de la placa PC510 mostrada en la **¡Error! No se encuentra el origen de la referencia.** la cual contiene los fototransistores para el encoder de los motores.

Tabla 4.3. Pines de la placa PC510.

Función del encoder	# Pin
Fototransistor P0	3
Fototransistor P1	4
Voltaje alimentación [V+]	2
Tierra [GND]	1

Los fototransistores de los encoders entregan un tren de pulsos desfasados entre ellos (P0 y P1) como se muestra en la Figura 4.5, para el nivel bajo se tiene un voltaje menor a 0.4 V y para el nivel alto se tiene un valor mayor a 4 V.

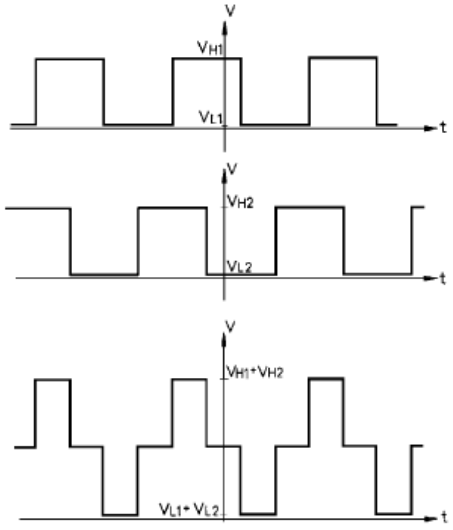


Figura 4.5. Tren de pulsos emitido por los fototransistores del encoder. (Intelitek, 2003)

Así como los encoders necesitan resistencias de pull-up los pines de los microswitches requieren una resistencia como se muestra en la Figura 4.6.

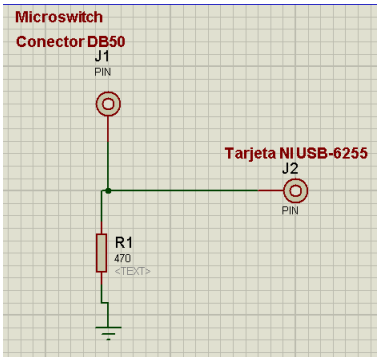


Figura 4.6. Conexión de microswitches con tarjeta NI USB-6255

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, ... q_n)	Cinemática directa → ← Cinemática inversa	Posición y orientación del extremo del robot (x, y, z, α, β, γ)

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.

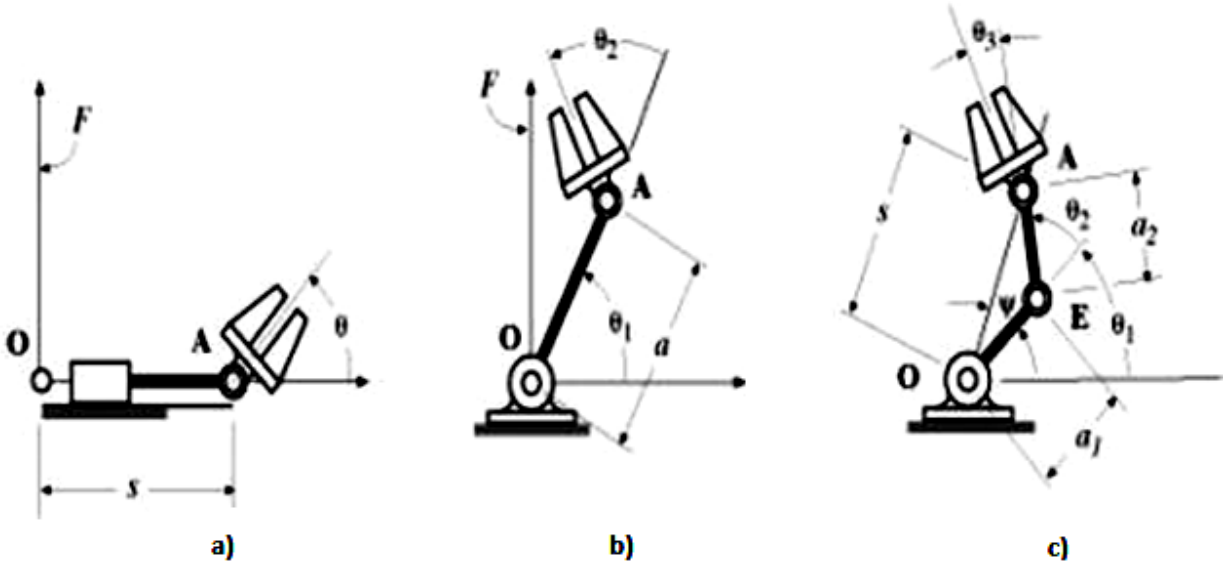


Figura 5.1. Articulaciones referidas a un sistema O situado en la base del robot y sus parámetros que los relacionan para: a) 1 articulación, b) 2 articulaciones, c) 3 articulaciones giratorias. (Barrientos, 2007)

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 . Así pues, describe la posición y orientación del sistema de referencia solidario al primer eslabón con respecto al sistema de referencia solidario a la base, 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 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 :

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

De manera análoga, la matriz representa la localización del sistema del tercer eslabón:

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

Cuando se consideran todos los grados de libertad, a la matriz 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:

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

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 (Si) 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 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 , relacione los sistemas y , 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} y 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 Xi y Xi-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 $T = {}^0A_n = {}^0A_1 \cdot {}^1A_2 \cdots {}^{n-1}A_n$.
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 y 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 desde el origen del sistema de coordenadas $(i-1)-$ hasta la intersección del eje con el eje . Se trata de un parámetro variable en articulaciones prismáticas.

Es a la distancia a lo largo del eje que va desde la intersección del eje con el eje 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 y .

Es el ángulo de separación del eje y el eje , medido en un plano perpendicular al eje , 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°	$\theta_1\,(0°)$
2	l_2	0	0	$\theta_2\,(0°)$
3	l_3	0	0	$\theta_3\,(0°)$
4	0	l_4	90°	$\theta_4\,(90°)$
5	0	l_5	0	$\theta_5\,(0°)$

5.1.2 CINEMÁTICA INVERSA.

El objetivo del problema cinemático inverso consiste en encontrar los valores que deben adoptar las coordenadas articulares del robot $q = (q_1, q_2, \ldots$ 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 explicita de la forma:

$$q_k = F_k(x, y, z, \alpha, ß, \gamma)$$
$$K = 1...n \text{ (grados de libertad)}$$

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 $(q_1, \ldots$ 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 liberta, 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 $(q_1, \ldots$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.

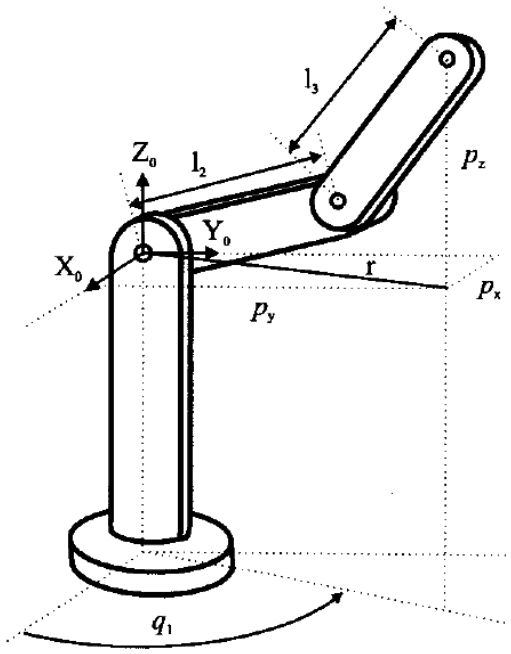


Figura 5.2. Parámetros para la cinemática inversa usando el método geométrico. (Barrientos, 2007)

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 $(P_X$, referidas a en las que se requiere posicionar su extremo.

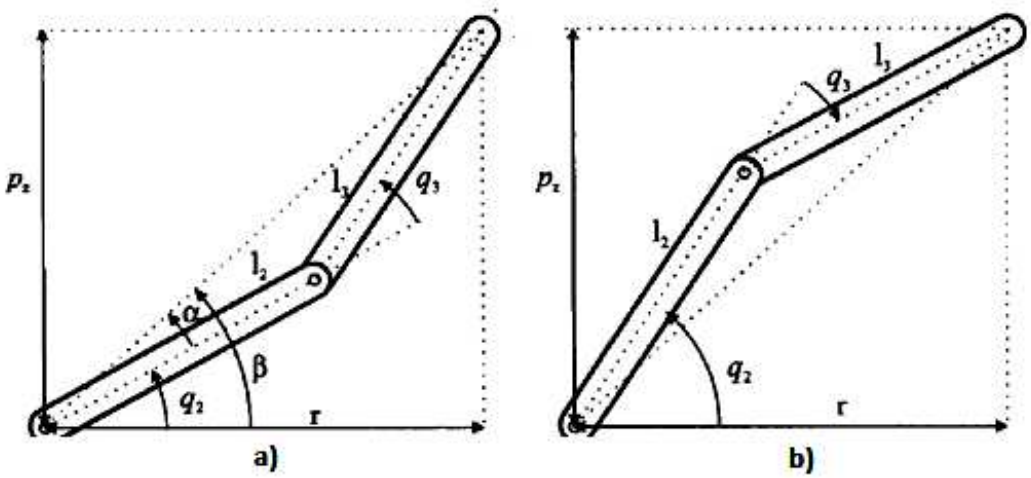


Figura 5.3. Configuración de parámetros para: a) codo arriba b) codo abajo. (Barrientos, 2007)

Como se ve, este robot posee una estructura planar, quedando este plano definido por el ángulo de la primera variable articular q1. El valor de q1 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 q3 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 q2 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, pos 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 llama 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.

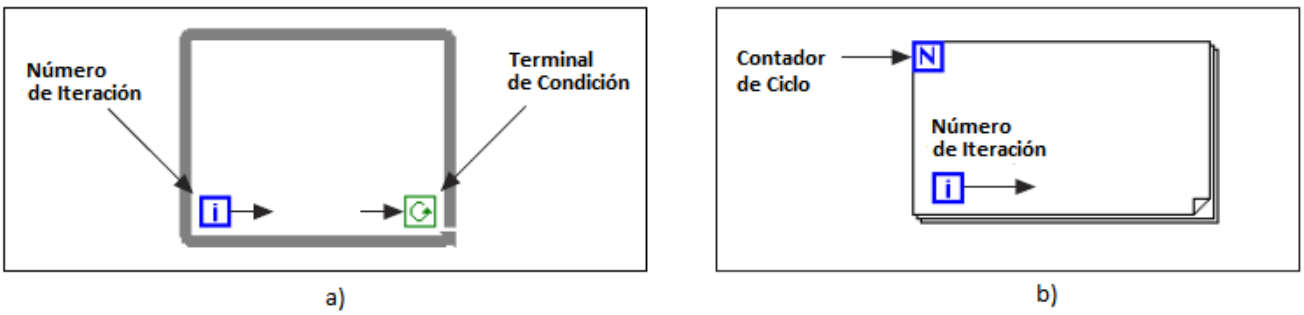


Figura 5.4. Estructuras cíclicas en LabVIEW. a) Estructura While b) Estructura For. (National Instruments, 1998)

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.

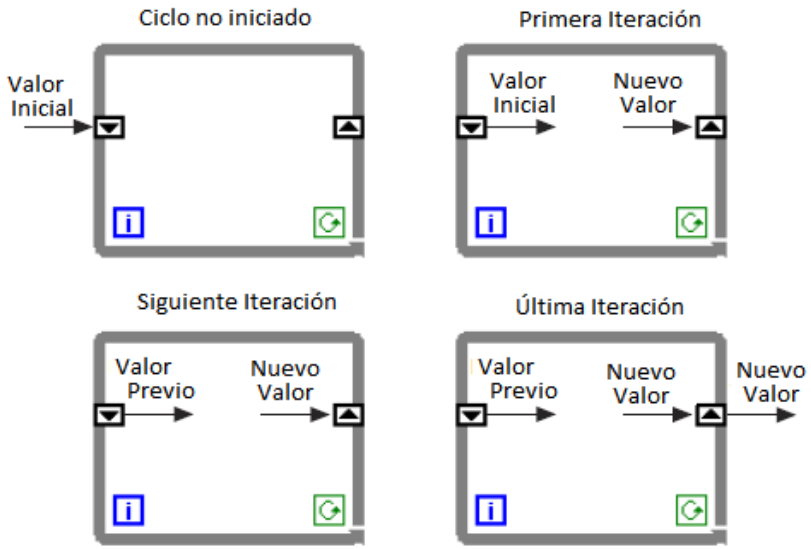


Figura 5.5. Seguimiento de los valores de Shift Register dentro de un ciclo While. (National Instruments, 1998)

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.

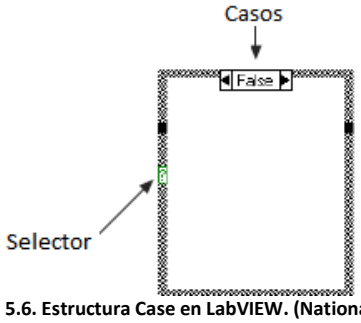


Figura 5.6. Estructura Case en LabVIEW. (National Instruments, 1998)

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.

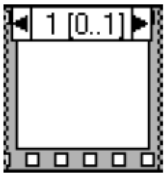


Figura 5.7. Estructura Sequence en LabVIEW. (National Instruments, 1998)

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.

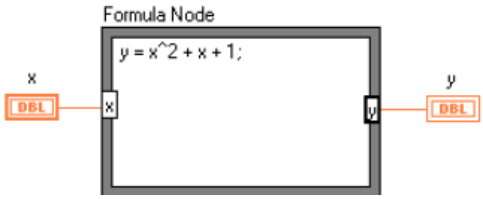
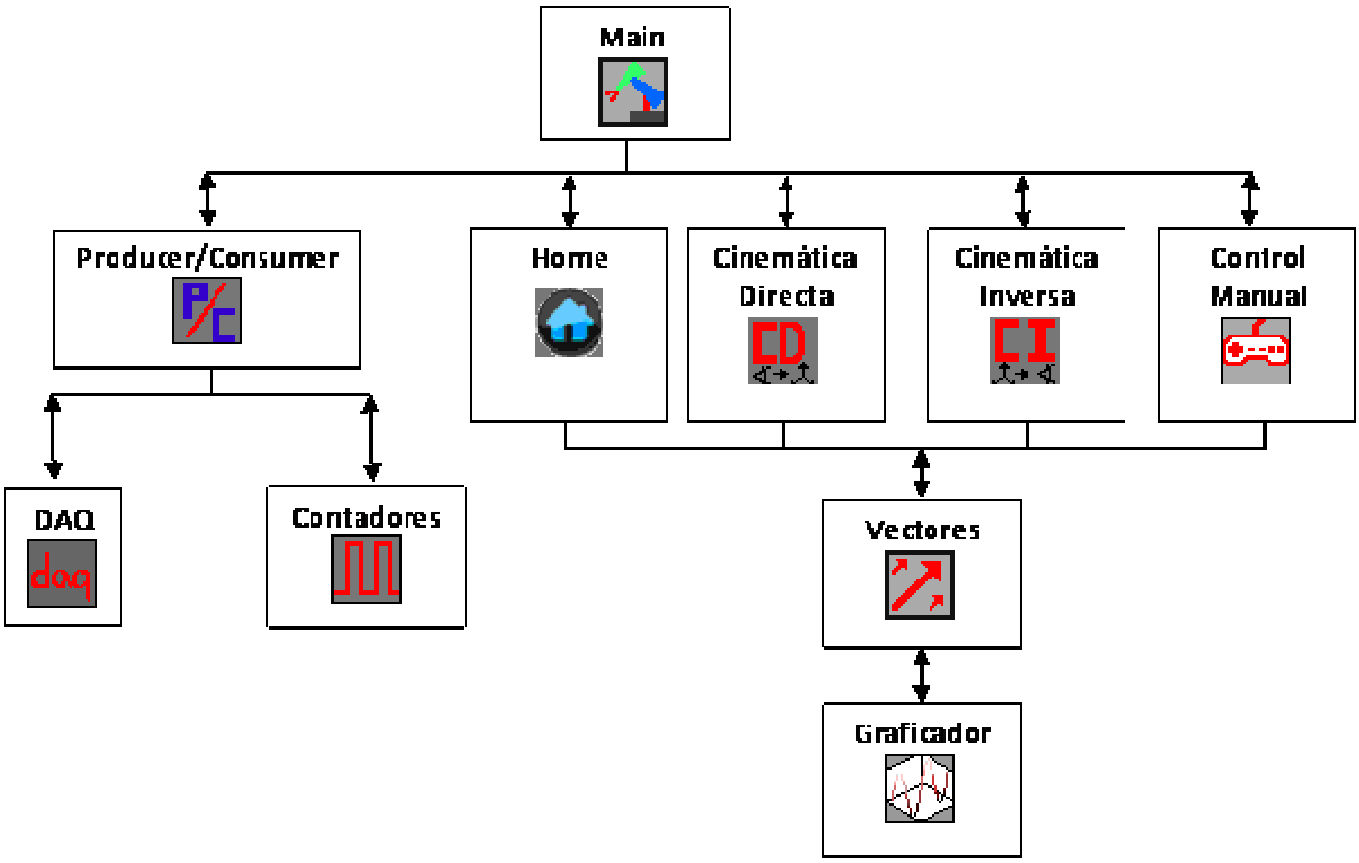


Figura 5.8. Formula Node en LabVIEW. (National Instruments, 1998)

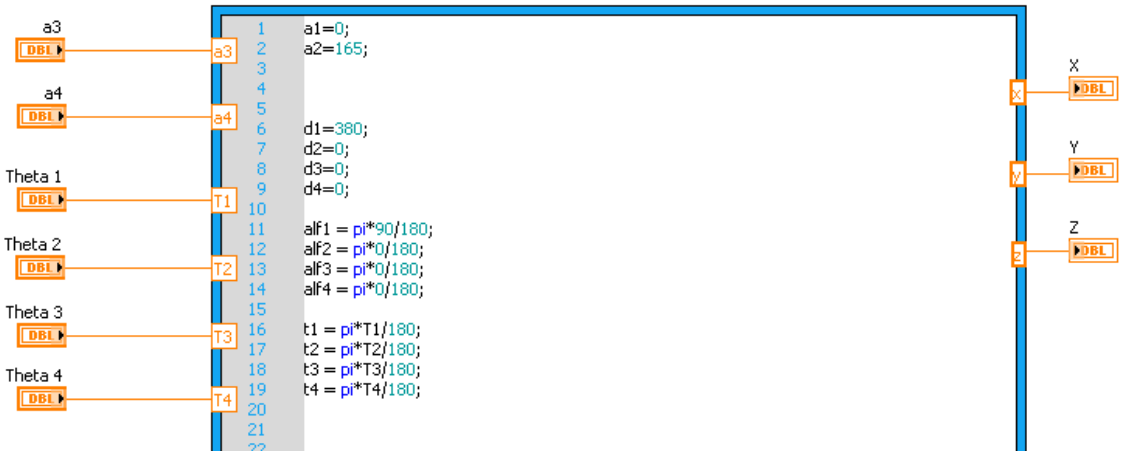
Para el uso del LabVIEW en la operación del robot, y con base en los fundamentos teóricos y las necesidades de lectura de encoder se siguió el siguiente esquema de programación e interconexión de subVI:



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

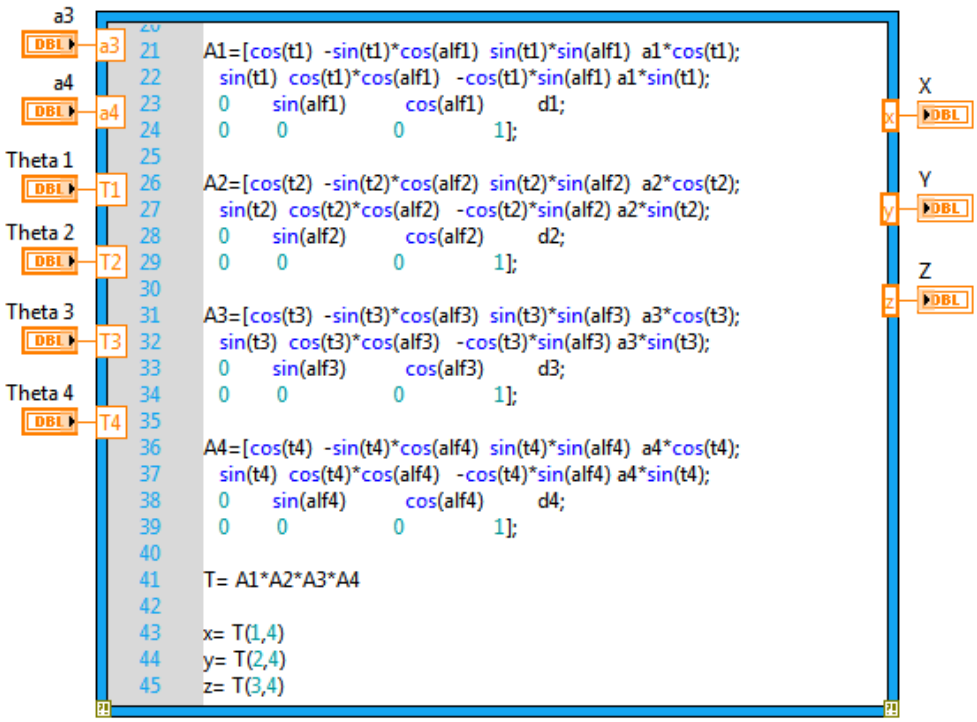


Las constantes no sólo serán las longitudes de las articulaciones (*d [mm]*), también están declarados los ángulos α en radianes.

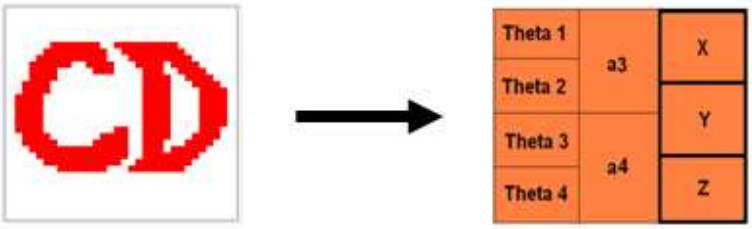
Se puede notar que los ángulos *a3* y *a4* están asignados a 2 entradas, por lo que se podría pensar que son variables, lo cual no es así, ya que las entradas a3 y a4 sólo reciben o el valor de 0, o el de la longitud de las 2 últimas articulaciones (la razón de esta forma de declaración se explica en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



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

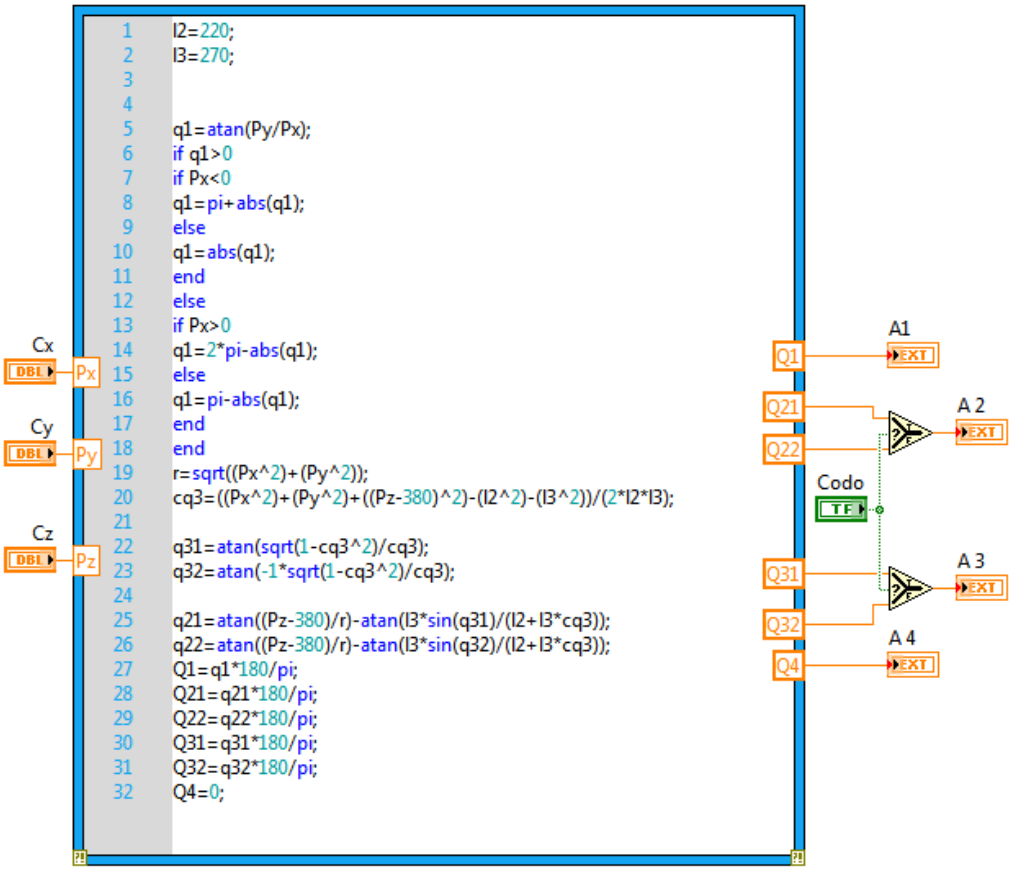


5.2.3 SUBVI CINEMÁTICA INVERSA

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.

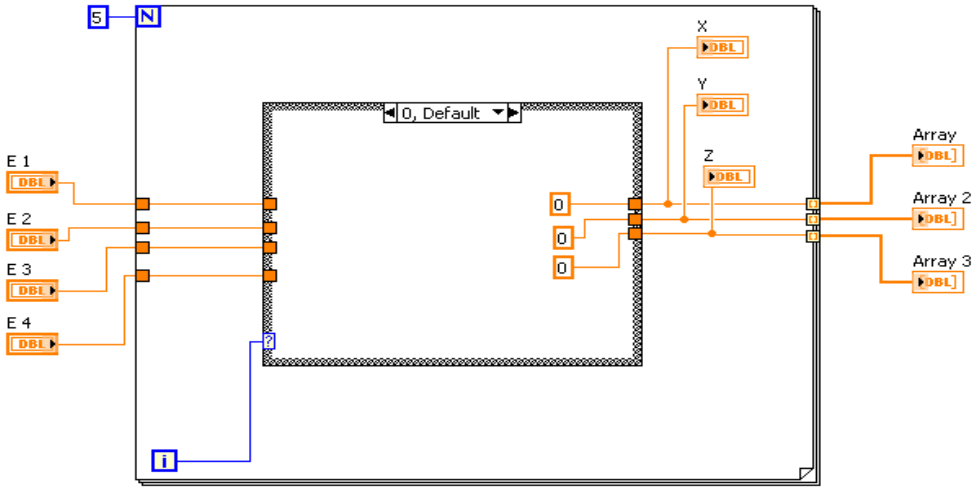
Finalmente se asignan los conectores para poder usarlo desde el Main.

5.2.4 SUBVI VECTORES

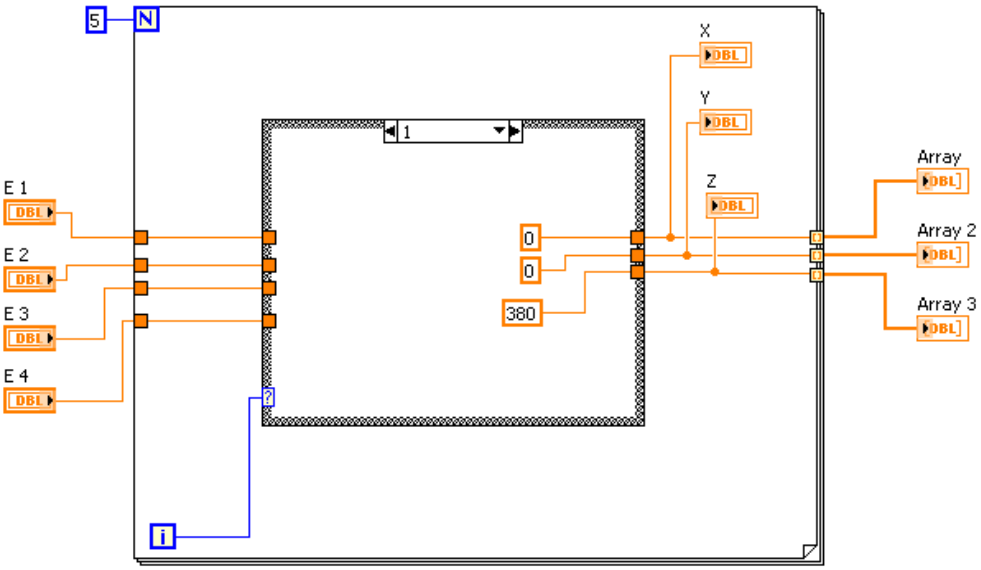
Ya que se tienen los SubVI básicos de control, podrán ser utilizados para crear una imagen 3D del brazo modelado. Para ello es necesario crear un nuevo SubVI que realice un ciclo que construya los 4 vectores a partir de la Cinemática Directa. Estos vectores corresponderán a cada eslabón del brazo.

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).

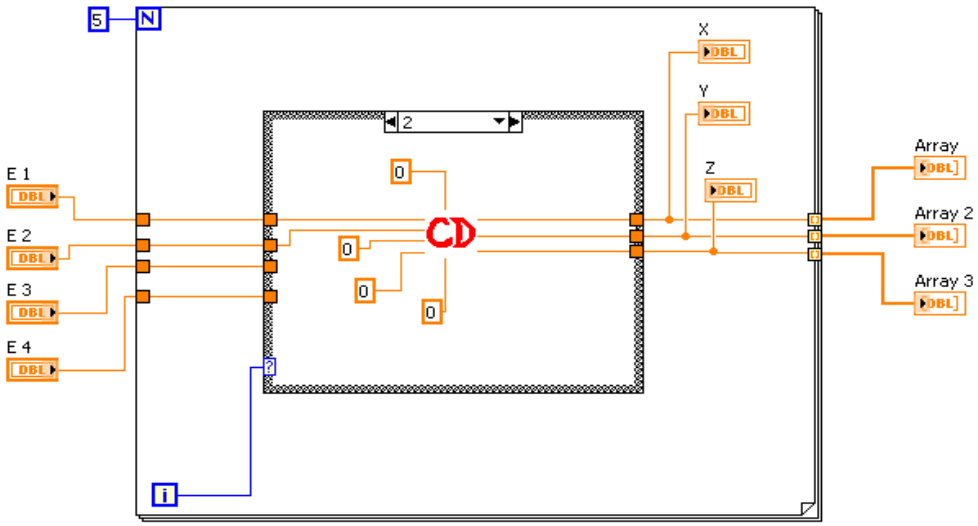


Para obtener la posición del codo es necesario llamar al SubVI de la Cinemática Directa. Si se analiza la CD se puede observar que Theta 1 y Theta 2 son las variables necesarias para obtener la posición del codo y, por lo tanto, serán conectadas directamente a E1 y E2; Theta 3 y Theta 4 no influyen en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

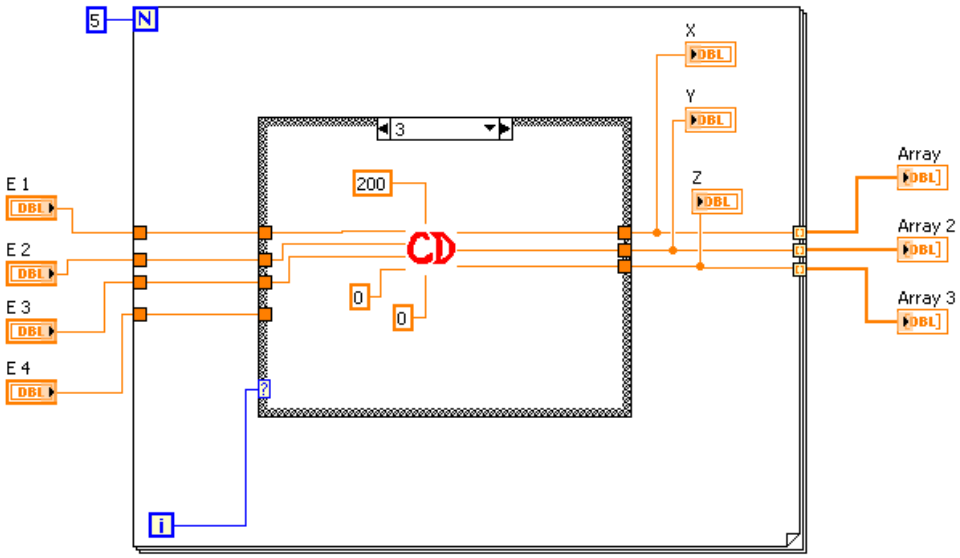
Nótese que en la CD se dejaron Alf3 y Alf4 conectadas a entradas, que se tendrán que asignar para este caso y ahora se puede entender el porqué de esta situación. La matriz de Transformación Homogénea (T) que se obtiene por la

multiplicación de las 4 matrices de rotación entrega la posición del efector final, pero al hacer Alf3 y Alf4 igual 0, puede considerarse que las matrices ${}^{2}A_{3}$ y ${}^{3}A_{4}$ serán matrices identidad y las coordenadas que la matriz T devolverá son las de la posición del codo.

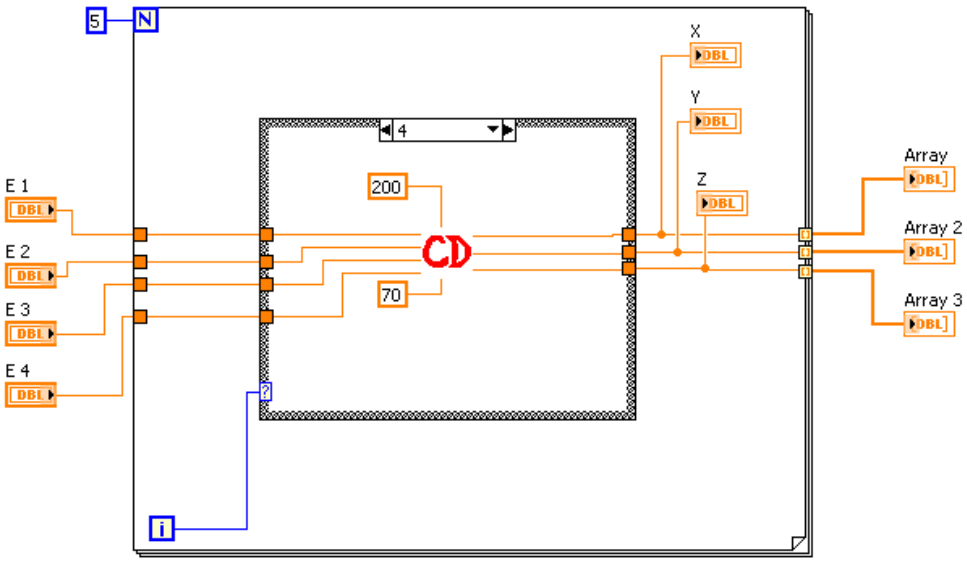
De esta manera se obtiene el valor de la tercera casilla de los arreglos



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, Alf3 toma el valor de la longitud del eslabón correspondiente y Alf4 y Theta 4 permanecen siendo 0. El resultado será almacenado en la cuarta casilla de los arreglos.

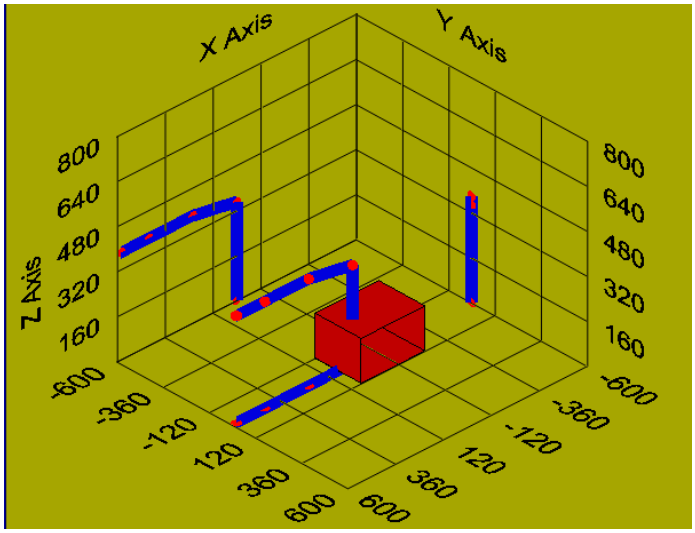


Con la última iteración se obtiene la posición del efector final, por lo cual los 4 valores de los controles E1, E2, E3, E4 serán conectados a su correspondiente ángulo Theta y Alf3 y Alf4 recibirán el valor de la longitud de su eslabón respectivo.

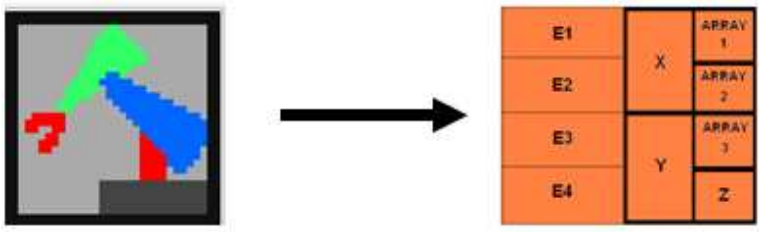


Ya que se tienen los 3 arreglos llenos y se puede ver que cada uno corresponde a un eje cartesiano (X, Y, Z), se observa que para cada conjunto de ángulos dados a las 4 articulaciones se obtienen los 3 arreglos que posicionan al robot.

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:

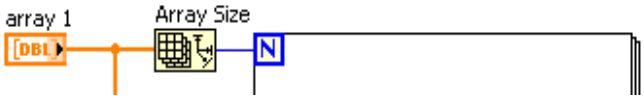


5.2.5 SUBVI TRAYECTORIA

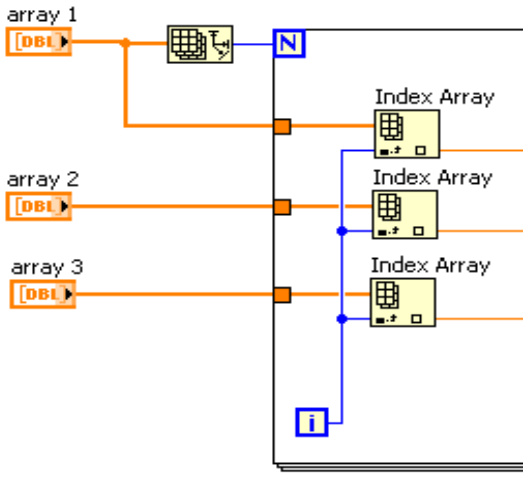
El objetivo principal de la simulación es permitir al usuario elegir los ángulos de las articulaciones o un punto dentro del espacio de trabajo del robot y obtener una imagen 3D del robot modelado basado en vectores. A partir de esto se puede crear un ciclo que permita enviar una serie de puntos que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

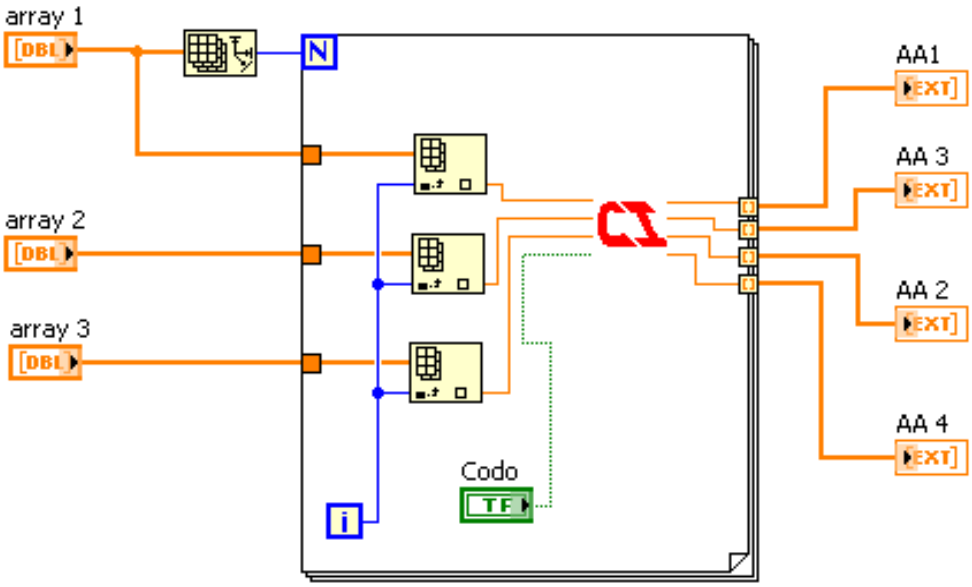
Para poder sacar todos los valores de los arreglos, sin importar de qué tamaño sean éstos se utiliza un ciclo *for* con el mismo número de iteraciones que de elementos tengan los arreglos. Para ello se utiliza el bloque Array Size, el cual recibe el arreglo y regresa el número de elementos que contiene.



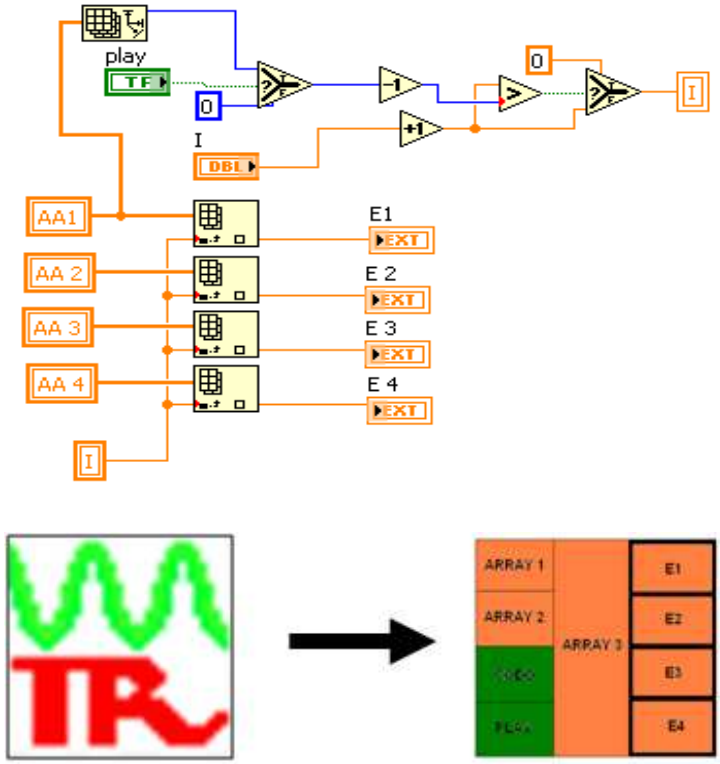
Ahora es necesario que con la primera iteración se obtenga la tercia de elementos correspondiente a la primera coordenada, con la segunda iteración la segunda coordenada y así sucesivamente hasta el final de los arreglos. El bloque Index Array recibe el arreglo y regresa el elemento que se encuentra 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 tercia 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 tercia 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 f*or* se guardan en arreglos.

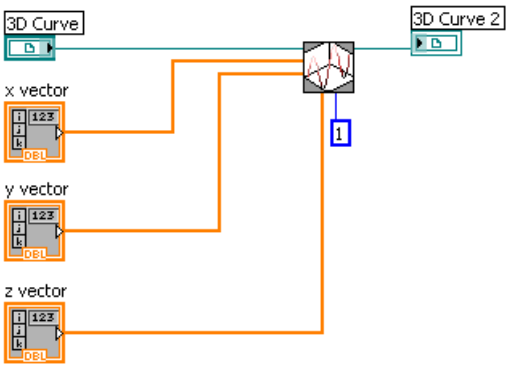


Hasta ahora solamente se ha pasado de arreglos de coordenadas cartesianas a arreglos de ángulos para las articulaciones, y es necesario enviar cada conjunto de ángulos al SubVI Graficador. Se empleará un ciclo similar al *for* pero que incremente una variable auxiliar desde 0 hasta el número de elementos 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.

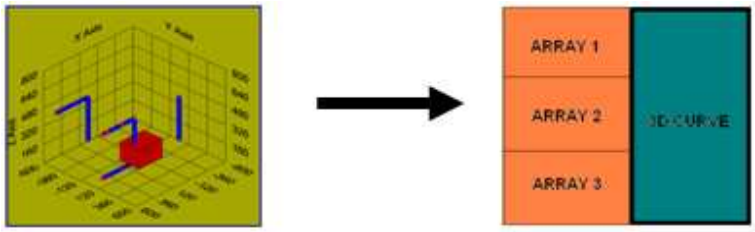


5.2.6 SUBVI GRAFICADOR

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.

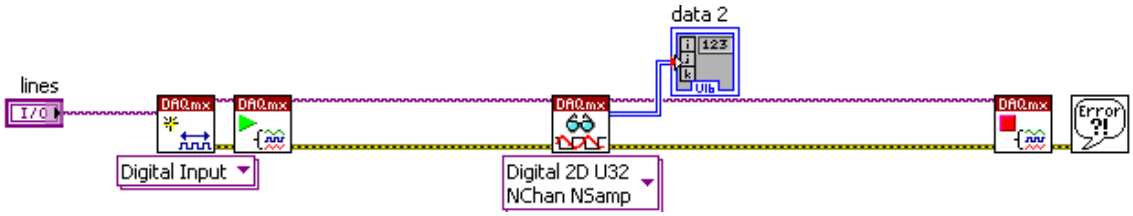


Figura 5.9. Diagrama de bloques para utilizar un puerto digital

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:

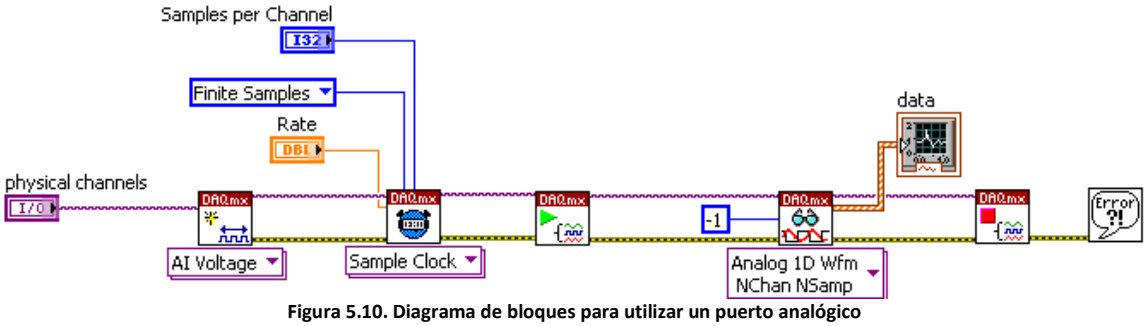


Figura 5.10. Diagrama de bloques para utilizar un puerto analógico

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.

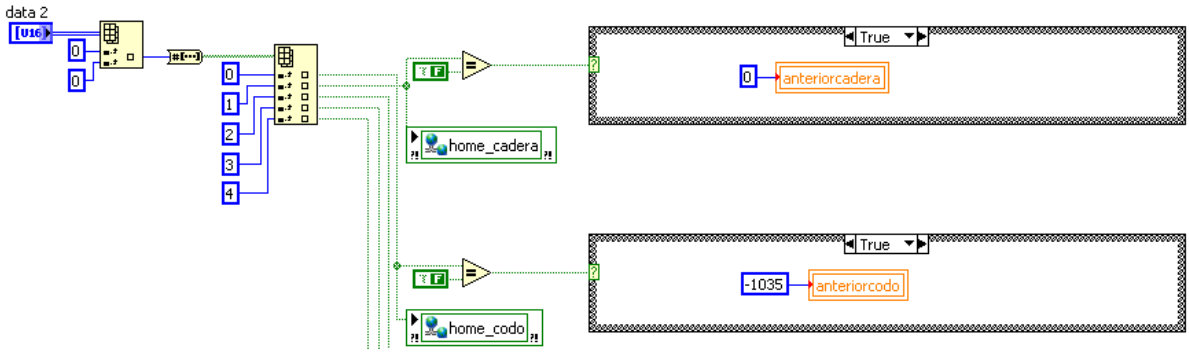


Figura 5.11.Análisis para los microswitches de la cadera y el codo

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.

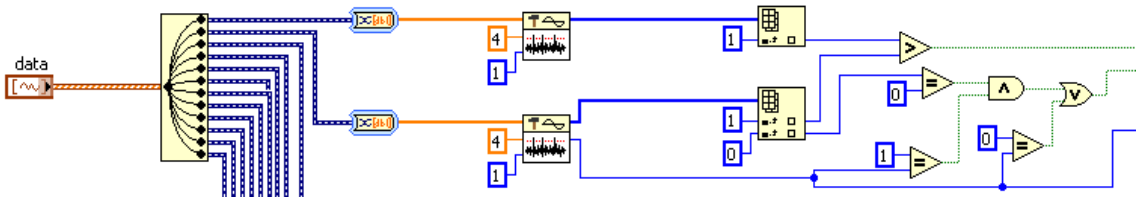


Figura 5.12. Análisis de las señales analógicas

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.

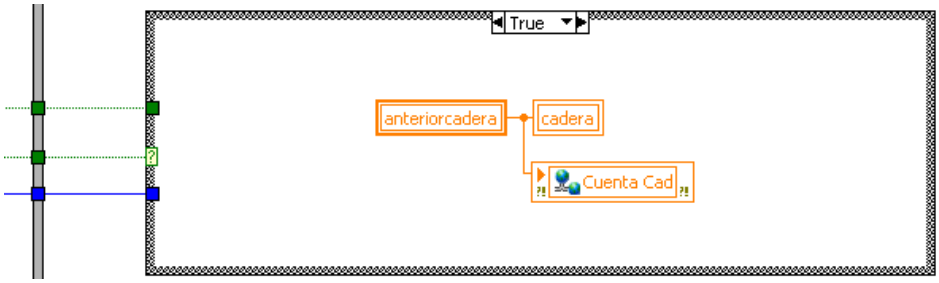
Ya que tenemos por separado las 2 señales de cada encoder, se debe realizar el conteo de pulsos bajo ciertas circunstancias, además de obtener la dirección de giro del motor a partir de las señales.

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.

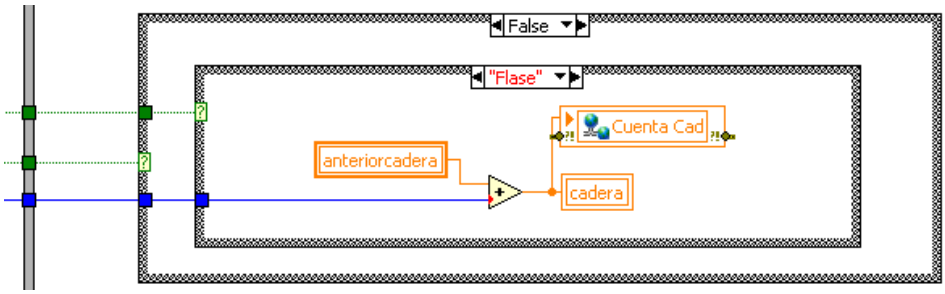
Sólo se llevará la cuenta de una de las dos señales, ya que ambas señales tendrán el mismo número de pulsos, pero es necesario saber en cuál de ambas señales se detectó antes el primer pico válido. Es por ello que se analiza el arreglo que se obtiene del bloque detector de picos, ya que dicho arreglo 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.

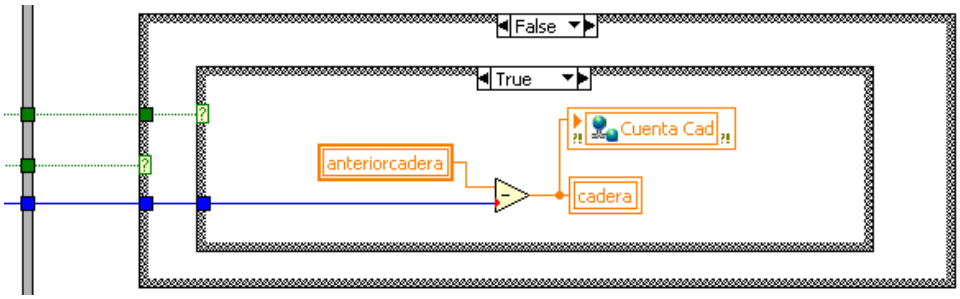
De la Figura 4 se observa que tenemos 3 cables con los resultados del análisis, los cuales se utilizarán para controlar dos estructuras case anidadas. De esta forma se tienen tres casos posibles: El primero será cuando las compuertas lógicas indiquen que no hay pulsos válidos durante el muestreo, dejando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

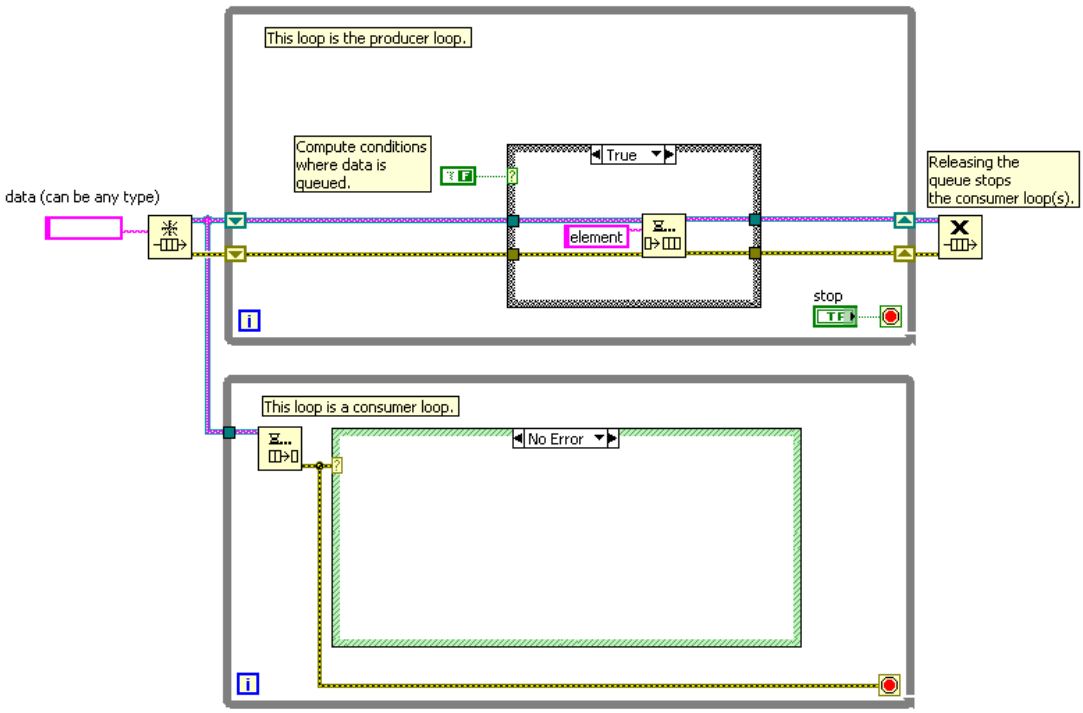


En la tercera opción el sentido de giro es contrario, para lo cual los pulsos válidos del muestreo son restados de la cuenta anterior.



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.

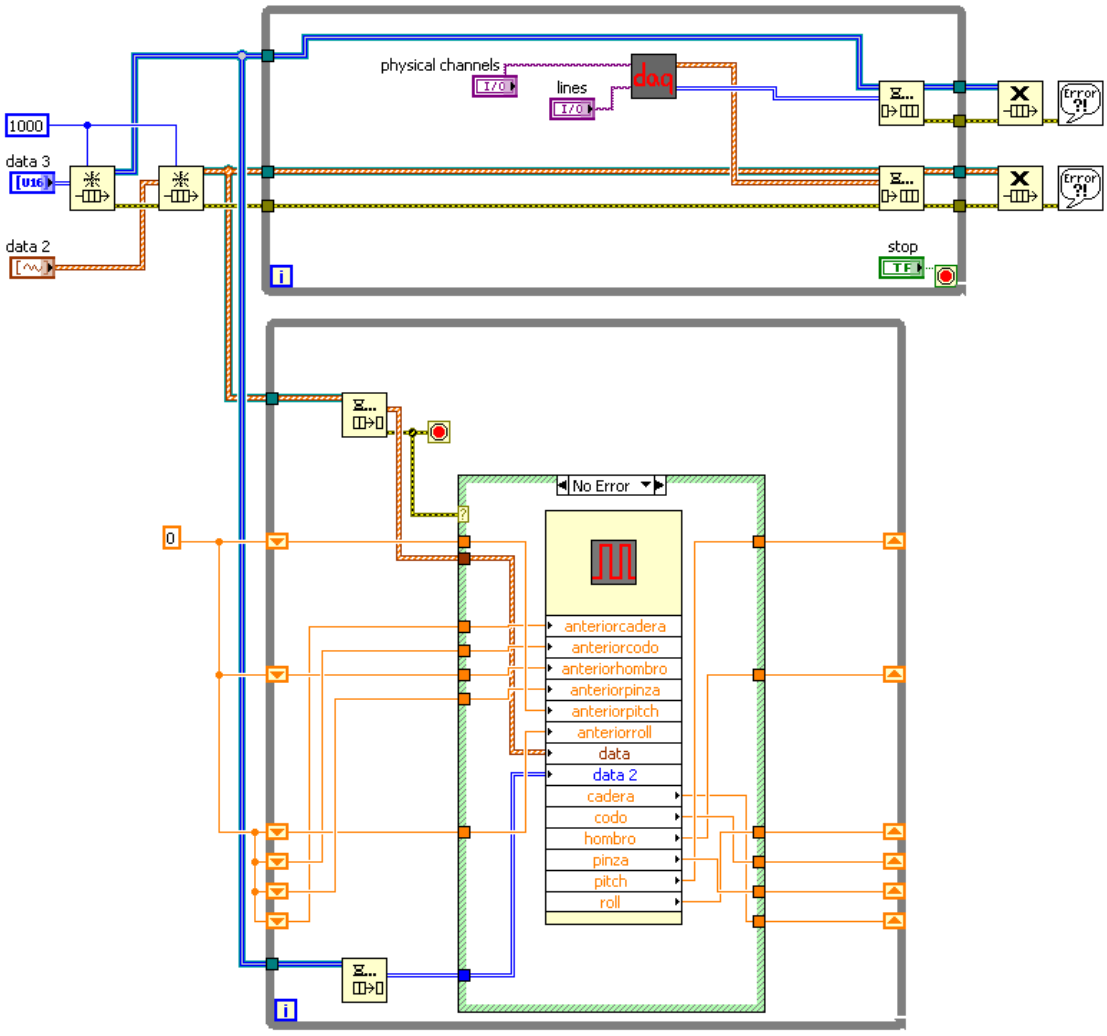


Figura 5.13. Estructura Productor-Consumidor para la lectura de encoder.

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.

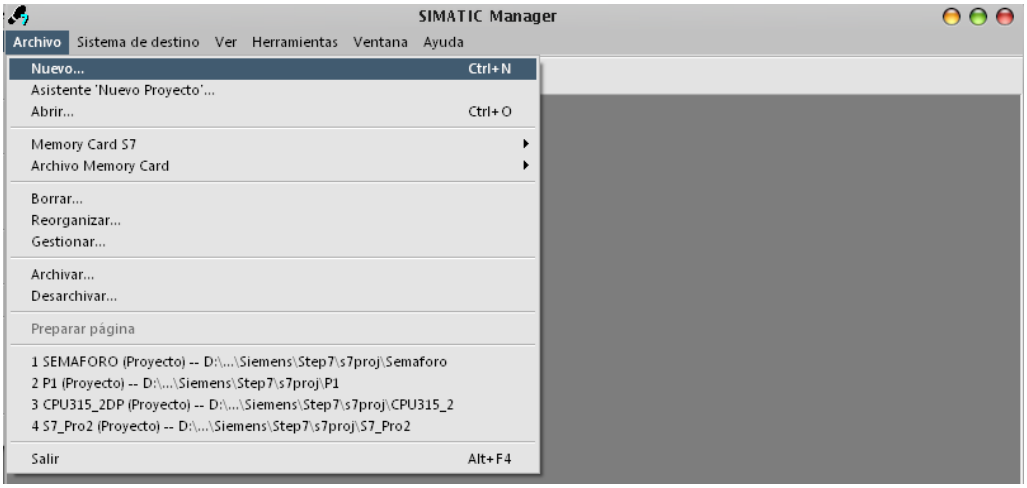
Para poder implementar un programa dentro del PLC es necesario configurar el hardware, definir el protocolo de comunicación entre la PC y el PLC y crear el bloque de programación mediante el cual el PLC encenderá o apagará las salidas de sus módulos. Todo ello está definido en los siguientes pasos.

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

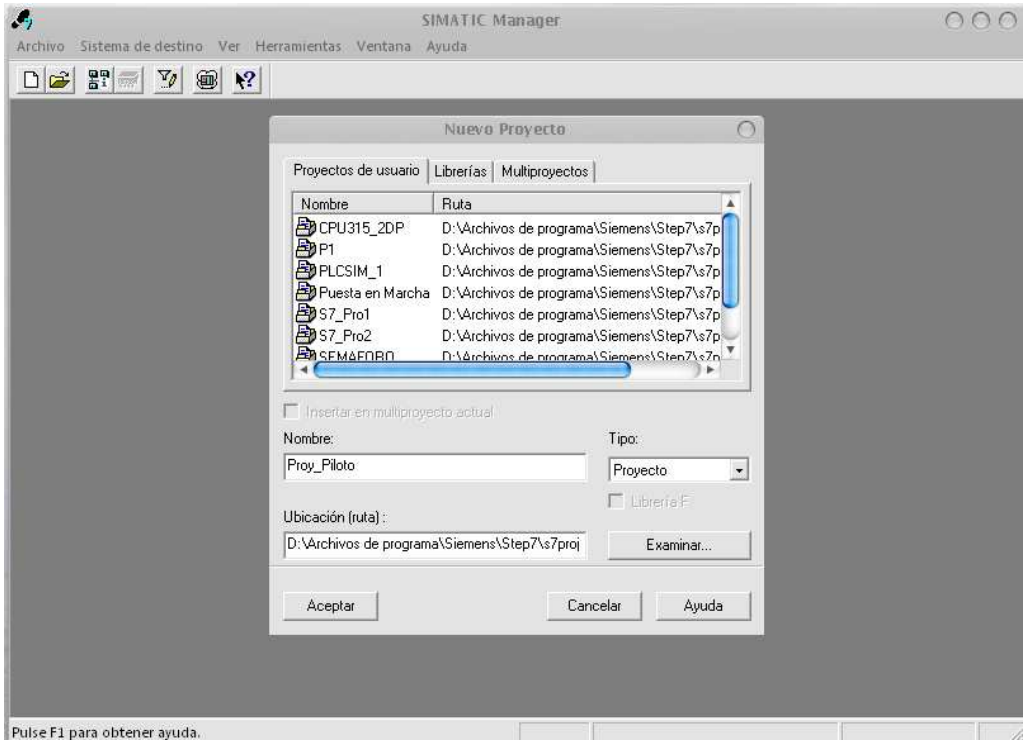
CREACIÓN DE UN PROYECTO

1. Abrir desde el escritorio el software SIMATIC Manager 

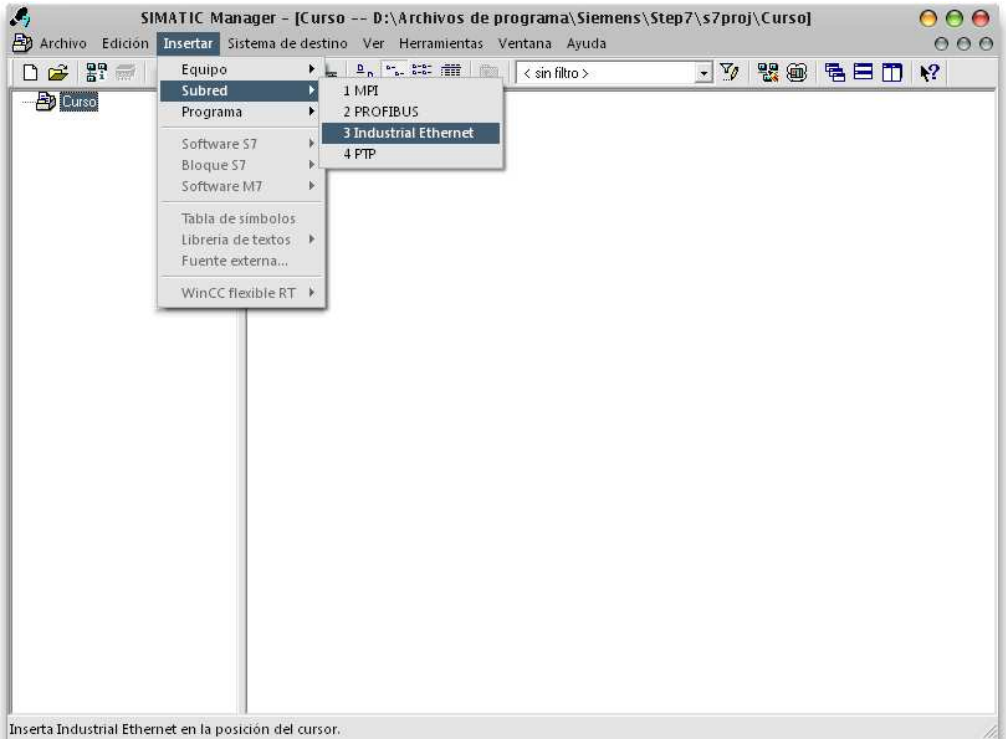
, desde aquí es posible llamar y configurar las herramientas de creación de programas, de configuración de módulos que contenga el PLC S7-300, comunicaciones, simulación, entre otras.
2. Una vez abierto, se crea un nuevo proyecto (Archivo→Nuevo, Ctrl+N)



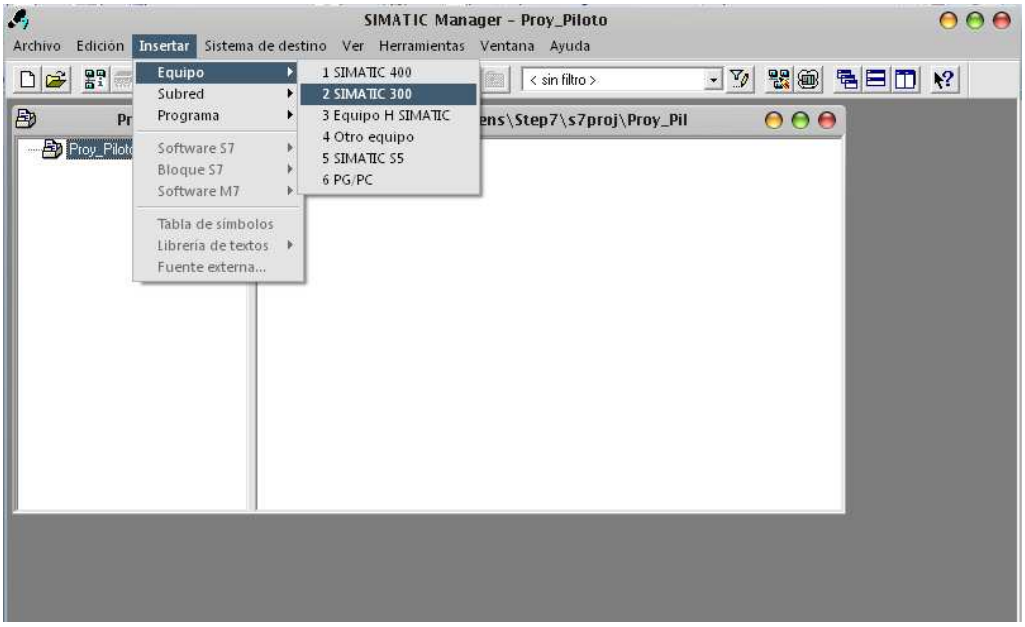
3. Se asigna un nombre de proyecto y se define una ruta o ubicacion diferente para la creación del proyecto.



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

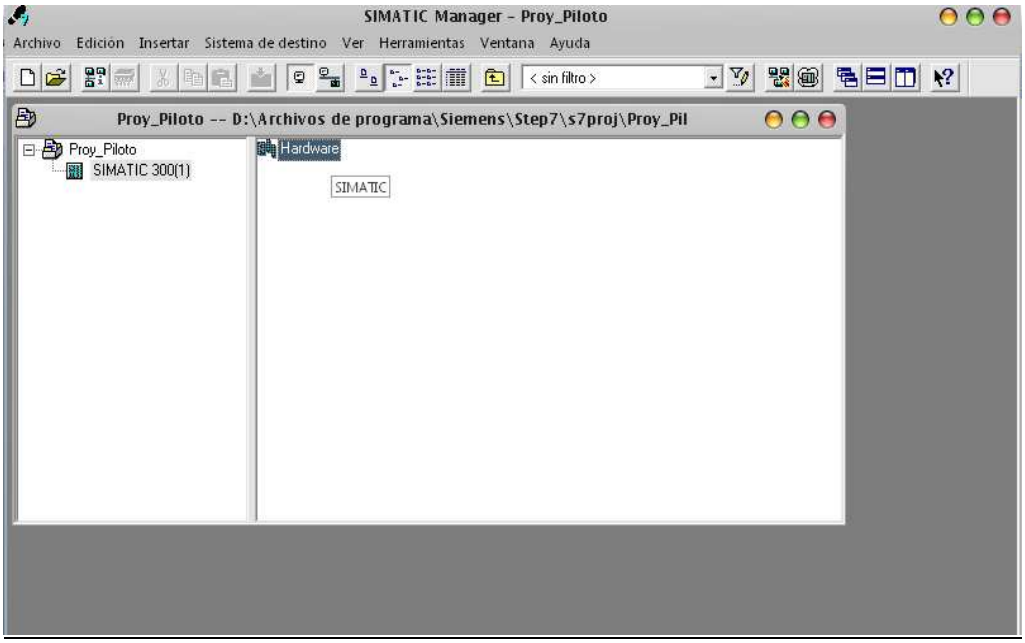


5. Insertar un equipo S7-300 (Insertar→ Equipo→S7-300).

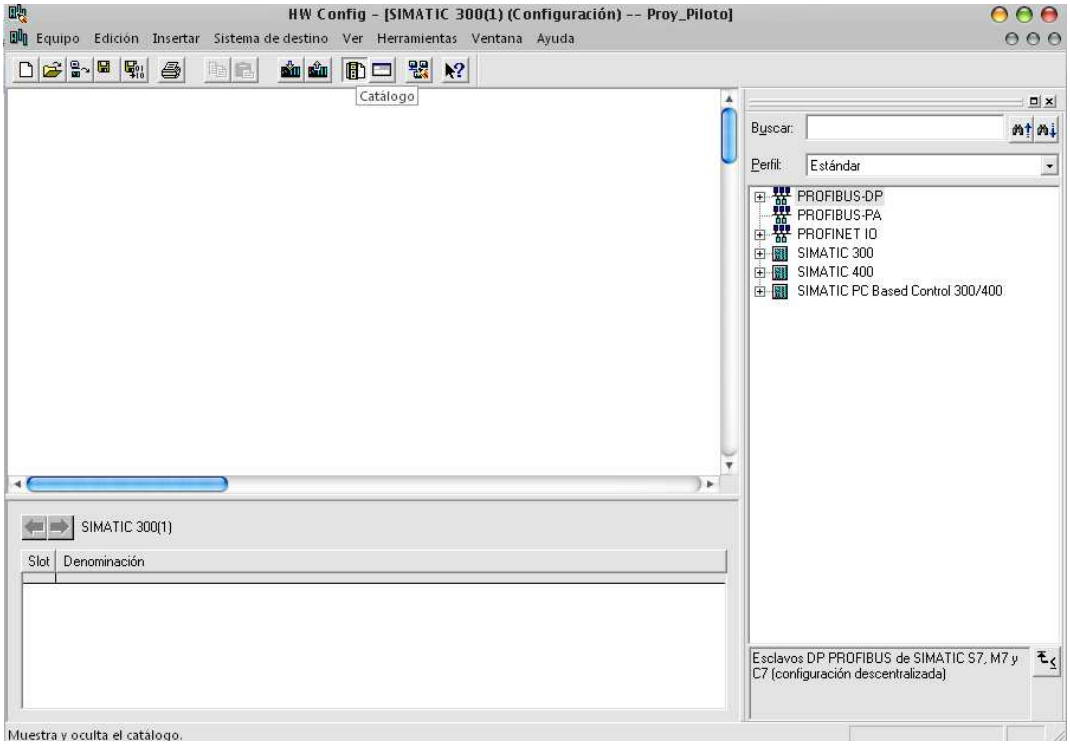


CONFIGURACION DEL HARDWARE

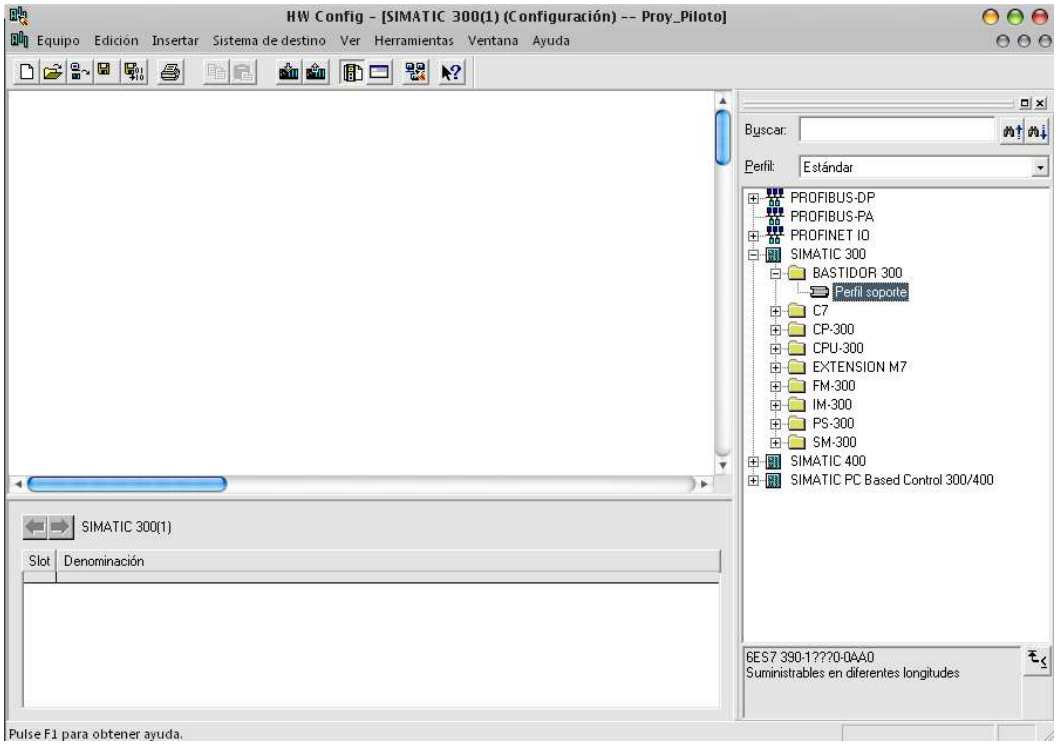
6. Una vez que es creado el PLC se muestra una ventana con la herramienta de configuración del hardware (doble clic en Hardware).



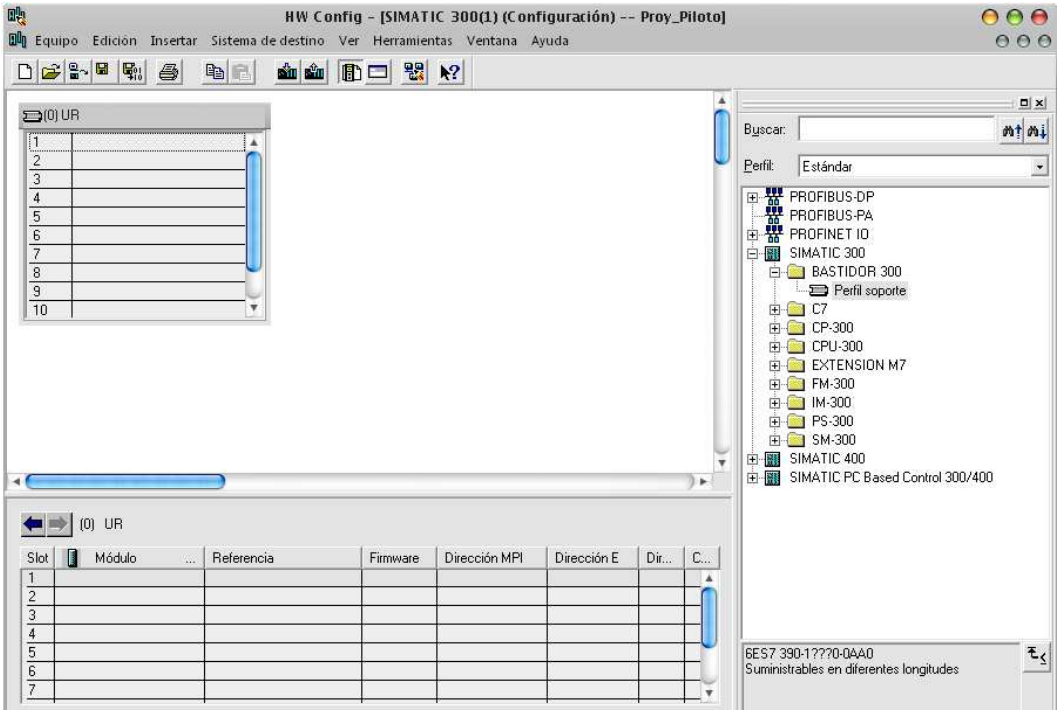
7. En la ventana de configuración del hardware, el icono del catálogo  debe estar activo, con lo cual se despliega la gama de módulos existentes para el S7-300 y S7-400, así como los PROFIBUS.



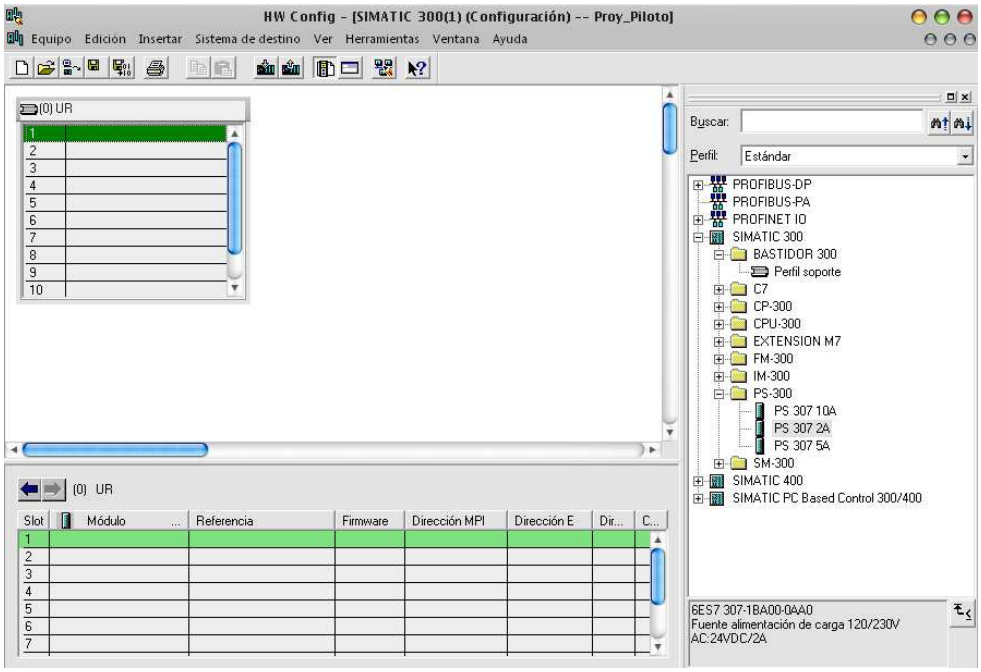
8. Hacer clic en Perfil de Soporte dentro del catálogo (SIMATIC 300→Bastidor 300→Perfil Soporte).



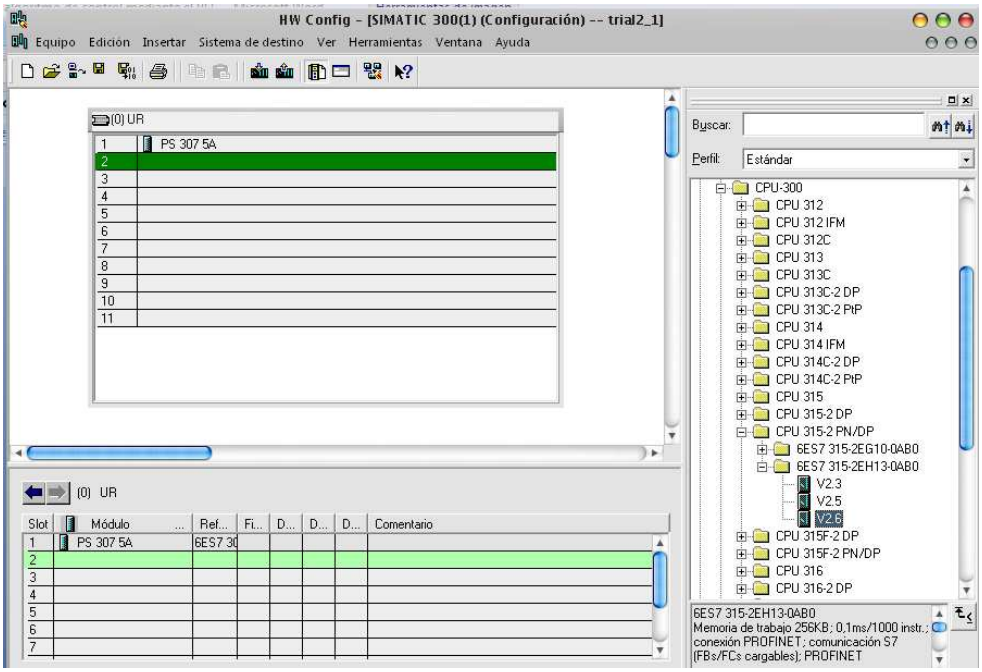
Se muestra una tabla en la cual se agregan los módulos.



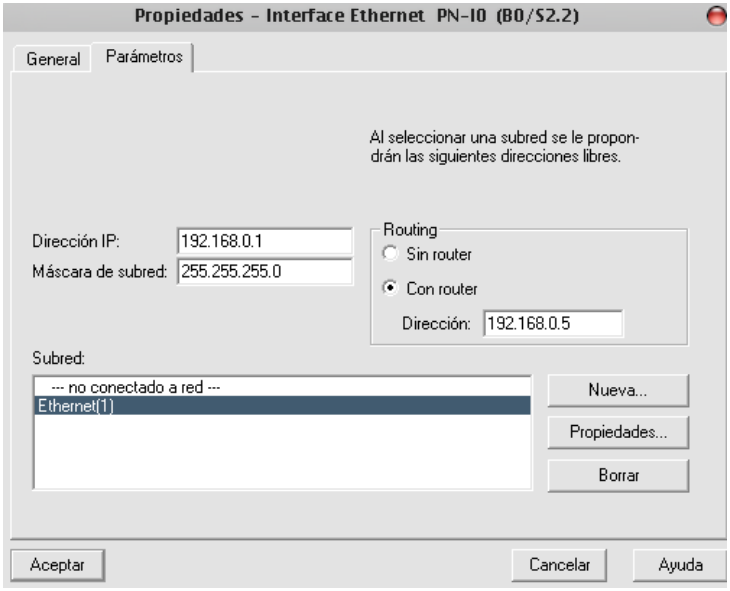
9. Ahora es posible seleccionar y arrastrar hasta el bastidor los módulos, es importante señalar que la primera línea siempre está reservada para la fuente, la segunda para el CPU y la tercera para módulos de comunicación, de este modo a partir de la línea 4 se pueden agregar los módulos de entradas 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.



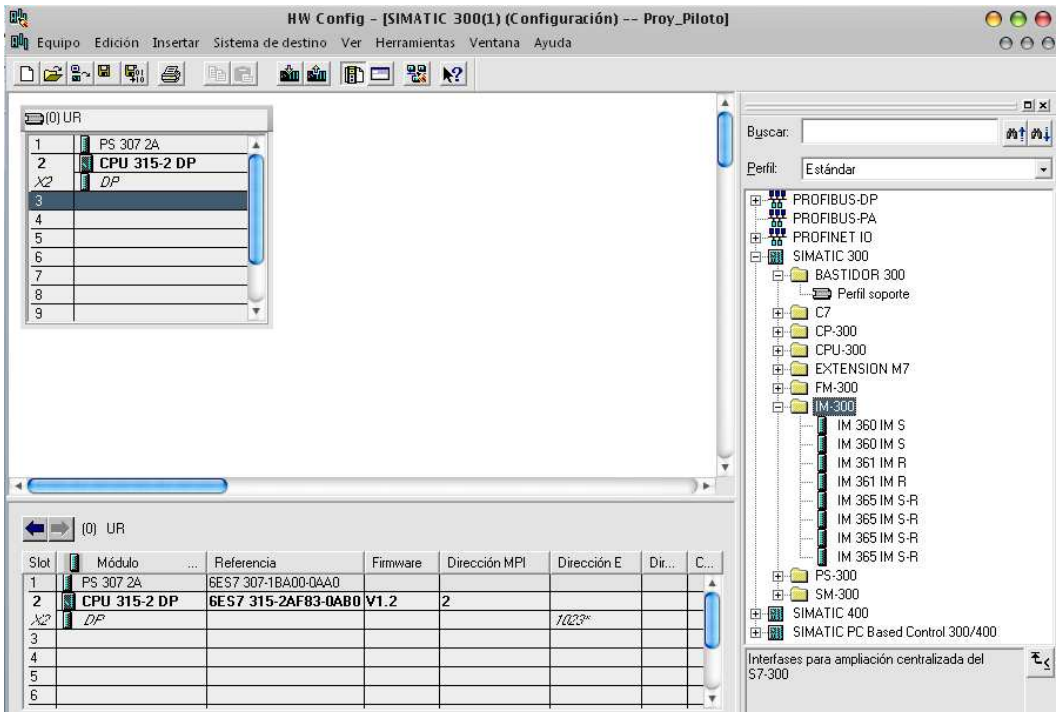
10. Posteriormente se selecciona el CPU en la segunda posición del bastidor, que en este caso será un 315F- PN/DP (SIMATIC 300→CPU 300→CPU 315- 2 DP→6ES7 315-2EH13-0AB0→V2.6)



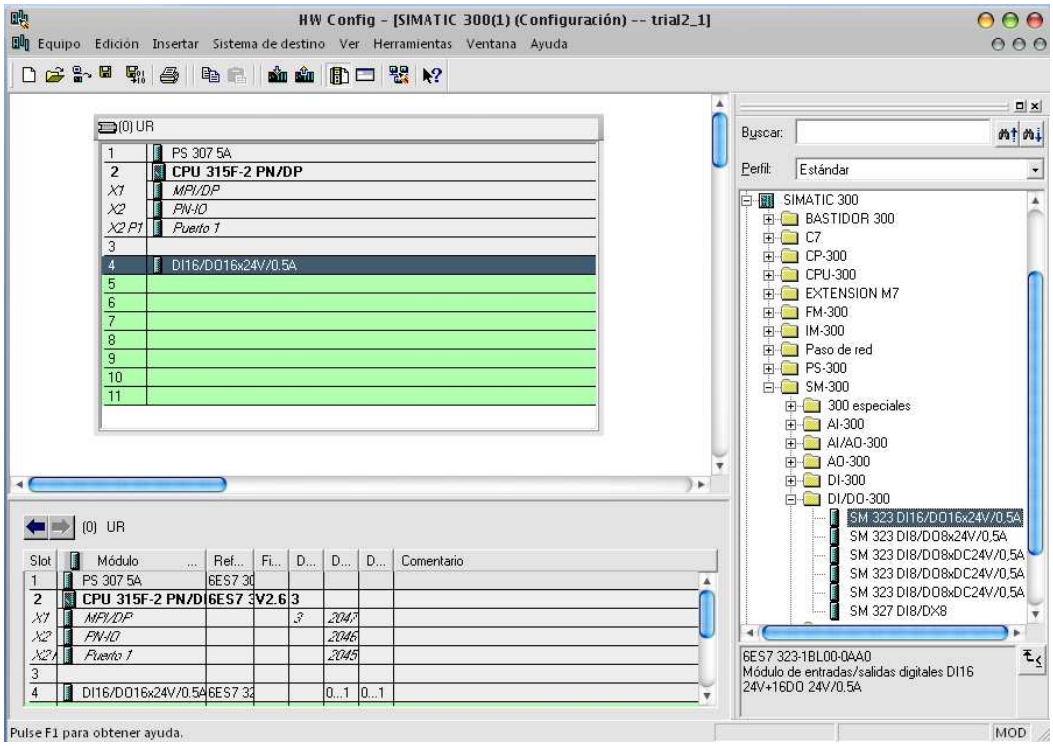
En la nueva ventana se ingresa la información para la configuración del puerto Ethernet, seleccionando la dirección que tiene el PLC dentro de la red, proporcionando además la dirección IP del ruteador que controla las comunicaciones entre los dispositivos, por último seleccionar la red Ethernet.



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.



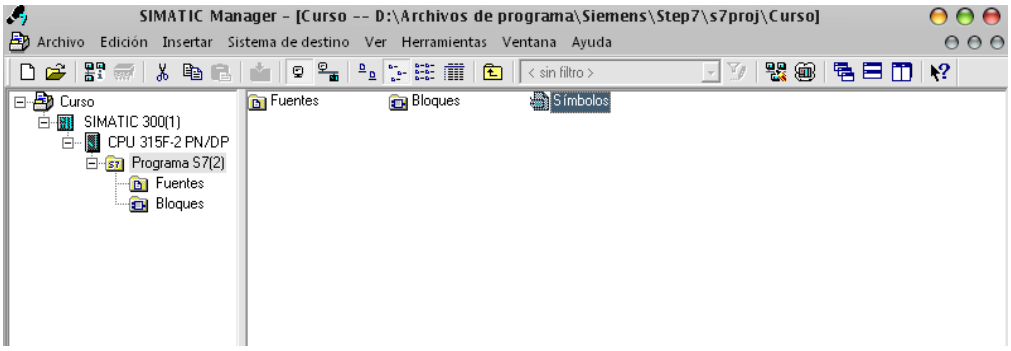
12. Posteriormente se agrega un módulo de entradas y salidas digitales en la cuarta línea del bastidor, en este caso un módulo de 16 entradas digitales y 16 salidas digitales a 24 VDC/0.5 A. (SIMATIC 300→SM300→DI/DO-300→SM323 DI16/DO16xDC24V/0.5 A)



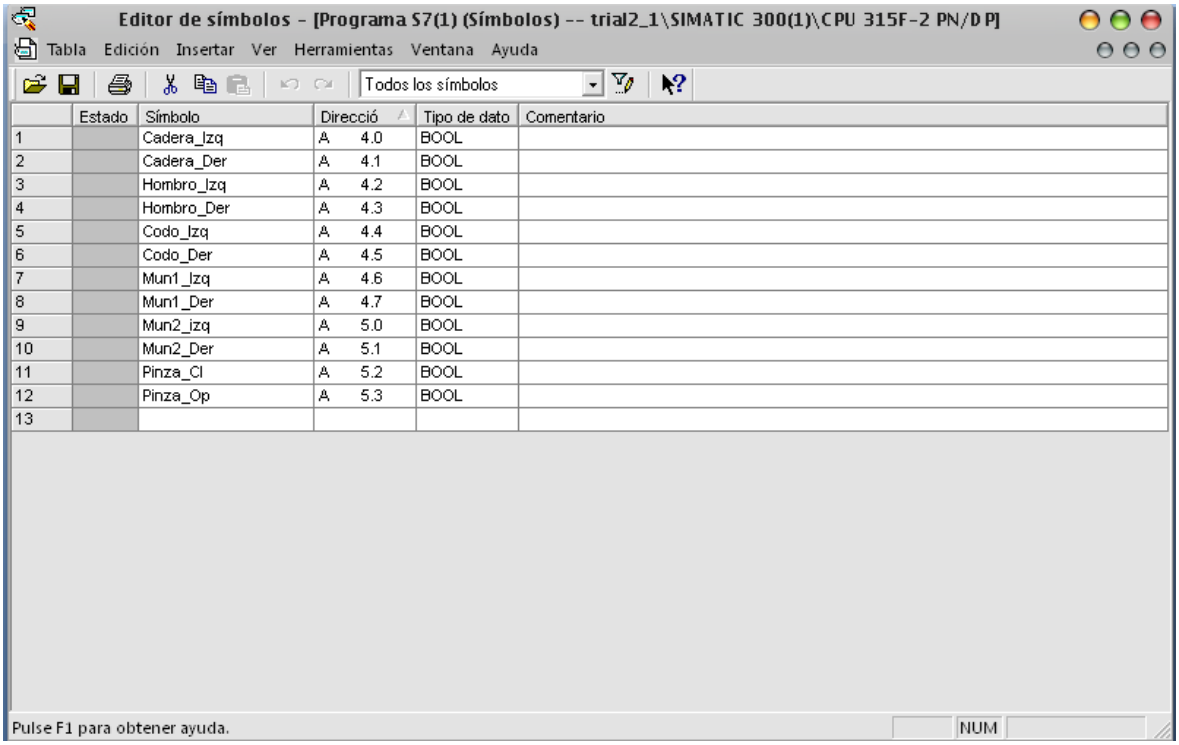
13. Los módulos de entradas y salidas analógicas en caso de ser usados se agregarán de igual manera. (Para entradas analógicas SIMATIC 300→SM300→AI-300. Para salidas analógicas SIMATIC 300→SM300→AO-300). Las propiedades de cada módulo pueden ser modificadas haciendo clic derecho sobre el módulo dentro 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.



	Estado	Símbolo	Dirección	Tipo de dato	Comentario
1		Cadera_Izq	A 4.0	BOOL	
2		Cadera_Der	A 4.1	BOOL	
3		Hombro_Izq	A 4.2	BOOL	
4		Hombro_Der	A 4.3	BOOL	
5		Codo_Izq	A 4.4	BOOL	
6		Codo_Der	A 4.5	BOOL	
7		Mun1_Izq	A 4.6	BOOL	
8		Mun1_Der	A 4.7	BOOL	
9		Mun2_izq	A 5.0	BOOL	
10		Mun2_Der	A 5.1	BOOL	
11		Pinza_Cl	A 5.2	BOOL	
12		Pinza_Op	A 5.3	BOOL	
13					

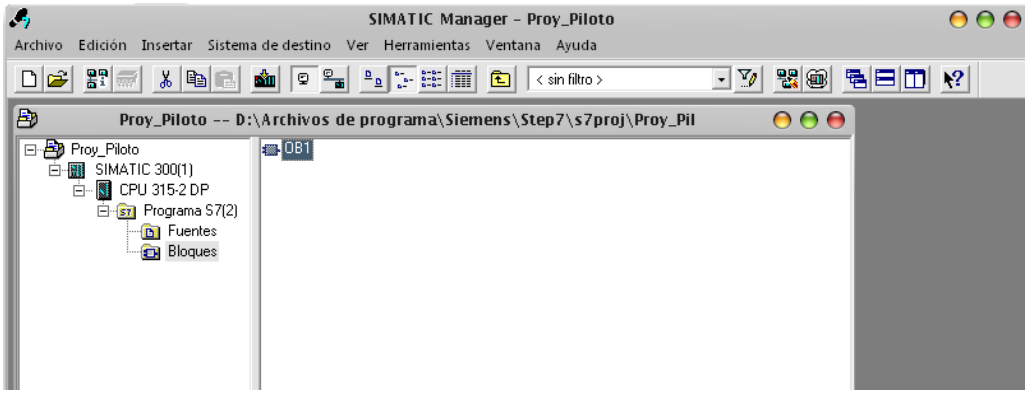
En la **Tabla 5.3** se muestran los tipos de datos que maneja el PLC.

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´

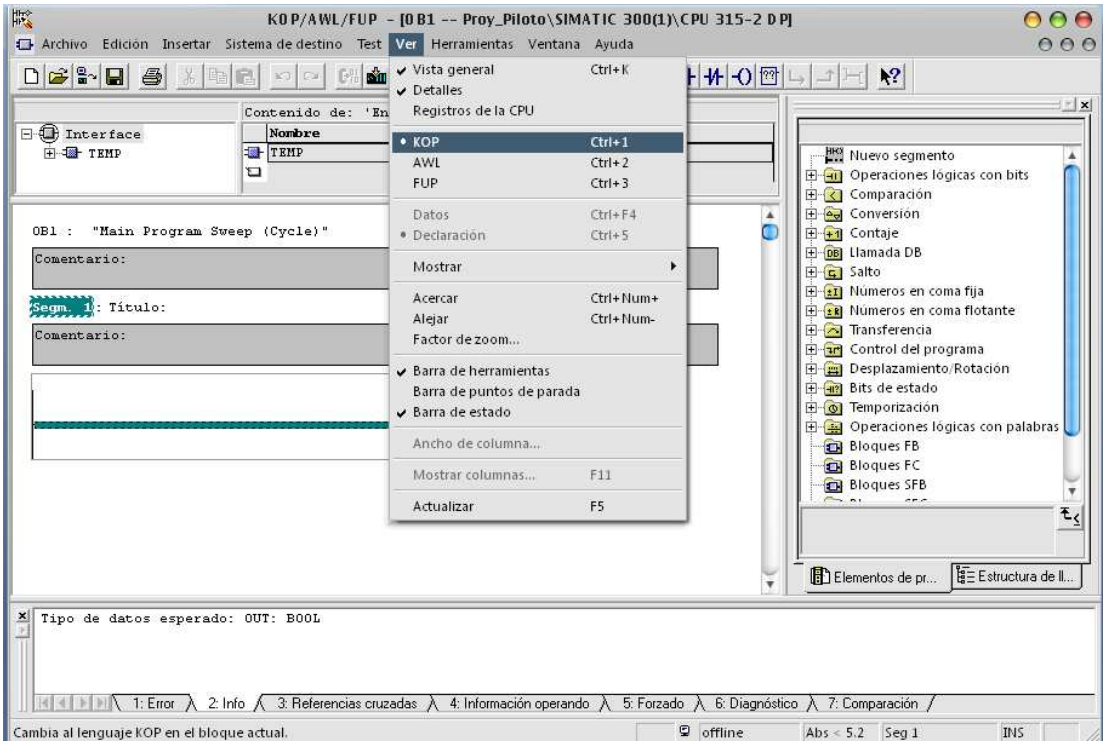
CREACIÓN DE UN PROGRAMA

17. 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).

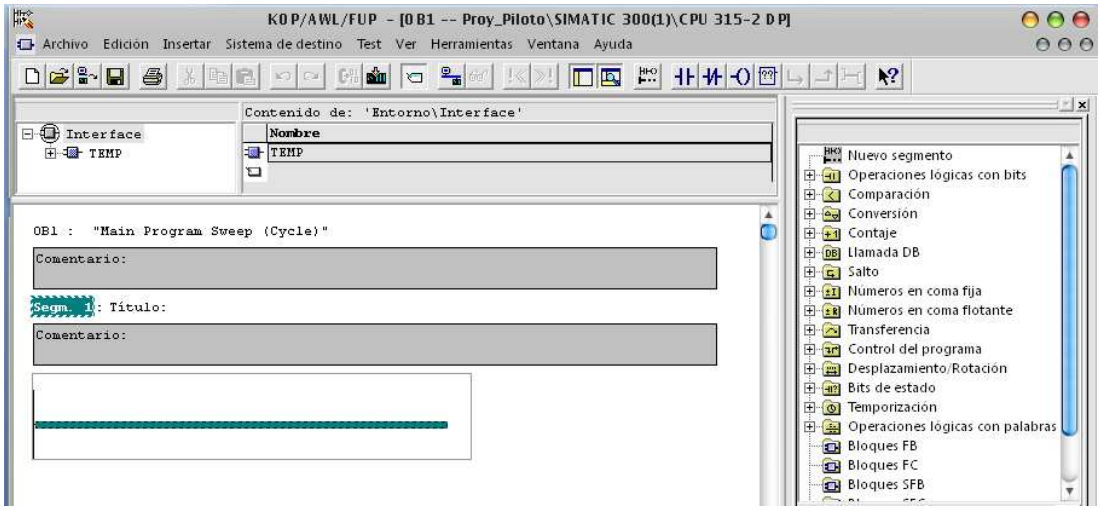


La nueva ventana muestra las propiedades del Bloque, por defecto se deja el bloque tal cual y se da clic en Aceptar, hecho esto se despliega una ventana con el Editor de Programas.

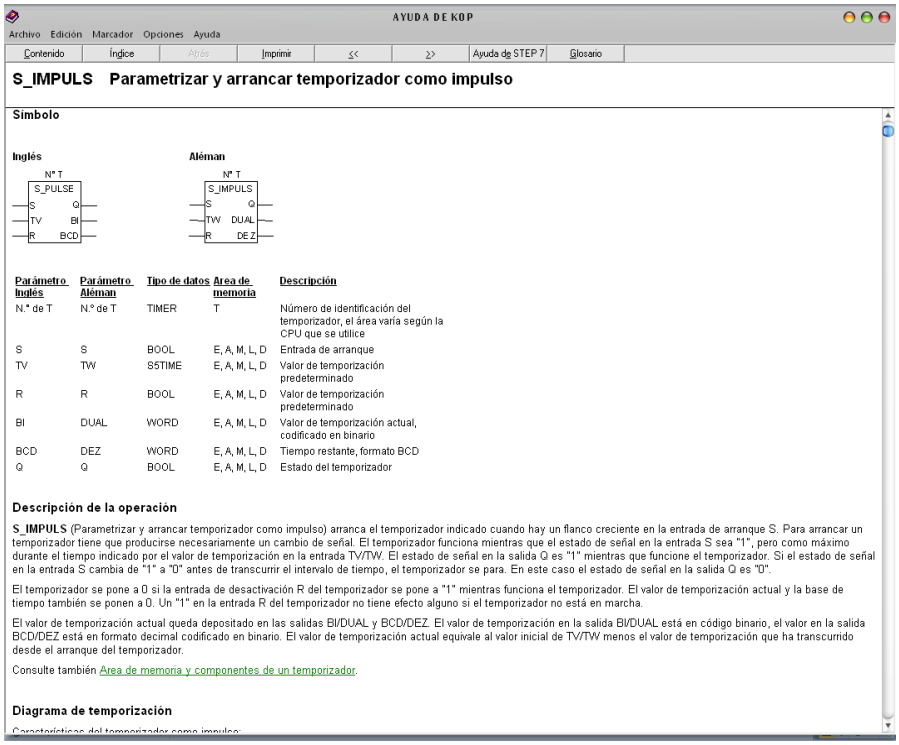
18. Para comenzar a escribir, se debe seleccionar el lenguaje que será utilizado, dentro del Editor de Programas se selecciona ya sea en lenguaje KOP, AWL o FUP dentro del menú "Ver" en este caso es usado el lenguaje KOP (Ver→KOP)



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



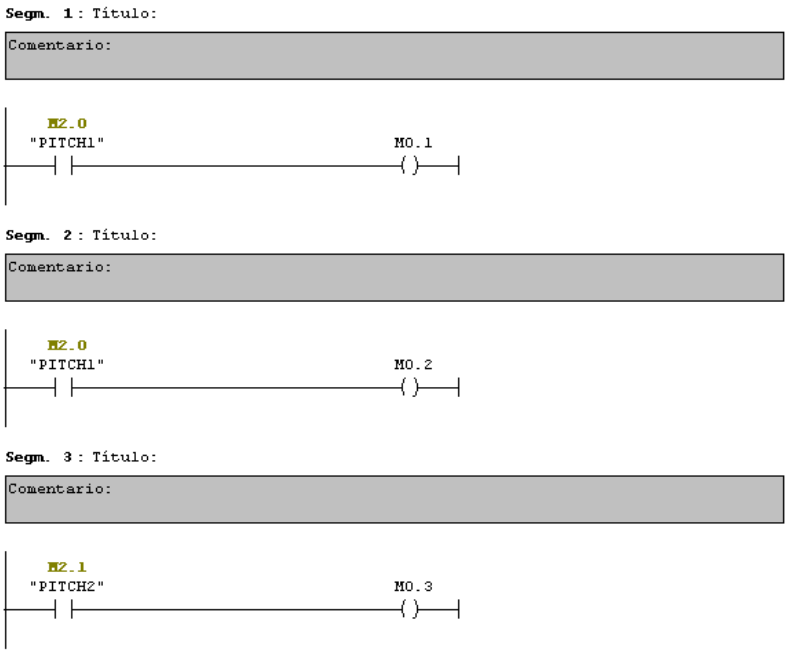
20. Con el icono de ayuda  dando clic sobre las funciones se despliega un menú que explica la simbología, la descripción de la forma de operación, diagrama de las funciones en caso de existir, palabras de estado y un ejemplo de su forma de uso.



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

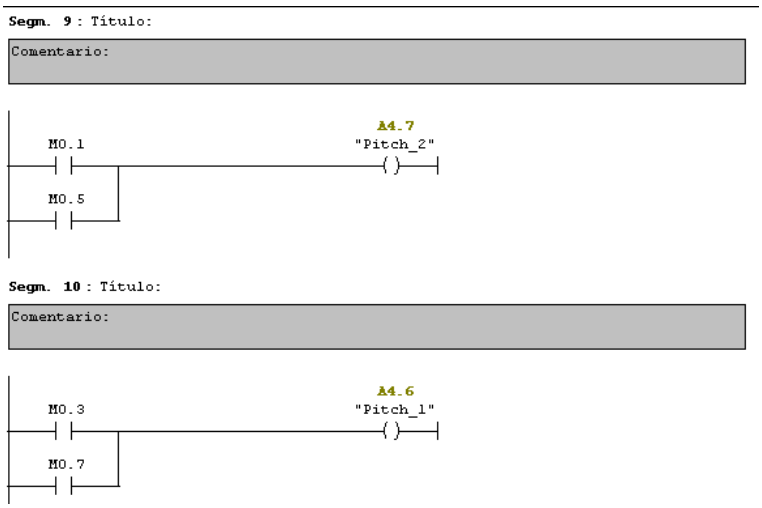
Para el giro del pitch o el roll, solamente es necesario establecer el sentido de giro de los dos motores encargados de estas tareas, esto es si giran hacia el mismo lado se tendrá movimiento de Pitch y si giran en sentido contrario habrá movimiento de Roll.

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

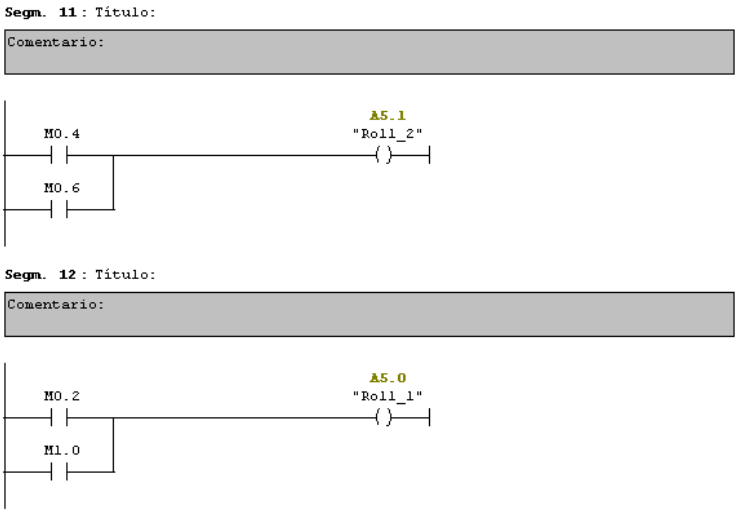


Definiendo Pitch1, Pitch2, Roll1 y Roll2 como las 4 salidas a encender, en diferentes combinaciones, y usando dos marcadores por cada salida como se muestra en la figura, se procede a encender las salidas usando dichos marcadores como se muestra a continuación.

Para el Pitch se tiene que



Mientras que para el Roll.



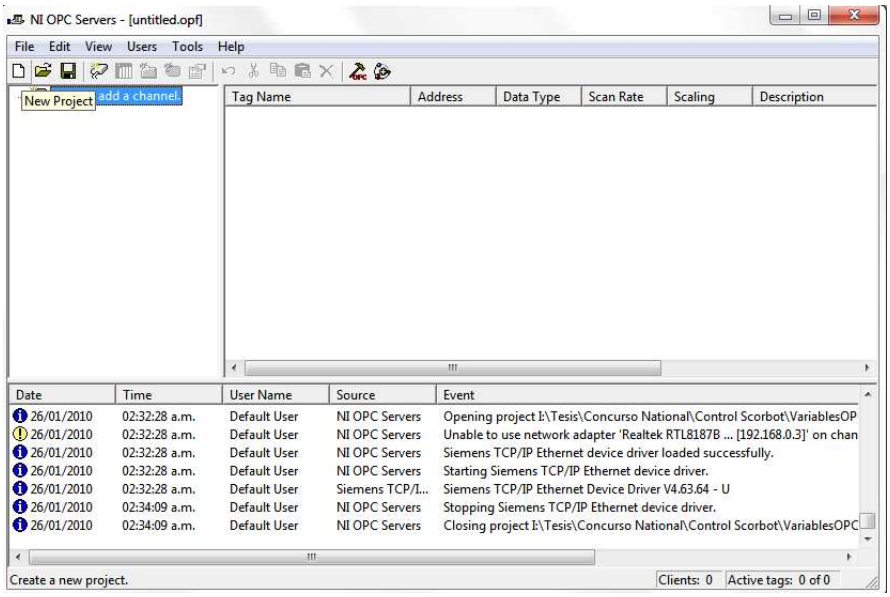
Con esto se logra crear las combinaciones de salidas que darán el movimiento necesario.

5.4 CONECTAR LABVIEW A UN PLC

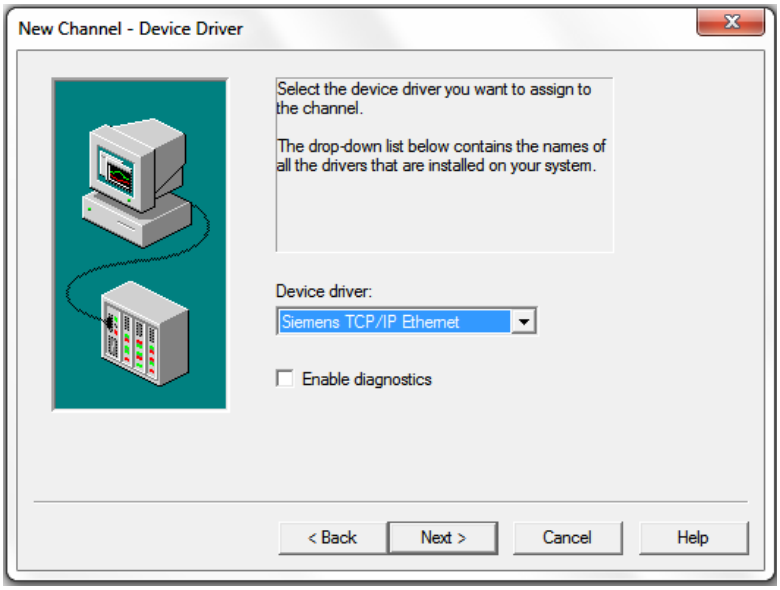
LabVIEW da la posibilidad de programar cualquier PLC en una amplia variedad de formas. OPC (OLE for Process Control) define el estándar para la comunicación de datos en tiempo real, entre los dispositivos de control y las interfaces HMI.

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.

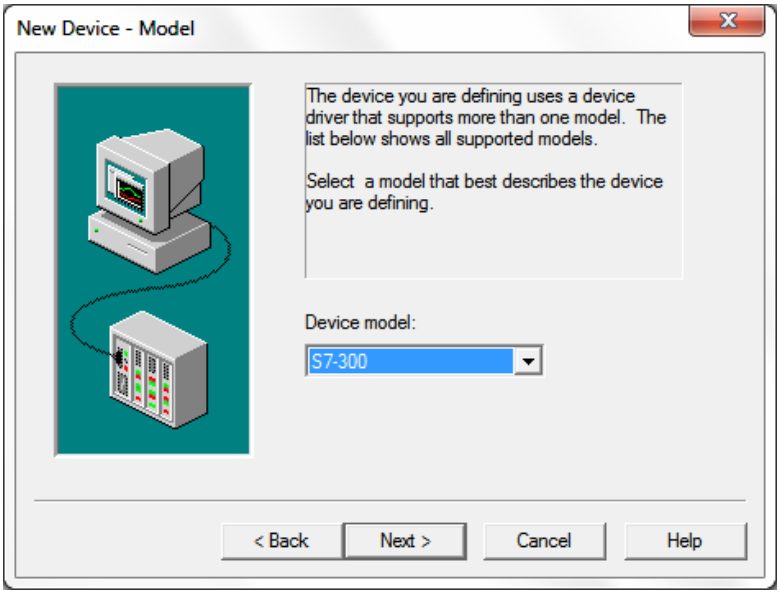


2. Una vez creado, se agrega un canal  , el cual contendrá la información acerca de la marca del PLC asociado, así como la forma de comunicación que se establecerá con el mismo. En este caso será un controlador Siemens TCP/IP Ethernet.

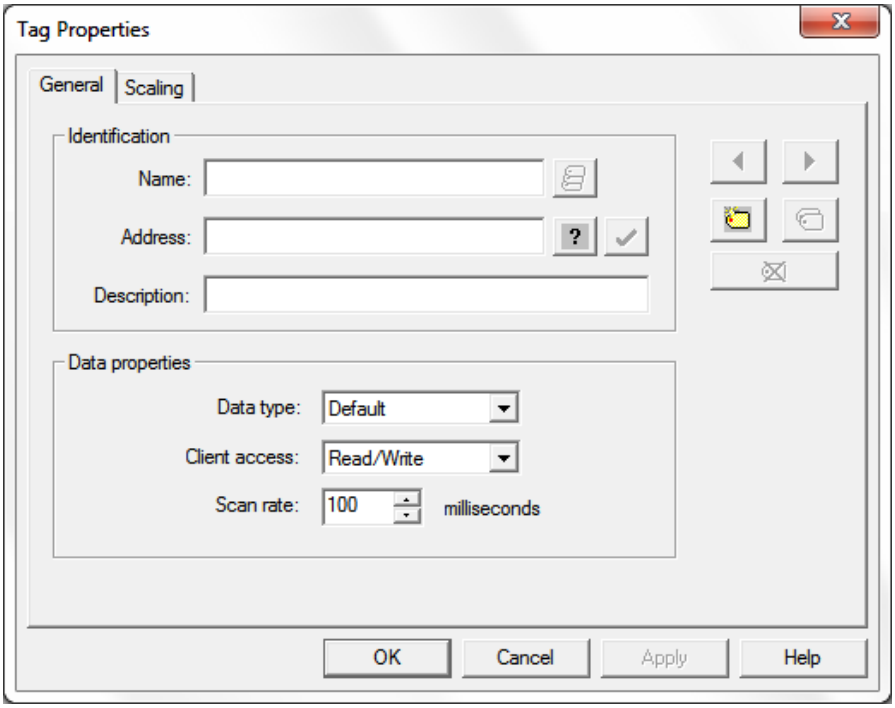


Posteriormente se selecciona el dispositivo presente en la PC por el cual se conecta al PLC, en este caso será una tarjeta de red inalámbrica con una dirección IP establecida. Se establece el tipo de escritura que se hará sobre las variables del PLC y se finaliza el asistente.

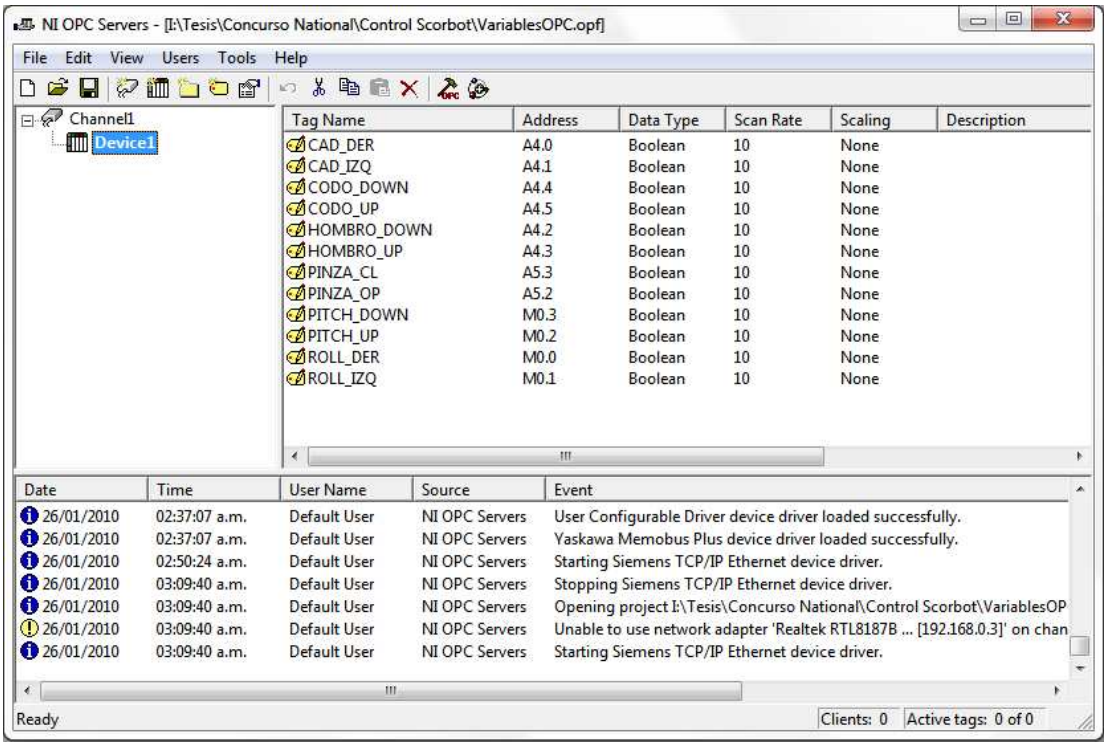
3. Una vez que se tiene el canal, se procede a agregar un dispositivo  , con lo cual se abrirá un nuevo asistente, dentro del cual se selecciona el PLC Siemens usado, en este caso el S7-300, así como su dirección IP (192.168.0.1), las demás configuraciones se dejan predeterminadas.



4. 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.

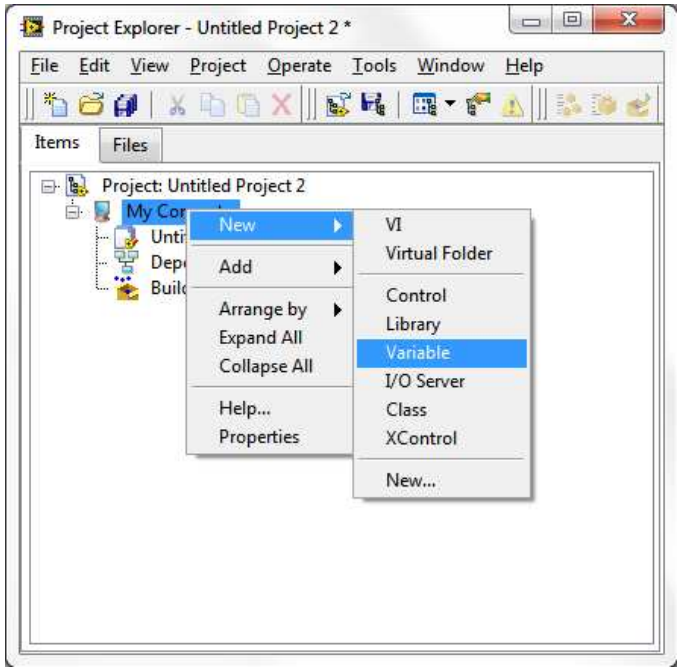


Se hace esto para cada de las variables quedando como se observa en la figura.

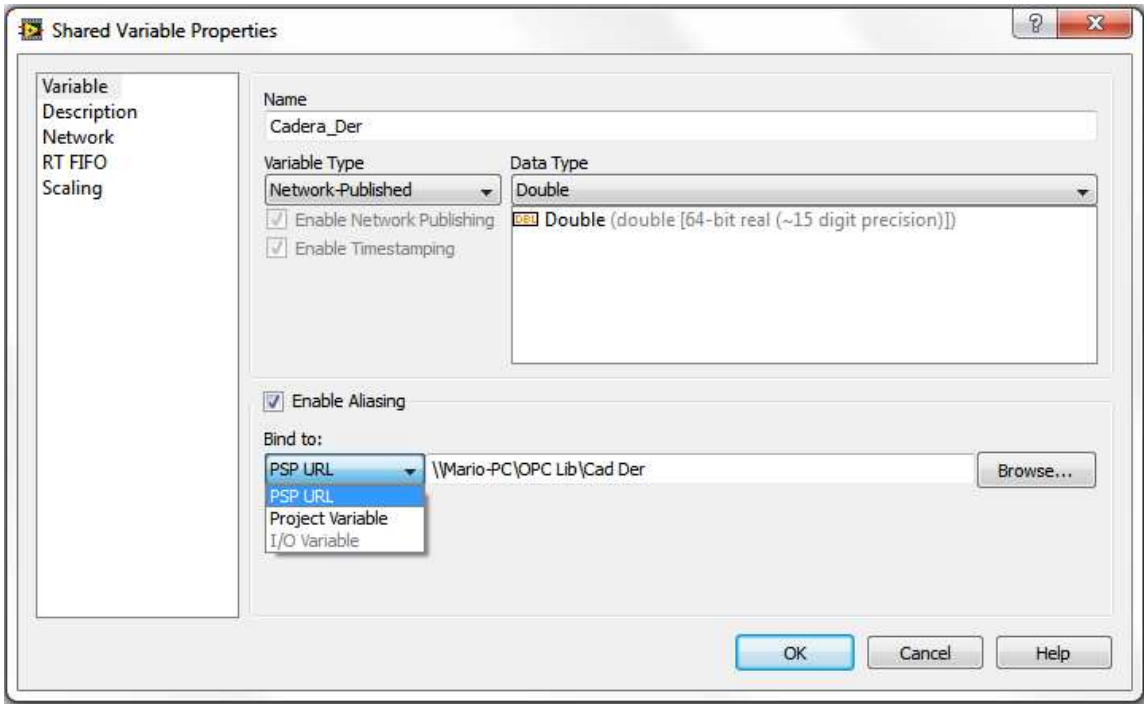


5.4.1 CONECTAR LABVIEW AL PLC USANDO UN SERVIDOR I/O.

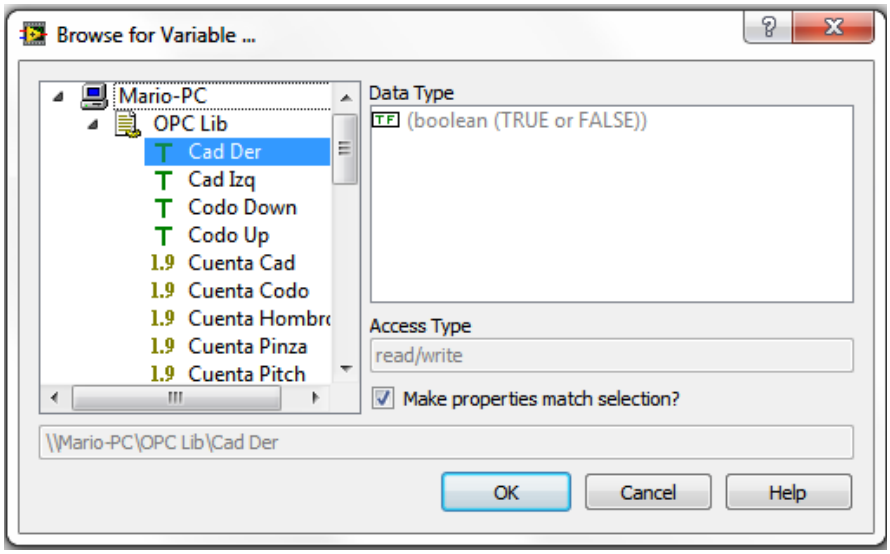
1. Para usar las etiquetas creadas anteriormente dentro de una aplicación, éstas deben ser agregadas al proyecto (New→Variable).



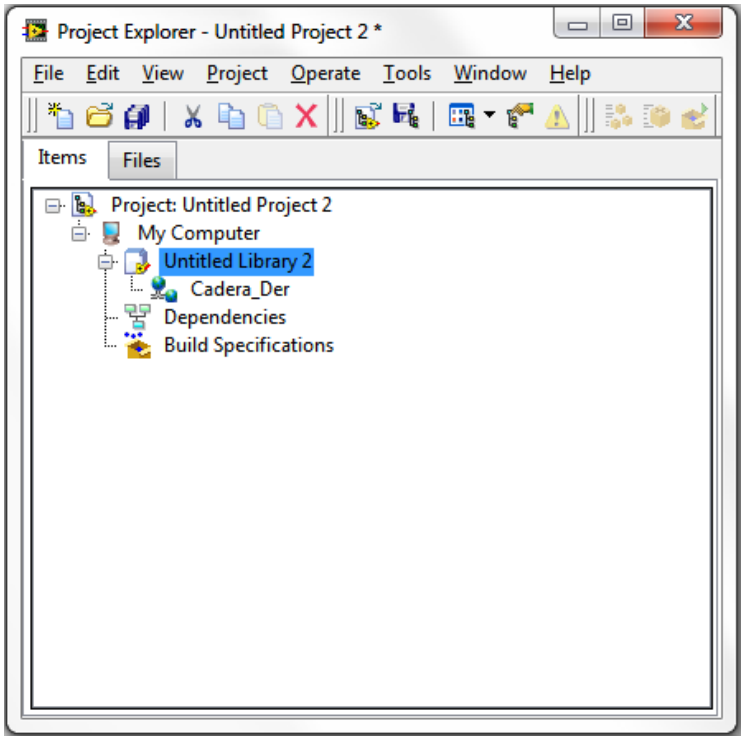
2. En la nueva ventana, se debe nombrar la variable, habilitar la casilla de “Enable Aliasing” y usar conexión tipo PSP URL.



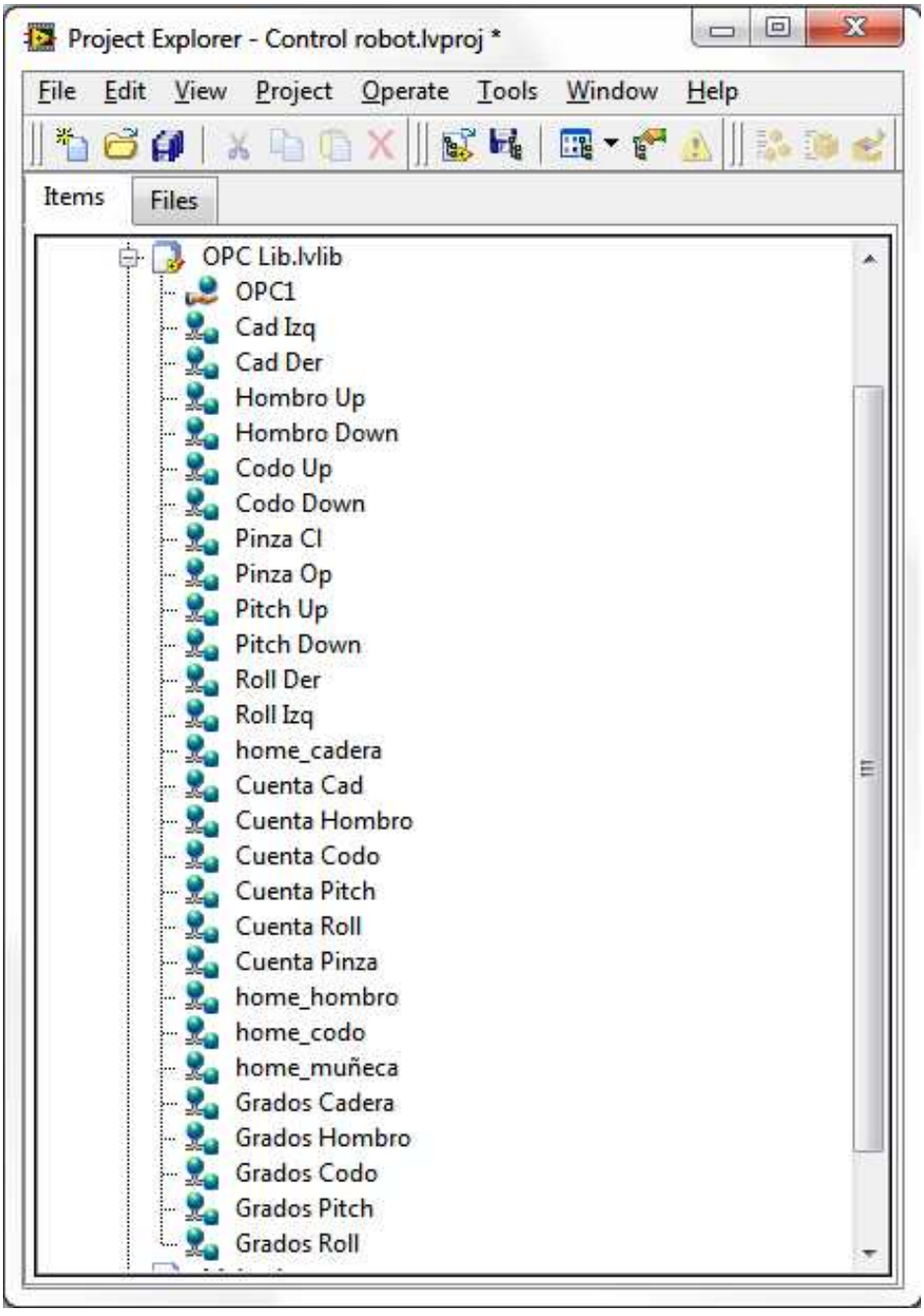
3. Posteriormente establecer la dirección donde se encuentran las variables del servidor OPC.



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.



6 IMPLEMENTACIÓN DE LA INTERFAZ DEL USUARIO

6.1 CONSTRUCCIÓN DE LA INTERFAZ GRÁFICA EN LA PC

6.1.1 VI MAIN

Ya que se han creado los 5 SubVI básicos del proyecto, se procede a diseñar un VI Main en el que se diseñe la interfaz de usuario para las simulaciones.

Antes de describir el Panel del Operador es necesario destacar que actualmente el único SubVI asociado a las variables globales del PLC es el Control Manual. Todos las demás funciones únicamente son simulaciones para el brazo modelado dentro de los SubVI.

El plan de diseño sugiere utilizar un *tab control* para mantener organizado el panel de usuario. Dicho *tab control* estará conectado a una *estructura case* que contiene en sus diferentes casos a los SubVI básicos de control.

6.1.2 CONTROL MANUAL

Como primer caso se tiene el control manual, el cual regresa como resultado la posición de cada articulación haciendo una simple conversión de cuentas a grados de cada una.

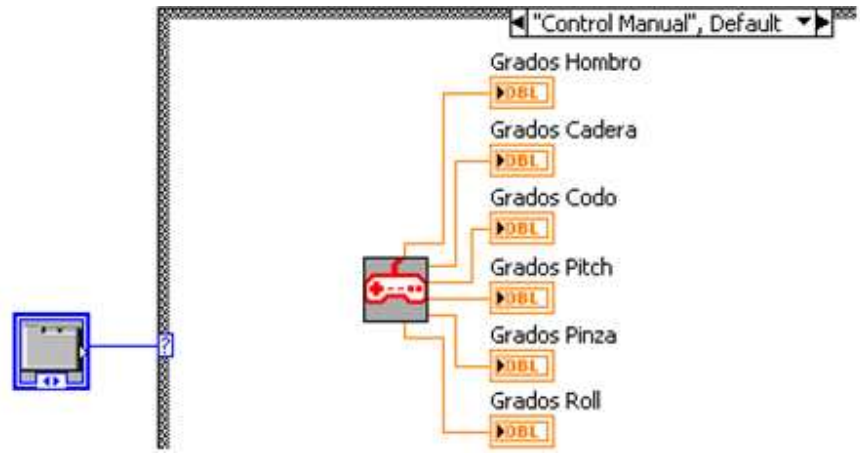


Figura 6.1. Diagrama de bloques del control manual.

Los botones que se muestran en la Figura 6.2 son las variables globales, provenientes del PLC, que se configuraron para el proyecto, éstas son arrastradas al panel frontal desde la ventana del proyecto y para diferenciarlas de un botón cualquiera tienen un indicador triangular sobre el botón, si éste se encuentra en color verde al ejecutar el programa significa que la variable está funcionando bien, si está en color rojo, puede haber un conflicto con la lectura de la variable.

Figura 6.2. Panel frontal del control manual.

Al agregar las variables globales al panel frontal también se obtienen las variables en el diagrama de bloques, éstas se posicionan fuera del ciclo while principal del Main y se observan cómo se muestra en la Figura 6.3.

Figura 6.3. Variables globales referidas al PLC.

Cabe mencionar que las variables globales fueron configuradas como de lectura y escritura, por lo que al activarse la variable desde la HMI se observará en el panel frontal y si se presionan los botones en el panel frontal controlaran las salidas del PLC.

6.1.3 CINEMÁTICA DIRECTA

La segunda pestaña del *Tab Control* está destinada a la Cinemática Directa, donde las entradas Theta 1, Theta 2, Theta 3, y Theta 4 están conectadas a 4 *controles Knob* y los ángulos alf3 y alf4 son constantes. Las salidas del SubVI CD serán las coordenadas cartesianas y se muestran mediante indicadores en el panel frontal.

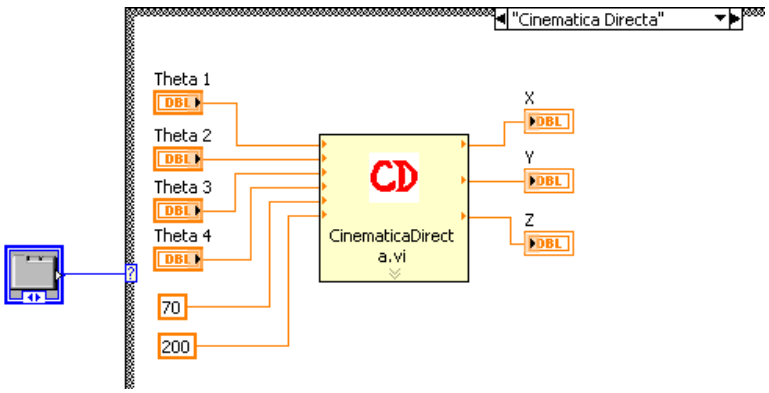


Figura 6.4. Diagrama de bloques para SubVI de cinemática directa.

Como se puede observar en la Figura 6.5 los controles Knob representan los grados de cada articulación por lo que es necesario restringirlos a los valores máximos que el Scorbot modelado permitiría físicamente, de esta forma la simulación se estará aproximando al espacio de trabajo real.

Figura 6.5. Panel frontal para cinemática directa.

6.1.4 CINEMÁTICA INVERSA

El tercer caso del Main contiene al SubVI CI, para el cual se requiere conectar a sus entradas tres controles numéricos para las coordenadas cartesianas y un botón booleano para elegir las dos posibilidades del codo. Como resultado obtenemos 4 ángulos que representan las posiciones de las articulaciones y serán mostrados en el panel frontal mediante 4 indicadores numéricos.

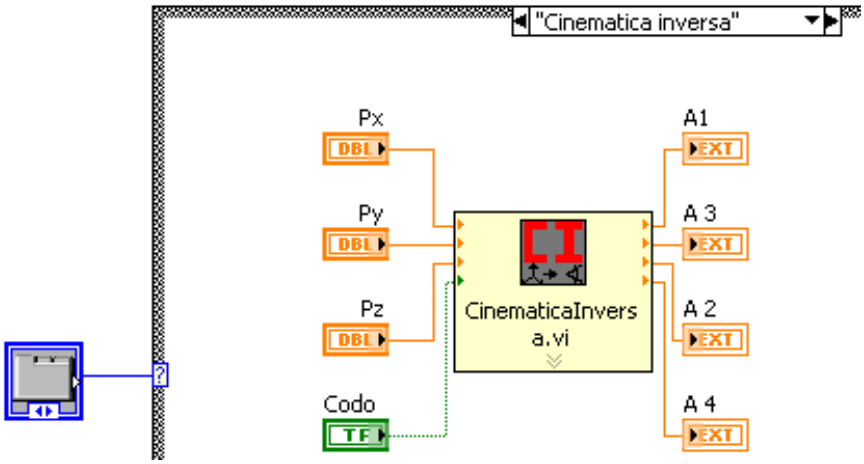


Figura 6.6. Diagrama de bloques para subVI de cinemática inversa.

Se podría suponer que al igual que en la CD se pueden restringir las coordenadas máximas para cada eje, pero el espacio de trabajo no es una superficie regular. Por otro lado, ya que las coordenadas que se introducen como PX, PY y PZ, se pueden comparar con las coordenadas del efector final de la imagen en 3D, si son diferentes quedará demostrado que el punto no puede ser alcanzado por el brazo.

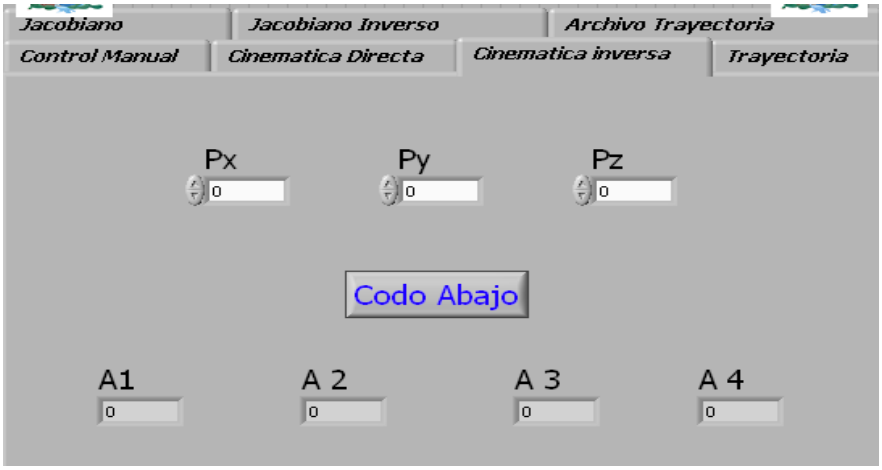


Figura 6.7. Panel frontal para cinemática inversa.

6.1.5 TRAYECTORIA

Como cuarta pestaña del tab control se tiene la Trayectoria, donde se requieren 3 arreglos de entrada y 2 botones booleanos. La constante True, que se puede observar en la Figura 6.8, conectada a la variable Recorrido es una forma de evitar que la secuencia de puntos sea observada en la Gráfica 3D mientras se trabaja en los otros casos.

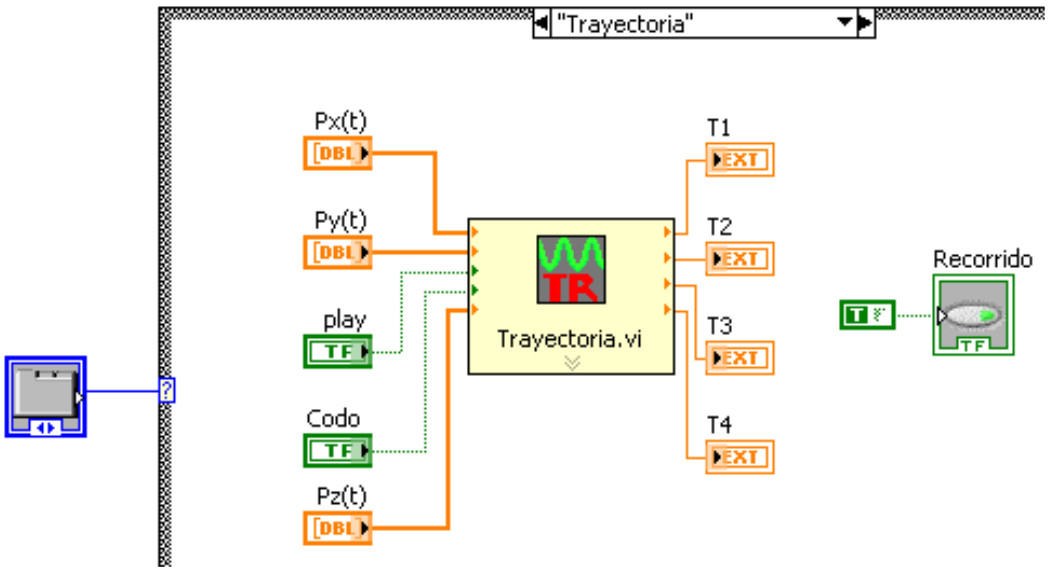


Figura 6.8. Diagrama de bloques para subVI de trayectoria.

Los arreglos de coordenadas pueden ser de dimensión mucho mayor a lo que se puede observar dentro del panel frontal, es lo que dará la fineza del movimiento y al igual que en la CI el movimiento puede hacerse con 2 configuraciones del codo. El botón de Play que se observa en la Figura 6.9 es el que inicia el movimiento comenzando el ciclo que se explicó en el capítulo 5.

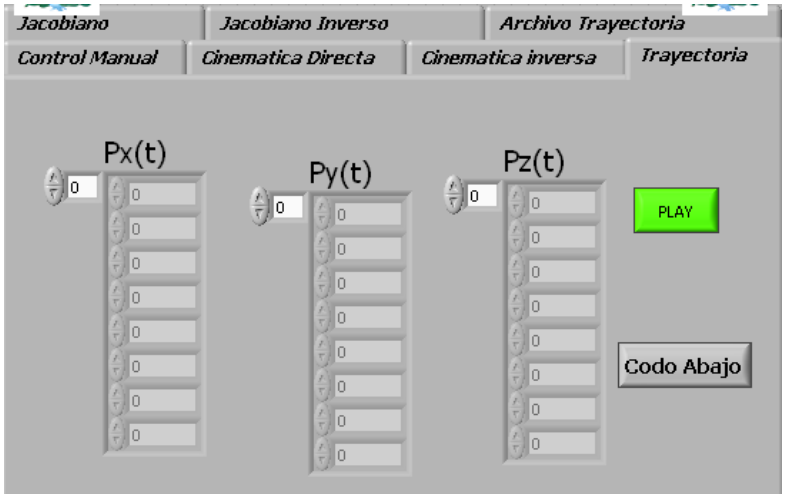


Figura 6.9. Panel frontal para trayectoria.

De esta forma al presionar Play, la simulación 3D recibirá tantas configuraciones del brazo como de coordenadas en los arreglos, llevando el brazo por cada punto de la trayectoria la cual también será observada en la gráfica, como se muestra en la Figura 6.10.

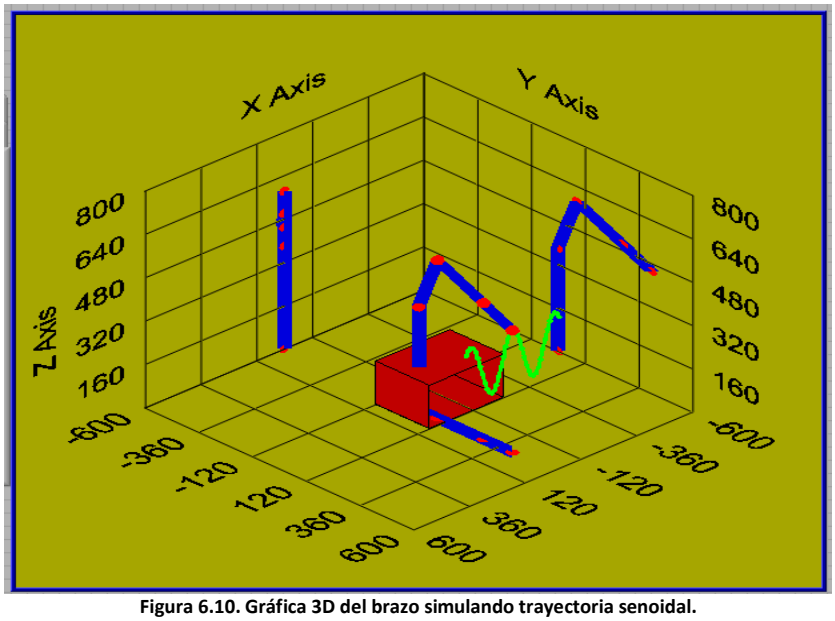


Figura 6.10. Gráfica 3D del brazo simulando trayectoria senoidal.

6.1.6 JACOBIANO Y JACOBIANO INVERSO

Otras funciones que se pueden programar para cubrir los principales temas de la Cinemática de un brazo son el Jacobiano y el Jacobiano Inverso.

Es importante aclarar que debido a que esas dos funciones se relacionan directamente con parámetros del movimiento real del brazo, es imposible hacer una simulación en la gráfica 3D, por lo que en el panel frontal sólo podrán obtenerse los resultados matemáticos para las velocidades de un punto en el caso del Jacobiano o las velocidades articulares en el Jacobiano Inverso.

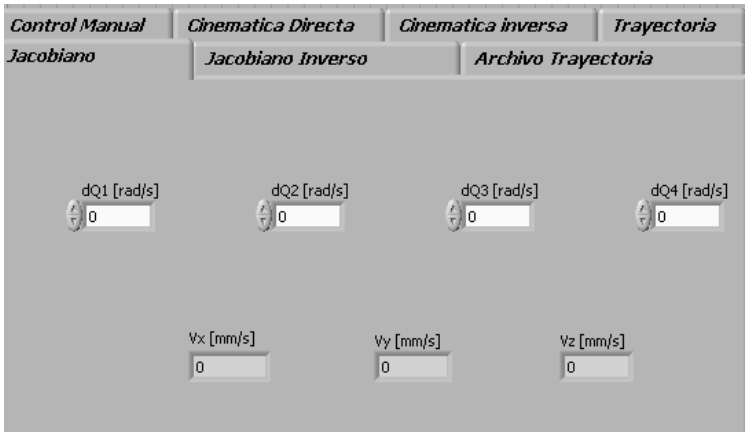


Figura 6.11. Panel frontal del Jacobiano

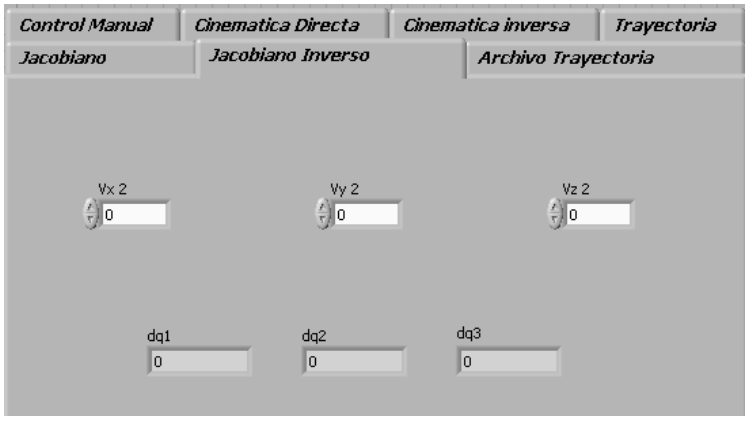


Figura 6.12. Panel Frontal del Jacobiano Inverso

Un SubVI auxiliar que se puede incluir en el Main es el utilizado para cargar un archivo con una trayectoria, lo que ahorra el trabajo de llenar los arreglos manualmente. Este archivo debe ser de texto (.txt) y los valores deberán tener la siguiente sintaxis:

x1, y1, z1, x2, y2, z2, x3,….zn

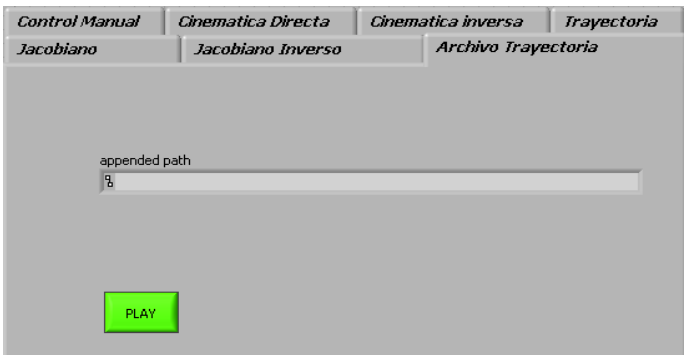


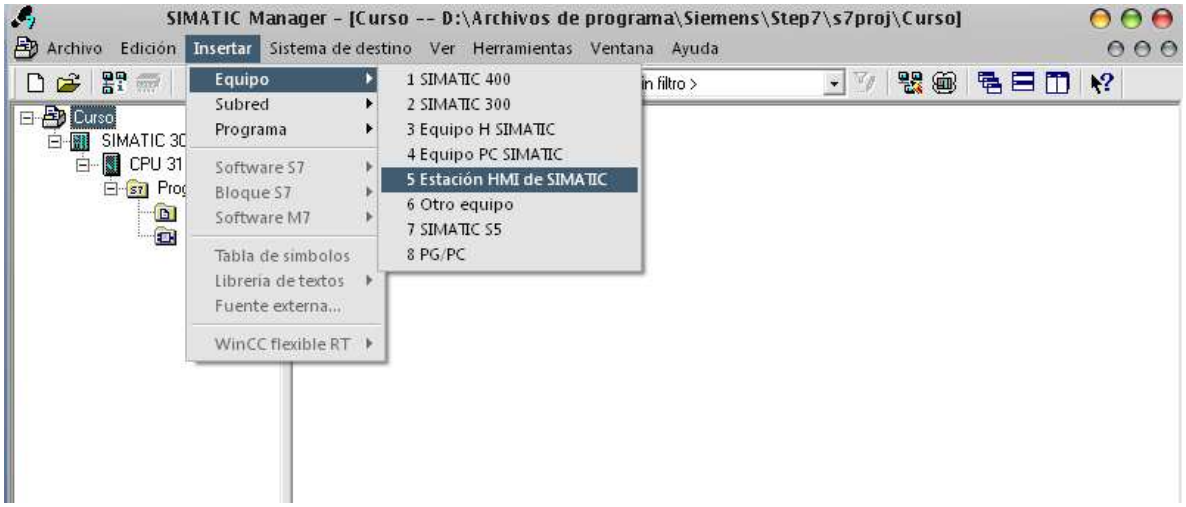
Figura 6.13. Panel frontal para cargar archivo de trayectoria

Cada caso entrega variables diferentes a su salida, por lo cual es necesario poner atención en las que serán enviadas al SubVI Graficador, el cual siempre necesita de los ángulos de las articulaciones. Por ejemplo, cuando se trabaja en el caso que contiene a la CD las variables de entrada a este VI serán las mismas que se envíen al Graficador y para el caso que contiene la CI los valores que se envíen al Graficador serán los que salgan de éste.

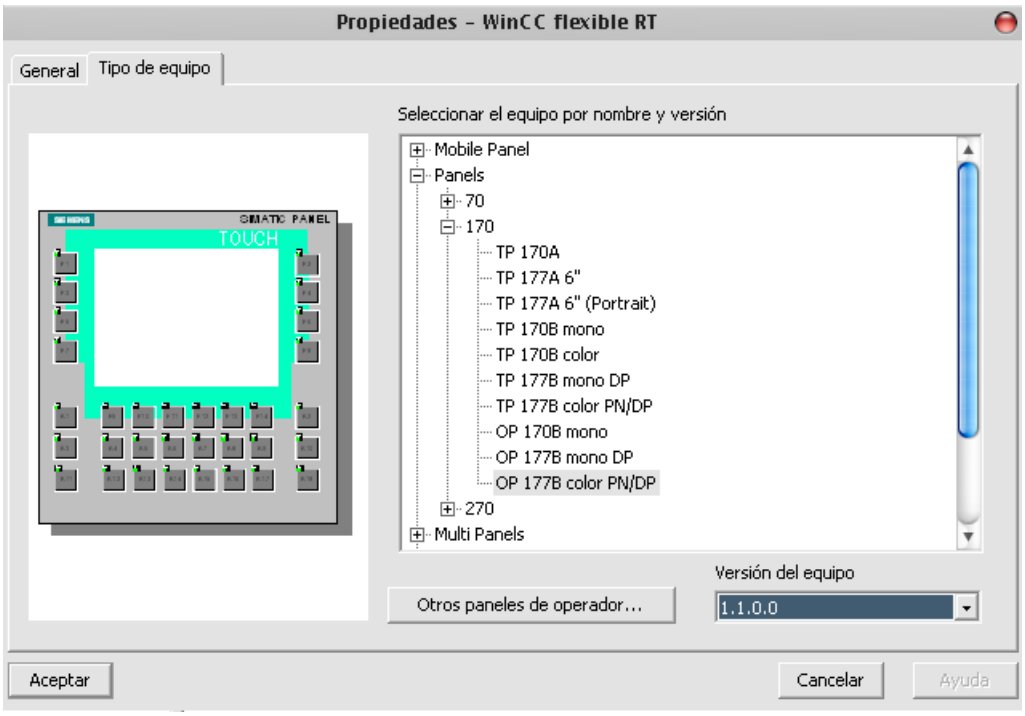
6.2 PROGRAMACIÓN DEL PANEL DE OPERADOR (HMI)

La interfaz hombre máquina (HMI) de Siemens requiere para su programación, del software WinCC Flexible, el cual es parte del SIMATIC Manager y permite configurar y programar la HMI. Para ello se realizan los siguientes pasos, tras la creación del proyecto para la configuración del PLC.

1. Agregar el equipo HMI dentro del proyecto mediante el SIMATIC Manager (Insertar→Equipo→Estación HMI de SIMATIC)

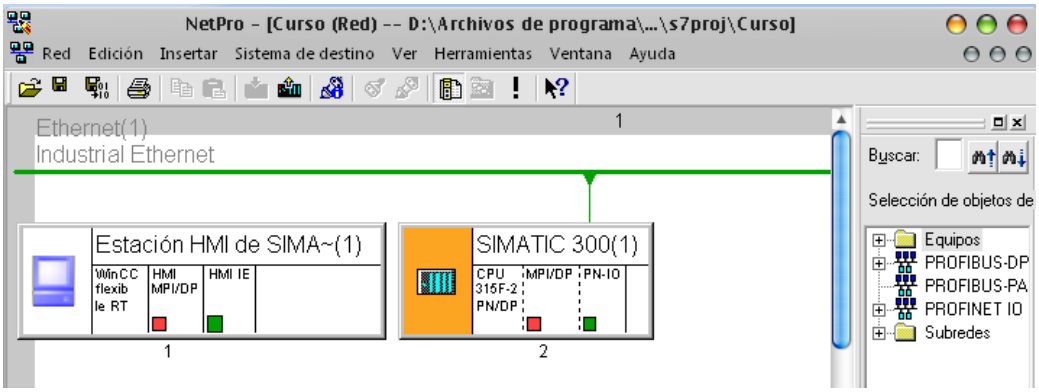


2. En la ventana que se despliega se muestra un asistente para seleccionar el tipo de pantalla usada. En este caso se trata de una OP 177B Color PN/DP. (Panels→170→OP 177B color PN/DP)

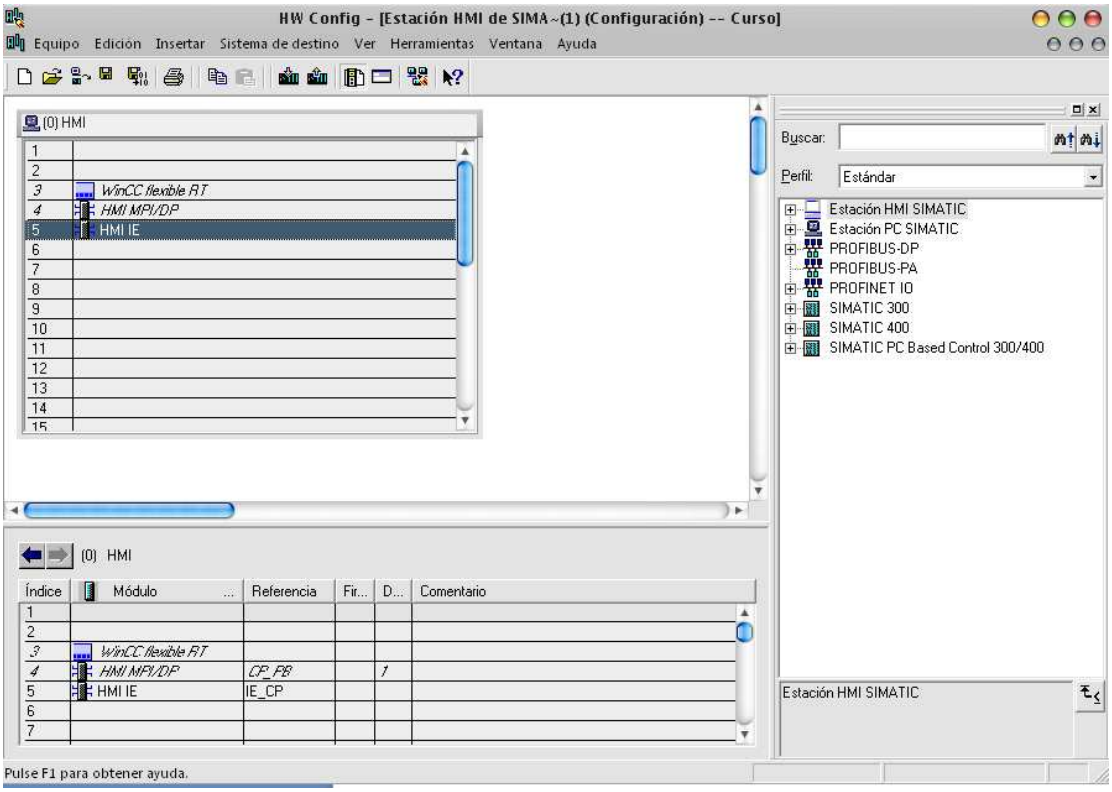


3. Una vez que se agrega se debe configurar para poder usarla en la red Ethernet. Para ello, dentro del SIMATIC Manager se debe abrir la red Industrial Ethernet 

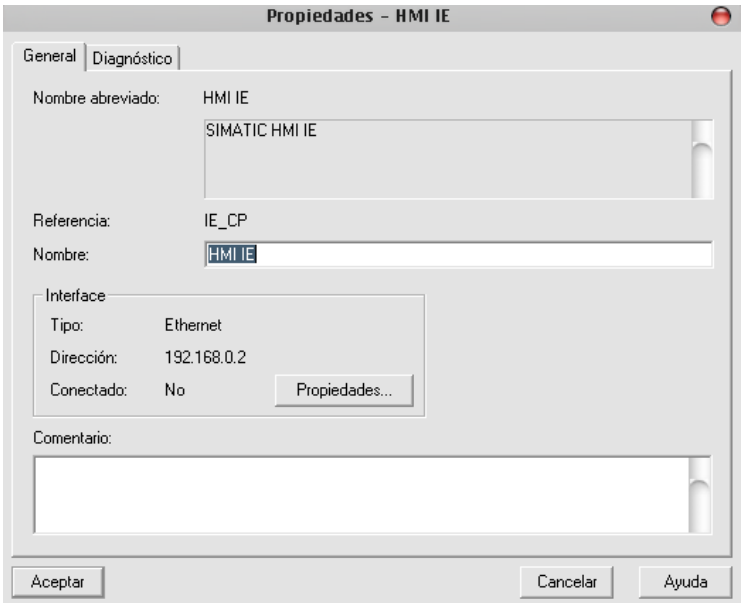
, la cual muestra una ventana con Net Pro cuya función es la configuración de las comunicaciones del proyecto.



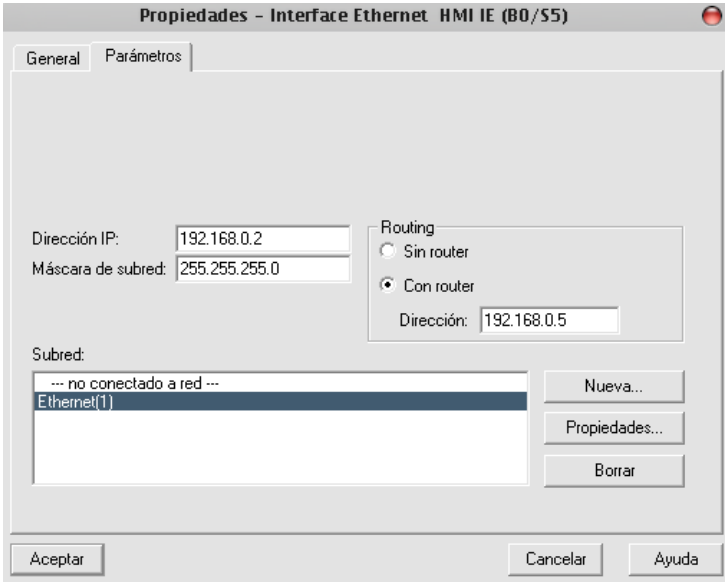
4. Se da doble clic sobre la estación HMI y se despliega una ventana con HW Config, pero ahora para la interfaz HMI. Dar doble clic sobre HMI IE dentro del bastidor de la pantalla.



5. En la nueva ventana se muestran las propiedades del puerto Ethernet de la pantalla, será visible un icono de Propiedades de la interfaz.



6. Configurar la dirección IP que tendrá la estación HMI, así como la dirección del ruteador, y seleccionar la red que se ha creado con anterioridad. Dar Aceptar en las dos ventanas que se abrieron, guardar los cambios y cerrar la ventana de HW Config.

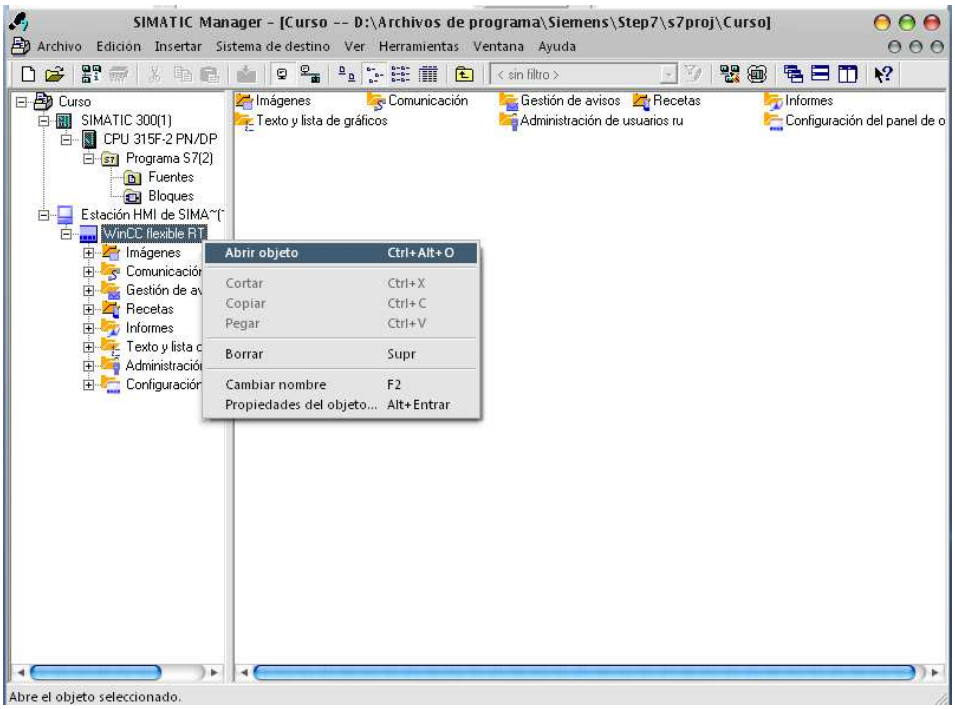


7. Ahora en la ventana de NetPro, se observa que la estación HMI, está conectada a la red Ethernet, representada mediante la línea verde, en caso de haber configurado un PLC también estará conectado a la misma red. Guardar los cambios y cerrar la ventana de NetPro.
   Con esto se concluye la parte de configuración de la red Ethernet, así como de sus componentes que son programados dentro del SIMATIC Manager.

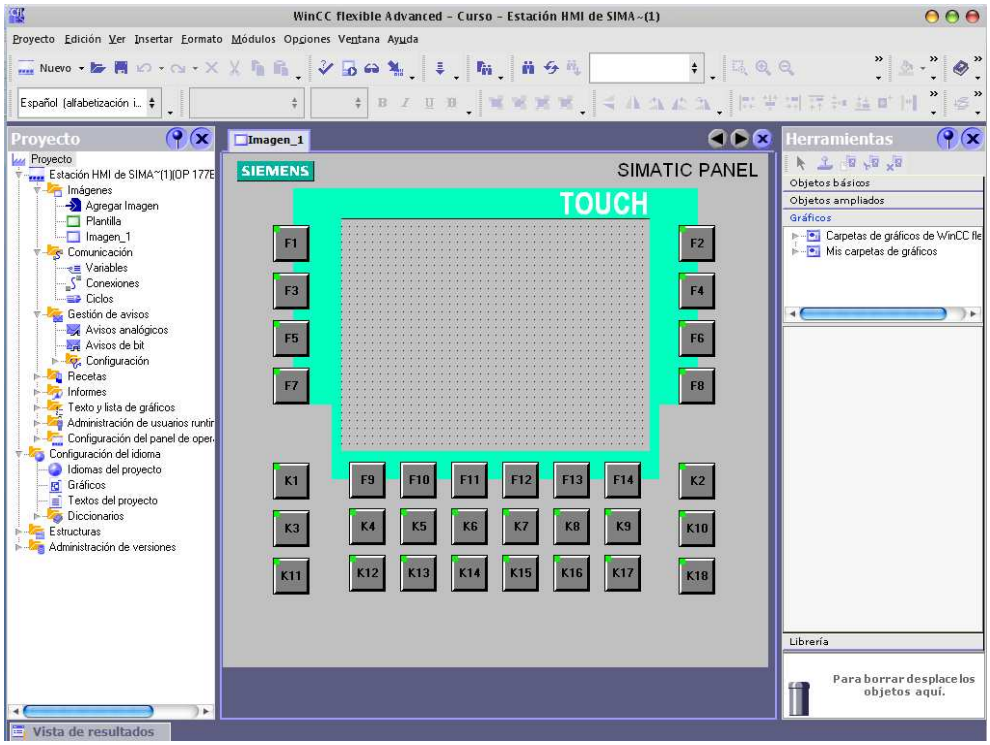
8. Por último transferir la configuración al PLC desde SIMATIC Manager  .

6.2.1 USO DE WINCC FLEXIBLE PARA PROGRAMACIÓN DE INTERFAZ HOMBRE MAQUINA (HMI)

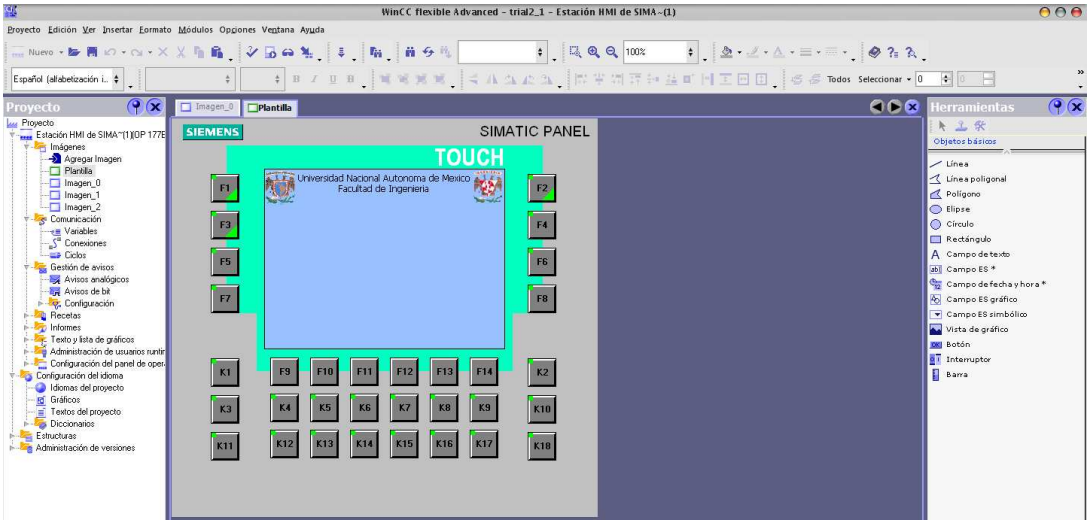
1. Dentro de SIMATIC Manager abrir WinCC Flexible para la interfaz configurada.

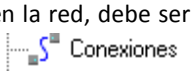


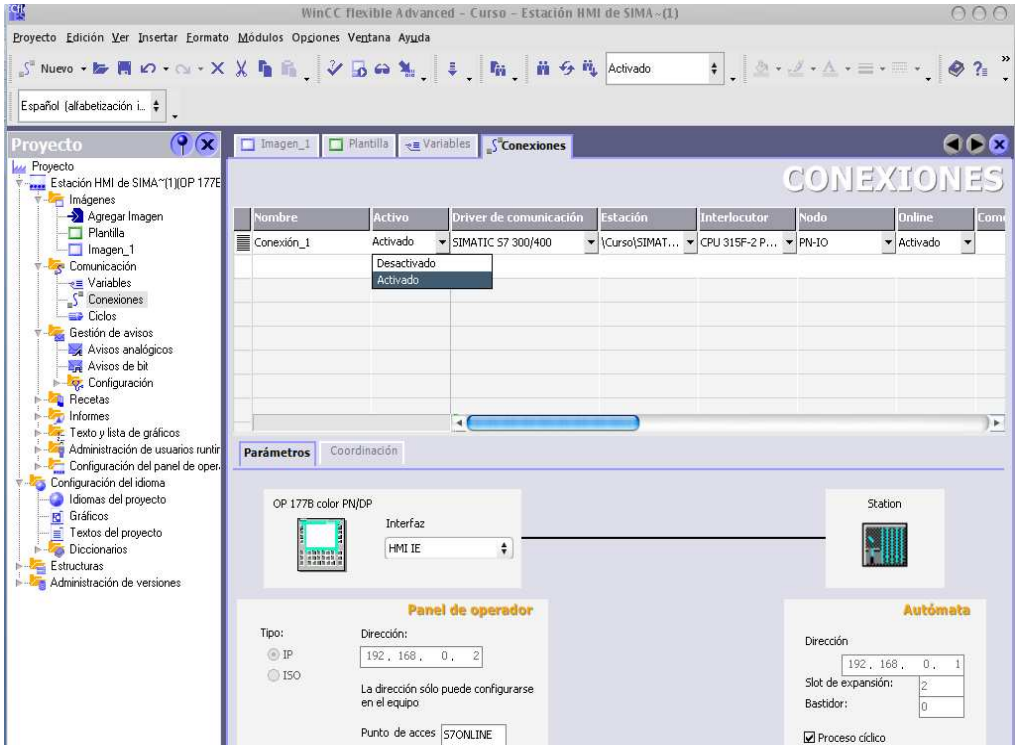
2. Una vez abierta la ventana con WinCC se visualiza una imagen que se asemeja al modelo de interfaz HMI configurado. En la parte de la izquierda un menú con los componentes del proyecto, y del lado derecho las herramientas para agregar imágenes a la pantalla.



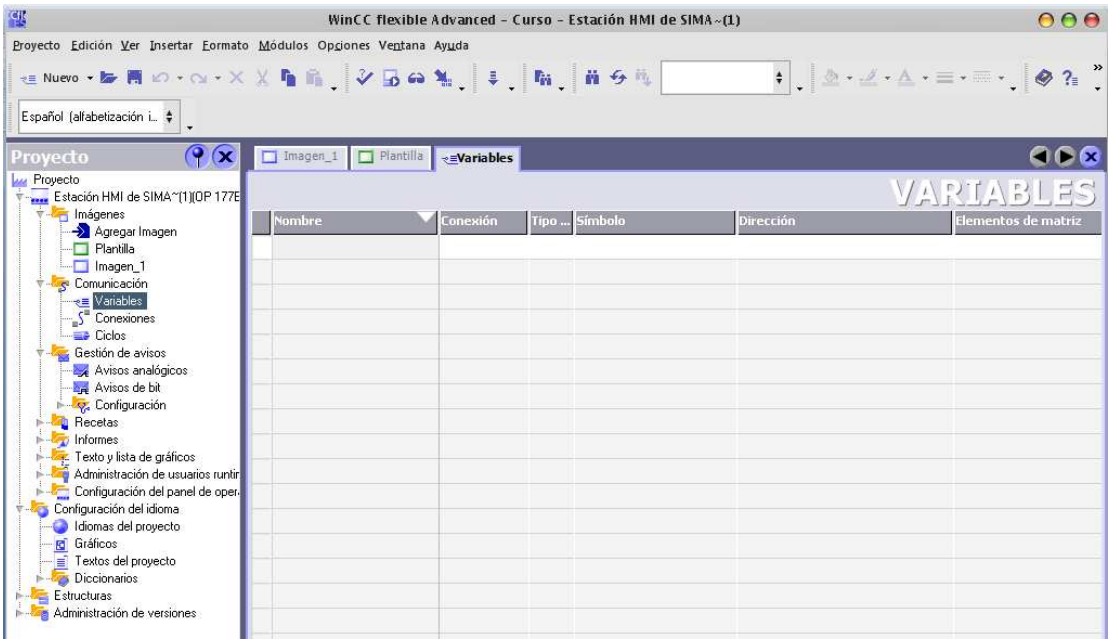
3. Dentro de la Plantilla del proyecto agregar las imágenes y títulos que irán como plantilla sobre todas las imágenes que se usen en el proyecto, como por ejemplo el logotipo de la empresa. Para ello basta con copiar la imagen deseada, y pegarla dentro del área de la pantalla y cambiar el tamaño y ubicación para ajustarla a las necesidades. De igual forma se pueden agregar un rectángulo de Objetos básicos del menú de herramientas, para establecer un color de fondo. Recalcando el ubicar al fondo  el rectángulo establecido como color de fondo. En este caso se usa esto para colocar los escudos de la Facultad de Ingeniería y de la UNAM.



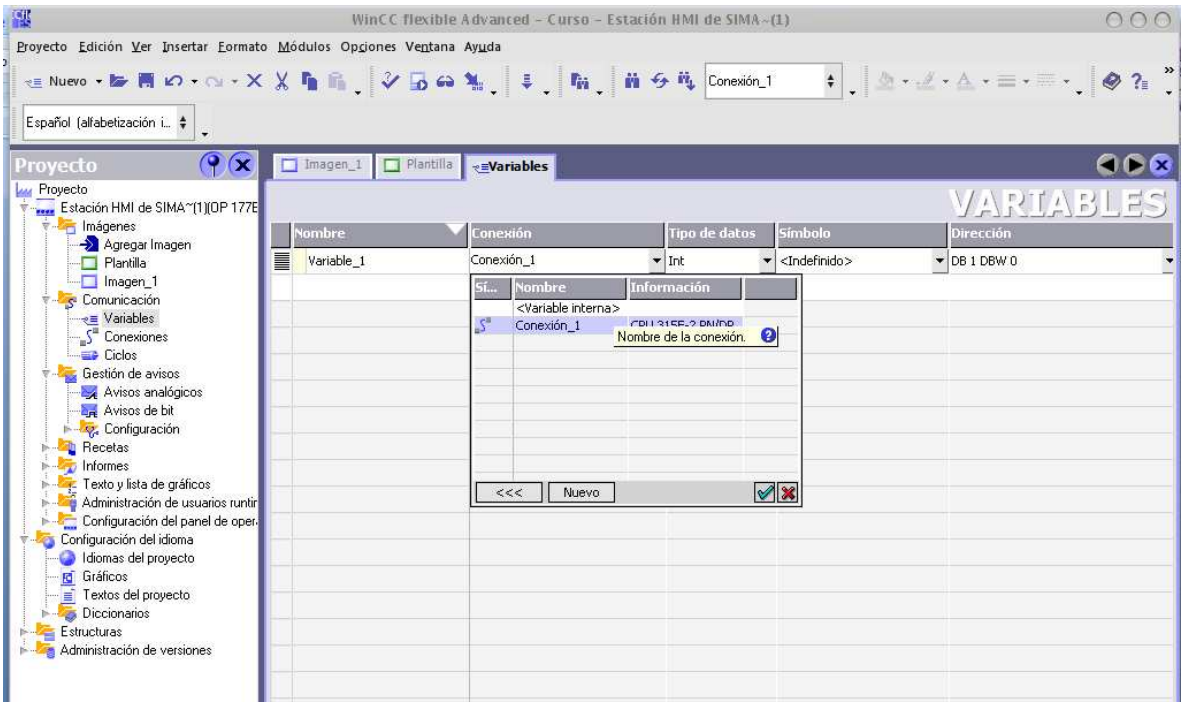
4. Para poder usar la pantalla en la red, debe ser activada la conexión, para ello dentro del Menú de Proyecto, abrir el Menú de Conexiones  y en la ventana que se abre seleccionar Activado.



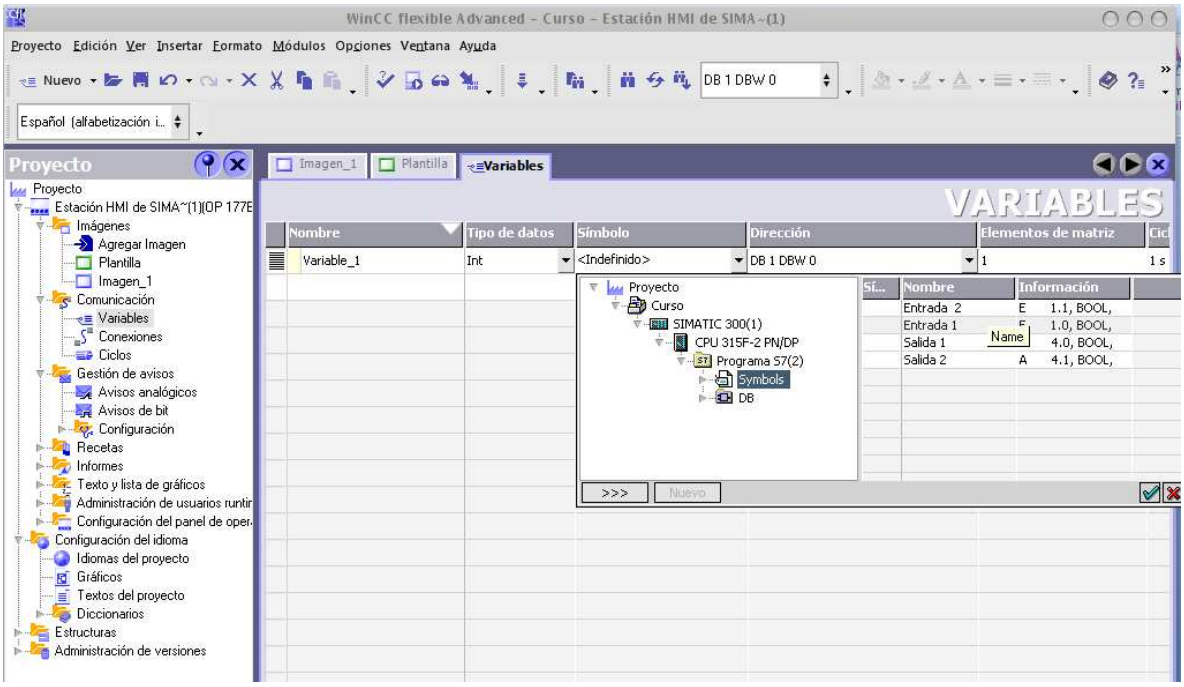
5. Ahora bien para poder usar la interfaz con el PLC es necesario definir las variables dentro de WinCC, para ello se abre la ventana de Variables Variables que se encuentra dentro del proyecto del lado izquierdo de la ventana de WinCC.



6. Se agregan las nuevas variables, las cuales están direccionadas a las variables usadas en el PLC.



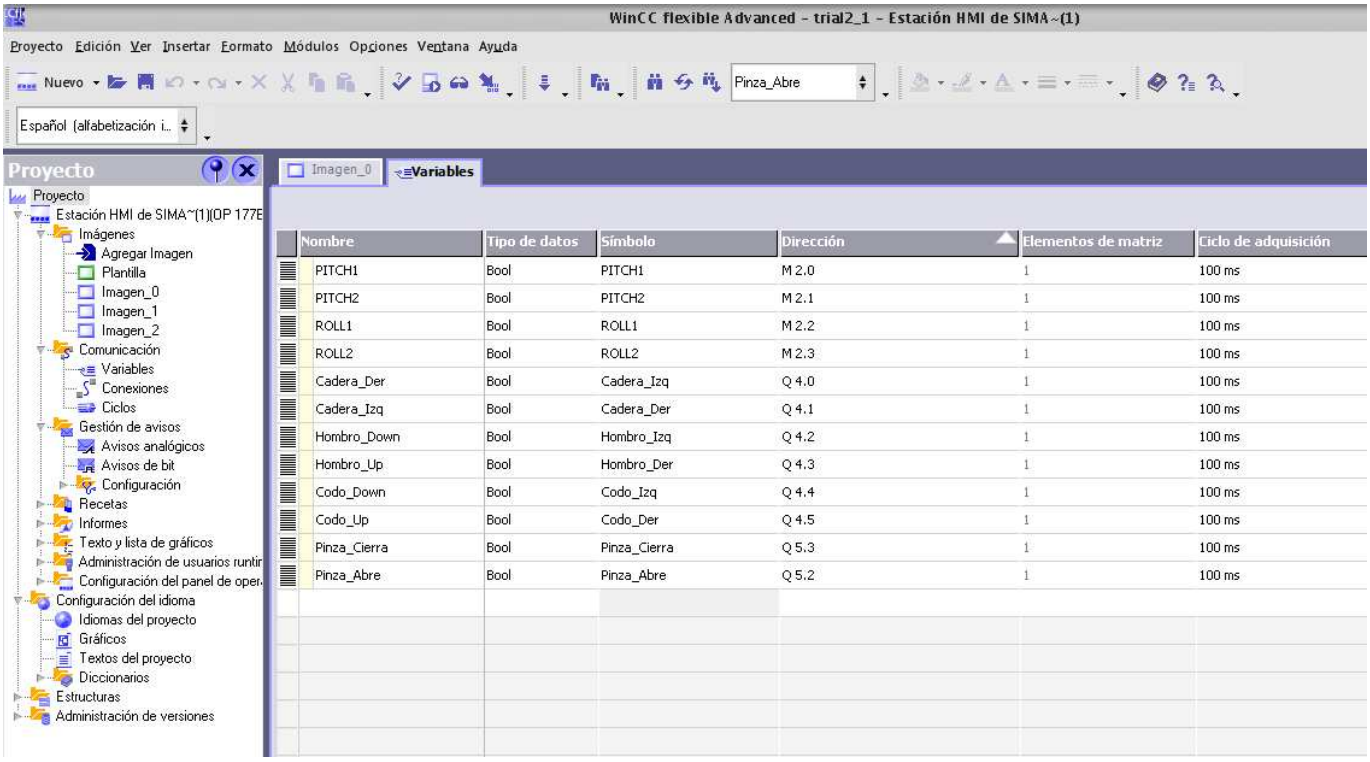
7. Para direccionar las variables, en la fila de Símbolo, se selecciona la tabla de símbolos creados para el PLC y seleccionar la variable a la que se vaya a asociar.



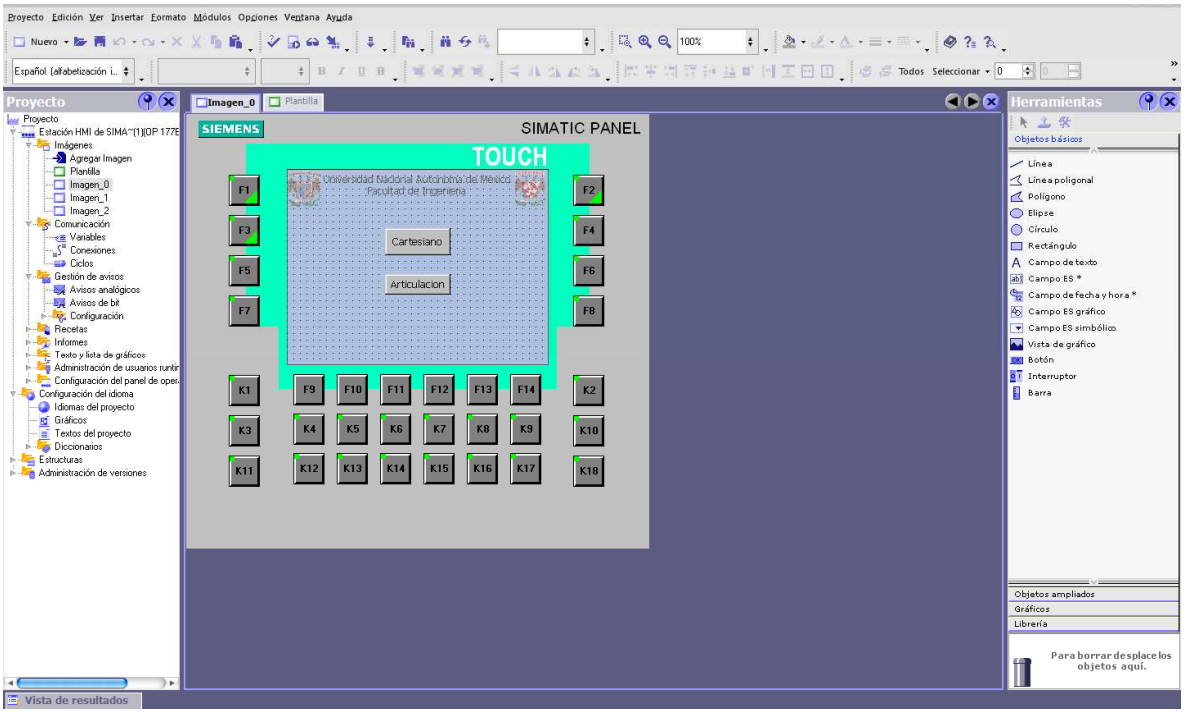
8. Para el control de las salidas del brazo se usan las variables que enciendan las entradas de los motores, así como se muestra en la tabla 6.1.

Tabla 6.1 Dirección de las variables asignadas a los botones de la interfaz HMI

Variable	Dirección
Cadera Izquierda	Q4.0
Cadera Derecha	Q4.1
Hombro Arriba	Q4.2
Hombro Abajo	Q4.3
Codo Abajo	Q4.4
Codo Arriba	Q4.5
Pitch Arriba	M2.0
Pitch Abajo	M2.1
Roll 1	M2.2
Roll2	M2.3
Pinza Abre	Q5.2
Pinza Cierra	Q5.3

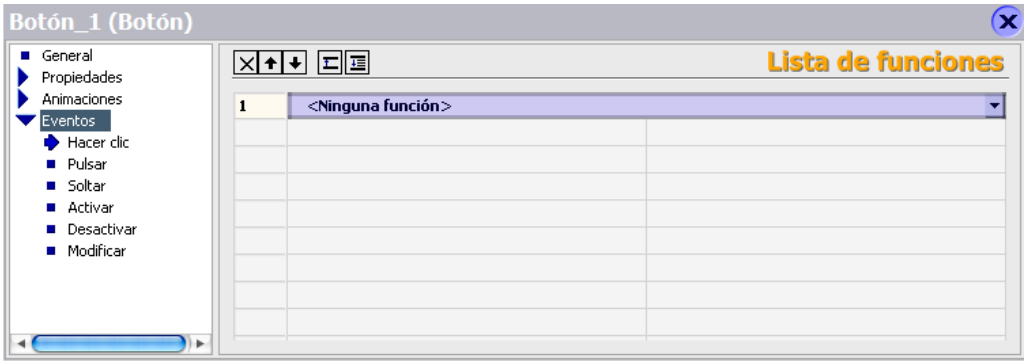


9. Como pantalla inicial se muestra un menú para seleccionar el tipo de movimiento a realizar ya sea por ejes cartesianos o por articulaciones. Para asignar las variables a botones desde la barra de Herramientas se arrastra un Botón de entre los Objetos Básicos.

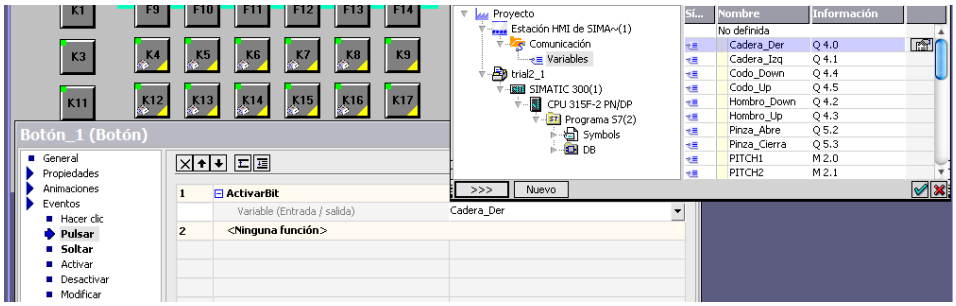


10. Dando doble clic sobre el botón se abrirá una barra que permite modificar diferentes parámetros, el de interés es el de cambiar de imagen dependiendo el botón que se presione.

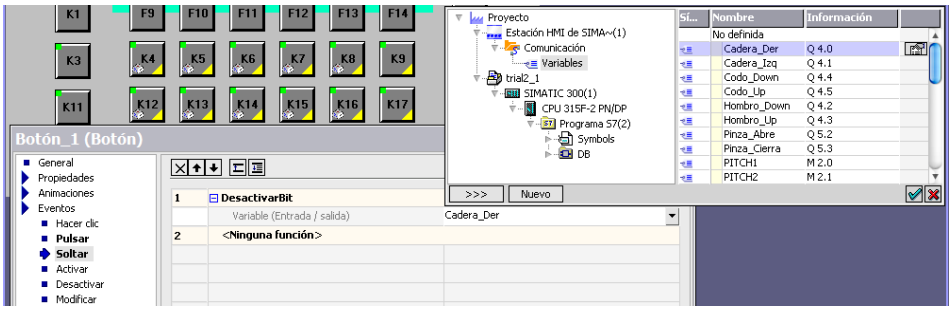
11. Para la imagen de movimiento por articulaciones, se tendrán botones que activen las salidas del PLC mediante el uso de las variables creadas. Para ellos se deben modificar los parámetros, el que nos interesa para esos botones son los Eventos.



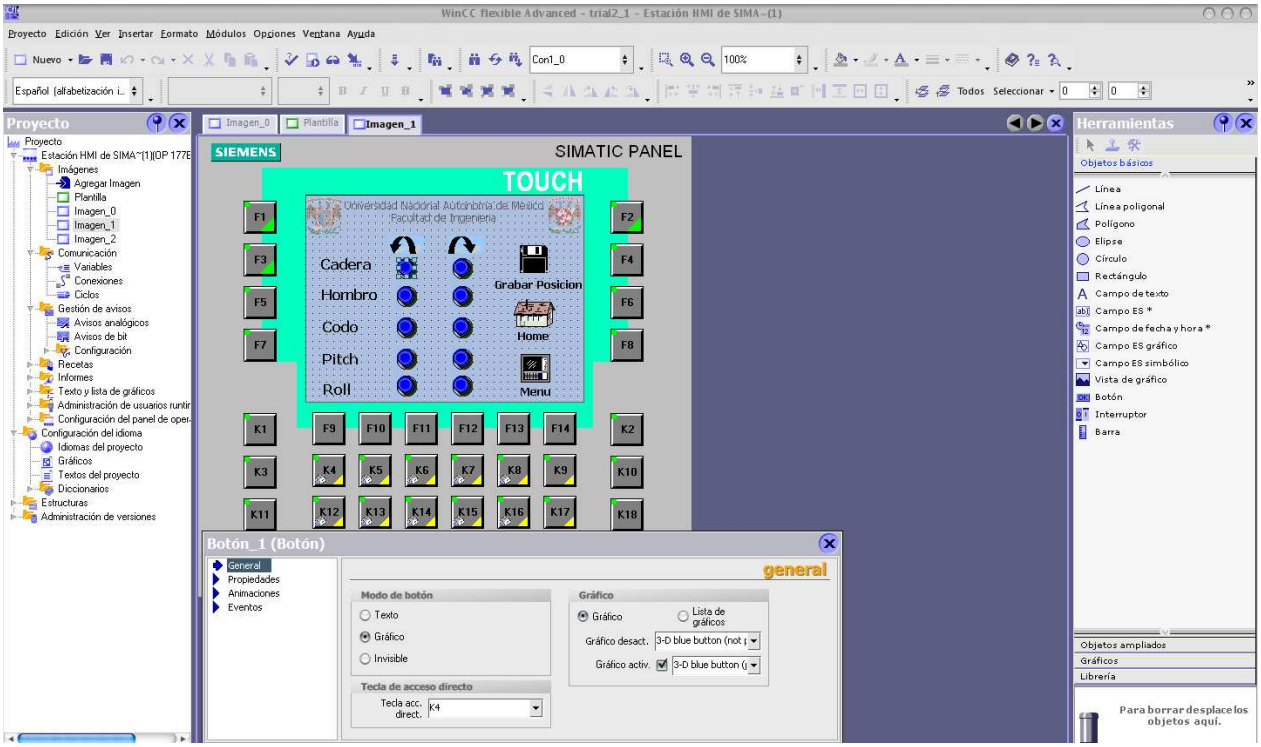
12. Para agregar una nueva función al evento **Pulsar**, en este caso **ActivarBit**, se mostrará un nuevo menú para seleccionar la variable a la cual estará asignada esta función.



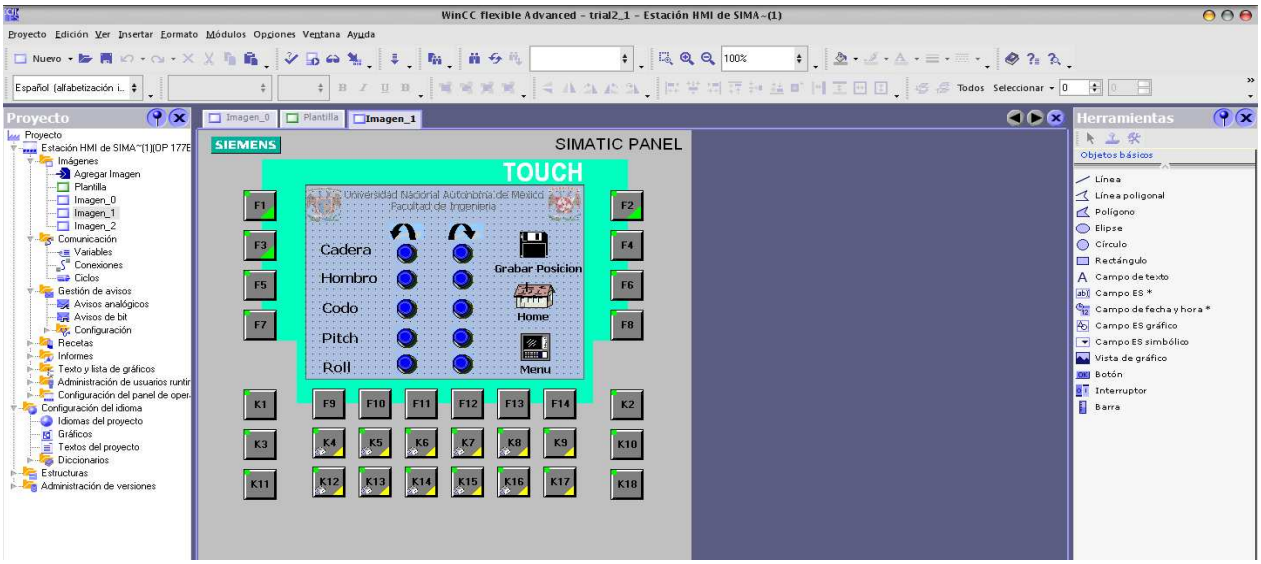
13. Para desactivar la entrada, se debe usar la función **DesactivarBit** del evento **Soltar**, y en el menú que se muestre seleccionar la variable a desactivar.



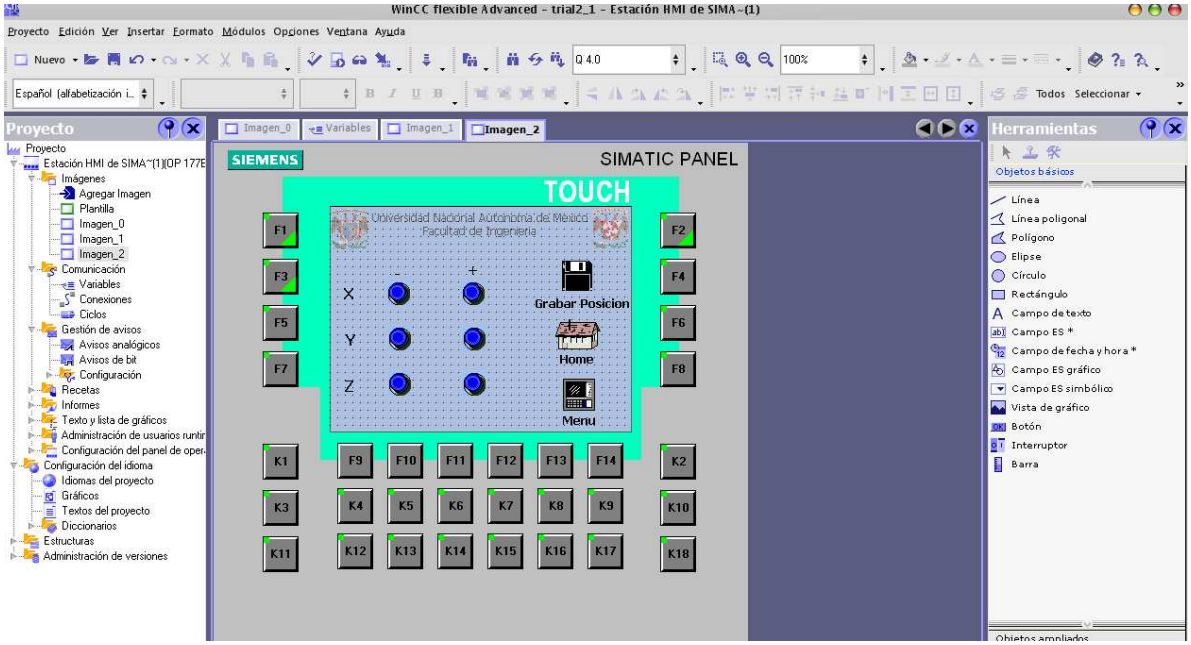
14. Del mismo modo para cada botón se asigna un botón físico de la interfaz HMI, para ello se selecciona dentro de las propiedades la tecla de acceso directo.



15. Esta asignación de eventos a los botones se hará para cada una de las variables descritas en el punto 9, quedando como se observa en la figura siguiente. Del mismo modo se crean botones para regresar al menú principal, un botón para ir a posición de Home y un botón para grabar la posición actual. Los botones de grabar posición y Home quedan de manera ilustrativa para el momento en que se tenga un control preciso sobre el robot.



Para el caso del movimiento por ejes cartesianos se crean de la misma forma botones para cada eje y los botones del menú, Home y grabar posición. En la siguiente imagen los botones para el movimiento por ejes se muestran solamente de manera ilustrativa para su implementación en trabajos futuros.



Por último una vez finalizado el proyecto, se guarda y se transfiere a la interfaz HMI (Proyecto→Transferir→Configuración de la transferencia).

7 SIMULACIÓN DE CONTROL MEDIANTE PIC

Para establecer una comparación entre el trabajo desarrollado con en PLC y el posible uso de microcontroladores PIC se hace uso de software de simulación de prototipos electrónicos como lo es PROTEUS, el cual da la posibilidad de utilizar una gran gama de componentes electrónicos entre ellos una amplia variedad de PIC. El desarrollo de los códigos de los microcontroladores se estableció usando el software PIC C Compile, el que permite el uso de lenguaje de programación basado en C, el cual tiene una serie de instrucciones genéricas enfocadas a la configuración de los PIC sin tener que estar revisando el set de instrucciones de cada uno en particular.

Una vez desarrollada dicha simulación, su manejo se hace mediante una interfaz en la PC la cual dará el control del movimiento de los motores así como la manipulación de algún otro tipo de datos que se requieran. Para ello es necesario contar con un tipo de comunicación entre la PC y el prototipo simulado, con lo cual se tiene una simulación aún más completa de cómo quedaría un control de los motores de un brazo robótico mediante la tecnología mencionada.

7.1 DISEÑO DEL CONTROL MEDIANTE PIC

7.1.1 PIC16F876A

El PIC16F876A es el utilizado en las simulaciones por ser uno de los más comunes en el mercado y dada la posibilidad de obtener muestras gratuitas por parte de la empresa que los fabrica. Forma parte de la subfamilia PIC16F87X de microcontroladores PIC (*Peripheral Interface Controller*) de gama media de 8 bits, fabricados por Microchip Technology Inc. Las características principales de esta familia son las siguientes:

- CPU de arquitectura RISC (*Reduced Instruction Set Computer*).
- Set de 35 instrucciones.
- Frecuencia de reloj de hasta 20MHz (ciclo de instrucción de 200ns).
- Todas las instrucciones se ejecutan en un único ciclo de instrucción, excepto las de salto.
- Hasta 8K x 14 palabras de Memoria de Programa FLASH.
- Hasta 368 x 8 bytes de Memoria de Datos tipo RAM.
- Hasta 256 x 8 bytes de Memoria de Datos tipo EEPROM.
- Hasta 15 fuentes de Interrupción posibles.
- Modo de bajo consumo (Sleep).
- Tipo de oscilador seleccionable (RC, HS, XT, LP y externo).
- Rango de voltaje de operación desde 2,0V a 5,5V.
- Convertidor Analógico/Digital de 10 bits multicanal.
- 3 Temporizadores.
- 2 módulos de captura/comparación/PWM.
- Comunicaciones por interfaz USART (Universal Synchronous Asynchronous Receiver Transmitter).
- Puerto Paralelo Esclavo de 8 bits (PSP).
- Puerto Serie Síncrono (SSP) con SPI e I^2C.

De las características mencionadas cabe destacar la memoria EEPROM, los temporizadores, los módulo CCP en modo PWM y el puerto serie e I2C, las cuales son usadas para el desarrollo del control de los motores.

TIMERS

Los temporizadores o TIMER son módulos integrados en el PIC, los cuales permiten realizar cuentas ya sean internas o de eventos externos. Cuando se usan internamente se usa el término timer mientras que cuando se usan externamente son contadores, por lo que mediante estos bloques se hará el conteo de los pulsos generados por el encoder de cada motor.

El TIMER0 es un contador de 8 bits, incrementado físicamente contando los eventos externos conectados al pin RA4/TOCK1 o mediante programación contando los pulsos internos del reloj.

Cuenta con un preescalador, el cual es un divisor de frecuencia configurable a dividir entre 2, 4, 8, 16, 32, 64, 128 ó 256. Cabe mencionar que la frecuencia de conteo será una cuarta parte de la frecuencia del reloj, y posteriormente dicha frecuencia de conteo es posible dividirla usando el preescalador.

El TIMER1 es otro contador el cual se diferencia por manejar 16 bits, su preescalador sólo puede dividirse entre 2, 4 y 8 y además cuenta con 3 modos de funcionamiento:

- Temporizador. Incrementándose cada ciclo de instrucciones.
- Contador Síncrono. El cual sincronizará la señal externa con la fase del reloj interno.
- Contador Asíncrono. La señal externa no está asociada al reloj interno, por lo tanto se puede seguir incrementando aún en modo SLEEP del PIC.

Cabe destacar que el TIMER1 en modo contador únicamente tomará los flancos de subida como flanco activo y una vez configurado se debe producir un flanco de bajada para comenzar el conteo.

El TIMER2 es de 8 bits, posee un preescalador con divisiones entre 4 y 16 y un postescalador divisible hasta entre 16. Asimismo el TIMER2 puede ser usado como base de tiempo para la modulación en ancho de pulso (PWM).

Para el uso de los timer en lenguaje C existen las siguientes funciones de configuración:

```
setup_timer_0(modo);

setup_timer_1(modo);

Setup_timer_2(modo, periodo, postescalador);
```

El parámetro modo es cambiado por la configuración de conteo interno o externo, y el uso de los escaladores, para el caso del TIMER2 el periodo es un valor entero de 8 bits, y el postescalador es un valor entre 1 y 14

Asimismo para la lectura o escritura en los timer el lenguaje C tiene las siguientes funciones:

```
set_timerX(valor);

valor=get_timerX();
```

MÓDULO CCP

El módulo CCP permite realizar tres funciones básicas basadas en el manejo de los timer:

- Comparador: compara el valor del timer con el valor de un registro para la realización de cierta acción.
- Captura: obtiene el valor del timer en un momento específico, fijado en la programación del PIC.
- PWM: genera una señal de tren de pulsos.

Cada módulo CCP posee un registro de 16 bits el cual puede ser usado para capturar el valor del timer, para comparar el valor del registro con el TIMER1 o como registro de 10 bits para el ciclo de trabajo de una señal PWM.

En este caso el de interés es el modo PWM, el cual permite obtener en los pines CCPx una señal periódica en la cual se puede variar su ciclo de trabajo.

El compilador C tiene ciertas funciones para el manejo del módulo CCP, para la configuración del mismo se usa:

setup_ccpx(modo)

en este caso el *modo* será CCP_PWM, para definir el ciclo de trabajo se usa:

set_pwm_duty(valor);

donde *valor* es un dato de 8 ó 16 bits.

MODOS DE COMUNICACIÓN.

Los PIC utilizan, dos modos de transmisión en serie:

- El puerto serie síncrono (SSP)
- La interfaz de comunicación serie (SCI) o receptor transmisor serie síncrono-asíncrono universal.

PUERTO SERIE SÍNCRONO (SSP)

El SSP se suele utilizar en la comunicación con otros microcontroladores o con periféricos. Las dos interfaces de trabajo son:

- Interfaz serie de periféricos (SPI). Comunicación entre microcontroladores de la misma, o diferente, familia en modo maestro esclavo; Full dúplex.
- Interfaz Inter-circuitos (I^2C). Interfaz con capacidad para comunicar microcontroladores y periféricos; Half dúplex.

MÓDULO USART

La función del módulo USART es recibir y transmitir datos en serie, ya se síncrona o asíncrona. Mientas que la señal síncrona utiliza una seña de reloj y una línea de datos, las transmisiones asíncronas emplean reloj en el emisor y en el receptor, los cuales deben ser de igual frecuencia y deben estar en fase, para ello la frecuencia del reloj se establece antes de la transmisión de datos configurando la velocidad. Cada trama de datos tiene un tamaño fijo y poseen un bit de arranque y un bit de parada, los cuales permiten realizar la sincronización, su ventaja se presenta al poder transmitir y recibir simultáneamente ya que usa dos líneas (full-dúplex) una transmisora o TX y otra receptora o RX.

Para el uso de esta comunicación los PIC tienen dos pines específicos para este fin, que son RC6/TX y RC7/RX.

Para el uso del módulo USART en C se debe establecer la configuración con la directiva:

#USE RS232 (opciones)

donde se configuran los parámetros como velocidad de transmisión, pines utilizados, etc.

Para la transmisión de datos existen varias funciones dentro del lenguaje C, las cuales son:

putc(cdata)

putchar(cdata)

donde *cdata* es un carácter de 8 bits, el cual será enviado usando el pin de TX.

puts(string)

donde string es una cadena de caracteres constante o matriz de caracteres terminada con un 0.

printf(fname, cstring, values)

donde *cstring* es una cadena de caracteres o matriz de caracteres, *fname* son las funciones a utilizar para escribir la cadena indicada y los valores son valores a incluir en la cadena separados por comas.

Para la recepción de datos las funciones existentes son:

value = getc()

value = getch()

valuer = getchar();

donde *value* es un carácter de 8 bits. Estas funciones esperan recibir un carácter por la línea y después devolver su valor.

Existe además una función que indica si existe un valor entrante, la función *kbhit*(), será 0 si no hay datos entrantes, y será 1 en el momento en que haya un carácter listo para ser leído.

INTERFAZ I²C

El modo de trabajo I2C se basa en la comunicación a través de 2 hilos. Cada dispositivo conectado al bus tiene una dirección específica. Puede configurarse como comunicación de un maestro y varios esclavos o una configuración multimaestro. En ambas instancias el maestro es el que tiene la iniciativa en la transferencia, decide con quién se comunicará, si será envío o recepción de datos y el momento en que finalice. Cuando el maestro inicia la comunicación envía la dirección del dispositivo con el que se comunicará y los esclavos comparan este dato con su dirección y el último bit de estos datos indica si será envío o lectura de datos.

Los hilos usados en el bus I2C son dos líneas de colector abierto: la señal del reloj SCL o pin RC3 y la línea de datos SDA o pin RC4. Se deben utilizar unas resistencias externas (pull-up) para asegurar un nivel alto cuando no hay dispositivos conectados al bus.

La transmisión de las señales se hace mediante un bit de inicio, posteriormente se envían los datos y al final se envía un bit de parada, como se muestra en la Figura 7.1.

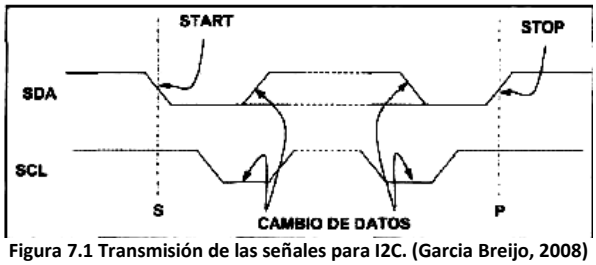


Figura 7.1 Transmisión de las señales para I2C. (Garcia Breijo, 2008)

Una vez que el maestro envía la dirección o datos, el esclavo genera un bit de reconocimiento, para informar que recibió datos, en caso de que el maestro no reciba dicho dato, la comunicación se interrumpe, generando una señal de Stop. En caso de que el maestro sea el que recibe datos, será éste el que envíe el bit de reconocimiento, como se muestra en la Figura 7.2.

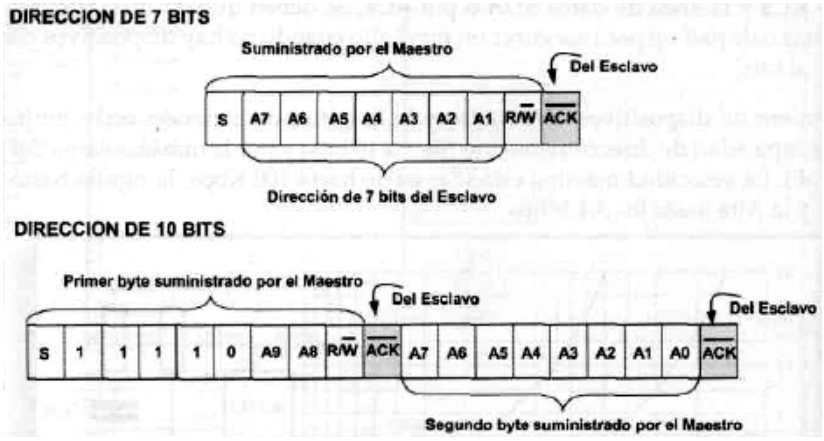


Figura 7.2. Formato de los datos enviados. (Garcia Breijo, 2008)

Para el uso de la comunicación I2C se debe configurar con la directiva:

#use I2C(opciones)

Las funciones asociadas son:

I2C_WRITE(dato). El dato es un entero de 8 bits enviado por el bus.

I2C_START(). La cual inicializa la transmisión de datos.

I2C_READ(). El dato recibido será un entero de 8 bits.

I2C_STOP(). Finaliza la transmisión.

I2C_POLL(). Se pone en 1 si se recibió un dato en el buffer y un 0 cuando no se ha recibido.

I2C_SLAVEADDR(adr). Especifica la dirección del dispositivo en modo esclavo.

7.1.2 PROGRAMACIÓN DE LOS PICS

Para el control de cada motor se tendrán algunas funciones simples como la asignación de dirección de giro usando pines de salida de alguno de sus puertos conectados a los pines de entrada del puente H, otra tarea aún más compleja es implementar una señal PWM usando el Timer2 conectado al pin Enable del puente para hacer el control de la velocidad. Y finalmente la función que podría ser la principal es la de llevar el conteo de los pulsos de los encoders para ello se usan dos timer en modo contador para que uno sea preescalado y cuente únicamente un pulso cada 128, mientras que el otro contador contará cada dos pulsos, lo cual dará un valor real del número de pulsos contados. Dadas las funciones a realizar se debe usar un PIC por cada motor a controlar, tomando esta referencia se presentan dos necesidades: establecer comunicación entre los dispositivos y con la PC así como repartir tareas ente ellos.

Para resolver ambos problemas de manera sencilla es posible utilizar un PIC maestro, el cual controle las comunicaciones y PIC esclavos que llevan la carga de interactuar con los motores y sus respectivos encoders. De este modo en la Figura 7.3 se muestra un esquema del modelo diseñado.

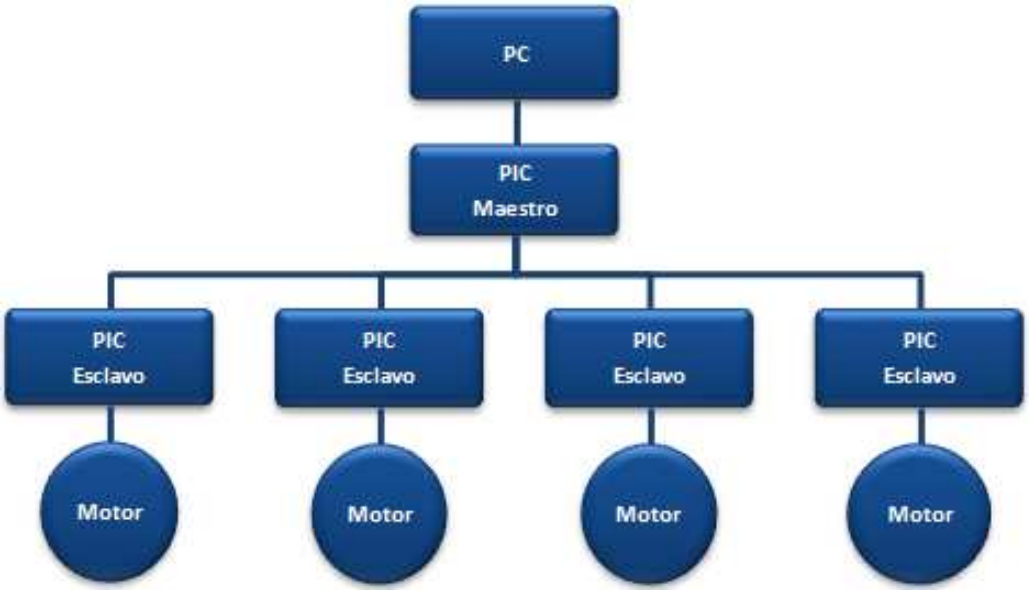


Figura 7.3. Diagrama de configuración para el control de motores.

Una vez establecido el esquema, se observa que la comunicación entre el PIC maestro y los esclavos, dado que son varios es posible hacerla usando comunicación I^2C, estableciendo una dirección a cada PIC, de esta forma si se quieren agregar más PIC bastará con asignarles una dirección y programar su código en el maestro. Ahora bien para la comunicación del maestro con la PC se usa una comunicación serial.

Para la programación de las labores a desarrollar por cada PIC esclavo se seguirá el esquema de la Figura 7.4.

Figura 7.4. Diagrama de pasos de PIC esclavo.

Inicialmente en la función Main se llama la subfunción leer la cual sirve para recibir datos como se muestra en la Figura 7.5.

```
void leer(void){            //función para leer datos por I2C como esclavo
   do{
      valor2=0;     //variable para mantener ciclo while hasta tener datos
      if(i2c_poll()==TRUE){//espera hasta recibir datos
      dato=i2c_read();      //leer datos recibidos
      if(dato==0xA2){       //comparar datos con direccion del pic
      dato=i2c_read();      //leer dato y guardarloen variable
      valor2=1;             //variable para terminar ciclo
      }
   }
   }while(valor2!=1);
```

Figura 7.5. Función Leer en lenguaje C para PIC Compiler.

Se observa que una vez llamada la función leer se mantendrá haciendo un ciclo hasta que reciba los datos.

Posteriormente el dato recibido será almacenado en la memoria EEPROM, esto se hará para cada uno de los 3 datos necesarios para el movimiento del motor, los cuales son velocidad, sentido de giro y número de cuentas a moverse.

```
do
{
  do{
     delay_ms(050);
     leer();                    //leer dato de velocidad
     write_eeprom(0x00,dato); //escribir en EEPROM
     leer();                    //leer dato de sentido
     write_eeprom(0x01,dato); //escribir en EEPROM
     leer();                    //leer dato de cuenta
     write_eeprom(0x02,dato); //escibir dato en EEPROM
   }while(valor2!=1);          //hacer ciclo mientras valor2 sea dif de
```

Figura 7.6. Código para escribir en memoria E^2PROM

Una vez que se tienen los datos serán cargados en variables y se analizan para establecer el sentido de giro como se muestra en la Figura 7.7.

```
valor2=0;                    //reseteo de variable para mantener ciclo
Vel=read_eeprom(0x00);       //leer velocidad de la EEPROM
Sen=read_eeprom(0x01);       //leer sentido de la EEPROM
Cuenta=read_eeprom(0x02);    //leer cuenta de la EEPROM

if(Sen=='R')           //comparacion del sentido de giro
{                      //si se recibe R
MoverDer();            //habilitar funcion de giro a la derecha
}
else if(Sen=='L')      //si se recibe L
{
MoverIzq();            //habilitar giro a la izquierda
}
else if(Sen=='S')      //si se recibe S
{
Stop();                //habilitar frenado rapido del motor
}
else                   //en caso de no recibir alguna de estas se detendra
{                      //al motor
Stop();                //habilitar frenado rapido del motor
}
```

Figura 7.7.Análisis de variables para establecer sentido de giro.

Aquí se observa que se está haciendo uso de subfunciones MoverDer(), MoverIzq() y Stop(), en las cuales se habilitan los pines para que el puente H haga girar al motor en cierto sentido, dichas subfunciones se pueden ver en la Figura 7.8.

```
void MoverDer()       //funcion para giro a la derecha
{
bit_set(portb,6);     //poner en 1 pin 6 del puertoB
bit_clear(portb,7);   //poner en 0 pin 7 del puerto B
}

void MoverIzq()       //funcion para giro a la izquierda
{
bit_clear(portb,6);   //poner en 0 pin 6 del puerto B
bit_set(portb,7);     //poner en 1 pin 7 del puerto B
}

void Stop()           //funcion para detener giro con frenado rapido.
{
bit_set(portb,6);     //poner en 1 pin 6 del puerto B
bit_set(portb,7);     //poner en 1 pin 7 del puerto B
}
```

Figura 7.8. Funciones para giro de motor.

Para establecer la velocidad de giro se envía una señal PWM al pin ENABLE del puente H, para la configuración de esta señal utiliza la subfunción PWM que se muestra en la Figura 7.9, en la cual después de configurar la señal se establece el ciclo de trabajo, de este modo el puente H tiene todas las señales necesarias para mover el motor.

```
void PWM()                //funcion para crear PWM
{
setup_timer_2(T2_DIV_BY_16,255,1);  //configura timer2
setup_ccp1(CCP_PWM);                //configura modulo CCP como PWM
set_pwm1_duty(Vel);                 //configura ciclo de trabajo
}                                   //con variable Vel
```

Figura 7.9. Configuración de la subfunción PWM.

Para el conteo de los pulsos, como se mencionó anteriormente se usan 2 timers, en este caso será el TIMER0 y el TIMER1, se usa la subfunción Contar() la cual tiene la codificación que se muestra en la Figura 7.10.

```
void Contar()         //funcion para contar pulsos de encoder
{
x=get_timer0();       //obtener numero de cuentas del timer0
y=get_timer1();       //obtener numero de cuentas del timer1
if(y==128)            //comparacion para limpiar el timer1 una
{                     //vez que ha llegado a 128 pulsos
set_timer1(0);        //pues el timer0 ya llevara una cuenta mas
}
write_eeprom(0x02,x);//escritura de dato de timer0 en dir 02 de EEPROM
write_eeprom(0x03,y);//escritura de dato de timer1 en dir 03 de EEPROM
}
```

Figura 7.10. Lectura de Timer asociados al encoder.

En este punto hace falta hacer la comparación entre las cuentas que se llevan y las cuentas que se recibieron como dato, dadas las limitantes del software de simulación esto no se llevó a cabo, dejando el conteo de pulsos como una teoría de su configuración y uso.

En el caso del PIC maestro el esquema que se usará será el que se muestra en la Figura 7.11.

Figura 7.11 Diagrama de funciones del PIC maestro.

La recepción de los datos en el PIC maestro, se hará por medio de comunicación serie, recibiendo como datos: el PIC que se quiere configurar para abrir comunicación con él, sentido de giro y la velocidad de giro. El PIC maestro se mantendrá a la espera de la tercia de datos como se muestra en la Figura 7.12.

```
if(kbhit())             //espera hasta que reciba datos
{                       //por puerto serie
  valor=fgetc(COM_PC);  //recepcion de dato de pic esclavo
  sentido=fgetc(COM_PC);//recepcion dato de sentido degiro
  Vel=fgetc(COM_PC);    //recepcion de dato de velocidad
```

Figura 7.12. Comunicación por puerto Serie del PIC maestro.

Posteriormente se hará un comparativo para saber a qué PIC enviar la información recibida, como se muestra en la Figura 7.13.

```
      if(valor=='A'){       //comparacion de valor con PIC A
      Slave(0xA2);          //escritura de datos a esclavo A2
      fprintf(COM_PC,"PIC A"); //enviar por puerto serie mensaje del pic
      }                        //esclavo usado
      else if(valor=='B'){  //comparacion de valor con PIC B
   Slave(0xA4);             //escritura de datos a esclavo A4
   fprintf (COM_PC,"PIC B");//mensaje por puerto serie del pic usado
}
   else if(valor=='C'){     //comparacion de valor con PIC C
   Slave(0xA6);             //escritura de datos a esclavo A6
   fprintf (COM_PC,"PIC C");//mensaje por puerto serie del pic usado
}
   else if(valor=='D'){   //comparacion de valor con PIC
   Slave(0xA8);             //escritura de datos a esclavo A6
   fprintf (COM_PC,"PIC D");//mensaje por puerto serie del pic usado
}
   else {                   //en caso de que no sea ningun valor
   putc(valor);             //regresa por puerto serie el dato escrito
   fprintf (COM_PC,"NINGUNO");//e imprime que no se envio a ningun PIC
  }
```

Figura 7.13. Comparativo de datos recibidos en PIC maestro y envío a PIC esclavo.

Aquí se puede observar que se hace uso de una subfunción llamada Slave(dir),donde *dir* es la dirección de cada PIC esclavo, de este modo dentro de la subfunción como se aprecia en la Figura 7.14, se envían los datos en el orden que la programación del PIC esclavo los requiere, es decir la velocidad establecida, el sentido de giro y por último el número de cuentas.

```
void Slave(PICS)           //Funcion para comunicacion
{                          //de datos con PIC esclavo
do{                        //seleccionado.
   delay_ms(250);
   escribir(Vel,PICS);     //Uso de funciòn escribir para enviar Velocidad
   if(sentido=='L')        //comparaciòn de variable sentido para
   {                       //no enviar dato incorrecto de sentido.
   escribir(sentido,PICS); //Funciòn escribir para enviar sentido en este
   }                       //caso "L"
   else if(sentido=='R')   //compara sentido con letra R
   {
   escribir(sentido,PICS); //Funciòn escribir para enviar sentido en este
   }                       //caso "R"
   else                    //Si la variable sentido no recibio una L o R
   {                       //el sentido sera S para detener al motor.
   sentido='S';            //Sentido se establece como S
   escribir(sentido,PICS); //Funcion escribir para enviar sentido en este
   }                       //caso "S"
   escribir(cuenta,PICS);  //Funcion escribir para envio de cuentas a moverse
   valor=1;                //variable para terminar ciclo while
   }while(valor!=1);
}
```

Figura 7.14. Subfunción de PIC maestro para envió de datos a PIC esclavo.

Al igual que en los PIC esclavos se tiene una subfunción que escribe datos mediante I2C dicha codificación se puede observar en la Figura 7.15.

```
void escribir(dato,direccion){//Funciòn para esritura de datos por I2C
        delay_ms(50);          //retrasos para recepcion de datos
        i2c_start();           //Inicia transmisiòn de datos
        delay_ms(50);
        i2c_write(direccion); //Envia variable direccion
        delay_ms(50);
        i2c_write(dato);       //Envia variable dato
        delay_ms(50);
        delay_ms(50);
        i2c_stop();            //Detiene comincaciòn
        delay_ms(50);
        valor1=1;              //variable para terminar ciclo while
}
```

Figura 7.15. Subfunción para envío de datos usando I²C

Después de enviar los datos al PIC esclavo, regresará por el puerto serie el nombre de PIC al que se enviaron los datos.

7.1.3 SIMULACIÓN

Una vez que se tienen los códigos, se construirá el modelo en PROTEUS para su simulación. Para ello se hará uso de cuatro PIC esclavos, cada uno acoplado a un motor mediante un puente H L293D; también irán conectados a los cables que son usados para la comunicación I2C con el PIC maestro, el cual tendrá acoplado un puerto serie virtual por el cual recibirá los datos desde la PC. En la Figura 7.16 se puede apreciar el diseño final del prototipo de simulación.

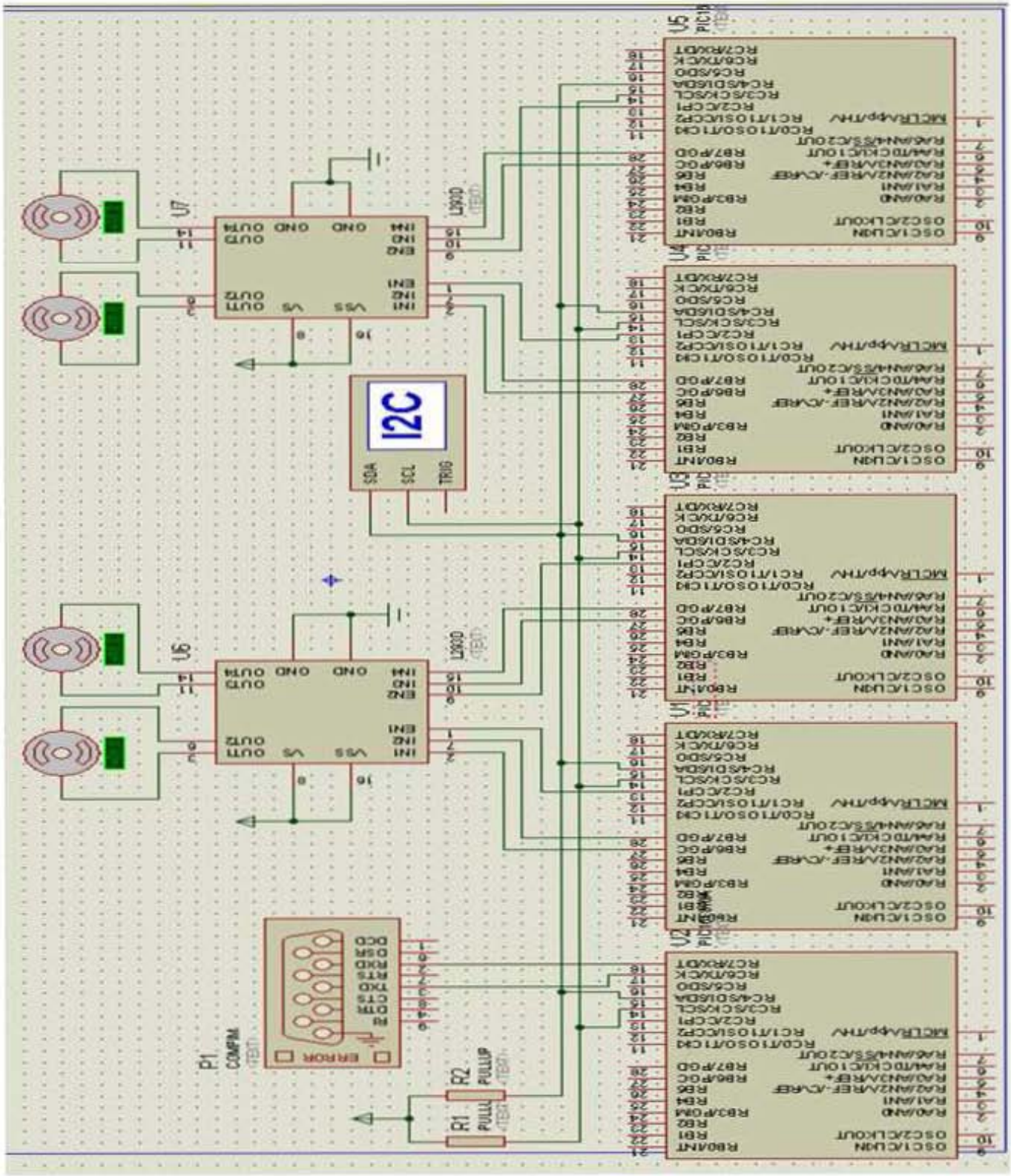


Figura 7.16. Diseño para simulación en PROTEUS

7.1.4 CREACIÓN DE UNA INTERFAZ

Para el envío de datos desde la PC, se desarrolló una interfaz en la cual fuese más sencilla la creación de botones de control para cada motor y establecer la comunicación serial para el envió de la tercia de datos necesarios en el PIC maestro, para esta tarea se creó un panel en Visual Basic 6.0, quedando como se muestra en la Figura 7.17.

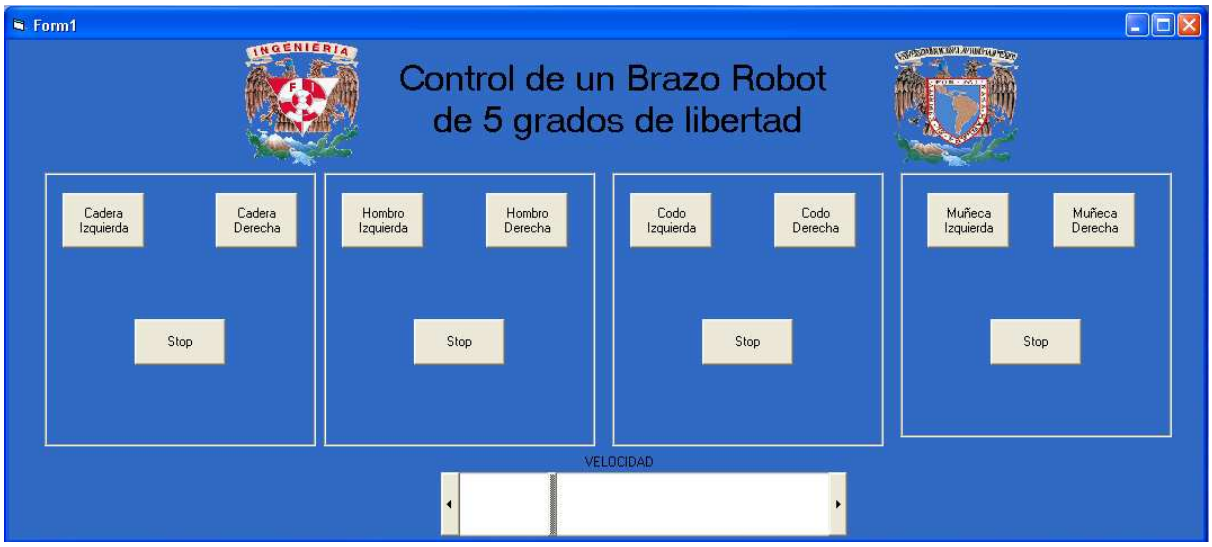


Figura 7.17. Panel de Visual Basic para control de motores.

En este panel se dividieron los botones por cada motor que será manipulado. Existen dos botones asociados al sentido de giro de cada motor, así como un botón de paro, en la parte inferior se tiene un scroll el cual nos proporciona el dato de la velocidad. Cada botón tendrá 3 parámetros a enviar por el puerto serial: una letra que lo identifique como articulación, otra que indique el sentido de giro y el valor de velocidad. En la siguiente tabla se muestran los datos que envía cada botón.

Tabla 7.1. Datos enviados por cada botón del panel de Visual Basic

Botón	Dirección PIC	Sentido	Velocidad
Cadera izquierda	A	L	Obtenida del Scroll
Cadera Derecha	A	R	
Hombro izquierda	B	L	
Hombro derecha	B	R	
Codo izquierda	C	L	
Codo derecha	C	R	
Muñeca izquierda	D	L	
Muñeca derecha	D	R	

Para realizar la comunicación es necesario configurar el comando MSCOMM el cual habilita el puerto, para ello se usan parámetros estándar de comunicación serial. Como se muestra en la siguiente figura.

```
Private Sub Form_Load()      'Configuracion del puerto de comunicaciones
With MSComm1                 'como MsComm1
.CommPort = 3                'puerto serie com3
.RThreshold = 1              'espera 1 dato en buffer de entrada para analizarlo
.RTSEnable = True            'peticion de envio habilitada
.Settings = "9600,n,8,1"     'velocidad, paridad,bits de informacion,bit de parada
.SThreshold = 1              'espera 1 dato en buffer de salida para analizarlo
.PortOpen = True             'habilitar puerto
End With
End Sub
```

Así, por ejemplo, si se presiona el botón *Cadera Izquierda*, se envía primero una A, posteriormente una L y por último el valor del scroll como se muestra en el siguiente código.

```
Private Sub Cad_Der_Click(Index As Integer) 'Funcion del Boton Cadera Derecha al hacer clic
Dim Letra As String                         'Variable "Letra" como cadena
Dim V As String                             'Variable "V" como cadena
Letra = "A"                                 'Variable letra se le asigna valor de A
MSComm1.Output = Letra                      'Envia por puerto de comunicacion la variable Letra
Letra = "R"                                 'Variable letra se le asigna valor de R
MSComm1.Output = Letra                      'Envia por puerto de comunicacion la variable Letra
V = Chr(HScroll1.Value)                     'variable V con valor del scroll convertido a cadena
MSComm1.Output = V                          'Envia por puerto de comunicacion la variable V
End Sub
```

7.1.5 COMPARATIVO ENTRE PLC Y PIC

Es posible implementar un dispositivo Programable de Control similar a un PLC basado en un microcontrolador PIC cualquiera, lo cual hace pensar en las ventajas y desventajas que se pueden tener al usar uno u otro.

De principio se sabe que el precio de los PIC es notablemente menor al de un PLC de cualquier marca. No obstante utilizar un PLC en un ambiente industrial puede tener muchas ventajas de seguridad y confiabilidad sobre un microcontrolador.

Es por ello que se pueden analizar algunas formas de disminuir las desventajas que presenta el de menor costo utilizando algunos circuitos básicos como acoplamiento entre la planta y el controlador.Se pueden utilizar opto acopladores en las entradas del PIC para lectura de señales todo o nada (TTL) y protegerlas de picos superiores a los 5 volts. El consumo de estos circuitos es mínimo así que puede con alimentarse la misma fuente que el microcontrolador.

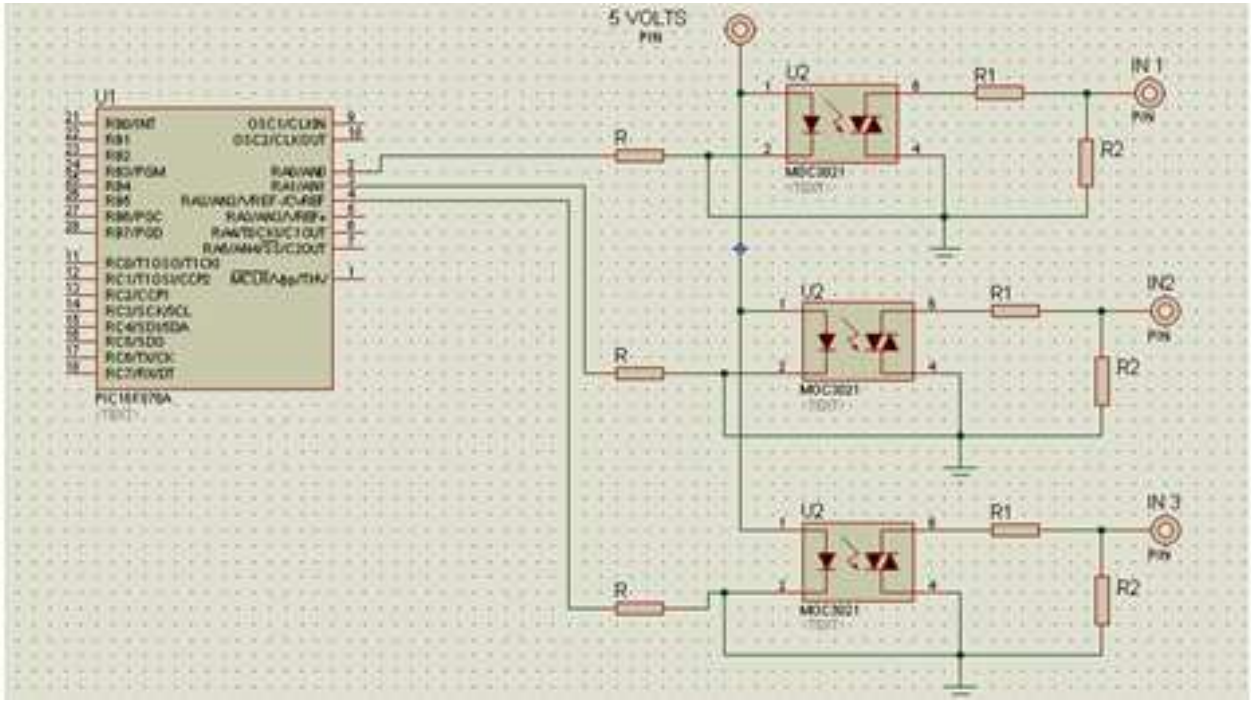


Figura 7.18. Entradas protegidas con optoacoplador MOC3021

De esta forma se puede complementar el circuito que se utilizó para la simulación, ya que no tiene contemplados los microswitches, que deberían estar conectados a las entradas digitales.

El controlador original del Scorbot tiene algunas salidas extras destinadas a controlar algunos elementos de una línea de producción como bandas y sensores. Este tipo de salidas a relevador son muy comunes en los PLC, lo cual es una ventaja para este proyecto, pero es posible utilizar un circuito integrado con un arreglo de transistores Darlington para controlar relevadores con un PIC. La función del circuito integrado es amplificar la corriente para lograr excitar las bobinas de los relevadores.

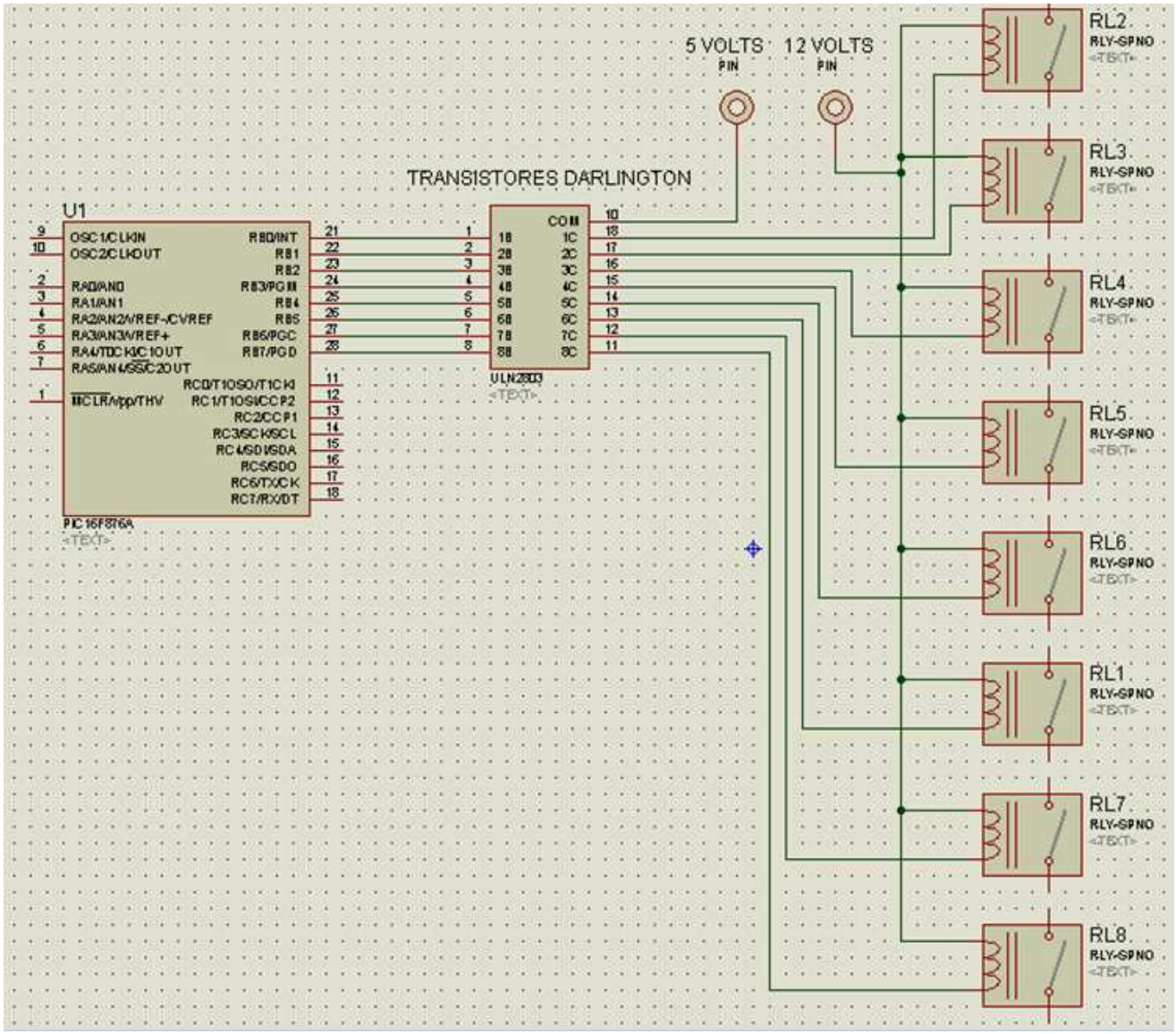


Figura 7.19. Salidas a relevador utilizando circuito ULN2803

Se puede notar el que PIC tiene muchas desventajas frente al PLC para aplicaciones donde se requieren numerosas entradas y salidas, y donde el ambiente industrial puede afectar la confiabilidad de las lecturas. Esto se debe a que algunos PLC son modulares, por lo cual incrementar el número de entradas y salidas a transistor o a relevador es muy sencillo.

Otro factor de gran importancia a favor para algunos PLC es la comunicación. Como se observó durante el desarrollo del proyecto, al tener que utilizar más de dos PIC para la simulación se hizo necesario implementar dos tipos de comunicaciones: una entre los dispositivos y otra diferente con la PC.

8 CONCLUSIONES

En general se puede observar que el reemplazo del controlador mediante el PLC es viable. Si bien en el presente trabajo solo se tiene una base de lo que se requiere para cumplir con todas las capacidades del dispositivo que lo controla originalmente ya que este cuenta con un controlador en lazo cerrado en forma de comparador el cual controla a través de los encoders el número de cuentas que se mueve, la velocidad y dirección, debido a esto, es necesario agregar algún tipo de controlador que realice las mismas funciones que el original. La interfaz electrónica fue pensada para ser fácilmente modificable potencializando el uso de un PLC modular, ya que el PLC S7- 300 permite usar módulos que facilitan la aplicación de controladores PID a motores de corriente directa como son: los de salidas analógicas 0 - 24 Volts y otros de conteo rápido con el cual en las entradas se puede obtener la lectura de encoders con altas velocidades y las salidas pueden ser usadas para manejar señales PWM.

Actualmente al no contar con algún controlador que regule el voltaje enviado a los motores, la alimentación a estos se detiene al apagar las salidas del PLC y por lo tanto el propio peso del brazo propicia que no se puedan mantener ciertas posiciones. Al contar con un controlador y un contador de pulsos adecuado para cada encoder del robot, se podrían implementar las rutinas de posicionamiento automático a Home o a cualquier punto en el área de trabajo y seguimiento de trayectorias.

Bajo el esquema de subfunciones que se utilizó en la programación y sabiendo de antemano que existe un toolkit de control PID de LabVIEW, es fácilmente integrable un subVI con un controlador y un subVI que manipule otros dispositivos de automatización como pueden ser bandas, pistones, u otros sensores y así ir creando un sistema de automatización.

Ahora bien en el comparativo entre el PLC y los microcontroladores, se puede iniciar con la parte de comunicaciones y programación de ambos dispositivos. En el caso del PLC, la conexión vía Ethernet con la PC proporciona una manera muy eficaz de comunicación tanto por las velocidades de transmisión así como por las distancias que puede recorrer sin perder datos (dependiendo de la tecnología usada pueden ser 100 metros o hasta 5 km como máximo) o bien utilizar antenas inalámbricas y tener la posibilidad de programar o monitorear las variables del PLC sin tener que estar frente al dispositivo. En cambio, los microcontroladores al tener una comunicación serial tienen una relación entre su velocidad de transmisión y la distancia máxima a la que se puede transmitir, además en este caso es necesario retirar el microcontrolador para poder reprogramarlo y posteriormente volver a integrarlo al circuito.

En el ámbito de las operaciones aritméticas si bien ni el PLC ni un PIC están dotados de muchas funciones matemáticas, éstas pueden ser programadas y realizadas en LabVIEW y sólo enviar instrucciones de movimiento al PLC, mientras que para el microcontrolador y su interfaz en Visual Basic la programación del cálculo de matrices no es tan sencillo como en LabVIEW, este punto es de suma importancia ya que en este caso solamente se calculan matrices y operaciones básicas, pero una vez que se implemente un controlador el cual requiera operaciones más complejas como derivar o integrar en caso de un controlador PID, éstas tendrían que ser implementadas bajo métodos numéricos lo cual conllevaría tener que usar más microcontroladores que realicen dichos cálculos.

La interfaz gráfica creada en el caso de Visual Basic o de la interfaz HMI permite tener los botones para el movimiento de cada motor en ambos sentidos y de esta forma mover el brazo robot. En el caso de LabVIEW además de tener los controles permite tener una imagen tridimensional del brazo con lo cual es posible apreciar la posición en la que se encuentra cada una de las articulaciones así como el efector final al manipular los controles, el cual puede ser usado sin la necesidad de tener el robot conectado, lo cual da la opción de poder usarlo como simulación, y no solamente con el control manual, sino también con la cinemática directa, inversa y el seguimiento de trayectorias.

De esta forma es posible darle un enfoque académico dando la posibilidad de tener un programa que permita interactuar con los movimientos del robot y familiarizarse con sus limitantes de movimiento y su espacio de trabajo, así como poder tener un mejor entendimiento de la teoría de control como es la cinemática directa y la cinemática inversa.

Para el caso en que la simulación 3D trabaja junto con el control manual y el brazo conectado, se tienen las limitantes de la tarjeta de National Instruments en cuanto al conteo de los pulsos generados por los encoders y a la ausencia del controlador y debido a ello no se pudo ligar de forma exacta el movimiento real del robot con el mostrado en la imagen tridimensional del mismo.

Para este punto también se previó utilizar Solid Edge el cual es un software de diseño para modelado de piezas mecánicas para realizar una imagen tridimensional idéntica al Scorbot no solamente en cuanto al aspecto físico sino también de sus partes mecánicas y ligarlo a LabVIEW para su animación y manipulación utilizando el panel de control que se implementó, lo cual finalmente no se llevó a cabo dada su complejidad dejándolo como un posible proyecto realizable a futuro.

Otra posibilidad que se tiene, utilizando LabVIEW, es la de manipular el panel de control remotamente utilizando un navegador de internet. Para ello, sólo es necesario conocer la dirección IP de la PC que esté controlando al brazo robot y que ésta tenga el proyecto de LabVIEW funcionando, se accede mediante el navegador y se toma control del panel permitiendo así al usuario mover el Scorbot desde cualquier sitio y en cualquier PC con acceso a internet. Para esto se utilizaría una cámara web que esté transmitiendo los movimientos que realiza el brazo robot para comprobar su funcionamiento.

Por último los manuales escritos se prevé lleven a que otros alumnos desarrollen proyectos utilizando un PLC Siemens S7-300 o una interfaz HMI, sin tener que tomar de cero la investigación de cómo empezar a utilizar dichos dispositivos, lo cual ya pudo ser observado, pues se colaboró proporcionando una copia del manual del uso de la HMI para el desarrollo de otro proyecto con lo cual se logró que se enfocaran en la parte del desarrollo de la aplicación que se requería.

9 BIBLIOGRAFÍA

Angulo Usategui, J. M. (2003). *Microcontroladores PIC diseño práctico de aplicaciones.* Madrid: MCGRAW-HILL .

Barr, M. (2001). Pulse Width Modulation. *Embedded Systems Programming* , 103-104.

Barrientos, A. (2007). *Fundamentos de Robótica.* Madrid: McGraw-Hill.

Garcia Breijo, E. (2008). *Compilador C CCS y Simulador PROTEUS para Microcontroladores PIC.* Barcelona: MARCOMBO.

Intelitek. (2003). *Scorbot-ER 5Plus.* Manchester.

Iovine, J. (2004). *PIC Robotics: A Beginner's Guide to Robotics Projects Using the PIC Micro.* Mc Graw Hill.

Lajara Vizcaino, J. R. (2007). *LabVIEW entorno grafico de programación.* D.F.: Alfaomega.

Mandado Pérez, E. (2005). *Autómatas Programables: Entorno y Aplicaciones.* Madrid: Thomson.

Microchip Techonolgy Inc. (2005). *PIC 16F87/88 Data Sheet.* Estados Unidos.

National Instruments. (1998). *LabVIEW User Manual.* Austin,Texas: National Instruments.

SIEMENS A&D. (2002). *Automatizacion Totalmente Integrada. Manual de Formacion.* Munich.

SIEMENS AG. (2007). *SIMATIC WinCC felxible.* Nuernberg.

Texas Instrument Inc. (2002). *L293, L293D Quadruple Half-H Drivers.* Dallas.

10 APÉNDICES

10.1 APÉNDICE A. CÓDIGO PARA SIMULACIÓN MEDIANTE PIC

10.1.1 CÓDIGO PIC MAESTRO

```
#include<16f876A.h>
#use delay(clock=20000000)
#FUSES XT,NOWDT,NOPROTECT,NOLVP,PUT,BROWNOUT
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7,STREAM=COM_PC)
#use I2C(master, sda=PIN_C4, scl=PIN_C3)

int dato=0;
int valor=0;
int valor1=0;
int direccion=0;
int sentido;
int cuenta=1;
int Vel=0;
int valor2=0;
char pic;
int PICS;

void leer(dirección){                    //Función para leer datos
  do{                                    //mediante I2C
    i2c_start();                         //Inicia transmisión de datos
    i2c_write(dirección);                //Envío de variable dirección
    dato=i2c_read(0);                    //variable dato recibe lectura
    i2c_stop();                          //Detiene la transmisión
    putc(dato);                          //por RS232 se envía dato recibido
    valor1=1;                            //variable para terminar ciclo while

  }while(valor1!=1);
}

void escribir(dato,direccion){           //Función para escritura de datos por I2C
      delay_ms(50);                      //retrasos para recepción de datos
      i2c_start();                       //Inicia transmisión de datos
      delay_ms(50);
      i2c_write(dirección);              //Envía variable dirección
      delay_ms(50);
      i2c_write(dato);                   //Envía variable dato
      delay_ms(50);
      delay_ms(50);
      i2c_stop();                        //Detiene comunicación
      delay_ms(50);
      valor1=1;                          //variable para terminar ciclo while
}

void Slave(PICS)                         //Función para comunicación
{                                        //de datos con PIC esclavo
```

```
do{                                  //seleccionado.
  delay_ms(250);
  escribir(Vel,PICS);                //Uso de función escribir para enviar Velocidad
  if(sentido=='L')                   //comparación de variable sentido para
  {                                  //no enviar dato incorrecto de sentido.
  escribir(sentido,PICS);            //Función escribir para enviar sentido en este
  }                                  //caso "L"
  else if(sentido=='R')              //compara sentido con letra R
  {
  escribir(sentido,PICS);            //Función escribir para enviar sentido en este
  }                                  //caso "R"
  else                               //Si la variable sentido no recibió una L o R
  {                                  //el sentido será S para detener al motor.
  sentido='S';                       //Sentido se establece como S
  escribir(sentido,PICS);            //Función escribir para enviar sentido en este
  }                                   //caso "S"
  escribir(cuenta,PICS);             //Función escribir para envío de cuentas a moverse
  valor=1;                           //variable para terminar ciclo while
  }while(valor!=1);
}
void main(){                         //Función principal la cual tiene un ciclo
do{                                  //infinito

    if(kbhit())                      //espera hasta que reciba datos
    {                                //por puerto serie
      valor=fgetc(COM_PC);           //recepción de dato de PIC esclavo
      sentido=fgetc(COM_PC);         //recepción dato de sentido de giro
      Vel=fgetc(COM_PC);              //recepción de dato de velocidad

      if(valor=='A'){                //comparación de valor con PIC A
      Slave(0xA2);                   //escritura de datos a esclavo A2
      fprintf(COM_PC,"PIC A");       //enviar por puerto serie mensaje del PIC
      }                               //esclavo usado
      else if(valor=='B'){           //comparación de valor con PIC B
    Slave(0xA4);                     //escritura de datos a esclavo A4
    fprintf (COM_PC,"PIC B");        //mensaje por puerto serie del PIC usado
  }
    else if(valor=='C'){             //comparación de valor con PIC C
    Slave(0xA6);                     //escritura de datos a esclavo A6
    fprintf (COM_PC,"PIC C");        //mensaje por puerto serie del PIC usado
  }
    else if(valor=='D'){             //comparación de valor con PIC
    Slave(0xA8);                     //escritura de datos a esclavo A6
    fprintf (COM_PC,"PIC D");        //mensaje por puerto serie del PIC usado
  }
    else {                           //en caso de que no sea ningún valor
```

10.1.2 CÓDIGO PIC ESCLAVO

```
#include<16f876A.h>
#use delay(clock=20000000)
#FUSES XT,NOWDT,NOPROTECT,NOLVP,PUT,BROWNOUT
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
#use I2C(slave, sda=PIN_C4, scl=PIN_C3,address=0xA2,FORCE_HW)
#byte porta=5
#byte portb=6
#byte portc=7

int dato=0;
int dato1=0;
int valor2=0;
int state=0;
int Vel=0;
int Sen=0;
int x=0;
int y=0;
int valor=0;
int Cuenta=0;



void leer(void){                        //función para leer datos por I2C como esclavo
  do{
    valor2=0;                           //variable para mantener ciclo while hasta tener datos
    if(i2c_poll()==TRUE){               //espera hasta recibir datos
    dato=i2c_read();                    //leer datos recibidos
    if(dato==0xA2){                     //comparar datos con dirección del PIC
    dato=i2c_read();                    //leer dato y guardarlo en variable
    valor2=1;                           //variable para terminar ciclo
    }
  }
  }while(valor2!=1);
}

void escribir(dato1){                   //función para escribir datos a PIC maestro
   do{
    valor2=0;                           //variable para mantener ciclo while hasta escribir datos
    i2c_write(dato1);                   //escribir dato
    i2c_write(dato1);                   //reescribir dato
    fputc(dato1);                       //mandar por puerto serie el dato enviado por I2C
    valor2=1;                           //variable para terminar ciclo while
  }while(valor2!=1);
}

void MoverDer()                         //función para giro a la derecha
{
bit_set(portb,6);                       //poner en 1 pin 6 del puerto B
bit_clear(portb,7);                     //poner en 0 pin 7 del puerto B
}
```

```
void MoverIzq()                         //función para giro a la izquierda
{
bit_clear(portb,6);                     //poner en 0 pin 6 del puerto B
bit_set(portb,7);                       //poner en 1 pin 7 del puerto B
}

void Stop()                             //función para detener giro con frenado rápido.
{
bit_set(portb,6);                       //poner en 1 pin 6 del puerto B
bit_set(portb,7);                       //poner en 1 pin 7 del puerto B
}

void Contar()                           //función para contar pulsos de encoder
{
x=get_timer0();                         //obtener numero de cuentas del timer0
y=get_timer1();                         //obtener numero de cuentas del timer1
if(y==128)                              //comparación para limpiar el timer1 una
{                                       //vez que ha llegado a 128 pulsos
set_timer1(0);                          //pues el timer0 ya llevara una cuenta mas
}
write_eeprom(0x02,x);                   //escritura de dato de timer0 en dir 02 de EEPROM
write_eeprom(0x03,y);                   //escritura de dato de timer1 en dir 03 de EEPROM
}

void Guardar()                          //función para guardar datos en EEPROM
{
write_eeprom(0x00,Vel);                 //escritura de dato de velocidad en dir 00 de EEPROM
write_eeprom(0x01,Sen);                 //escritura de dato de sentido en dir 01 de EEPROM
}

void PWM()                              //función para crear PWM
{
setup_timer_2(T2_DIV_BY_16,255,1);      //configura timer2
setup_ccp1(CCP_PWM);                    //configura modulo CCP como PWM
set_pwm1_duty(Vel);                     //configura ciclo de trabajo
}                                       //con variable Vel

void main(){

SET_TRIS_A( 0xFF );                     //Puerto A como entradas
SET_TRIS_B( 0x00 );                      //Puerto B como salidas
//SET_TRIS_C( 0xBB );                   //Puerto C como salidas
setup_timer_0 (RTCC_DIV_256|RTCC_EXT_L_TO_H);          //configurar timer0
setup_timer_1 ( T1_EXTERNAL | T1_DIV_BY_2 );           //configurar timer1
set_timer0(0);                          //resetear timer0
set_timer1(0);                           //resetear timer1
portb=0x00;                             //limpiar puerto B
delay_ms(050);                           //retraso de 50 ms

do
{
  do{
    delay_ms(050);
```

```
   leer();                          //leer dato de velocidad
   write_eeprom(0x00,dato);         //escribir en EEPROM
   leer();                          //leer dato de sentido
   write_eeprom(0x01,dato);         //escribir en EEPROM
   leer();                           //leer dato de cuenta
   write_eeprom(0x02,dato);         //escribir dato en EEPROM
  }while(valor2!=1);                //hacer ciclo mientras valor2 sea diferente de 1
 valor2=0;                          //reseteo de variable para mantener ciclo
 Vel=read_eeprom(0x00);             //leer velocidad de la EEPROM
 Sen=read_eeprom(0x01);             //leer sentido de la EEPROM
 Cuenta=read_eeprom(0x02);          //leer cuenta de la EEPROM

 if(Sen=='R')                       //comparación del sentido de giro
 {                                  //si se recibe R
 MoverDer();                        //habilitar función de giro a la derecha
 }
 else if(Sen=='L')                  //si se recibe L
 {
 MoverIzq();                        //habilitar giro a la izquierda
 }
 else if(Sen=='S')                  //si se recibe S
 {
 Stop();                            //habilitar frenado rápido del motor
 }
 else                               //en caso de no recibir alguna de estas se detendrá
 {                                   //al motor
 Stop();                             //habilitar frenado rápido del motor
 }
 PWM();                              //habilitar función para salida de PWM
 Contar();                          //llamado de la función contar
 x=read_eeprom(0x02);                //lectura de la EEPROM para timer0
 y=read_eeprom(0x03);               //lectura de la EEPROM para timer1
}while (1);                         //ciclo infinito
}
```

10.1.3 CÓDIGO DE PANEL EN VISUAL BASIC

```
Private Sub Cad_Der_Click(Index As Integer)          ‘Función del Botón Cadera Derecha al hacer clic
Dim Letra As String                                  'Variable "Letra" como cadena
Dim V As String                                      'Variable "V" como cadena
Letra = "A"                                          'Variable letra se le asigna valor de A
MSComm1.Output = Letra                               'Envía por puerto de comunicación la variable Letra
Letra = "R"                                          'Variable letra se le asigna valor de R
MSComm1.Output = Letra                               'Envía por puerto de comunicación la variable Letra
V = Chr(HScroll1.Value)                              'variable V con valor del scroll convertido a cadena
MSComm1.Output = V                                   'Envía por puerto de comunicación la variable V
End Sub

Private Sub Cad_Izq_Click(Index As Integer)          Función del Botón Cadera Izquierda al hacer clic
Dim Letra As String                                  'Variable "Letra" como cadena
Dim V As String                                      'Variable "V" como cadena
Letra = "A"                                          'Variable letra se le asigna valor de A
MSComm1.Output = Letra                               'Envía por puerto de comunicación la variable Letra
Letra = "L"                                          'Variable letra se le asigna valor de L
MSComm1.Output = Letra                               'Envía por puerto de comunicación la variable Letra
V = Chr(HScroll1.Value)                              'variable V con valor del scroll convertido a cadena
MSComm1.Output = V                                   'Envía por puerto de comunicación la variable V
End Sub

Private Sub Cod_Der_Click(Index As Integer)          ‘Función del Botón Codo Derecha al hacer clic
Dim Letra As String                                  'Variable "Letra" como cadena
Dim V As String                                      'Variable "V" como cadena
Letra = "C"                                          'Variable letra se le asigna valor de C
MSComm1.Output = Letra                               'Envía por puerto de comunicación la variable Letra
Letra = "R"                                          'Variable letra se le asigna valor de R
MSComm1.Output = Letra                               'Envía por puerto de comunicación la variable Letra
V = Chr(HScroll1.Value)                              'variable V con valor del scroll convertido a cadena
MSComm1.Output = V                                   'Envía por puerto de comunicación la variable V
End Sub

Private Sub Cod_Izq_Click(Index As Integer)          Función del Botón Codo Izquierda al hacer clic
Dim Letra As String                                  'Variable "Letra" como cadena
Dim V As String                                      'Variable "V" como cadena
Letra = "C"                                          'Variable letra se le asigna valor de C
MSComm1.Output = Letra                               'Envía por puerto de comunicación la variable Letra
Letra = "L"                                          'Variable letra se le asigna valor de L
MSComm1.Output = Letra                               'Envía por puerto de comunicación la variable Letra
V = Chr(HScroll1.Value)                              'variable V con valor del scroll convertido a cadena
MSComm1.Output = V                                   'Envía por puerto de comunicación la variable V
End Sub
```

```
Private Sub Hom_Izq_Click(Index As Integer)
Dim Letra As String
Dim V As String
Letra = "B"
MSComm1.Output = Letra
Letra = "L"
MSComm1.Output = Letra
V = Chr(HScroll1.Value)
MSComm1.Output = V
End Sub
```

```
'Función del Botón Hombro Izquierda al hacer clic
'Variable "Letra" como cadena
'Variable "V" como cadena
'Variable letra se le asigna valor de B
'Envía por puerto de comunicación la variable Letra
'Variable letra se le asigna valor de L
'Envía por puerto de comunicación la variable Letra
'variable V con valor del scroll convertido a cadena
'Envía por puerto de comunicación la variable V
```

```
Private Sub Hom_Der_Click(Index As Integer)
Dim Letra As String
Dim V As String
Letra = "B"
MSComm1.Output = Letra
Letra = "R"
MSComm1.Output = Letra
V = Chr(HScroll1.Value)
MSComm1.Output = V
End Sub
```

```
‘Función del Botón Hombro Derecha al hacer clic
'Variable "Letra" como cadena
'Variable "V" como cadena
'Variable letra se le asigna valor de B
'Envía por puerto de comunicación la variable Letra
'Variable letra se le asigna valor de R
'Envía por puerto de comunicación la variable Letra
'variable V con valor del scroll convertido a cadena
'Envía por puerto de comunicación la variable V
```

```
Private Sub Mun_Izq_Click(Index As Integer)
Dim Letra As String
Dim V As String
Letra = "D"
MSComm1.Output = Letra
Letra = "L"
MSComm1.Output = Letra
V = Chr(HScroll1.Value)
MSComm1.Output = V
End Sub
```

```
'Función del Botón Muñeca Izquierda al hacer clic
'Variable "Letra" como cadena
'Variable "V" como cadena
'Variable letra se le asigna valor de D
'Envía por puerto de comunicación la variable Letra
'Variable letra se le asigna valor de L
'Envía por puerto de comunicación la variable Letra
'variable V con valor del scroll convertido a cadena
'Envía por puerto de comunicación la variable V
```

```
Private Sub Mun_Der_Click(Index As Integer)
Dim Letra As String
Dim V As String
Letra = "D"
MSComm1.Output = Letra
Letra = "R"
MSComm1.Output = Letra
V = Chr(HScroll1.Value)
MSComm1.Output = V
End Sub
```

```
‘Función del Botón Muñeca Derecha al hacer clic
'Variable "Letra" como cadena
'Variable "V" como cadena
'Variable letra se le asigna valor de D
'Envía por puerto de comunicación la variable Letra
'Variable letra se le asigna valor de R
'Envía por puerto de comunicación la variable Letra
'variable V con valor del scroll convertido a cadena
'Envía por puerto de comunicación la variable V
```

```
Private Sub Stop_Cad_Click()
Dim Letra As String
Dim V As String
Letra = "A"
MSComm1.Output = Letra
Letra = "S"
MSComm1.Output = Letra
V = Chr(HScroll1.Value)
MSComm1.Output = V
End Sub
```

```
'Función de Stop Cadera
'Variable "Letra" como cadena
'Variable "V" como cadena
'Variable letra se le asigna valor de A
'Envía por puerto de comunicación la variable Letra
'Variable letra se le asigna valor de S
'Envía por puerto de comunicación la variable Letra
'variable V con valor del scroll convertido a cadena
'Envía por puerto de comunicación la variable V
```

```
Private Sub Stop_Hombro_Click(Index As Integer)
Dim Letra As String
Dim V As String
Letra = "B"
MSComm1.Output = Letra
Letra = "S"
MSComm1.Output = Letra
V = Chr(HScroll1.Value)
MSComm1.Output = V
End Sub
```

```
'Función de Stop Hombro
'Variable "Letra" como cadena
'Variable "V" como cadena
'Variable letra se le asigna valor de B
'Envía por puerto de comunicación la variable Letra
'Variable letra se le asigna valor de S
'Envía por puerto de comunicación la variable Letra
'variable V con valor del scroll convertido a cadena
'Envía por puerto de comunicación la variable V
```

```
Private Sub Stop_Codo_Click()
Dim Letra As String
Dim V As String
Letra = "C"
MSComm1.Output = Letra
Letra = "S"
MSComm1.Output = Letra
V = Chr(HScroll1.Value)
MSComm1.Output = V
End Sub
```

```
'Función de Stop Codo
'Variable "Letra" como cadena
'Variable "V" como cadena
'Variable letra se le asigna valor de C
'Envía por puerto de comunicación la variable Letra
'Variable letra se le asigna valor de S
'Envía por puerto de comunicación la variable Letra
'variable V con valor del scroll convertido a cadena
'Envía por puerto de comunicación la variable V
```

```
Private Sub Stop_Muñeca_Click()
Dim Letra As String
Dim V As String
Letra = "D"
MSComm1.Output = Letra
Letra = "S"
MSComm1.Output = Letra
V = Chr(HScroll1.Value)
MSComm1.Output = V
End Sub
```

```
'Función de Stop Muñeca
'Variable "Letra" como cadena
'Variable "V" como cadena
'Variable letra se le asigna valor de D
'Envía por puerto de comunicación la variable Letra
'Variable letra se le asigna valor de S
'Envía por puerto de comunicación la variable Letra
'variable V con valor del scroll convertido a cadena
'Envía por puerto de comunicación la variable V
```

```
Private Sub Form_Load()     With MSComm1                    'Configuración del puerto de comunicaciones como MsComm1

.CommPort = 3                                               'puerto serie com3
 .RThreshold = 1                                            'espera 1 dato en buffer de entrada para analizarlo
.RTSEnable = True                                           'petición de envío habilitada
 .Settings = "9600,n,8,1"                                   'velocidad, paridad, bits de información, bit de parada
.SThreshold = 1                                             'espera 1 dato en buffer de salida para analizarlo
.PortOpen = True                                            'Habilitar puerto
End With
End Sub


Private Sub Form_Unload(Cancel As Integer)                  'Cierra puerto de comunicación
MSComm1.PortOpen = False
End Sub
```