IMPLEMENTACION DE UNA INTERFAZ GRAFICA CON MATLAB PARA LA OPERACIÓN DE UN MODULO DE CONTROL DE POSICION O VELOCIDAD UTILIZANDO UN PLC S7-300 DE LA EMPRESA SIEMENS

HECTOR FERNANDO QUINTERO RAMIREZ

UNIVERSIDAD PONTIFICIA BOLIVARIANA
ESCUELA DE INGENIERÍA
FACULTAD DE INGENIERIA ELECTRONICA
BUCARAMANGA
2015

IMPLEMENTACION DE UNA INTERFAZ GRAFICA CON MATLAB PARA LA OPERACIÓN DE UN MODULO DE CONTROL DE POSICION O VELOCIDAD UTILIZANDO UN PLC S7-300 DE LA EMPRESA SIEMENS.

INVESTIGADOR:
HECTOR FERNANDO QUINTERO RAMIREZ

DIRECTOR DEL PROYECTO:
Ing. OMAR PINZÓN ARDILA, PhD

UNIVERSIDAD PONTIFICIA BOLIVARIANA
ESCUELA DE INGENIERÍA
FACULTAD DE INGENIERIA ELECTRONICA
BUCARAMANGA
2015

Nota de aceptación.

_____
_____
_____
_____
_____

_____
Presidente del Jurado

_____
Firma del Jurado

_____
Firma del Jurado

DEDICATORIA

Dedico este proyecto de tesis a mis padres quienes han sido pilar fundamental a lo largo de mi vida, velando siempre por mi bienestar y educación, siendo mi apoyo no solo económico sino emocional, dándome su amor incondicional en cada dificultad presentada a lo largo de mi proceso como profesional, confiando ciegamente en mis actitudes y habilidades para resolver cada obstáculo en el camino. Gracias a ellos soy la persona y el profesional que se llena de orgullo al graduarse como ingeniero electrónico.

Contenido

Tabla de Figuras

RESUMEN GENERAL DE TRABAJO DE GRADO

TITULO:	IMPLEMENTACION DE UNA INTERFAZ GRAFICA CON MATLAB PARA LA OPERACIÓN DE UN MODULO DE CONTROL DE POSICION O VELOCIDAD UTILIZANDO UN PLC S7-300 DE LA EMPRESA SIEMENS.
AUTOR(ES):	HECTOR FERNANDO QUINTERO
FACULTAD:	Facultad de Ingeniería Electrónica
DIRECTOR(A):	OMAR PINZÓN ARDILA

RESUMEN

El documento muestra el uso de software para la implementación de una interfaz gráfica que controla un sistema de posición o velocidad, del módulo del Laboratorio de Electrónica Industrial de la Universidad Pontificia Bolivariana seccional Bucarramanga. En el proyecto se utiliza el software Totally Integrated Automation Portal (TIA portal) para la programación del controlador PLC S7-300 de la empresa Siemens y el software SIMATIC WinCC Runtime Advanced, incluido en las herramientas de visualización de proceso en el software TIA portal. Se configura un servidor OPC para establecer una comunicación entre el TIA portal y MATLAB. En el software de MATLAB se utilizan las herramientas GUIDE y OPC para el diseño de la interfaz gráfica y comunicación con el servidor OPC configurado en SIMATIC WinCC Runtime Advanced. La configuración de la comunicación Profibus entre el variador de velocidad Micromaster 440, el encoder absoluto y el PLC S7-300 de la empresa Siemens complementa el lazo de realimentación del sistema de control para el modulo, permitiendo la implementación de diferentes técnicas de control en dispositivos modernos como los PLC.

PALABRAS CLAVES: TIA portal, SIMATIC WinCC, MATLAB, PLC, encoder, OPC server

V° B° DIRECTOR DE TRABAJO DE GRADO

GENERAL SUMMARY OF WORK OF GRADE

TITLE: IMPLEMENTATION OF A GUI WITH MATLAB FOR THE OPERATION OF A MODULE POSITION CONTROL OR SPEED CONTROL USING A PLC S7-300 FROM SIEMENS.

AUTHOR(S): HECTOR FERNANDO QUINTERO

FACULTY: Facultad de Ingeniería Electrónica

DIRECTOR: OMAR PINZÓN ARDILA

ABSTRACT

The document shows the use of software for implementing a graphical interface that controls a position or velocity system, module Industrial Electronics Laboratory of the Universidad Pontificia Bolivariana sectional Bucarramanga. In the project the Totally Integrated Automation Portal (TIA Portal) software for programming the controller PLC Siemens S7-300 and SIMATIC WinCC Runtime Advanced software, included in the process visualization tools in the TIA Portal software is used. An OPC server is configured to establish communication between the TIA portal and MATLAB. MATLAB software in the GUIDE and OPC tools for the design of the graphical interface and communication with the OPC server configured in SIMATIC WinCC Runtime Advanced are used. The configuration of the Profibus communication between the VSD Micromaster 440, the absolute encoder and the PLC S7-300 from Siemens complements the feedback loop control system for the module, allowing the implementation of different control techniques in devices modern and the PLC.

KEYWORDS:

TIA portal, SIMATIC WinCC, MATLAB, PLC, encoder , OPC server

V° B° DIRECTOR OF GRADUATE WORK

INTRODUCCION

El crecimiento del sector industrial se debe a la implementación de la electrónica industrial en la rama de producción. La electrónica industrial es una rama de la ingeniería eléctrica que consigue adaptar y transformar la electricidad, con la finalidad de alimentar otros equipos, transportar energía, controlar el funcionamiento de máquinas eléctricas, etc. La electrónica industrial se refiere a la aplicación de dispositivos electrónicos, principalmente semiconductores, al control y transformación de potencia eléctrica. Esto incluye tanto aplicaciones en sistemas de control como de suministro eléctrico e incluso la interconexión sistemas eléctricos de potencia [1].

Constantemente en la industria, la aplicación de la electrónica industrial se ve sujeta a la solución de problemas en procesos industriales teniendo en cuenta el precio de las maquinarias, los sistemas, los sensores, y todo instrumento que se necesite para optimizar la producción y reducir costos, viendo así una ganancia en el producto final. El PLC o controlador lógico programable es una opción muy recurrente en la industria debido a su eficiencia y versatilidad, ya que su programación es sencilla y además se pueden realizar modificaciones sin cambiar el cableado, mantenimiento económico y a un menor costo [1].

Las variables de proceso y los instrumentos con los cuales se miden en procesos industriales son de vital importancia para la implementación de técnicas de modelado y el control industrial. Las variables de procesos industriales tales como posición y velocidad son variables secundarias muy utilizadas en el control de motores eléctricos de cualquier proceso industrial. Una de las aplicaciones más comunes en el sector industrial para el control de posición o velocidad es el aplicado a los motores con bandas transportadoras y elevadores [2].

La importancia de una interfaz gráfica en el campo de la electrónica industrial y en cualquier proceso de producción ha tenido un gran auge en las últimas décadas, mejorando la interacción entre el operador y la máquina que se ve reflejada a largo plazo en la producción, por tal motivo se requiere de una interfaz gráfica amigable, con un lenguaje natural fácil de manipular y con la información necesaria para entender el comportamiento del proceso en ejecución, ordenes como inicio, parar, ver, graficar, entre otros, son ordenes limitadas en el lenguaje pero que permiten una fácil interacción con la máquina y el proceso [3].

En este proyecto se realiza el diseño de una interfaz gráfica en MATLAB para la implementación de técnicas de control en dispositivos modernos como los PLC, en un módulo del Laboratorio de Electrónica Industrial de la Universidad Pontificia Bolivariana, proporcionando a los estudiante habilidades en la identificación del proceso, diseño e implementación de controladores.

1. OBJETIVOS

1.1. Objetivo general.

Implementar una interfaz gráfica con MATLAB para la operación y sintonía de un módulo del control de posición o velocidad utilizando un PLC S7-300 de la empresa Siemens.

1.2. Objetivos específicos.

- Caracterizar y describir el módulo de control de posición o velocidad del laboratorio de electrónica industrial.
- Caracterizar y programar un encoder absoluto utilizando un S7-300 de la empresa Siemens.
- Implementar el control de posición angular mediante el conjunto motor-reductor, variador de velocidad y el PLC S7-300.
- Diseñar una interfaz gráfica con MATLAB para la identificación y control de posición de un motor de inducción.
- Diseñar una interfaz gráfica con MATLAB para la identificación y control de velocidad de un motor de inducción.
- Sintonizar un controlador PID utilizando el bloque PID del PLC S7-300 para controlar la posición o velocidad en un motor de inducción en lazo cerrado.
- Elaborar las prácticas de laboratorio para la utilización del módulo de control de posición o velocidad.

2. CONCEPTOS PRELIMINARES

La implementación de un PLC en la industria es de gran utilidad ya que facilita la programación y el control de los procesos. La variedad de PLC en el mercado son muy amplias y la implementación de éste depende de su aplicación y costo para un proceso industrial [4]. En el proyecto se utiliza el módulo de posición o velocidad del Laboratorio de Electrónica Industrial para el estudio de las variables de control. En la figura 1 se observa el módulo de posición o velocidad del Laboratorio de Electrónica Industrial de la Universidad Pontificia Bolivariana seccional Bucaramanga.

Figura 1. Módulo de posición o velocidad del Laboratorio de Electrónica Industrial de la Universidad Pontificia Bolivariana seccional Bucaramanga.

2.2. Hardware

2.2.1. SIMATIC S7-300

El SIMATIC S7-300 permite soluciones de sistema innovadoras con especial énfasis en tecnología de fabricación y como sistema de automatización universal, constituye una solución óptima para aplicaciones en estructuras centralizadas y descentralizadas [4].

Potentes módulos centrales con interfaz industrial Ethernet/PROFINET, funciones tecnológicas integradas o versión de seguridad en un sistema coherente evitan inversiones adicionales [4].

El S7-300 se puede configurar de forma modular donde no hay ninguna regla de asignación de *slots* (direcciónes de memoria) para los módulos periféricos. El S7-300 cuenta con una amplia gama de módulos, tanto para estructuras centralizadas, como para estructuras descentralizadas con ET-200M [4].

El uso de la *Micro Memory Card* como memoria de datos y de programa ahorra costos de mantenimiento. Además, en esta tarjeta de memoria se puede guardar un proyecto asociado con símbolos y comentarios para simplificar el trabajo del servicio técnico [4].

Asimismo, la *Micro Memory Card* permite la actualización sencilla del programa o del firmware sin programadora. Además se puede utilizar durante el funcionamiento para guardar y consultar datos, por ejemplo, para archivar medidas o para procesar recetas [4].

2.2.2. Encoder Absoluto

Los encoder son eficaces instrumentos de medición de posición con una extrema precisión, transformando el movimiento mecánico de rotación en señales digitales. Los encoder funcionan con un mínimo desgaste debido a su detección fotoeléctrica o magnética, para ello tienen acoplado un disco graduado en su eje o en un soporte magnético móvil [5].

Los encoders absolutos asignan a cada posición angular un valor inequívoco, incluso durante varias revoluciones. Incluso después de una caída de tensión el encoder detecta de forma rápida y segura la posición momentánea. Los encoders absolutos se dividen en dos tipos: los encoders monovuelta que dividen una revolución mecánica en un número determinado de pasos de medición y los encoders multivuelta que no sólo registran la posición angular, sino que también cuentan las revoluciones para poder calcular la velocidad [5].

2.2.3. Variador de Velocidad

Los variadores de velocidad o VSD (*Variable Speed Drive*) son convertidores de energía eléctrica utilizados para controlar la velocidad de giro en motores asíncronos trifásicos. Para controlar la velocidad de giro del motor, estos dispositivos convierten la tensión suministrada por la red eléctrica que tiene una frecuencia y magnitud fija en una tensión con frecuencia y magnitud variable que se entregada al motor [6].

Éste sistema es ampliamente utilizado en la industria para regular o controlar la velocidad de un motor aplicado a un proceso y de esta manera mejorar el aprovechamiento de la potencia y con ello un ahorro de energía.

2.3. Software

2.3.1 TIA Portal

Es el software que optimiza todos los procedimientos de procesamiento, operación de máquinas y planificación. Con su intuitiva interfaz de usuario, la sencillez de sus funciones y la completa transparencia de datos de fácil utilización. Los datos y proyectos preexistentes pueden integrarse sin ningún esfuerzo [7].

Es un editor para configurar hardware, efectuar programaciones lógicas, paramétrizar un convertidor de frecuencias o diseñar pantallas de la HMI, los entornos del software tienen un diseño intuitivo, lo que le permitirá ahorrar tiempo. Las funciones, propiedades y librerías se despliegan de forma automática con la vista más intuitiva y apropiada para cada ocasión [7].

El TIA Portal es una arquitectura de software avanzada diseñada a partir de un esquema de navegación muy sencillo. La ergonomía sofisticada asegura la máxima eficiencia [7].

2.3.1.1. SIMATIC WinCC Runtime Advanced

El software de runtime está incluido en los paneles de mando SIMATIC HMI y ofrece diferentes funcionalidades, en las cuales podemos encontrar, visualización, señalización y creación de informes, administración de usuarios, ampliable de forma flexible mediante *scripts* (archivo de ordenes). Es un software integrable en soluciones de automatización basadas en redes TCP/IP [8].

Algunas de las aplicaciones que ejecuta el software SIMATIC WinCC Runtime Advanced son [8]:

- Visualización con una interfaz de usuario compatible con Windows,
- Alarmas y avisos.
- Posibilidad de archivar avisos y valores de proceso.
- Recetas.
- Documentación de datos de proceso, eventos de aviso y recetas.
- WinCC SmartServer para manejo remoto a través de Intranet e Internet.
- Comunicación abierta, por medio del servidor OPC, usa el sistema de visualización como servidor de datos para componentes de automatización superiores.

2.3.2. MATLAB

MATLAB es el lenguaje de alto nivel y un entorno interactivo utilizado por millones de ingenieros y científicos de todo el mundo que les permite explorar y visualizar las ideas y colaborar en todas las disciplinas, incluyendo procesamiento de señales e imágenes, comunicaciones, sistemas de control y las finanzas computacionales [9].

El MATLAB se utiliza en proyectos tales como el consumo de energía de modelado para construir las redes eléctricas inteligentes; desarrollo de algoritmos de control para vehículos hipersónicos; en el análisis de datos de tiempo para visualizar la trayectoria y la intensidad de los huracanes; y que ejecutan millones de simulaciones para determinar la dosis óptima de antibióticos, entre muchas otras aplicaciones [9].

2.3.2.1. Características de MATLAB

Entre las características del software MATLAB se encuentran [9]:

- Lenguaje de alto nivel para el cálculo numérico, visualización y desarrollo de aplicaciones.
- Entorno interactivo para la exploración iterativa, el diseño y la resolución de problemas.
- Funciones matemáticas para álgebra lineal, estadística, análisis de Fourier, filtrado, optimización, integración numérica, y la resolución de ecuaciones diferenciales ordinarias.
- Construir gráficos para la visualización de datos y herramientas para la creación de parcelas personalizados
- Herramientas de desarrollo para mejorar la calidad del código y facilidad de mantenimiento y maximizar el rendimiento.
- Herramientas para la creación de aplicaciones con interfaces gráficas personalizadas.
- Funciones para integrar los algoritmos basados en MATLAB con aplicaciones externas y lenguajes como C, Java, .NET y Microsoft Excel

2.3.2.2. Interfaz gráfica de MATLAB

Graphical user interface design environment (GUIDE) proporciona herramientas para el diseño de interfaces de usuario para aplicaciones personalizadas. Usar el editor de diseño GUIDE, permite diseñar gráficamente la interfaz de usuario. GUIDE genera automáticamente el código de MATLAB para la construcción de la interfaz de usuario, que se puede modificar para programar el comportamiento de la aplicación [10].

Para obtener más control sobre el diseño y el desarrollo, también se puede crear el código en MATLAB que define todas las propiedades de los componentes y comportamientos. MATLAB contiene funcionalidad integrada para ayudar a crear la interfaz gráfica de usuario para la aplicación mediante programación. Usted puede agregar cuadros de diálogo,

controles de interfaz de usuario (como botones y controles deslizantes) y contenedores (tales como paneles y grupos de botones) [10].

2.3.2.3. Herramienta OPC de MATLAB

La Herramienta OPC de MATLAB proporciona el acceso de datos directamente desde MATLAB donde se pueden leer, escribir y registrar datos OPC de dispositivos, tales como los sistemas distribuidos de control, control de supervisión y los sistemas de adquisición de datos y controladores lógicos programables. *OPC Toolbox* permite trabajar con los datos de los servidores que cumplen el estándar OPC *Historical Data Access* (HDA), y de la Arquitectura Unificada OPC (UA) estándar OPC *Data Access* (DA) [11].

La Herramienta OPC de MATLAB, tiene la alternativa de conectarse mediante la aplicación OPC Client, bloques en Simulink, o directamente desde los comandos en la líneas de código, para ser utilizadas en el desarrollo de cualquier aplicación [11].

MATLAB crea un objeto como cliente del servidor OPC, al momento de conectarse este explora y recupera todas las propiedades de los elementos almacenados en el servidor, teniendo asi la posibilidad de conectarse con más de un servidor e interactuar con estos, modificando las propiedades de cada uno de los elementos contenidos en cada uno de ellos [11].

2.3.3. OPC Server

Un servidor OPC es una aplicación de software (*driver*) que cumple con una o más especificaciones definidas por la *OPC Foundation*. El Servidor OPC hace de interfaz, comunicando por un lado con una o más fuentes de datos utilizando sus protocolo nativos (típicamente PLCs, DCSs, básculas, Módulos I/O, controladores, etc.) y por el otro lado con los Clientes OPC (típicamente SCADAs, HMIs, generadores de informes, generadores de gráficos, aplicaciones de cálculos, etc.) [12].

Los Servidores OPC clásicos utilizan la infraestructura COM/DCOM (métodos de comunicación entre procesos) de Microsoft Windows para el intercambio de datos. Lo que significa que esos Servidores OPC deben instalarse bajo el Sistema Operativo de Microsoft Windows. Un Servidor OPC puede soportar comunicaciones con múltiples Clientes OPC simultáneamente [12].

La principal función de un Servidor OPC es la de traducir datos nativos de la fuente de datos en un formato OPC para que sean compatibles con una o más especificaciones del OPC [12].

2.4. Comunicación

2.4.1. Profibus

Los buses de campo se usan en la actualidad de forma prioritaria como un sistema de comunicación para el intercambio de información entre sistemas de automatización y sistemas de campo distribuidos. Profibus es un bus de campo que fue estandarizado bajo la norma EN 50 170 [13].

Con Profibus los componentes de distintos fabricantes pueden comunicarse sin necesidad de ajustes especiales de interfaces. Profibus puede ser usado para transmisión crítica en el tiempo de datos a alta velocidad y para tareas de comunicación extensas y complejas. Ésta versatilidad viene dada por las tres versiones compatibles que componen la familia Profibus [13].

PROFIBUS PA (bus de campo cuyo protocolo está optimizado para realizar las transferencias de información necesarias entre los sistemas electrónicos de control y los sistemas de instrumentación):

- Diseñado para automatización de procesos.
- Permite la conexión de sensores y actuadores a una línea de bus común incluso en áreas especialmente protegidas.
- Permite la comunicación de datos y energía en el bus mediante el uso de 2 tecnologías (norma IEC 1158-2).

PROFIBUS DP (bus de campo cuyo protocolo está optimizado para realizar las transferencias de información a alta velocidad y bajo coste):

- Optimizado para alta velocidad.
- Conexiones sencillas y económicas.
- Diseñada especialmente para la comunicación entre los sistemas de control de automatismos y las entradas/salidas distribuidas.

PROFIBUS FMS (bus de campo optimizada para realizar las transferencias de información, en la fábrica, a niveles superiores a los que operan las otras dos redes comentadas):

- Solución general para tareas de comunicación a nivel de célula.
- Gran rango de aplicaciones y flexibilidad.
- Posibilidad de uso en tareas de comunicaciones complejas y extensas.

2.4.2. Profinet

Process Field Net (Profinet) es un estándar basado en Ethernet que cumple con las especificaciones IEC 61158 y IEEE 802 para la automatización industrial, los objetivos de Profinet son [14]:

- Utilizar dispositivos Ethernet conjuntamente para aplicaciones en el entorno industrial.
- Usar estándares TCP (*Transmission Control Protocol*).
- Usar estándares IP (*Internet Protocol*).
- Automatización de aplicaciones con requisito de tiempo real.
- Integración directa de sistemas con el bus de campo.

Un dispositivo Profinet siempre dispone de una interfaz Profinet (eléctrica, óptica, inalámbrica). Muchos dispositivos vienen también con una interfaz PROFIBUS DP para la conexión de dispositivos PROFIBUS. Los dispositivos más usados en este estándar son:

- Sistemas de automatización (PLCs, PCs)
- Sistemas de periferia descentralizada
- Aparatos de campo (PLCs, PCs, aparatos hidráulicos y neumáticos)
- Componentes de red activos (switches, routers)

El uso del Profinet permite a cualquier dispositivo acceder a la red Ethernet para establecer una óptima comunicación entre los sistemas de automatización, teniendo una trasmisión de datos simultáneamente entre cada uno de ellos [14].

2.5. Control PID

El control PID se utiliza de forma casi general en todos los sistemas de control. En casos en que el modelo de la planta no es conocido y no se puede hacer uso de métodos analíticos, es ahí donde el control PID es más útil y eficiente. En los sistemas de control de procesos, se resalta el control PID ya que ha demostrado ser óptimo para brindar un control muy satisfactorio [15].

En particular, los controladores PI y PID funcionan bien en procesos industriales, por ejemplo se afirma que el 98% de los lazos de control son controlados por controladores PI y que en el control de proceso de aplicaciones, más del 95% de los controladores son de tipo PID [15].

Las tres componentes del control PID, integral, proporcional y derivativo, cada uno se suma al bucle de programación con la finalidad de corregir eficazmente y en el mínimo tiempo posible los efectos de perturbaciones [16].

La parte proporcional consiste en el producto entre la señal de error y la constante proporcional como para hacer el error en estado estacionario casi nulo, pero en la mayoría de los casos, estos valores solo serán óptimos en una determinada porción del rango total de control, siendo distintos los valores óptimos para cada porción del rango. Sin embargo,

existe también un valor límite en la constante proporcional a partir del cual, en algunos casos, el sistema alcanza valores superiores a los deseados. Este fenómeno se llama sobrepaso y, por razones de seguridad no debe sobrepasar el 30%, aunque es conveniente que la parte proporcional no produzca sobrepaso [16].

Existe una relación lineal continua entre el valor de la variable controlada y la posición del elemento final de control. La parte proporcional no considera el tiempo, por lo tanto, la mejor manera de solucionar el error permanente y hacer que el sistema contenga alguna componente que tenga en cuenta la variación respecto al tiempo es incluyendo y configurando las acciones integral y derivativa. El modo de control Integral tiene como propósito disminuir y eliminar el error en estado estacionario, provocado por el modo proporcional. El control integral actúa cuando hay una desviación entre la variable y el punto de consigna, integrando esta desviación en el tiempo y sumándola a la acción proporcional con el propósito de obtener una respuesta estable del sistema sin error estacionario. Es decir el control integral se utiliza para obviar el inconveniente del *offset* (desviación permanente de la variable con respecto al punto de consigna) de la banda proporcional [16].

La acción derivativa se manifiesta cuando hay un cambio en el valor absoluto del error (si el error es constante, solamente actúan los modos proporcional e integral). El error es la desviación existente entre el punto de medida y el valor consigna, o "*Set Point*". La función de la acción derivativa es mantener el error al mínimo corrigiéndolo proporcionalmente con la misma velocidad que se produce, de esta manera se evita que el error se incremente. El error se deriva con respecto al tiempo y se multiplica por una constante derivativa que luego se suma a las señales anteriores (P+I). Es importante adaptar la respuesta de control a los cambios en el sistema ya que una mayor acción derivativa corresponde a un cambio más rápido y el controlador debe responder acordemente. Cuando el tiempo de acción derivativo es grande, hay inestabilidad en el proceso. Cuando es pequeño la variable oscila demasiado con relación al punto de consigna. Suele ser poco utilizada debido a la sensibilidad al ruido que manifiesta y a las complicaciones que ello conlleva. El tiempo óptimo de acción derivativa es el que retorna la variable al punto de consigna con las mínimas oscilaciones [16].

3. METODOLOGIA Y PROCEDIMIENTO

3.1. Planteamiento del problema.

El controlador de tipo PID ha sido durante más de siete décadas el eje de la ingeniería de control tanto en la práctica como en la academia. Este tipo de controlador es uno de los más usados debido a características importantes como proveer una realimentación en el sistema, eliminar el *offset* por medio de la acción integral y la capacidad de anticiparse a cambios en el valor deseado a través de una acción derivativa [15].

Actualmente en la industria se observa la dificultad al integrar estas estrategias de control a dispositivos modernos como los PLC, en nuestro caso como estudiantes de ingeniería electrónica de la Universidad Pontificia Bolivariana no todos cursamos en nuestro pensum académico la asignatura de Autómatas Programables (PLCs), por lo cual nace la necesidad

de implementar un módulo que nos permita acercarnos al manejo de estos dispositivos para el estudio de sistemas de control, donde el objeto de estudio es la variable posición o velocidad.

El módulo permitirá conocer algunas técnicas de control implementadas en los PLCs y no sólo las aprendidas durante la asignatura en el pregrado, proporcionando las habilidades en el campo de la automatización y el control industrial para una futura implementación en el ámbito laboral.

3.2. Descripción del módulo de control de posición o velocidad del Laboratorio de Electrónica Industrial

En la figura 2 se muestra el diagrama funcional del módulo de posición o velocidad que se encuentra en el Laboratorio de Electrónica Industrial de la Universidad Pontificia Bolivariana seccional Bucaramanga. El modulo se compone por un motor de inducción de la empresa Siemens de 1/2 hp que se acopla a una caja reductora de velocidad, esta a su vez tiene en su eje acoplado un encoder absoluto. El motor de inducción se conecta al variador de velocidad Micromaster 440 de la empresa Siemens, el variador de velocidad y el encoder absoluto se conectan al PLC S7-300 de la empresa Siemens por medio de una red Profibus DP. El PLC se controla desde un computador personal (PC) utilizando una interfaz gráfica en MATLAB.

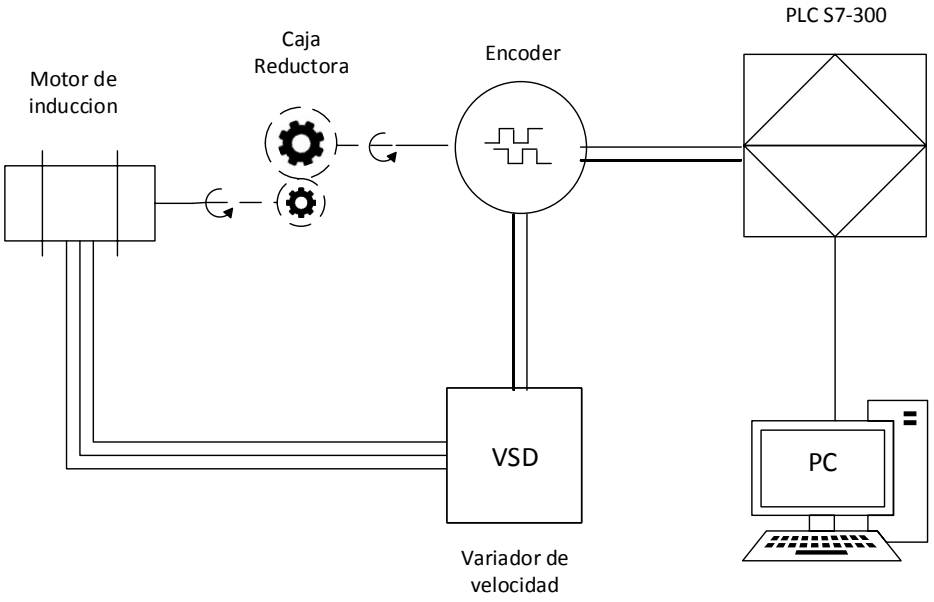


Figura 2. Diagrama funcional del módulo de posición o velocidad.

3.2.1. Motor

El modulo cuenta con un motor de inducción trifásico industrial de la marca Siemens (1LA7 070-4YA60), las características técnicas del motor se muestran en la tabla 1.

Tabla 1. Características del motor de inducción del módulo de posición o velocidad

Características del motor	Valor
Potencia	0.5HP/0.37KW
Eficiencia	63.3%
Factor de servicio	1.15
Tensión	220Vac(1.9A)/440Vac(0.95A)
Frecuencia	60Hz
Factor de potencia	0.81
Velocidad máxima	1590 rpm
Par máximo	2.9 Nm

Figura 3. Motor de inducción del módulo de posición o velocidad

3.2.2. Encoder absoluto

El módulo de posición o velocidad cuenta con un encoder absoluto de la empresa IFM de referencia RN3001 con conexión Profibus, funciona con una alimentación de 10-30 Vdc y posee una resolución de 13 bits (8192 posiciones). La elección de este dispositivo se debe a la gran resolución que posee y la conectividad por medio de Profibus lo que nos garantiza una comunicación del valor leído con gran precisión y sin ninguna variación. En la figura 4 se puede observar el encoder absoluto acoplado a la caja reductora de velocidad.

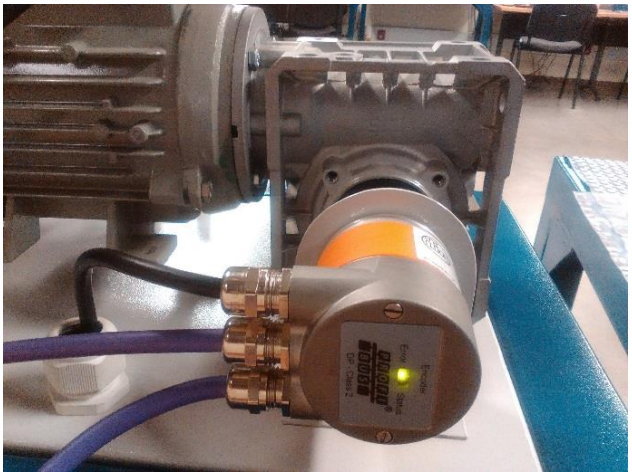


Figura 4. Encoder absoluto del módulo de posición o velocidad acoplado a la caja reductora de velocidades

3.2.3. Micromaster 440

El módulo de posición o velocidad cuenta con un variador de velocidad de la empresa Siemens Micromaster 440 que posee un rango de alimentación trifásica de 200-240 V a una frecuencia 47-63 Hz, una potencia de salida de 1/2 hp y protección de sobrecarga (150%) para el motor. En la figura 5 se observa el Micromaster 440 instalado en el módulo de posición o velocidad.

*Figura 5.*Micromaster 440 instalado en el módulo de posición o velocidad.

3.2.3. PLC S7-300

El PLC S7-300 dela empresa Siemens posee un módulo de alimentación monofásica que proporciona una salida de 24 Vdc con una corriente máxima de 5 A. Este módulo alimenta tanto la CPU como la instrumentación, al igual que contiene el módulo CPU315F-2J14-0AB0 que dispone de los puertos MPI, Profibus, Profinet y una interfaz PG/PC que permiten que las variables del PLC sean manipuladas y se pueda observar el funcionamiento. Por otro lado, cuenta con dos módulos de salidas/entradas digital y análogo. En la figura 6 se observa el PLC S7-300 instalado en el módulo de posición o velocidad.

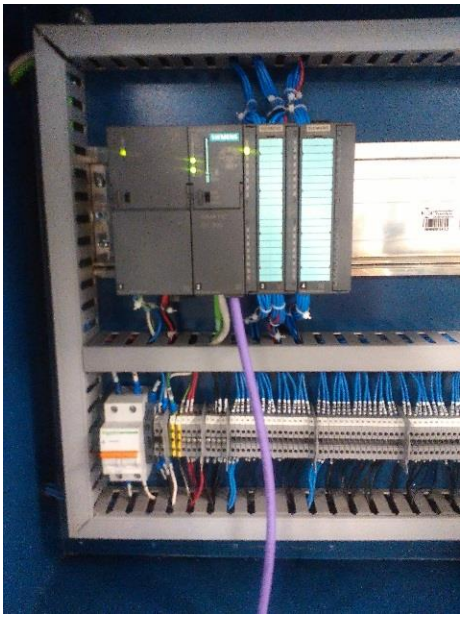


Figura 6. PLC S7-300 instalado en el módulo de posición o velocidad

3.3. Descripción eléctrica y de comunicaciones del módulo de posición o velocidad

En la figura 7 Diagrama eléctrico y de comunicaciones del módulo de posición o velocidad, se describe el diagrama eléctrico y de comunicaciones del módulo. En el sistema se utiliza el conector B del módulo de posición o velocidad para acoplar las señales analógicas y la tensión de 24 Vdc. Las salidas analógicas del PLC se conectan al variador de velocidad por medio de un relé que se acciona automáticamente al encender el PLC, si este no se acciona las entradas del variador de velocidad están conectadas a un potenciómetro.

La conexión trifásica de 220 Vac a 60 Hz proviene de la red eléctrica y se conectada a las entradas del variador de velocidad por medio de un interruptor magneto-térmico. En la salida del variador de velocidad se entrega una tensión trifásica con magnitud y frecuencia variable que alimenta al motor de inducción.

La comunicación Profibus se compone por un maestro (PLC) y 2 esclavos (variador de velocidad y encoder absoluto). El bus comienza con un conector RS-485 en el PLC, después el bus sigue con el encoder absoluto y termina en el variador de velocidad con un segundo conector RS-485.

La comunicación TCP/IP entre el PLC S7-300 y el PC se realiza utilizando la red Ethernet de la Universidad Pontificia Bolivariana.

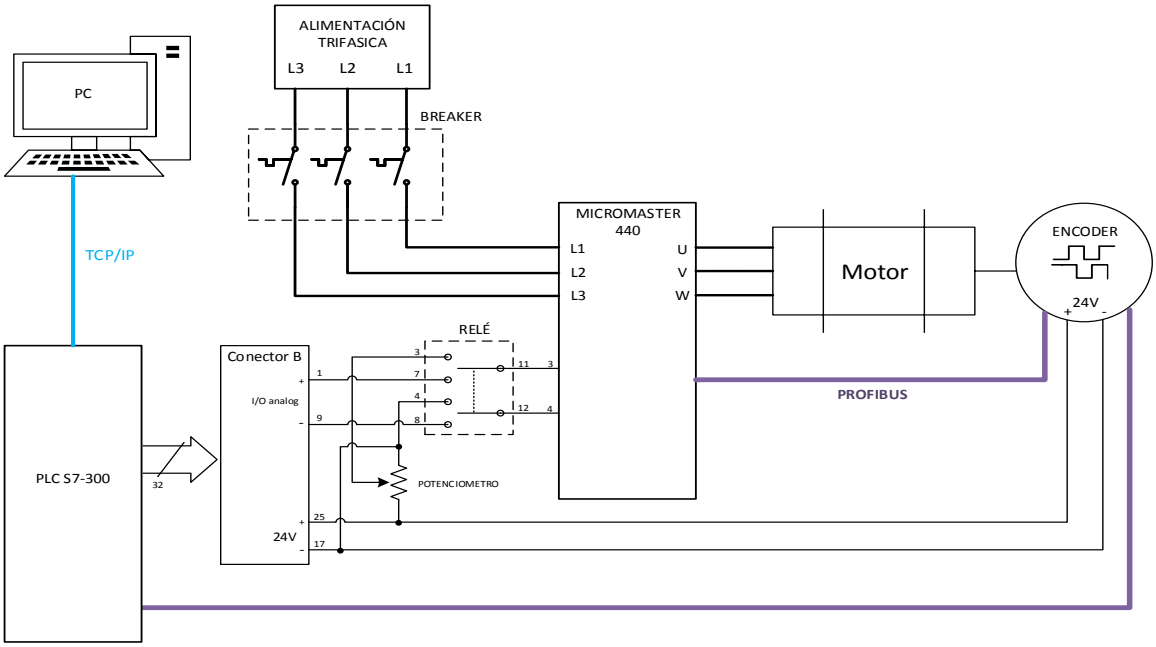


Figura 7. Diagrama eléctrico y de comunicaciones del módulo de posición o velocidad.

4. DESARROLLO DEL PROYECTO

Para lograr el correcto funcionamiento del módulo de posición o velocidad se utiliza el software TIA portal de Siemens, donde se implementa un programa de control utilizando el bloque PID y la configuración de los dispositivos periféricos pertenecientes a la red de comunicación Profibus, así se logran leer los datos provenientes del encoder absoluto y también el escribir los datos de salida en el formato adecuado para actuar sobre el variador de velocidad por medio de este protocolo.

El software *SIMATIC WinCC Runtime Advanced* contenido en el TIA portal se configura para operar como un servidor OPC que facilita la manipulación y visualización de las variables en el PLC, para esto se crea una HMI de visualización donde se configuran todas las variables HMI asociadas a las variables del PLC deseadas.

Para diseñar la HMI en Matlab se utiliza la herramienta de diseño GUIDE donde se utilizan objetos de control como *sliders* (objeto interactivo de una interfaz gráfica), botones, cuadros de texto, paneles, y demás, para poder manipular las variables del PLC. Las órdenes de la

herramienta OPC de MATLAB son agregadas al código de programación de la interfaz gráfica de MATLAB, para establecer la comunicación OPC entre el cliente de MATLAB y el servidor OPC de *SIMATIC WinCC Runtime Advanced.* En la Figura 8 Diagrama de bloques de la Comunicación OPC se muestra el diagrama de bloques donde se resalta la comunicación entre el sistema de control de movimiento y la interfaz gráfica de MATLAB.

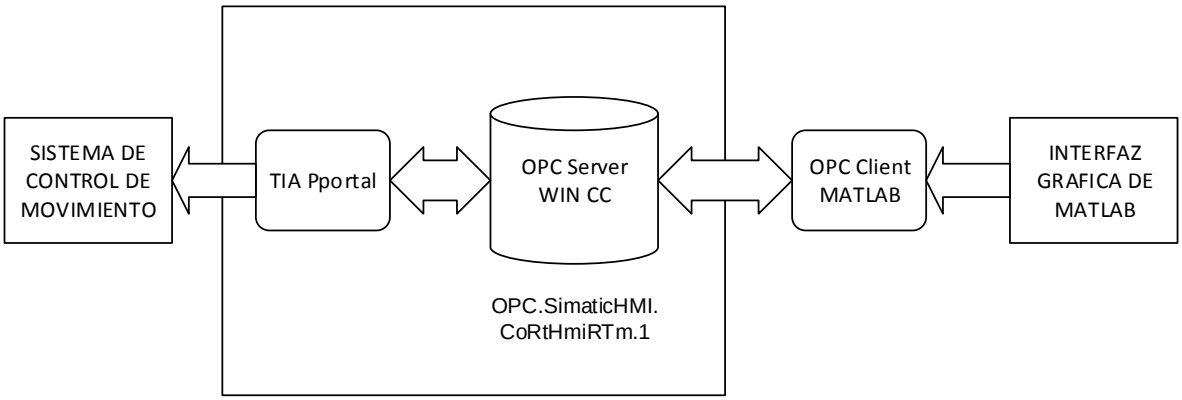


Figura 8. Diagrama de bloques de la Comunicación OPC.

Las ordenes utilizadas para establecer la comunicación entre la interfaz gráfica de MATLAB y las variables del servidor OPC de *SIMATIC WinCC Runtime Advanced* (OPC.SimaticHMI.CoRtHmiRTm.1) se muestran en el ejemplo 1 para la variable Set_Point del PLC.

Ejemplo 1:

```
da=opcda('localhost','OPC.SimaticHMI.CoRtHmiRTm.1');% crea un objeto
OPC(DA) como servidor OPC
connect(da);% conecta el servidor con el cliente OPC
grp=addgroup(da,'Group_1'); %crea un objeto como cliente del del servidor
y lo configura como grupo
set(grp, 'UpdateRate', 1);% determina la frecuencia con que suceden los
eventos para cada dato en el grupo
Set_Point = additem(grp, 'Set_Point');% agrega un elemento al objeto
grupo
disconnect(da);% desconecta el objeto OPC(DA)
```

Además de las ordenes de la herramienta OPC de MATLAB también se utilizan las ordenes *read* y *write*, para complementar la manipulación de las variables del servidor OPC como se muestran el ejemplo 2 para la variable Angle del PLC.

Ejemplo 2:

```
r=read(Angle);% lee el valor de la variable Angle agregada al objeto
grupo
y1={r.Value};% estrae el valor del arreglo (celda) para la variable leida
y2=cell2mat(y1);% convierte el arreglo (celda) en una variable flotante
write(Angle,x); % sobrescribe el valor de X en la variable Angle
```

4.1. Programa de control en TIA portal

En la figura 9 se muestra el programa de control implementado para el módulo de posición o velocidad. Este programa se desarrolla en un lenguaje *ladder* (lenguaje de programación gráfico dentro de los autómatas programables) el código puede verse en el Anexo 1 Programa de control.

El programa comienza ejecutando la variable *Start Word* que inicializa las variables relacionadas con la configuración del controlador. Posteriormente se selecciona cuál de las dos variables posición o velocidad se desea controlar y en qué modo, ya que el bloque PID tiene dos modos de ejecución: manual (lazo abierto) y automático (lazo cerrado). El bloque PID realiza la lectura del *setpoint* y de la variable controlada por medio del encoder absoluto y se calcula el error al comparar el *setpoint* y la variable controlada para calcular la acción de control donde la señal de mando se encuentra en unidades de Hz. La señal de mando se compara con los límites de saturación del variador de velocidad y se realizan las conversiones necesarias de la señal de mando para adaptarla al formato de la señal de control del variador. Por último se escribe el valor de mando en el variador de velocidad y se hacen las conversiones de las variables para visualizarlas en la HMI, actualizando las variables y se ejecuta todo el programa nuevamente hasta finalizar ejecutando la variable *Stop Word.*

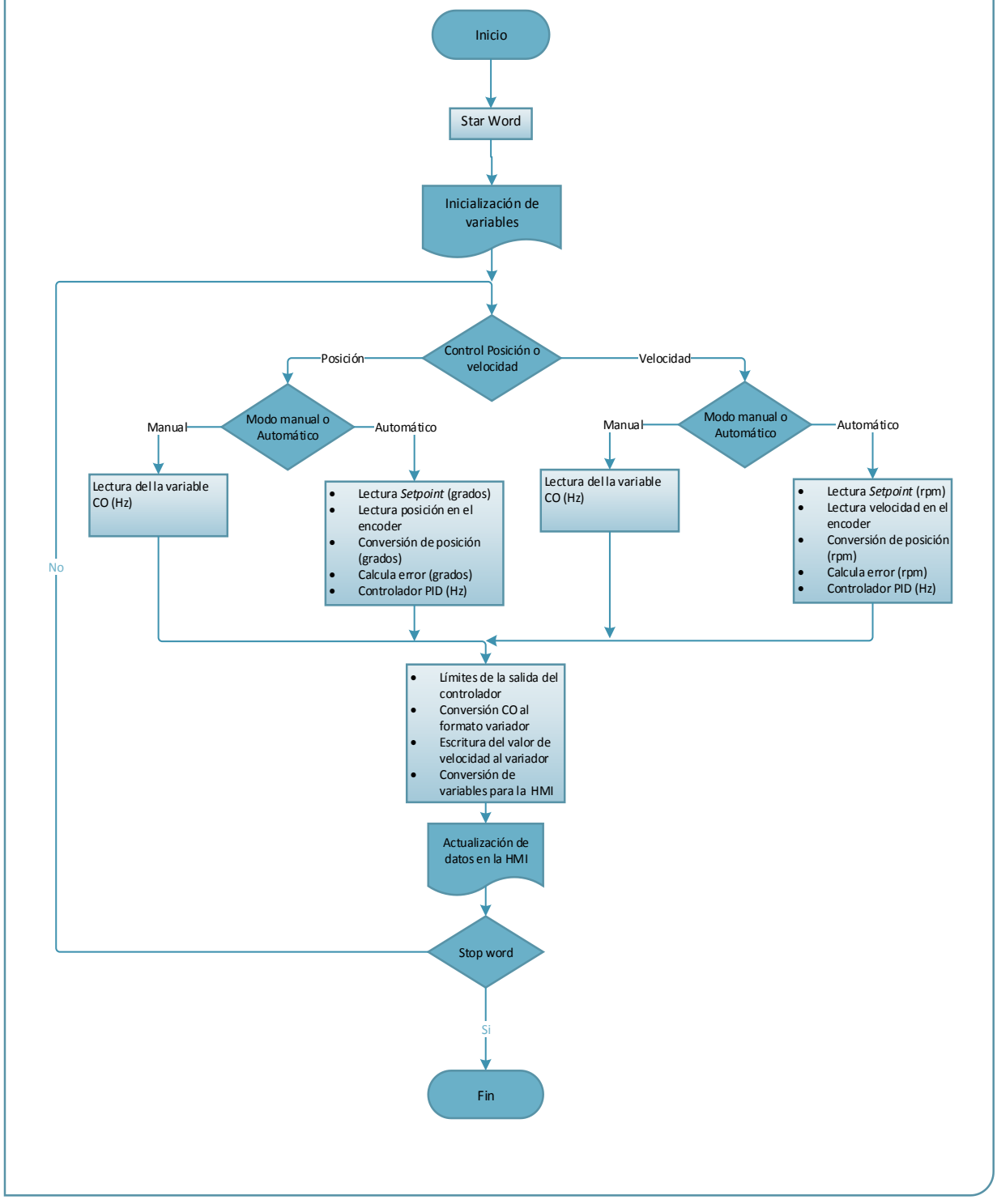


Figura 9. Programa de control del módulo de velocidad o posición.

4.2. Configuración del PLC S7-300 y la red Profibus

Para la configuración del PLC, es necesario agregar los módulos periféricos que integran la red Profibus y asignarles sus respectivas IP para realizar una comunicación Ethernet mediante el protocolo TCP/IP entre el PLC y el PC. También es necesario agregar los dispositivos de red Profibus al software, asignando una dirección para los respetivos telegramas de cada uno de ellos y así garantizar la comunicación entre todos los dispositivos. En el Anexo 2 Configuración del enlace Profibus con el PLC se puede observar más detalladamente la configuración de comunicación entre estos dispositivos

4.3. Configuración de los parámetros del Micromaster 440

El variador de velocidad requiere la introducción previa de los parámetros del motor que se desea manipular. Para la selección de la fuente de comandos y la consigna de frecuencia de salida en el variador de velocidad utilizando los parámetros de la red Profibus. Estos parámetros se configuran con los datos observados en el Anexo 3 Configuración de parámetros en el variador de velocidad.

4.4. Diseño de la HMI en SIMATIC WinCC Runtime Advanced

Para crear la HMI en *SIMATIC WinCC Runtime Advanced* inicialmente se agrega el módulo de Sistemas PC en el TIA portal y seleccionamos el *SIMATIC WinCC Runtime Advanced*. En este módulo se agrega además una interfaz Profinet para realizar la conexión por medio del protocolo TCP/IP. En las aplicaciones de este módulo se encuentra la configuración del *runtime* donde se configura la HMI de *SIMATIC WinCC Runtime Advanced* para actuar como servidor OPC, se crean las imágenes para la HMI en *SIMATIC WinCC Runtime Advanced* y se configuran las variables de la HMI asociadas a las variables del PLC, además de la creación de ficheros, en caso de requerir una base de datos para su uso posteriormente. Todas estas configuraciones se pueden observar más detalladamente en el Anexo 4 Creación de la HMI en *SIMATIC WinCC Runtime Advanced*.

La HMI en *SIMATIC WinCC Runtime Advanced* consta de 3 ventanas, la ventana inicial en donde se selecciona el tipo de variable que se desea controlar y dos ventanas adicionales una para cada variable controlada, donde se interactúa con las variables del PLC seleccionando: el s*etpoint*, el modo de control (manual ó automático), los parámetros de control (Kp,Ti,Td), el inicio, la parada, la saturación máxima y mínima en el variador de velocidad y la visualización en tiempo real de la variable controlada en una gráfica con tabla. En la Figura 10 se muestra la Ventana de inicio de la HMI en *SIMATIC WinCC Runtime Advanced*, Figura 11 se muestra la Ventana de posición de la HMI en *SIMATIC WinCC Runtime Advanced* y en la Figura 12 se muestra la Ventana de velocidad de la HMI en *SIMATIC WinCC Runtime Advanced*.

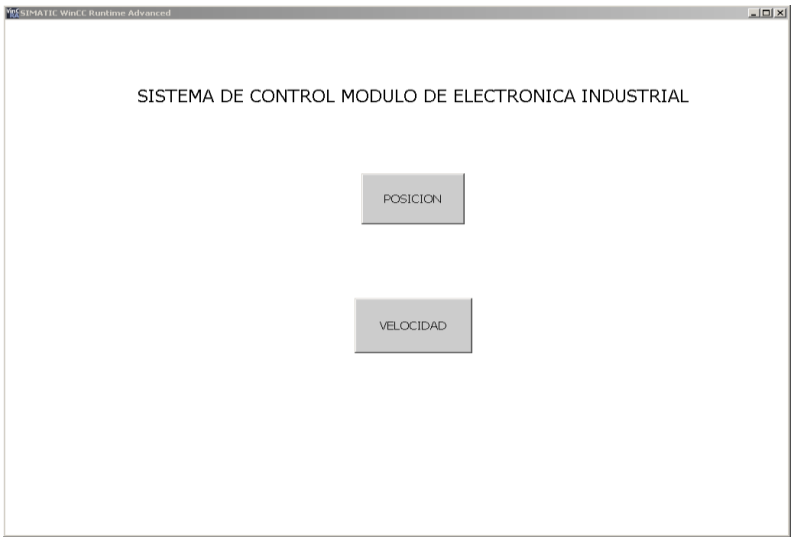


Figura 10. Ventana de inicio de la HMI en SIMATIC WinCC Runtime Advanced.

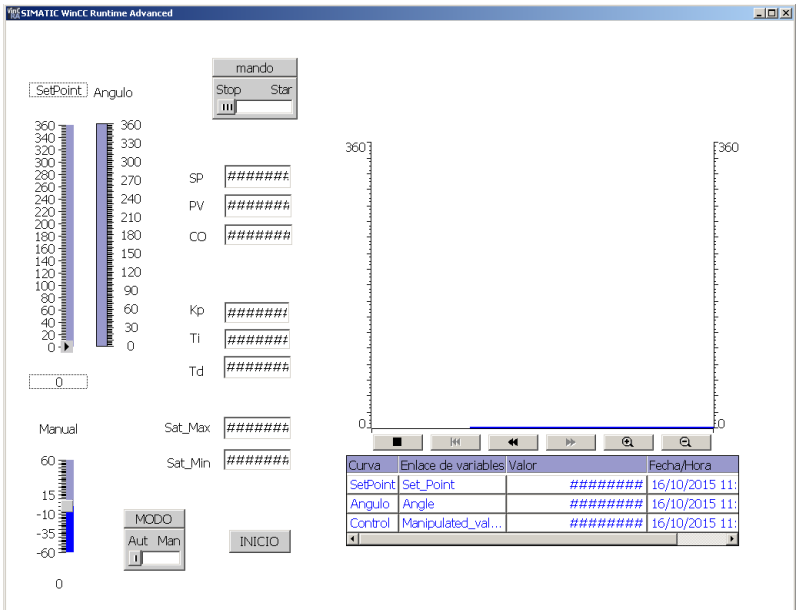


Figura 11. Ventana de posición de la HMI en SIMATIC WinCC Runtime Advanced.

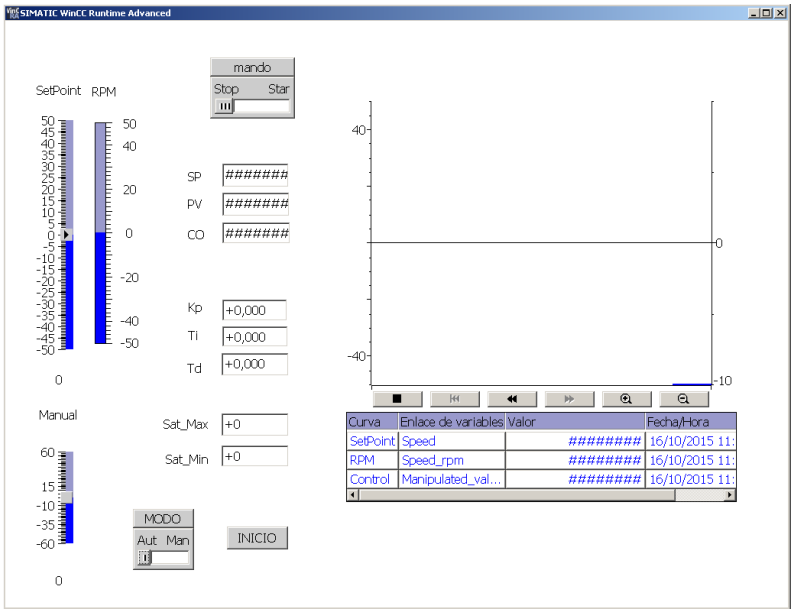


Figura 12. Ventana de velocidad de la HMI en SIMATIC WinCC Runtime Advanced.

4.5. Verificacion del servidor OPC por medio de MATLAB

Para continuar con el desarrollo del proyecto y tener una comunicación entre la interfaz gráfica de MATLAB y el PLC S7-300, inicialmente se requiere verificar el correcto funcionamiento de la HMI en *SIMATIC WinCC Runtime Advanced* como servidor OPC, ya que esta acción no es visible desde la interfaz gráfica de MATLAB. Por tal motivo utilizamos la aplicación OPC Client de la herramienta OPC de MATAB para visualizar el servidor OPC de *SIMATIC WinCC Runtime Advanced*.

Para iniciar la aplicación se abre el software MATLAB y en la pestaña de APPS se selecciona la aplicación OPC client incluida a la librería de la herramienta OPC de MATLAB, en ella se visualiza una pantalla con dos paneles, uno de ellos para agregar el servidor OPC y el otro para agregar el cliente OPC. En el panel del cliente se crea un objeto y se agrega las variables requeridas para la interfaz gráfica en MATLAB configuradas en el servidor OPC de *SIMATIC WinCC Runtime Advanced*. En el panel del cliente se puede visualizar la lectura de la variable ó interactuar con ella sobrescribiendo el valor y modificando sus parámetros en tiempo real. En el Anexo 5 Confirmación del servidor OPC por medio de MATLAB se puede observar en más detalle el proceso para confirmar el funcionamiento del servidor OPC.

En la Figura 13 Verificación del servidor OPC con la aplicación OPC Client de MATLAB, se observa un ejemplo de la aplicación de OPC client con el servidor OPC.SimaticHMI. CoRtHmiRTm.1 de *SIMATIC WinCC Runtime Advanced* y las variables cargadas en el cliente OPC.

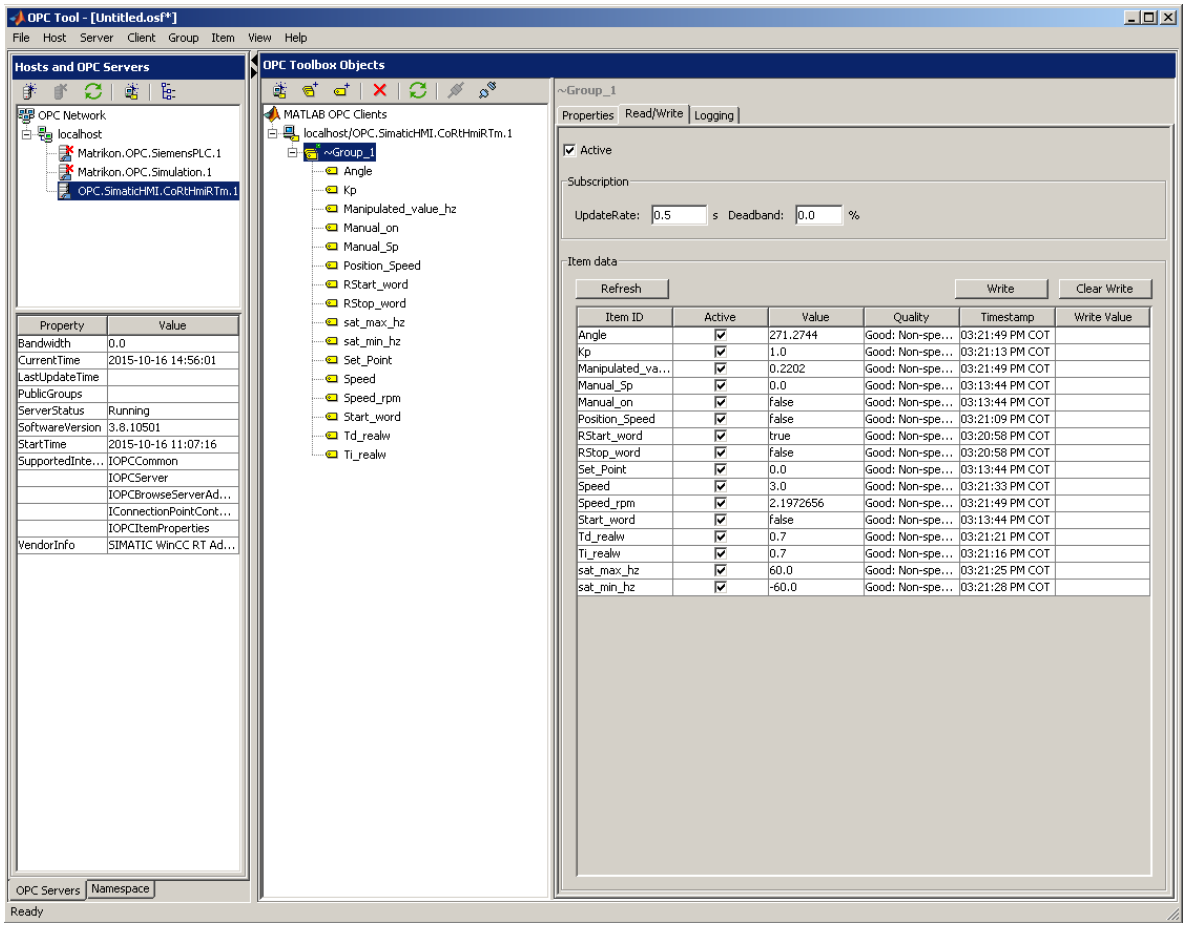


Figura 13. Verificación del servidor OPC con la aplicación OPC Client de MATLAB.

Hay que tener en cuenta que para observar el funcionamiento del servidor OPC se debe ejecutar el *runtime* del *SIMATIC WinCC Runtime Advanced*, si este no se ejecuta al tiempo, la herramienta del OPC de MATLAB no puede conectarse con el servidor OPC.

4.6. Diseño de la interfaz gráfica en MATLAB

El diseño de la interfaz gráfica en MATLAB se desarrolla por medio de la herramienta GUIDE, en el Anexo 6 Diseño de la interfaz gráfica en MATLAB se detalla más el proceso de diseño.

La interfaz gráfica para la operación del módulo de control de posición o velocidad cuenta con una sola ventana, esta ventana tiene un encabezado principal, un panel de inicio con cuatro botones (*push button*) para *Position, Speed, Graphic, Out*. Un panel con una gráfica auxiliar de observación, un panel de portada y dos paneles de selección que se visualizan según la selección de los botones del panel de inicio. En la Figura 14 Interfaz gráfica de MATLAB, panel de portada, se observa la ventana de la interfaz gráfica con el panel de portada.

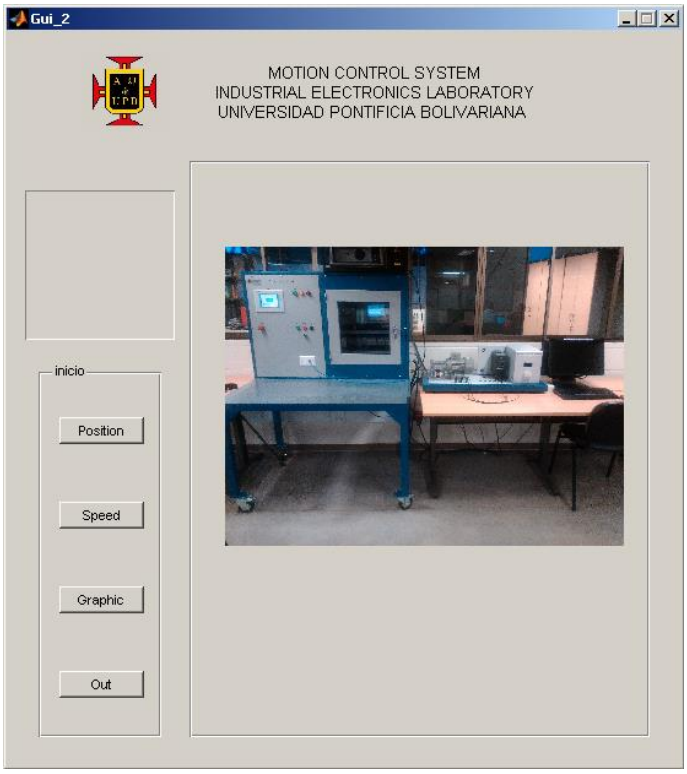


Figura 14. Interfaz gráfica de MATLAB, panel de portada.

El primer panel de selección es el panel de trabajo. En este se accede seleccionando el botón *Position* o el botón *Speed* , el panel cuenta con dos *sliders*, el *slider* de *setpoint* y el *slider* de visualización de la variable de proceso, tres cuadros de texto donde se observa el valor de las variables *setpoint,* proceso y *out control*, con una precisión de nueve cifras decimales, cinco cuadros editables para ingresar los parámetros del bloque PID del controlador (Kp, Ti, Td, saturación máxima, saturación mínima), dos *button group,* el primero para iniciar o parar la acción de mando del controlador y el segundo para seleccionar el tipo de control, manual o automático, si el tipo de control es manual se cuenta con un *slider* que varía el valor de la variable *out control*.

En la Figura 15 Interfaz gráfica de MATLAB, panel de trabajo, variable de posición y la Figura 16 Interfaz gráfica de MATLAB, panel de trabajo, variable de velocidad se observa la interfaz gráfica con el panel de trabajo asignado a cada una de las variables de proceso respectivamente.

Para un mejor aprovechamiento de la herramienta se diseña la interfaz gráfica de MATLAB para que los objetos contenidos en el panel de trabajo varíen sus parámetros y se ajusten tanto al sistema de control de posición como al sistema de control de velocidad dependiendo de cuál botón se seleccione.

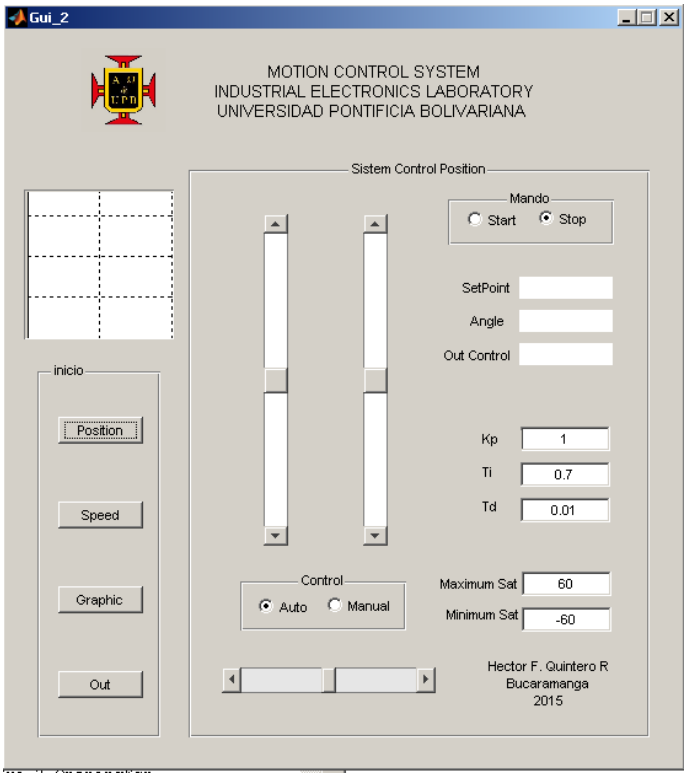


Figura 15. Interfaz gráfica de MATLAB, panel de trabajo, variable de posición.

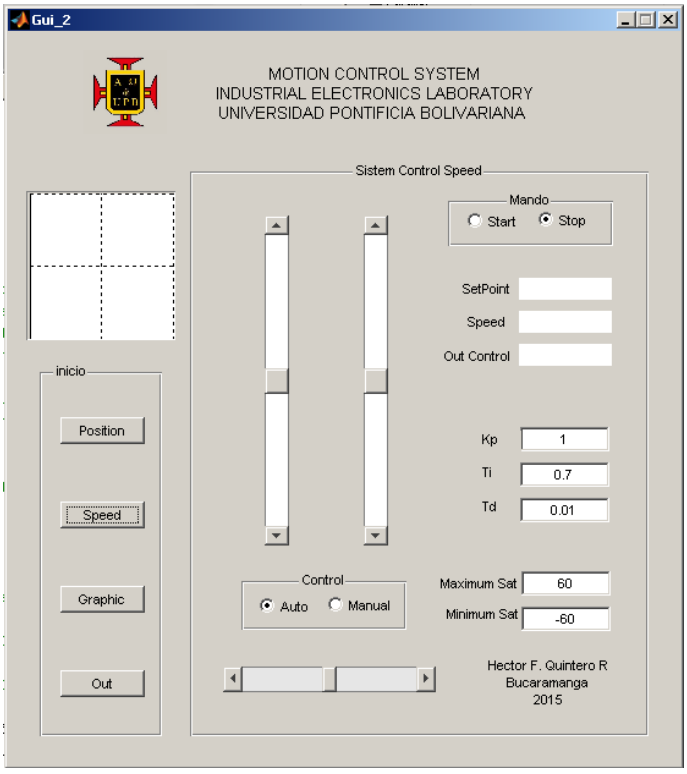


Figura 16. Interfaz gráfica de MATLAB, panel de trabajo, variable de velocidad.

El segundo panel de selección es el panel de gráficas. En este se accede seleccionando el botón *Graphic*, el panel cuenta con dos cuadros de grafica (*axes*). El primero visualiza la variable de proceso y el segundo la variable de control, ambas graficas se ejecutan en tiempo real. Adicionalmente el cuadro de grafica cuenta con una señal y un puntero que registran el valor leído de las variables. El panel además cuenta con dos botones, asociados a las variables de mando para poder iniciar y parar el control del módulo cuando se desee, y un botón de limpiado para reiniciar la visualización en las gráficas, de esta manera se visualiza mejor el comportamiento del sistema de control de posición o de velocidad. En la Figura 17 Interfaz gráfica de MATLAB, panel de graficas se observa la interfaz gráfica con el panel de gráficas.

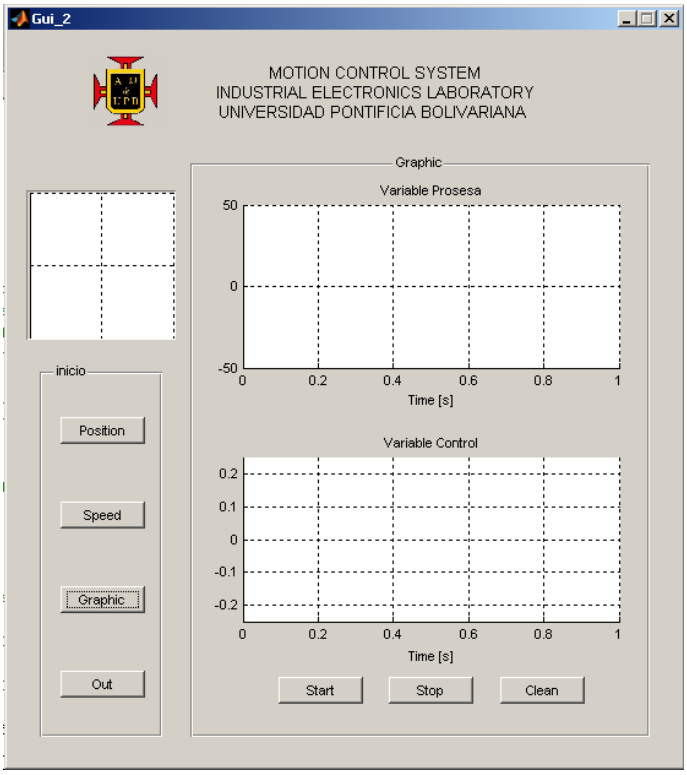


Figura 17. Interfaz gráfica de MATLAB, panel de graficas.

4.7. Sintonización del módulo de control de posición o velocidad

La sintonización del módulo de posición o velocidad se hace por medio de las herramientas *System Identification Toolbox* y *Simulink* del software MATLAB, para la identificación del módulo de posición o velocidad se debe tener en cuenta que la identificación del sistema se hace al conjunto conformado por el variador de velocidad, motor y la caja reductora de velocidad.

Para la identificación del módulo de posición se toma el modelo de la planta obtenido en proyectos realizados anteriormente [17], el modelo obtenido es:

$$G(s) = \frac{2.6383}{s(0.0239s^2 + 0.2033s + 1)} \quad (1)$$

Teniendo como base el modelo matemático obtenido anteriormente se realiza el diagrama de bloques del simulación en *Simulink,* mostrado en la Figura 18 Diagrama de bloques para la simulación del módulo de posición, y se realiza la sintonización del controlador PID por medio de la herramienta *autotuning* del bloque PID, teniendo en cuenta la saturación del controlador (-60 Hz / 60 Hz) y los límites de la salida en el modelo del sistema (0°/360°).

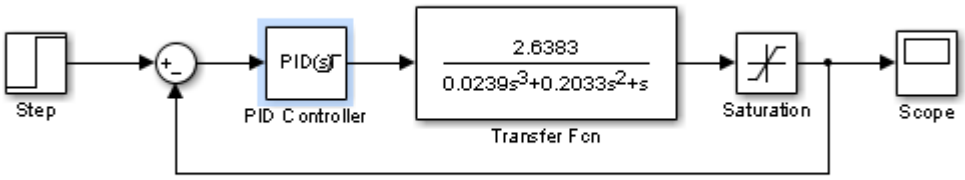


Figura 18. Diagrama de bloques para la simulación del módulo de posición.

Al utilizar la herramienta de *autotuning* se obtienen los siguientes parámetros para el controlador PID, Kp= 1.6476, Ti=0.60061, Td=0.32868, tal y como se puede observar en la Figura 19 Autotuning del bloque PID para el módulo de posición. En la Figura 20 Respuesta del comportamiento del módulo de posición con el bloque PID también se analiza que el modulo tiene una respuesta rápida en el comportamiento de su salida aplicando el bloque PID teniendo un tiempo de establecimiento aproximado de 6 segundos y con un sobrepaso menor al 10%.

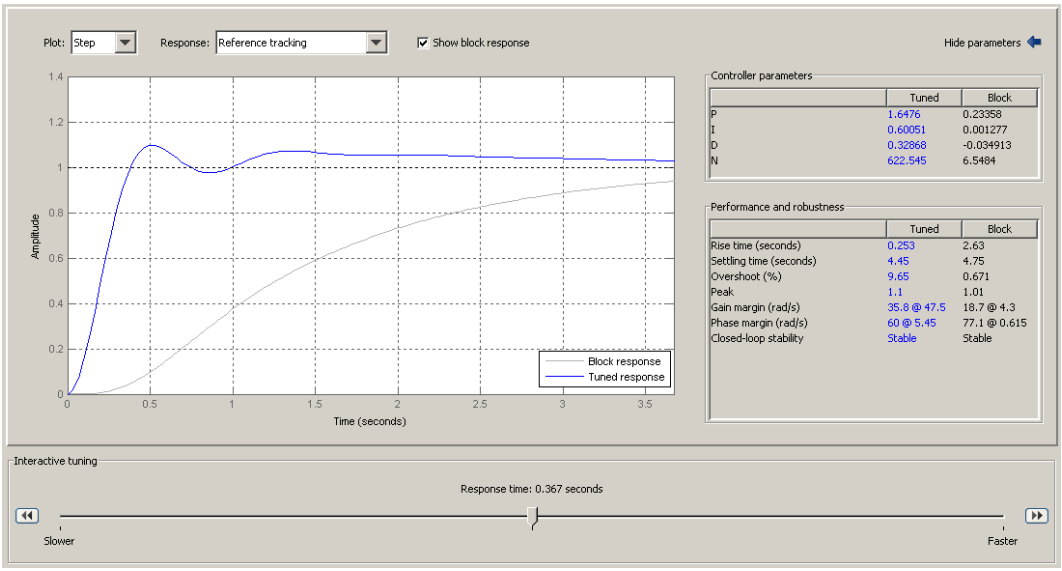


Figura 19. Autotuning del bloque PID para el módulo de posición.

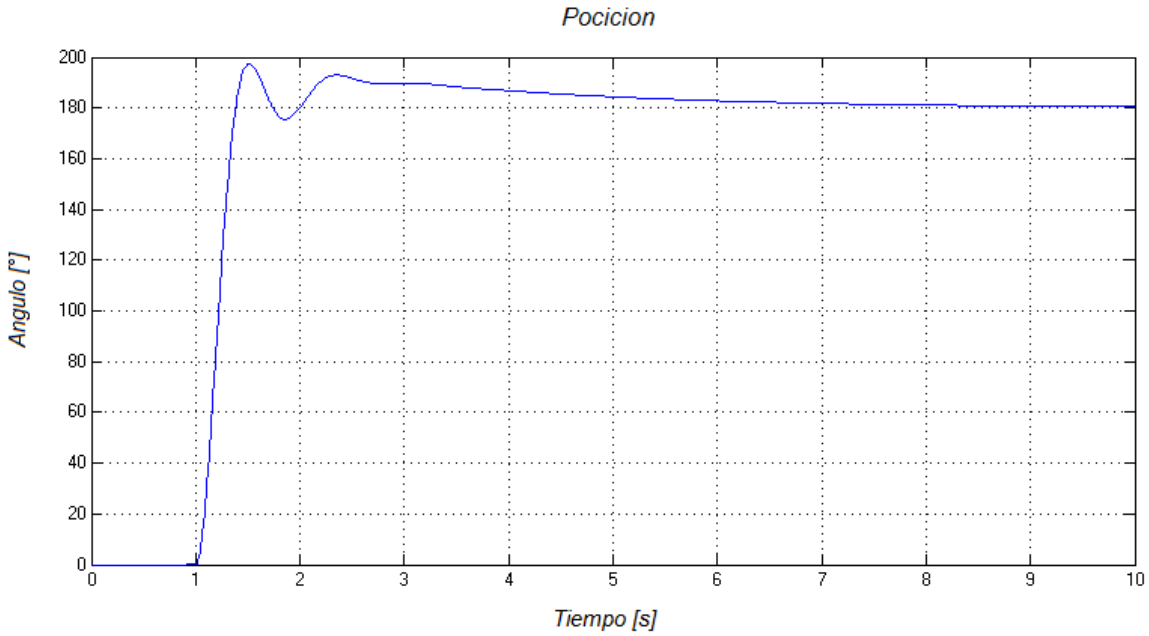


Figura 20. Respuesta del comportamiento del módulo de posición con el bloque PID.

Para la identificación del módulo en velocidad se utilizan las bases de datos configurados en el *SIMATIC WinCC Runtime Advanced* para acceder a los datos de caracterización del módulo. En la Figura 21 Segmento de la base de datos para la variable velocidad, se muestra un segmento de la base de datos donde se pueden observar los valores obtenidos y guardados correspondiente a la variables de velocidad y control, estos datos se exportan a MATLAB para utilizar la herramienta *Ident* de *System Identification Toolbox* que determina el modelo matemático del sistema.

En la Figura 22 Identificación del módulo de velocidad utilizando la herramienta *Ident,* se muestran las ventanas asociadas al proceso de identificación.

Manipulatec	20/10/2015 0	0.06973	1	4.2298E+10	Speed_rpm	20/10/2015 0	1.09863281	1	4.2298E+10
Manipulatec	20/10/2015 0	0.06973	1	4.2298E+10	Speed_rpm	20/10/2015 0	4.76074219	1	4.2298E+10
Manipulatec	20/10/2015 0	0.06973	1	4.2298E+10	Speed_rpm	20/10/2015 0	1.83105469	1	4.2298E+10
Manipulatec	20/10/2015 0	0.06973	1	4.2298E+10	Speed_rpm	20/10/2015 0	2.56347656	1	4.2298E+10
Manipulatec	20/10/2015 0	0.06973	1	4.2298E+10	Speed_rpm	20/10/2015 0	4.76074219	1	4.2298E+10
Manipulatec	20/10/2015 0	0.06973	1	4.2298E+10	Speed_rpm	20/10/2015 0	4.76074219	1	4.2298E+10
Manipulatec	20/10/2015 0	0.06973	1	4.2298E+10	Speed_rpm	20/10/2015 0	4.76074219	1	4.2298E+10
Manipulatec	20/10/2015 0	0.09175	1	4.2298E+10	Speed_rpm	20/10/2015 0	7.69042969	1	4.2298E+10
Manipulatec	20/10/2015 0	0.09542	1	4.2298E+10	Speed_rpm	20/10/2015 0	7.32421875	1	4.2298E+10
Manipulatec	20/10/2015 0	0.09542	1	4.2298E+10	Speed_rpm	20/10/2015 0	7.69042969	1	4.2298E+10

Figura 21. Segmento de la base de datos para la variable velocidad.

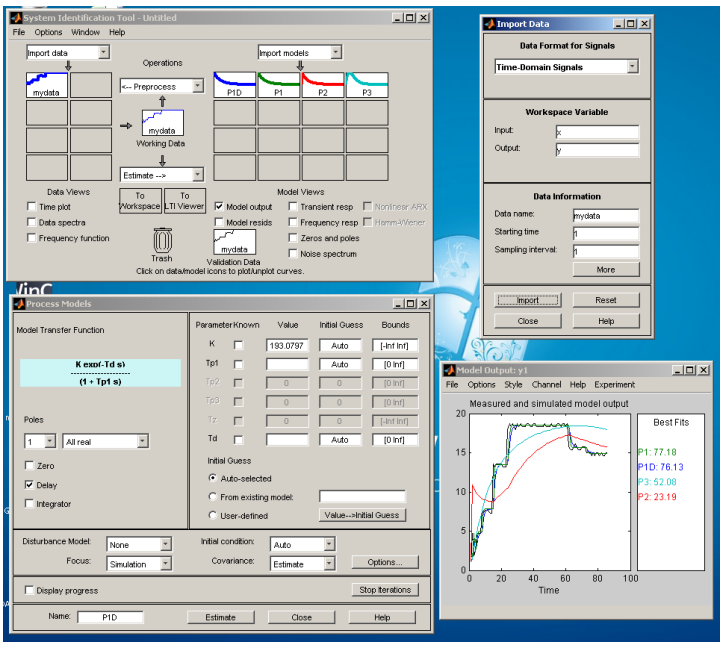


Figura 22. Identificación del módulo de velocidad utilizando la herramienta Ident.

De la identificación del sistema se obtiene el siguiente modelo matemático:

$$G(s) = \frac{193.74}{0.36125s + 1} \tag{2}$$

Partiendo del modelo matemático se realiza el diagrama de bloques del sistema en *Simulink* y que se muestra en la Figura 23 Diagrama de bloques para la simulación del módulo de velocidad. La sintonización del controlador PID se realiza por medio de la herramienta *autotuning* del bloque PID, teniendo en cuenta la saturación del controlador (-60 Hz /60 Hz) y los límites de la salida en el modelo del sistema (-50 rpm /50 rpm).

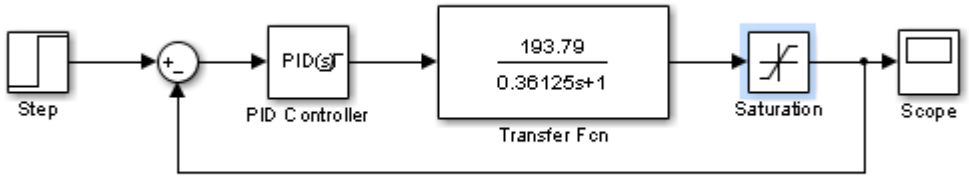


Figura 23. Diagrama de bloques para la simulación del módulo de velocidad.

Al utilizar la herramienta de *autotuning* se obtienen los siguientes parámetros para el controlador PID, Kp= 0.096742, Ti=0.47607, Td=0.0, tal y como se puede observar en la Figura 24 Autotuning del bloque PID para el módulo de posición. En la Figura 25 Respuesta del comportamiento del módulo de velocidad con el bloque PID, también se analiza que el modulo tiene una respuesta rápida en el comportamiento de su salida aplicando el bloque PID teniendo un tiempo de establecimiento aproximado de 0.4 segundos y con un sobrepaso un poco mayor al 8%.

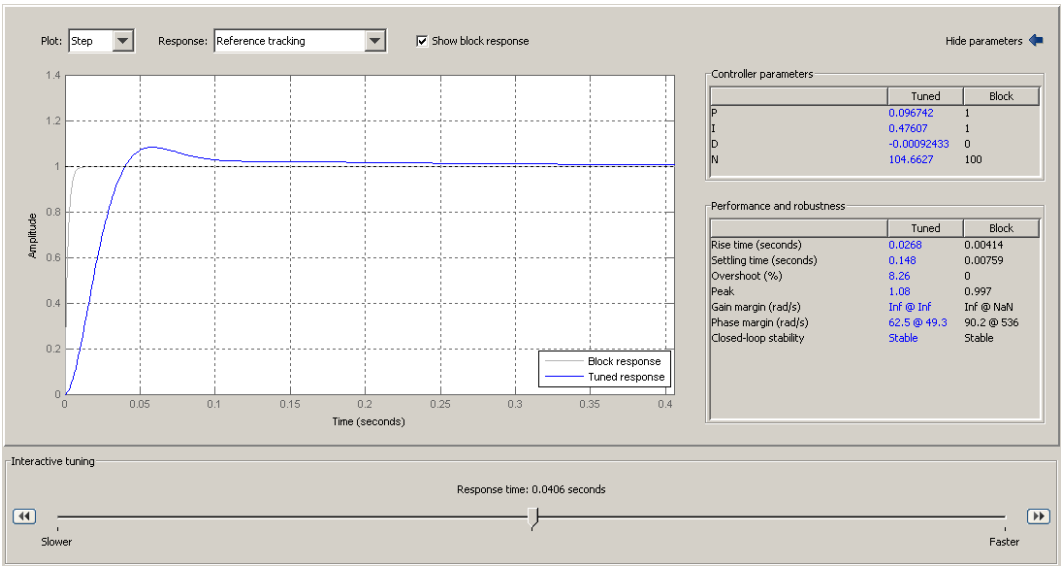


Figura 24. Autotuning del bloque PID para el módulo de velocidad.

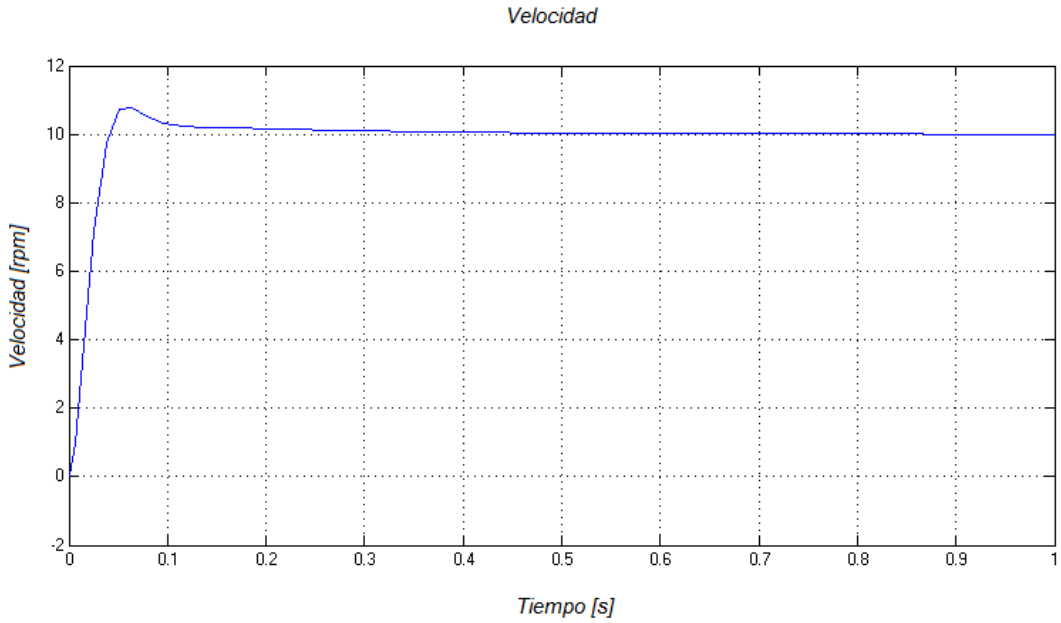


Figura 25. Respuesta del comportamiento del módulo de velocidad con el bloque PID.

5. Conclusiones

Se implementó una interfaz gráfica de control, que facilita la visualización de procesos afianzando los conocimientos en la identificación, sintonización y modelado matemático de sistemas de control.

Se caracterizó y describió el módulo de control de posición o velocidad del Laboratorio de Electrónica Industrial de la Universidad Pontificia Bolivariana.

Se implementó y diseño el control de posición y velocidad para el módulo de posición o velocidad del Laboratorio de Electrónica Industrial de la Universidad Pontificia Bolivariana.

Se sintonizo un controlador PID utilizando el bloque PID del PLC S7-300 para controlar la posición o velocidad en un motor de inducción en lazo cerrado.

Se elaboraron las prácticas de laboratorio para la utilización del módulo de control de posición o velocidad.

El uso del servidor OPC de *SIMATIC WinCC Runtime Advanced* para establecer la comunicación entre la interfaz gráfica en MATLAB y el software TIA portal, abre las posibilidades de trabajos futuros con el módulo de posición o velocidad permitiendo trabajar y establecer una comunicación con más clientes OPC simultáneamente, asemejándolo aún más a los sistemas actuales de producción y al control utilizado en la industria.

Al utilizar un estándar de comunicación como el protocolo Profibus y un encoder absoluto, aseguramos una entrega y lectura confiable de los datos de las variables de proceso en el lazo de control del módulo de posición o velocidad.

El desarrollo de este proyecto permite la aplicación de técnicas de control estudiadas durante el pregrado para aplicarse a un sistema utilizado en la industria.

6. Bibliografía

[1] «INTRODUCCION A LA ELECTRONICA INDUSTRIAL,» [En línea]. Available: http://es.scribd.com/doc/96345882. [Último acceso: 14 octubre 2015].

[2] «CONTROLADOR DE VELOCIDAD Y POSICION,» [En línea]. Available: http//:www.unipamplona.edu.co/unipamplona/portallG/home_40/recursos/01_general/revista_6/13102011. [Último acceso: 14 octubre 2015].

[3] «INTERFACES AMIGABLES,» [En línea]. Available: http://www.ibersid.eu/ojs/index.php/scire/article/view/1037/1019. [Último acceso: 14 octubre 2015].

[4] «Siemens S7-300,» [En línea]. Available: http://www.swe.siemens.com/spain/web/es/industry/automatizacion/simatic/controladores/pages/s7300.aspx. [Último acceso: 14 octubre 2015].

[5] «ENCODERS - DESCRIPCION DEL SISTEMA,» [En línea]. Available: http://www.ifm.com/ifmmx/web/pinfo015_010_040.htm. [Último acceso: 15 octubre 2015].

[6] ABB, «WHAT IS A VARIABLE SPEED DRIVE?,» [En línea]. Available: www.abb.com/cawp/db0003db002698/a5bd0fc25708f141c12571f10040fd37.aspx. [Último acceso: 15 octubre 2015].

[7] «TIA Portal,» [En línea]. Available: http://www.industry.siemens.com/topics/global/es/tia-portal/tia-portal-framework/Pages/default.aspx. [Último acceso: 14 octubre 2015].

[8] «SIMATIC WinCC (TIA Portal) - Sistemas Runtime basadas en PC,» [En línea]. Available: http://www.industry.siemens.com/topics/global/es/tia-portal/hmi-sw-tia-portal/wincc-tia-portal-rt/pages/default.aspx. [Último acceso: 14 octubre 2015].

[9] «MATLAB,» [En línea]. Available: http://www.mathworks.com/products/matlab/features.htlm. [Último acceso: 14 octubre 2015].

[10] «CREACION DE APLICACIONES CON INTERFACES GRAFICAS DE USUARIO EN MATLAB,» [En línea]. Available: http://www.mathworks.com/discovery/matlab-gui.html. [Último acceso: 14 octubre 2015].

[11] «OPC TOOLBOX DE MATLAB,» [En línea]. Available: http://www.mathworks.com/products/opc/. [Último acceso: 14 octubre 2015].

[12] «OPC SERVER,» [En línea]. Available: http://matrikonopc.es/opc-servidor/index.aspx. [Último acceso: 14 octubre 2015].

[13] U. P. d. Cartagena, «PROFIBUS,» [En línea]. Available: http://www.etitudela.com/entrenadorcomunicaciones/downloads/profibusteoria.pdf. [Último acceso: 15 octubre 2015].

[14] SIEMENS, «PROFINET Descripcion del sistema,» [En línea]. Available: https://www.google.com.co/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=0CCAQFjABahUKEwiw9c6flobJAhVETCYKHTc8Bvs&url=https%3A%2F%2Fsupport.industry.siemens.com%2Fcs%2Fattachments%2F19292127%2Fprofinet_system_description_es-ES_es-ES.pdf%3Fdow. [Último acceso: 10 Novienbre 2015].

[15] A. O'Dwyer, Handbook of PI and PID controller tuning rules, imperial College Press, 2009.

[16] Astrom, PID Controllers theory design and tuning, ISA INTERNATIONAL, 1995.

[17] M. A. D. Arevalo, CONTROL DE POSICIÓN ANGULAR DE UN MOTOR DE INDUCCIÓN UTILIZANDO UN PLC SIEMENS S7-300, bucarramanga, 2015.

ANEXOS

Anexo 1: Programa de control.
Anexo 2: Configuración del enlace Profibus con el PLC.
Anexo 3: Configuración de parámetros en el variador de velocidad.
Anexo 4: Creación de la HMI en *SIMATIC WinCC Runtime Advance*.
Anexo 5: Verificacion del servidor OPC por medio de MATLAB.
Anexo 6: Diseño de la interfaz gráfica en MATLAB.
Anexo 7: Experiencia 1. Sintonización Módulo de Posición.
Anexo 8: Experiencia 2. Identificación del Módulo de Velocidad.
Anexo 9: Experiencia 3. Sintonización Módulo de Velocidad.

Anexo 1: Programa de control

Anexo2: Configuración del enlace Profibus con un PLC

El procedimiento para configurar un enlace Profibus entre un PLC S7-300, un variador de velocidad Micromaster 440 y un Encoder RN3001 es el siguiente:

1. Adicionar los módulos que integran el PLC (Fuente de alimentación, CPU, I/O digitales y I/O Analógicas) en el software TIA Portal ilustrado en la Figura 1. Asignar las direcciones de la red Profibus en cada uno de los dispositivos utilizados en la red, junto con la dirección IP del computador y el PLC.

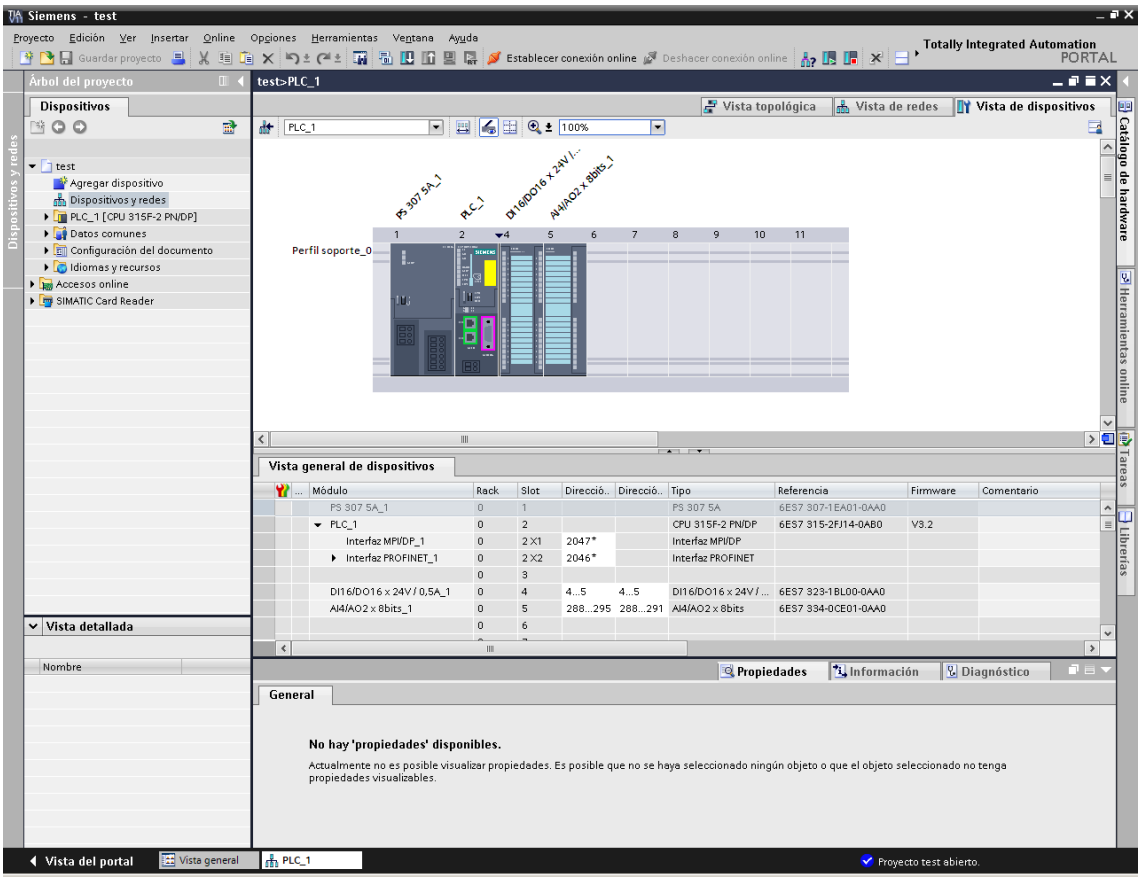


Figura 26. Configuración de los módulos que integran el PLC Siemens S7-300.

2. Adicionar los dispositivos que integran la red de Profibus. En este ejemplo, se adiciona el archivo con extensión .GSD correspondiente al Encoder RN3001 (DPV0_ifm_0E65_V4_3)[1] y el GSD del variador de velocidad Micromaster 4

[1] Ifm, «Automation made in Germany,» [En línea]. Available: http://www.ifm.com/ifmza/web/motion-control.htm. [Último acceso: 27 04 2015].

(6SE640X-1PB00-0AA0)[2] [2]. Configurar ambos dispositivos en el TIA Portal como se observa en la Figura 2 teniendo en cuenta la dirección asignada previamente por Hardware y se establecer el maestro (en este caso **PLC_1**) y crear la red Profibus virtual utilizando el TIA portal.

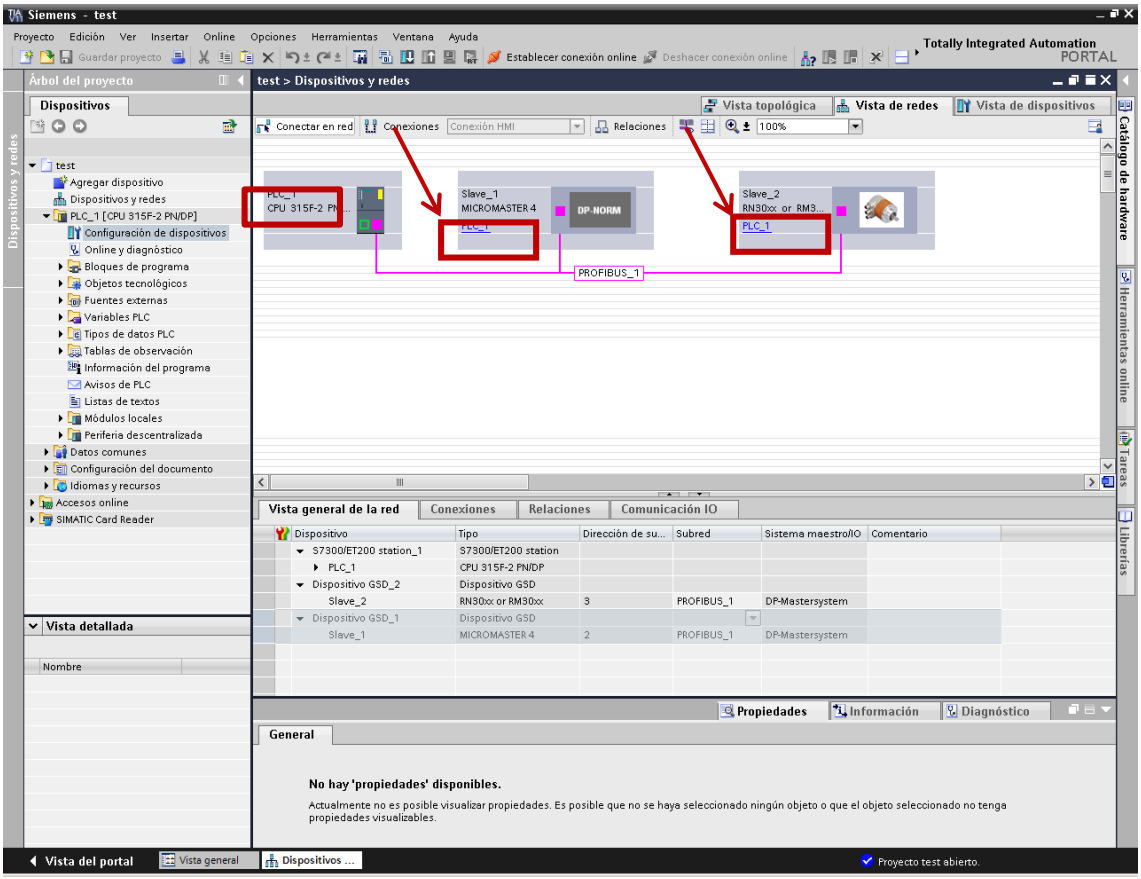


Figura 27. Dispositivos comprendidos dentro de la red Profibus.

3. En la Figura 3 se muestra la configuración de los dispositivos que fueron agregados con el archivo con extensión .GSD, donde se añaden los telegramas para los esclavos que integran la red Profibus (Encoder y Variador de velocidad), asegurando que se asigna una dirección diferente para cada dispositivo. En el caso del variador se selecciona los telegramas: *4 PKW, 2 PZD (PPO_1) _2_1* y *4 PKW, 2 PZD (PPO_1) _2_2.* Con este tipo de telegrama PPO 1 (Objeto Parámetros-datos de Proceso) se logra leer, escribir parámetros del Variador, enviar la palabra de control y leer la palabra de estado. El telegrama PPO_1 cuenta con 4 palabras del área de trabajo (PKW) y 2 palabras de área de datos de proceso (PZD). [3][3]

[2] SIEMENS AG, «Micromaster 4 GSD Files,» [En línea]. Available: https://support.industry.siemens.com/cs/#document/6567719?lc=en-WW. [Último acceso: 27 04 2015].

[3] SIEMENS AG, Manual: Micromaster- Módulo opcional PROFIBUS, 2002.

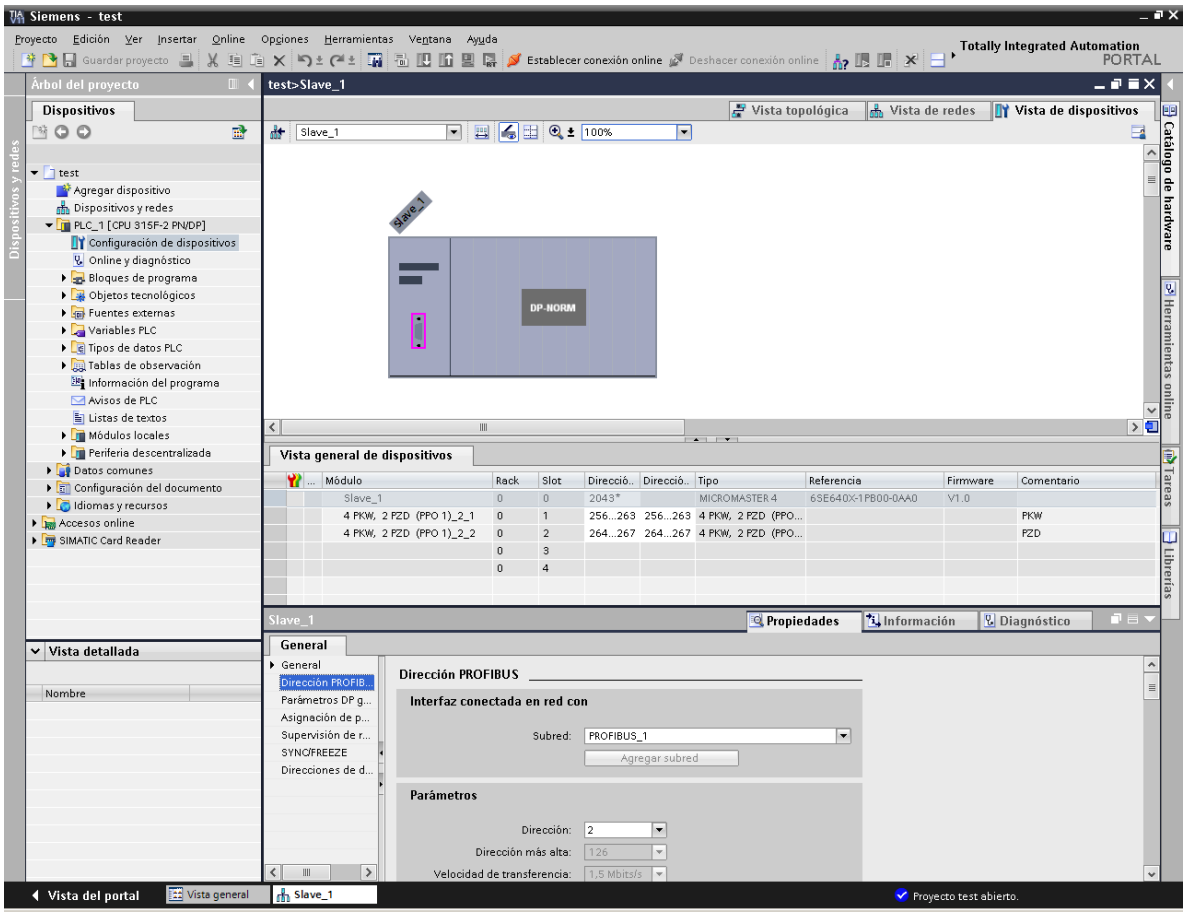


Figura 28. Configuración de los telegramas para el Variador de velocidad.

4. En la Figura 4, se observa la configuración para el Encoder absoluto, seleccionando el telegrama **ifm 2.2 Singleturn** y se arrastra hacia el Slot 1 del Encoder, nótese que las direcciones de entrada y de salida se asignan automáticamente.[4]

[4] Ifm Electronic, Device manual PROFIBUS encoder, 2013.

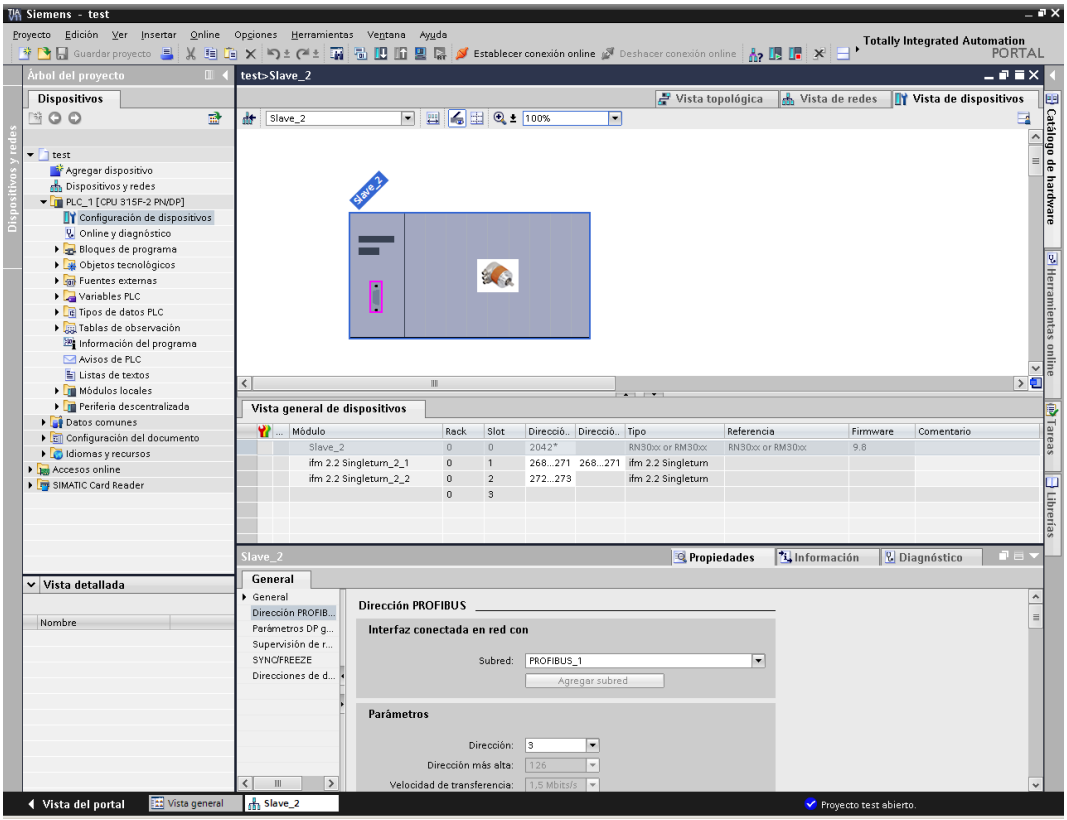


Figura 29. Configuración de los telegramas para el Encoder absoluto.

5. Para enviar la palabra de mando al variador se tienen en cuenta las direcciones asignadas previamente en el del telegrama. En la tabla 1 se muestra las distintas direcciones utilizadas en la palabra de mando y de estado del variador para el telegrama seleccionado.En la Tabla 2 se puede observar la estructura de la palabra de mando enviada del PLC al variador.

Área de parámetros (PKW)					**Área de datos de proceso (PZD)**			
Para leer/escribir valores de parámetros, p. ej. Lectura de fallos, así como lectura de información sobre las características de un parámetro.					*palabras de mando y valores de consigna de velocidad*, así como información de estado y valores reales			
Comunicación aciclica					Comunicación cíclica			
PKW					**PZD1**		**PZD2**	
	PKE: Identificador de parámetro	IND: Índice	PWE: Valor de parámetro		STW: Palabra de mando 1	HSW: Valor de consigna principal (velocidad)	ZSW: Palabra de estado	HIW: Valor real principal (vel actual)
			PWE1	PWE2				
LECT	IW256	IW258	IW260	IW262			IW264	IW266
ESCR	QW256	QW258	QW260	QW262	QW264	QW266		

Fuente: http://atsingenieros.com/2014/06/07/comunicacion-profibus-con-variador_micromaster-440-siemens/.[5]

Tabla 1. Direccionamiento utilizado en el telegrama PPO_1.

[5] Ats ingeniería, [En línea]. Available: :http://atsingenieros.com/2014/06/07/comunicacion-profibus-con-variador-micromaster-440-siemens/.. [Último acceso: 2015 04 27].

ORGANIZACIÓN PALABRA DE MANDO EN EL PLC (STD)																
WORD	MW0 ------ MOVE ----- PQW264															
BYTE	MB0								MB1							
BIT PLC	M0.7	M0.6	M0.5	M0.4	M0.3	M0.2	M0.1	M0.0	M1.7	M1.6	M1.5	M1.4	M1.3	M1.2	M1.1	M1.0
BIT VARIADOR	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Valor por defecto PLC	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF	ON	ON	ON	ON	ON	ON	ON/OFF
PALABRA DE MANDO VARIADOR MICROMASTER 420/440 SIEMENS	Local/Remote	Potenciómetro hacia abajo	Potenciómetro hacia arriba	No utilizado	Inversión del valor de consigna	Control desde PLC	Mando Izquierda	Mando Derecha	Acuse de fallo	Activar valor de consigna	Generador rampa	Activar generador rampa	Activar pulsos	OFF3: stop rápido	OFF2: stop eléctrico	ON/OFF1
VELOCIDAD ENVIADA DEL PLC AL VARIADOR (HSW)																
WORD	MW2: valor de velocidad que se le va a asignar al variador															
SALIDA AL VARIADOR	MW2 se mueve por una instrucción MOVE a la dirección PQW266															

Fuente: http://atsingenieros.com/2014/06/07/comunicacion-profibus-con-variador-micromaster-440-siemens/. [5]
Tabla 2. Estructura de la palabra de mando para el Variador de velocidad.

6. Comenzar con la programación del controlador en lenguaje Ladder en el TIA Portal siguiendo los siguientes pasos:

a. En el primer segmento del programa establece el código para se habilitar el controlador y enviar la palabra de mando de inicio hacia el variador con el pulsador PB1 asociado a “Start_Word”, el pulsador PB2 está relacionado a la palabra de mando de parada “Stop_Word” asignado a PB2 y la “SALIDA” es asignada al piloto YL1 el cual queda permanentemente activo hasta que se active el “Stop_Word”.

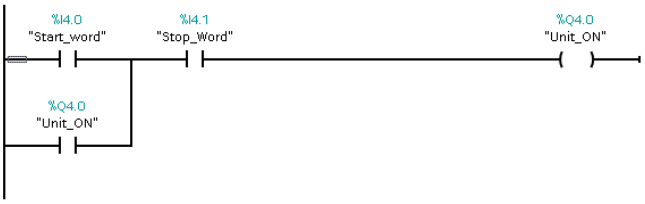


Figura 30. Inicio del controlador

b. El Segundo segmento, al activar “Stop_Word” se envía la palabra x447E (en hexadecimal) a la dirección QW264 que corresponde a la primera palabra de mando del variador como se observa en la Tabla 2. Esta palabra corresponde a una acción de parada en el Variador que debe ser activada antes de habilitar la escritura del valor de frecuencia en el variador presionando el pulsador PB1 en el paso c.

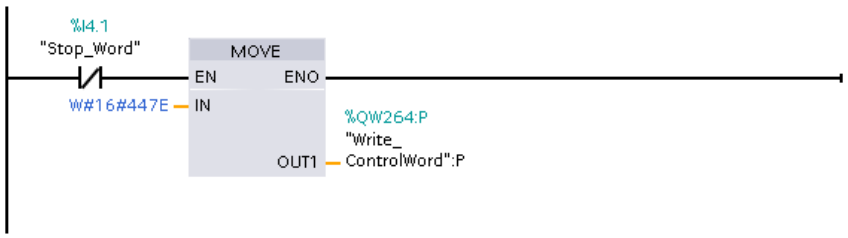


Figura 31.Envío palabra de mando en parada hacia el variador.

c. El Tercer segmento se activa con “Start_Word”, en este segmento se envía al Variador la palabra de mando x447F a la dirección. La palabra anterior realiza una acción de mando que habilita el arranque del variador y esperar una consigna de frecuencia.

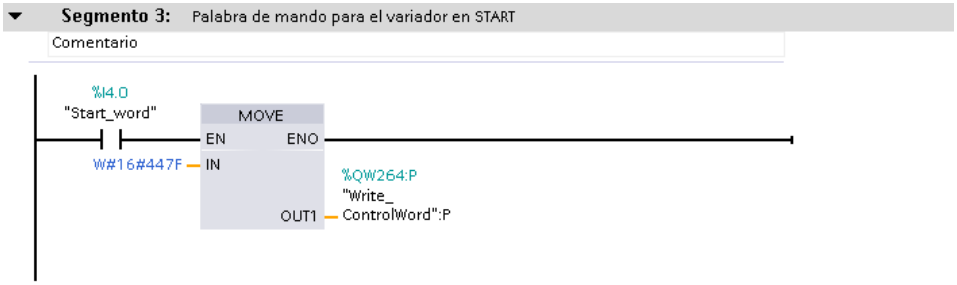


Figura 32. Envío palabra de mando en arranque hacia el variador.

d. La lectura del Encoder se realiza en la dirección asignada en el paso 4, en este caso la variable IW268 contiene un valor entero de 13 bits (valor máximo 8192), al cual se realiza una conversión a real equivalente al valor de posición en grados. Para el proceso anterior es necesario pasar de un número entero a doble entero y finalmente de número doble entero a un número real. Al realizar la conversión numérica, se procede a convertir el valor real anterior a grados mediante la siguiente ecuación:

$$Angulo = \frac{Lectura\_encoder * 360}{8192} \qquad (1)$$

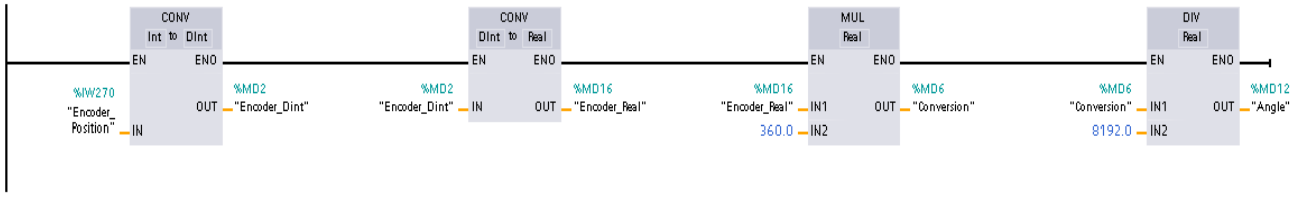


Figura 33. Conversión lectura Encoder en grados.

e. El bloque de control PID continuo se habilita con “Unit_ON” que se activa al iniciar el controlador como se observa en el paso A, y se deshabilita con “Stop_Word” en el paso B. Un interruptor llamado “Manual_On” cumple la función de conmutar el modo de operación del controlador de modo manual a automático. Cuando el mando se encuentra en modo manual el *SetPoint* se asocia a la variable “Manual_Sp” y cuando el mando se encuentra en modo automático el SP se asocia a la variable “Set_Point”. Nótese que la variable de salida del controlador se asocia a la variable “Manipulated_Value”.[6]

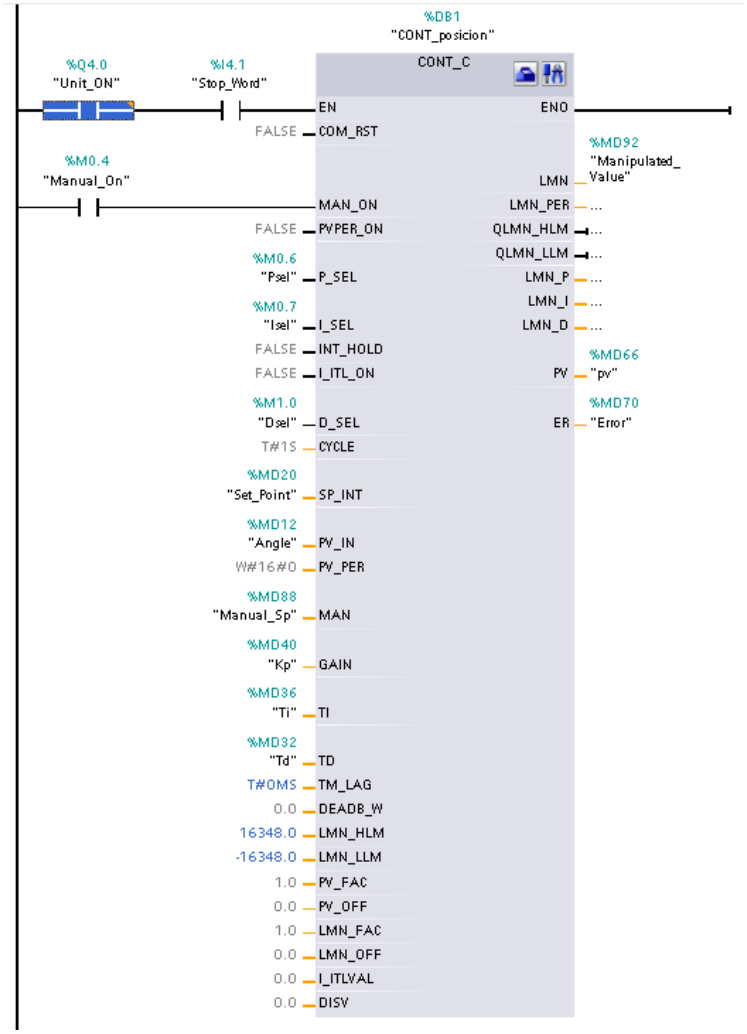


Figura 34. Controlador PID.

[6] SIEMENS AG, Manual: SIMATIC- Standard Software for S7-300 and S7-400 PID Control.

f. El envío del valor de consigna de frecuencia al Variador se realiza moviendo el valor de la acción de control hacia la dirección QW266 configurada anteriormente en el punto 4. Para escribir el valor de consigna en frecuencia en el Variador es necesario hacer una conversión de la variable de control “Manipulated_Value” de un número real a entero como se observa en la Figura 10.

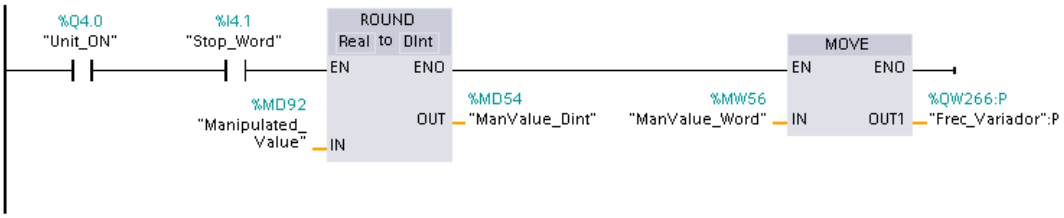
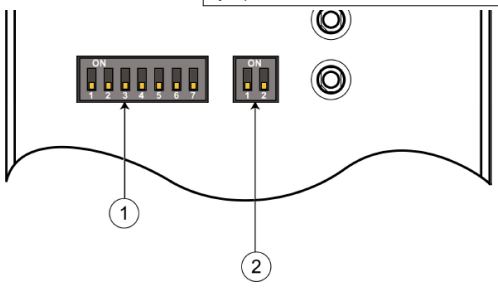


Figura 35. Conversión de la variable de control a un número entero.

Anexo 3: Configuración de parámetros en el variador de velocidad

1. Configurar la dirección que se asigna al dispositivo en la red Profibus utilizando los *DIP Switch* que posee del modulo de comunicación (ver Figura 1), la dirección asignada al variador de velocidad es 2. Posteriormente se debe reiniciar el dispositivo.

Número de interruptor:	1	2	3	4	5	6	7
Añadir a la dirección:	1	2	4	8	16	32	64
Ejemplo 1: dirección = 3 = 1 + 2	**on**	**on**	off	off	off	off	off
Ejemplo 2: dirección = 88 = 8 + 16 + 64	off	off	off	**on**	**on**	off	**on**



1 Interruptores de dirección de PROFIBUS (interruptores DIP)
2 (sólo para uso interno de Siemens)

Figura 36. Configuración de la dirección en la red Profibus del variador. [7]

2. Configurar el parámetro P0010 (Comenzar Puesta en Servicio Rápida) en 1 para iniciar la puesta en servicio rápida. Acceder a los parámetros de configuración por medio de la tecla P y se navegar mediante las flechas ubicadas en el panel frontal.

3. Iniciar la Puesta en Servicio Rápida para realizar la parametrización del motor siguiendo el diagrama mostrado en la Figura 2 e ingresando los valores indicados en la Tabla 1.

[7] SIEMENS AG, Manual: Micromaster 440 - Módulo opcional Profibus, 2002.

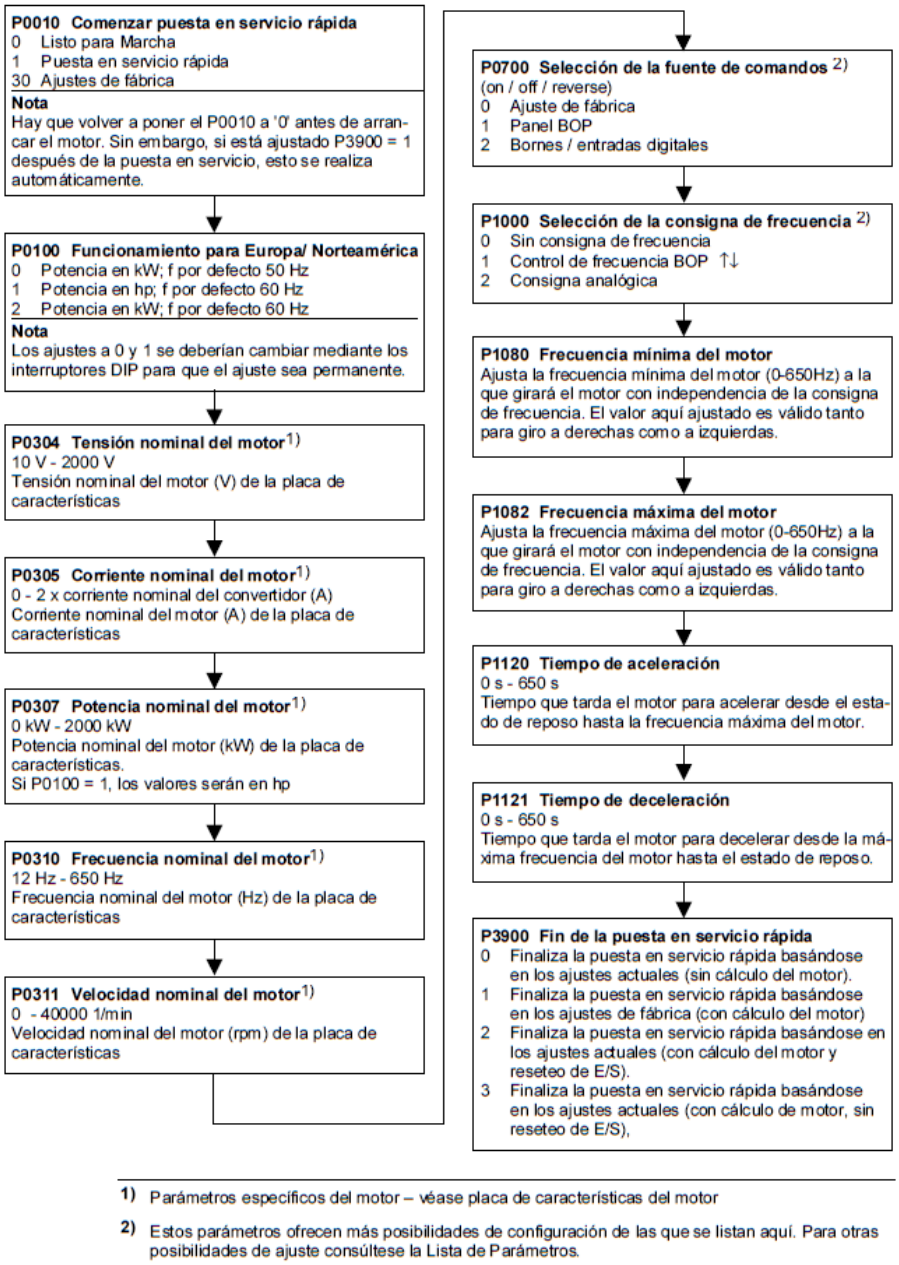


Figura 37. Organigrama puesta en servicio rápida.[8]

[8] SIEMENS AG, Manual: Micromaster 440, 2001.

Los valores de los parámetros se ajustan de acuerdo a la Tabla 1:

Parámetro	Valor	Descripción
P0003	1	Nivel de acceso del usuario
P0004	0	Filtro de parámetro
P0010	1	Comenzar puesta en servicio rápido.
P0100[9]	1	Funcionamiento para Europa/ Norte América
P0304[1]	220 V	Tensión nominal Motor
P0305[1]	1.9 A	Corriente nominal Motor
P0307[1]	0.5 hp	Potencia nominal Motor
P0310[1]	60 Hz	Frecuencia nominal Motor
P0311[1]	1590 rpm	Velocidad nominal Motor
P0700[10]	6	Selección de la fuente de comandos
P1000[2]	6	Selección de la consigna frecuencia
P1080	0 hz	Frecuencia mínima motor
P1082	60 hz	Frecuencia máxima motor
P1120	1 s	Tiempo de aceleración
P1121	2 s	Tiempo de desaceleración
P3900	1	Fin de la puesta en servicio rápida.

TABLA 3. VALORES DE CONFIGURACIÓN VARIADOR DE VELOCIDAD.

Adicional al valor de configuración asignado para el parámetro P0700 en la tabla 1. Este parámetro puede ser configurado según la fuente de comandos utilizada como se indica en la Tabla 2.

Valor	Descripción
0	Ajuste por defecto de fábrica
1	Panel frontal
2	Terminal
4	Comunicación serial en el panel frontal
5	Comunicación serial en conexión COM
6	Profibus

TABLA 4. VALORES DE CONFIGURACIÓN PARA EL PARÁMETRO P0700.[11]

Al igual que en el parámetro anterior, el parámetro P1000 correspondiente al medio de selección de la consigna puede ser configurado como se indica en la Tabla 3.

[9] Parámetros de placa del motor.

[10] Parámetro necesario para el control y acceso al variador utilizando una red Profibus.

[11] SIEMENS AG, Manual: Micromaster 440- Lista de parámetros, 2001.

Valor	Descripción
1	Consigna motor por potenciómetro entrada Analógica 2
3	Consigna de frecuencia Fija
4	Comunicación serial en el panel frontal
5	Comunicación serial en conexión COM
6	Profibus

TABLA 5. VALORES DE CONFIGURACIÓN PARA EL PARÁMETRO P0700. [1]

4. Tras realizar el procedimiento anterior, se recomienda reiniciar el dispositivo.

Anexo 4: Creación de la HMI en SIMATIC WinCC Runtime Advanced.

En el proceso de la creación de la HMI en *SIMATIC WinCC Runtime Advanced* se deben de seguir los siguientes pasos en el software TIA portar:

1. Agregar en el programa de control el modulo PC-System y seleccionar la opción WinCC RT Advanced.

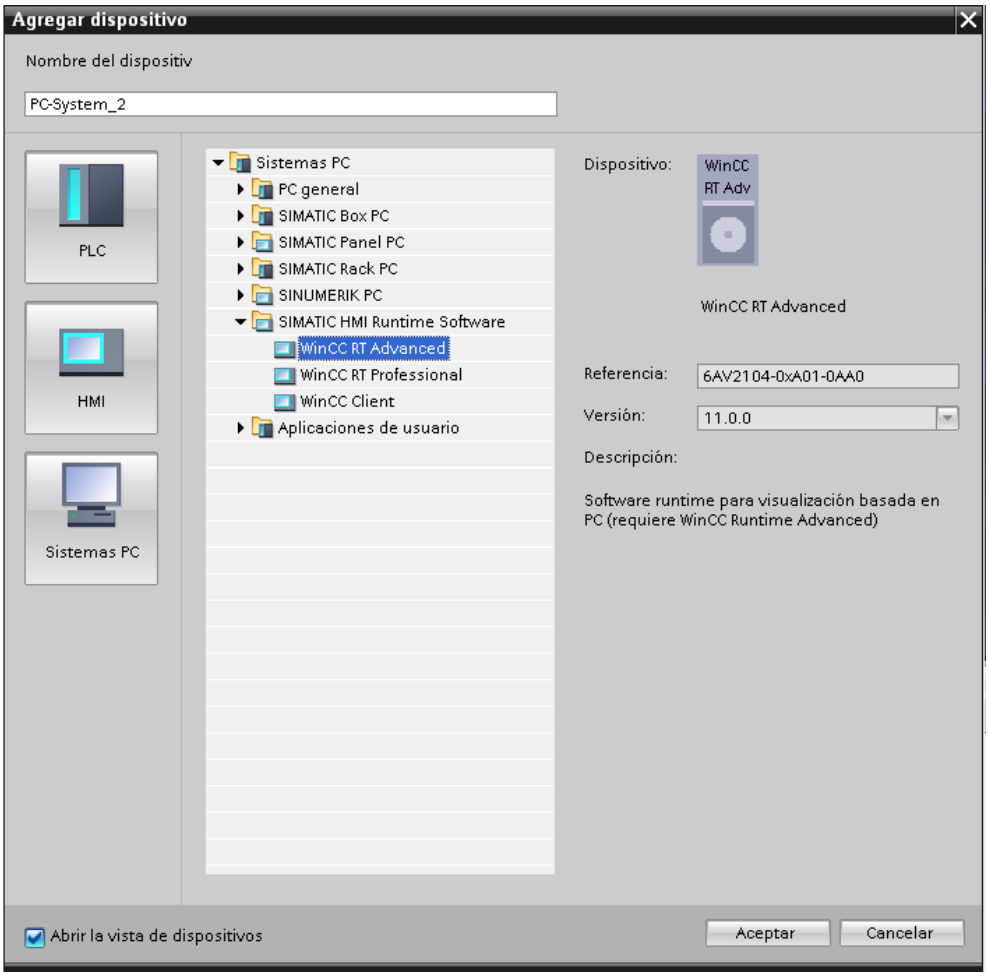


Figura 1. Agregar el modulo PC-System.

2. En el módulo de PC- System agregar un un módulo de comunicación TCP/IP para configurar la dirección IP y tener una comunicación con el modulo principal del PLC, se selecciona el módulo de comunicación IE general. Las direcciones IP en el módulo de comunicación IE general y en el PLC deben de ser diferentes.

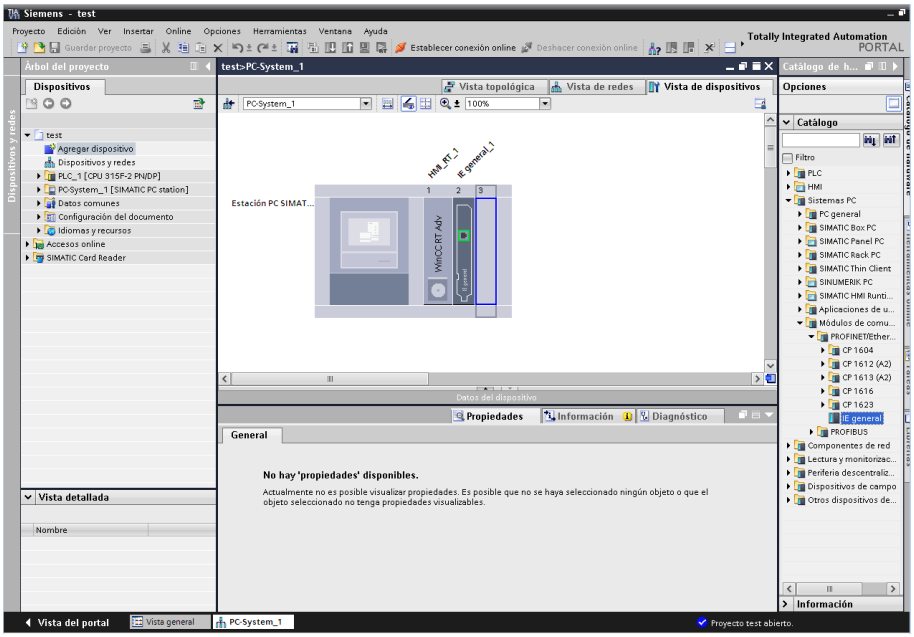


Figura 2. Módulo de comunicación IE general agregado al PC-System.

3. Crear una imagen en la HMI en *SIMATIC WinCC Runtime Advanced*, se selecciona agregar nueva imagen en la carpeta de imágenes en el árbol de proyecto del módulo PC- System.

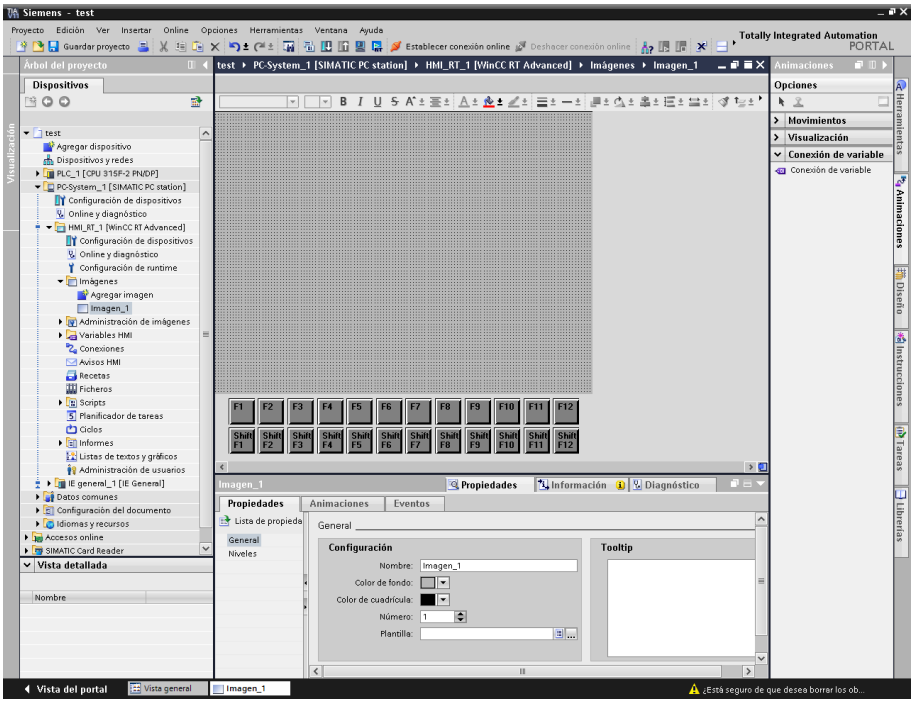


Figura 3. Creación de una imagen para la HMI de visualización.

4. En la parte derecha de la ventana de trabajo de TIA portal se encuentra la viñeta herramientas en donde se pueden seleccionar los objetos de trabajo con los que la HMI interactúa, estos se agregan a la imagen de la HMI arrastrando y posicionando de la mejor forma que nos parezca, entre los objetos que podemos seleccionar se encuentran: cuadros de texto, botones, barras, indicadores, tanques, graficas entre otros.

Figura 4. Selección de objetos para la HMI.

5. Seleccionamos el objeto a modificar y en el panel de propiedades, opción variables seleccionamos variables del PLC, asignamos la variable con la que deseamos interactuar, automáticamente el programa crea una variable HMI asociada a la variable del PLC que es modificada por los eventos del objeto.

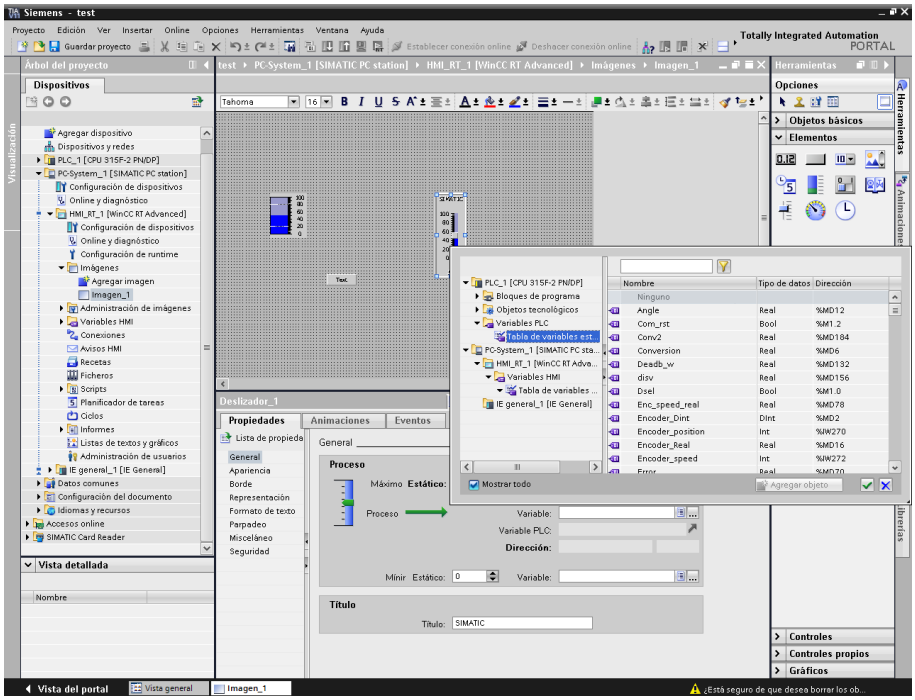


Figura 5. Asignación de la variable del PLC al objeto de la HMI.

6. Configurar las animaciones y eventos de los objetos de la HMI en *SIMATIC WinCC Runtime Advanced,* en las pestañas de animaciones y eventos. Para seleccionar una animación o evento no es necesario que el objeto tenga asociada una variable del PLC.

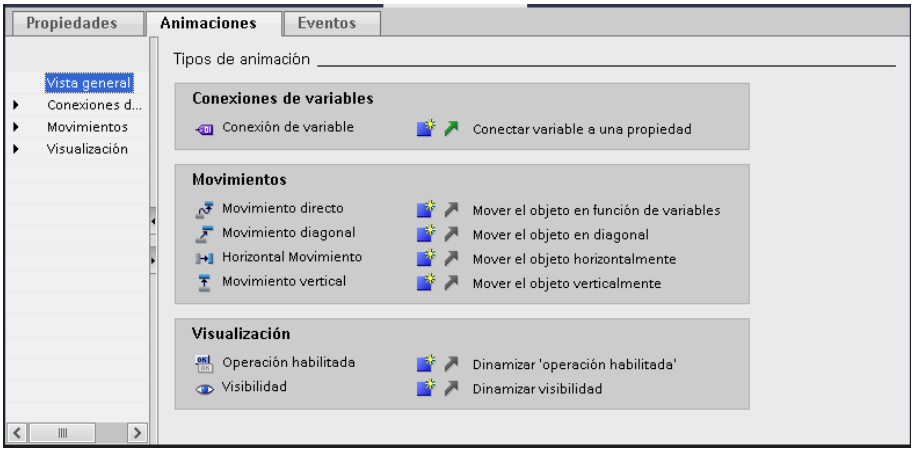


Figura 6. Propiedades de animación de los objetos de la HMI.

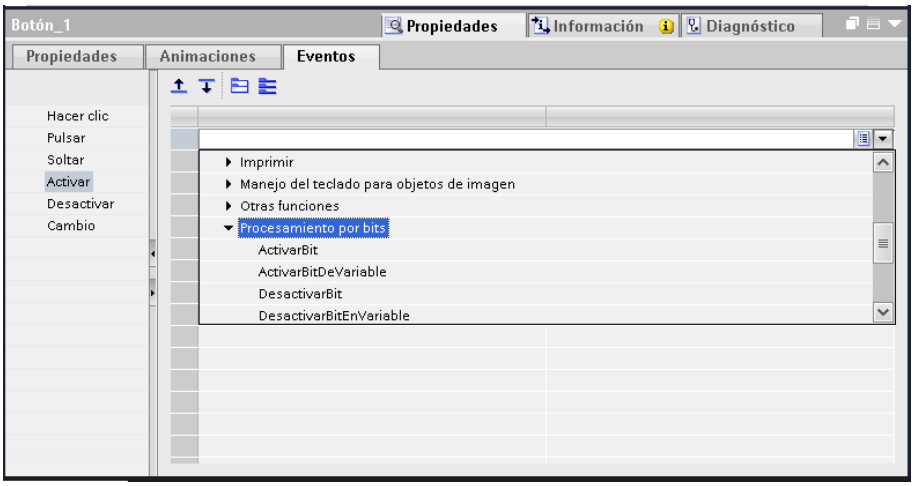


Figura 7. Propiedades de eventos de los objetos de la HMI.

7. En el árbol del proyecto del módulo PC- System se visualizan las variables de la HMI en una tabla de variables en la carpeta variables HMI, en esta tabla se visualizan todos los parámetros de las variables HMI asociadas a las variables del PLC.

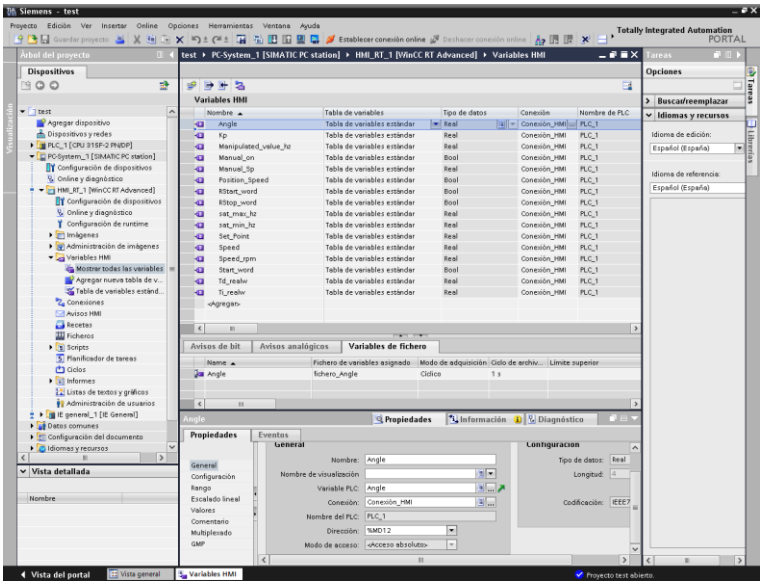


Figura 8. Tabla de variables HMI en SIMATIC WinCC Runtime Advanced.

8. Crear un fichero es opcional ya que este no afecta el funcionamiento de la HMI en *SIMATIC WinCC Runtime Advanced* pero, permite crear un registro de los valores de las variables que contiene la tabla de variables HMI. Para crear un fichero se selecciona la carpeta fichero del árbol del proyecto del módulo PC-System y agregamos un fichero nuevo, seleccionamos el fichero creado y agregamos una variable HMI, en las propiedades seleccionamos la forma de registro que deseamos.

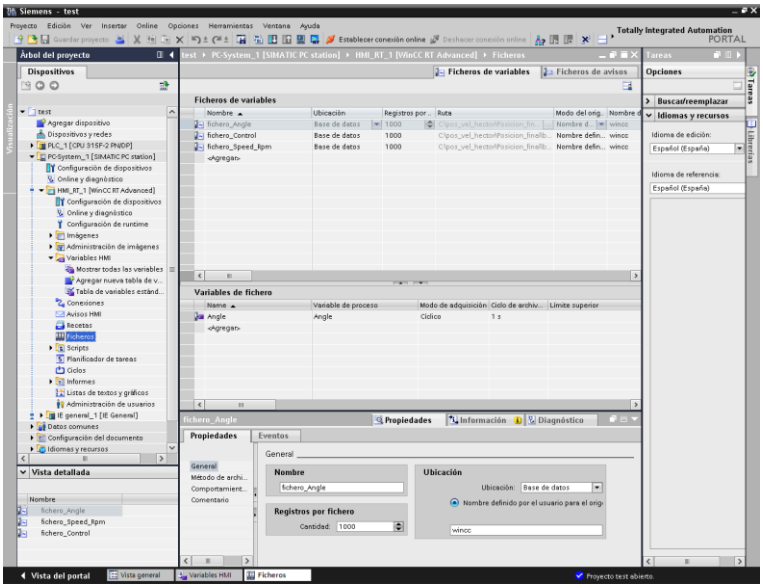


Figura 9. Creación de ficheros

9. Configurar la HMI en *SIMATIC WinCC Runtime Advanced* como servidor OPC, se selecciona en el árbol de proyecto configuración *runtime* y en servicios se elige actuar como servidor OPC.

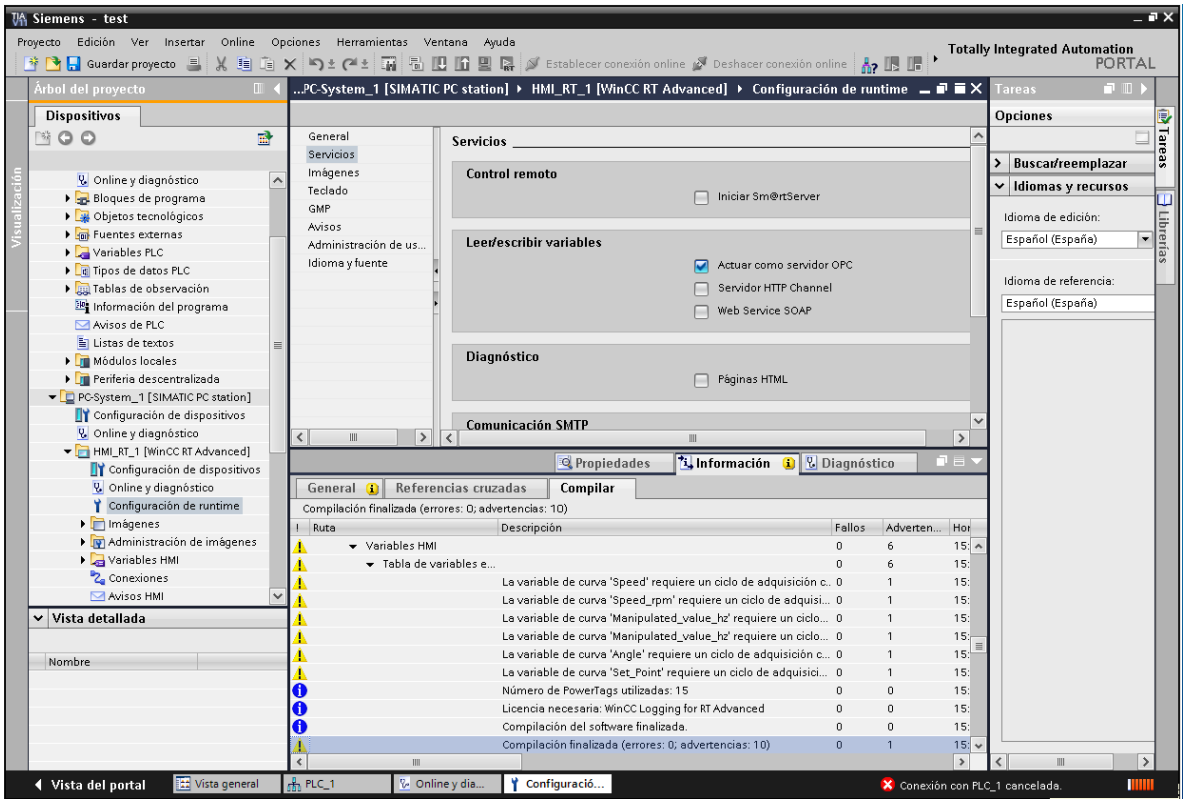


Figura 10. Configuración de la HMI en SIMATIC WinCC Runtime Advanced como servidor OPC.

Anexo 5: Verificación del servidor OPC por medio de MATLAB

Para verificar el correcto funcionamiento del servidor OPC de *SIMATIC WinCC Runtime Advanced* por medio de la herramienta OPC de MATLAB, se siguen los siguientes pasos:

1. Se abre MATLAB y en la parte superior en la pestaña APPS se selecciona la aplicación OPC client.

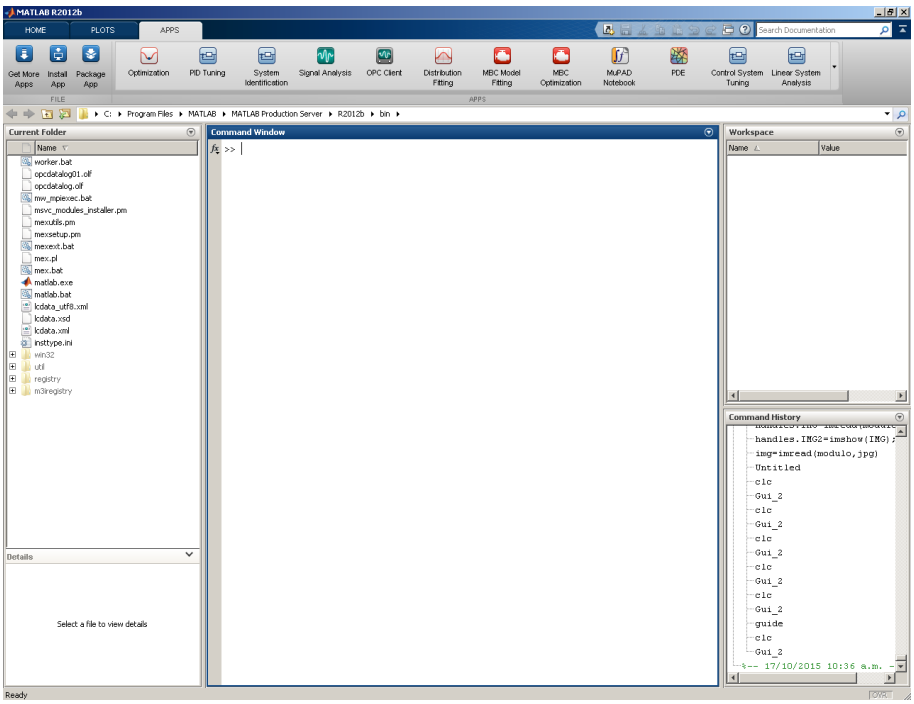


Figura 1. Abrir la aplicación OPC client.

2. Se abre la ventana de la aplicación OPC Client de la herramienta OPC de MATLAB, donde observamos dos paneles de trabajo.

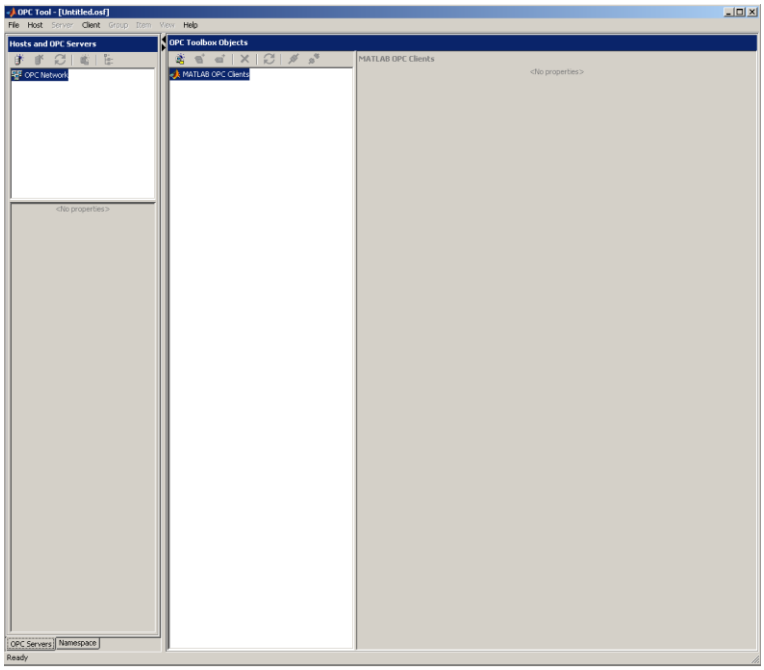


Figura 2. Ventana de trabajo de la aplicación OPC client

3. Se selecciona el panel OPC Toolbox objects, se agrega el servidor local y seleccionamos el nombre del servidor de *SIMATIC WinCC Runtime Advanced* (OPC.SimaticHMI. CoRtHmiRTm.1).

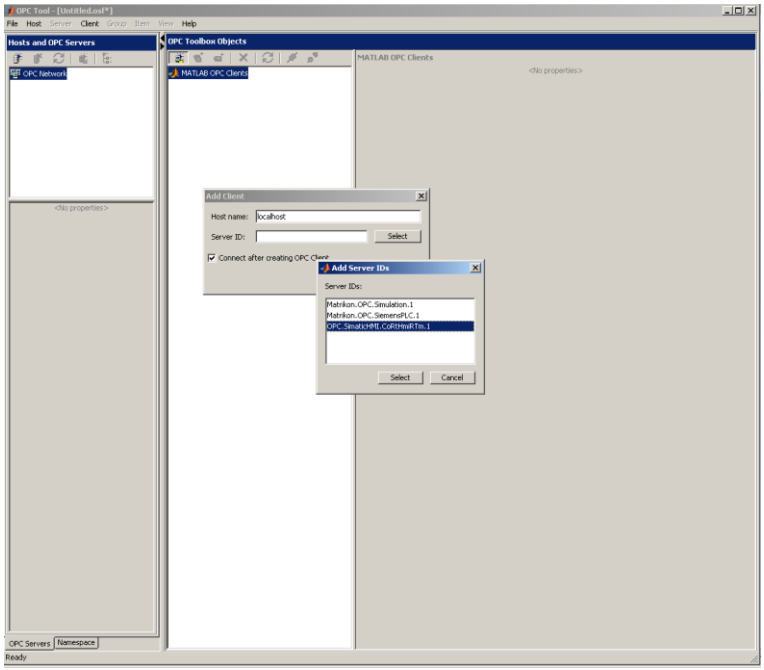


Figura 3. Agregar el servidor y el cliente OPC.

4. Al agregar el cliente OPC el servidor OPC es agregado y conectado automáticamente.

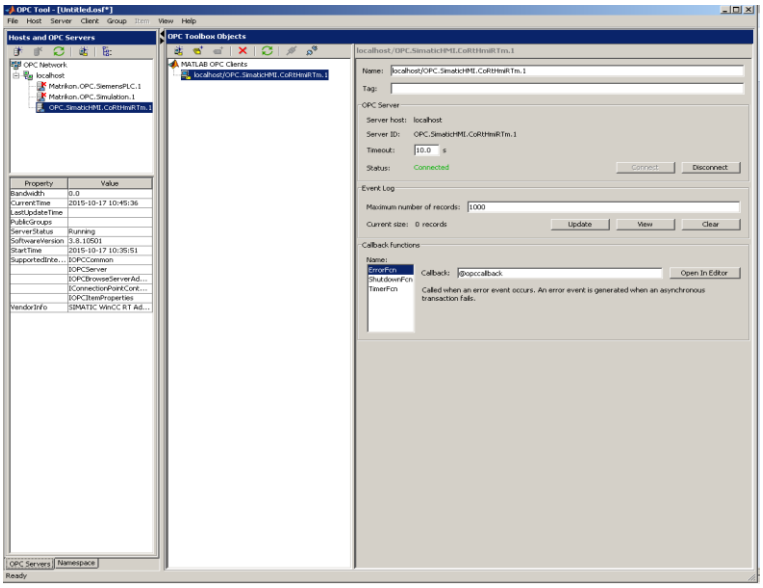


Figura 4. Conexión del servidor OPC

5. En el panel de OPC Toolbox objects se agrega un objeto de grupo donde se guardan las variables del servidor OPC de *SIMATIC WinCC Runtime Advanced*.

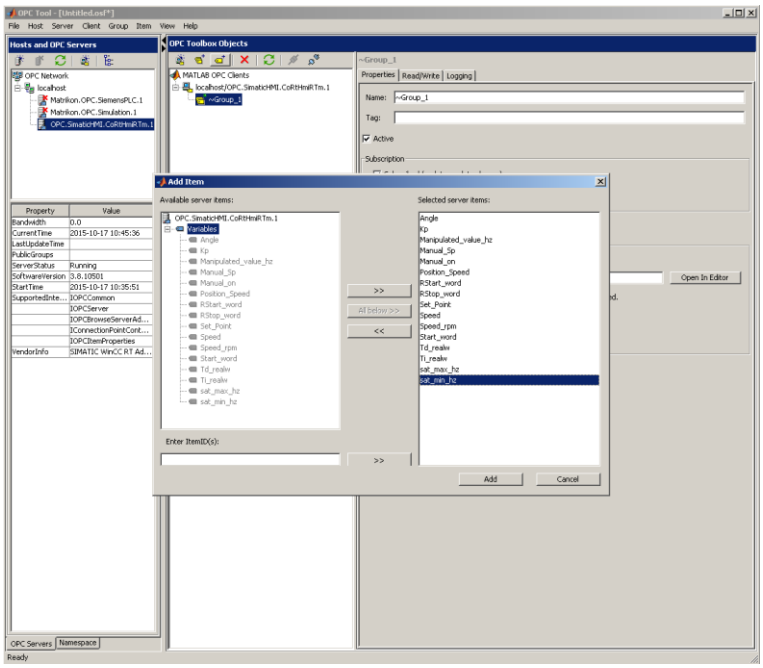


Figura 5. Agregar las variables del servidor OPC al OPC Toolbox objects.

6. Al agregar las variables del servidor OPC en el OPC Toolbox objects se vizualiza en la pestaña read/write los valores de las lecturas de las variables en tiempo real verificando la conexión con el servidor OPC.

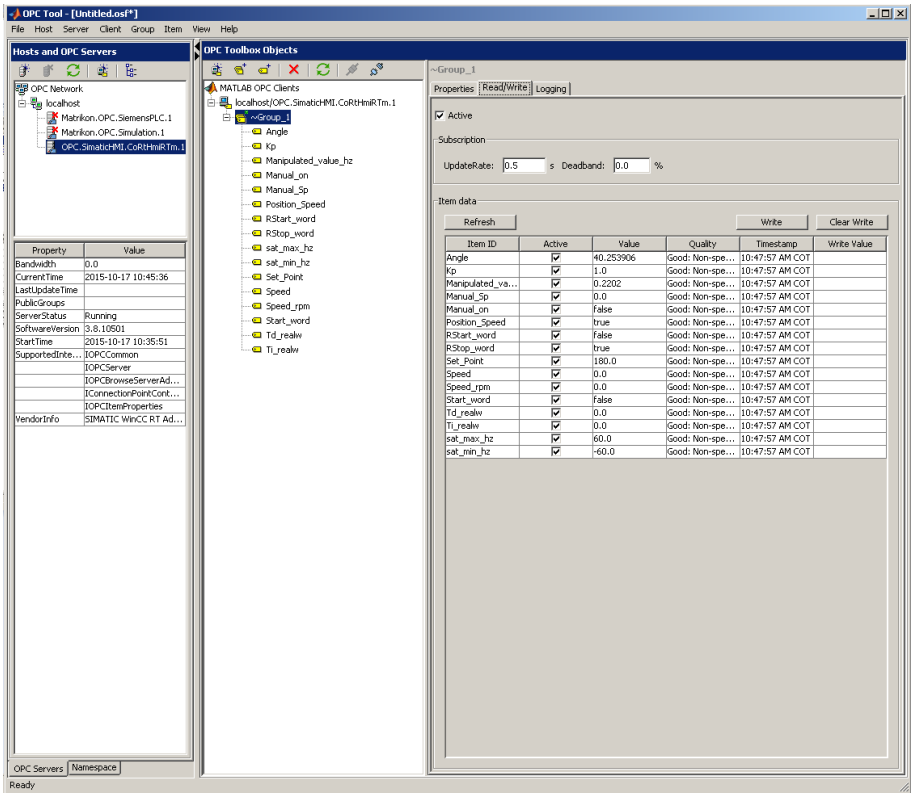


Figura 6. Visualización de las variables del servidor OPC en el OPC Toolbox objects

Anexo 6: Diseño de la interfaz gráfica en MATLAB

Para el diseño de la interfaz gráfica en MATLAB se utiliza la herramienta *GUIDE* la cual nos facilita su desarrollo de la siguiente manera:

1. Se abre MATLAB y escribimos guide en el espacio de trabajo o creamos un nuevo proyecto seleccionando New – Graphical user interface, abrirá una ventana de dialogo se selecciona Blank GUI y la dirección de donde guardar el archivo
   .

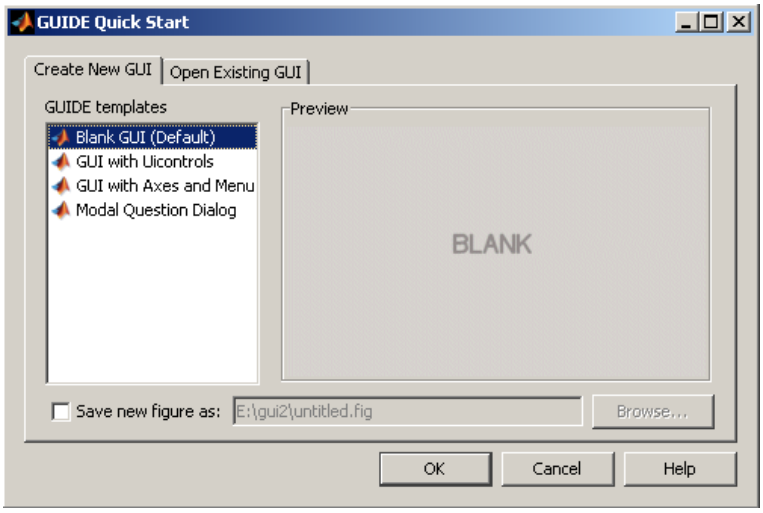


Figura 1. Crear una GUI en blanco

2. Al seleccionar Blank GUI se cargara la ventana de trabajo en donde se diseña la interfaz gráfica de MATLAB, los objetos de la interfaz gráfica se agregan arrastrando desde la parte superior izquierda al espacio de trabajo

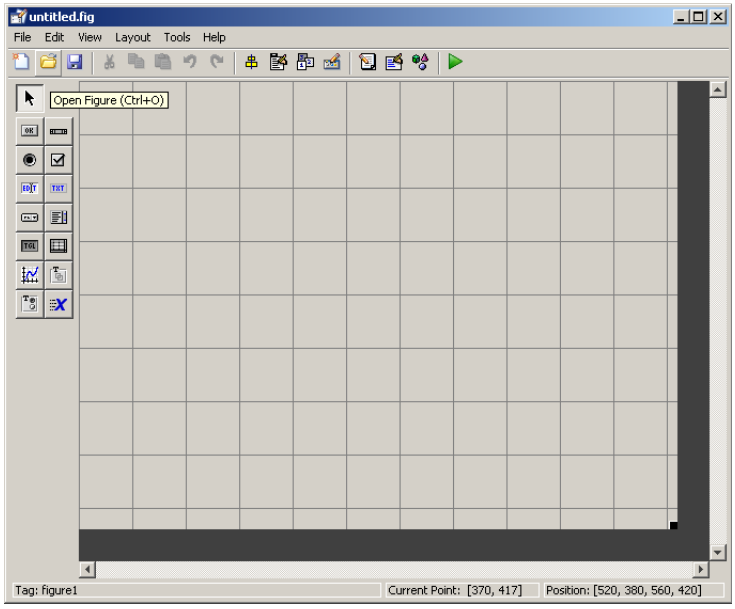


Figura 2. Ventana de trabajo de Guide.

3. Se diseña el espacio de trabajo base para la implementación de la interfaz gráfica cargando los diferentes paneles ah configurar.

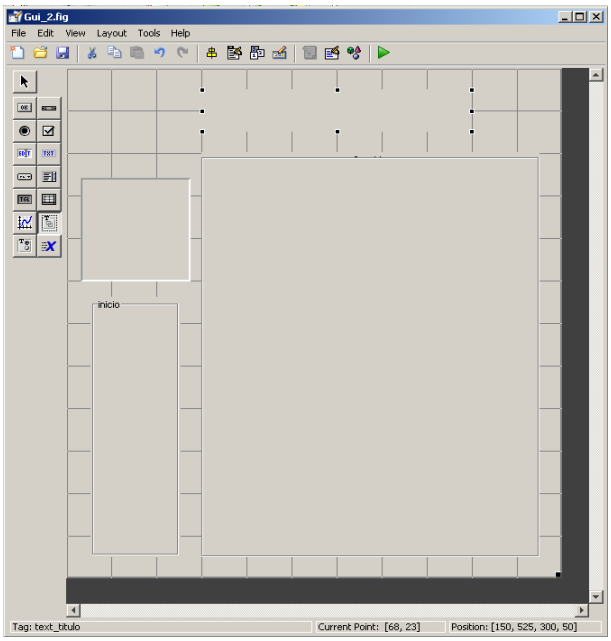


Figura 3. Espacio de trabajo de la interfaz gráfica.

4. Se agregan los botones (*push button)* de selección, el encabezado principal y los paneles a configurar uno encima del otro en la misma posición, para seleccionar cual panel trabajar damos *click* derecho*, Bring to Front* o *Send to Back,* esto nos permite mover al frente o al fondo el panel que deseemos configurar

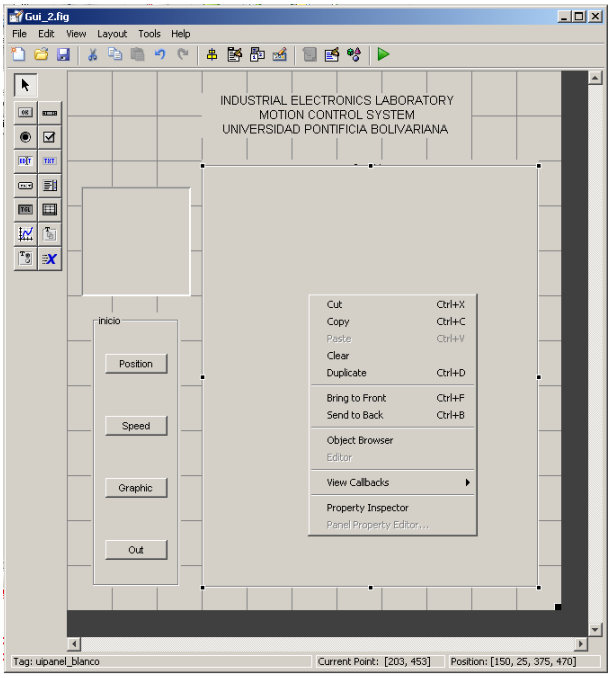


Figura 4. Paneles de configuración.

5. Se agregan los objetos de trabajo como los *slider, push button, radio button, edit text, static text, button group* y *axes,* correspondientemente a cada panel.

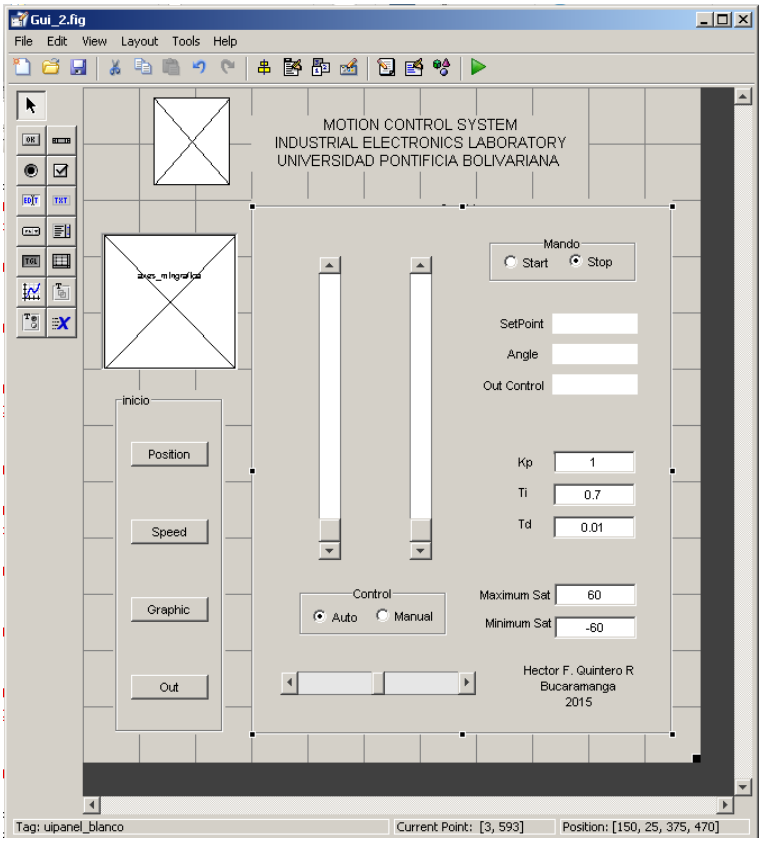


Figura 5. Interfaz gráfica panel de trabajo.

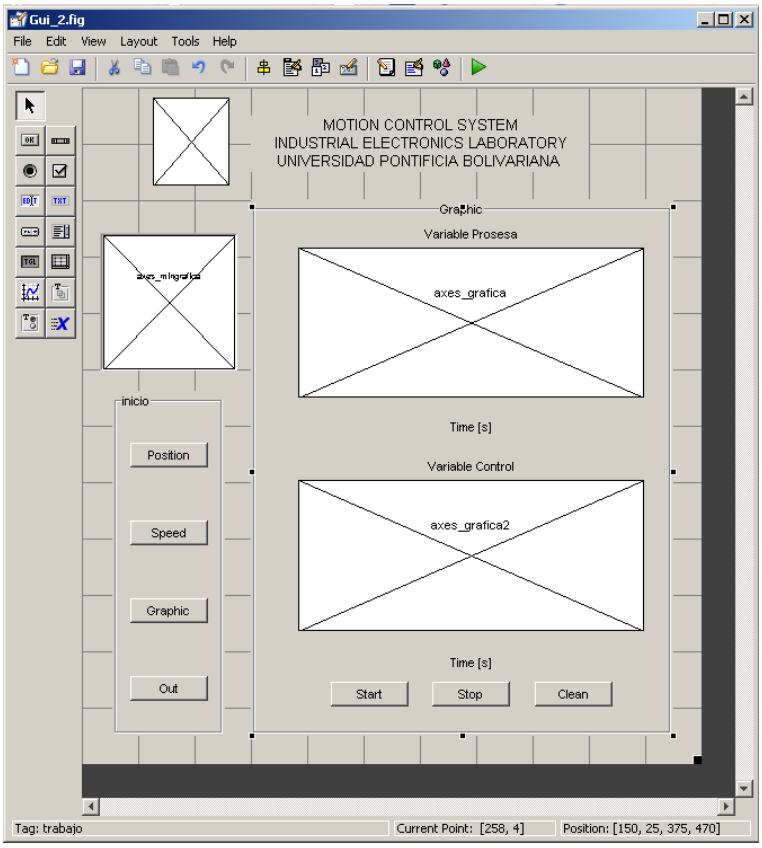


Figura 6. Interfaz gráfica panel de gráficas.

6. Los parámetros de cada objeto se pueden configurar por medio de la herramienta inspector, coda objeto difiere algunos de sus parámetros pero los más comunes y utilizados son: tag, position, title, value, max, min, entre otros.

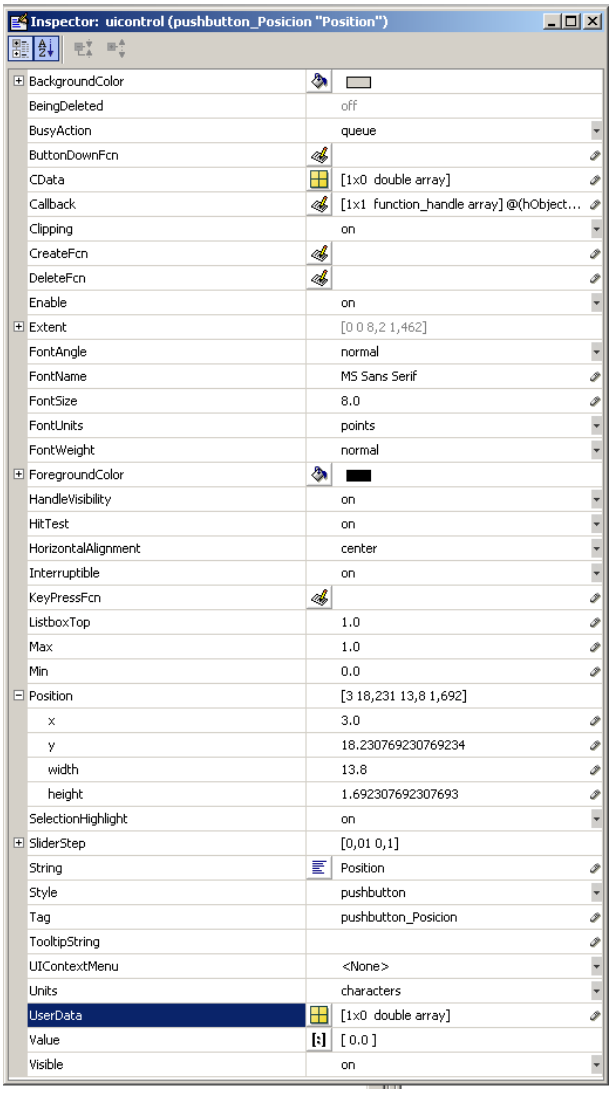


Figura 7. Inspector de Guide

7. Teniendo el diseño de la interfaz gráfica y todos los parámetros de los objetos configurados, se da clic en el icono editor, este abre automáticamente el código de la función de la interfaz gráfica, en este código se agrega las ordenes de la herramienta OPC de MATLAB para establecer la comunicación OPC entre la interfaz gráfica de MATLAB y el TIA portal. A continuación se muestra el código de la interfaz gráfica de MATLAB con las órdenes de la herramienta OPC incluidas.

```
function varargout = Gui_2(varargin)
%GUI_2 M-file for Gui_2.fig
%      GUI_2, by itself, creates a new GUI_2 or raises the existing
%      singleton*.
%
%      H = GUI_2 returns the handle to a new GUI_2 or the handle to
%      the existing singleton*.
%
%      GUI_2('Property','Value',...) creates a new GUI_2 using the
%      given property value pairs. Unrecognized properties are passed via
%      varargin to Gui_2_OpeningFcn.  This calling syntax produces a
%      warning when there is an existing singleton*.
%
%      GUI_2('CALLBACK') and GUI_2('CALLBACK',hObject,...) call the
%      local function named CALLBACK in GUI_2.M with the given input
%      arguments.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Gui_2

% Last Modified by GUIDE v2.5 09-Nov-2015 15:51:10

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
'gui_Singleton',  gui_Singleton, ...
'gui_OpeningFcn', @Gui_2_OpeningFcn, ...
'gui_OutputFcn',  @Gui_2_OutputFcn, ...
'gui_LayoutFcn',  [], ...
'gui_Callback',   []);
if nargin && ischar(varargin{1})
gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
[varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before Gui_2 is made visible.
function Gui_2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   unrecognized PropertyName/PropertyValue pairs from the
%            command line (see VARARGIN)

% Choose default command line output for Gui_2
handles.output = hObject;
%&% inicializacion de los paneles
set(handles.trabajo,'visible','off');% set, asigna el valor o la
configuracion del parametro al objeto selecionado
set(handles.uipanel_grafica,'visible','off');
set(handles.axes_mingrafica,'visible','off');
set(handles.uipanel_blanco,'visible','on');

%%%creacion del objeto timer
handles.timer=timer('TimerFcn',{@user_timercallback,hObject},...
'ExecutionMode','fixedSpacing',...
'Period',0.05);
%%% creacion e inializacion de variables globales
global i;
global t;
global out;
global out2;


i=0;
out=[  i];
out2=[  i];
t=[  i];

%%% configuracion de las señales y apuntadores de las
%%% grafiacas
handles.signal=line('parent',handles.axes_grafica,...
'Xdata',[],...
'Ydata',[],...
'color',[0 0 1],...
'EraseMode','background');

handles.dot=line('parent',handles.axes_grafica,...
'Xdata',[],...
'Ydata',[],...
'marker','o',...
'markeredgecolor','r',...
'EraseMode','background');
```

```
82.   handles.signal2=line('parent',handles.axes_grafica2,...
83.   'Xdata',[],...
84.   'Ydata',[],...
85.   'color',[1 0 0],...
86.   'EraseMode','background');

87.   handles.dot2=line('parent',handles.axes_grafica2,...
88.   'Xdata',[],...
89.   'Ydata',[],...
90.   'marker','o',...
91.   'markeredgecolor','b',...
92.   'EraseMode','background');

93.   set(handles.axes_grafica2,'ylim',[-0.25 0.25]);

94.   % Update handles structure
95.   guidata(hObject, handles);

96.   % UIWAIT makes Gui_2 wait for user response (see UIRESUME)
97.   % uiwait(handles.UI);


98.   % --- Outputs from this function are returned to the command line.
99.   function varargout = Gui_2_OutputFcn(hObject, eventdata, handles)
100.  % varargout  cell array for returning output args (see VARARGOUT);
101.  % hObject    handle to figure
102.  % eventdata  reserved - to be defined in a future version of MATLAB
103.  % handles    structure with handles and user data (see GUIDATA)

104.  % Get default command line output from handles structure
105.  varargout{1} = handles.output;

106.  % --- Executes on slider movement.
107.  function slider_SP_Callback(hObject, eventdata, handles)% se asigna
      el valor del slider a las setpoin de cada uno de los diferentes modulos
      dependiendo de su seleccion

108.  x=get(hObject,'Value'); % get, estrae el valor del parametro del
      objeto seleccionado y lo asigna a la variable indicada
109.  set(handles.text_SP,'String',x);

110.  da=opcda('localhost','OPC.SimaticHMI.CoRtHmiRTm.1');% opcda, crea
      un objeto OPC(DA) como servidor OPC

111.  connect(da);% connect, conecta el objeto OPC(DA)con el cliente de
      MATLAB
112.  grp=addgroup(da,'Group_1');% addgroup, crea un objeto dentro del
      cliente como arreglo donde estaran incluidas las variables del servidor
      OPC
```

```
113.  set(grp, 'UpdateRate', 1);% UpdateRate, determina la frecuencia con
      la que suceden los eventos para cada variable en el grupo
114.  Set_Point = additem(grp, 'Set_Point');% additem, agrega un objeto
      como variable incluida en el grupo
115.  Speed = additem(grp, 'Speed');

116.  sel=get(handles.trabajo,'title');

117.  if strcmpi(sel,'Sistem Control Position');% strcmpi, compara una
      variable chart con otra
118.  write(Set_Point,x);% write, sobre escrible el valor de la primer
      dato con el segundo
119.  else
120.  if strcmpi(sel,'Sistem Control Speed');
121.  write(Speed,x);
122.  else
123.  set(handles.text_SP,'String','error');
124.  end
125.  end

126.  disconnect(da);% disconnect, desconecta el objeto OPC(DA)con el
      cliente de MATLAB

127.  % --- Executes during object creation, after setting all
      properties.
128.  function slider_SP_CreateFcn(hObject, eventdata, handles)

129.  if isequal(get(hObject,'BackgroundColor'),
      get(0,'defaultUicontrolBackgroundColor'))
130.  set(hObject,'BackgroundColor',[.9 .9 .9]);
131.  end

132.  % --- Executes on slider movement.
133.  function slider_PV_Callback(hObject, eventdata, handles)

134.  % --- Executes during object creation, after setting all
      properties.
135.  function slider_PV_CreateFcn(hObject, eventdata, handles)

136.  if isequal(get(hObject,'BackgroundColor'),
      get(0,'defaultUicontrolBackgroundColor'))
137.  set(hObject,'BackgroundColor',[.9 .9 .9]);
138.  end

139.  function edit_Kp_Callback(hObject, eventdata, handles)%sobre
      escribe el valor asignado en la variable X sobre la variable Kp del
      servidor OPC

140.  x=get(hObject,'String');
```

```
141.  da=opcda('localhost','OPC.SimaticHMI.CoRtHmiRTm.1');
142.  connect(da);
143.  grp=addgroup(da,'Group_1');
144.  set(grp, 'UpdateRate', 1);
145.  Kp = additem(grp, 'Kp');

146.  write(Kp,x);

147.  disconnect(da)

148.  % --- Executes during object creation, after setting all
      properties.
149.  function edit_Kp_CreateFcn(hObject, eventdata, handles)

150.  if ispc && isequal(get(hObject,'BackgroundColor'),
      get(0,'defaultUicontrolBackgroundColor'))
151.  set(hObject,'BackgroundColor','white');
152.  end

153.  function edit_Ti_Callback(hObject, eventdata, handles)%sobre
      escribe el valor asignado en la variable X sobre la variable Ti del
      servidor OPC

154.  x=get(hObject,'String');

155.  da=opcda('localhost','OPC.SimaticHMI.CoRtHmiRTm.1');
156.  connect(da);
157.  grp=addgroup(da,'Group_1');
158.  set(grp, 'UpdateRate', 1);
159.  Ti_realW = additem(grp, 'Ti_realW');

160.  write(Ti_realW,x);

161.  disconnect(da);

162.  % --- Executes during object creation, after setting all
      properties.
163.  function edit_Ti_CreateFcn(hObject, eventdata, handles)

164.  if ispc && isequal(get(hObject,'BackgroundColor'),
      get(0,'defaultUicontrolBackgroundColor'))
165.  set(hObject,'BackgroundColor','white');
166.  end

167.  function edit_Td_Callback(hObject, eventdata, handles)%sobre
      escribe el valor asignado en la variable X sobre la variable Td del
      servidor OPC

168.  x=get(hObject,'String');
```

```
da=opcda('localhost','OPC.SimaticHMI.CoRtHmiRTm.1');
connect(da);
grp=addgroup(da,'Group_1');
set(grp, 'UpdateRate', 1);
Td_realw = additem(grp, 'Td_realw');

write(Td_realw,x);

disconnect(da);

% --- Executes during object creation, after setting all
properties.
function edit_Td_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

function edit_SMax_Callback(hObject, eventdata, handles)%sobre
escribe el valor asignado en la variable X sobre la variable sat_max_hz
del servidor OPC

x=get(hObject,'String');

da=opcda('localhost','OPC.SimaticHMI.CoRtHmiRTm.1');
connect(da);
grp=addgroup(da,'Group_1');
set(grp, 'UpdateRate', 1);
sat_max_hz = additem(grp, 'sat_max_hz');

write(sat_max_hz,x);

disconnect(da);

% --- Executes during object creation, after setting all
properties.
function edit_SMax_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

function edit_SMin_Callback(hObject, eventdata, handles)%sobre
escribe el valor asignado en la variable X sobre la variable sat_min_hz
del servidor OPC

x=get(hObject,'String');
```

```
197.  da=opcda('localhost','OPC.SimaticHMI.CoRtHmiRTm.1');
198.  connect(da);
199.  grp=addgroup(da,'Group_1');
200.  set(grp, 'UpdateRate', 1);
201.  sat_min_hz = additem(grp, 'sat_min_hz');

202.  write(sat_min_hz,x);

203.  disconnect(da);

204.  % --- Executes during object creation, after setting all
      properties.
205.  function edit_SMin_CreateFcn(hObject, eventdata, handles)

206.  if ispc && isequal(get(hObject,'BackgroundColor'),
      get(0,'defaultUicontrolBackgroundColor'))
207.  set(hObject,'BackgroundColor','white');
208.  end



209.  % --- Executes on slider movement.
210.  function slider_man_Callback(hObject, eventdata, handles)% se
      asigna el valor del slider a la variable de contol Manual_Sp del
      servidor OPC

211.  x=get(hObject,'Value');

212.  da=opcda('localhost','OPC.SimaticHMI.CoRtHmiRTm.1');

213.  connect(da);
214.  grp=addgroup(da,'Group_1');
215.  set(grp, 'UpdateRate', 1);
216.  Manual_Sp = additem(grp, 'Manual_Sp');

217.  write(Manual_Sp,x);

218.  disconnect(da);

219.  % --- Executes during object creation, after setting all
      properties.
220.  function slider_man_CreateFcn(hObject, eventdata, handles)

221.  if isequal(get(hObject,'BackgroundColor'),
      get(0,'defaultUicontrolBackgroundColor'))
222.  set(hObject,'BackgroundColor',[.9 .9 .9]);
223.  end



224.  % --- Executes on button press in pushbutton_Posicion.
```

```
function pushbutton_Posicion_Callback(hObject, eventdata, handles)% modifica las caracteristicas de los objetos del panel de trabajo y las graficas en el panel graficas

% configura la visisvilidad de los paneles de la intefaz grafica
set(handles.trabajo,'visible','on','title','Sistem Control Position');
set(handles.uipanel_grafica,'visible','off');
set(handles.uipanel_blanco,'visible','off');
set(handles.axes_mingrafica,'visible','on');

% combia el nombre de la variable de proceso
set(handles.text_Var,'String','Angle');

% configura los limites de los ejes de la graficas
set(handles.axes_grafica,'ylim',[0 360]);
set(handles.axes_mingrafica,'ylim',[0 360]);

% configura los limites del valor de los slider
set(handles.slider_SP,'max',360,'min',0,'value',180);
set(handles.slider_PV,'max',360,'min',0,'value',180);

da=opcda('localhost','OPC.SimaticHMI.CoRtHmiRTm.1');

connect(da);
grp=addgroup(da,'Group_1');
Position_Speed = additem(grp, 'Position_Speed');
write(Position_Speed,'true');

disconnect(da);

guidata(hObject, handles);

% --- Executes on button press in pushbutton_Velocidad.
function pushbutton_Velocidad_Callback(hObject, eventdata, handles)% modifica las caracteristicas de los objetos del panel de trabajo y las graficas en el panel graficas

% configura la visisvilidad de los paneles de la intefaz grafica
set(handles.trabajo,'visible','on','title','Sistem Control Speed');
set(handles.uipanel_grafica,'visible','off');
set(handles.uipanel_blanco,'visible','off');
set(handles.axes_mingrafica,'visible','on');

% configura los limites de los ejes de la graficas
set(handles.text_Var,'String','Speed');

% configura los limites de los ejes de la graficas
set(handles.axes_grafica,'ylim',[-50 50]);
set(handles.axes_mingrafica,'ylim',[-50 50]);
```

```
% configura los limites del valor de los slider
set(handles.slider_SP,'max',50,'min',-50,'value',0);
set(handles.slider_PV,'max',50,'min',-50,'value',0);

da=opcda('localhost','OPC.SimaticHMI.CoRtHmiRTm.1');

connect(da);
grp=addgroup(da,'Group_1');
Position_Speed = additem(grp, 'Position_Speed');
write(Position_Speed,'false');

disconnect(da);

guidata(hObject, handles);

% --- Executes on button press in pushbutton_Graficar.
function pushbutton_Graficar_Callback(hObject, eventdata, handles)
%modifica las caracteristicas de los objetos del panel de trabajo y las
graficas en el panel graficas

% configura la visisvilidad de los paneles de la intefaz grafica
set(handles.trabajo,'visible','off');
set(handles.uipanel_grafica,'visible','on');
set(handles.uipanel_blanco,'visible','off');
set(handles.axes_mingrafica,'visible','on');

% --- Executes on button press in pushbutton_salir.
function pushbutton_salir_Callback(hObject, eventdata, handles)%
finaliza la ejecuicion de la interfaz grafica

delete(handles.timer);% borra el objeto de tiempo
delete (handles.UI);% finaliza la ejecucion de la interfaz grafica

% --- Executes on button press in pushbutton_Start.
function pushbutton_Start_Callback(hObject, eventdata, handles)% da
la orden para que la variable del servidor OPC RStar_word se active

da=opcda('localhost','OPC.SimaticHMI.CoRtHmiRTm.1');

connect(da);
grp=addgroup(da,'Group_1');
set(grp, 'UpdateRate', 1);
RStart_word = additem(grp, 'RStart_word');
RStop_word = additem(grp, 'RStop_word');

write(RStart_word,1);
write(RStop_word,0);
```

```
289.  start(handles.timer);% start, inicializa la ejecucion del objeto
      timer

290.  % --- Executes on button press in pushbutton_Stop.
291.  function pushbutton_Stop_Callback(hObject, eventdata, handles)% da
      la orden para que la variable del servidor OPC RStop_word se active

292.  da=opcda('localhost','OPC.SimaticHMI.CoRtHmiRTm.1');

293.  connect(da);
294.  grp=addgroup(da,'Group_1');
295.  set(grp, 'UpdateRate', 1);
296.  RStart_word = additem(grp, 'RStart_word');
297.  RStop_word = additem(grp, 'RStop_word');

298.  write(RStart_word,0);
299.  write(RStop_word,1);

300.  stop(handles.timer); % stop, para la ejecucion del objeto timer


301.  % --- Executes on button press in pushbutton_clean.
302.  function pushbutton_clean_Callback(hObject, eventdata, handles) %
      reinicializa las variables de salida dando el efecto de limpiado en las
      graficas

303.  global i;
304.  global t;
305.  global out;
306.  global out2;

307.  i=0;
308.  out=[  i+1];
309.  out2=[  i+1];
310.  t=[  i+1];

311.  set(handles.axes_grafica,'xlim',[0 t(end)]);
312.  set(handles.signal,'xdata',t,'ydata',out);
313.  set(handles.dot,'xdata',t(end),'ydata',out(end));

314.  set(handles.axes_grafica2,'xlim',[0 t(end)]);
315.  set(handles.signal2,'xdata',t,'ydata',out2);
316.  set(handles.dot2,'xdata',t(end),'ydata',out2(end));

317.  plot(handles.axes_mingrafica,t,out);

318.  function user_timercallback(obj,event,hObject)% funcion del objeto
      timer, se ejecuta continuamente

319.  handles=guidata(hObject);
```

```
global i;
global t;
global out;
global out2;

i=i+1;

t(i)=(0.3331*i)+(0.2121);%convercion a tiempo real

da=opcda('localhost','OPC.SimaticHMI.CoRtHmiRTm.1');
connect(da);
grp=addgroup(da,'Group_1');
set(grp, 'UpdateRate', 1);

Angle = additem(grp, 'Angle');
Speed = additem(grp, 'Speed');
Manipulated_value_hz = additem(grp, 'Manipulated_value_hz');

sel=get(handles.trabajo,'title');

% seleccion de cual variable de proceso se decea leer del servidor OPC
if strcmpi(sel,'Sistem Control Position')
r=read(Angle);
else
r=read(Speed);
end

y1={r.Value};
y2=cell2mat(y1);
r2=read(Manipulated_value_hz);
w1={r2.Value};
w2=cell2mat(w1);

%registro de la variables de salida como vector de posiciones
out(i)=y2;
out2(i)=w2;

set(handles.text_PV,'String',y2);
set(handles.slider_PV,'Value',y2);
set(handles.text_CO,'String',w2);


set(handles.axes_grafica,'xlim',[0 t(end)]);
set(handles.signal,'xdata',t,'ydata',out);
set(handles.dot,'xdata',t(end),'ydata',out(end));

set(handles.axes_grafica2,'xlim',[0 t(end)]);
```

```
355.  set(handles.signal2,'xdata',t,'ydata',out2);
356.  set(handles.dot2,'xdata',t(end),'ydata',out2(end));

357.  plot(handles.axes_mingrafica,t,out);

358.  disconnect(da);

359.  guidata(hObject,handles);

360.  % --- Executes when selected object is changed in mando.
361.  function mando_SelectionChangeFcn(hObject, eventdata, handles)%
      selecciona la acion de mando con una de las variables (Rstart_word ó
      Rstop_word) del servidor OPC

362.  x=get(hObject,'String');

363.  da=opcda('localhost','OPC.SimaticHMI.CoRtHmiRTm.1');

364.  connect(da);
365.  grp=addgroup(da,'Group_1');
366.  set(grp, 'UpdateRate', 1);
367.  RStart_word = additem(grp, 'RStart_word');
368.  RStop_word = additem(grp, 'RStop_word');

369.  switch x
370.  case 'Start'
371.  write(RStart_word,1);
372.  write(RStop_word,0);
373.  start(handles.timer);
374.  case 'Stop'
375.  write(RStart_word,0);
376.  write(RStop_word,1);
377.  stop(handles.timer);
378.  end


379.  % --- Executes when selected object is changed in control.
380.  function control_SelectionChangeFcn(hObject, eventdata,
      handles)%seleccion del modo de mando del control

381.  x=get(hObject,'String');% extrae el valor de la variable

382.  da=opcda('localhost','OPC.SimaticHMI.CoRtHmiRTm.1');
383.  connect(da);
384.  grp=addgroup(da,'Group_1');
385.  set(grp, 'UpdateRate', 1);
386.  Manual_on = additem(grp, 'Manual_on');

387.  switch x
```

```
388.  case 'Manual'
389.  write(Manual_on,1);

390.  case 'Auto'
391.  write(Manual_on,0);

392.  end

393.  % --- Executes during object creation, after setting all
      properties.
394.  function axes_escudo_CreateFcn(hObject, eventdata, handles)
395.  IMG=imread('escudo UPB','jpg');
396.  IMG2=imshow(IMG);

397.  % --- Executes during object creation, after setting all
      properties.
398.  function axes_imgmodulo_CreateFcn(hObject, eventdata, handles)
399.  IMG=imread('modulo','jpg');
400.  IMG2=imshow(IMG);
```

Anexo7: Experiencia 1. Sintonización Módulo de Posición.

Sintonización Módulo de Posición

Objetivos:

- Calcular las constantes del controlador PID del módulo de posición.
- Conocer las herramientas de sintonización de MATLAB (*simulink, autotuning*).
- Utilizar la Interfaz gráfica de MATLAB para la implementación y operación del controlador PID del módulo de posición.

Herramientas de trabajo:

- Módulo de posición.
- Software de MATLAB.
- Interfaz gráfica de MATLAB.

Procedimiento:

1. Elaborar los diagramas de bloques de *simulink* para el control del módulo de posición
2. Utilizar la función de transferencia de la ecuación 1 en el bloque de la plata.

$$G(s) = \frac{2.6383}{s(0.0239s^2 + 0.2033s + 1)} \qquad (1)$$

3. Utilizar la herramienta *autotuning* del bloque PID de *simulink* y observar el comportamiento de la salida en el simulador.
4. Implementar las constantes del bloque PID de simulink en el controlador PID del módulo de posición, por medio de la interfaz gráfica de MATLAB.
5. Observar el comportamiento del módulo de posición.

Anexo 8: Experiencia 2. Identificación del Módulo de Velocidad.

Identificación del Módulo de Velocidad

Objetivos:

- Analizar la operación del módulo de velocidad en lazo abierto.
- Conocer la herramienta de identificación de MATLAB (*ident*).
- Utilizar la Interfaz gráfica de MATLAB para la implementación y operación del módulo de velocidad en lazo abierto.

Herramientas de trabajo:

- Módulo de posición.
- Software de MATLAB.
- Interfaz gráfica de MATLAB.
- Base de datos.

Procedimiento:

1. Configurar el módulo de velocidad en lazo abierto por medio de la interfaz gráfica de MATLAB, seleccionando el modo de control manual.
2. Variar los valores de la salida de control por medio del *slider* para el control manual de la interfaz gráfica de MATLAB, observar los valores de la variable de proceso (*Speed*).
3. Importar los datos de las variables de proceso desde la base de datos (archivo en Excel) a MATLAB.
4. Abrir la herramienta *ident* de MATLAB, importar los datos a ident, validar los datos y encontrar un modelo matemático del proceso con un *fit* mayor al 80%.
5. Exportar el modelo matemático del módulo de velocidad obtenido al workspace de MATLAB.

Anexo 9: Experiencia 3. Sintonización Módulo de velocidad.

Sintonización Módulo de Velocidad

Objetivos:

- Calcular las constantes del controlador PID del módulo de velocidad.
- Conocer las herramientas de sintonización de MATLAB (*simulink, autotuning*).
- Utilizar la Interfaz gráfica de MATLAB para la implementación y operación del controlador PID del módulo de velocidad.

Herramientas de trabajo:

- Módulo de velocidad.
- Software de MATLAB.
- Interfaz gráfica de MATLAB.

Procedimiento:

1. Elaborar los diagramas de bloques de *simulink* para el control del módulo de velocidad.
2. Utilizar el modelo matemático obtenido en la experiencia anterior para el bloque de la planta en simulink.
3. Utilizar la herramienta *autotuning* del bloque PID de *simulink* y observar el comportamiento de la salida en el simulador.
4. Implementar las constantes del bloque PID de simulink en el controlador PID del módulo de velocidad, por medio de la interfaz gráfica de MATLAB.
5. Observar el comportamiento del módulo de velocidad.