1

Introducción

El aumento en el poder de procesamiento de los computadores permitió que los sistemas Scada se vuelvan cada vez más comunes dentro de los procesos industriales. En la actualidad, los sistemas Scada son los encargados de supervisar, controlar, monitorear, y adquirir el estado de las variables de un proceso. Para la implementación y creación de los sistemas Scada, se han desarrollado aplicaciones o software que presentan las herramientas necesarias al usuario final, para desarrollar su propio sistema Scada. Sin embargo, la mayoría de aplicaciones de desarrollo de sistemas Scada son de tipo propietario, por lo que su licencia de ejecución y módulos para tareas específicas tienen un costo. Además, son aplicaciones cerradas, es decir no se puede modificar su código fuente, en caso de que sea necesario realizar una adaptación al sistema. Como alternativa a este tipo de software propietario y de alto costo, se desarrolló **CM,** una aplicación basada en software de código abierto y distribución libre para la creación de sistemas HMI/Scada. **CM** tiene como principio ser un software de distribución libre, y cuyo código puede ser modificado por cualquier persona, de acuerdo a sus necesidades.

1.1 Motivación

Este trabajo fue motivado por la necesidad de encontrar una alternativa a las diferentes aplicaciones propietarias de desarrollo de sistemas HMI/Scada como: InTouch, iFix, XA/21, etc.

Durante los dos últimos años he tenido la oportunidad de observar el gran campo de aplicación que tienen los sistemas Scada dentro de la industria. En el verano de 2008, durante mi pasantía en el Terminal de Despacho y Almacenamiento de Combustibles de Petrocomercial, tuve la oportunidad de observar por primera vez el funcionamiento de un sistema Scada en un proceso real. El sistema se encargaba de la supervisión y monitoreo del despacho de combustibles hacia los tanqueros. Estaba desarrollado mediante el Software Intouch de Wonderware, al que tuve la oportunidad de manipular y ver sus capacidades. Al terminar la pasantía, tuve la intención de seguir investigando las

cualidades de este software y las herramientas que brindaba a los usuarios, pero presentaba un problema, el alto costo de su licencia. Entonces, para la aplicación en un sistema propio y pequeño, no sería rentable adquirir la licencia de un programa como Intouch. En ese instante, nació la idea de buscar alternativas libres y sin costo para el desarrollo de sistemas Scada. Por otro lado, llevo casi dos años como usuario de Linux, y siempre tuve el problema, que la mayoría de software se desarrolla sobre plataforma Windows, como por ejemplo Matlab. Entonces, tenía dos posibilidades, cambiar mi sistema operativo a Windows, o buscar alternativas dentro de Linux. Atraído por el principio de libertad de Linux, cada vez que era necesario buscaba alternativas con respecto al software ejecutado sobre Windows. Durante este tipo de búsquedas, encontré muchos proyectos desarrollados a partir de software libre y cuyo código podía ser modificado por cualquier persona. Entonces, combinando mi necesidad por un software de desarrollo de sistemas HMI/Scada, y la idea de generar alternativas libres y accesibles para cualquier persona, decidí comenzar con mi propio proyecto desde cero. **CM** (lo llamé así por las primeras letras de mi nombre y apellido: **C**hristian **M**oya), es un proyecto personal diseñado, desarrollado e implementado con el objetivo de generar aplicaciones HMI/Scada.

1.2 Objetivos

1.2.1 Objetivo General

El objetivo principal de este trabajo es generar una alternativa libre y de código abierto, a las aplicaciones de desarrollo de sistemas HMI/Scada, cuya licencia de ejecución tiene un costo elevado, y a las cuales no se puede modificar para adaptarlas a un sistema en particular.

1.2.2 Objetivos Específicos

- Diseñar, desarrollar, e implementar una plataforma a través de un ordenador, que permita implementar tareas de control, supervisión y automatización con orientación a procesos industriales.
- Desarrollar el software para que sea aplicado en sistemas de control reales, especialmente en el desarrollo de procesos industriales.

- Generar una aplicación de desarrollo de sistemas HMI/Scada, que garantice el control del proceso en tiempo real.
- Desarrollar la aplicación con la capacidad de adquirir datos provenientes de los dispositivos de campo.
- Desarrollar un módulo que permite ejecutar algoritmos de control en tiempo real.
- Establecer una comunicación bidireccional, tanto de entrada como de salida, es decir permite recibir señales y entregar acciones de control.
- Desarrollar un soporte de comunicación industrial Modbus.
- Procesar señales de entrada.
- Generar aplicaciones orientadas al usuario, para desarrollo de Interfaces Hombre-Máquina (HMI's), y supervisión de sistemas de control en tiempo real.
- Desarrollar un sistema interno de almacenamiento y gestión de datos.

1.3 Contenido del Documento

El contenido de la tesis es el siguiente:

Capítulo 2: Sistemas Scada

Este capítulo desarrolla el fundamento teórico de los sistemas Scada. Parte desde sus funciones básicas hasta métodos de implementación. Describe los componentes de un sistema Scada con particular interés en los utilizados para la construcción de la aplicación CM, para el desarrollo de sistemas HMI/Scada.

Capítulo 3: Software Libre y Software de Código Abierto

Se desarrolla una introducción breve a los distintos tipos de software utilizados en la creación de la aplicación CM. Software de distribución libre y código abierto como Python, MySQL, wxpython entre otros paquetes y módulos, son descritos con el objetivo de mostrar sus características y el porqué fueron utilizados para el desarrollo de la aplicación.

Capítulo 4: Sistemas de control en tiempo real y Sistemas de control Multivariable

Los procesos industriales son presentados como un problema de control multivariable en tiempo real. Se desarrolla una introducción a las características principales que debe cumplir un sistema de control de varias variables y en tiempo real. Como resultado, se determina las propiedades que debe tener una aplicación de desarrollo de sistemas HMI/Scada para garantizar un control fiable y eficiente sobre el proceso.

Capítulo 5: CM-HMI/Scada

Este capítulo desarrolla de manera detallada el perfil de la aplicación CM. Se describe todas las características funcionales y estructurales del software de desarrollo de sistemas HMI/Scada **CM**. Se presenta su alcance y la evolución temporal del sistema, los módulos desarrollados y una introducción básica al uso del sistema en el desarrollo de aplicaciones HMI/Scada mediante CM.

Capítulo 6: Conclusiones y Trabajo Futuro

En el último capítulo son presentadas las conclusiones. Se discute los alcances del sistema y el desarrollo posterior de CM; es decir, las propuestas para la evolución futura del sistema.

Apéndices

El trabajo concluye con dos apéndices:

- A: Manual de CM
- B: Código de los módulos de CM

2

Sistemas Scada

2.1 Descripción General del los Sistemas Scada

2.1.1 Definición

Scada, acrónimo de “Supervisory Control and Data Acquisition”, que en español significa Control de Supervisión y Adquisición de Datos. Los sistemas Scada se refieren a la combinación de la telemetría (medición remota de magnitudes físicas), con la adquisición de datos. Sistemas Scada abarcan la colección de información a través de unidades terminales remotas UTR’s, transfiriéndola a un servidor central, realizando el análisis y control necesario, para luego desplegar la información en un número de pantallas de operación o despliegues. Las acciones de control son transmitidas de vuelta hacia el proceso [1].

2.1.2 Antecedentes y Evolución de los Sistemas Scada

Los primeros sistemas Scada solo se configuraban para transmitir el estado de determinadas variables y condiciones de la planta, sin realizar ninguna actividad de control sobre el proceso. Además, se presentaban los datos en paneles de control industrial, usando señales visuales. La capacidad de control y supervisión de estos sistemas eran limitadas. Sin embargo, con el desarrollo tecnológico de los años posteriores, fueron los ordenadores y hardware específico los encargados del almacenamiento y adquisición de los datos de la planta. Estos ordenadores incorporaban comandos de control, además adquirieron la función de presentar los datos de la planta en tiempo real y de manera visual en el ordenador. Finalmente, mientras la velocidad de procesamiento de información de los ordenadores aumentaba, de la misma manera aumentó la capacidad de programar el sistema para que realice funciones de control.

2.1.3 Aplicaciones de los Sistemas Scada

Los sistemas Scada pueden ser relativamente simples, como en el caso del control de las condiciones ambientales en una oficina, o pueden ser relativamente complejos, como en una planta de Generación Hidroeléctrica o Nuclear. Los sistemas Scada son usados en la mayoría de entornos industriales complejos, o en procesos industriales que cubren un área geográfica amplia (Supervisión y Control de Líneas de Transmisión de Energía Eléctrica, Oleoducto, Sistema de Agua Potable), debido a que se puede adquirir información muy rápidamente y desde lugares remotos, para luego ser presentada en la pantalla de un ordenador.

Los sistemas Scada son aplicados en industrias como:

- Comunicaciones.
- Control de Aguas Residuales y Desperdicios.
- Generación de Energía.
- Refinerías de Gas y Aceite.
- Industria Petroquímica, entre otras aplicaciones.

2.2 Características Generales de los Sistemas Scada

Entre las principales características de los sistemas Scada tenemos:

- Emplean computadores y protocolos de comunicación industrial para automatizar el monitoreo y control de diversos procesos industriales.
- Permiten obtener la representación de los datos de una planta en tiempo real. Además, los sistemas Scada actuales tienen la capacidad de ejecutar algoritmos de control que modifican la respuesta de la planta.
- Fueron desarrollados para sustituir sistemas de control obsoletos en industrias.
- Permiten optimizar la energía utilizada en el proceso.
- Para evitar accidentes ocasionados por los sistemas de control obsoletos.
- Constituyen una ventana del proceso. Permiten conocer el estado actual del mismo.
- Permiten maximizar la producción.
- Buscan reducir los costos de personal.

- A partir de los sistemas Scada se puede realizar análisis basados en datos actuales y pasados, gracias a la gran capacidad de almacenamiento de datos que poseen.

2.3 Características Operativas de los Sistemas Scada

Los sistemas Scada en la actualidad se están convirtiendo en parte integral de la estructura de gerenciamiento de información corporativa de la industria, ya que a través de los sistemas Scada se adquiere la información necesaria para la toma de decisiones, por lo que constituyen más que una herramienta operacional. Por ejemplo, en una estación de almacenamiento y distribución de combustible, la gerencia planifica el despacho de un determinado combustible en base al nivel que se tiene en el tanque de almacenamiento. Otro caso, es el de una planta de Generación Eléctrica en la cual se planifica un despacho para cada hora en base al estudio de carga. El estudio de carga es un análisis estadístico de la potencia activa y reactiva requerido por el sistema, y cuyos datos son proporcionados mediante el sistema Scada. Por lo tanto, los sistemas Scada constituyen el núcleo de la parte operacional de la planta, pero a su vez presentan datos a los sistemas o usuarios fuera del ambiente del operador. Estos datos se usan para tomar decisiones, por lo que es necesario que los mismos sean confiables y seguros.

Un sistema Scada adquiere información (por ejemplo, si ocurrió una variación en la frecuencia de un sistema de generación de Energía Eléctrica), esta información es transferida hacia el Servidor Scada, el cual genera un aviso o alerta en un terminal de operador, informando la variación de frecuencia ocurrida. Además, la información viene analizada de acuerdo a parámetros preestablecidos, y comunica al operador la acción de control realizada. Por ejemplo, si la variación de frecuencia es mayor a uno de los límites el operador procede a realizar las acciones necesarias para recuperar el sistema (por ejemplo: dependiendo del signo de la variación, compensar la potencia necesaria, variando la velocidad de los generadores o en casos más extremos desconectar carga o generación). Esta información generalmente se despliega en la terminal del operador en las Interfaces Hombre-Máquina (HMI), de manera ordenada y clara.

2.4 Arquitectura de los Sistemas Scada

Los procesos industriales actuales requieren de una operación correcta y eficiente del sistema. Para lograr esta operación, la arquitectura implementada en el sistema es muy importante. Los desarrolladores actuales de sistemas Scada eligen arquitecturas escalables, con el objetivo de aumentar las tareas de los sistemas en el futuro [2]. Los sistemas Scada han sido diseñados siguiendo dos tipos de arquitecturas, las cuales son:

- Arquitectura Centralizada
- Arquitectura Distribuida

2.4.1 Sistemas Scada Centralizados

Los sistemas Scada centralizados son los que desarrollan todo el sistema alrededor de un solo computador maestro. Este computador se encarga de adquirir, almacenar y controlar los datos del sistema [2]. A continuación se nombran algunas desventajas de los sistemas centralizados:

- El sistema no puede ser completamente escalable. Al depender de un solo servidor maestro, la escalabilidad es función de la capacidad del ordenador o servidor central.
- Para lograr la redundancia del sistema, se requerirá duplicar completamente al mismo. Ya que todo el sistema va a través de un servidor central maestro [2].
- El servidor central debe ser un computador de alta capacidad y con costos de mantenimiento elevados [2].

2.4.2 Sistemas Scada Distribuidos

Los sistemas Scada son considerados distribuidos, cuando las actividades de adquirir, almacenar y controlar el sistema son compartidas entre un grupo de ordenadores. Aunque, la arquitectura distribuida es ideal para generar sistemas escalables y redundantes, se pueden presentar los siguientes problemas:

- La comunicación entre computadores no es fácil, y puede ocasionar errores de configuración [2].

- El sistema de arquitectura distribuida no optimiza el flujo de la información. Por ejemplo, si dos operadores diferentes requieren el valor de una variable en el mismo instante de tiempo, la UTR que adquiere el valor de la variable será consultada en dos ocasiones, lo cual no es óptimo [2].

2.4.3 Sistemas Redundantes

Existen procesos industriales críticos, en los cuales una falla puede generar grandes pérdidas, debido a la falta de producción. En este tipo de sistemas se requiere desarrollar estrategias y arquitecturas redundantes, que garanticen el desarrollo normal del proceso. En la siguiente figura, se muestra un sistema no redundante, y en el cual una falla en el Servidor Scada puede ocasionar el colapso del sistema [2].

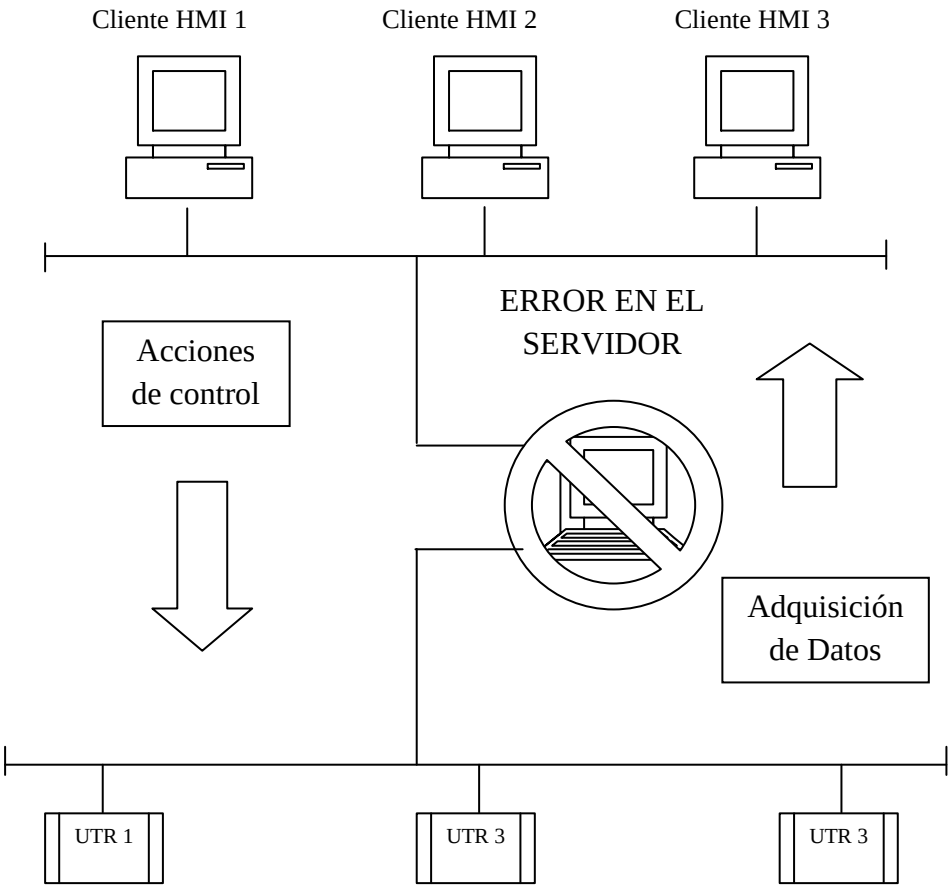

Figura 2.1: Sistema Scada sin redundancia

Para evitar colapsos generales en los sistemas, se han desarrollado arquitecturas redundantes de los sistemas Scada que garanticen la operación normal del proceso. Para el

desarrollo de las arquitecturas redundantes, se ha generado las aplicaciones de manera que las tareas definidas puedan ser ejecutadas en diferentes nodos o servidores. El objetivo de la arquitectura es el siguiente: Si falla el servidor principal, el servidor de respaldo, que opera en paralelo, se convierte en el servidor principal, y transfiere la información necesaria para el funcionamiento normal del sistema. La figura 2.2 muestra un esquema de la operación de este tipo de sistemas redundantes.

En sistemas estratégicos, como es el caso de una Central de Generación Hidroeléctrica, en donde existe una cantidad importante de variables a controlar, se desarrolla sistemas redundantes no solo a nivel de servidores, sino también a nivel de dispositivos de campo. La figura 2.3 muestra un esquema de este tipo de sistemas.

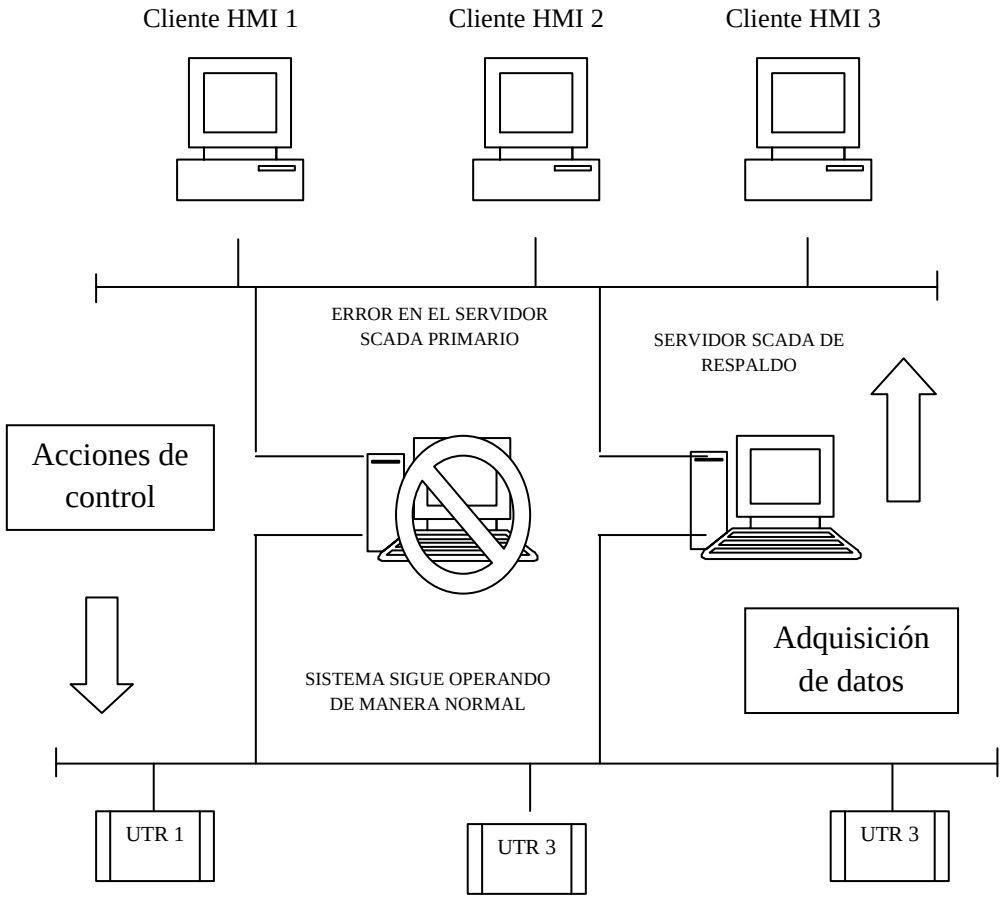

Figura 2.2: Sistema Scada con Redundancia (Servidor de Respaldo)

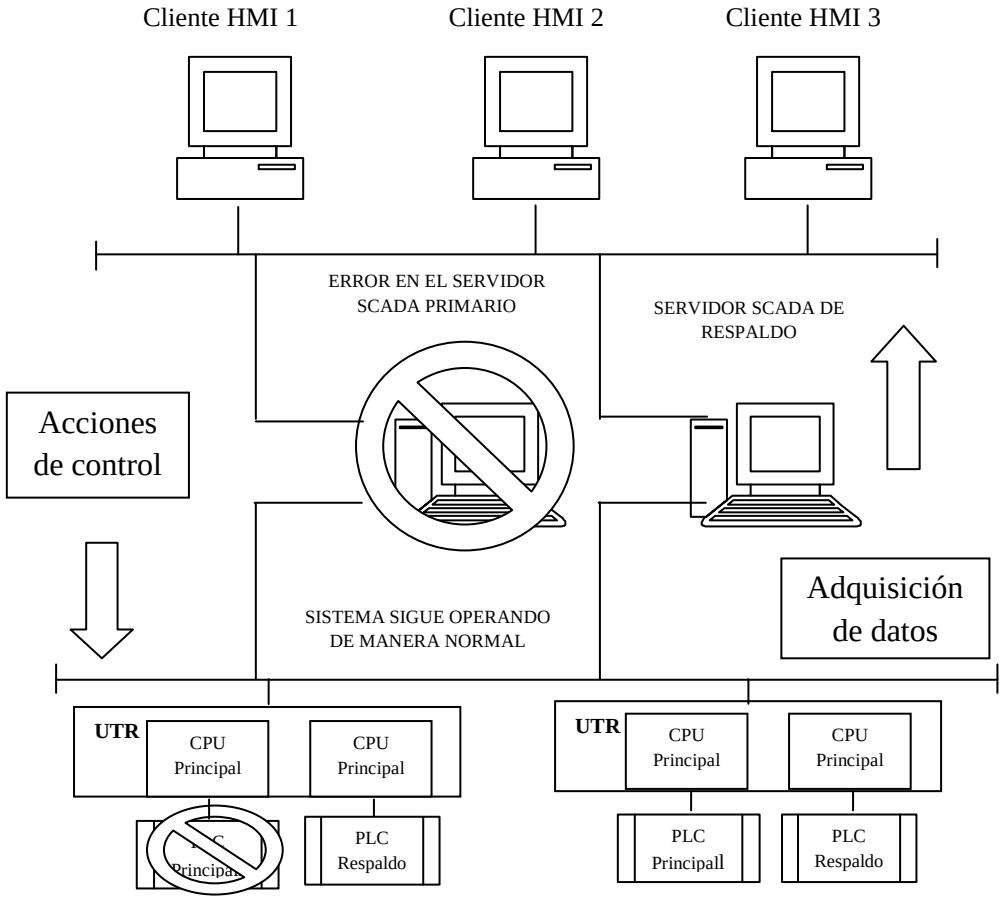

Figura 2.3: Sistema Scada con Redundancia de Servidor y Dispositivos de campo

2.4.4 Sistemas Expandibles o Escalables

Los procesos industriales actuales requieren que los sistemas sean diseñados de manera que puedan expandirse en el futuro. Los requerimientos que debe cumplir el sistema para que sea posible realizar una expansión son los siguientes [2]:

- La aplicación no deberá requerir de modificaciones al añadir nuevas UTR's o terminales de operador [2].
- El sistema operativo de la aplicación deberá ser capaz de soportar los requerimientos adicionales producidos por la expansión del sistema.
- Las instalaciones físicas del proceso deben ser capaces de soportar la adición de nuevo hardware [2].
- El hardware adicional a ser instalado no deberá ejercer ningún impacto sobre el hardware existente instalado [2].

- La arquitectura del sistema Scada no deberá cambiar con la adición de nuevo hardware.

2.5 Estructura Funcional de los Sistemas Scada

Los sistemas Scada son estructurados de manera que se garanticen a todos sus niveles escalabilidad, seguridad, fiabilidad y rendimiento óptimo del sistema. Como resultado, los sistemas Scada han sido divididos en los siguientes bloques funcionales:

- Hardware de Adquisición de datos (UTR's, PLC's).
- Red de comunicación industrial (Modbus RTU).
- Sistema de gestión de datos.
- Software HMI/Scada.

2.6 Hardware de Adquisición de datos

Un sistema Scada consiste en un número de unidades terminales remotas o controladores lógicos programables adquiriendo los datos provenientes de los elementos de campo, y enviándolos al servidor central a través del sistema de comunicación [3].

2.6.1 Elementos de Campo

Los elementos de campo permiten conocer en cada instante de tiempo, el estado del sistema o proceso. Por ejemplo, para establecer la condición de operación de un generador síncrono, se utilizan diversos elementos de campo, entre los que tenemos: el tacómetro para determinar la velocidad de rotación, sensores de temperatura (rotor, estator, intercambiadores de calor), medidores de presión en la turbina, sensores de presión de aceite para el control de velocidad de la máquina, medidores de voltaje, corriente, potencia activa generada y potencia reactiva generada, entre otros elementos. Todos estos proveen la información necesaria al operador para determinar el estado actual y las condiciones operativas del generador síncrono. En conclusión, los elementos de campo son todos los sensores y actuadores que adquieren los datos y actúan sobre el sistema.

2.6.2 Unidades Terminales Remotas (UTR's)

Para que los elementos de campo puedan enviar los datos hacia el Servidor Scada, es necesario, que se envíen a través de un sistema de comunicación compatible. Para que se logre enviar la información en el formato correcto, se usa las UTR's (unidades terminales remotas), que proveen la interfaz necesaria. Las UTR's son usadas fundamentalmente para convertir las señales electrónicas recibidas por un elemento de campo (por ejemplo, una señal de 4 a 20 mA), a un lenguaje o protocolo de comunicación (por ejemplo: Modbus); y así lograr la transmisión adecuada de los datos obtenidos por los elementos de campo [4].

Las Unidades Terminales Remotas de un sistemas Scada, son ordenadores ubicados en el campo, los cual se encargan de procesar la información y la interacción entre el Servidor Scada y los distintos elementos de campo (Sensores, Actuadores). Las UTR's son unidades independientes, encargadas de la adquisición de datos y en la actualidad, también poseen la capacidad de realizar algoritmos de control.

Características de la UTR

- Tienen una unidad de procesamiento y en ocasiones unidad de procesamiento de respaldo.
- Poseen una memoria de programa, datos y de acceso aleatorio.
- Puertos de comunicación (Serial, Ethernet, etc).
- Fuente de alimentación y en ocasiones fuente de alimentación de respaldo.
- Diversas protecciones necesarias para garantizar el funcionamiento correcto y confiable de la UTR.
- Módulos e interfaces de entrada y salida.
- Driver para la comunicación con el Servidor Scada (Por ejemplo: Driver Modbus).
- Sistema de ejecución en tiempo real.
- Driver para el manejo de los módulos de entrada y salida.

Operación de la UTR

La Unidad Terminal Remota se encarga de escanear sus entradas, cada cierto tiempo t, definido como el tiempo de acceso o adquisición. Puede realizar ciertas operaciones, que incluyen el procesamiento de los datos adquiridos, o el cambio de estado

de una determinada variable. Generalmente utilizan la forma de comunicación pedido/respuesta, es decir el Servidor Scada es el que inicia la comunicación para la obtención de un determinado valor. Ciertas UTR's, en la actualidad contienen dentro de su programación algoritmos de control, y procesamiento de alarmas, tal como un PLC, por lo que en ocasiones la nomenclatura es intercambiable [4].

2.6.3 Controladores Lógicos Programables (PLC's)

Los PLC's tienen sus orígenes en la automatización industrial, y en el pasado eran usados en sistemas donde la necesidad de conectar los PLC's con el canal de comunicación no era grande. Usualmente, servían para reemplazar sistemas de lógica de relés o de control neumático. Por otro lado, los sistemas Scada tienen sus orígenes en la necesidad de determinar el estado actual de un proceso, a través de enlaces de comunicación. Sin embargo, en los inicios de los sistemas Scada solo era necesario conocer información básica de un elemento remoto. Por esta razón, las UTR's, conectadas a los sistemas Scada no tenían la necesidad de realizar acciones de control programadas, debido a que el algoritmo local de control era realizado por la lógica de relés. Con el paso de los años, los PLC's han sido usados para reemplazar a los sistemas de control con lógica de relés. Como resultado, protocolos de comunicación industrial han sido implementados de manera más frecuente con los PLC's, lo que permite que el PLC pueda comunicar su estado a un Servidor Scada y forme parte de un sistema de control distribuido mediante una red de comunicación industrial.

2.6.4 PLC's vs UTR's

Un PLC (Programmable Logic Controller), es un ordenador que se utiliza en las industrias con el objetivo de automatizar un determinado proceso. Los PLC's fueron originalmente usados para reemplazar la lógica de relés, pero con el tiempo incorporaron módulos de comunicación que les permitieron ser parte integral de los sistemas Scada. Los PLC's tienen tal como las UTR's, módulos de entrada y salida, pero dentro de su programación contienen algoritmos de control y automatización. En la actualidad, existen UTR's con la capacidad de ejecutar algoritmos de control, por lo que PLC's y UTR's cumplirían la misma función. En conclusión, PLC's y UTR's compiten en la actualidad por el mismo mercado dentro de los sistemas Scada. Sin embargo, en ocasiones se pueden usar alternativas de combinación de ambos elementos. Esta combinación se la realiza cuando

las condiciones lo requieren debido a limitaciones en la comunicación, el procesamiento, o simplemente cuando se requiere aumentar el tamaño de la red Scada.

2.7 Red de Comunicación Industrial

La red de comunicación industrial otorga los medios mediante los cuales, los datos son transmitidos entre el Servidor Scada, y las UTR's ubicadas en el campo. La red de comunicación industrial, se refiere al equipo necesario para transferir datos desde y hacia diferentes lugares. Los medios usados pueden ser cable, radio entre otros. El cable como medio de propagación de datos es el más usado a nivel industrial; sin embargo, cuando el sistema cubre grandes áreas geográficas no es recomendable. Durante los últimos años los sistemas Scada han aumentado el uso de las redes LAN y WAN, para el desarrollo de redes de computadores [3].

2.7.1 Modos de Transmisión de datos y Protocolos de Comunicación Industrial

Todo sistema de comunicación se compone de un transmisor, un receptor y un medio de comunicación, que permite el envío de la información. Para lograr que el receptor sea capaz de entender la información enviada por el transmisor, ambos deben reconocer los siguientes factores [2]:

- El tipo de señales eléctricas utilizadas.
- Los códigos que identifican los símbolos.
- El significado de los símbolos.
- La forma en la que el flujo de los datos son controlados.
- La forma de detectar errores.

Los factores de tipo físico descritos anteriormente son descritos dentro de la interfaz de comunicación, en cambio los factores de identificación y procesamiento de la información son desarrollados por los protocolos de comunicación industrial [2].

En cualquier medio de comunicación que conecta a un transmisor con un receptor, existen tres modos de transmisión de datos [2]. Estos son:

- **Simplex:** La información solo puede ser enviada en una sola dirección.
- **Half-Duplex:** La información puede fluir en dos direcciones, pero solo una dirección en cada instante de tiempo.
- **Full-Duplex:** La información puede fluir en ambas direcciones en cualquier instante de tiempo.

Otras características que determinan la forma de enviar los datos son el tipo de sistema de comunicación (Asíncrono o Sícrono) y si la línea de transmisión es o no balanceada. Todas estas características tienen que ser definidas por los diferentes estándares de interfaces de comunicación [2]. Entre los principales estándares de interfaces de comunicación usados a nivel industrial tenemos:

- Estándar de Interfaz EIA-232 (Serial).
- Estándar de Interfaz EIA-485.

En la actualidad, existe un marco de trabajo en los sistemas de comunicación conocido como modelo de interconexión de sistemas abiertos (OSI: Open System Interconnection). El objetivo de este modelo es permitir a los estándares existentes y en desarrollo, sean colocados dentro del marco de trabajo común, para asegurar la conectividad entre diferentes sistemas.

Por otro lado, el significado de los mensajes que contiene la información dentro de un sistema Scada está definido por los protocolos de comunicación industrial, que a su vez definen la forma de detectar errores en la información que se transmite. Entre los principales protocolos de comunicación industrial tenemos:

- Modbus (Implementado en CM).
- Fieldbus.
- Hart.
- DNP3.0.
- DeviceNet, entre otros.

2.7.2 Estándar de Interfaz de Comunicación Serial

El estándar de comunicación serial RS-232C define la interfaz entre un equipo terminal de datos (DTE Data Terminal Equipment) y un equipo de comunicación de datos

(DCE Data communication Equipment) empleando un intercambio de datos binarios de manera serial. La versión actual es la **EIA-232E** [3**].**

El estándar RS-232 está compuesto de tres componentes, los cuales definen:

- Las características eléctricas de las señales: Definen los niveles de voltaje y la referencia entre las señales que se intercambian y sus circuitos asociados.
- Las características mecánicas de la interface entre el DTE y el DCE.
- La descripción funcional de los circuitos de intercambio de información, por ejemplo, las señales de control de flujo, la función de los datos, y el tiempo de transmisión y recepción usado en la interfaz entre el DTE y el DCE.

Características Eléctricas de las Señales

El estándar de comunicación serial RS-232 esta diseñado como un sistema de comunicación entre dos dispositivos conocidos como:

- DTE: Equipo terminal de datos
- DCE: Equipo de comunicación de datos

Estos dispositivos entienden datos binarios (1 o 0) definidos mediante rangos de voltaje (Tabla 2.1) [3].

	Rango de Voltajes
0 Lógico	De +3V a +25V
1 Lógico	De -3V a -25V
No definido	De +3V a -3V

Tabla 2.1: Niveles de Voltaje del receptor en el estándar de comunicación RS-232

Para que la transmisión sea efectiva, se debe generar una señal de voltaje en el rango de +5V a+25V y entre -5V a -25V, esto debido a que el nivel de voltaje puede entrar en la zona no definida debido a la atenuación de la señal provocada por pérdidas. Por ejemplo, el circuito Integrado Max 232 opera con valores de +7 V y -7 V.

Características Mecánicas de la Interfaz

Aunque no se especifica en el estándar de Comunicación Serial RS-232, los conectores DB-25 y DB- 9 han sido asociados de manera cercana con el estándar RS-232 [3].

En la tabla 2.2 se presenta la distribución de los pines más importantes en los dos tipos de conectores.

DB-9		**DB-25**	
PIN	**Descripción**	**PIN**	**Descripción**
1	**DCD:** Detector de Transmisión	**1**	Shield: Protección
2	**RXD:** Recibir datos	**2**	**TXD:** Transmitir Datos
3	**TXD:** Transmitir Datos	**3**	**RXD:** Recibir datos
4	**DTR:** Terminal de datos listo	**4**	**RTS:** Permiso para transmitir
5	**GND:** Señal de Tierra (Común)	**5**	**CTS:** Libre para enviar
6	**DSR:** Conjunto de datos listo	**6**	**DSR:** Conjunto de datos listo
7	**RTS:** Permiso para transmitir	**7**	**GND:** Señal de Tierra (Común)
8	**CTS:** Libre para enviar	**8**	**DCD:** Detector de Transmisión
9	**RI:** Indicador de llamada	**20**	**DTR:** Terminal de datos listo
		22	**RI:** Indicador de llamada

Tabla 2.2: Distribución de pines conectores DB9 y DB25

Desventajas del estándar de Comunicación serial RS-232

El estándar de comunicación serial RS-232 presenta ciertas desventajas entre las cuales tenemos [3]:

- La interfaz de comunicación RS-232 solo permite comunicación punto a punto, lo cual es un problema cuando se tienen varios dispositivos y se quiere generar una red con los mismos.
- La interfaz de comunicación RS-232 tiene una limitación de distancia (generalmente 50 metros), lo cual es un problema cuando los dispositivos están separados por distancias mayores.
- La velocidad de transmisión para la interfaz de comunicación RS-232 (por ejemplo: 19200 bps), es demasiado lenta para diversas aplicaciones.
- El estándar presenta alta susceptibilidad al ruido, ya que se trata de un estándar no balanceado.

Todas estas desventajas han obligado a los diseñadores de sistemas de comunicación a buscar alternativas, como por ejemplo: los estándares RS-422, RS-485, entre otros, los cuales superan las limitaciones antes descritas para la interfaz de comunicación RS-232 [3].

2.7.3 Estándar de Interfaz RS-485

La versión actual del estándar de Interfaz RS-485 es: EIA-485.

Características Principales de la Interfaz de Comunicación RS-485

- Generar una red de hasta 32 receptores por un mismo canal o línea de comunicación.
- RS-485 permite aumentar la distancia de comunicación serial hasta 1200 metros.
- RS-485 permite aumentar la velocidad de transmisión hasta 10Mbps.

En la tabla 2.3 se define la equivalencia entre rangos de voltaje y el dato binario equivalente [3].

Una de las ventajas del estándar de comunicación RS-485 es que posee tres estados de operación: 0 lógico, 1 lógico y alta impedancia.

El estado de alta impedancia es conocido como el estado deshabilitado, ya que el dispositivo que entra en este estado virtualmente se desconecta de la red. Este estado de operación es el que permite generar una red de hasta 32 dispositivos; sin embargo, en cada

instante de tiempo solo 1 dispositivo puede estar activo. Para que el sistema funcione adecuadamente a cada dispositivo se le asigna una dirección [2].

	Rango de Voltajes
0 Lógico	De +1.5V a +6V
1 Lógico	De –1.5V a -6V
Alta Impedancia	De -1.5V a +1.5V

Tabla 2.3: Niveles de Voltaje del receptor en el estándar de comunicación RS-485

2.7. 4 Protocolo de Comunicación Modbus

Modbus es un protocolo de intercambio de mensajes ubicado en la capa aplicación, que se ubica en el nivel o capa 7 del modelo de la OSI (Open System Interconection). Provee una comunicación maestro/esclavo, y se utiliza entre diferentes dispositivos interconectados dentro de una red. Se lo conoce como el protocolo de comunicación industrial estándar. Permite que muchos dispositivos de automatización puedan comunicarse entre sí. Fue desarrollado y publicado por Modicom en año de 1979 [3].

El protocolo de comunicación modbus se implementa usando:

- TCP/IP sobre Ethernet
- Transmisión serial asincrónica sobre una variedad de medios como por ejemplo: cable, radio, fibra, entre otros.
- Modbus Plus, que es una red de alta velocidad y la cual a diferencia de Modbus, ya que no es de distribución libre.

Características del protocolo Modbus Maestro/Esclavo

- El protocolo Modbus Maestro/Esclavo establece que solo un maestro en el mismo instante de tiempo, esta conectado al bus de datos, y que uno o varios esclavos (247 como máximo), están conectados al bus [3].
- La comunicación Modbus siempre es iniciada por el maestro. El esclavo nunca puede transmitir datos sin previo pedido del maestro.

- Dentro de la comunicación Modbus los esclavos no pueden comunicarse entre ellos.
- El nodo maestro solo puede iniciar una transacción en un determinado instante de tiempo.
- Cada dispositivo dentro de una red Modbus tiene que tener una dirección diferente.

Podemos distinguir dos variantes para la representación numérica de los datos y detalles del protocolo, las cuales tienen ciertas diferencias entre si [2]:

- Modbus RTU.
- Modbus ASCII.

Modbus RTU

Es una representación binaria compacta de los datos. Dicho formato termina la trama de cada mensaje con una suma de control de redundancia cíclica (CRC).

Formato de cada byte en modo RTU

Sistema de Codificación: 8 bits binarios, hexadecimal, 0-9, A-F, 2 caracteres hexadecimal en cada campo de 8 bits de mensaje [5].

Bits por Byte:

- 1 bit de inicio.
- 8 bits de datos, el bit menos significativo se envía primero.
- 1 bit de paridad (par o impar), o ninguno si no hay paridad.
- 1 o 2 bits de parada.

Campo de Control de error: Control de Redundancia Cíclica (CRC)

Modbus ASCII

Es una representación legible del protocolo pero menos eficiente. Su implementación es en serie, y al finalizar la trama este formato utiliza una suma de control de redundancia longitudinal (LCR).

Formato de cada byte en Modo ASCII

Sistema de Codificación: Hexadecimal, un caracter hexadecimal en cada caracter ASCII del mensaje [5].

Bits por Byte:

- 1 bit de Inicio
- 7 bits de datos, el bit menos significativo se envía primero
- 1 bit de paridad (par o impar), o ninguno si no hay paridad
- 1 o 2 bits de parada

Campo de Control de error: Control de Redundancia Longitudinal (LRC).

Finalmente, la versión Modbus/TCP es muy semejante al formato RTU, pero estableciendo la transmisión mediante paquetes TCP/IP.

Modbus define dos marcos dentro del mensaje:

- PDU= Unidad de datos del protocolo (Protocol Data Unit).
- ADU= Unidad de datos de la aplicación (Aplication Data Unit).

La unidad de datos de aplicación ADU tiene un tamaño máximo de 256 bytes y está compuesto por: la dirección del esclavo (1 byte), la unidad de datos de protocolo (PDU) y el control de error respectivo (2 bytes, CRC para Modbus RTU) [5].

La unidad de datos de protocolo PDU tiene un tamaño máximo calculado de la siguiente manera: 256 bytes – dirección de esclavo (1 byte) – CRC (2 bytes) = (hasta 253 bytes).

El protocolo de comunicación industrial Modbus define 3 tipos de unidad de datos de protocolo (PDU).

- Pedido Modbus (Modbus Request PDU).
- Respuesta Modbus (Modbus Response PDU).
- Respuesta de excepción Modbus (Modbus exception response PDU).

Códigos de Función e implementación en Modbus RTU

Los códigos de función son los que determinan el servicio requerido por el maestro, a un determinado esclavo. El código de función dentro de la trama de un mensaje Modbus

está codificado en 1 byte, por lo que los códigos válidos se encuentran en el rango de 1 a 255 decimal [6].

Existen tres categorías de Códigos de Función de Modbus:

1. **Códigos de Función Públicos:** Son códigos de Función bien definidos, y validados por la comunidad MODBUS-IDA.org.
2. **Códigos de Función definidos por el Usuario:** Se encuentran entre los rangos, 65 a 72 y 100 a 110 decimal. El usuario puede generar un código de función que no está especificado en la aplicación de Modbus.
3. **Códigos de Función Reservados:** Estos códigos de función no son de acceso público y están reservados para compañías y sus productos.

Descripción de los códigos de Función Modbus Implementados en la Aplicación CM

01 (0x01) Lectura del Estado de Bobinas

Descripción

La función 01 es usada para la lectura de 1 hasta 2000 bobinas ubicadas de manera continua en un dispositivo remoto. Lee el estado ON=1/OFF=0 de las bobinas especificadas en la trama del mensaje. Las bobinas en un dispositivo remoto son referenciadas como 0x [6].

Implementación de consulta

La tabla 2.4 especifica los campos dentro de la consulta. Cabe resaltar que se omiten los campos de dirección de esclavo, y de control de error los cuales son generales, para todas las funciones.

Implementación de respuesta

El estado de cada bobina viene empaquetado en el campo de datos de respuesta. Cada bit determina el estado de una bobina (0 = desactivada y 1 = activada). El bit menos significativo del primer byte del campo de datos contiene el estado de la bobina especificada en la dirección inicial, las demás bobinas se ubican en orden ascendente en los siguientes bits del byte, y de la misma manera en los siguientes bytes. Si la cantidad de

bobinas devueltas no es múltiplo de 8, los bits restantes del último byte y que no especifican ninguna bobina serán llenados con ceros. La tabla 2.5 muestra los diferentes campos en una trama de mensaje de respuesta.

Campo	Definición	Tamaño	Valores
Código de Función	Función especifica para la lectura del estado de bobinas	1 Byte	0x01
Dirección Inicial	Dirección de la primera bobina especificada	2 Bytes	0x0000 a 0xFFFF
Cantidad de Bobinas	Número de bobinas a leer, a partir de la inicial especificada.	2 Bytes	1 a 2000 (0x7D0)

Tabla 2.4: Campos de un mensaje de consulta usando la función 0x01

Campo	Definición	Tamaño	Valores
Código de Función	Función especifica para la lectura del estado de bobinas	1 Byte	0x01
Conteo de Bytes	Número de bytes que contienen el estado de todas las bobinas	1 Bytes	*
Estado de Bobinas	Estado de las bobinas especificadas en la consulta	n Bytes	n= N o N+1 bytes de resultado

Tabla 2.5: Campos de un mensaje de respuesta usando la función 0x01

* El número de bytes de datos que contienen el estado de las bobinas se calcula de la siguiente forma: N = cantidad de bobinas a leer / 8, si el resto es diferente de cero, entonces n = N+1, caso contrario n = N.

Error

Existe otro tipo de respuesta por parte de un esclavo, la cual se origina cuando la consulta por parte del maestro no puede ser realizada, ya que existen inconsistencias o campos mal especificados en el mensaje de consulta. La siguiente tabla muestra los

campos especificados en una respuesta de excepción o error por parte del esclavo para el código de función 01 y en general para cualquier código de función.

Campo	Definición	Tamaño	Valores
Código de Función	Código de error (0x80) + Código de función de consulta (0x01)	1 Byte	0x01+0x80 = 0x81
Código de Excepción	Determina el error específico encontrado en la consulta.	1 Bytes	01 o 02 o 03 o 04

Tabla 2.6: Campos de un mensaje de respuesta de excepción usando la función 0x01

La figura 2.4 muestra un diagrama de flujo donde se desarrolla los diferentes estados que se presentan en una consulta por parte del maestro, cuando se usa el código de función 01. Además, se especifica los distintos códigos de excepción y su significado.

A continuación se describe un ejemplo que especifica un pedido de lectura de bobinas por parte del maestro. Desde la bobina 25 hasta la bobina 45. La siguiente tabla describe los valores de los campos del mensaje de consulta y respuesta del ejemplo descrito.

Consulta		***Respuesta***	
Nombre del Campo	**Hex**	**Nombre del Campo**	**Hex**
Función	01	**Función**	01
Dirección Inicial (Byte 1)	00	**Conteo de Bytes**	03
Dirección Inicial (Byte 2)	18	**Estado de Bobinas 32-25**	A2
Cantidad de Bobinas (Byte 1)	00	**Estado de Bobinas 40-33**	7F
Cantidad de Bobinas (Byte 2)	15	**Estado de Bobinas 45-41**	16

Tabla 2.7: Valores de los campos de los mensajes de consulta y respuesta del ejemplo usando la función 0x01 (Estado de las bobinas 25 a 45).

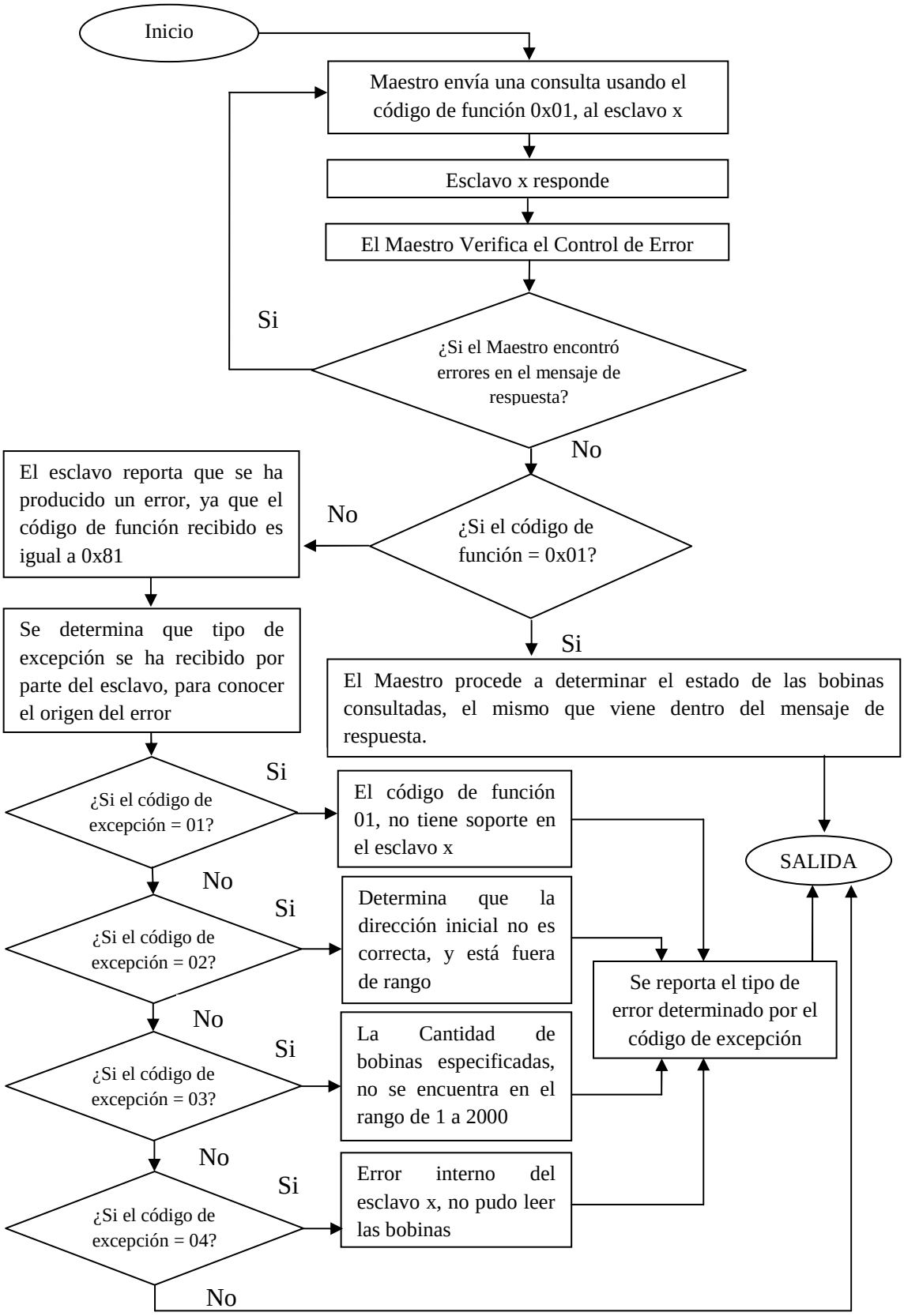

Figura 2.4: Diagrama de flujo de la lectura de estado de bobinas

Análisis

La **consulta** fue realizada para conocer el estado de 21 bobinas (0x15), comenzando desde la bobina 25. Se puede observar en la consulta que la dirección de la bobina inicial es 24 (18 en hexadecimal), esto se debe a que Modbus direcciona las bobinas, registros, y entradas discretas desde la dirección 0x, por lo que la bobina 1 del dispositivo remoto será direccionada con el valor 0, y así sucesivamente.

El mensaje de **respuesta** contiene el mismo valor como código de función, si no han existido errores de excepción. El siguiente campo nos determina el número de bytes en los cuales se incluirá el estado de las bobinas. En el primer byte de resultado tenemos el estado de las bobinas 25 hasta la bobina 32. El valor en binario de A2 es: 10100010, por lo que podemos observar que las bobinas 26, 30 y 32 están activadas. El mismo análisis se lo debe realizar para el conjunto de bobinas 33 hasta la bobina 40. El último byte de respuesta tiene un valor de 16 y corresponde al estado de las bobinas 41 hasta la bobina 45. Si observamos, el valor 16 hexadecimal en binario es: 00010110, por lo que se determina que tanto las bobinas 42, 43 y 45 están activadas. Finalmente los bits 5, 6 y 7 del último byte no están asignados al estado de una determinada bobina, por lo que necesariamente deben tener un valor de 0.

02 (0x02) Lectura de Entradas Discretas

Descripción

La función 02 es usada para la lectura de 1 hasta 2000 entradas discretas ubicadas de manera continua en un dispositivo remoto. Lee el estado ON=1/OFF=0 de las entradas discretas especificadas en la trama del mensaje. Las entradas discretas en un dispositivo remoto son referenciadas como 1x [6].

Implementación de consulta

La tabla 2.8 especifica los campos dentro de la consulta.

Campo	Definición	Tamaño	Valores
Código de Función	Función especifica para la lectura de entradas discretas	1 Byte	0x02
Dirección Inicial	Dirección de la primera entrada discreta especificada	2 Bytes	0x0000 a 0xFFFF
Cantidad de Entradas Discretas	Número de entradas discretas a leer	2 Bytes	1 a 2000 (0x7D0)

Tabla 2.8: Campos de un mensaje de consulta usando la función 0x02

Implementación de respuesta

La respuesta se estructura de manera idéntica a la respuesta a la función 0x01, cambiando solo el código de función a 0x02.

La tabla 2.9 muestra los diferentes campos en una trama de mensaje de respuesta.

Campo	Definición	Tamaño	Valores
Código de Función	Función especifica para la lectura de entradas discretas	1 Byte	0x01
Conteo de Bytes	Determina el número de bytes en los cuales se empaquetó el estado de las entradas discretas	1 Bytes	*
Estado de Bobinas	Estado de las entradas discretas especificadas en la consulta	n Bytes	n= N o N+1 bytes de resultado

Tabla 2.9: Campos de un mensaje de respuesta usando la función 0x02

* El número de bytes de datos que contienen el estado de las entradas discretas se calculan de la misma forma en la que se describió para el código de función 0x01.

Error

La siguiente tabla describe la respuesta de error específica para la función 0x02

Campo	Definición	Tamaño	Valores
Código de Función	Código de error para la función 0x02	1 Byte	0x82
Código de Excepción	Determina el error específico encontrado en la consulta.	1 Bytes	01 o 02 o 03 o 04

Tabla 2.10: Campos de un mensaje de respuesta de excepción usando la función 0x02

En la figura 2.5 se muestra el diagrama de flujo de la lectura de entradas discretas, desde el punto de vista del maestro.

Ejemplo

La siguiente tabla describe los valores de los campos de los mensajes de consulta y respuesta del ejemplo de uso de la función 0x02

Consulta		***Respuesta***	
Nombre del Campo	**Hex**	**Nombre del Campo**	**Hex**
Función	02	**Función**	02
Dirección Inicial (Byte 1)	00	**Conteo de Bytes**	01
Dirección Inicial (Byte 2)	05	**Estado de Entradas 12-06**	69
Cantidad de Bobinas (Byte 1)	00		
Cantidad de Bobinas (Byte 2)	07		

Tabla 2.11: Valores de los campos de los mensajes de consulta y respuesta del ejemplo usando la función 0x02 (Estado de entradas discretas 6 a 12)

Análisis

La **consulta** se realiza para determinar el estado de las entradas discretas 6 (Dirección Inicial 0x05 en Hexadecimal), 7, 8, 9, 10, 11 y 12.

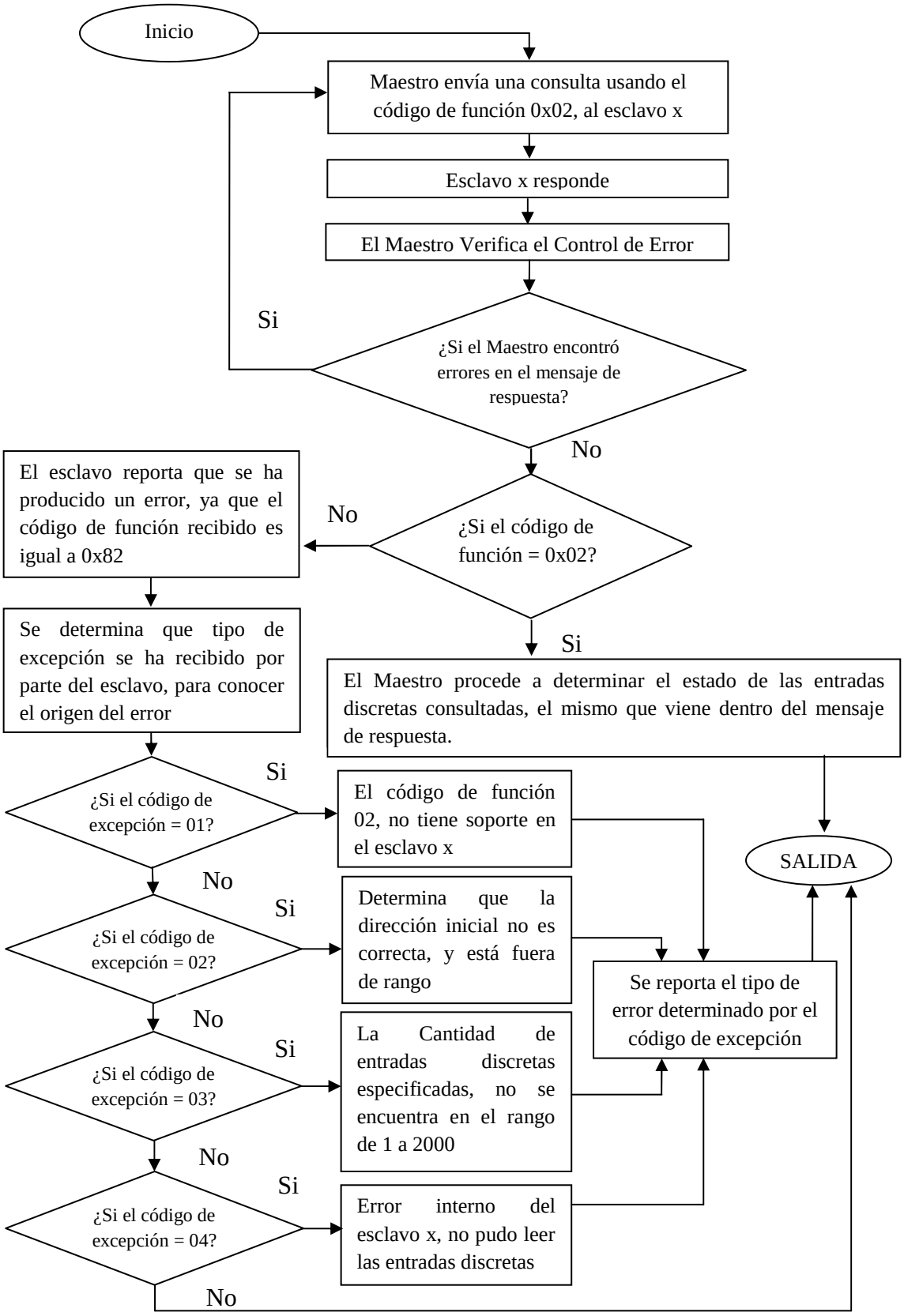

Figura 2.5: Diagrama de flujo de la lectura de estado de entradas discretas

La **respuesta** contiene el valor de la función 0x02, el número de bytes 1, debido a que solo se desea conocer el estado de 7 entradas discretas, que se puede enviar en un solo byte. El campo de datos (en binario) nos entrega el estado de las entradas discretas. El valor es de: 01101001, lo que nos indica que las entradas 6, 9, 11 y 12 están activadas. El bit 7 no se asigna a ninguna entrada discreta y su valor debe ser 0.

03 (0x03) Lectura de Registros Internos

Descripción

La función 03 es usada para la lectura de los contenidos de un bloque de registros internos de un dispositivo remoto. El número máximo de registros internos continuos que se pueden leer es de 125. Los registros internos de un dispositivo remoto son referenciados como 4x [6].

Implementación de consulta

La consulta especifica la dirección del registro inicial y el número de registros internos que se van a leer. La consulta se realiza desde la dirección 0, que ubica al registro interno 1 tal como en las funciones anteriores, para los bloques de bobinas y entradas discretas.

La tabla 2.12 especifica los campos dentro de la consulta:

Campo	Definición	Tamaño	Valores
Código de Función	Función especifica para la lectura de registros internos	1 Byte	0x03
Dirección Inicial	Dirección del primer registro interno especificado	2 Bytes	0x0000 a 0xFFFF
Cantidad de Registros Internos	Número de registros internos a leer, a partir del registro inicial especificado.	2 Bytes	1 a 125 (0x7D)

Tabla 2.12: Campos de un mensaje de consulta usando la función 0x03

Implementación de respuesta

La respuesta se estructura con el conteo de bytes de respuesta, cuyo valor siempre es el doble del número de registros (N), ya que el valor de un determinado registro interno se expresa en campos de 16 bits.

La siguiente tabla muestra los diferentes campos en una trama de mensaje de respuesta para el código de función 0x03.

Campo	Definición	Tamaño	Valores
Código de Función	Función especifica para la lectura de registros internos	1 Byte	0x03
Conteo de Bytes	Número de bytes que contienen los valores de todos los N registros consultados	1 Byte	2*N
Valores de los Registros Internos	Valor de los registros internos especificados en la consulta.	2* N Bytes	

Tabla 2.13: Campos de un mensaje de respuesta usando la función 0x03

Error

La tabla 2.14 describe la respuesta de error específica para la función 0x03

Campo	Definición	Tamaño	Valores
Código de Función	Código de error para la función 0x03	1 Byte	0x83
Código de Excepción	Determina el error específico encontrado en la consulta.	1 Bytes	01 o 02 o 03 o 04

Tabla 2.14: Campos de un mensaje de respuesta de excepción usando la función 0x03

La siguiente figura muestra el diagrama de flujo de la lectura de los valores de los registros internos, desde el punto de vista del maestro.

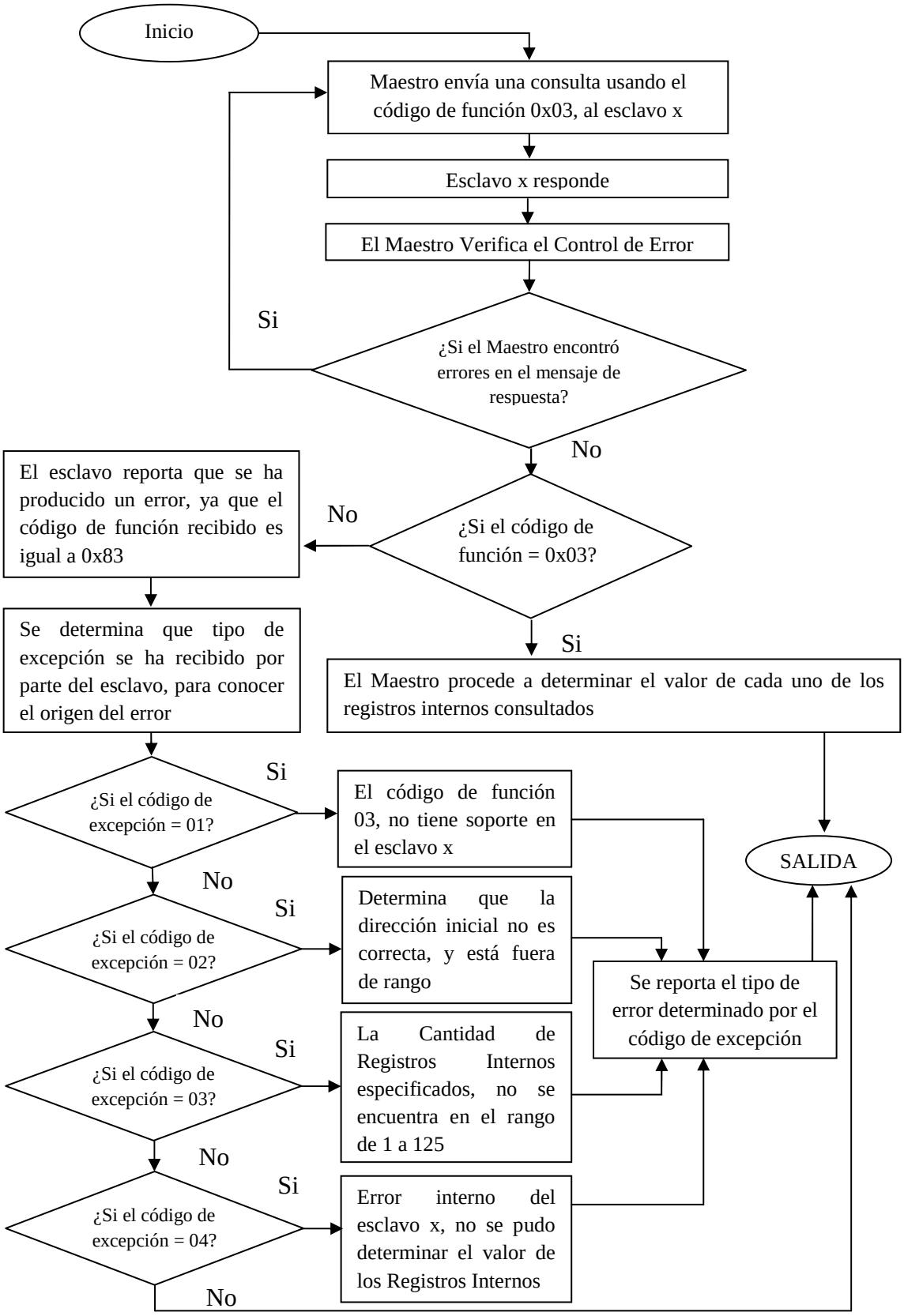

Figura 2.6: Diagrama de flujo de la lectura de registros internos

Ejemplo

La siguiente tabla describe los valores de los campos de los mensajes de consulta y respuesta del ejemplo de uso de la función 0x03.

Consulta		***Respuesta***	
Nombre del Campo	**Hex**	**Nombre del Campo**	**Hex**
Función	03	**Función**	03
Dirección Inicial (Byte 1)	00	**Conteo de Bytes**	04
Dirección Inicial (Byte 2)	1A	**Valor del Registro 27 (Byte 1)**	01
Cantidad de Bobinas (Byte 1)	00	**Valor del Registro 27 (Byte 2)**	3F
Cantidad de Bobinas (Byte 2)	02	**Valor del Registro 28 (Byte 1)**	00
		Valor del registro 28 (Byte 2)	29

Tabla 2.15: Valores de los campos de los mensajes de consulta y respuesta del ejemplo usando la función 0x03 (Estado de registros internos 27 y 28)

Análisis

La **consulta** se realiza para determinar el valor de los registros internos 27 (dirección inicial 26, 0x1A en hexadecimal) y 28.

La **respuesta** contiene el valor de la función 0x03, el número de bytes 4, debido a que el valor de los registros internos viene en campos de 16 bits, 2 bytes, por lo que el número de bytes de respuesta es el doble de la cantidad de registros especificada en la consulta. El campo de datos determina los valores de los registros internos 27 y 28, El valor del registro 27 es de 0x013F, que en decimal equivale a 319, en cambio que el valor del registro 28 es de 0x29, que en decimal equivale a 41.

04 (0x04) Lectura de Registros de Entrada

Descripción

La función 04 es usada para la lectura de los contenidos de un bloque de registros de entrada de un dispositivo remoto. El número máximo de registros de entrada continuos que se pueden leer es de 125. Los registros de entrada de un dispositivo remoto son referenciados como 3X [6].

Implementación de consulta

La consulta especifica la dirección del registro de entrada inicial y el número de registros de entrada que se van a leer. La tabla 2.16 especifica los campos dentro de la consulta:

Campo	Definición	Tamaño	Valores
Código de Función	Función especifica para la lectura de registros de entrada	1 Byte	0x04
Dirección Inicial	Dirección del primer registro de entrada especificado	2 Bytes	0x0000 a 0xFFFF
Cantidad de Registros de Entrada	Número de registros de entrada a leer, a partir del registro inicial especificado.	2 Bytes	1 a 125 (0x7D)

Tabla 2.16: Campos de un mensaje de consulta usando la función 0x04

Implementación de respuesta

La respuesta se estructura con el conteo de bytes de respuesta, cuyo valor siempre es el doble del número de registros (N), ya que el valor de un determinado registro de entrada se expresa en campos de 16 bits.

La siguiente tabla muestra los diferentes campos en una trama de mensaje de respuesta para el código de función 0x04.

Campo	Definición	Tamaño	Valores
Código de Función	Función especifica para la lectura de registros internos	1 Byte	0x04
Conteo de Bytes	Número de bytes que contienen los valores de todos los N registros consultados	1 Byte	2*N
Valores de los Registros de Entrada	Valor de los registros de entrada especificados en la consulta.	2* N Bytes	

Tabla 2.17: Campos de un mensaje de respuesta usando la función 0x04

Error

La tabla 2.18 describe la respuesta de error específica para la función 0x04

Campo	Definición	Tamaño	Valores
Código de Función	Código de error para la función 0x04	1 Byte	0x84
Código de Excepción	Determina el error específico encontrado en la consulta.	1 Bytes	01 o 02 o 03 o 04

Tabla 2.18: Campos de un mensaje de respuesta de excepción usando la función 0x04

La figura 2.7 describe un diagrama de flujo de la lectura de los valores de los registros de entrada, desde el punto de vista del maestro.

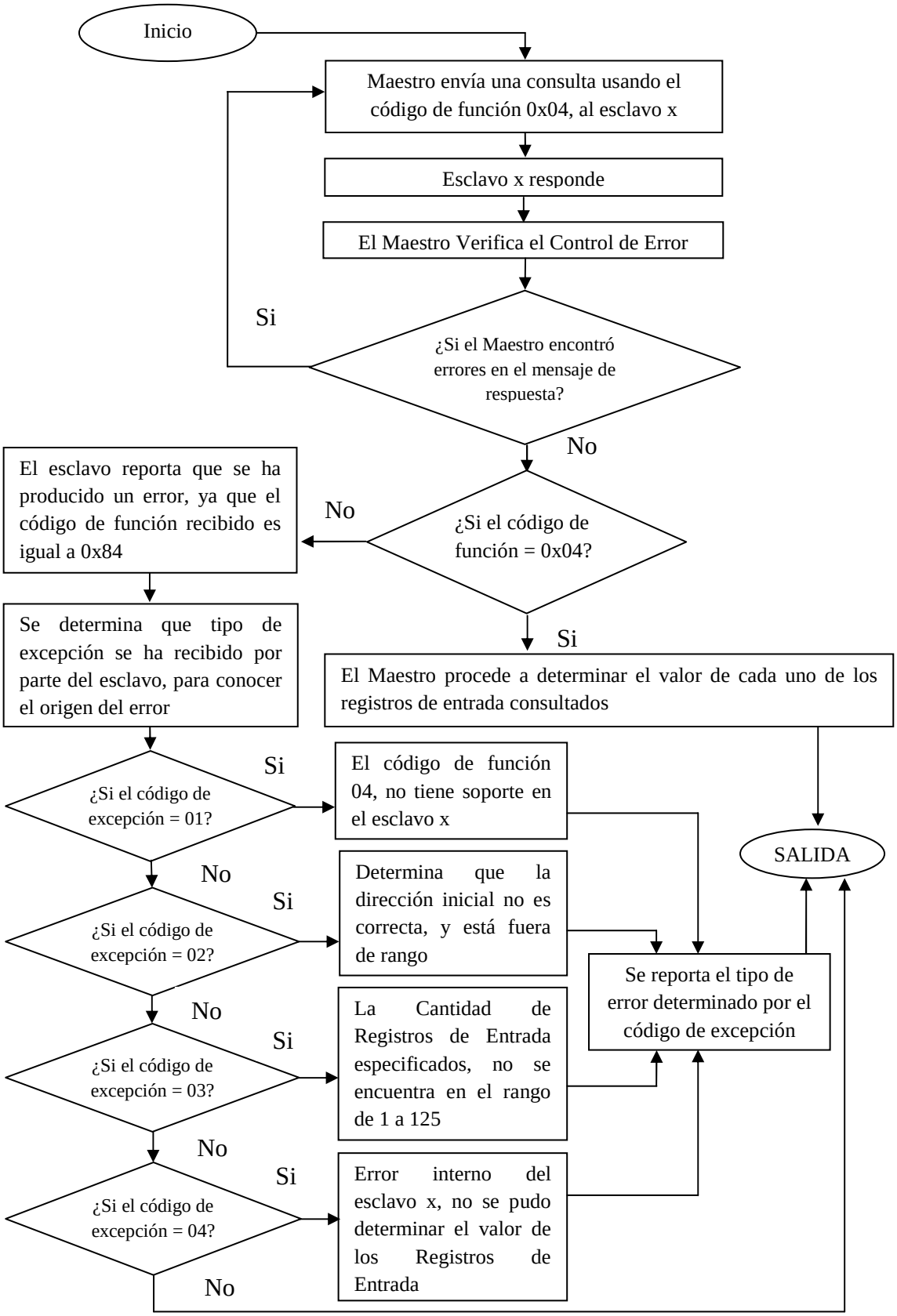

Figura 2.7: Diagrama de flujo de la lectura de registros de entrada

Ejemplo

La siguiente tabla describe los valores de los campos de los mensajes de consulta y respuesta del ejemplo de uso de la función 0x04.

Consulta		***Respuesta***	
Nombre del Campo	**Hex**	**Nombre del Campo**	**Hex**
Función	04	**Función**	03
Dirección Inicial (Byte 1)	00	**Conteo de Bytes**	06
Dirección Inicial (Byte 2)	46	**Valor del Registro 77 (Byte 1)**	02
Cantidad de Bobinas (Byte 1)	00	**Valor del Registro 77 (Byte 2)**	10
Cantidad de Bobinas (Byte 2)	03	**Valor del Registro 78 (Byte 1)**	00
		Valor del registro 78 (Byte 2)	53
		Valor del registro 79 (Byte 1)	00
		Valor del registro 79 (Byte 2)	0A

Tabla 2.19: Valores de los campos de los mensajes de consulta y respuesta del ejemplo usando la función 0x04 (Estado de registros de entrada 77, 78 y 79)

Análisis

La **consulta** se realiza para determinar el valor de los registros de entrada 77 (dirección inicial 76, 0x46 en hexadecimal), 78 y 79.

La **respuesta** contiene el valor de la función 0x04, y el valor 0x06 el cual representa el número de bytes que contienen los valores de los tres registros especificados en la consulta. El campo de datos determina los valores de los registros internos 77, 78 y 79, El valor del registro 77 es de 0x0210, que en decimal equivale a 528, en cambio que el valor del registro 78 es de 0x53, que en decimal equivale a 83; finalmente, el valor del registro 79 es 0x0A, que en decimal equivale a 10.

05 (0x05) Escribir bobinas individualmente

Descripción

La función 05 es usada para escribir una salida individualmente. Permite cambiar el estado de una bobina a Encendido (ON) o a Apagado (OFF). Las bobinas en un dispositivo remoto son referenciadas como 0x [6].

Implementación de consulta

La consulta especifica la dirección de la bobina a ser forzada. Las bobinas son direccionadas comenzando desde 0. Por lo tanto, la primera bobina será direccionada como 0. El valor Encendido/Apagado (ON/OFF) requerido es especificado por una constante en el campo de valor de la bobina. Un valor de 0xFF00 requiere que la bobina sea encendida (ON). Un valor de 0x0000 requiere que la bobina sea apagada (OFF). Todos los demás son ilegales y no afectarán el estado de una determinada bobina. La tabla 2.20 especifica los campos dentro de la consulta:

Campo	Definición	Tamaño	Valores
Código de Función	Función especifica para forzar bobinas individualmente	1 Byte	0x05
Dirección de la Bobina	Dirección de de la bobina a ser forzada	2 Bytes	0x0000 a 0xFFFF
Valor de la Bobina	Determina a que estado será forzada la bobina direccionada	2 Bytes	0x0000 o 0xFF00

Tabla 2.20: Campos de un mensaje de consulta usando la función 0x05

Implementación de respuesta

La respuesta por parte del esclavo, es un eco de la consulta, es decir si no ha ocurrido ningún error, el esclavo transmite los mismos valores de los campos especificados en la consulta.

Error

La siguiente tabla describe la respuesta de error específica para la función 0x05

Campo	Definición	Tamaño	Valores
Código de Función	Código de error para la función 0x05	1 Byte	0x85
Código de Excepción	Determina el error específico encontrado en la consulta.	1 Bytes	01 o 02 o 03 o 04

Tabla 2.21: Campos de un mensaje de respuesta de excepción usando la función 0x05

La figura 2.8 describe un diagrama de flujo de la escritura individual de bobinas, desde el punto de vista del maestro.

Ejemplo

La siguiente tabla describe los valores de los campos de los mensajes de consulta y respuesta del ejemplo de uso de la función 0x05.

Consulta		***Respuesta***	
Nombre del Campo	**Hex**	**Nombre del Campo**	**Hex**
Función	05	**Función**	05
Dirección Bobina (Byte 1)	00	**Dirección Bobina (Byte 1)**	00
Dirección Bobina (Byte 2)	2E	**Dirección Bobina (Byte 2)**	2E
Valor de la Bobina (Byte 1)	FF	**Valor de la Bobina (Byte 1)**	FF
Valor de la Bobina (Byte 2)	00	**Valor de la Bobina (Byte 2)**	00

Tabla 2.22: Valores de los campos de los mensajes de consulta y respuesta del ejemplo usando la función 0x05 (Cambiar a encendido (1) la bobina 47)

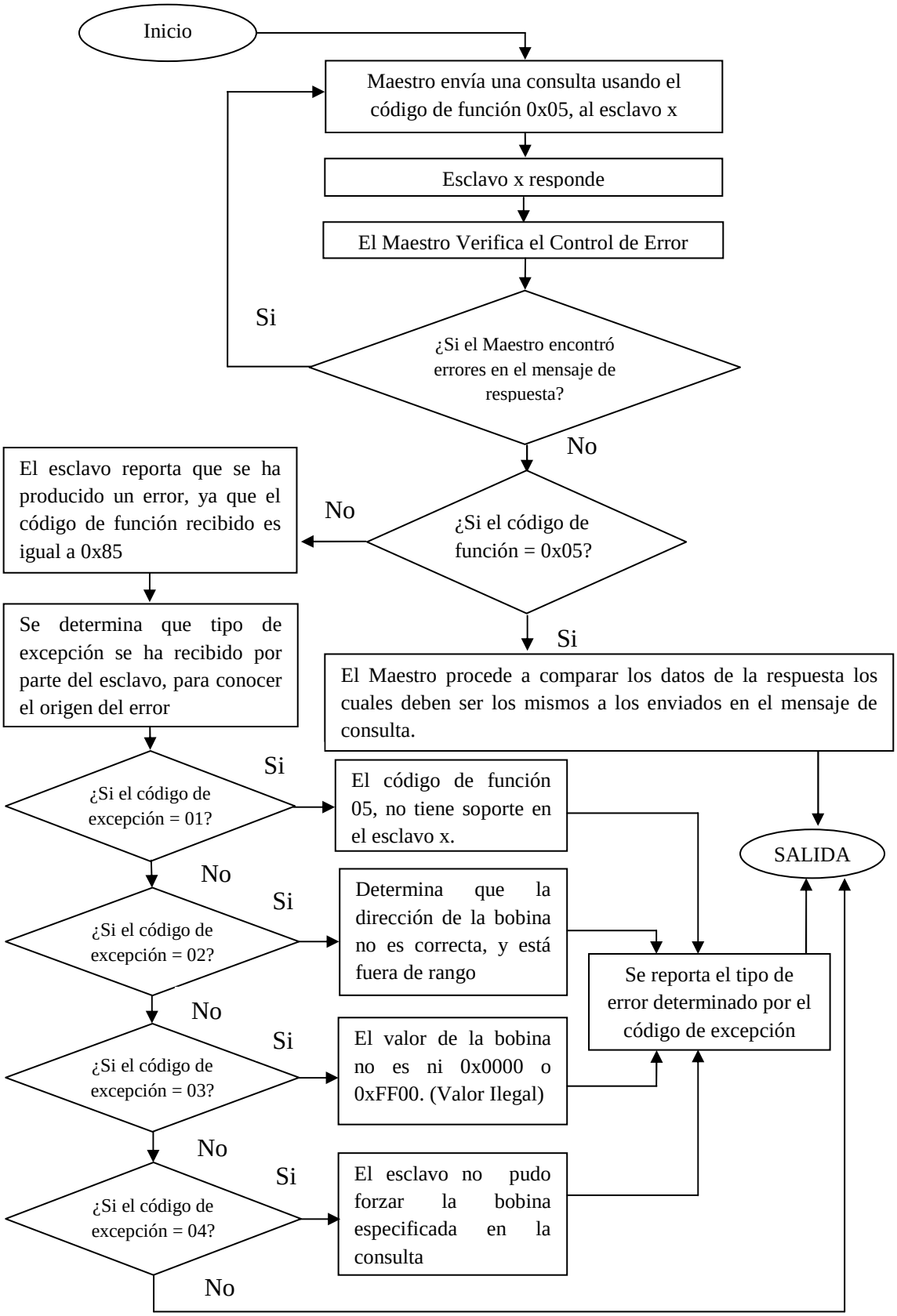

Figura 2.8: Diagrama de flujo de la escritura de bobinas individualmente

Análisis

La **consulta** se realiza para forzar la bobina 47 (dirección 46, 0x2E en hexadecimal) a un estado de encendido (ON).

La **respuesta** es un eco del mensaje de consulta, es decir el esclavo transmite los mismos valores de los campos del mensaje de consulta.

06 (0x06) Escribir registros individualmente

Descripción

La función 06 es usada para fijar un valor en un registro interno de un dispositivo remoto. Los registros internos son referenciados como 4x [6].

Implementación de consulta

La consulta especifica la dirección del registro al cual se va a modificar su valor. Los registros internos son direccionados comenzando desde 0. Por lo tanto, el primer registro interno será direccionado como 0. La tabla 2.23 especifica los campos dentro de la consulta:

Campo	Definición	Tamaño	Valores
Código de Función	Función especifica para escribir registros internos individualmente	1 Byte	0x06
Dirección del registro interno	Dirección del registro al cual se le asignará un nuevo valor	2 Bytes	0x0000 a 0xFFFF
Valor del Registro Interno	Valor que será asignado al registro direccionado	2 Bytes	0x0000 a 0xFFFF

Tabla 2.23: Campos de un mensaje de consulta usando la función 0x06

Implementación de respuesta

La respuesta normal por parte del esclavo, es un eco de la consulta, enviado después de que se ha escrito el nuevo valor en el registro direccionado de la consulta.

Error

La siguiente tabla describe la respuesta de excepción error específica para la función 0x04.

Campo	Definición	Tamaño	Valores
Código de Función	Código de error para la función 0x06	1 Byte	0x86
Código de Excepción	Determina el error específico encontrado en la consulta.	1 Bytes	01 o 02 o 03 o 04

Tabla 2.24: Campos de un mensaje de respuesta de excepción usando la función 0x06

La figura 2.9 muestra un diagrama de flujo de la ecritura de registros internos individualmente.

Ejemplo

La siguiente tabla describe los valores de los campos de los mensajes de consulta y respuesta del ejemplo de uso de la función 0x06.

Consulta		***Respuesta***	
Nombre del Campo	**Hex**	**Nombre del Campo**	**Hex**
Función	06	**Función**	06
Dirección del Registro (Byte 1)	00	**Dirección del Registro (Byte 1)**	00
Dirección del Registro (Byte 2)	07	**Dirección del Registro (Byte 2)**	07
Valor del Registro (Byte 1)	00	**Valor del Registro (Byte 1)**	00
Valor del Registro (Byte 2)	1D	**Valor del Registro (Byte 2)**	1D

Tabla 2.25: Valores de los campos de los mensajes de consulta y respuesta del ejemplo usando la función 0x06 (Fijar el valor de 29 decimal en el registro 07)

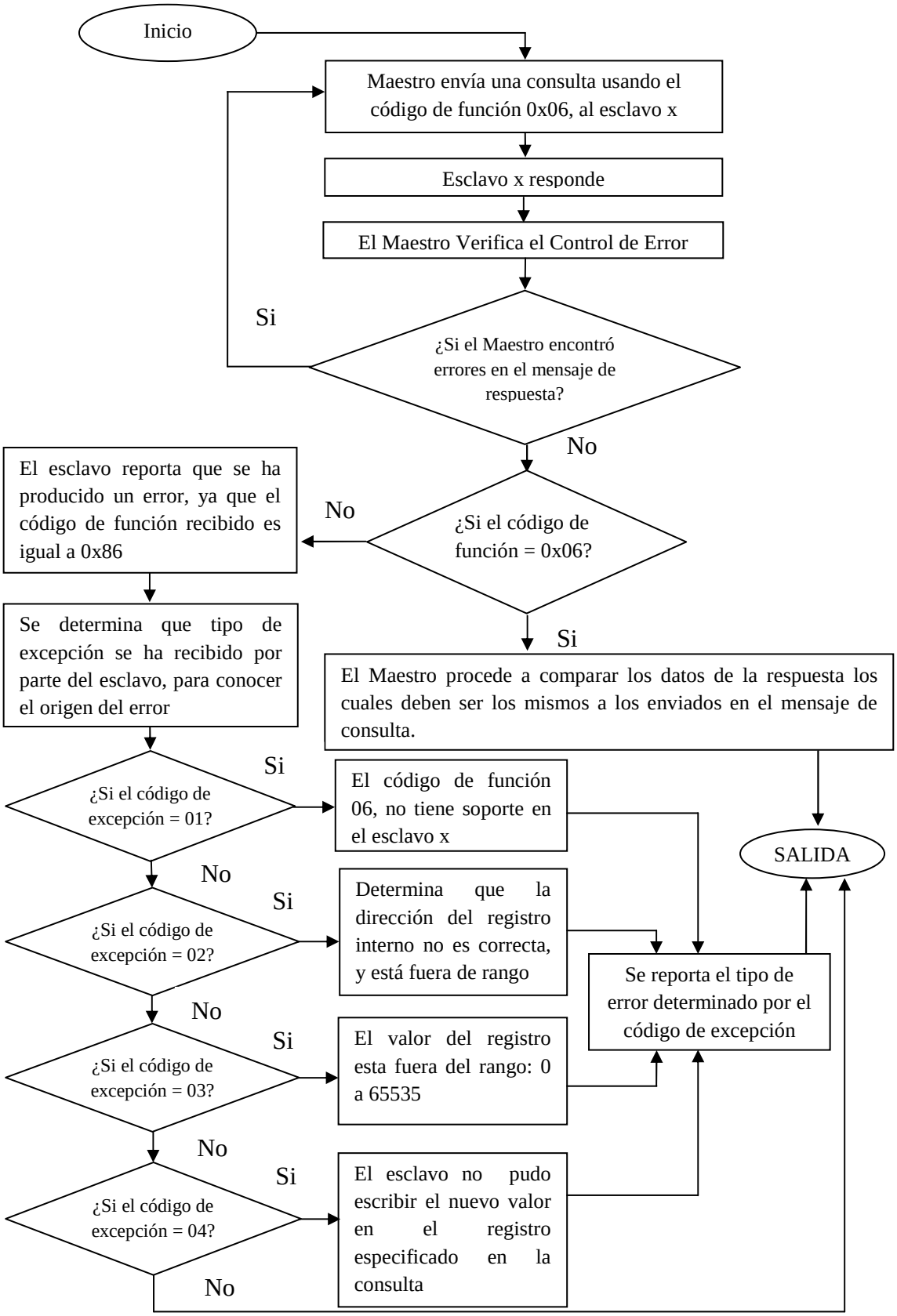

Figura 2.9: Diagrama de flujo de la escritura de registros individualmente

Análisis

La **consulta** se realiza para escribir el valor 29 (0x1D en hexadecimal) en el registro interno 08 (dirección 0x07).

La **respuesta** es un eco del mensaje de consulta, es decir el esclavo transmite los mismos valores de los campos del mensaje de consulta.

Control de Redundancia Cíclica CRC

- El Protocolo de comunicación Modbus RTU incluye un campo de comprobación de errores basado en el método de control de redundancia cíclica, ejecutado sobre el contenido de los mensajes.
- El campo CRC verifica el contenido de todo el mensaje. Es aplicado sin importar el método de paridad usado para la transmisión individual de los caracteres del mensaje.
- El campo CRC contiene un valor de 16 bits, implementado en 2 bytes de 8 bits cada uno.
- El campo CRC se lo añade al final del mensaje como el último campo de la trama del mismo. El byte de menor orden es transmitido primero, seguido por el byte de mayor orden. El byte de mayor orden del campo de CRC es el último byte transmitido del mensaje Modbus RTU[5].
- El valor CRC es calculado por el dispositivo que transmite el mensaje, el cual lo añade al mismo. El dispositivo receptor calcula nuevamente el CRC durante la recepción del mensaje, y compara el valor calculado al valor actual recibido en el campo CRC. Si los dos valores no son iguales, se genera un error [5].

El proceso para generar el campo de CRC es [5]:

1. Cargar un registro de 16 bits con 0xFFFF (todos los bits 1). A este registro se lo conoce como el registro CRC.
2. Ejecutar un OR Exclusivo entre el primer byte (8 bits) del mensaje con el byte de menor orden del registro de 16 bits CRC, colocando el resultado en el registro CRC

3. Rotar el registro CRC un bit a la derecha. Se coloca un cero en el bit más significativo. Se extrae y examina el bit menos significativo.
4. Si el bit menos significativo fue 0, entonces se repite el paso 3 (otra rotación).
   Si el bit menos significativo fue 1, entonces se debe realizar un OR exclusivo entre el registro CRC y un registro conocido como el polinomio generador cuyo valor es 0xA001 (para Modbus RTU) (1010 0000 0000 0001).
5. Repetir los pasos 3 y 4 hasta que un total de 8 rotaciones hayan sido realizadas. Al terminar, un byte de 8 bits ha sido procesado.
6. Repetir el procedimiento desde el paso 2 hasta el paso 5 para los siguientes bytes de 8 bits del mensaje. Continuar con este procedimiento hasta que todos los bytes del mensaje hayan sido procesados.
7. El contenido final del registro CRC es el valor CRC.
8. Cuando el campo CRC sea adjuntado al mensaje se debe colocar el valor de tal forma que el byte de menor orden sea transmitido primero.

2.8 Sistemas de Gestión de Bases de Datos

En los sistemas Scada actuales existe cada vez más una mayor demanda de datos. Esto se debe a que los sistemas Scada en la actualidad no solo controlan la operatividad del proceso, sino también son usados para tomar decisiones tanto operativas como administrativas. Como resultado, los sistemas actuales deben tener la capacidad de acceso y administración de los datos del proceso, no solo a nivel de planta, sino también a nivel administrativo. Los sistemas de gestión de bases de datos en los procesos industriales tienen su origen en carpetas y fichas en las que se almacenaban los datos de manera manual (un operador tomaba los datos que presentaban los instrumentos analógicos ubicados en el campo en un determinado intervalo de tiempo). En la actualidad los sistemas de gestión de bases de datos son sistemas de información integrales (almacenan y administran la información de un proceso), esto debido a la necesidad de gestionar las acciones que realiza una determinada industria.

2.8.1 Descripción de Sistemas de Gestión de Base de Datos

El objetivo principal de un sistema de gestión de base de datos, es que las aplicaciones puedan acceder a los datos sin necesidad de conocer la forma en la que están almacenados los mismos [7].

2.8.2 Elementos en un sistema de base de datos

Los elementos de un sistema de base de datos son los siguientes:

- **Datos:** Constituyen toda la información que se necesita almacenar de un determinado proceso o sistema.
- **Hardware:** Dispositivo físico (Servidor) en el cual se almacenan las bases de datos.
- **Software:** Es el sistema que permite gestionar las bases de datos (ejemplo: MySQL).
- **Usuarios:** Son las personas o aplicaciones que hacen uso de los datos del sistema.

2.8.3 Tipos de Datos

Las bases de datos están compuestas por los siguientes tipos de datos:

- **Datos almacenados**: Son los datos obtenidos del sistema y los cuales son administrados por el usuario mediante el sistema de gestión de base de datos [7].
- **Datos complementarios:** Son los datos que especifican la estructura de la base de datos; por ejemplo, el tipo de datos almacenados (si son texto, números, entre otros), o el nombre del campo al que pertenece cada dato (nombre de la variable, valor, entre otros) [7].

2.8.4 Administración y funciones dentro de una base de datos

El usuario debe tener la posibilidad de modificar y utilizar los datos de una base de datos. El usuario o aplicación puede realizar las siguientes acciones:

- **Buscar datos:** Los sistemas de gestión de bases de datos permiten al usuario obtener datos, a partir de condiciones de búsqueda establecidas (consultas).
- **Añadir datos:** El usuario puede añadir datos en el instante que sea necesario.

- **Modificar datos:** Los sistemas de gestión de bases de datos permiten al usuario modificar los datos y los campos que definen los tipos de datos almacenados, cuando sea necesario.
- **Eliminar datos:** El usuario puede eliminar los datos que no están siendo utilizados en cualquier instante de tiempo.

2.8.5 Aplicaciones y Ventajas de las Bases de Datos

Actualmente las bases de datos son parte de nuestra vida diaria, ya que tienen aplicación en diversos campos. Podemos nombrar aplicaciones que van desde bases de datos que almacenan el historial de los alumnos de una determinada escuela, hasta bases de datos que almacenan los valores de miles de variables en un sistema de control de transporte de crudo a través de un oleoducto.

Por otro lado, la necesidad de intercambiar información a altas velocidades y de manera fiable en el mundo actual, ha hecho de las bases de datos elementos indispensables [7]. Entre las principales ventajas que tiene el usuario al implementar una base de datos tenemos:

- **Generación de reportes:** Las bases de datos permiten a los usuarios obtener información sobre el sistema de manera flexible.
- **Velocidad:** Las bases de datos permiten a usuarios y aplicaciones almacenar y obtener de manera asíncrona y rápida la información.
- **Precisión:** Las bases de datos administran de manera adecuada y con mucho cuidado los datos de entrada; en consecuencia, proveen resultados precisos y consistentes de sus datos.
- **Reportes detallados:** Las bases de datos pueden almacenar y generar resultados completos y detallados a altas velocidades.

2.8.6 Bases de datos Relacionales:

Las bases de datos relacionales son un tipo de bases de datos basado en relaciones o tablas [7]. Las características principales de una base de datos relacional son las siguientes:

- Una base de datos relacional se compone de varias tablas o relaciones (interconexiones).

- Las tablas que componen una base de datos relacional son un conjunto de registros, filas o tuplas.
- No puede existir dos tablas con el mismo nombre (no permite la interconexión).
- El lenguaje estándar para generar consultas a bases de datos relacionales es SQL (Structured Query Language).
- Garantiza una integridad referencial. Si se elimina un registro, se eliminan todos los registros dependientes.
- La estructura referencial permite generar sistemas escalables.

Finalmente, existe una variedad de sistemas de gestión de bases de datos relacionales. Entre los más conocidos tenemos:

- **MySQL** (Implementado en CM)
- Microsoft SQL Server
- Oracle
- PostgreSQL
- DB2

2.9 Sistemas de Visualización de Datos: Interfaces Hombre Máquina (HMI: Human Machine Interface)

2.9.1 Definición

La interfaz Hombre-Máquina puede describirse como una ventana o una descripción gráfica de un proceso. El HMI (por sus siglas en inglés) recibe las señales de las variables involucradas en un proceso a través de un Servidor Scada o grupo de Servidores Scada, los cuales se encargan de la obtención y almacenamiento de los datos provenientes de los dispositivos de campo [4].

2.9.2 Funciones de aplicaciones HMI

- **Monitorear el proceso:** El HMI tiene la capacidad de obtener y mostrar datos de las variables de un determinado proceso en tiempo real. La aplicación HMI presenta estos datos mediante gráficas, texto, o valores numéricos [4].

- **Generar Históricos y Gráficas de Tendencia:** El HMI permite mostrar gráficas del comportamiento pasado de la planta, y así verificar las condiciones operativas del proceso en cualquier instante de tiempo. Además, tiene la capacidad de generar gráficas en tiempo real, para verificar el comportamiento de una determinada variable del proceso.
- **Mostrar alarmas y Generar reportes:** La aplicación HMI muestra gráficamente mediante tablas y señales visuales alarmas generadas por condiciones fuera de los rangos preestablecidos de las variables del proceso o por errores dentro del sistema. Además, presentan al operador reportes del comportamiento del sistema, resaltando eventos importantes encontrados.

2.9.3 Desarrollo de Aplicaciones HMI

Las aplicaciones HMI son un conjunto de despliegues que le permiten al operador determinar el comportamiento en tiempo real del proceso. El software de desarrollo de sistemas HMI/Scada presentan módulos que permiten generar despliegues. Estos despliegues son generados y caracterizados de acuerdo al proceso que representan. Sin embargo, existen organizaciones que desarrollan estándares con el objetivo de normalizar el desarrollo de interfaces Hombre-Máquina dentro de sistemas de control distribuido. El objetivo de estas organizaciones es que los desarrolladores de despliegues sigan ciertas reglas a la hora de representar a las variables de manera gráfica o mediante identificaciones (Tags). Además, intentan normalizar el uso de colores y los significados de los mismos. Como resultado, se tiene una normalización en la presentación del proceso desde el punto de vista estructural, pero no así desde el punto de vista funcional, que depende de las características del proceso.

2.9.4 Estándar ANSI /ISA 5.1 1984 (R1992)

Identificación y símbolos de Instrumentación

Propósito

El propósito de este estándar es establecer medios uniformes a la hora de identificar instrumentos y sistemas de instrumentación. Genera un sistema de designación de símbolos y códigos de los instrumentos y sistemas de instrumentación [8].

Aplicación en la Industria

El estándar es adecuado para el uso en la industria química, petroquímica, generación de energía, y entre otros numerosos procesos industriales.

Aplicaciones en actividades de trabajo

El estándar ANSI/ISA 5.1 tiene su aplicación especialmente en las siguientes áreas:

- Diseño de planos de construcción.
- Ejemplos académicos.
- Documentos científicos, literatura, y discusiones.
- Diagramas de sistemas de instrumentación, diagramas tipo bucle, o diagramas lógicos.
- Especificaciones, entre otros.
- Identificación (**tagging**) de los instrumentos y funciones de control.

El estándar tiene la intención de proveer la suficiente información que permita a cualquier persona conocedora del proceso, y que este revisando un documento de control de procesos o un despliegue en un sistema de control, entender los medios de medida y control del proceso. Los símbolos y el sistema de identificación provistos por el estándar son aplicables para toda clase de medida dentro del proceso y a toda la instrumentación de control [8].

Definiciones

El estándar define diversos términos de aplicación en los procesos industriales, entre los principales términos definidos para el uso y entendimiento del estándar se tiene los siguientes [8]:

Alarma (Alarm): Un dispositivo o función que señala la existencia de una condición que no es normal, por medio de un cambio discreto visible y/o audible, con el objetivo de atraer la atención del operador.

Binario (Binary): Término aplicado a una señal o dispositivo que posee dos estados o posiciones discretas, contrario a una señal análoga, el término es usado para identificar o denotar dos estados por ejemplo: “prendido-apagado”, “alto-bajo”, “abierto-cerrado”

Dispositivo de Computación (Computing Device): Dispositivo que permite realizar uno o varios cálculos, operaciones lógicas, o ambas y transmite los resultados mediante señales de salida.

Controlador (Controller): Un dispositivo que tiene una salida que varía para regular una variable controlada de una manera específica. Existen controladores automáticos los cuales varían su salida automáticamente en respuesta a una señal de entrada directa o indirecta de una variable medida del proceso. Por otro lado, tenemos el controlador manual el mismo que no depende del valor de una variable medida y puede ser asignado de manera manual mediante un ajuste.

Conversor (Converter): Un dispositivo que recibe una señal de un instrumento de una forma, y transmite la señal de salida de otro forma.

Digital: Un término aplicado a una señal o dispositivo que usa datos binarios para representar datos continuos o discretos.

Sistema de control Distribuido (Distributed Control System): Un sistema que es funcionalmente integrado, y consiste de varios subsistemas que se encuentran físicamente separados y remotamente ubicados uno de otro.

Elemento de Control Final (Final Control Element): Un dispositivo que directamente controla el valor de una variable manipulada de un determinado bucle de control.

Identificación (Identification): La secuencia de letras o dígitos usados para designar un instrumento individual.

Instrumento (Instrument): Un dispositivo usado directa o indirectamente para medir y/o controlar una determinada variable. El término incluye: elementos primarios, elementos finales de control, dispositivos de computación, y dispositivos eléctricos.

Bucle (Loop): La combinación de dos o más instrumentos o funciones de control agrupadas para que las señales pasen de un lugar a otro con el propósito de medir y/o controlar una variable del proceso.

Medida (Measurement): Magnitud de una variable.

Elemento primario (Primary Element): Sinónimo de sensor.

Proceso (process): Cualquier operación o secuencia de operaciones que envuelven el cambio de energía, estado, composición, dimensión u otras propiedades.

Variable del proceso (Process variable): Cualquier variable proveniente del proceso. El término variable del proceso en este estándar es usado para todas las variables cuyas señales provienen de la instrumentación

Programa (Program): Una secuencia repetitiva de acciones que define el estado de las salidas que son modificadas en relación a un grupo de entradas.

Controlador Lógico Programable (Programmable logic controller): Controlador con múltiples entradas y salidas que contiene un programa alterable.

Sensor: Parte de un bucle o instrumento que primero determina el valor de una variable del proceso y asume una correspondiente y predeterminada salida. El sensor también es conocido como elemento primario.

Transductor (Transducer): Término general para un dispositivo que recibe información en una o varias cantidades físicas, modifica la información y/o su forma y produce una señal de salida resultante.

Descripción del Sistema de Identificación:

Cada instrumento o función a ser identificada es designada por un código alfanumérico o tag como se muestra en la tabla 2.26.

Características de las identificaciones (Tags)

- La identificación dentro de un bucle de control determinado generalmente es común a todos los instrumentos y funciones que componen un bucle de control.
- El usuario puede adicionar prefijos o sufijos de manera opcional, y así completar la información necesaria para describir el instrumento.
- El número del bucle de control incluye información codificada, tal como área determinada de la planta, o función dentro del proceso, etc.
- La función de un determinado instrumento se identifica mediante una primera letra de identificación que designa la medida o la variable, seguido de una o varias letras subsiguientes que designan las funciones realizadas.

Composición típica de una identificación o tag	
TIC 103	Identificación del instrumento (TAG)
T 103	Identificación dentro de un Bucle de control determinado.
103	Número del Bucle de control determinado
TIC	Identificación funcional
T	Primera letra de identificación
IC	Letras subsiguientes de identificación
Identificación o tag expandida	
10-PAH-5A	Identificación del instrumento
10	Prefijo opcional
A	Sufijo opcional

Tabla 2.26: Ejemplo de designación de la identificación de un instrumento usando el estándar ANSI/ISA 5.1

En la tabla 2.27 se presenta la identificación de las letras, en la cual se define y explica la designación usada para cada letra individualmente.

Análisis de la Tabla

- Una letra cuya designación es “Opción de usuario” será utilizada de manera repetida en un proyecto en particular, y tendrá que ser presentado su significado tanto para una primera letra, o para letras subsiguientes.
- Una letra cuya designación es “no clasificada”, será utilizada en una ocasión, o en un número limitado de ocasiones. Las Definiciones deben ser especificadas fuera del diagrama de flujo del proceso.
- Si una primera letra es usada en combinación con una letra modificadora D (Diferencial), F (Radio), M (Momentáneo), K (Tasa de Cambio), Q (Totalizador), o cualquier combinación de estas letras es usada para representar una nueva y separada variable de medida. Dicha combinación es tratado como una primera letra. Por lo tanto, los instrumentos PDI y PI identifican dos variables diferentes,

designadas como: Indicador de Presión Diferencial e Indicador de Presión. Los Modificadores deben ser usados solo cuando sean aplicables.

- Los términos modificadores alto, bajo, medio e intermedio corresponden a valores de la variable medida. Por ejemplo, una alarma de nivel alto debe ser designada como LAH (Primera Letra = “Level”, Función Pasiva= “Alarm”, Modificador=”High”) [8].

	Primera Letra		**Letras Subsiguientes**		
	Medida o inicio de variables	**Modificador**	**Lectura de Salida o Función Pasiva**	**Función de Salida**	**Modificador**
A	Análisis		Alarma		
B	Quemador, Combustión		Opción de usuario	Opción de usuario	Opción de usuario
C	Opción de usuario			Control	
D	Opción de usuario	Diferencial			
E	Voltaje		Sensor (Elemento Primario)		
F	Flujo	Radio (Fracción)			
G	Opción de usuario		Vidrio, Dispositivo de visualización		
H	Mano				Alto
I	Corriente Eléctrica		Indicador		
J	Potencia	Escaneo			
K	Tiempo, horario	Tasa de cambio		Estación de Control	
L	Nivel		Luz		Bajo
M	Opción de usuario	Momentáneo			Medio, Intermedio
N	Opción de usuario		Opción de usuario	Opción de usuario	Opción de usuario
O	Opción de usuario		Orificio, Restricción		
P	Presión, Vacío		Punto de Prueba de conexión		
Q	Cantidad	Integrado, Totalizador			
R	Radiación		Registro		
S	Velocidad, Frecuencia	Seguridad		Conmutador	

T	Temperatura			Transmisión	
U	Multivariable		Multifunción	Multifunción	Multifunción
V	Vibración, Análisis Mecánico			Válvula, Amortiguador	
W	Peso, Fuerza		Pozo		
X	No Clasificado	Eje X	No Clasificado	No Clasificado	No clasificado
Y	Evento, Estado o Presencia	Eje Y		Relé, Conversor, cálculo	
Z	Posición, Dimensión	Eje Z		Actuador, Elemento final de Control no clasificado	

Tabla 2.27: Significado e identificación de letras, para la designación de instrumentos funciones o variables mediante tags

Si se requiere mayor información sobre la designación de indicadores o Tags de variables, se debe referir al Estándar ANSI/ISA-5.1-1984(R1992) Símbolos de Instrumentación e Identificación del Estándar Nacional Americano (ANSI American National Standards Institute) y la Sociedad de Instrumentación, Sistemas y Automatización (ISA The Instrumentation, Systems, and Automation Society).

2.9.5 Estándar ISA-5.5-1985

Símbolos Gráficos para Visualización de Procesos

El propósito de este estándar es desarrollar un sistema de símbolos gráficos y colores para los despliegues del proceso que son usados para controlar y monitorear un determinado proceso industrial. El sistema tiene como objetivo facilitar la rápida comprensión por parte del usuario de la información que es presentada en los diferentes despliegues o pantallas de visualización, y establecer una uniformidad en la presentación de la información por parte de las diferentes industrias. Además, el estándar se complementa con el Estándar ISA-5.1 “Símbolos de Instrumentación e Identificación”

Los resultados que se buscan al desarrollar este estándar son los siguientes:

- Reducir los tiempos de entrenamiento de los operadores.
- Reducir los errores de los operadores.

- Mejor comunicación y entendimiento entre el diseñador del sistema de control y los usuarios del sistema.

En este documento se desarrolla la distribución de colores en una pantalla de visualización del proceso debido a que el sistema CM desarrollado utiliza despliegues sin símbolos gráficos de instrumentos. Para mayor información sobre los símbolos gráficos de los instrumentos del proceso, se debe referir al Estándar ISA-5.5-1985 desarrollado por la Sociedad de Instrumentación, Sistemas, y Automatización (ISA).

Color

El color es una técnica efectiva de codificación con símbolos, formas y códigos alfanuméricos.

La siguiente tabla muestra un ejemplo de planificación de colores, que establece una relación entre los colores, su significado genérico, y la asociación con los diferentes significados y procesos dentro de una pantalla de visualización del proceso [9].

Color	**Significado Genérico**	**Elemento Asociado**
Negro	Fondo	
Rojo	Emergencia	a) Parada b) Alarma de alta prioridad c) Cerrado d) Apagado (OFF)
Amarillo	Alerta	a) Condición fuera de lo normal b) Alarma de prioridad media
Verde	Seguro	a) Operación Normal b) Inicio c) Abierto d) Encendido (ON)
Celeste	Estático y significante	a) Equipo de Proceso en servicio b) Etiquetas Principales
Azul	No es esencial	a) Equipo de Proceso en espera b) Tags, etiquetas, etc.
Morado	Radiación	a) Alarmas de Radiación b) Valores Cuestionables
Blanco	Datos Dinámicos	a) Medidas e informaciones de estado b) Mensajes del sistema c) Tendencias d) Pasos secuenciales activos

Tabla 2.28: Significado de Colores Sugerido, para el diseño de despliegues

Finalmente, el diseñador de un determinado proyecto debe establecer antes del desarrollo, las asociaciones específicas entre colores y elementos. En algunos casos especiales, como la industria de generación y transmisión de Energía, los colores pueden tener significados diferentes a los recomendados por el estándar. En este caso en particular el rojo indica cerrado, activo o unidad energizada o rotando. Este cambio de significado de un color es aceptado, siempre y cuando los significados de los colores sean definidos anteriormente para un proyecto en particular [9].

2.10 Software de Desarrollo de Sistemas HMI/SCADA

2.10.1Antecedentes y Descripción del Software de Desarrollo de sistemas HMI/Scada existente.

Las primeras herramientas para la creación de sistemas Scada fueron desarrolladas para aplicaciones específicas, dependiendo de las características del proceso a supervisar y controlar. Por esta razón, los sistemas Scada eran adaptados a las necesidades de un proceso específico. Como resultado, los proveedores de software de desarrollo de sistemas Scada adaptaron su trabajo previo en aplicaciones específicas, para que el software pueda ser utilizado en otro tipo de industria. Con el tiempo, varios fabricantes desarrollaron paquetes de software capaces de comunicarse con los sistemas y dispositivos de control de una determinada planta, y le dieron al sistema en general escalabilidad y flexibilidad. Sin embargo, ciertos procesos requerían de aplicaciones adicionales, las cuales fueron desarrolladas como módulos específicos [4]. En la actualidad, el objetivo de los proveedores de software para sistemas HMI/Scada es desarrollar una arquitectura abierta que permita su utilización en diversos procesos industriales, con la adición de módulos específicos para determinadas industrias. Por ejemplo, los sistemas Scada en las industrias de potencia, tienen incorporados módulos específicos conocidos como sistemas de administración de Energía (EMS Energy Management System) y sistemas de administración de Generación (GMS Generation management system), entre otros. Sin embargo, los proveedores mantienen todavía la preferencia hacia determinados procesos, por lo que si se quiere desarrollar un sistema Scada sobre un proceso no muy común, los ingenieros deben estar preparados para generar los sistemas adicionales que dicho proceso pueda necesitar. Por esta razón, las industrias escogen a sus proveedores en función al

conocimiento que tienen los mismos sobre el proceso. En conclusión, si se contacta con un proveedor con conocimientos amplios sobre el proceso, el desarrollo de sistemas adicionales será mínimo, pero el costo será mayor, en cambio si se contacta con un proveedor con conocimientos limitados, el costo será menor, pero el sistema requerirá del desarrollo de sistemas adicionales. Actualmente, existen numerosas opciones de fabricantes de software para el desarrollo de sistemas HMI/Scada. Entre los más conocidos tenemos:

- Intellution: IFIX
- Wonderware: InTouch
- Siemens: WinCC
- Rockwell Automation: RS-View
- GE-Fanuc Cimplicity

Sin embargo, todas estas aplicaciones son de tipo propietario y tienen un determinado costo. Como resultado, en ocasiones industrias medianas y pequeñas no pueden acceder a este software para su proceso debido al alto costo de la Licencia de distribución de la aplicación [4].

2.10.2 Funciones del Software de desarrollo de Sistemas HMI/Scada

- **Supervisar el proceso:** El sistema HMI/Scada genera aplicaciones que permiten modificar el estado de una determinada variable de manera remota, a partir del análisis del proceso en un determinado instante de tiempo.
- **Generar Reportes y Alarmas:** El Sistema HMI/Scada permite configurar el sistema para determinar si ha ocurrido un evento no deseado dentro del sistema para después generar la alarma y el reporte respectivo. Estas alarmas se determinan a partir de límites establecidos por el supervisor del sistema.
- **Generar Algoritmos de Control:** Actualmente el software para desarrollo de HMIs permite generar algoritmos de control. Es decir, permite modificar o ajustar el valor de una determinada variable (variable manipulada) del proceso a partir de los valores de ciertas señales de entrada, con el objetivo de mantener una variable (variable controlada) del proceso dentro de valores preestablecidos.

- **Configura el sistema de Comunicación:** Mediante el software de desarrollo de sistemas HMI/Scada, se configura los canales de comunicación con los diversos dispositivos de campo.
- **Desarrollar despliegues:** El software de desarrollo de sistemas Scada tiene módulos que permiten generar los despliegues que describen gráficamente el proceso.
- **Configurar los sistemas de Gestión de Base de datos:** El software de desarrollo de sistemas HMI/Scada permite configurar la base de datos del proceso y las funciones específicas a realizar con los datos almacenados. Mediante este módulo se estructura el sistema de control y supervisión del proceso, ya que define el grupo de variables involucradas en el sistema.

3

Software Libre (Free Software) y Software de Código Abierto (Open Source Software)

En la actualidad existen dos movimientos importantes que promueven el desarrollo de Software de libre distribución, estos son:

- La Fundación de Software Libre (Free Software Foundation).
- La Iniciativa de Código Abierto (Open Source Initiative).

3.1 Fundación de Software Libre (Free Software Foundation)

3.1.1 Descripción General

La **Fundación de Software libre,** desarrolló la Licencia Pública General (GPL: General Public Licence) GNU/GPL, la misma que promueve la libre distribución de Software, incluyendo su código fuente. El objetivo principal es permitir a la comunidad de programadores realizar cambios al código fuente. De acuerdo a la licencia GNU/GPL ninguna aplicación que declare el uso de esta licencia puede ser distribuida sin su código fuente. Compartir entre los programadores el código fuente permite responder a defectos y problemas en las aplicaciones de manera más rápida y efectiva. Por ejemplo, un problema detectado en una distribución de Linux puede solucionarse en días; en cambio, aplicaciones comerciales pueden tardar meses en solucionar un determinado error [10].

3.1.2 Características definidas por la Fundación de Software Libre

Las siguientes características son definidas por la Fundación de Software libre, y determinan si un programa o aplicación cumple con las características de un software libre (Free Software).

La Fundación de Software Libre (Free Software Foundation), define lo siguiente: “El software libre es una cuestión de libertad, no de precio”, esto permite que los usuarios

finales tengan la capacidad de ejecutar, estudiar, copiar, cambiar, distribuir y mejorar un determinado Software bajo el concepto de 4 libertades definidas a continuación [10]:

- **Libertad 0:** Libertad de ejecutar el software, para cualquier aplicación final.
- **Libertad 1:** Libertad de entender y estudiar el funcionamiento del programa, y modificarlo para adaptarlo de acuerdo a las necesidades del usuario.
- **Libertad 2:** Libertad para distribuir copias del software, con el objetivo de beneficiar a la comunidad.
- **Libertad 3:** Libertad de mejorar el programa y publicar la versión modificada y mejorada, con el objetivo de beneficiar a toda la comunidad, para esto el acceso al código fuente del programa es necesario.

En conclusión, un programa es considerado como software libre (Free Software) si los usuarios poseen las libertades mencionadas anteriormente. Además, el propósito del software libre radica en el principio de ejecución y uso del usuario, y no del programador original, por lo que es necesario y obligatorio el acceso al código fuente y la libre distribución de los manuales del software desarrollado. Finalmente, software libre no significa que no sea comercial. El usuario puede haber obtenido una copia de software libre, mediante un pago, o puede haber obtenido una copia sin costo. Sin embargo, sin tener en cuenta como se obtuvo una copia, el usuario tiene la libertad de copiar, modificar y redistribuir el software.

3.2 Iniciativa de Código Abierto (Open Source Initiative)

3.2.1 Descripción General

La **Iniciativa de Código Abierto** difiere un poco del movimiento de Software Libre, aunque comparten el objetivo principal de distribuir libremente el software. El movimiento de Código Abierto no se preocupa mucho si cierta persona obtiene ganancias de un determinado sistema, el movimiento está más preocupado de la distribución libre del mismo y su código fuente, lo que no implica que sea gratis. (Se debe garantizar que sin importar que el sistema tenga un costo inicial, su posterior distribución libre debe ser asegurada) [10].

3.2.2 Características definidas por la Iniciativa de Código Abierto

La Iniciativa de Código Abierto (OSI Open Source Initiative) se encarga de mantener la definición de código abierto en beneficio de la comunidad y transmitir los beneficios del código abierto [11].

Código Abierto (Open Source) no solo significa acceso al código fuente de un determinado programa. Para que un software o aplicación sea catalogada de código abierto debe cumplir con las siguientes características [11]:

- **Redistribución libre:** Permite a los usuarios vender o distribuir el software libremente.
- **Código fuente:** El Software debe incluir el código fuente, o que pueda ser obtenido libremente. El código fuente debe ser distribuido en forma adecuada, en la cual el programador pueda realizar cambios.
- **Trabajos derivados:** Las modificaciones y los trabajos derivados, pueden ser distribuidos.
- **Integridad del código fuente del autor:** Existe la posibilidad de que la licencia establezca que las modificaciones en el código fuente solo puedan ser distribuidas en forma de parches. Caso contrario, la licencia debe especificar el permiso de distribución del software con su código fuente modificado.
- **Sin Discriminación a personas o grupos:** Todas las personas pueden acceder al Software de Código Abierto.
- **Sin Discriminación en contra de los campos de aplicación:** No se puede restringir el uso en un determinado campo de aplicación. Por ejemplo, en el uso en un negocio particular.
- **Distribución de la licencia:** Todas los usuarios a quienes se redistribuyó el programa, tienen los derechos especificados sobre el mismo.
- **La licencia no debe ser específica a un determinado producto:** Los derechos establecidos de un determinado programa no deben depender en el hecho que el mismo este formando parte de una distribución particular de software.
- **La licencia no debe restringir otro software:** La licencia no puede obligar a otro software que se distribuya de manera conjunta con el Software de Código Abierto, a que sea necesariamente de código abierto.

- **La licencia debe ser tecnológicamente neutral:** No se requiere ninguna aceptación de la licencia.

3.3 Sistema Operativo Linux

3.3.1 Descripción General del sistema Operativo Linux

Linux es un sistema operativo inicialmente creado por Linus Torvalds en la Universidad de Helsinki, Finlandia. En 1994, Linus Torvalds presentó la versión 1.0 del Kernel de Linux. El Kernel, es el corazón de todos los sistemas Linux, y esta desarrollado y liberado bajo la licencia general pública (GPL por sus siglas en Inglés) GNU, y su código fuente está libremente disponible para cualquier usuario. Es el Kernel el que forma los cimientos sobre los cuales el sistema operativo Linux es desarrollado. Además de ser de distribución libre, la funcionalidad, adaptabilidad y eficiencia han hecho de Linux una alternativa para los sistemas operativos propietarios de Unix y Microsoft. Otros componentes como comandos administrativos y aplicaciones son añadidos al Kernel, a partir de software de código abierto y distribución libre, lo que hace de Linux un sistema operativo completo [11].

3.3.2 Kernel de Linux

El componente principal de un sistema operativo Unix y tipo Unix se lo conoce como Kernel. El Kernel se encarga de la comunicación entre los componentes de hardware básicos. Todas las interacciones entre el hardware y cualquiera de los programas debe ser negociado a través del Kernel. En otras palabras, el Kernel es el software que se ejecuta en el momento en el cual la computadora arranca y controla la comunicación con el hardware del equipo [11].

3.3.3 Características Principales de Linux

Linux presenta ciertas características diferentes a otros sistemas operativos, como por ejemplo [11]:

- **El usuario no necesita reiniciar el sistema de manera constante**. El tiempo de trabajo constante es esencial en sistemas Linux y otros sistemas Unix, los cuales

generalmente son usados como servidores, trabajando las 24 horas del día, 7 días a la semana.

- **Instalación de aplicaciones sin necesidad de reiniciar el sistema**. Después de la instalación original de una aplicación, se puede instalar o remover la misma sin necesidad de reiniciar el sistema en la mayoría de distribuciones Linux.
- **Inicia y termina servicios sin interrumpir otros**. El usuario puede iniciar o parar servicios individuales, sin afectar a otros servicios, esto evita que se reinicie el sistema en caso del colapso de una aplicación.
- **Aplicaciones descargables**. Si las aplicaciones requeridas por el usuario no vienen con la versión de Linux usada, el usuario puede descargar e instalar las aplicaciones con un solo comando a través de los repositorios de las distribuciones en internet.
- **Libertad**. La ventaja de Linux es que el usuario puede escoger entre la distribución que quiere utilizar de acuerdo a sus necesidades. Se puede añadir y remover el software que el usuario necesite.
- **Seguridad.** Linux es desarrollado para ser seguro por defecto, por lo que se necesita una cuenta administrativa (root), la cual es la encargada de todos los cambios que afectan al sistema.

3.3.4 Beneficios de Linux sobre otros sistemas operativos.

- **Es gratis:** Un gran porcentaje de las distribuciones de Linux se distribuyen sin costo. Otros sistemas operativos tienen un valor de distribución considerable [10].
- **Es descargable:** Con una conexión a internet rápida, el usuario puede obtener una distribución de Linux como Ubuntu en un corto período de tiempo. El usuario no necesita ordenar el sistema operativo, o ir a un local de venta de software y hardware para obtener una copia del mismo [10].
- **Es desarrollado por voluntarios alrededor del mundo:** Un gran porcentaje de las distribuciones de Linux son desarrolladas por voluntarios y compañías que apoyan a la causa. Este principio de desarrollo contribuye a generar un sistema operativo estable. Otros sistemas operativos son creaciones de compañías, y todo el trabajo se lo realiza dentro de la misma [10].
- **Es distribuido libremente:** El usuario puede realizar las copias que desee de distribuciones como Debian o Ubuntu. La única restricción es que el código fuente

del sistema debe ser distribuido. Otros sistemas operativos requieren de la compra de una licencia por cada instalación [10].

- **Código fuente disponible:** Linux permite a los usuarios, ajustes individuales, modificaciones y arreglos del sistema operativo, mediante la modificación del código fuente que siempre esta disponible. Como resultado, correcciones pueden ser realizados en poco tiempo. Por otro lado, sistemas operativos que no incluyen el código fuente, no permiten al usuario realizar modificaciones. Errores en este tipo de sistemas operativos demoran un tiempo considerable en ser reparados.
- **Es fiable:** Aunque no es una característica solo de Linux, es importante expresar que Linux es estable. Se conoce que existe servidores Linux funcionando ininterrumpidamente por meses, sin necesidad de ser reiniciados. Otros sistemas operativos necesitan ser reiniciados diariamente para que puedan ofrecer un rendimiento fiable.
- **Es flexible:** Existe una gran cantidad de aplicaciones para Linux, lo que permite que sea usado en sistemas tan complejos como estaciones de trabajo, realizando grandes cálculos [10].

3.3.5 Distribuciones de Linux

Algunos programadores han desarrollado distribuciones del Sistema Operativo especiales añadiéndoles sus propios programas [11]. A continuación se encuentra una lista de las Distribuciones más populares de Linux:

- Arch Linux.
- CentOS.
- Debian.
- Fedora.
- Gentoo.
- Mandriva.
- OpensuSE.
- Red Hat.
- Ubuntu.

3.4 Ubuntu

3.4.1 Descripción General

Ubuntu Linux, es una distribución de Linux basada en Debian GNU/Linux. Ubuntu es un sistema operativo construido por un conjunto de desarrolladores expertos a nivel mundial y que reciben el soporte de la empresa Canonical, la cual vende servicios relacionados con la distribución. Ubuntu se desarrolla para trabajar en computadores portátiles, computadores de escritorio y servidores. Ubuntu contiene todas las aplicaciones que el usuario necesita, desde procesadores de palabras hasta herramientas de programación como Python [12].

3.4.2 Características principales

- Ubuntu es distribuido bajo la licencia de Software libre GNU/GPL o bajo la licencia de código abierto.
- Ubuntu es desarrollado y distribuido sin costo alguno, en todas sus versiones.
- Linux libera sus versiones cada 6 meses, con un soporte de desarrollo de al menos 18 meses. La versión actual de Ubuntu es: Ubuntu 9.10 (Karmic Koala).
- Al igual que cualquier distribución basada en GNU/Linux, Ubuntu puede actualizar todas las aplicaciones de la máquina a través de los repositorios de Ubuntu.
- Ubuntu soporta oficialmente las siguientes arquitecturas de Hardware (procesadores): Intel x86 (IBM-compatible PC), AMD64(Hammer) y Power PC (Apple iBook y Powerbook, G4 y G5).
- Las versiones Ubuntu Desktop y Ubuntu Server pueden ser descargadas libremente desde la siguiente dirección electrónica: www.ubuntulinux.org.

3.5 Python

3.5.1 Descripción General

Python es un lenguaje de programación interpretado y orientado a objetos creado por Guido Van Rossum en 1991. Python es un lenguaje de programación poderoso, pero a su vez muy fácil de aprender y entender [13].

3.5.2 Características Principales

- La implementación de Python se desarrolla bajo la licencia de código abierto, lo que hace que Python pueda ser libremente usado y distribuido, inclusive para propósito comercial. La Licencia de Python es administrada por la Fundación de Software de Python (Python Software Foundation).
- Python es un lenguaje de programación modular. El usuario puede generar módulos reutilizables por otros programas [14].
- Es un lenguaje de programación multiparadigma (varios estilos):
    - Programación orientada a objetos.
    - Programación estructurada.
    - Programación funcional.
- Puede usar un intérprete (Modo interactivo, como Matlab)
- Python soporta herencia múltiple y polimorfismo [14].
- Python puede ser ejecutado en Windows, Linux/Unix, Mac OS X, inclusive existe versiones de Python que corren sobre máquinas virtuales .NET y Java [14].
- Python posee un extenso número de librerías estándar y módulos especiales desarrollados para diversas tareas [14].
- Python se enfoca en desarrollar códigos entendibles, coherentes, y de calidad. El código de Python es desarrollado para ser entendible, y como consecuencia, reusable y fácil de mantener, de manera más profunda comparado con otros lenguajes de programación.

3.5.3 Aplicaciones

Python es usado en muchas aplicaciones, entre las cuales podemos nombrar:

- Desarrollo de Internet y Web.
- Acceso a Base de Datos: Existen varias Interfaces de Base de Datos para Python entre las cuales tenemos a la librería MySQL para Python, que permite gestionar una base de datos relacional MySQL [15].
- Desarrollo de interfaces gráficas de usuario (GUI): Python posee varias librerías para el desarrollo de interfaces gráficas de usuario. Tkinter es la librería que viene incluida en la mayoría de distribuciones de Python. Otras aplicaciones muy usadas para el desarrollo de GUI’s son wxPython, PyQT, entre otras.

- Cálculos científicos y numéricos: Existen diversas librerías para desarrollo científico y numérico para Python. Entre las librerías más conocidas se tiene: NumPy, SciPY, PyGTS, entre otras [15].
- Programación de redes.
- Desarrollo de software.
- Juegos, etc.

3.5.4 Instalación

Linux

La mayoría de distribuciones como Ubuntu tienen Python instalado.

Windows

Existen varias versiones de instaladores de Python para Windows. Los archivos ejecutables (.exe) de las distintas versiones para Windows se pueden encontrar en la siguiente dirección electrónica: http://www.python.org/download/releases.

3.6 MySQL

3.6.1 Descripción General

El Software MySQL proporciona un servidor de base de datos SQL (Structured Query Language). MySQL proporciona a sus usuarios una solución rápida, robusta, multi-hilo y multiusuario para generar y gestionar una base de datos. Una base de datos es un conjunto estructurado de datos, puede ser muy simple, como almacenar las características y tipos de equipos que se encuentran dentro de una oficina, hasta una aplicación extensa como la base de datos de una red industrial, distribuida geográficamente. Para adquirir nuevos datos, procesarlos, y acceder a los mismos se necesita de un software de gestión de base de datos como MySQL. Una base de datos funciona de manera autónoma, o como parte de una aplicación extensa como es el caso de los sistemas Scada [16].

3.6.2 Lenguaje SQL (Structured Query Language)

Es el lenguaje más común para el manejo y acceso de una base de datos, y esta definido por el estándar ANSI/ISO SQL. De acuerdo con ANSI (Americal national Standars Institute), es el lenguaje estándar para administración de base de datos relacionales. Las declaraciones SQL son usadas para realizar tareas, como actualizar una base de datos, o adquirir datos provenientes de la misma [17].

3.6.3 Características

- MySQL es un software de distribución libre, bajo la licencia GNU/GPL Licencia Pública General; sin embargo, se puede adquirir una licencia estándar de MySQLAB (Desarrolladores de MySQL). Al ser una distribución Libre MySQL puede ser usado y modificado por cualquier persona, su distribución es a través del internet y de manera gratuita [18].
- MySQL es un sistema de gestión de base de datos relacionales. Como resultado, MySQL no almacena todos sus datos en un mismo lugar, lo que permite tener un aumento de la velocidad y escalabilidad en la generación de un servidor de base de datos [18].
- MySQL es un software fácil de usar, confiable y rápido, ya que originalmente fue creado para manejar grandes bases de datos mucho más rápido que soluciones existentes.
- Es un sistema de base de datos multi-hilo y multiusuario.
- Lenguajes de programación: Existen varias interfaces de programación de aplicaciones (APIs) que permiten, a aplicaciones escritas en diversos lenguajes de programación acceder a las bases de datos MySQL, entre los lenguajes de programación tenemos: C, C++, Python, Delphi, Java, Perl, Lisp, Ruby, entre otros [18].
- Funciona en diferentes plataformas: Linux, Mac OS X, Windows, entre otras.
- MySQL está escrito en C y en C++ [16].
- MySQL puede usarse en una red con múltiples computadoras [16].
- MySQL se adapta a la demanda de sistemas de alta velocidad. Por ejemplo, MySQL es ideal para sistemas de varias lecturas, a velocidades como las del internet, en aplicaciones para la web.

- MySQL permite añadir al sistema general otro sistema de almacenamiento. Esto es útil si se desea añadir una interfaz SQL para una base de datos propia [18].
- MySQL posee un sistema de reserva de memoria basado en hilos.
- MySQL hace uso de tablas temporales conocidas como “hash”, que permite la optimización en el manejo de los datos [18].
- MySQL tiene baja sobrecarga, lo que garantiza su operación estable incluso cuando la memoria RAM disponible es baja.
- Las funciones SQL están implementadas por MySQL usando una librería altamente optimizada, para una maximización en la velocidad de ejecución [18].
- El servidor MySQL está disponible como un programa separado para usar en un entorno de red cliente/servidor.
- MySQL permite diversos tipos de datos en las columnas de las tablas de almacenamiento. Se permiten enteros con/sin signo de 1, 2, 3, 4, y 8 bytes, datos de los tipos: FLOAT, DOUBLE, CHAR, VARCHAR, TEXT, DATE, TIME, YEAR, ENUM. Registros de longitud fija y variable [18].
- MySQL permite mezclar tablas de distintas base de datos en la misma consulta [16].
- MySQL es un sistema de privilegios y contraseñas muy flexible y seguro. Permite verificación basada en el host. Las contraseñas son seguras debido a que todo el tráfico de contraseñas está encriptado cuando se conecta con un servidor.
- MySQL es sistema escalable. Soporta grandes bases de datos. Las tablas generadas en MySQL pueden crecer hasta un tamaño considerable, solo limitado por el propio sistema operativo [18].
- MySQL permite generar tablas de hasta 64 campos o índices.
- MySQL brinda alta conectividad. Los clientes pueden conectarse con el servidor MySQL usando Sockets TCP/IP en cualquier plataforma [18].

3.6.4 Instalación

En la actualidad la versión recomendada de MySQL es la 5.1.

Linux

Descargar el módulo (tar.gz) de la siguiente dirección electrónica: http://dev.mysql.com/downloads/mysql/5.1.html#linux, extraer el paquete y ejecutar los siguientes comandos en el terminal de Linux.

$ groundpadd mysql

$ useradd –q mysql mysql

$ cd mysql

$chown –R mysql

$chgrp –R mysql

$scripts/mysql install_db –user=mysql

$chown –R root

$chown –R mysql data

$bin/mysqld_safe –user=mysql $

Ubuntu

MySQL se puede instalar en Debian/Ubuntu desde los repositorios (servidores que contienen paquetes de programas) de Ubuntu, ejecutando el siguiente comando:

$ sudo apt-get install mysql-server

Windows

Descargar el instalador de Windows (.exe) desde la siguiente dirección electrónica: http://dev.mysql.com/downloads/mysql/5.1.html#win32. Ejecutar el archivo .exe.

3.7 pySerial

3.7.1 Descripción General

pySerial es un módulo de Pyton que administra el acceso al puerto serial.

3.7.2 Características Generales

- **Estado de desarrollo:** Módulo Estable.
- **Licencia:** aprobada por la **Iniciativa de Código Abierto** (OSI Open Source Initiative) y por la **Fundación Python Software.**
- **Lenguaje:** Inglés.
- **Sistemas Operativos:** Windows, Linux.
- **Lenguaje de Programación:** Python.
- Permite acceder a las propiedades del puerto serial a través de Python.
- Soporta diferentes tamaños de bits de datos, bits de parada, paridad y control de flujo.
- El usuario puede emplear o no un tiempo de espera (timeout), para la recepción de datos.
- Para su uso se requiere de Python 2.3 o una versión más actualizada. Si se lo utiliza en Windows se debe comprobar que python contenga las extensiones para Windows (ctypes), que vienen por defecto como una librería estándar desde Python 2.5 [13].

3.7.3 Instalación:

Linux

Descargar el módulo (tar.gz) desde la siguiente dirección electrónica: http://pypi.python.org/pypi/pyserial, y extraer el paquete. Ingresar usando el terminal al directorio pyserial-x.y ($ cd pyserial-x-y), donde x.y es la versión del módulo. Finalmente, ejecutar el siguiente comando desde el terminal:

$ python setup.py install

Ubuntu

pySerial se puede instalar en Debian/Ubuntu desde los repositorios (servidores que contienen paquetes de módulos instalables) de Ubuntu, ejecutando el siguiente comando:

$ sudo apt-get install python-serial

Windows

Descargar el instalador de Windows (.exe) desde la siguiente dirección electrónica: http://pypi.python.org/pypi/pyserial. Ejecutar el archivo .exe.

3.8 MySQLdb

3.8.1 Descripción General

MySQLdb es una interfaz compatible con el gestor de base de datos MySQL. Es la interfaz de programación de aplicaciones (API) de base de datos, que provee Python para el manejo de un servidor de base de datos MySQL [19].

3.8.2 Características Generales

- **Estado de desarrollo:** Módulo estable.
- **Licencia:** GNU/GPL Licencia Pública General y aprobada por la Iniciativa de Código Abierto (OSI).
- **Lenguaje:** Inglés.
- **Sistemas Operativos:** Independiente del sistema operativo. Puede ser ejecutado en Linux, Windows o Mac OS X.
- **Lenguaje de Programación:** Python y C.
- MySQLdb permite conectarse con la base de datos, a través de objetos de conexión (Mediante un constructor y parámetros: connect(parameters…)) [19].
- El objeto generado para la conexión con la base de datos tiene métodos que permiten realizar diversas acciones. Por ejemplo, el método .close(), termina la conexión. Otro método importante es .cursor(), que genera otro objeto, el cual permite obtener la descripción de una tabla dentro de la base de datos, el número de filas, la creación de una tabla, la ejecución de operaciones de búsqueda, la obtención de datos, entre otras acciones [19].
- MySQLdb genera avisos y errores relacionados a las diversas operaciones realizadas mediante la interfaz sobre la base de datos [19].

3.8.3 Instalación

Linux

Descargar el módulo (tar.gz) desde la siguiente dirección electrónica: http://souceforge.net/projects/mysql-python, extraer el paquete. Ingresar usando el terminal al directorio MySQL-python-x.y, y ejecutar el siguiente comando:

$ python setup.py install

Ubuntu

MySQLdb se puede instalar en Debian/Ubuntu desde los repositorios (servidores que contienen paquetes de módulos instalables) de Ubuntu, corriendo el siguiente comando desde el terminal:

$ sudo apt-get install python-mysqldb

Windows

Descargar el instalador de Windows (.exe) desde la siguiente dirección electrónica: http://souceforge.net/projects/mysql-python. Ejecutar el archivo .exe.

3.9 Matplotlib

3.9.1 Descripción General

Matplotlib es una librería que permite realizar gráficas en dos dimensiones mediante Python. Sin embargo, los comandos de Matplotlib fueron desarrollados emulando los comandos de generación de gráficas de Matlab; pero, los desarrolladores le agregaron la idea de programación orientada a objetos. Para un mejor uso de los datos a graficar, Matplotlib, hace uso de la librería Numpy, la misma que constituye el paquete fundamental utilizado para realizar cálculos científicos en Python [20].

El código de Matplotlib se divide básicamente en las siguientes interfaces:

- Interfaz pylab: Conjunto de funciones provistas por matplotlib.pylab el cual permite al usuario crear gráficas con comandos similares a Matlab.

- Interfaz de aplicación de Matplotlib: Es un conjunto de clases las cuales permiten crear y manejar las figuras, gráficas, texto, líneas, etc.

3.9.2 Características Generales

- **Licencia:** Licencia administrada y aprobada por la Fundación Python Software.
- **Sistemas Operativos:** Puede ser ejecutado en plataformas Windows, Linux, Unix, Mac OSX.
- **Lenguaje de Programación:** Python.
- Para su instalación, Matplotlib requiere de los siguientes paquetes: Python 2.4 o una versión más actualizada, pero no Python 3, y numpy 1.1 o una versión más actualizada [20].

3.9.3 Instalación

Linux

Descargar el módulo (tar.gz) desde la siguiente dirección electrónica: http://matplotlib.sourceforge.net, extraer el paquete. Ingresar usando el terminal al directorio Matplotlib y ejecutar el siguiente comando:

$ python setup.py install

Ubuntu

Matplotlib se puede instalar en Debian/Ubuntu desde los repositorios (servidores que contienen paquetes de módulos instalables) de Ubuntu, ejecutando el siguiente comando en el terminal:

$ sudo apt-get install python-matplotlib

Windows

Descargar el instalador de Windows (.exe) desde la siguiente dirección electrónica: http://matplotlib.sourceforge.net. Ejecutar el archivo .exe.

3.10 wxPython

3.10.1 Descripción General

wxPython es un conjunto de herramientas que permite el desarrollo de programas para la generación de interfaces gráficas de usuario reales (GUIs) muy poderosa, pero a su vez muy simple de usar. wxPython permite al usuario crear y manipular elementos comunes de las interfaces tal como botones y menús, pero también permite crear y manipular elementos menos comunes, como tablas, árboles y editores de texto. Tal como Python, wxPython es un proyecto de código abierto y multiplataforma [21].

3.10.2 Características Generales

- **Licencia:** GNU/GPL Licencia Pública General (GPL) y aprobada por la Iniciativa de Código Abierto (OSI).
- **Sistemas Operativos:** Puede ser ejecutado en las plataformas: Windows, Linux, Unix y Mac OSX.
- **Lenguaje de Programación:** Python.
- wxPython tiene una amplia librería de herramientas (botones, cuadros de texto, listas, etc) [21].
- wxPython permite a sus usuarios desarrollar interfaces gráficas de usuario (GUI's) flexibles [21].
- Existen muchas interfaces gráficas para el desarrollo de aplicaciones de wxPython, (por ejemplo: Boa Constructor, wxGlade, wxDesigner, entre otros), las cuales hacen más fácil la construcción de aplicaciones [21].
- wxPython permite a sus usuarios desarrollar diversos tipos de interfaces gráficas, desde las más simples, hasta interfaces gráficas complejas basadas en eventos (Movimiento del mouse, click en un determinado botón, timers, etc.) [21].

3.10.3 Instalación

Linux

Descargar el módulo wxGTK (tar.gz) desde la siguiente dirección electrónica: http://www.wxwidgets.org, extraer el paquete. Ingresar usando el terminal al directorio wxGTK-x.y, y ejecutar el siguiente comando desde el terminal:

$ ./configure –with-gtk

Luego de instalar wxGTK, el usuario debe instalar wxPython. Primero el usuario debe descargar el paquete wxPython (tar.gz) desde la siguiente dirección electrónica: http:/www.wxpython.org/ download.php, extraer el paquete. Ingresar usando el terminal al directorio wxPython-x.y, y ejecutar el siguiente comando:

$ python setup.py install

Ubuntu

wxPython puede ser instalado en Debian/Ubuntu desde los repositorios (servidores que contienen paquetes de módulos instalables) de Ubuntu, ejecutando el siguiente comando desde el terminal:

$ sudo apt-get install python-wxgtk2.6

Windows

Descargar el instalador de Windows (.exe) desde la siguiente dirección electrónica: http:/www.wxpython.org/download.php. Ejecutar el archivo .exe.

3.11 wxGlade

3.11.1 Descripción General

wxGlade es un diseñador de interfaces gráficas de usuario, el mismo que está escrito en Python y posee una interfaz gráfica de usuario propia creada mediante el uso de las herramientas que provee wxPython. La interfaz gráfica de usuario de wxGlade permite a los usuarios generar interfaces wxpython y su correspondiente código Python [20].

3.11.2 Características Generales

- **Licencia:** Aprobada por la Iniciativa de Código Abierto (OSI).
- **Sistemas Operativos:** Es independiente del sistema operativo. Puede ser ejecutado en Windows, Linux y Mac OS X.
- **Lenguaje de Programación:** Python.

- wxGlade utiliza una interfaz gráfica de usuario, la cual permite al usuario generar sus propias interfaces de usuario y el código Python/wxPython correspondiente a la interfaz generada.

3.11.3 Instalación

Linux

Descargar el módulo (tar.gz) desde la siguiente dirección electrónica: http://sourceforge.net/projects/wxglade/files/, extraer el paquete. Ingresar usando el terminal al directorio wxglade y ejecutar el siguiente comando:

$ python wxglade.py

Ubuntu

wxGlade se puede instalar en Debian/Ubuntu desde los repositorios (servidores que contienen paquetes de módulos instalables) de Ubuntu, ejecutando el siguiente comando:

$ sudo apt-get install python-wxglade

Windows

Descargar el instalador de Windows (.exe) desde la siguiente página web: http://sourceforge.net/projects/wxglade/files/. Ejecutar el archivo .exe.

3.12 Diseño Estructural de la Aplicación CM a partir del Software Libre y Software de Código Abierto Descrito.

3.12.1 Descripción General

La aplicación de desarrollo de sistemas HMI/Scada descrita en este trabajo (CM), fue diseñada tomando en cuenta los siguientes parámetros:

- **Funcionalidad de desarrollo Scada:** CM debe proporcionar todas las herramientas necesarias en el diseño de sistemas Scada.
- **Desarrollo a partir de Software de Código Abierto y Software Libre:** CM es una aplicación cuyo principio es desarrollar un sistema de generación de sistemas

Scada usando el software libre y software de código abierto descrito en este capítulo.

En la figura 3.1 se presenta un diseño funcional del sistema y como fue construido, usando recursos de código abierto y distribución libre.

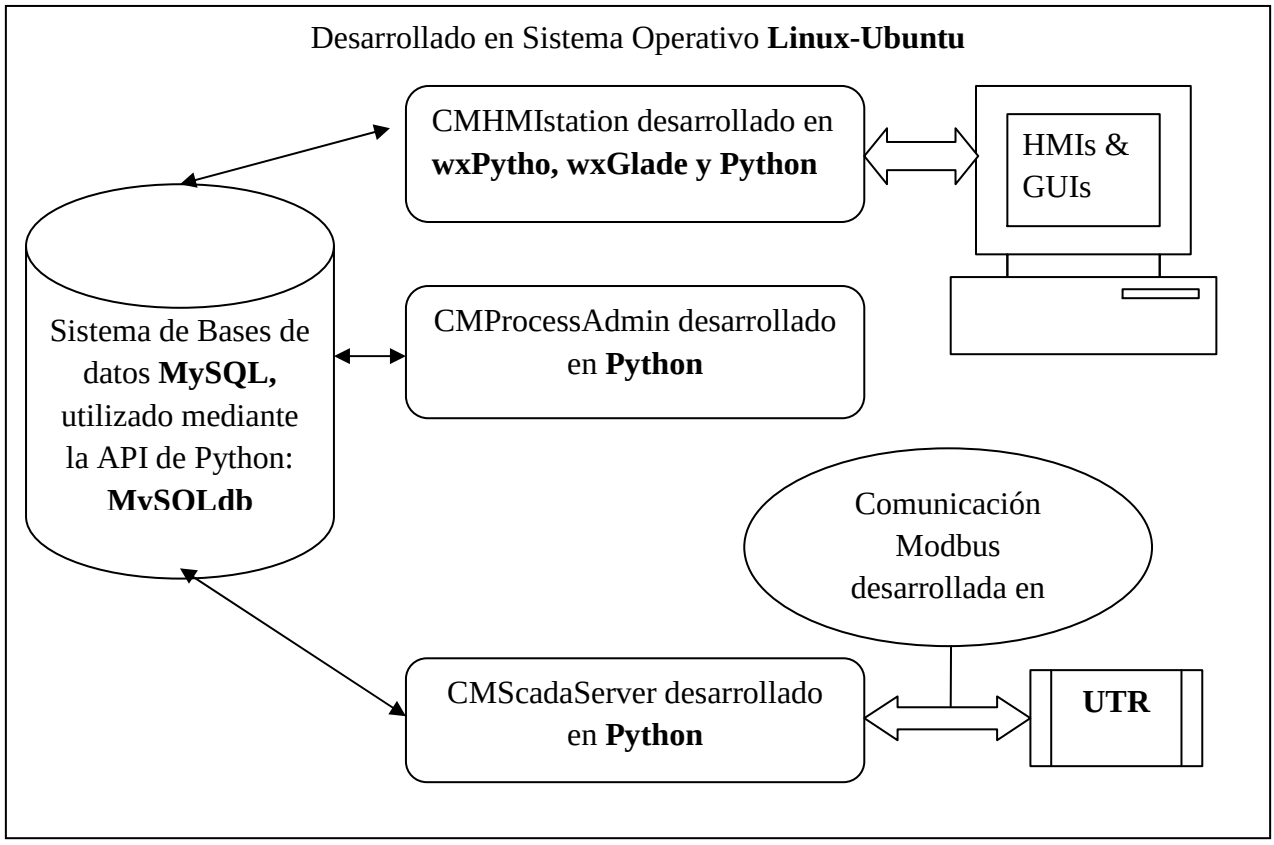

Figura 3.1: Diagrama estructural de la construcción de CM mediante distintas aplicaciones libres y de código abierto

4

Control de Procesos Industriales mediante Aplicaciones Scada

Bucles de Control desarrollados sobre una red de comunicación industrial son cada vez más comunes en la actualidad, ya que los dispositivos de hardware utilizados como nodos en la red, son cada vez más económicos. Como resultado, los desarrolladores de sistemas de control de procesos utilizan este hardware para desarrollar y ejecutar algoritmos de control (software) [22]. En la actualidad, muchos sistemas y aplicaciones Scada, tienen la capacidad de ejecutar algoritmos de control sobre variables del proceso. Sin embargo, para que el proceso industrial sea controlado de manera fiable y eficiente, la aplicación debe ser considerada como:

- Un Sistema de Control en Tiempo Real.
- Un Sistema de Control Multivariable.

4.1 Sistemas de Control en Tiempo Real

4.1.1 Definición de Sistemas de Control en Tiempo Real

Un Sistema de Control en Tiempo Real es un sistema que debe satisfacer un específico tiempo de respuesta limitado o se corre el riesgo de consecuencias controlables o severas, incluyendo la falla total del sistema [23].

4.1.2 Características de los Sistemas de Control en Tiempo Real

La característica principal de los Sistemas de Control en Tiempo Real es que sus acciones deben producirse dentro de intervalos de tiempo determinados por la dinámica del sistema físico (proceso) que supervisan o controlan[23].

Un Sistema de Control en Tiempo Real, es un sistema informático que tiene las siguientes características:

- Interacciona repetidamente con su entorno físico.

- Responde a las señales de entrada provenientes del entorno físico dentro de un plazo de tiempo determinado.
- Ejecuta de algoritmos de control en tiempo real.
    - De forma periódica: Se ejecuta el algoritmo de control cada intervalo de tiempo determinado.
    - Basada en un evento: Se ejecuta el algoritmo de control solo cuando un evento externo lo ha activado.

4.1.3 Problema de Control en Sistemas de Tiempo Real

La introducción de los computadores digitales en los bucles de control ha permitido desarrollar sistemas de control más flexibles, incluyendo funciones de alto nivel y algoritmos de control avanzados. Por otro lado, la mayoría de sistemas de control complejos actuales no podrían ser implementados sin la aplicación de hardware digital. Sin embargo, la secuencia simple de un bucle de control (sensar-controlar-actuar) de un sistema de control realimentado clásico se vuelve más complejo a la vez [22]. En la actualidad, la secuencia de control puede ser implementada de la siguiente manera: sensar- adquisición de datos- algoritmo de control- actuación- actualización de base de datos, lo que describe la arquitectura de un sistema Scada. Como resultado, el sistema de control contiene no solo componentes conectados sino también algoritmos de control, los cuales deben ser programados. En consecuencia, software debe ser incorporado en el bucle de control. Esto lleva a que se tengan en cuenta los siguientes aspectos en el desarrollo de Sistemas de Control en Tiempo Real:

- El uso de software y el desarrollo de algoritmos de control es propenso a errores. Entonces, el desarrollador del sistema debe verificar que el programa o algoritmo de control realice la acción esperada.
- Errores provenientes del hardware o dispositivos de campo a la hora de conversiones por ejemplo en el caso de señales analógicas.
- El tiempo computacional de los algoritmos de control puede cambiar entre cada uno de los intervalos de acceso a los datos. Esta variación de retardo es conocida como “control jitter”. La IEEE (Instituto de ingenieros eléctricos electrónicos) define el término jitter como el tiempo relacionado inesperado, o variación no deseada en la duración de cualquier intervalo específico relacionado [22].

- En la realidad se observa que existen retardos debido al tiempo consumido en el desarrollo del algoritmo de control o en la retroalimentación de la señal. A este retardo generalmente se lo conoce como retardo de control o retroalimentación.
- Retardo ocurrido por la transmisión de los datos a través de una red de comunicación industrial.

4.1.4 Consideraciones de diseño de Sistemas de Control en Tiempo Real

Para que el funcionamiento del sistema sea correcto no basta con que las acciones realizadas sobre el sistema sean las adecuadas, sino que dichas acciones deben ser realizadas dentro de un intervalo de tiempo determinado. Una acción retardada puede dar origen a una situación peligrosa en términos de un proceso [23].

El diseño e implementación de un Sistema de Control en Tiempo Real requiere atención en diversas áreas, en las que podemos incluir las siguientes:

- La selección cuidadosa del hardware y el software adecuado, para reducir los retardos originados en hardware y software del sistema.
- Especificación del diseño del sistema en tiempo real y la correcta representación del comportamiento temporal del sistema.
- Entender las características del lenguaje de programación y las implicaciones en tiempo real, resultado de la traducción del lenguaje de programación a código de máquina.
- Maximizar la tolerancia a fallas del sistema y la fiabilidad del mismo a través de un diseño cuidadoso.
- Es conveniente analizar el comportamiento temporal del sistema antes de probarlo.
- Finalmente, medir y predecir la respuesta en tiempo y reducirla. El computador debe reaccionar a tiempo en los cambios en el sistema que controla.

4.2 Sistemas de Control Multivariable

4.2.1 Definición de Sistemas de Control Multivariable

Los procesos industriales son sistemas cuya complejidad puede variar. Podemos encontrar sistemas como el control del nivel de un tanque de agua hasta sistemas

complejos como el control de Generación de Energía Eléctrica en una Central Térmica. Por lo tanto, en muchos procesos Industriales el objetivo es controlar varias variables a partir de varias señales de entrada.

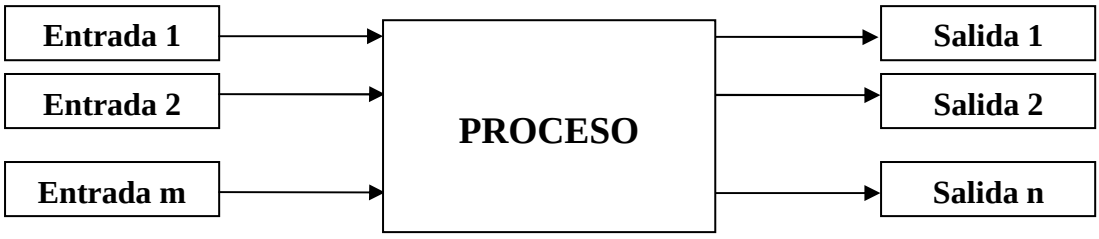

Figura 4.1: Sistema de Múltiples Entradas Múltiples Salidas (MIMO system)

Este tipo de procesos se los conoce como sistemas de múltiples entradas múltiples salidas (MIMO: multiple inputs multiple outputs). La figura 4.1 muestra la estructura general de un sistema de múltiples entradas, múltiples salidas. El objetivo de un sistema de control multivariable es controlar 2 o más variables del proceso, para que su valor de salida sea establecido dentro de parámetros deseados por el desarrollador del sistema de control [24].

4.2.2 Características de los sistemas de control multivariable

- Si un número independiente de variables van a ser controladas, entonces se necesita que al menos el mismo número de variables independientes sean manipuladas [24].
- Si existe más variables manipuladas que controladas, entonces existe más opciones de controlar el proceso y es esperado alcanzar mejores resultados. [24]
- En general, en un sistema de control de un determinado procesos, mientras más sensores, mejor es el comportamiento del sistema.

4.2.3 Problema de Control en Sistemas de Múltiples Entradas Múltiples Salidas

En general, una o más variables de entrada van a ser relacionadas con una variable controlada o grupo de variables controladas. A esta clasificación en grupos se lo conoce como agrupamiento. Para generar esta clasificación el desarrollador debe tomar en cuenta dos aspectos claves a la hora de diseñar el sistema de control:

- **Interacción:** En Sistemas Multivariables, la interacción entre las variables del proceso es el efecto más importante en el desarrollo y sincronización de los sistemas de control. La interacción dentro de un proceso se presenta cuando una variable de entrada afecta a más de una variable de salida [24].
- **Dominio:** Si el efecto de la variable manipulada correspondiente es mayor que el de otras, se puede decir que la variable presenta dominio sobre la variable controlada, lo que afecta al diseño del sistema de control [24].

Otro problema referente al control multivariable de procesos es las diferentes ganancias que presenta el mismo de acuerdo a la combinación de las entradas. Si las ganancias encontradas son significativamente diferentes, el proceso será muy difícil de controlar [25].

4.2.4 Consideraciones de Diseño de Sistemas de Control Multivariable

Generalmente, un sistema de control debe ser diseñado para trabajar con un proceso ya existente [26]. El diseño del sistema de control puede ser desarrollado de manera local, o mediante un sistema de supervisión y control (Scada). A continuación, se presentan diversas características a tomar en cuenta a la hora de diseñar el sistema de control para varias variables [24]:

- Definir cuales son las variables del proceso a ser controladas y las variables a ser manipuladas.
- Definir el comportamiento requerido para las variables a ser controladas.
- Desarrollar el modelo del proceso. Distribución de variables manipuladas y controladas.
- Desarrollar una estrategia de control adecuada.
- Desarrollar el algoritmo de control (Esta característica es el componente principal en el desarrollo de controladores mediante Software, en sistemas de control Scada).
- Validar la estrategia de control diseñada, mediante la simulación o experimentación. Definir los parámetros de los controladores.
- Implementar la estrategia de control en el proceso.
- Evaluar la respuesta del sistema.

5

CM – HMI/Scada

5.1 Antecedentes

Los sistemas Scada actuales son sistemas propietarios, cuya distribución es restringida, y la licencia de operación muy costosa. Como resultado, procesos industriales de tamaño mediano, pequeño o simplemente de carácter educacional no pueden ser implementados mediante un sistema Scada debido al costo del software de desarrollo.

En la actualidad la mayoría de procesos industriales están desarrollados a partir de software propietario, cuyas desventajas son:

- Costo de las licencias de ejecución.
- Dependencia tecnológica (No se puede realizar determinadas acciones debido a que tiene un costo el módulo especificado).
- Son sistemas cerrados (Aplicaciones específicas no incorporadas en el paquete original, vienen en for4ma de módulos externos, y tienen un costo adicional).

Por esta razón, muchas personas alrededor del mundo generan aplicaciones de distribución libre con el objetivo de romper la barrera del software propietario. Siguiendo este lineamiento se desarrolló **CM**, una aplicación de desarrollo de Sistemas HMI/Scada, que pretende cumplir con todas las características necesarias para construir un sistema Scada en procesos pequeños, medianos o en aplicaciones con fines educacionales.

5.2 Objetivo de CM

CM es un proyecto cuyo objetivo principal es crear una plataforma de desarrollo de sistemas de control, supervisión y adquisición de datos (Scada), usando recursos de código abierto y distribución libre (Open Source and Free Software), computadores personales (PC), y mediante el uso de protocolos de comunicación industrial (Modbus). La importancia del proyecto se basa principalmente en el desarrollo de una plataforma integral

de control en tiempo real, lo que permitirá usar el software en aplicaciones de control y automatización tanto a nivel de laboratorio, como a escala industrial pequeña y mediana.

El software para sistemas de control tiene requerimientos específicos que permitirán un desempeño correcto y robusto a la hora de realizar el control de sistemas en tiempo real. A continuación se definen los alcances del sistema, como objetivos particulares.

- Desarrollar un Servidor Scada que se encargue de la adquisición de datos.
- Desarrollar un módulo que permita ejecutar algoritmos de control en tiempo real.
- Desarrollar un sistema de comunicación bidireccional, tanto de entrada como de salida, es decir permite recibir señales y entregar acciones de control.
- Desarrollar una aplicación del protocolo de comunicación industrial Modbus.
- Procesamiento de señales de entrada (Maestro Modbus).
- Crear aplicaciones orientadas al usuario, para el desarrollo de HMIs, y supervisión de sistemas de control en tiempo real.
- Generar un sistema interno de almacenamiento de datos.
- Crear una página web que contenga los alcances del proyecto, y el código fuente de la plataforma para que pueda ser usado y modificado posteriormente.

5.3 Evolución Temporal de CM

La idea de generar una aplicación que permita desarrollar sistemas HMI/Scada, comenzó alrededor del mes de Enero y se cristalizó a mediados de Febrero. Desde ese instante, CM ha sufrido una evolución constante que se describe en los siguientes períodos.

Febrero – Junio

- Desarrollo de una estrategia de diseño del sistema.
- Elegir la plataforma de desarrollo y los lenguajes de programación a utilizar. (Linux, Python, MySQL, wxPython).
- Diseño de aplicaciones simples de prueba.

Julio-Septiembre

- Desarrollo de la interfaz de comunicación serial usando Python.
- Inicio del desarrollo de los HMI's y los GUI's de configuración.
- Desarrollo del sistema mediante el uso de base de datos MySQL y la interfaz de aplicación de Python.

Octubre-Diciembre

- Generación del diagrama estructural final de la aplicación. Se determinó que la estructura final del sistema sería dividida en tres grandes módulos: CMScadaServer, CMHMIStation y CMProcessAdmin.
- Desarrollo del módulo de controladores.
- Desarrollo del protocolo de comunicación Modbus RTU usando Python.
- Desarrollo del módulo de alarmas.
- Continuación del desarrollo de los HMI's y los GUI's de configuración.

En la actualidad:

- CM es liberada como una aplicación en prueba. Generalmente las aplicaciones de distribución libre desarrolladas tardan de 6 a 18 meses en ser consideradas versiones estables, ya que requieren una constante verificación de errores. Si se desarrolla un error determinado se procede a corregir el error y liberar la versión corregida.
- Pruebas en el sistema usando varias plataformas (Linux, Windows).
- Pruebas del sistema basado en un solo computador, y mediante el uso de varios computadores (2 computadores conectados en red).

En el futuro:

- Generar una página web del proyecto. CM cumple con las normativas de un programa de distribución libre, por lo que se debe conseguir que se distribuya el mismo bajo licencia GNU/GPL.
- Liberar el proyecto en el internet, para que pueda ser usado y modificado por la comunidad. Esto permitirá determinar si la aplicación tiene errores durante su

empleo en diferentes tareas. Finalmente, luego de alrededor de 18 meses de mantenimiento del sistema se lo podrá considerar como estable.

- Con la ayuda de ingenieros en sistemas desarrollar un módulo que permita transformar un archivo .dwg (autocad, o archivos similares de distribuciones libres como FreeCad) en despliegues dinámicos para los clientes HMI de CM.
- Generar un módulo CMHMIstation dínamico y con muchas herramientas de desarrollo.
- Los módulos de CM deben ser probados de manera independiente y sobre distintas plataformas.

5.4 Detalles Constructivos de CM

5.4.1 Lenguaje de Programación: Python

Después de analizar diversas alternativas de lenguajes de programación de distribución libre y código abierto, se determinó que Python era un lenguaje con las características necesarias para ser la base de la aplicación. Al ser CM una aplicación escrita **en Python,** se garantiza un código fuente modular, robusto y escalable.

5.4.2 Plataforma/Sistema Operativo: Ubuntu Linux (Windows)

La plataforma escogida para el desarrollo del sistema fue Linux, y específicamente la distribución de Linux Ubuntu. Linux se escogió como plataforma base debido a las siguientes razones:

- **Linux** es un sistema operativo de distribución libre y código abierto, es decir sigue la idea principal del desarrollo de la aplicación CM: Usar Recursos de libre distribución.
- La mayoría de distribuciones de Linux tienen instalado Python, como lenguaje de programación base del sistema.
- **Linux** es un sistema operativo que ha demostrado un alto rendimiento en sistemas de tiempo real, y en sistemas de base de datos (Los servidores Linux tienen un desempeño muy alto y efectivo, y pueden trabajar de manera constante sin necesidad de ser reiniciados).

La versatilidad de Python hace posible que CM pueda ser ejecutado en la plataforma Windows. Sin embargo, Windows no cumple con la idea general del proyecto.

5.4.3 Sistema de gestión de base de datos: MySQL y python-MySQLdb

La gestión de base de datos se la realiza usando el software de distribución libre MySQL. Para poder administrar la base de datos, Python posee una interfaz de aplicación python-MySQLdb. Mediante MySQLdb el usuario puede realizar consultas o almacenar datos usando código Python.

5.4.4 Generación de interfaces Hombre-Máquina: wxPython

Para la generación de despliegues (HMI), CM usa las herramientas wxPython y wxGlade. wxPython es una librería de desarrollo de GUI's para python muy poderosa, en cambio wxGlade es una aplicación que permite desarrollar un GUI a través de una interfaz propia de wxGlade. Finalmente, wxGlade permite originar el código equivalente wxpython-python del GUI diseñado.

5.4.5 Generación de Gráficas de Tendencia y Gráficos de Históricos: Matplotlib

Para la generación de gráficos de tendencia se usa la librería Matplotlib distribuida en conjunto con Python.

5.4.6 Generación de Algoritmos de control: Python-Numpy

Para el desarrollo de algoritmos de control, CM hace uso de Python y si es necesario, se puede usar Numpy, que es la librería de cálculo científico de Python.

5.4.7 Sistema de Comunicación: Python-pyserial

Para el desarrollo de la interfaz de comunicación serial, se usó pySerial. pySerial es la librería que administra las acciones del puerto serial. Por otro lado, para el desarrollo del protocolo Modbus se utilizó exclusivamente Python.

5.5 CM Características del Sistema

CM es un software de desarrollo de sistemas HMI/Scada, principalmente basado en software de distribución libre y código abierto, por lo que no tiene ningún costo de licencias, es fácil de utilizar, y tiene la ventaja de que su código puede ser modificado para adaptarse a las necesidades del usuario.

5.5.1 Características principales de la aplicación CM

- CM se caracteriza por generar una plataforma HMI/Scada con el objetivo de controlar y supervisar un proceso. CM es un sistema modular en el cual se distingue el Servidor Scada el cual se encarga de la adquisición de datos, el Servidor de Base de Datos y el Generador de Despliegues (HMI).
- **CM es de distribución Libre:** CM no tiene costos de licencia y su código fuente puede ser modificado para adaptarse a las características de un determinado proyecto.

- **Adquisición de datos en tiempo real:** CM adquiere los datos de un determinado proceso, en tiempo real, es decir, la respuesta del sistema Scada a las entradas se lo realiza en un tiempo razonable. Los datos adquiridos se los envía al operador final, o a otras aplicaciones.
- **Representación histórica de los datos:** CM presenta al usuario final datos pasados y actuales del sistema en forma de tablas, gráficas de tendencia y gráficos de históricos.
- **Almacenamiento de datos:** La estructura de CM se basa en un sistema de base de datos. El sistema de gestión de base de datos es el encargado de administrar los distintos datos del proceso.
- **Controladores en tiempo real:** CM se caracteriza por tener un módulo de generación de algoritmos de control. Es decir, el usuario puede programar un controlador dentro de la aplicación para modificar la respuesta de una variable controlada.
- Combina una gran cantidad de aplicaciones desarrolladas con software de código abierto para generar un sistema confiable de adquisición de datos y control. El medio de comunicación usado para la implementación es cable, aunque se podrían incorporar diversos medios de comunicación serial.

5.5.2 Características Básicas de un Scada desarrollado en CM

- Ofrece una vista real del proceso y la posibilidad de monitorear, supervisar y ejecutar algoritmos de control sobre el mismo.
- CM se ejecuta sobre computadores personales, y puede funcionar de manera individual, o en una red de gestión del sistema Scada. La red incluye Servidor Scada que se encarga de la adquisición de datos, un servidor para la administración de alarmas y controladores de CM; y el sistema que permite desarrollar y correr las diversas interfaces Hombre-Máquina.
- CM basa su ejecución en un sistema de gestión de información mediante bases de datos MySQL. La base de datos se encargan de almacenar los datos pasados, actuales, y las configuraciones básicas del sistema.

5.5.3 Características del sistema de gestión de Interfaces Hombre-Máquina desarrollado en CM

- CM presenta al operador una interfaz simple, lógica y correcta del proceso.
- Un sistema de gestión de interfaces Hombre-Máquina (HMI) desarrollado en CM presenta datos en tiempo real de las distintas variables, alarmas, reportes de eventos, tablas de datos, gráficos históricos y gráficos de tendencia.

5.6 CM Aplicación HMI/Scada. Descripción Funcional

5.6.1 Análisis Funcional

Para industrias pequeñas, mediante un análisis de costo/beneficio, no es rentable el realizar un sistema Scada usando software propietario, por lo que las soluciones de Software de distribución libre son adecuadas. Desde el punto de vista funcional, CM propone una herramienta de desarrollo de sistemas Scada para procesos pequeños y medianos. CM tiene como objetivo reducir los costos de implementación del sistema, y una arquitectura abierta para que se pueda adaptar a las necesidades del usuario final.

CM desarrolla sistemas HMI/Scada de manera simple, para procesos industriales pequeños y medianos empleando software libre y aprobado por la Iniciativa de Código

Abierto. CM se crea como una alternativa para el alto costo del Software propietario existente para el desarrollo de sistemas HMI/Scada.

5.6.2 Funciones HMI/Scada

- El sistema HMI/Scada desarrollado en CM ofrece al usuario la posibilidad de comunicarse y adquirir los datos usando las siguientes estrategias de comunicación.
  - Serial.
  - Serial Modbus RTU (Versión en prueba).
  - TCP/IP Modbus ASCII (Versión en desarrollo).
- El sistema HMI/Scada desarrollado en CM ofrece al usuario un sistema completo de gestión de datos.
  - Monitores de datos.
  - Alarmas.
  - Generación de reportes.
  - Redundancia.
  - Datos en tiempo real.

La figura 5.1 muestra las diferentes funciones llevadas a cabo por un sistema HMI/Scada

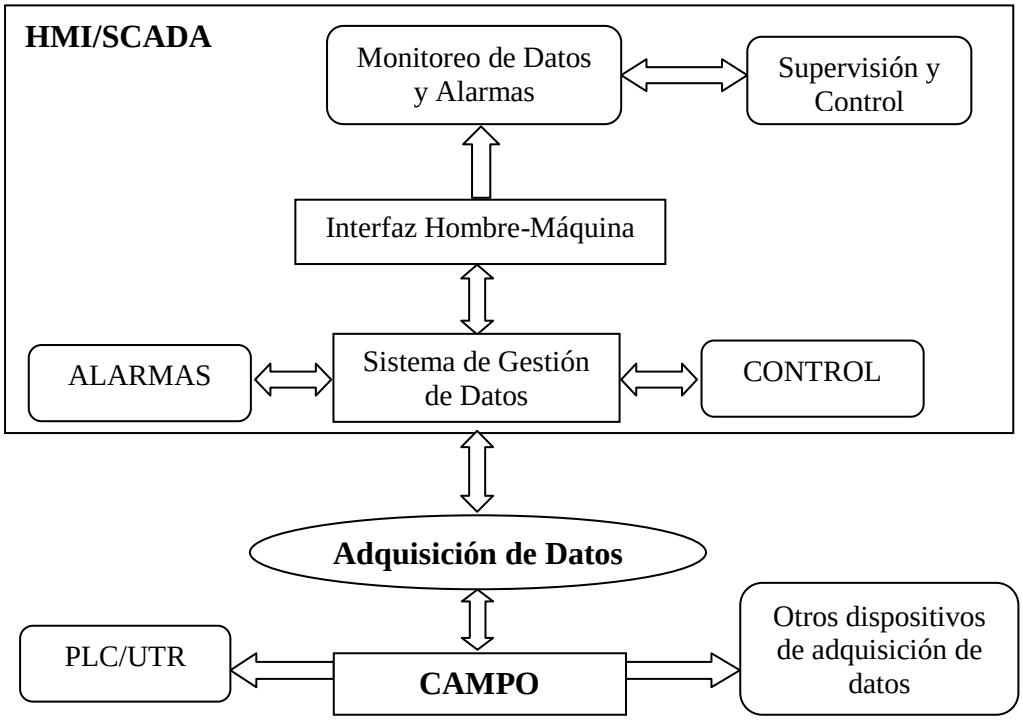

Figura 5.1: Esquema funcional de un sistema HMI/Scada

5.6.3 Funciones de CM

- Adquisición de datos.
    - Comunicación directa de los dispositivos de campo mediante el protocolo de comunicación industrial Modbus RTU, a través de la interfaz de comunicación serial.
- Sistema de gestión de datos.
    - Capacidad de administrar y procesar los datos adquiridos en una base de datos MySQL.
- Sistema de Control y Alarmas.
    - Capacidad de generar algoritmos de control e identificar alarmas, provenientes de valores fuera de los rangos definidos por el usuario.
- Sistema de Generación de Interfaces Hombre-Máquina (HMI).

5.7 Flujo de Datos en CM

El Flujo de la información en un sistema HMI/Scada desarrollado en CM es el siguiente:

- El Servidor Scada (CMScadaServer) adquiere datos desde los dispositivos de campo.
    - Datos adquiridos mediante la interfaz de comunicación Modbus RTU, el Servidor Scada adquiere los datos del dispositivo de campo X.
    - Los datos son transferidos a la tabla de adquisición de datos del proceso en tiempo real, específico para el dispositivo de campo X.
    - El tiempo en el cual el Servidor Scada adquiere los datos provenientes del dispositivo de campo X, se lo conoce como tiempo de acceso o tiempo de adquisición de datos (PollTime).
- El Administrador del Proceso (CMProcessAdmin) escanea las tablas de adquisición de datos del proceso en tiempo real para cada dispositivo.
    - El Administrador del Proceso verifica los datos adquiridos desde la tabla de datos del proceso en tiempo real.

  - Los datos son enviados a la base de datos del proceso.
  - El tiempo en el cual el Administrador del Proceso escanea los datos de las tablas de datos del proceso específicas de cada dispositivo, se lo conoce como tiempo de escaneo (ScanTime)
- El Cliente HMI (CMHMIstation) adquiere los datos provenientes de la Base de datos del Proceso.
  - CMHMIstation presenta la información del proceso en forma de despliegues.

Flujo de datos: Escritura en dispositivos de campo.

- CM permite escribir datos a los diversos dispositivos de campo.
  - Existen dos formas de escribir datos a los dispositivos de campo:
    - A partir del módulo de control.
    - Desde un Cliente HMI.
  - Si los datos son resultado del algoritmo de control, se transmiten desde el módulo de control de manera directa hacia la tabla de datos del proceso en tiempo real respectiva, y luego se almacena en la base de datos del proceso.
  - Los datos enviados desde un Cliente HMI, son enviados a la Base de datos del proceso y luego a la tabla de datos del proceso en tiempo real.
  - Finalmente, el Servidor Scada escribe los datos desde la tabla de datos del proceso en tiempo real hacia el dispositivo de campo.

La figura 5.2 muestra el flujo de datos dentro del sistema CM.

5.8 Arquitectura de CM

La figura 5.3 muestra la arquitectura general del software de desarrollo de sistemas HMI/Scada CM.

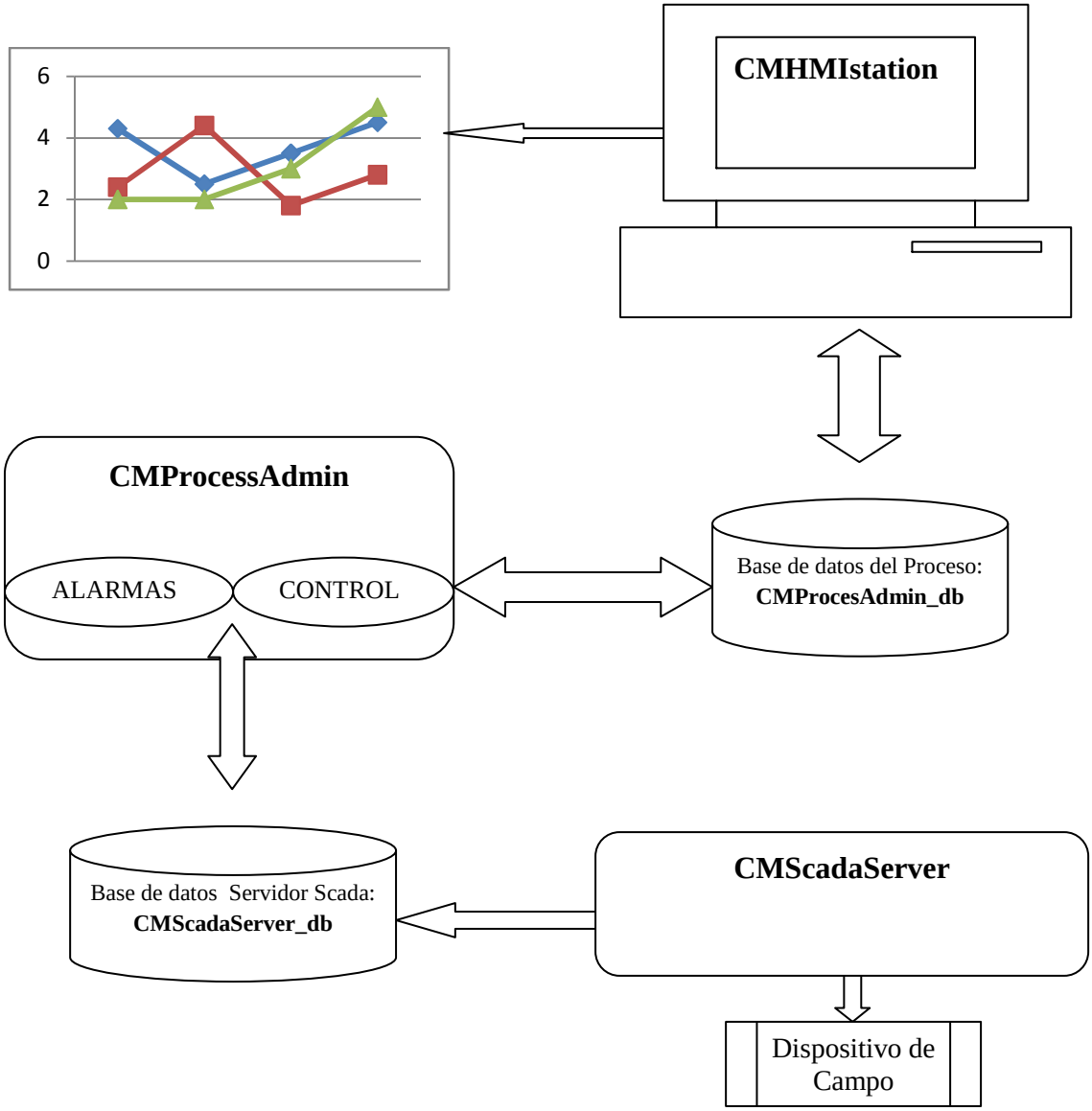

Figura 5.2: Flujo de datos dentro del sistema CM

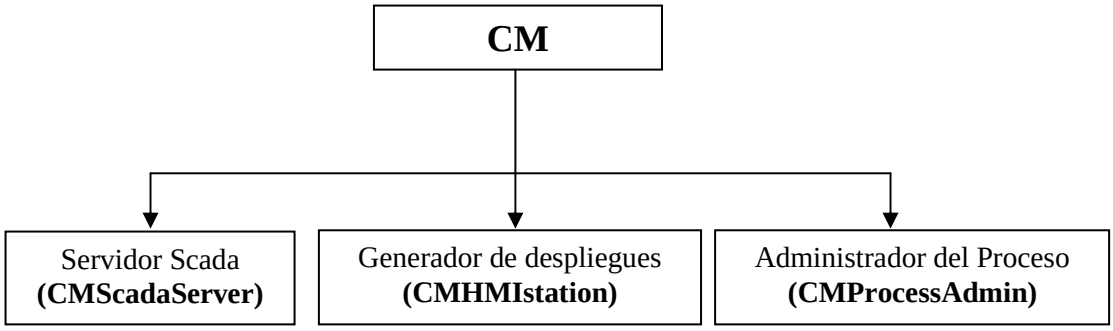

Figura 5.3: Arquitectura general de CM

5.8.1 Servidor Scada (CMScadaServer)

Es un ordenador/servidor encargado de adquirir los datos directamente de los dispositivos de campo.

Funciones

- Establece la comunicación con los dispositivos de campo, mediante el protocolo de comunicación Modbus RTU.
- Carga las **tablas de datos del proceso en tiempo real** específica para cada dispositivo**.**

Datos del Proceso

- CM trabaja conjuntamente con los dispositivos de campo que son los encargados de adquirir los datos del proceso.
- CMScadaServer adquiere los datos provenientes de los dispositivos de campo.

Comunicación con los dispositivos de campo

- La comunicación entre CM y un dispositivo de campo se lo realiza mediante el protocolo de comunicación industrial Modbus RTU. CMScadaServer se encarga de configurar y establecer la comunicación Modbus con los dispositivos de campo.
- CMScadaServer se encarga además de transferir los datos desde y hacia las tablas de datos del proceso en tiempo real específica para cada dispositivo de campo.

Tablas de datos del proceso en tiempo real.

- Tablas ubicadas en la base de datos de CMScadaServer, y en la cual el Servidor Scada almacena los datos adquiridos desde los dispositivos de campo. CMScadaServer se encarga de generar una tabla específica por cada dispositivo de campo o esclavo, dependiendo de las variables o datos provenientes del mismo.
- La Tablas de datos del proceso en tiempo real son actualizada en cada intervalo de tiempo conocido como: tiempo de acceso o tiempo de adquisición de datos (PollTime).

La siguiente figura muestra un ejemplo del funcionamiento de CMSacadaServer con dos PLCs

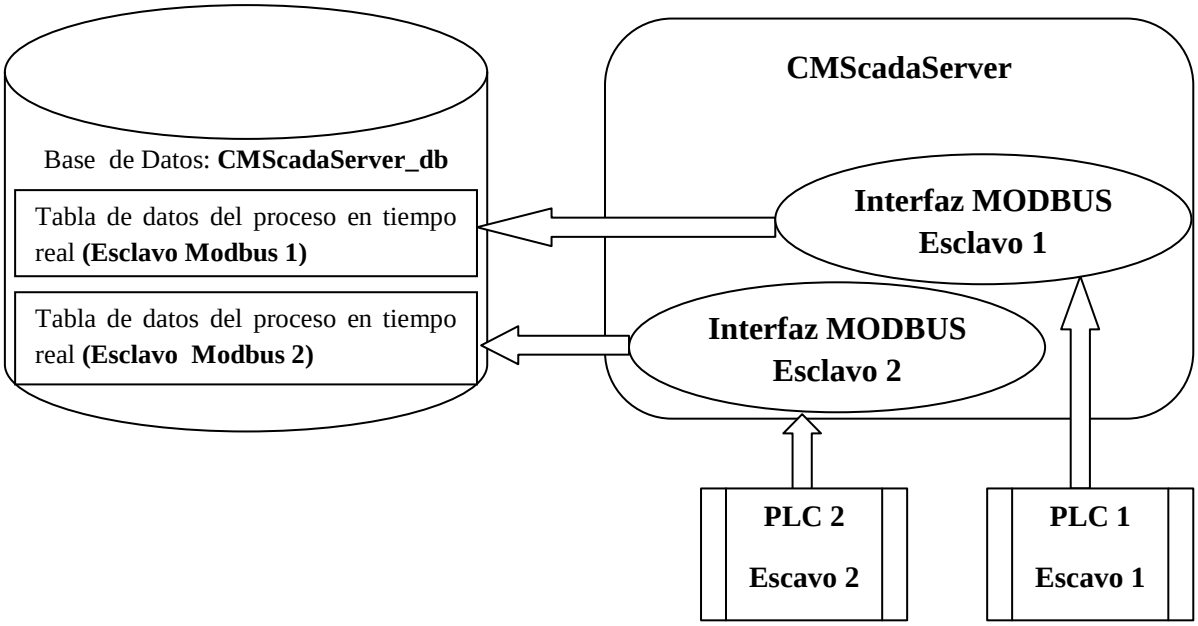

Figura 5.4: Diagrama del funcionamiento de CMScadaServer con dos PLC's

Finalmente, la figura 5.5 muestra el diagrama de flujo de la operación de CMScadaServer en cada intervalo de tiempo de acceso (PollTime).

5.8.2 Administrador del Proceso (CMProcessAdmin)

Es un ordenador/servidor, encargado de procesar los datos adquiridos por el Servidor Scada.

Funciones

- Cargar y mantener la **Base de Datos del Proceso.**
- Escanea las tablas de datos del proceso en tiempo real, en cada intervalo de tiempo conocido como tiempo de escaneo (Scan Time).
    - Cada variable o Tag tiene su propio tiempo de escaneo (Scan Time).
- Procesar alarmas establecidas por el usuario. Determinar si se encuentran fuera del rango preestablecido.
- Configurar y ejecutar el módulo de controladores del proceso.

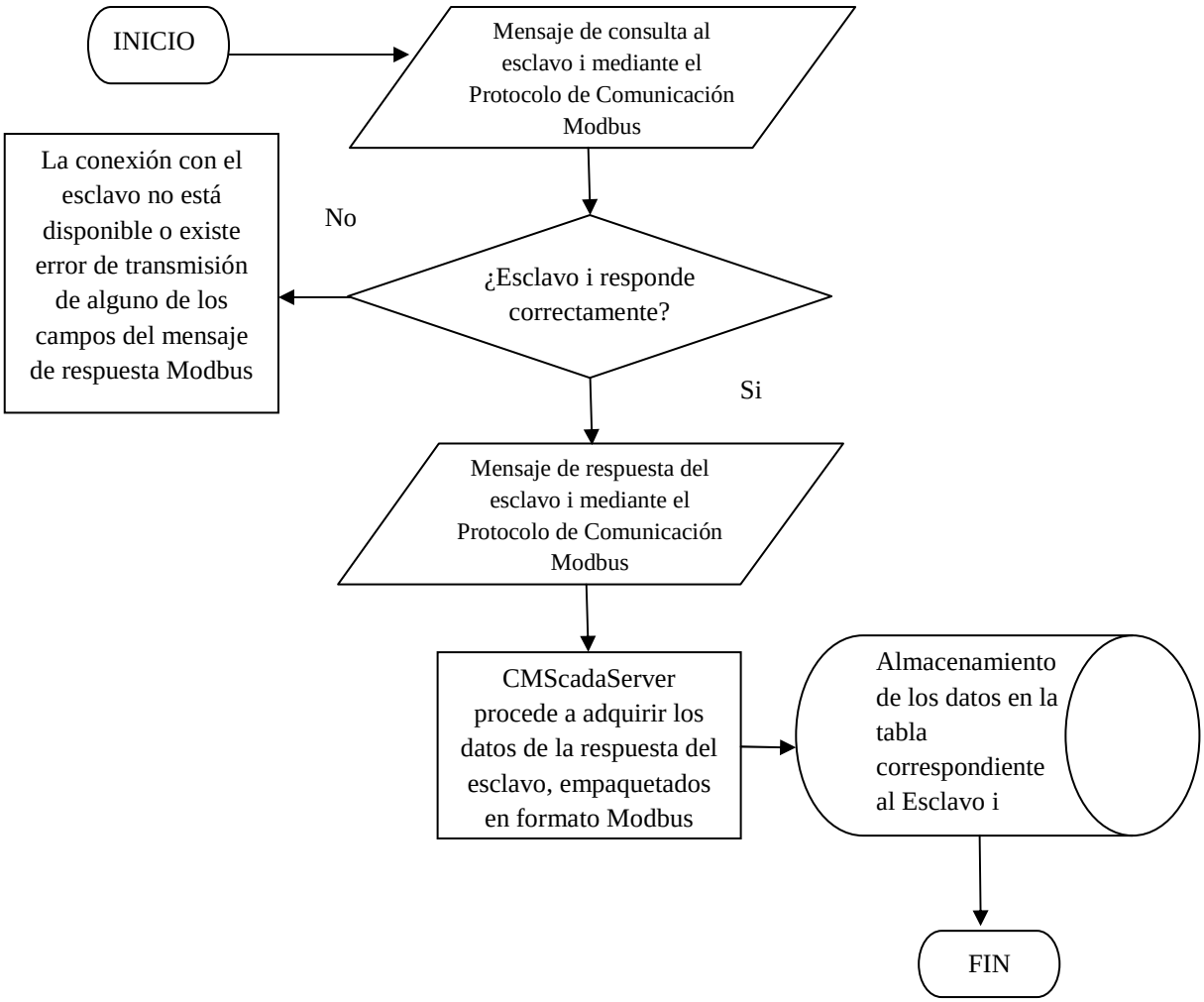

Figura 5.5: Diagrama de flujo de la operación básica del Servidor Scada de CM durante cada intervalo de acceso o PollTime

Base de Datos del Proceso (CMProcessAdmin_db)

- Almacena los datos de cada uno de los Tags asignados a las variables que intervienen en el proceso.
- Un Tag es una unidad de información asignada a una variable, un cálculo matemático, o a una función específica del proceso.

Módulo de Alarmas

- Determina si el valor de una variable ha salido del rango establecido para su operación normal, o ha entrado en un estado inesperado.
- Establece el tipo de alarma encontrado.
- Almacena y presenta las características de la alarma encontrada.
- Genera un reporte, indicando las causas de la alarma encontrada.

Módulo de Control

- Permite desarrollar y ejecutar algoritmos de control desarrollados en Python.

La figura 5.6 muestra un diagrama de flujo de operación de CMProcessAdmin en cada tiempo de escaneo para un Tag determinado.

5.8.3 Cliente HMI (CMHMIstation)

Es un ordenador o grupo de ordenadores encargados de recibir los datos procesados por el Módulo Administrador del Proceso y presentarlos en despliegues o pantallas de visualización del proceso (Interfaces Hombre-Máquina).

Funciones

- Presentar los datos y la información del proceso al usuario final. Por ejemplo:
    - Históricos.
    - Tablas de datos.
    - Gráficos de tendencia.
    - Reportes del sistema.
    - Datos en tiempo real.
    - Información adicional de las variables del sistema (unidades, estado, etc.).
    - Información de alarmas.

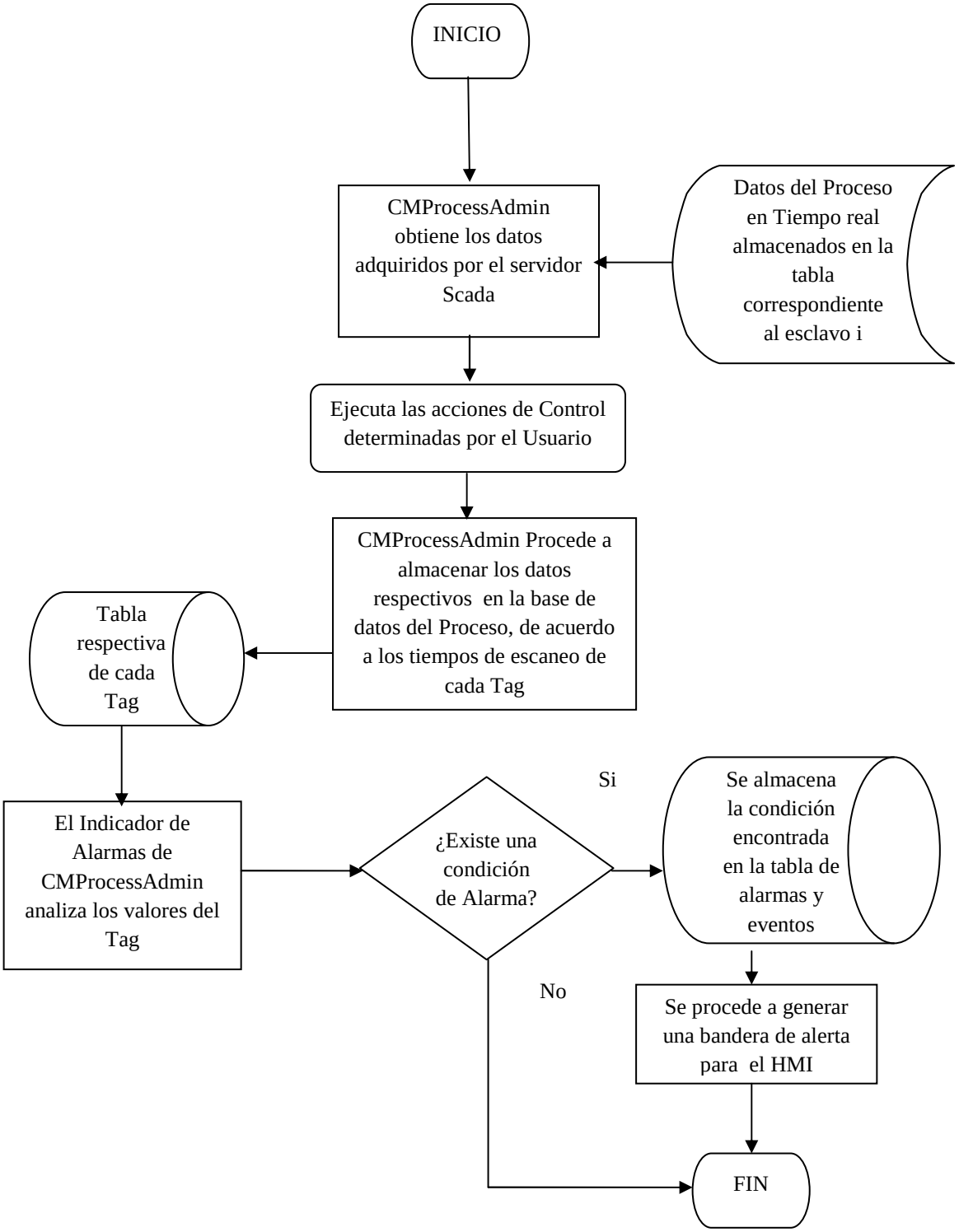

Figura 5.6: Diagrama de flujo de la operación básica del Administrador del Proceso de CM durante un intervalo de tiempo de escaneo de un Tag.

Operación del sistema de visualización del proceso (CMHMIstation)

El módulo CMHMIstation, tiene un modo de operación diferente a los módulos anteriores, ya que no sigue un flujo continuo de acciones. El módulo CMHMIstation es un sistema basado en eventos, es decir realiza sus acciones a partir de eventos originados por el usuario o por el propio sistema. La siguiente figura, muestra los distintos eventos que actúan sobre el módulo.

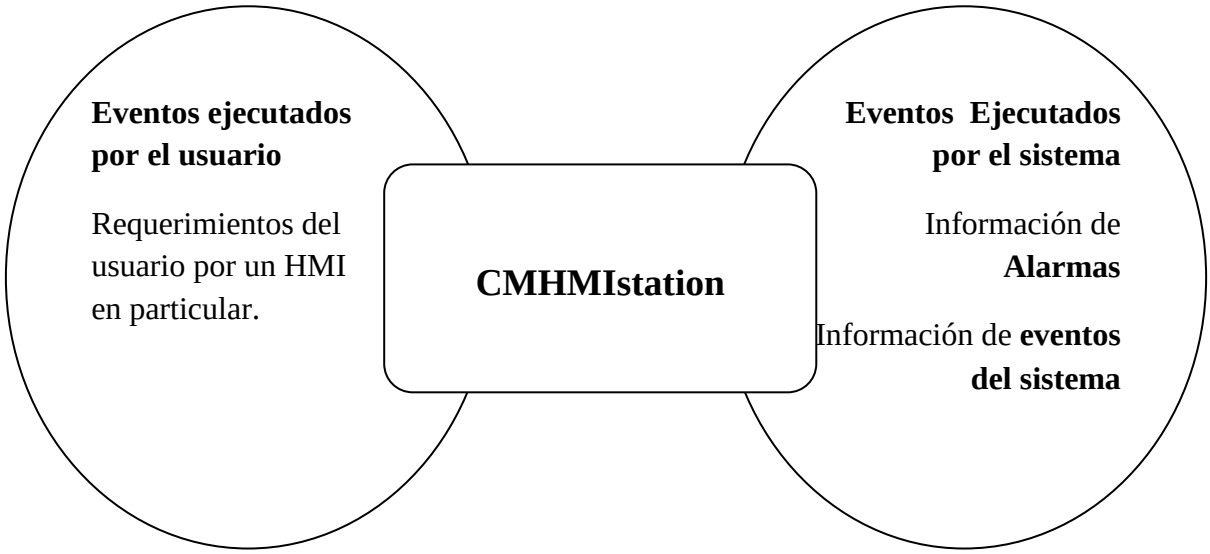

Figura 5. 7: Eventos que influyen sobre el comportamiento del módulo CMHMIstation.

Por otro lado, CMHMIstation tiene otra función la cual es configurar las variables del proceso, en bloques denominados Tags. Los Tags definen el tipo de variable o dato dentro del proceso.

Tipos de Tags configurados por CMHMIstation

Entrada Analógica (AI: Analog Input): Recibe datos de una señal analógica provenientes de la tabla de datos del proceso en tiempo real, cada tiempo de escaneo o acceso definido por el usuario.

Salida Analógica (AO: Analog Output): Envía una señal analógica hacia la tabla de datos del proceso en tiempo real. Por ejemplo, una salida analógica puede ser el resultado de un cálculo, o la salida de un controlador ejecutado por el módulo de controladores.

Cálculo (CA: Calculation): Permite ejecutar cálculos matemáticos a partir de una fórmula.

Entrada Digital (DI: Digital Input): Recibe una señal con dos estados posibles (0 o 1) proveniente de la tabla de datos del proceso en tiempo real.

Salida Digital (DO: Digital Output): Envía una señal con dos estados posibles (0 o 1), hacia una tabla de datos del proceso en tiempo real.

Gráfico de tendencia (RTP: real time plot): Permite visualizar el estado actual de una señal.

Para mayor información sobre los alcances de configuración de Tags referirse al Apéndice A, Manual de Usuario.

5.9 Arquitectura de la Base de Datos

El sistema de base de datos se subdivide en tres bases de datos de la misma forma que los módulos de CM. Las bases de datos son:

- CMScadaserver_db.
- CMHMIstation_db.
- CMProcessAdmin_db.

Este sistema se lo conforma de manera modular, para administrar la información del sistema a partir de los módulos. Sin embargo, se podría desarrollar todo el sistema en un solo servidor central y su respectivo respaldo, pero se necesitaría de un ordenador con una capacidad de procesamiento muy alta.

5.9.1 CMScadaServer_db

La base de datos CMScadaServer_db es la base de datos del Servidor Scada de CM (CMScadaServer). Las funciones de la base de datos son:

- Almacenar los datos provenientes del proceso, para las diferentes interfaces o esclavos en cada intervalo definido por el tiempo de acceso (Polltime).
    - Interfaz de comunicación Serial (Comunicación definida por el usuario).
    - Interfaz Serial Modbus RTU.
- Almacenar la configuración del Servidor Scada para su correcto funcionamiento.
    - Banderas de ejecución del sistema.
    - Contraseña.
    - Parámetros de la interfaz de comunicación Serial.
    - Definición del tiempo de escaneo (Polltime).

La base de datos CMScadaServer_db contiene tres tablas que cumplen las funciones antes mencionadas. Las tablas dentro de la base de datos CMScadaServer_db son:

- Tabla de configuración de CMScadaServer (**CMSSsettings_t**).
- Tabla de almacenamiento de los datos del proceso a través de la interfaz Serial de comunicación (**CMSSserial_t**).
- Tablas de almacenamiento de datos del proceso a través del Protocolo Serial Modbus RTU (**CMSSmodbusi_t**) donde i =0,1,2,3…n, donde n es el número de esclavo dentro del sistema.
- Tabla de configuración del protocolo de comunicación Modbus. (**CMSSmodbussettings_t**).

CMSSsettings_t

Descripción

La tabla CMSSsettings_t se encarga de almacenar los datos de configuración del Servidor Scada de CM y de la interfaz de comunicación Serial.

Campos

Los campos definidos por la tabla CMSSsettings_t son Parámetros y su correspondiente valor:

- **Parámetro:** Identifica los parámetros de configuración de CMScadaServer. Los parámetros definidos son:
    - Se definen los parámetros de configuración de la interfaz de comunicación Serial.

        - Serial, **valores:** Enabled, Disabled.
        - BaudRate, **valores:** Los definidos por la interfaz Serial (Ejemplo: 9600 bps, 19200bps, etc.).
        - Bits de datos, **valores:** 7 o 8.
        - Bits de Parada: **valores:** 1 o 2.
        - Paridad: **valores:** ODD, NONE, EVEN.
    - Banderas de control en la ejecución de la interfaz gráfica de usuario.
        - FlagSet, **valores**: 0 o 1.
        - FlagRun, **valores**: 0 o 1.
    - Definición del tiempo de escaneo (Polltime).
        - PollTime: **valores:** número entero.
    - Configuración de la contraseña que permite al usuario modificar la configuración del Servidor Scada.
        - Password, **valores**: cadena de caracteres sin espacio.

CMSSserial_t

Descripción

La tabla CMSSserial_t se encarga de almacenar los datos provenientes de un esquema de comunicación, a través de la interfaz Serial.

Campos

Los campos definidos por la tabla CMSSserial_t son Interface, isCreated y Value:

- **Interface:** Tiene dos componentes que son SerialIN (datos de entrada) y SerialOUT (datos de salida).
- **isCreated:** Define si SerialIN y SerialOUT han sido configurados.
- **Value:** Es el valor obtenido en cada intervalo de tiempo de escaneo por las interfaces.

CMSSmodbusi_t

Descripción

La tabla CMSSmodbusi_t se encarga de almacenar los datos obtenidos mediante el protocolo de comunicación Modbus del esclavo i.

Campos

Los campos definidos por la tabla CMSSmodbusi_t son Data y Value:

- **Data:** Son los tipos de datos usados en el protocolo de comunicación modbus. (Coils, DiscreteInputs, InputRegisters, HoldingRegisters)
- **Value:** Es el valor obtenido en cada dato, mediante las consultas del protocolo de comunicación Modbus RTU.

CMSSmodbussettings_t

La tabla de configuración de consultas Modbus se encarga de definir, las características de consulta para cada esclavo.

Campos

Los campos definidos por la tabla CMSSmodbussettings_t son los siguientes:

- Slave: Número del esclavo Modbus.
- f1enabled.
- f1start.
- f1number.
- f2enabled.
- f2start.
- f2number.
- f3enabled.
- f3start.
- f3number.
- f4enabled.
- f4start.
- f4number.
- f5enabled.
- f5data.
- f5value.
- f6enabled.
- f6data.
- f6value .

Como se puede observar, fi corresponde a la función i del protocolo. Además se define los campos correspondientes a los datos que estructuran las consultas del Maestro Modbus, para las funciones Modbus implementadas en CM.

5.9.2 CMHMIstation_db

La base de datos CMHMIstation_db es la base de datos del Cliente HMI de CM (CMHMIstation). Las funciones de la base de datos son:

- Almacenar la configuración del Cliente HMI para su correcto funcionamiento.
    - Banderas de ejecución del sistema.
- Almacenar las características de los Tags creados por el usuario.
- Almacenar eventos del sistema.

La base de datos CMHMIstation_db contiene tres tablas que desarrollan las funciones antes mencionadas. Las tablas dentro de la base de datos CMHMIstation_db son:

- Tabla de banderas de CMHMIstation (**CMHMIstflags_t**).
- Tabla del administrador de Tags (**CMHMIsttagsmanager_t**).
- Tabla de eventos (**CMHMIstEvents_t**).

CMHMIstflags_t

Descripción

La tabla CMHMIstflags_t se encarga de almacenar el valor de de las banderas de control de ejecución de las interfaces gráficas.

Campos

Los campos definidos por la tabla CMHMIstflags_t son flag y su correspondiente valor:

- **flag:** Identifica el nombre de la bandera de control.
- **Value:** (0 o 1) se determina si la bandera esta activada o no.

CMHMIsttagmanager_t

Descripción

La tabla CMHMIsttagamanager_t se encarga de almacenar las características de los Tags creados por el usuario.

Campos

Los campos definidos por la tabla CMHMIsttagamanager_t son:

- **Tagnumber:** Número de Tag creado.
- **Tagname:** Nombre del Tag.
- **Type:** Tipo de Tag (Entrada Analógica, Entrada discreta, Cálculo, etc.).
- **Communication:** Tipo de Comunicación.
- **I_O Address:** Dirección o interfaz de comunicación. En el caso del Tag tipo RTP o gráfico de tendencia, este campo se llena con el correpondiente Tag al que pertenece.
- **CurrValue:** Valor actual de los Tags.
- **Scan_Time:** Tiempo de escaneo y almacenamiento en la base de datos del proceso.
- **Description:** Descripción del Tag.
- **Low_off:** Corresponde al límite inferior en el caso de Tags de tipo analógico, o la etiqueta tipo off en los Tags de tipo digitales.
- **High_on:** Corresponde al límite superior en el caso de Tags de tipo analógico o la etiqueta tipo on en los Tags de tipo digitales.
- **Units_DA:** Unidades de los Tags analógicos, y tipo de alarmas de los tags digitales.
- **isCreated:** Permite determinar si el Tag ha sido creado (1) o eliminado (0).
- **Priority:** Prioridd de alarmas.
- **HHA_TDA:** Alarma de límite alto alto, y tipo de alarma en los Tags digitales.
- **HA:** Alarma de límite alto.
- **LLA:** Alarma de límite bajo bajo.
- **LA**: Alrma de límite bajo.
- **ROCA**: Alarma de velocidad de cambio.
- **V1:** Variable 1 del Tag de cálculo.
- **V2:** Variable 2 del Tag de cálculo.
- **V3:** Variable 3 del Tag de cálculo.
- **V4:** Variable 4 del Tag de cálculo.
- **V5:** Variable 5 del Tag de cálculo.
- **V6:** Variable 6 del Tag de cálculo.

CMHMIstEvents_t

Descripción

La tabla CMHMIstEvents_t se encarga de almacenar los eventos ocurridos en el sistema.

Campos

Los campos definidos por la tabla CMHMIstEvents_t son:

- **date:** Fecha del evento.
- **time:** Hora del evento.
- **Event:** Descripción del evento.

5.9.3 CMProcessAdmin_db

Descripción

La base de datos CMProcessAdmin_db es la base de datos del Administrador del Proceso de CM (CMProcessAdmin). Las funciones de la base de datos son:

- Almacenar y administrar las variables del proceso conocidas como Tags, en las tablas de la base de datos del proceso.
- Almacenar la base de datos de las alarmas.
- Almacenar los parámetros de configuración del Administrador del Proceso.

La base de datos CMProcessAdmin_db contiene tablas que desarrollan las funciones antes mencionadas. Las tablas dentro de la base de datos CMProcessAdmin_db son:

- Tabla de configuración de CMProcessAdmin (**CMPASettings_t**).
- Tabla para cada Tag requerido (**CMPATagi_t**).
- Tabla de alarmas (**CMPAlarms_t**).

CMPASettings_t

Descripción

La tabla CMPASettings_t se encarga de almacenar los parámetros de configuración del sistema.

Campos

Los campos definidos por la tabla CMPASettings_t son:

- **Parameter:** Nombre del parámetro de configuración o de la bandera de control ejecución.
- **Value:** Valor del parámetro.

CMPATagi_t

Descripción

La tabla CMPATagi_t se encarga de almacenar los datos del proceso para el Tag i.

Campos

Los campos definidos por la tabla CMPATagi_t son:

- **date:** Fecha del almacenamiento.
- **time:** Hora del almacenamiento.
- **value:** Valor almacenado.
- **alarms:** Descripción de alarmas ocurridas.

CMPAAlarms_t

Descripción

La tabla CMPAAlarms_t se encarga de almacenar las alarmas ocurridas en el sistema.

Campos

Los campos definidos por la tabla CMPAAlarms_t son:

- **Priority:** Prioridad de la alarma.
- **Date:** Fecha del evento.
- **Time:** Hora del evento.
- **TagName:** Nombre del Tag que ejecutó la alarma.
- **Status:** Tipo de alarma.
- **Valor:** Valor del Tag que ejecutó la alrma.

5.10 Descripción Estructural de CM

Estructuralmente los tres módulos que componen la aplicación, clasifican los programas que lo componen en dos grupos, los cuales son:

- **Gui:** Grupo constituido por los programas que estructuran las interfaces Hombre-Máquina (HMI) o interfaces gráficas de usuario del módulo.

- **Modules:** Grupo constituido por los programas que controlan el funcionamiento lógico del módulo.

Los programas desarrollados en Python tienen la terminación .py

5.10.1 CMScadaServer

Contenido dentro de la carpeta GUI de CMScadaServer

CMSSAboutGUIv01.py : Programa para generar el diálogo de presentación del sistema CM, desarrollado en HTML. La siguiente figura muestra la interfaz gráfica de usuario del diálogo de presentación.

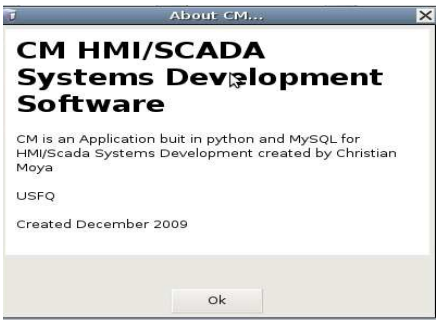

Figura 5.8: Diálogo de presentación del sistema CM, en HTML

CMSSModbussettingsmodv01.py: Programa para crear una interfaz gráfica de usuario, que permite configurar los parámetros de las consultas del Protocolo Modbus RTU. Los parámetros configurados son los siguientes:

- Número de esclavo dentro del sistema de control.
- Tipo de Protocolo Modbus (RTU).
- Características de la consulta para las 6 funciones implementadas en CM. En particular, se identifica el valor del registro, bobina o entrada discreta inicial, y el número de bobinas, registros o entradas discretas a ser consultadas. Para la escritura, se debe colocar en el campo de Inicio (Start), la dirección de la bobina, o registro interno a ser modificado.

CMSSRunHMIv01.py: Programa para implementar una interfaz Hombre-Máquina que se despliega cuando el Servidor Scada está operando. Tiene la capacidad de detener el módulo, y presenta una pequeña información sobre el sistema. Dentro de esta interfaz se

ejecuta el comando de inicio de adquisición de datos por parte del Servidor Scada. La figura 5.9 muestra la interfaz Hombre-Máquina generada.

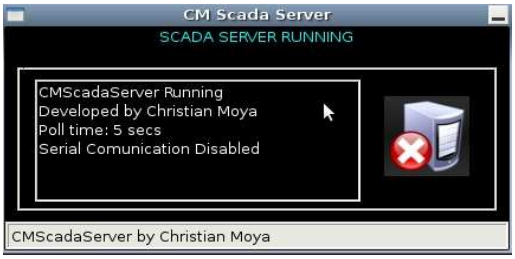

Figura 5. 9: HMI desplegado en la ejecución del Servidor Scada CM

CMSSSelectconfigGUIv01.py: Programa para generar una interfaz gráfica de usuario que permite seleccionar el tipo de configuración que se quiere realizar. Se puede escoger la configuración del Servidor Scada o la configuración del Protocolo de Comunicación Modbus.

CMSSSettingsGUIv01.py: Programa para crear la interfaz gráfica de usuario que permite modificar los siguientes parámetros dentro del Servidor Scada:

- Contraseña (Password).
- Tiempo de acceso o adquisición de datos (PollTime).
- Parámetros del puerto serial.
    - Puerto.
    - Velocidad de transmisión (Baudrate).
    - Número de bits de datos.
    - Número de bits de parada.
    - Paridad.

CMSSStartGUIv01.py: Programa que desarrolla la interfaz gráfica de usuario inicial, que permite ejecutar el Servidor Scada o modificar sus parámetros de configuración. Finalmente, a partir de esta interfaz se accede al diálogo de información del sistema. La siguiente gráfica muestra el GUI de Inicio del Servidor Scada.

Figura 5.10: GUI de inicio del Servidor Scada CM

Contenido de la carpeta MODULES dentro de CMScadaServer

CMSSCrc16modv01.py

Descripción General

Este programa se encarga de calcular el control de redundancia cíclica necesaria para la comunicación Modbus RTU.

Funciones

- Calcula el CRC (Control de Rredundancia cíclica) de una trama de bytes.

CMSSDatastoremanagerv01.py

Descripción General

Este programa se encarga de administrar las tablas del proceso en tiempo real, específicas para cada dispositivo.

Funciones

- Crear las tablas respectivas para los dispositivos de campo. Se crea tablas Serial y Modbusi, donde i representa el número de esclavo Modbus.
- Se encarga del almacenamiento de los datos provenientes del proceso.
- Las tablas generadas constituyen un mapa de memoria de acuerdo a la interfaz de comunicación utilizada (Esquema Serial o Modbus).

CMSSDialogsmodv01.py

Descripción General

Este programa se encarga de administrar los diálogos que se presentan en la interfaz gráfica, cuando se desea desplegar una información.

Funciones

- Genera diálogos dentro de la interfaz gráfica del Servidor Scada. Los tipos de diálogos presentados son:
    - Error.
    - Preguntas SI/NO.
    - Información.
    - Confirmación..

CMSSExecutionmodv01.py

Descripción General

Es el programa de ejecución del servidor.

Funciones

- Permite incorporar los distintos módulos y generar la actividad funcional del Servidor Scada.
- Determina el número de dispositivos de campo en la red, el número de interfaces y los datos esperados por la tabla del proceso en tiempo real.
- Ejecuta los comandos de adquisición, envío y almacenamiento de datos.

CMSSEventmanagermodv01.py

Descripción General

Es el programa que administra la información de eventos del Servidor Scada

Funciones

- Permite almacenar en la tabla CMHMIstEvents_t los eventos ocurridos en el Servidor Scada.

CMSSMathconvmodv01.py

Descripción General

Programa para realizar conversiones matemáticas.

Funciones

- Permite convertir de decimal a binario, de decimal a hexadecimal y de hexadecimal a binario.

CMSSModbusconfigmodv01.py

Descripción General

Programa que administra la configuración de Modbus.

Funciones

- Permite almacenar los tipos de consulta, las características de la consulta, el tipo de comunicación Modbus y el esclavo dentro de la red industrial.

CMSSModbusrtumodv01.py

Descripción General

Programa que se encarga de la comunicación Modbus.

Funciones

- Emplea las funciones necesarias para generar consultas y recibir respuestas de dispositivos de campo mediante el protocolo de comunicación industrial Modbus.

CMSSRs232modv01.py

Descripción General

Programa que administra el uso del puerto serial.

Funciones

- Permite la adquisición de datos mediante el puerto serial.

- Permite la escritura de datos mediante el puerto serial.

CMSSSettingsmodv01.py

Descripción General

Programa para administrar la configuración de la tabla que almacena la configuración del Servidor Scada.

Funciones

- Permite modificar la contraseña del usuario del Servidor Scada.
- Permite la modificación y obtención del tiempo de acceso o Polltime
- Emplea funciones para modificar la configuración del puerto serial.

CMSSTimermodv01.py

Descripción General

Programa que permite llamar, de manera continua y en intervalos iguales de tiempo a una determinada función. Para realizar la función de un timer continuo se usa el módulo threading propio de Python.

Funciones

- Permite ejecutar funciones de manera periódica.

CMSSTimemodv01.py

Descripción General

Programa para acceder a fechas y horas.

Funciones

- Emplea funciones para obtener la fecha y hora actual, vitales en el almacenamiento de datos.

5.10.2 CMProcessAdmin

Contenido dentro de la carpeta GUI de CMProcessAdmin

CMPAControllermanagerHMIv01.py: Programa para generar una interfaz Hombre-Máquina (HMI) para el desarrollo de controladores. Permite ejecutar algoritmos de control mediante el uso de código Python y código estructurado de CM. El código estructurado de CM se refiere al uso de Python para el sistema de base de datos, ya que es necesario tener acceso a las variables del proceso. La siguiente figura muestra el HMI para desarrollar algoritmos de control.

Figura 5. 11: HMI para diseño de controladores basados en Python

CMPARunHMIv01.py: Programa para desarrollar una interfaz Hombre-Máquina (HMI) equivalente a la descrita en el Servidor Scada, y la cual se despliega cuando el Administrador de Procesos entra en ejecución.

CMPASatartGUIv01.py: Desarrolla la interfaz gráfica de inicialización del sistema que se muestra en la figura 5.12. Permite escoger las acciones a realizar por el usuario. Mediante esta interfaz gráfica de usuario (GUI) el desarrollador del sistema puede generar controladores o ejecutar el Administrador del Proceso.

Figura 5. 12: GUI de inicialización del Administrador del Proceso

Contenido de la carpeta MODULES dentro de CMProcessAdmin

CMPAAlarmsindicatormodv01.py

Descripción General

Este programa se encarga de verificar la existencia de una alarma.

Funciones

- Se encarga de comparar el valor recibido desde los dispositivos de campo, con los límites establecidos por el usuario.
- Verifica cambios de estado en las variables.
- Genera un reporte de la alarma encontrada.

CMPAAlarmsmanagermodv01.py

Descripción General

Este programa se encarga de almacenar las alarmas encontradas.

Funciones

- Almacena las características de una alarma encontrada.
- Determina si una alarma ha sido reconocida o no.
- Permite obtener las características de una alarma.

CMPADatastoreinterfacemodv01.py

Descripción General

Este programa se encarga de establecer una conexión con las tablas de datos en tiempo real.

Funciones

- Permite obtener los datos de un determinado Tag provenientes de las tablas en tiempo real.

CMPADialogsmodv01.py

Descripción General

Este programa se encarga de administrar los diálogos que se presentan en la interfaz gráfica, cuando se desea desplegar una información.

Funciones

- Genera diálogos dentro de la interfaz gráfica del Administrador del Proceso. Los tipos de diálogos presentados son:
    - Error.
    - Preguntas SI/NO.
    - Información.
    - Confirmación.

CMPAEventinterfacemodv01.py

Descripción General

Es el programa que administra la información de eventos del Administrador del Proceso.

Funciones

- Permite almacenar en la tabla CMHMIstEvents_t los eventos ocurridos en CMProcessAdmin.

CMPAExecutionmodv01.py

Descripción General

Es el programa de ejecución del Administrador del Proceso.

Funciones

- Permite incorporar los distintos módulos y generar la actividad funcional del Administrador del Proceso.
    - Administrar la base de datos del proceso
    - Administrar las alarmas
    - Ejecutar algoritmos de control.
- Carga la base de datos del proceso en cada tiempo de escaneo propio de cada Tag.

CMPAProcessdatabasemodv01.py

Descripción General

Es el programa que almacena los datos en la base de datos del proceso.

Funciones

- Carga la base de datos del proceso en cada tiempo de escaneo definido para cada Tag.

CMPATimermodv01.py

Descripción General

Programa que permite llamar, de manera continua y en intervalos iguales de tiempo a una determinada función. Para realizar la función de un timer continuo se usa el modulo threading propio de Python.

Funciones

- Permite ejecutar funciones de manera periódica.

CMPATimemodv01.py

Descripción General

Programa para acceder a fechas y horas.

Funciones

- Emplea funciones para obtener la fecha y hora actual, vitales en el almacenamiento de datos.

CMPATagsinterfacemodv01.py

Descripción General

Programa para acceder a la tabla CMHMIstTagmanager_t.

Funciones

- Permite cargar el valor actual de las variables o Tags.
- Permite acceder a los valores que ejecutan las alarmas dentro del sistema.

5.10.3 CMHMIstation

Contenido de la carpeta GUI dentro de CMHMIstation

CMHMIstAboutGUIv01.py: Programa para generar un diálogo de presentación del sistema CM, en lenguaje HTML.

CMSSStartGUIv01.py: Programa que desarrolla la interfaz gráfica de usuario (GUI) inicial, que permite ejecutar el Servidor Scada o modificar sus parámetros de configuración. Finalmente, a partir de esta interfaz se accede al diálogo de información del sistema.

CMSSRunHMIv01.py: Programa para implementar una interfaz Hombre-Máquina (HMI) que se despliega cuando el Servidor Scada está operando. Tiene la capacidad de detener el módulo, y presenta una pequeña información sobre el sistema. Dentro de esta interfaz se ejecuta el comando de inicio de adquisición de datos por parte del Servidor Scada.

CMHMIstAlarmsHMIv01.py: La figura 5.13 muestra la interfaz Hombre-Máquina (HMI) desarrollada mediante este programa para la presentación de las alarmas en tiempo real. Además, la interfaz Hombre-Máquina desarrollada es el medio para ejecutar el comando de reconocimiento de alarmas.

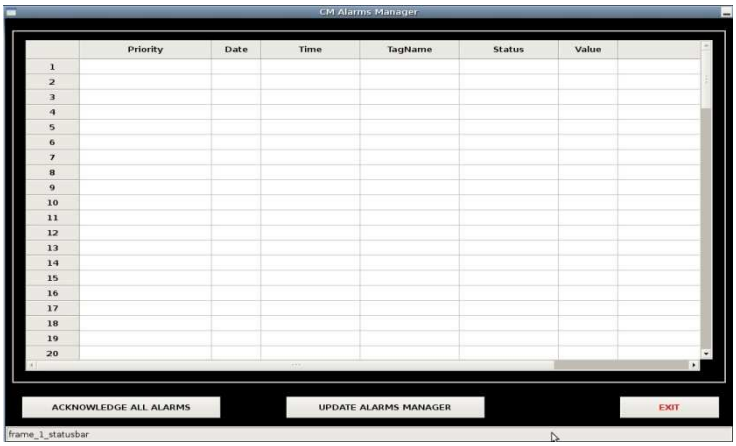

Figura 5.13: HMI de Alarmas

CMHMIstAnaloginputHMIv01.py: Programa para desarrollar una interfaz gráfica de usuario que permite configurar un Tag de tipo entrada analógica. Permite configurar la comunicación, límites y tiempo de escaneo de la variable.

CMHMIstAnaloginputHMIv01.py: Este programa crea una interfaz Hombre-Máquina que presenta los datos de una entrada analógica. Además, permite acceder a gráficos de tendencia, históricos y tablas de datos almacenados. La siguiente figura muestra el HMI de una entrada Analógica.

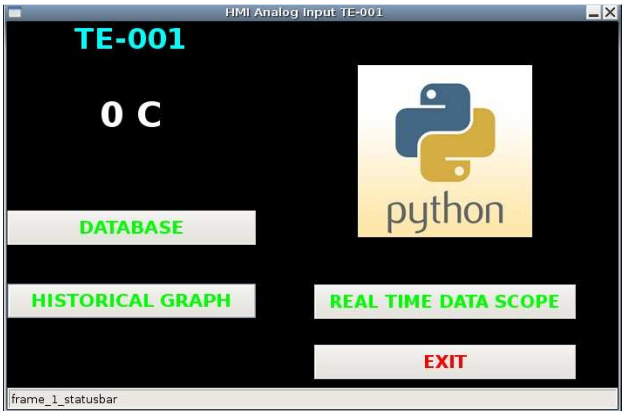

Figura 5.14: HMI Entrada Analógica

CMHMIstAnalogoutputGUIv01.py: Programa para desarrollar una interfaz gráfica de usuario que permite configurar un Tag de tipo salida analógica. Permite configurar la comunicación, límites y valor inicial del Tag.

CMHMIstAnalogoutputHMIv01.py Programa para desarrollar una interfaz Hombre-Máquina (HMI) de una salida analógica. Esta implementación se caracteriza por tener un "Slider", que permite modificar el valor en tiempo real. La figura 5.15 muestra el HMI resultante.

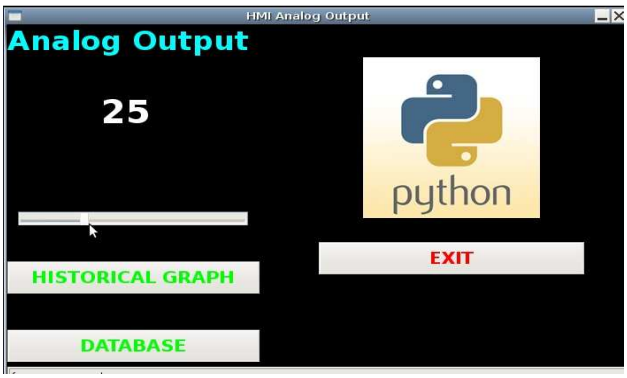

Figura 5.15: HMI Salida Analógica

CMHMIstCalculationGUIv01.py: Programa para desarrollar una interfaz gráfica de usuario que permite configurar un Tag de tipo cálculo. Permite configurar la las variables o Tags que intervienen en el cálculo, y la expresión correspondiente.

CMHMIstCalculationHMIv01.py: Programa para desarrollar una interfaz Hombre-Máquina (HMI) de un cálculo. El diseño del mismo es igual al del HMI de una entrada analógica.

CMHMIstDatabaseHMIv01.py: El programa desarrolla un HMI como el que se muestra en la figura 5.16, y que presenta los datos almacenados en forma de tabla, a partir de una consulta.

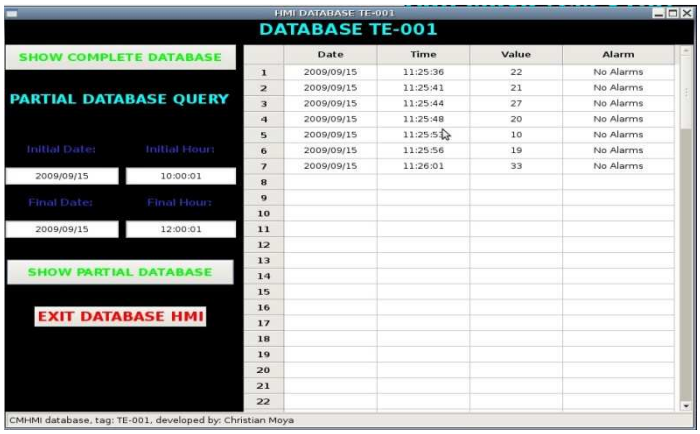

Figura 5.16: HMI Tabla de datos almacenados

CMHMIstDigitalinputGUIv01.py: Programa para desarrollar una interfaz gráfica de usuario que permite configurar un Tag de tipo entrada digital. Permite configurar las etiquetas de acuerdo al valor (0 o 1), la comunicación y el tiempo de escaneo.

CMHMIstDigitalinputHMIv01.py: La siguiente figura muestra una interfaz Hombre-Máquina (HMI) desarrollada a partir de este programa, el cual describe un Tag tipo entrada digital. Se puede observar que tiene inhabilitado los botones de cambio de estado.

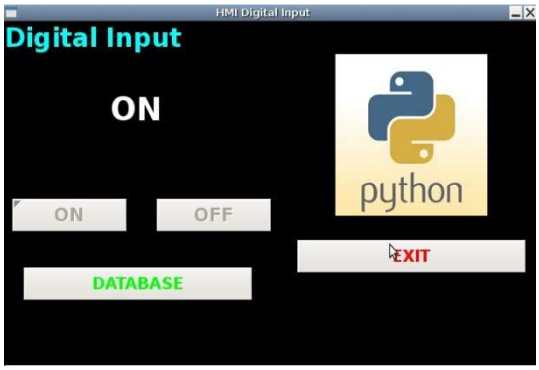

Figura 5.17: HMI Entrada Digital

CMHMIstDigitaloutputGUIv01.py: Programa para desarrollar una interfaz gráfica de usuario que permite configurar un Tag de tipo salida digital. Permite configurar las etiquetas de acuerdo al valor (0 o 1), la comunicación y el valor inicial.

CMHMIstDigitaloutputHMIv01.py: Este Programa desarrolla un HMI de la misma estructura que el HMI de una entrada digital, pero este tiene habilitados los botones de cambio de estado.

CMHMIstEventlogHMIv01.py: Programa para desarrollar un HMI con una estructura similar al HMI de las alarmas, y que permite presentar al usuario los eventos que han sucedido dentro del sistema.

CMHMIstHistplotHMIv01.py: Programa para desarrollar un HMI que presenta históricos, es decir gráficos con la evolución de las variables en el tiempo a partir de una consulta. La siguiente figura muestra el HMI resultante.

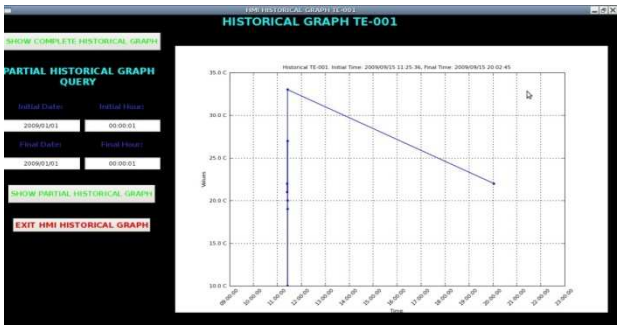

Figura 5.18: HMI Históricos

CMHMIstMainHMIv01.py: Programa que desarrolla la interfaz Hombre- Maquina (HMI), principal del sistema y mediante la cual el usuario puede acceder a los Tags, al administrador de alarmas o al administrador de eventos. La figura 5.19 muestra el despliegue principal de CM.

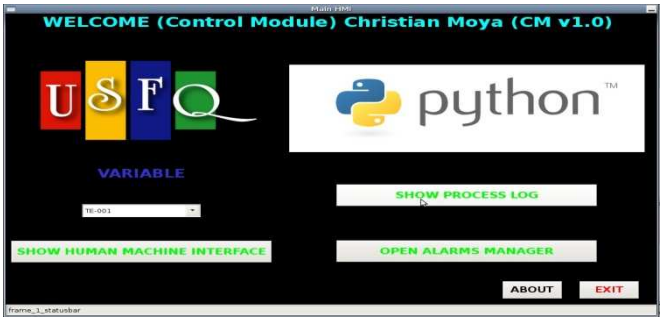

Figura 5.19: HMI Principal de CM

CMHMIstRealtimeplotGUIv01.py: Programa para desarrollar una interfaz gráfica de usuario que permite configurar un Tag de tipo gráfico de tendencia. Permite configurar límites, y el Tag del cual se va realizar el gráfico.

CMHMIstRealtimeplotHMIv01.py: Programa que desarrolla un HMI para la generación de gráficos de tendencia o gráficos en tiempo real. La 5.20 muestra el HMI desarrollado por este programa.

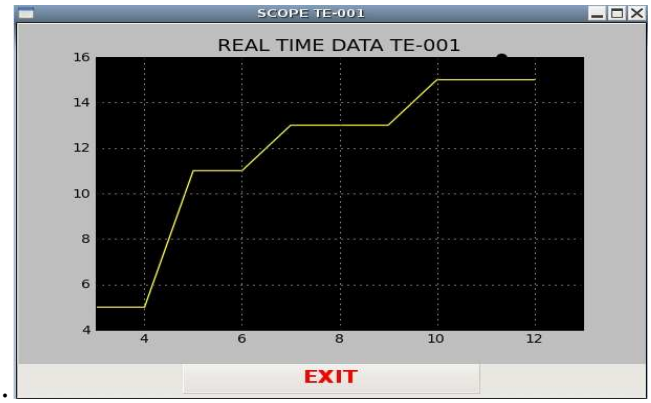

Figura 5.20: HMI Gráfico de Tendencia

CMHMIstSelecttypeGUIv01.py: Programa para desarrollar una interfaz gráfica de usuario que permite escoger el tipo de Tag a ser configurado.

CMHMIstStartGUIv01.py: Programa que desarrolla la interfaz gráfica de usuario de inicio de CMHMIstation. A partir de esta interfaz, el usuario puede ejecutar el Cliente HMI o realizar configuraciones. La siguiente figura muestra la interfaz gráfica de usuario de inicio de CMHMIstation.

Figura 5.21: Interfaz gráfica de usuario de inicio de CMHMIstation

CMHMIstTagmanagerGUIv01.py: Programa para desarrollar una interfaz gráfica de usuario que permite crear, eliminar y observar los Tags dentro del sistema. La figura 5.22 muestra la interfaz resultante.

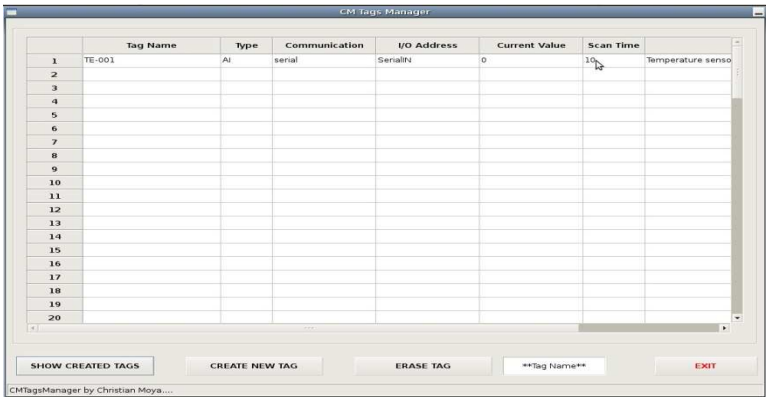

Figura 5.22: Interfaz gráfica de usuario de configuración y administración de Tags

Contenido de la carpeta MODULES dentro de CMHMIstation

CMHMIstAlarmsinterfacemodv01.py

Descripción General

Este programa se encarga de acceder a las alarmas encontradas por el administrador del proceso.

Funciones

- Permite obtener las características de las alarmas.
- Ejecuta el comando de reconocimiento de alarmas.

CMHMIstDialogmodv01.py

Descripción General

Este programa se encarga de administrar los diálogos que se presentan en la interfaz gráfica, cuando se desea desplegar una información.

Funciones

- Genera diálogos dentro de la interfaz gráfica del Cliente HMI. Los tipos de diálogos presentados son:

- Error.
- Preguntas SI/NO.
- Información.
- Confirmación.

CMHMIstEventmanagermodv01.py

Descripción General

Es el programa que administra la información de eventos del sistema.

Funciones

- Almacena los eventos del Cliente HMI.
- Permite desplegar el historial de eventos del sistema.

CMHMIstFlagsmanagermodv01.py

Descripción General

Es el programa que administra las banderas de control de ejecución del sistema de visualización de CM.

Funciones

- Cambia el estado de las banderas para generar una presentación visual lógica del proceso.
- Permite acceder a los distintos tipos de bandera que controlan la ejecución de despliegues visuales.

CMHMIstHistmodv01.py

Es el programa que genera las gráficas de históricos de los datos.

Funciones

- Permite generar gráficas de datos.
- Permite modificar títulos, etiquetas, y los componentes que definen un gráfico de los datos almacenados.

CMHMIstPDInterfacemodv01.py

Descripción General

Es el programa que accede a los datos almacenados en el proceso.

Funciones

- Permite realizar consultas de datos en diversos intervalos de tiempo.

CMHMIstTagmanagermodv01.py

Descripción General

Programa para administrar los Tags definidos para el proceso.

Funciones

- Permite almacenar las características de los Tags creados.
- Permite eliminar Tags.

CMHMIstTimemodv01.py

Descripción General

Programa para acceder a fechas y horas.

Funciones

- Emplea funciones para obtener la fecha y hora actual, vitales en el almacenamiento de datos.

5.11 Desarrollar un Sistema HMI/Scada en CM

Desarrollo de un sistema HMI/Scada usando CM.

1. Identificación del proceso
   - Comprensión total del proceso que se quiere representar, monitorear y controlar.

- Determinar las características e información específica del proceso y los instrumentos que participan en el mismo.
    - Determinar características de los PLCs, UTRs o dispositivos de adquisición de datos.
        - Determinar si tienen soporte de comunicación Modbus.
        - Tipos de datos que se va a monitorear.
        - Dirección de esclavo y funciones Modbus necesarias para acceder a los datos.
- Verificar que los ordenadores tengan instalado el software necesario para un normal funcionamiento de CM (Sistema Operativo: Linux, Python, wxpython, matplotlib, wxGlade y MySQL como gestor de bases de datos).
- Determinar la arquitectura(Número de ordenadores) a usar para el desarrollo del sistema HMI/Scada.

2. Desarrollar un sistema HMI/SCADA mediante CM
    - El Proceso de desarrollo del sistema debe cumplir los siguientes aspectos.
        - Instalación de los diferentes módulos (CMHMIstation, CMScadaServer y CMAlarms) en los ordenadores en los que serán distribuidos. En caso de ser un sistema individual de un solo ordenador, se debe cargar todos los módulos en el mismo.
        - Generación de la base de datos respectiva para cada módulo.
        - Configuración básica del sistema.
        - Definición de las variables a ser controladas mediante los Tags.
        - Incorporar funcionalidades específicas.
            - Gráficos de tendencia, alarmas, reportes, controladores, etc.
3. Determinar las características que debe tener CM, para obtener una visualización y control correcto del proceso.
    - Software Instalado.
        - El ordenador/servidor debe tener instalado todos los programas necesarios para correr un módulo de CM.
    - Base de datos del proceso.
        - Tipos de datos (Tags) a ser utilizados (Analógicos, Digitales, gráfico de Tendencia).

  - Determinar la forma de nombrar las variables. Por ejemplo, se puede usar la identificación definida en el estándar ANSI/ISA-5.1 “Símbolos de Instrumentación e Identificación”.
- Alarmas
  - Definir las alarmas en la configuración de CM. El usuario debe definir los rangos de las variables y los tipos de alarmas.
- Generación de Reportes
  - CM genera reportes de diferentes actividades del sistema, además puede generar reportes de actividades establecidas por el usuario.
- Control
  - CM posee un módulo que permite generar un número limitado de controladores, de acuerdo a las necesidades del sistema
- Almacenamiento de datos
  - CM almacena los datos en tablas cuyas características son definidas por el usuario (Base de datos relacional).
- Formas de visualizar los datos
  - Históricos, tabla de valores anteriores, gráficos de tendencia, valores en tiempo real, etc.

6

Consideraciones Finales

6.1 Conclusiones

CM es una aplicación creada a partir de software de código abierto y distribución libre, para el desarrollo de sistemas HMI/Scada. CM se desarrolló como una alternativa libre y sin costo para las aplicaciones propietarias de desarrollo de sistemas HMI/Scada.

- CM puede ser empleado, en un solo computador (no eficiente), pero también puede ser dividido en varios computadores conectados en red, gracias a su estructura modular.
- CM se puede comunicar con cualquier dispositivo de campo mediante el protocolo de comunicación Modbus RTU Serial.
- CM fue diseñado y estructurado de manera que pueda ejecutar algoritmos de control en tiempo real.
- Se desarrollo una aplicación usando exclusivamente recursos de distribución libre y código abierto, para el desarrollo de sistemas HMI/Scada.
- En la aplicación se desarrolló un sistema completo de gestión de información, mediante base de datos MySQL, la cual permite administrar y procesar los datos del proceso.
- CM permite configurar distintos tipos de datos, entre los cuales tenemos, entradas digitales, entradas analógicas, salidas digitales, salidas analógicas, entre otros.
- CM es capaz de controlar un proceso industrial de varias variables y en tiempo real.
- CM permite visualizar datos entiempo real, gráficos de tendencia, e históricos de las variables de la planta.

6.2 Trabajo Futuro

Para continuar con el desarrollo del sistema se ha propuesto lo siguiente:

- Desarrollo de un sistema de visualización de datos más dinámico.

- Desarrollo de una página web, para liberar el sistema y que pueda ser verificado por toda la comunidad.
- Mantener las pruebas sobre el sistema por 6 meses como mínimo y hasta 18 meses para que pueda ser considerado estable.
- Un trabajo de adecuación posible sería la creación de un módulo que permita generar HMIs, animados a partir de archivos creados en AutoCad o las versiones libres de Cad.
- Generar los instaladores para distintas plataformas, especialmente Windows.
- Generar un servidor OPC mediante Python, para comunicación con dispositivos de campo.
- Otro trabajo propuesto puede ser el desarrollo del protocolo Modbus TCP/IP, usando el módulo sockets de Python.

Otro tipo de trabajos futuros posibles, siguiendo la misma línea de software de código abierto y de distribución libre son:

- Desarrollar una plataforma para simulación sistemas de control estilo Simulik usando Python.
- Generar una caja de herramientas (toolbox) mediante Python para DSP.
- Generar una caja de herramientas (toolbox) mediante Python para sistemas de control.

Referencias

[1] D. Bailey and E. Wright, *Practical SCADA for Industry*, Oxford: Elsevier, 2003.

[2] IDC Technologies, *Practical distributed control systems (DCS) for engineers and technicians.* IDC Technologies, 2003.

[3] G. Clarke, D. Reynders and E. Wright, *Practical Modern SCADA Protocols: DNP3, 60870.5 and Related Systems*. Oxford: Elsevier, 2004.

[4] A. Rodriguez, *Sistemas Scada: Guía Práctica*. Barcelona: Marcombo, 2007.

[5] Modbus-IDA, *MODBUS over Serial Line Specification & Implementation guide V1.02*. 2006.

[6] Modbus-IDA, *Modbus Application Protocol Specification V1.1b*. 2006.

[7] J. Sanchez, "Apuntes de Sistemas de Gestión de Base de Datos,".[Online]. Disponible: http://www.jorgesanchez.net . [Acceso: Nov. 16, 2009].

[8] American National Standard, *ANSI/ISA-5.1-1984 (R1992): Instrumentation Symbols and Identification*. 1992.

[9] Instruments Society of America, *Standard ISA-5-5-1985*: *Graphic Symbols for Process Displays.* North Carolina: ISA, 1985.

[10] S. Hunger, *Debian GNU/Linux Bible*. New York: Hungry Minds, Inc., 2001.

[11] C. Negus, *Linux Bible 2006 Edition: Bootup to Fedora, KNoPPIX, Debian, Suse, Ubuntu and 7 Other Distributions*. Indianapolis: Wiley Publishing, Inc., 2006.

[12] M. F. Krafft, *The Debian System Concepts and Techniques*. Munich: Open Source Press, 2005.

[13] R. Gupta , *Making Use of Python*. New York: Wiley Publishing, 2002.

[14] G. Van Rossum, *Python Tutorial Release 2.6.1*. Python Software Foundation, 2009

[15] M. Lutz, *Programming Python*, third Edition. United States of America: O'Reilly, 2006.

[16] S. Suehring, *MySQL Bible*. New York: Wiley Publishin, Inc, 2002.

[17] P. Nielsen, *SQL Server 2005 Bible*. Indianapolis: WiIley Publishing, Inc., 2007.

[18] MySQL AB, *MySQL 5.0 Reference Manual*, 2009.

[19] A. Dustman, “MySQLdb User’s Guide”. [Online]. Disponible: http://mysql-python.surceforge.net/MySQLdb.html. [Accesed: Nov. 25, 2009].

[20] D. Dale, M. Droettboom, E. Firing, J. Hunter, *Matplotlib Release 0.985.1*. 2008.

[21] N. Rappin R. Dunn, *wxPython in Action*. United States of America: Manning, 2006.

[22] J. Nilsson, “Real-Time Control Systems with Delays,” Ph.D Thesis, Lund Institute of Technology, Lund, 1998.

[23] P. A. Laplante, *Real-Time Systems Design and Analysis.* United States of America: IEEE Press, 2004.

[24] P. Albertos and A.Sala, *Multivariable Control Systems: An Engineering Approach*. United States of America: Springer, 2004.

[25] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control Analysis and Design.* England: John Wiley & Sons.

[26] O. N. Gasparyan, *Linear and Nonlinear Multivariable Feedback Control: A Classical Approach.* England: John Wiley & Sons, 2008.

A

Manual de Configuración de la Aplicación CM

CM fue diseñado para ser un sistema funcional, pero a la vez muy fácil de implementar y configurar. La figura A.1 muestra un diagrama de los procesos de configuración necesarios, para un funcionamiento correcto de la Aplicación. Antes de comenzar con la configuración se debe verificar que todos los programas descritos en la Unidad 3 estén correctamente instalados.

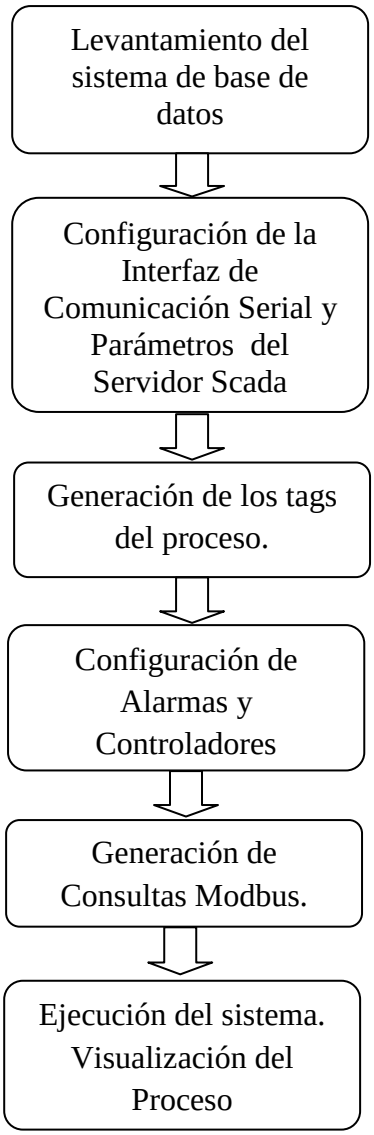

Figura A.1: Diagrama de flujo de los Procesos de configuración de CM

A.1 Levantamiento del sistema de base de datos

El primer paso de configuración del sistema es levantar el sistema de gestión de información, para esto es necesario crear las siguientes bases de datos:

- CMHMIstation_db
- CMProcessAdmin_db
- CMScadaServer_db

Para la creación de las bases se debe realizar el siguiente proceso (Linux):

1. Abrir el terminal de Linux.
2. Ejecutar los siguientes comandos dentro del terminal:
    - $mysql –h localhost –u root –p
    - ENTER PASSWORD:
    - mysql > CREATE DATABASE CMHMIstation_db;
    - mysql > CREATE DATABASE CMScadaServer_db;
    - mysql > CREATE DATABASE CMProcessAdmin_db;
    - quit

Para la creación de las tablas necesarias del proceso se necesita realizar lo siguiente:

Tablas CMHMIstation

Ubicarse en el lugar donde se encuentra la carpeta CMHMIstation, mediante el comando cd. Por ejemplo en este caso sería.

$ cd /home/Christian/Desktop/CM/CMHMIstation

Dentro de la carpeta mencionada se debe ejecutar el siguiente comando, el cual creará las tablas necesarias del módulo:

$python CMHMIstCreatetablesmodv01.py

Tablas CMProcessAdmin

Ubicarse en el lugar donde se encuentra la carpeta CMProcessAdmin, mediante el comando cd. Por ejemplo en este caso sería.

$ cd /home/Christian/Desktop/CM/CMProceesAdmin

Dentro de la carpeta mencionada se debe ejecutar el siguiente comando, el cual creará las tablas necesarias del módulo:

$python CMPACreatetablesmodv01.py

Tablas CMScadaServer

Ubicarse en el lugar donde se encuentra la carpeta CMScadaServer, mediante el comando cd. Por ejemplo en una computadora ejecutando Ubuntu, de nombre Christian y con la ubicación correspondiente sería.

$ cd /home/Christian/Desktop/CM/CMScadaServer

Dentro de la carpeta mencionada se debe ejecutar el siguiente comando, el cual creará las tablas necesarias del módulo:

$python CMSSCreatetablesmodv01.py

Creado el sistema de gestión de base de datos y las tablas necesarias, se puede comenzar a utilizar cada uno de los módulos. Cada módulo tiene una interfaz gráfica de inicio y que debe ser ejecutada para comenzar a utilizarlo. Si es necesario el usuario puede crear atajos o links de las interfaces de inicio, al Escritorio por ejemplo. Para crear los links se los puede realizar de manera gráfica o mediante los siguientes comandos.

$ln –s /path/to real-file/CM/CMScadaServer/gui/CMSSStartGUIv01.py /path/to-new /file.

$ln –s /path/to real-file/CM/CMProcessAdmin/gui/CMPAStartGUIv01.py /path/to-new /file.

$ln –s /path/to real-file/CM/CMHMIstation/gui/CMHMIstStartGUIv01.py /path/to-new /file.

A.2 Configuración de la interfaz de comunicación Serial y parámetros del Servidor Scada

1. Se ejecuta la interfaz gráfica de inicio del Servidor Scada, que se encuentra en forma de Link en la ubicación especificada por el usuario.

2. Dentro de la interfaz gráfica escoger la opción Scada Server settings. La contraseña que se utiliza en la primera ocasión es CM. Después, el usuario debe escoger la opción ScadaServer configuartion lo cual desplegará una interfaz gráfica de usuario tipo notebook(Con pestañas: Basic y Serial). La figura A.1 muestra la interfaz gráfica de la pestaña Basic.

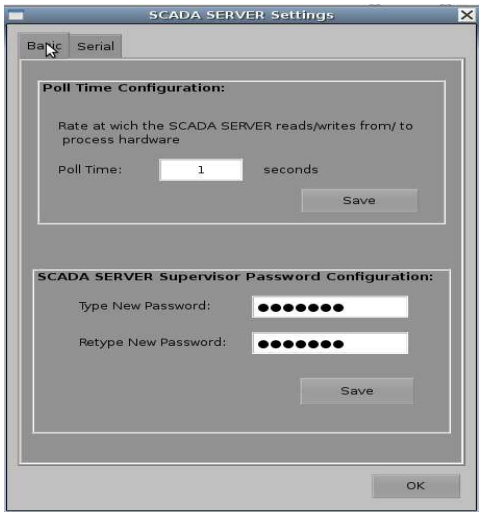

Figura A.2: Interfaz gráfica de usuario para la configuración básica del Servidor Scada

3. El Usuario puede modificar la contraseña que le permite realizar cambios en la configuración del sistema y el tiempo de acceso o de adquisición de datos(Poll Time).
4. La siguiente figura muestra el contenido de la pestaña Serial.

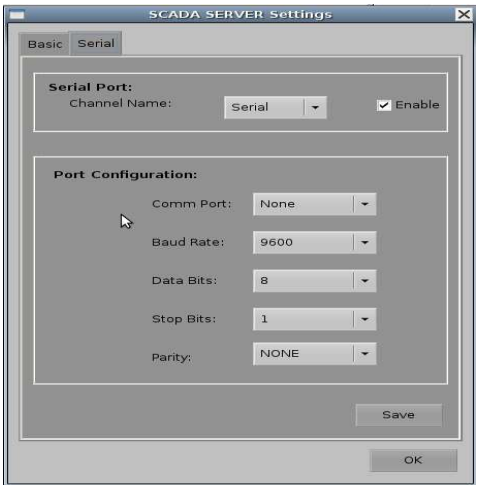

Figura A.3: Interfaz gráfica de usuario para la configuración del puerto serial

5. El usuario puede modificar las características del puerto serial y activar la comunicación. CMScadaServer permite la activación solo si existe físicamente la misma y el puerto escogido por el usuario puede ser utilizado.

A.3 Generación de los tags del proceso

El usuario debe ingresar esta vez a la interfaz gráfica de usuario (GUI) de inicio del módulo CMHMIstation, y escoger la opción OPEN CMHMIstation configuration, tal como se muestra en la siguiente figura.

Figura A.4: Interfaz gráfica de usuario de inicio del módulo CMHMIstation

Aparecerá el administrador de Tags, el cuál permite crear cinco tipos de Tags:

- Entrada Analógica (Analog Input)
- Salida Analógica (Analog Output)
- Entrada Discreta (Digital Input)
- Salida Discreta (Digital Output)
- Gráfico de Tendencia (Real Time Plot)
- Cálculo (Calculation)

A.3.1 Configuración de Entradas Analógicas (Analog Input)

La figura A.5 muestra la interfaz de configuración de una entrada analógica. El usuario define los siguientes campos:

- **Nombre del Tag**
- **Descripción**
- **Comunicación.** Define si la comunicación es Serial o Serial Modbus RTU.

- **I/O Address:** El usuario define la dirección. Por ejemplo, un registro de entrada Modbus tiene dirección: Slave01D3001.
- **Scan Time:** Define el tiempo de acceso y almacenamiento en la base de datos del proceso.
- **Limits:** Define los límites de la variable.

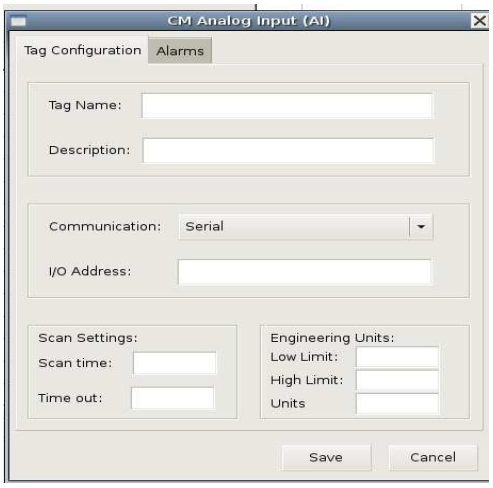

Figura A.5: Interfaz gráfica de usuario de configuración de entradas analógicas

A.3.2 Configuración de Salidas Analógicas (Analog Output)

La configuración de salidas analógicas se la realiza en una interfaz como la desplegada en la siguiente figura.

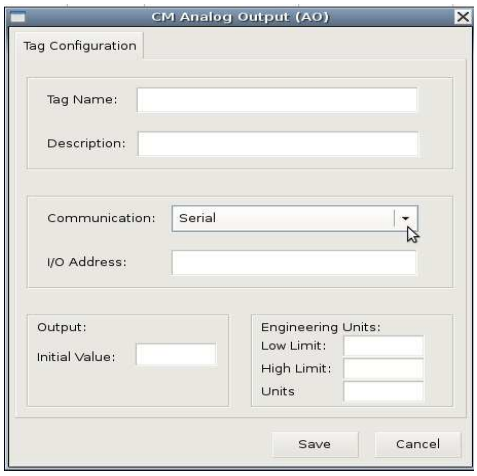

Figura A.6: Interfaz gráfica de usuario de configuración de salidas analógicas

Los campos de configuración son similares a los de la configuración de entradas analógicas, sin embargo no se define tiempo de escaneo ya que el mismo depende del desarrollo de los controladores.

Configuración de Entradas Discretas (Digital Input)

La figura A.7 muestra la interfaz de configuración de entradas discretas, que tienen la misma estructura descrita anteriormente, sin embargo el usuario puede definir la etiqueta cuando el tag tenga valor de 1 (ON) o el valor de 0 (OFF).

Figura A.7: Interfaz gráfica de usuario de configuración de entradas digitales

Configuración de Salidas Discretas (Digital Output)

Las salidas discretas se configuran casi de la misma forma que las entradas discretas, sin embargo permiten al usuario definir el estado inicial de la variable. No se define un tiempo de escaneo ya que el usuario genera eventos como cambios de estado en cualquier instante de tiempo.

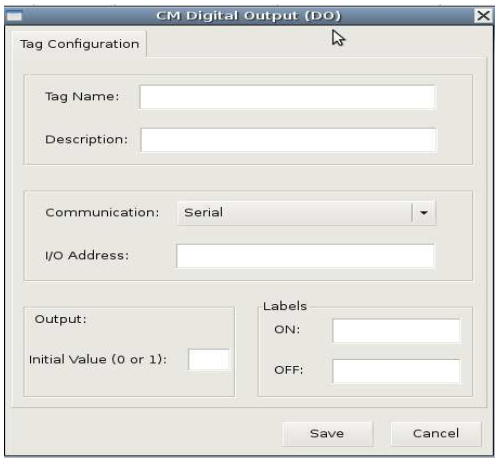

Figura A.8: Interfaz gráfica de usuario de configuración de salidas digitales

Configuración de Gráfico de Tendencia (Real Time Plot)

Los gráficos de tendencia permiten que variables analógicas puedan tener un despliegue o visualización de datos en tiempo real. La siguiente figura muestra la interfaz de configuración de los gráficos de tendencia.

Figura A.9: Interfaz gráfica de usuario de configuración de gráficos de tendencia

Configuración de Cálculos (Calculation)

CM permite realizar cálculos a través de una expresión y el uso de seis variables de entrada. La figura A.10 presenta la interfaz de configuración de cálculos.

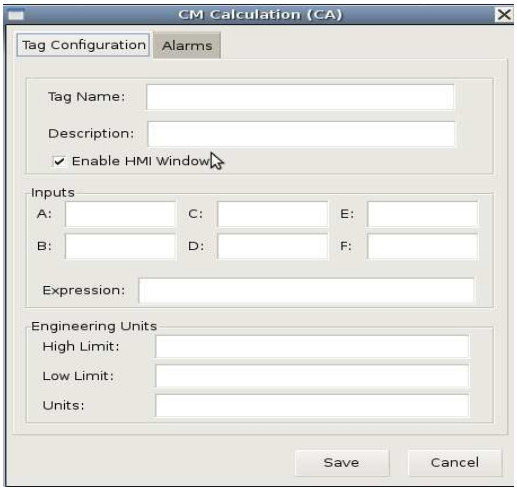

Figura A.10: Interfaz gráfica de usuario de Configuración de Cálculos

A.4 Configuración de Alarmas y Controladores

Alarmas

CM permite configurar alarmas dependiendo del tipo de variable analizada. Podemos definir dos tipos de configuraciones de alarmas:

- Configuración de alarmas a partir de entradas analógicas o cálculos.
- Configuración de alarmas a partir de entradas digitales.

La figura A.11 muestra la interfaz de configuración de Alarmas a partir de variables analógicas. Dentro de esta interfaz se define la prioridad, y los valores para los cuales se ejecuta una acción de alarma.

Los tipos de Alarmas definidas para una variable analógica son:

- High High (HH)
- High (H)
- Low Low (LL)
- Low (L)
- Rate of Change (ROC)

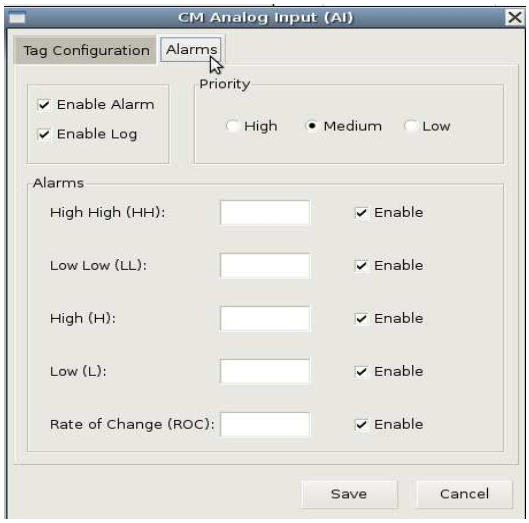

Figura A.11: Interfaz Gráfica de Usuario de Configuración de Alarmas de Entradas Analógicas y Cálculos

Por otro lado, en la figura A.12 podemos observar la interfaz de configuración de alarmas de variables discretas. Se define la prioridad, y el tipo de alarmas. Los tipos de alarmas definidos son:

- None
- Change of State
- Cambio a estado ON
- Cambio a estado OFF

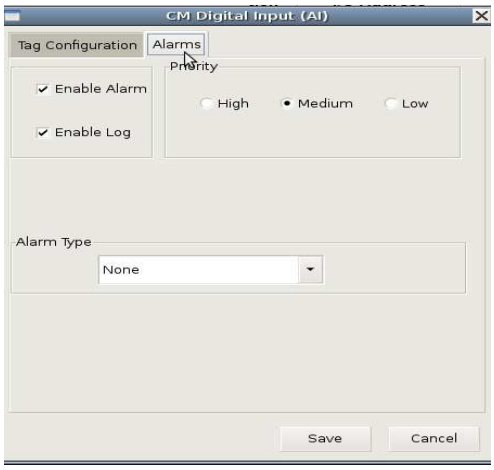

Figura A.12: Interfaz gráfica de usuario de configuración de alarmas de entradas digitales.

Controladores

La figura A.13, muestra la Interfaz Hombre-Máquina (HMI) que se despliega a partir de la interfaz gráfica de usuario de inicio del Administrador del Proceso. A partir de este HMI el usuario puede configurar los algoritmos de control del proceso.

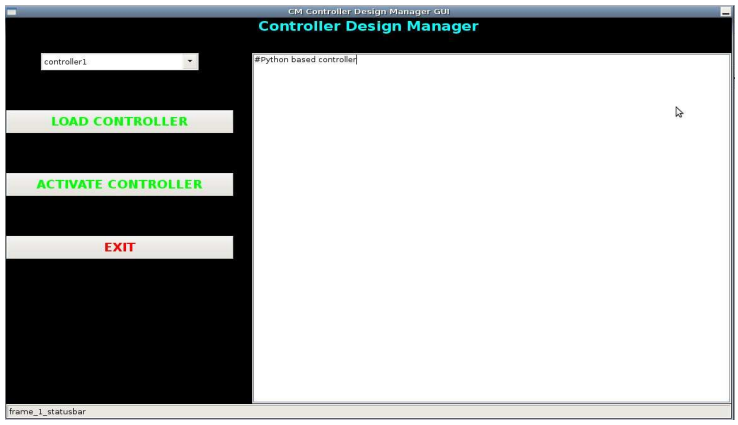

Figura A.13: Interfaz Hombre-Máquina para la generación de algoritmos de control

A.5 Generación de consultas Modbus

Después de definir los tags dentro del proceso, el usuario debe definir cada una de las consultas Modbus necesarias para adquirir las variables del proceso. Mediante la interfaz gráfica de usuario (GUI) que se muestra a continuación, se definen las características de los datos adquiridos, y se puede definir variables de salida mediante las dos últimas funciones.

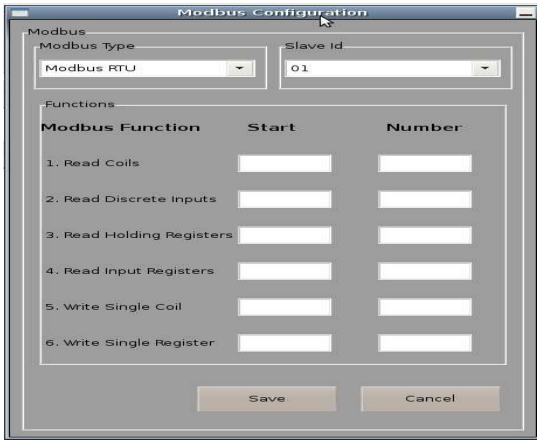

Figura A.14: Interfaz gráfica de usuario de configuración de consultas Modbus

A.6 Ejecución del sistema y Visualización del Proceso

Ejecución del Sistema

Para ejecutar el sistema CM el usuario debe correr cada uno de los módulos a partir de las interfaces gráficas de usuario (GUI) de inicio de los mismos.

Visualización del Proceso

La visualización del Proceso se la administra a través del despliegue que se muestra en la figura A.15. Mediante esta Interfaz Hombre-Máquina el usuario puede acceder a los datos del proceso distribuidos en Tags y a administradores de alarmas y eventos.

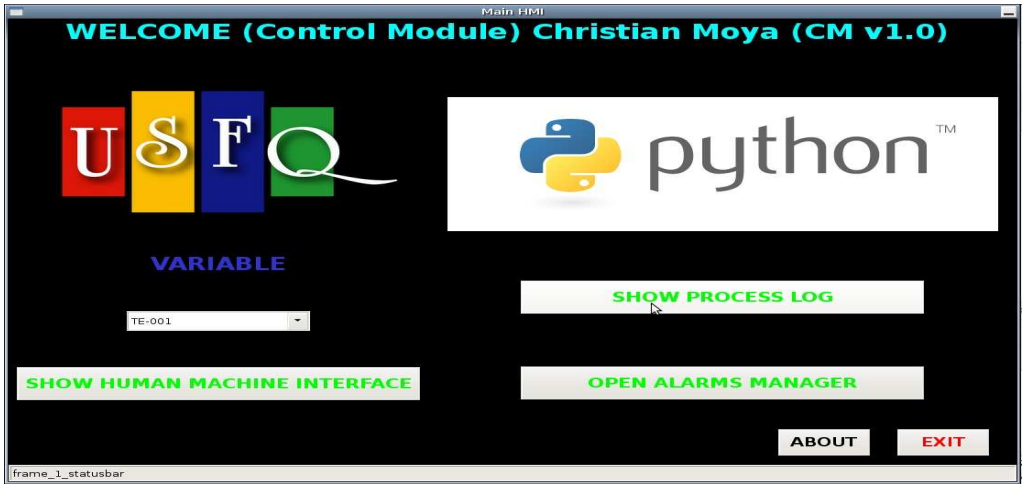

Figura A.15: Interfaz Hombre-Máquina Principal del Proceso

B

Programas de CM

Los programas dentro de CM se dividen en tres grandes grupos que son: CMHMIstation, CMProcessAdmin y CMScadaServer.

Programas dentro de CMHMIstation

- Carpeta GUI: Los programas dentro de esta carpeta tienen la siguiente nomenclatura: **CMHMIstXXXXGUIvXX.py** si es un programa para una interfaz gráfica de usuario o **CMHMIstXXXXHMIvXX.py** si es un programa para desarrollar una interfaz Hombre-Máquina.
- Carpeta MODULES: Los programas dentro de esta carpeta tienen la siguiente nomenclatura: **CMHMIstXXXXXmodvXX.py**

Programas dentro de CMProcessAdmin

- Carpeta GUI: Los programas dentro de esta carpeta tienen la siguiente nomenclatura: **CMPAXXXXGUIvXX.py** si es un programa para una interfaz gráfica de usuario o **CMPAXXXXHMIvXX.py** si es un programa para desarrollar una interfaz Hombre-Máquina.
- Carpeta MODULES: Los programas dentro de esta carpeta tienen la siguiente nomenclatura: **CMPAXXXXXmodvXX.py**

Programas dentro de CMScadaServer

- Carpeta GUI: Los programas dentro de esta carpeta tienen la siguiente nomenclatura: **CMSSXXXXGUIvXX.py** si es un programa para una interfaz gráfica de usuario o **CMSSXXXXHMIvXX.py** si es un programa para desarrollar una interfaz Hombre-Máquina.
- Carpeta MODULES: Los programas dentro de esta carpeta tienen la siguiente nomenclatura: **CMSSXXXXXmodvXX.py**