Universidad Central “Marta Abreu” de Las Villas

Facultad de Ingeniería Eléctrica

Departamento de Automática y Sistemas Computacionales

TRABAJO DE DIPLOMA

Módulo de funciones genéricas que faciliten la creación de aplicaciones SCADA

Autor: Yaidel Sánchez Reyes

Tutor: Msc. José Omar Padrón Ramos

Santa Clara

2013

"Año 55 de la Revolución"

Universidad Central “Marta Abreu” de Las Villas

Facultad de Ingeniería Eléctrica

Departamento de Automática y Sistemas Computacionales

TRABAJO DE DIPLOMA

Módulo de funciones genéricas que faciliten la creación de aplicaciones SCADA

Autor: Yaidel Sánchez Reyes

Email: yaidel@uclv.edu.cu

Tutor: Msc. José Omar Padrón Ramos

Dpto. de Automática, Facultad de Ing. Eléctrica, UCLV

Email: jpadron@uclv.edu.cu

Santa Clara

2013

"Año 55 de la Revolución"

Hago constar que el presente trabajo de diploma fue realizado en la Universidad Central "Marta Abreu" de Las Villas como parte de la culminación de estudios de la especialidad de Ingeniería en Automática, autorizando a que el mismo sea utilizado por la Institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos, ni publicados sin autorización de la Universidad.

_____

Firma del Autor

Los abajo firmantes certificamos que el presente trabajo ha sido realizado según acuerdo de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____ _____

Firma del Autor

Firma del Jefe de Departamento donde se defiende el trabajo

_____

Firma del Responsable de Información Científico-Técnica

PENSAMIENTO

"Gran parte de las dificultades por las que atraviesa el mundo se debe a que los ignorantes están completamente seguros, y los inteligentes llenos de dudas"

Russell, Bertrand

DEDICATORIA

A mis padres, Rebeca y José Carlos por ser las personas más importantes en mi vida, por tanto cariño, apoyo y sobre todo porque su prioridad fue educarme, brindándome amor y fuerza de voluntad para alcanzar metas superiores.

A mi bisabuela Cuca y mi abuela Xiomara a las que quiero mucho.

A mi tíos Roberto y Raulito, ellos han sido mi principal guía para mi vida profesional.

AGRADECIMIENTOS

A mi tutor José Omar Padrón Ramos por su disponibilidad para colaborar aún en los momentos más difíciles, siempre con entusiasmo y dedicación.

A mis compañeros de aula por su amistad y compañerismo, demostrado durante los últimos cinco años, donde hemos compartido grandes momentos.

A mis amigos, en general por su preocupación y sus buenos consejos.

A mi familia, de todo corazón por el apoyo incondicional que hasta el final me han brindado para poder llegar hasta aquí.

A todas las personas que de una manera u otra incidieron en el transcurso de mis estudios y que han colaborado con la culminación de este trabajo.

A todos, Muchas gracias.

TAREA TÉCNICA

1. Diagnóstico sobre los sistemas SCADA y los lenguajes de programación que estos utilizan.
2. Sistematización de fundamentos teóricos que permitan el estudio del lenguaje Visual Basic Scripts.
3. Diseño y codificación de las funciones para scripts de aplicaciones SCADA.
4. Evaluación de la efectividad con pruebas simuladas y reales en la plataforma Movicon.
5. Análisis económico del trabajo.

Esta página debe estar firmada por el estudiante y por el tutor, expresión del compromiso entre ambos.

Firma del Autor

Firma del Tutor

RESUMEN

Los sistemas SCADA están presentes en la industria moderna y en los servicios, no se concibe un proceso automatizado que no posea un mínimo nivel de supervisión. Estos programas complementan y potencian sus funcionalidades con editores y compiladores de código para crear funciones programadas en lenguajes de alto nivel. Pero pese al gran auge que han tenido los sistemas de supervisión, en la actualidad no existe de forma compartida o socializada ninguna biblioteca o paquetes de códigos genéricos que agilicen la programación y aumenten la funcionalidad de las aplicaciones SCADA. En el presente trabajo de diploma se trata de revertir en alguna medida esta situación proponiendo como objetivo general crear varios módulos de funciones genéricas para facilitar el desarrollo de los sistemas supervisorios. Para lograr esta meta se realizó la programación de algunos módulos que pudieran ser necesarios en una aplicación SCADA genérica. Dichos módulos se programaron en el lenguaje Visual Basic Script y las pruebas se realizaron en la plataforma Movicon v.11.3, obteniéndose los resultados esperados en todas las pruebas realizadas.

TABLA DE CONTENIDOS

INTRODUCCIÓN

Un sistema SCADA (*Supervisory Control And Data Acquisition*) es un software para computadoras o estaciones de visualización de planta, que permite controlar y supervisar procesos industriales a distancia, facilita la retroalimentación en tiempo real con los dispositivos de campo como sensores y actuadores, provee toda la información que se genera en el proceso productivo, como: supervisión, control de la calidad, control de producción, almacenamiento de datos y permite su gestión e intervención. Muchas de las acciones que se llevan a cabo en un sistema SCADA son procesadas o generadas mediante funciones codificadas en un lenguaje de programación de alto nivel, lo cual le confiere una potencia muy elevada y una gran versatilidad al sistema. Los utilitarios para la programación en lenguajes de alto nivel que posee un sistema SCADA suelen estar formados por dos programas: el Editor y el Motor de Ejecución (este incorpora un compilador). A través del primero se crea el código de las aplicaciones, aprovechando también la incorporación de *frameworks*, macros, sublenguajes y ayudas disponibles; con el segundo se realiza el proceso de compilación para obtener el fichero de ejecución continua tras el arranque. Entre los lenguajes de uso general que se utilizan para el desarrollo de sistemas SCADA se puede encontrar C, Pascal y Visual Basic.

El proceso de configuración y programación de un sistema SCADA se realiza en un tiempo finito determinado por diversos factores como: la cantidad de variables a manejar, el tamaño y la cantidad de área del proceso productivo, la rigurosidad del tratamiento a las variables en las bases de datos, la cantidad de informes y reportes, la complejidad en el tratamiento de la alarmas, entre otros. Por ello, el programador de estos sistemas debe poseer conocimientos sólidos y dominar la plataforma que seleccione, pero aun así, existen muchas acciones que deben ser programadas, pues no se encuentran de forma explícita en la plataforma y para ello se auxilia de segmentos de códigos realizados en algún lenguaje de programación, a estos códigos o pequeños módulos de software se les llama *scripts*. Los *scripts* en un sistema SCADA son indispensables para efectuar cálculos, funciones o personalizar funcionalidades complejas, tales como, acciones para la conexión a las bases de datos, generación de reportes no ordinarios, gestión de periféricos especiales,

comunicación con aplicaciones externas, tratamiento de variables calculadas y planificación de tareas.

Sucede que en muchas ocasiones el programador de sistemas SCADA pierde mucho tiempo en crear determinada funcionalidad a través de un *script*, que de existir en forma de alguna función de biblioteca u otro modo, le ahorraría un tiempo valioso, ya que generalmente este tipo de proyecto se realiza con requisitos de tiempo específicos y en coordinación con los niveles de automatización inferiores.

Ante la importancia y el costo en tiempo que revisten estos módulos de software para el programador de sistemas SCADA, surge el siguiente problema científico:

¿Existen en la actualidad bibliotecas disponibles, paquetes o un algún conjunto específico de funciones de *scripts* que le permitan a los programadores de sistemas SCADA acortar tiempo de trabajo y utilizar un mayor nivel de abstracción tal y como se hace para la creación de un software de gestión?

¿Cómo contribuir a la reducción del tiempo y la funcionalidad en la creación de aplicaciones SCADA, a través de la programación de *scripts* de código estructurado y funciones relacionadas?

Para brindar solución de esta problemática se elaboró el siguiente **objetivo general**:

- Crear un conjunto de funciones en Visual Basic Script que faciliten el desarrollo de aplicaciones SCADA para plataformas que utilicen este lenguaje de programación.

Este a su vez se desglosa en los siguientes **objetivos específicos**:

- Analizar los fundamentos sobre sistemas SCADA y los lenguajes de programación utilizados para estos.
- Definir necesidades de módulos de programación que pueda necesitar un sistema SCADA.

- Diseñar y programar algunos de los módulos definidos que tributen a necesidades comunes a todos los sistemas SCADA.
- Probar las funciones realizadas en una plataforma SCADA.

Este trabajo pretende contribuir a la programación de aplicaciones SCADA, así como, facilitar el accionar del personal implicado en su creación, mejorando el tiempo de entrega de los proyectos y la planificación de empresas o desarrolladores. También intenta dar soluciones a problemáticas modernas vinculadas con la adquisición de software de gran complejidad y valor cuya adquisición se dificulta en el país. Con la creación de funciones *script* se reduce el tiempo de entrega y ahorro de recursos en el desarrollo de sistemas SCADA, que pueden ser determinantes en el costo de un proyecto de automatización. Con el cumplimiento de los objetivos trazados se incentivará la realización de estudios y análisis comparativos, que en función de la disponibilidad de los recursos, logren ajustar las funciones para su mejor utilización teniendo en cuenta los resultados prácticos alcanzados y así continuar el desarrollo de nuevas funciones.

Los resultados de esta investigación tendrán una aplicación práctica de gran trascendencia para especialistas en el campo de la automática, diseñadores y programadores de sistemas SCADA. El producto final de este trabajo podrá ser aplicable a plataformas SCADA compatibles con el lenguaje Visual Basic Scripts.

Organización del informe:

El informe de la investigación se estructurará en introducción, capitulario, conclusiones, referencias bibliográficas y anexos.

El capitulario está compuesto de la siguiente forma:

Capitulario:

CAPÍTULO I: Se elabora un marco teórico sobre los sistemas SCADA y los lenguajes de programación que estos utilizan.

CAPÍTULO II: Se explica la necesidad de realización de las funciones, el diseño y codificación.

CAPÍTULO III: Se describen las pruebas realizadas en una plataforma SCADA, de los *scripts* creados. Se expresan los resultados y se demuestra la efectividad.

Capítulo 1. ANÁLISIS BIBLIOGRÁFICO SOBRE LOS SISTEMAS SCADA Y LOS LENGUAJES DE PROGRAMACIÓN UTILIZADOS POR ESTOS

1.1. Introducción al capítulo.

El presente capítulo aborda el material teórico referente a los Sistemas SCADA donde se encuentran aspectos generales como: breve reseña, definición general, componentes de hardware, características, aplicaciones, ventajas y desventajas. Se muestran las plataformas que más se utilizan en el mundo y sus aplicaciones. Provee además todos los elementos a tener en cuenta para el diseño de funciones genéricas para facilitar el desarrollo de aplicaciones SCADA.

1.2. Breve reseña de los sistemas SCADA.

Según Castellanos (2010) los primeros sistemas SCADA eran simples programas que proporcionaban reportes periódicos de las variables de campo, muestreando el estado de la planta o proceso a controlar desde ubicaciones generalmente remotas, en muchos casos lo que se hacía era imprimir o registrar en papel la información de las variables del estado del proceso, llevando un histórico de los eventos que ocurrían durante la operación la planta. Estos sistemas carecían de herramientas y funciones, lo cual los hacía incapaces de manejar grandes procesos industriales. La visión del operador en el proceso estaba basada en instrumentos y señalizaciones lumínicas, montadas en paneles de indicadores y alarmas.

Con el desarrollo de la tecnología, los ordenadores empezaron a aplicarse en el control industrial, permitiendo realizar tareas de retención y almacenamiento de datos, generación de comandos de control, y una nueva función muy importante: la presentación de la información sobre una pantalla, que en aquel entonces eran monocromáticas. Muchas empresas, conociendo la necesidad y lo rápido que avanzaba el desarrollo de los ordenadores, fueron realizando programas de aplicación específicos para atender requisitos de algún proyecto en particular. Así aparecieron los pequeños sistemas SCADA nacidos de

empresas desarrolladoras de software, constituyendo una nueva experiencia para muchas de ellas (Castellanos 2010).

La mayoría de los sistemas SCADA modernos que son instalados, constituyen parte integral de la estructura de dirección y gerencia de cualquier planta. Estos sistemas ya no son vistos por la gerencia simplemente como herramientas operacionales para la supervisión y el control automático, sino como un recurso importante de información corporativa, sin el cual sería imposible administrar la empresa.

1.3. Definición general de SCADA.

El término SCADA es el acrónimo de *Supervisory Control And Data Acquisition* (Supervisión, Control y Adquisición de Datos), es una aplicación de software diseñada con la finalidad de controlar y supervisar datos a distancia, los cuales se basan en la adquisición de variables de los procesos remotos utilizando las herramientas de comunicación necesarias en cada caso (Meza 2007).

Un sistema SCADA puede controlar, monitorear y supervisar desde un centro de control los procesos de estaciones remotas distantes, empleando diversos tipos de enlaces de comunicaciones, como: sistema satelital, red de microondas, radiocomunicaciones, fibra óptica y telefonía celular. Proveen toda la información que se genera en el proceso productivo a diversos usuarios, tanto del mismo nivel como de otros niveles superiores dentro de la empresa (Penin 2007).

Existen diferentes opiniones sobre los componentes de un sistema SCADA pero según Lakhoua (2009) de forma general se puede decir que está compuesto por:

- Una Interfaz hombre-máquina (HMI) que se encarga de presentar los datos del proceso a un operador humano, y a través de ella el operador supervisa y controla el proceso.
- Un sistema de control, encargado de la adquisición de datos del proceso y el envío de comandos para el proceso.

- Unidades Terminales Remotas (RTU) reciben las señales directamente de los sensores de campo y a su vez comandan a los actuadores y demás elementos de control final.
- La infraestructura de comunicación que conecta el sistema de control con la RTU.

Para instalar y aprovechar al máximo las potencialidades que brinda un sistema SCADA se deben cumplir ciertas condiciones (Lozano 2009):

1. Deben ser sistemas de arquitectura abierta (capaces de adaptarse según las necesidades de la entidad).
2. Deben comunicar con facilidad al usuario con el equipamiento de la planta y con el resto de la entidad (redes locales y de gestión).
3. Deben ser programas sencillos de instalar, sin excesivas exigencias de hardware.

1.4. Arquitectura de los SCADA.

Los sistemas SCADA se han desarrollado con el paso de los años. Durante este desarrollo se han podido distinguir tres etapas fundamentales:

1. Monolítico (Primera Generación): Cuando los primeros sistemas SCADA fueron desarrollados, las redes generalmente no existían, por lo que carecían de comunicación con otros sistemas. Como consecuencia los sistemas SCADA eran independientes.
2. Distribuido (Segunda Generación): Aprovechó los progresos y mejoras en la miniaturización del sistema y la tecnología local con establecimiento de una red de área local (LAN) para distribuir el proceso a través de sistemas múltiples. Las estaciones múltiples tenían una función específica cada una, y fueron conectadas con una red LAN compartiendo la información unas con otras en tiempo real. Estas estaciones eran típicamente mini-ordenadores, más pequeños y menos costosos que los procesadores de primera generación.
3. En Red (*Networked*) (Tercera Generación): La tercera y actual generación de las arquitecturas de sistemas SCADA se relaciona de forma muy cercana con la de la

segunda generación, siendo la diferencia primaria que esta última presenta una arquitectura de sistema abierto, siendo la segunda generación de ambiente propietario. La mejora principal en la tercera generación es la de abrir la arquitectura del sistema, de utilizar estándares y protocolos abiertos y de permitir distribuir la funcionalidad de supervisión a través de una red WAN y no de una red LAN.

1.5. Módulos software de los SCADA.

Los sistemas de supervisión y adquisición de datos contienen los siguientes módulos de software (Vidal. 2002; Janus 2006):

1. Configuración: Permite al usuario definir el entorno de trabajo de su SCADA, adaptándolo a la aplicación particular que se desea desarrollar.
2. Interfaz Gráfica del Operador: Proporciona al operador las funciones de control y supervisión de la planta. El proceso se representa mediante sinópticos gráficos (HMI).
3. Módulo de Proceso: Ejecuta las acciones de mando pre-programadas a partir de los valores actuales de variables leídas. La programación se realiza por medio de bloques de programa en lenguajes como C o Basic.
4. Gestión de Archivo de Datos: Se encarga del almacenamiento y procesado ordenado de los datos, de forma que otra aplicación o dispositivo pueda tener acceso a ellos.
5. Comunicación: Se encarga de la transferencia de información entre la planta y la arquitectura hardware que soporta el SCADA, y entre ésta y el resto de elementos informáticos de gestión.

1.6. Componentes Hardware.

Los componentes de hardware están principalmente compuestos por la MTU y la RTU.

1.6.1. La MTU.

El término MTU "Estación Maestra" se refiere a los servidores y el software, encargados de la comunicación con el equipamiento de campo, en los que se encuentra el software HMI

corriendo para las estaciones de trabajo en el cuarto de control, o en cualquier otro lado. En un sistema SCADA pequeño, la estación maestra puede estar en un solo computador. A gran escala, en los sistemas SCADA la estación maestra puede incluir muchos servidores, aplicaciones de software distribuido, y sitios de recuperación de desastres (Chacon, Dijort et al. 2001).

La parte más visible de un sistema SCADA es la estación central o MTU. Este es el "centro neurálgico" del sistema, y es el componente del cual el personal de operaciones se valdrá para observar un modelo simulado de la mayoría de la planta.

Las funciones principales de la MTU de un SCADA son:

- Adquisición de datos: Recolección de datos de los RTU.
- Tendencias: Salvar los datos en una base de datos, y ponerlos a disposición de los operadores en forma de gráficos.
- Procesamiento de Alarmas: Analizar los datos recogidos de los RTU para ver si han ocurrido condiciones anormales y alertar al personal de operaciones sobre las mismas.
- Control: Control a lazo cerrado e iniciados por operador.
- Visualizaciones: Gráficos del equipamiento actualizado para reflejar datos del campo.
- Informes: La mayoría de los sistemas SCADA tienen un ordenador dedicado a la producción de reportes conectado en red con el principal.
- Mantenimiento del Sistema Espejo (*Mirror*): Es decir, mantener un sistema idéntico con una capacidad segura de asumir el control inmediatamente si el principal falla.
- Interfaces con otros sistemas: Transferencia de datos hacia y desde otros sistemas corporativos para el procesamiento de órdenes de trabajo, de compra y actualización de bases de datos.
- Seguridad: Control de acceso a los distintos componentes del sistema.
- Administración de la red: Monitoreo de la red de comunicaciones.
- Administración de la Base de datos: Agregar nuevas estaciones, puntos, gráficos, puntos de cambio de alarmas y en general reconfigurar el sistema.

- Aplicaciones especiales: Casi todos los sistemas SCADA tendrán cierto software de aplicación especial, asociado generalmente al monitoreo y al control de la planta (Chacon, Dijort et al. 2001; Montero, Barrantes et al. 2004; Castellanos 2008).

1.6.2. La RTU.

Las unidades terminales remotas consisten en una pequeña y robusta computadora que almacena datos y los transmite a la terminal maestra para que esta controle los instrumentos. Es una unidad independiente (*stand-alone*) de adquisición y control de datos.

Su función es controlar el equipamiento de proceso en el sitio remoto, adquirir datos del mismo y transferirlos al sistema central SCADA. Hay dos tipos básicos de RTU de un solo módulo (*single boards*) compactos, que contienen todas las entradas de datos en una sola tarjeta. Una RTU *single board* tiene normalmente E/S fijas y los "modulares" que tienen un módulo CPU separado y pueden tener otros módulos agregados, normalmente enchufándolos en una placa común (similar a una PC con una placa madre (*Motherboard*) donde se montan el procesador y los periféricos) (Montero, Barrantes et al. 2004).

Características principales de la RTU:

- Comunicaciones a través de la red telefónica fija y móvil, radio enlaces, líneas dedicadas, buses de campo.
- Adquisición y mando (señales digitales y analógicas, conteos).
- Capacidad: entre 280 y 700 variables (según las aplicaciones).
- Procesamientos y automatismos parametrizables.
- Almacenamiento de datos a largo plazo (alarmas, medidas, conteos, informes).
- Alerta hacia estaciones maestras, buscapersonas y teléfonos móviles.
- Módulos especializados (automatización y gestión de las estaciones de elevación).
- Enlaces entre instalaciones (entre remota y remota, entre remotas y módulos).
- Compatibilidad con otros productos (autómatas programables, analizadores, controladores, medidores, ordenadores de supervisión) (Montero, Barrantes et al. 2004).

El hardware de una RTU según varios autores (Montero, Barrantes et al. 2004; Bailey and Wright 2006). se compone de los siguientes componentes:

- CPU y memoria volátil (RAM).
- Memoria no volátil para grabar programas y datos.
- Capacidad de comunicaciones a través de puertos seriales o a veces con módem incorporado.
- Fuente de alimentación segura con salvaguardia de batería.
- Perro Guardián (*Watchdog timer*): asegura reiniciar el RTU si algo falla.
- Protección eléctrica contra fluctuaciones en la tensión.
- Interfaces de entrada-salida ED/SD/EA/SA.
- Reloj de tiempo real.

En el diagrama (Fig.1.1) aparecen los componentes anteriormente mencionados (no se muestra el *Watchdog timer* ni el Reloj de tiempo real):

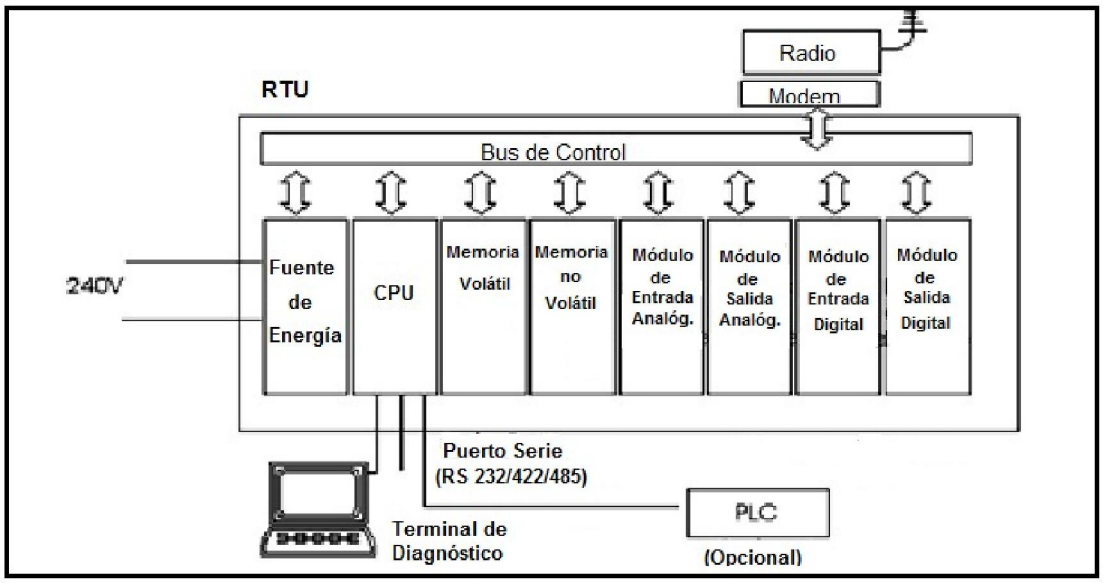


Fig. 1.1: Diagrama general de una RTU.

1.7. Funciones de un sistema SCADA.

Dentro de las funciones básicas de un sistema SCADA se pueden encontrar las siguientes (Meza 2007):

- Supervisión remota de instalaciones y equipos: Permite al operador conocer el estado de desempeño de las instalaciones y los equipos alojados en la planta, lo que permite dirigir las tareas de mantenimiento y estadística de fallas.
- Control remoto de instalaciones y equipos: Mediante el sistema se puede activar o desactivar los equipos remotamente, por ejemplo: abrir válvulas, activar interruptores, prender motores, de manera automática y también manual. Es posible ajustar parámetros, valores de referencia y algoritmos de control
- Procesamiento de datos: El conjunto de datos adquiridos conforman la información que alimenta el sistema, esta información es procesada, analizada, y comparada con datos anteriores, y con datos de otros puntos de referencia, dando como resultado una información confiable y veraz.
- Visualización gráfica dinámica: El sistema es capaz de brindar imágenes en movimiento que representen el comportamiento del proceso, dándole al operador la impresión de estar presente en planta real. Estos gráficos también pueden corresponder a curvas de las señales analizadas en el tiempo.
- Generación de reportes: El sistema permite generar informes con datos estadísticos del proceso en un tiempo determinado por el operador.
- Representación de señales de alarma: A través de las señales de alarma se logra alertar al operador frente a una falla o la presencia de una condición perjudicial o fuera de lo aceptable. Estas señales pueden ser tanto visuales como sonoras.
- Almacenamiento de información histórica: Se cuenta con la opción de almacenar los datos adquiridos, esta información puede analizarse posteriormente, el tiempo de almacenamiento dependerá del operador o del autor del programa.
- Programación de eventos: Está referido a la posibilidad de implementar subprogramas que brinden automáticamente reportes, estadísticas, gráfica de curvas, activación de tareas automáticas, entre otros.

1.8. Ventajas y desventajas de los sistemas SCADA.

Los sistemas SCADA proporcionan diversas ventajas y desventajas las que se reflejan a continuación:

De acuerdo con Castellanos (2008), las principales ventajas de los sistemas SCADA son:

1. Reducción de los costos de producción, operación y mantenimiento.
2. Aumento de producción.
3. Diversificación de la producción.
4. Mejoramiento de la coordinación con el área de mantenimiento.
5. Se dispone de información precisa para efectos de estudio, análisis y estadística.
6. No se requiere de personal para realizar labores de lectura de las variables ya que estos son leídos y enviados a centros de cómputos a través de la red.
7. Sistema de medición más rápido y confiable.

Las principales desventajas de los sistemas SCADA son:

1. Se requiere de una red industrial fiable, pues resultaría crítico no contar con la misma.
2. Alto costo inicial, por concepto de adquisición de los equipos e implantación del sistema acorde a las necesidades y requisitos exigidos.
3. Se requiere además realizar gastos en conexión a la red de datos.

1.9. 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 fábricas, pero donde más se utilizan es en entornos industriales complejos, o en procesos industriales que cubren un área geográfica amplia, 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 procesos y servicios como:

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

1.10. Principales plataformas SCADA.

Algunos de los principales software para la implementación de SCADA en el mundo son:

- Ø USDATA de Factory Link 7.
- Ø Lookout, de National Instrument.
- Ø WinCC, de Siemens.
- Ø SYSMAC SCS, de OMROM.
- Ø Paradym-31, de Advantech.
- Ø Virgo 2000, de AlterSys Inc.
- Ø WizFactory, de eMation.
- Ø Cimplicity, de GE Fanuc.
- Ø Genesis32, de Iconics.
- Ø Intouch, de Wonderware
- Ø RSView32, de Rockwell
- Ø CITECT, de Schneider Electric

Entre los más populares y más usados se pueden mencionar al CITECT de Schneider Electric, WinCC de Siemens e INTOUCH de Wonderware.

CITECT es una empresa de desarrollo de software especializada en la industria de la automatización y de control. Los principales productos de software desarrollados por la compañía incluyen a: CITECTSCADA, CITECT SCADA Reports y Ampla. CITECTSCADA es un paquete de software HMI/SCADA con controladores OPC del proveedor o sus controladores nativos propios. Contiene una gran colección de símbolos de

equipos industriales para la elaboración de los escenarios de aplicaciones hechas por CITECT con HMI/GUI reconocido por CITECT Gráficos Builder y una lógica de la aplicación en tiempo de ejecución, expresadas en el lenguaje de programación CICODE. La estructura y la sintaxis de Cicode es muy similar a la del lenguaje de programación Pascal, la principal diferencia es que no incluye punteros y conceptos asociados. CITECT ofrece una API de programación rica que incluye construcciones sofisticadas de programación (Schneider-Electric 2012).

WinCC, es un potente sistema HMI (*Human Machine Interface*) que se utiliza bajo Microsoft Windows. El control sobre el proceso lo tiene el autómata, por un lado hay una comunicación entre WinCC y el operador, y por otro lado entre WinCC y los autómatas programables. Con este sistema se visualiza el proceso y se programa la interfaz gráfica de usuario para el operador. Permite que el operador observe y monitoree el proceso, mediante la visualización gráfica en la pantalla (Siemens 2005).

Sistemas básicos de WinCC:

- Sistema de gráficos
- Sistema de avisos
- Sistema de archivos
- Sistema de informes
- Comunicación
- Administración de usuario

InTouch, ofrece funciones de visualización gráfica que llevan sus capacidades de gestión de operaciones, control y optimización a un nivel completamente nuevo. Lo que ahora se conoce en la industria como HMI comenzó hace más de veinte años con el software InTouch. Ningún otro en el mercado puede compararse al InTouch en términos de innovación, integridad de arquitectura, conectividad e integración de dispositivos, ruta de migración de versiones de software sin interrupciones y facilidad de uso. Esto se traduce en sistemas basados en estándares que permiten incrementar al máximo la productividad,

optimizar la efectividad del usuario, mejorar la calidad y reducir los costos operacionales, de desarrollo y de mantenimiento (Wonderware 2013).

Beneficios:

- Facilidad de uso que le permite a desarrolladores y operarios ser más productivos de manera simple y rápida.
- Gran integración de dispositivos y conectividad a prácticamente todos los dispositivos y sistemas.
- Sus capacidades de representación gráfica y la interacción con sus operaciones permiten entregar la información correcta a las personas correctas en el momento correcto.
- Migración de versiones de software sin interrupción, lo que significa que la inversión en sus aplicaciones HMI está protegida.

Capacidades:

- Gráficos de resolución independiente y símbolos inteligentes que visualmente dan vida a su instalación directamente en la pantalla de su computadora.
- Sofisticado sistema de scripting para extender y personalizar aplicaciones en función de sus necesidades específicas.
- Alarmas distribuidas en tiempo real con visualización histórica para su análisis
- Graficado de tendencias históricas integradas y en tiempo real.
- Integración con controles Microsoft ActiveX y controles .NET.
- Librería extensible con más de 500 de objetos y gráficos prediseñados, "inteligentes" y personalizables.

1.11. Los Lenguajes de programación más utilizados en sistemas SCADA.

Los lenguajes de programación más utilizados en los sistemas SCADA generalmente son: C, Pascal y Visual Basic Script. Cada empresa desarrolladora se ha especializado en alguno de ellos; aunque VBScript, en comparación con los demás lenguajes de programación provee de un gran ahorro de licencias bajo coste por copia y esto rápidamente se compensa

con los costos de desarrollo mucho más altos y los costos de soporte. Además de presentar un entorno de desarrollo de programación es claramente más flexible que un entorno de desarrollo configuración, pero hay un coste significativo asociado con la programación que hace que sea una alternativa atractiva para HMI/SCADA.

Visual Basic Script.

Según refiere Jones (2004) el lenguaje Visual Basic Script (VBScript) es uno de los lenguajes de secuencias de comandos de Microsoft, que se asocian comúnmente a lado del servidor y del lado del cliente de aplicaciones web. Sin embargo, Microsoft ha abierto VBScript a los desarrolladores y VBScript ahora se puede encontrar en una variedad de aplicaciones. Muchas empresas desarrolladoras de sistemas SCADA han utilizado como su lenguaje nativo el VBScript, ya que proporciona un subconjunto significativo de la funcionalidad de Microsoft Visual Basic. VBScript está soportado en todas las plataformas de sistemas operativos de Microsoft como Windows CE, a diferencia de VBA (Visual Basic para Aplicaciones), que no puede soportar el Windows CE como entorno de ejecución.

De acuerdo con Childs (2003) VBScript es un lenguaje de programación que se observa como un sucesor de VBA (Visual Basic para Aplicaciones), aunque en realidad es su propia lengua. El lenguaje VBScript intenta equilibrar flexibilidad, capacidad y facilidad de uso. VBA es un subconjunto de Visual Basic que se ha desarrollado para automatizar aplicaciones de Microsoft Office, mientras que VBScript fue desarrollado originalmente para apoyar lado del servidor y del lado del cliente de aplicaciones web. Aunque VBScript y VBA proporcionan muchas de las mismas características, existen algunas diferencias entre ellos, principalmente debido a las aplicaciones que se han desarrollado para apoyar a cada uno. VBScript proporciona un fácil uso de entorno de desarrollo que configura los objetos predefinidos para apoyar una aplicación HMI/SCADA. Las aplicaciones pueden ser construidas rápidamente y son relativamente fáciles de soportar, incluso por alguien que no sea el desarrollador original.

1.12. Definición general de *Scripts*.

De acuerdo a lo expuesto por Jones (2004) Muchas de las acciones que se realizan como parte de la supervisión de un proceso industrial, por ejemplo: la manipulación y tratamiento de las variables, la personalización de características, la creación de comandos, la comunicación con terceros, y otras; no vienen predefinidas o con la posibilidad de ser configurables en las plataformas SCADA, pero estas acciones, a fin de cuentas, se realizan en la práctica. ¿Cómo se logra esto? Ello se logra a través de bloques de código realizados en algún lenguaje de programación de alto nivel. Estos bloques de código se denominan *scripts*.

Un *script*, según Jones (2004) es una secuencia de comandos o serie de comandos de ordenador que son ejecutados en una secuencia, son considerados también pequeños programas informáticos escritos en un lenguaje de alto nivel como VBScript o JavaScript.

1.13. Análisis del contexto actual en cuanto a los *scripts* en los sistemas SCADA.

Tomando en consideración toda la información anteriormente revisada, se puede llegar a varias conclusiones: actualmente no hay ningún conjunto de funciones específicas que sean diseñadas específicamente para utilizar en *scripts* de sistemas SCADAS. Hay muchos códigos en Visual Basic Script pero ninguno vinculado al tratamiento de la información en sistemas SCADA. Los códigos encontrados se pueden adaptar fácilmente, con un mínimo de cambios, al funcionamiento de sistemas supervisorios.

Con este análisis se verifica la necesidad de realizar el presente trabajo, o sea, la creación de un conjunto de funciones genéricas que faciliten el desarrollo de aplicaciones SCADA y acorten el tiempo de programación.

1.14. Herramientas utilizadas en el trabajo.

Plataforma de programación de sistemas SCADA “Movicon”.

Movicon**,** acrónimo de Monitoreo, Visión y Control, es una herramienta para compañías que trabajan en el campo de la automatización y el control de procesos que pertenece a la compañía italiana Progea que ha desarrollado productos software para la automatización industrial desde 1990 (Progea 2007).

Movicon permite la adquisición de datos a través de su comunicación con el PLC (controlador lógico programable), red y bus de campo, así como la configuración de herramientas y sensores. Los datos adquiridos son coleccionados dentro de una base de datos en tiempo real (RTDB) y luego están disponibles para todos los objetos y recursos para crear de forma animada, sinópticos, alarmas, recetas, gráficos y reportes.

Movicon v.11.3 presenta las siguientes características (Progea 2012):

- Sistema SCADA/ HMI para Win32.
- Trabaja bajo sistemas operativos de Windows.
- Es un sistema Cliente/Servidor de 32 bit.
- Contiene una amplia biblioteca de símbolos, objetos y gráficos.
- Totalmente compatible con de .NET
- OPC cliente y servidor.
- OPC-DA (Data Access.)
- OPC-AE (Alarma y Eventos.)
- Programación en Basic Script que es 100% compatible con VBA (*Visual Basic for Aplication*).
- Contiene editores de menú y cajas de diálogo.
- Posibilidad de usar lenguaje de PLC en las lógicas que él dispone.
- Tiene un administrador de alarmas.
- Soporte de tecnología ActiveX.
- Soporte de OLE2, ODBC, DDE, DAO/ADO, SQL y OPC.
- Red de Cliente/Servidor de TCP/IP.
- Objetos PID integrados.
- Gráficos y hojas de trabajo.

- Un depurador (*debugger*) integrado.
- Administración de estadísticas de evento o producción.

Se compone de un ambiente formado por un editor de objetos, que en conjunto de bibliotecas gráficas pueden implementar mímicos animados. Los mímicos también ofrecen una interfaz a VBScript, suministrando al programador eventos, métodos y propiedades. De esta manera el usuario puede implementar cualquier tipo de objeto gráfico a través de la manipulación de las funciones. En la figura 1.2 se pueden observar algunos de los símbolos de su amplia biblioteca estos se pueden usar para la conformación de mímicos empleando este software.

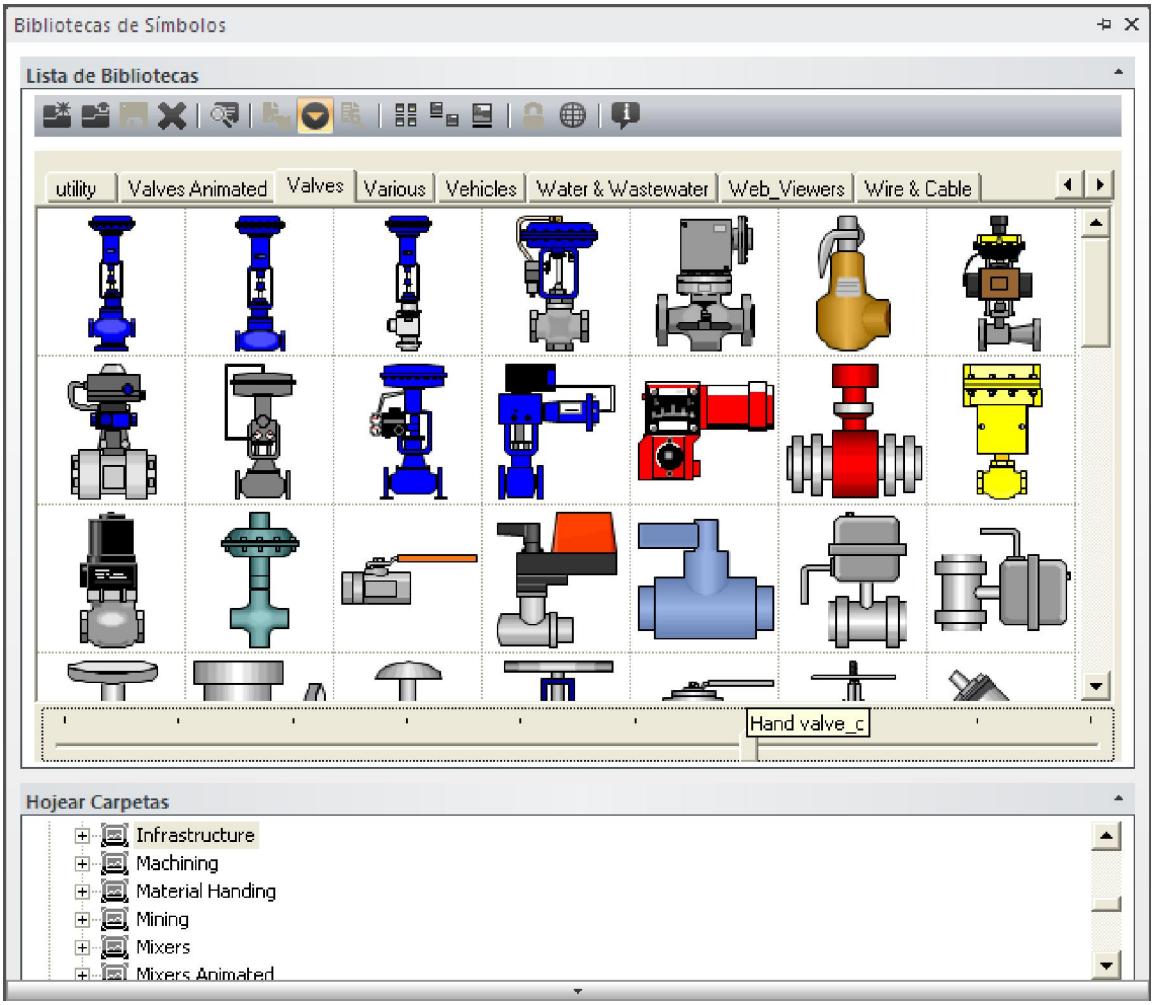


Fig. 1.2: Biblioteca de Símbolos de Movicon.

Movicon soporta dos tipos de elementos: símbolos y objetos. Los símbolos están organizados en librerías con clases que son expandibles. Los objetos son elementos vectoriales que tienen una función y pueden ser configurados con funciones de estilo y control. Presenta además un componente que implementa la interfaz gráfica para el diálogo con el operador del sistema, es una caja de diálogo que se utiliza para poder manipular los puntos de ajuste, ajustar datos, selecciones y opciones.

Para la evaluación de los resultados obtenidos en el Trabajo Diploma se utilizará la plataforma Movicon v.11.3 ya que su lenguaje nativo para la creación de *scripts* es el Visual Basic Script, además de ser compatible con .NET, bibliotecas a las que se hace referencia en el trabajo. Por otra parte esta plataforma es muy didáctica y a la vez funcional, siendo utilizada en varias asignaturas de la especialidad de Automática y por si esto fuera poco, durante años ha sido empleado por una gran cantidad de industrias tanto en Cuba como en el mundo.

Herramienta de programación gráfica "LabVIEW".

LabVIEW es una herramienta diseñada por la empresa norteamericana National Instruments especialmente para monitorear, controlar, automatizar y realizar cálculos complejos de señales analógicas y digitales capturadas a través de tarjetas de adquisición de datos, puertos serie y GPIBs (Buses de Intercambio de Propósito General). Es un lenguaje de programación de propósito general, como es el Lenguaje C o Basic, pero con la característica que es totalmente gráfico, facilitando de esta manera el entendimiento y manejo de dicho lenguaje para el diseñador y programador de aplicaciones tipo SCADA.

Incluye librerías para la adquisición, análisis, presentación y almacenamiento de datos, GPIB y puertos serie. Está basado en la programación modular, lo que permite crear tareas muy complicadas a partir de módulos o sub-módulos mucho más sencillos. Estos módulos pueden ser usados en otras tareas, lo cual permite una programación más rápida y provechosa.

Es un sistema abierto, en cuanto a que cualquier fabricante de tarjetas de adquisición de datos o instrumentos en general puede proporcionar el controlador de su producto en forma de VI dentro del entorno de LabVIEW (Hernández 2013).

.NET Framework.

Según Microsoft (2013) cada versión de.NET Framework contiene el *Common Language Runtime* (CLR), las bibliotecas de clases base y otras bibliotecas administradas. En este tema se describen las características clave de .NET Framework por versión, se proporciona las versiones del CLR subyacente y los entornos de desarrollo asociados, y se identifican las versiones que se instalan en el sistema operativo de Windows.

En la siguiente ilustración se resume el historial de versiones y se identifican las versiones que se instalan en Windows:

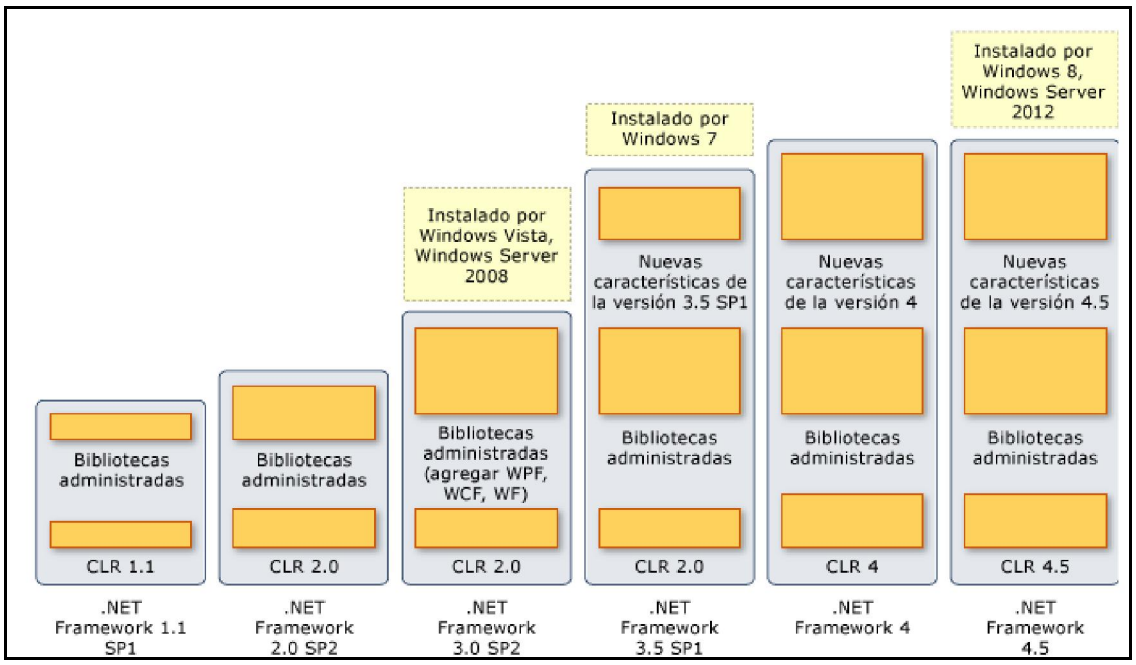


Fig.1.3 Historial de versiones .Net *Framework*.

Cada versión de .NET Framework contiene características de versiones anteriores e incorpora nuevas características. El CLR se identifica por su propio número de versión. Algunas versiones de .NET Framework incluyen una nueva versión de CLR, aunque otras

utilizan una versión anterior. Por ejemplo, .NET Framework 4 contiene la versión 4 de CLR, mientras que .NET Framework 3.5 incluye CLR 2.0. (No existe la versión 3 de CLR). Aunque .NET Framework 4.5 es una actualización en contexto de .NET Framework 4, el número de versión de CLR subyacente se denomina CLR 4.5.

En general, no debe desinstalar ninguna versión de .NET Framework que esté instalada en el equipo, porque una aplicación que use puede depender de una versión concreta y se puede interrumpir si se quita esa versión. Puede cargar varias versiones de .NET Framework en un único equipo simultáneamente, esto significa que puede instalar sin tener que desinstalar las versiones anteriores.

Historial de Versiones de .NET.

Las versiones 2.0, 3.0 y 3.5 de .NET Framework están compiladas con la misma versión del CLR (CLR 2.0). Estas versiones representan los sucesivos niveles de una única instalación. Cada versión se compila mejorado las versiones anteriores de .NET Framework. No es posible ejecutar las versiones 2.0, 3.0 y 3.5 en paralelo en un equipo. Al instalar .NET Framework 3.5 SP1, obtiene las capas 2.0 y 3.0 automáticamente. Sin embargo, .NET Framework 4 finaliza este enfoque de capas. A partir de .NET Framework 4, puede usar el hospedaje en paralelo en el mismo proceso para ejecutar varias versiones de CLR en un único proceso. Las aplicaciones que se compilaron para las versiones 2.0, 3.0 y 3.5, pueden todas ejecutarse en la versión 3.5, pero no funcionarán en la versión 4 o posterior.

.NET Framework 4.5 es una actualización en contexto que reemplaza a .NET Framework 4 en el equipo. Después de instalar esta actualización, las aplicaciones de .NET Framework 4 deben seguir ejecutándose sin requerir la recompilación. Sin embargo, como se han realizado ciertos cambios en .NET Framework, es posible que se deba modificar el código de la aplicación.

1.15. Consideraciones parciales del capítulo.

En el capítulo que concluye, se abordaron definiciones, características, arquitectura, aplicaciones, ventajas y desventajas de los Sistemas SCADA y además se mostraron las plataformas más utilizadas en el mundo. Con ayuda de esto se determinó la creación de un módulo de varias funciones programadas y utilizar el software Movicon v.11.3 para analizar los resultados del trabajo. También se mencionaron los lenguajes de programación más utilizados en el desarrollo de aplicaciones SCADA y se hace énfasis en Visual Basic Script que es el lenguaje sobre el cual se trabaja. Con estos datos en el siguiente capítulo, se describe cómo realizar la programación de los Script para el desarrollo de sistemas SCADA.

Capítulo 2. CODIFICACIÓN DE LOS MÓDULOS *SCRIPTS*

2.1. Introducción al capítulo.

En el presente capítulo se explica el análisis y la necesidad de la funcionalidad de los *scripts* de software diseñados y programados como parte del trabajo. Cada uno de ellos no es mero capricho del autor realizar, sino resultado de necesidades reales surgidas en el transcurso de proyectos SCADA anteriores en plantas reales, en los cuales se perdió mucho tiempo creando códigos de script para realizar acciones que no estaban incluidas en la plataforma y que bien pudieran ser de necesidad general para muchos especialistas que se dedican a esta rama. Se incluye también los métodos, el desarrollo matemático, los algoritmos y el software que forman parte del trabajo.

2.2. Módulos.

Según una valoración inicial y necesidades observadas en trabajos precedentes de programación de sistemas SCADA, se decidió implementar un conjunto de funcionalidades genéricas que pueden ser utilizadas, incluso modificadas al oficio, por quienes ejecutan estas tareas. Estas funciones seleccionadas son:

- Separación en bits individuales de una palabra (16 bits).
- Traducción de formato de un número y conversión del mismo, representado en Complemento a 2.
- Almacenamiento de datos de proceso o configuración en un archivo de texto.
- Lectura de datos de proceso o configuración desde un archivo de texto.
- Almacenamiento de valores de proceso y configuración en una base de datos de propósito general.

- Almacenamiento y transmisión de valores de proceso y configuración en formato XML.

2.3. Separación en bits individuales de una palabra (16 bits).

Generalmente las variables de alarma están representadas en un solo bit y no es necesidad hacerlo con un byte o una palabra para ahorrar memoria en los PLC. Los programadores utilizan frecuentemente la agrupación de las alarmas en palabras por lo cual, a la aplicación SCADA que monitorea los datos le es necesario separar los bits de la palabra para detectar el cambio de los mismos cuando se generan las alarmas.
Algunos sistemas SCADA, ya traen este tipo de tratamiento, pero otros no, por lo cual se hace necesario hacerlo a través de un script.
El procedimiento consiste en la conversión de un número entero (≤ 65535) a binario (16 bits) y se obtiene un arreglo con los resultados de cada uno de los dígitos del número binario.

Para saber exactamente la información real que viene en la palabra, se realizan las siguientes acciones:

1. Convertir el número a binario.
2. Separar sus bits.

Ver código en el Anexo 1.

2.4. Traducción de formato de un número y conversión del mismo, representado en Complemento a 2.

Los programadores de PLC, en muchas ocasiones realizan el tratamiento de los datos recibidos por los sensores en complemento a 2, para la representación de los valores negativos de los mismos. Una de las causas puede ser que la propia configuración de hardware del PLC así lo exija o que le sea más cómodo en determinado momento representar de esta forma el número.

Cómo algunos de los sistemas SCADA no tienen forma de conocer si el dato que les llega es un número normal o su complemento a 2, pero el programador sí conoce el formato del número, pues se hace necesario crear un script que haga la conversión del dato obtenido.

El valor que llega del PLC es un número en decimal, pero que a su vez es la representación de un número binario convertido a complemento a 2.

Para saber exactamente la información real que se recibe, se realizan las siguientes acciones:

1. Convertir el número a binario.
2. Separar sus bits.
3. Hallar el complemento a 1 (teniendo en cuenta el bit de signo).
4. Sumar 1 al complemento a 1.
5. Convertir a decimal (este paso no es necesario, solo es para verificar que se hizo bien la operación, el resultado que interesa se queda en el paso 4)

Ver código en el Anexo 2.

2.5. Almacenamiento de datos de proceso o configuración en un archivo de texto.

Para el desarrollo de aplicaciones SCADA es necesario que el programa que se realiza tenga la posibilidad de comunicación con aplicaciones que puedan ser de utilidad. Los archivos de texto o archivos con extensión **txt** tienen un formato plano, por lo cual pueden ser utilizados por cualquier aplicación sin un tratamiento complejo adicional. Por medio de este tipo de ficheros, utilizándolo como una especie de base de datos temporal, se pueden exportar variables, elementos de configuración u otro tipo de datos que se desee.

Para llevar a vías de hecho un intercambio de datos a través de ficheros de texto se hacen las siguientes acciones:

1. Crear en código un fichero de texto en una ubicación seleccionada o por selección del usuario.
2. Escribir los datos deseados del sistema SCADA en el fichero texto.

Después de esto, cualquier aplicación que necesite estos datos, puede leer los mismos desde el fichero escrito.

Proceso de escritura:

Para hacer el paso número 1, se crea primeramente el archivo de texto con la función *CreateObject* a la cual se le pasan los parámetros con la siguiente sintaxis:

CreateObject(servername.typename [, location]).

Con la función *OpenTextFile* se abre el archivo creado:

Object.OpenTextFile(filename[, iomode[, create[, format]]]).

Posteriormente para escribir en el archivo creado se utiliza la función WriteLine de la forma:

Debug.WriteLine([str1 [, str2 [, ... [, strN]]]]).

En caso que se desee incluir una o varias líneas en blanco en el archivo se utiliza la función *WriteBlankLines* de la forma:

Object.WriteBlankLines(lines).

Ver código en el Anexo 3.

2.6. Lectura de datos de proceso o configuración desde un archivo de texto.

En otras ocasiones el sistema SCADA puede necesitar importar datos desde ficheros de texto que le ha brindado otra aplicación. Para esto se crea un script que realiza las siguientes acciones.

1. Carga los datos deseados del fichero de texto según la ubicación de este.
2. Utiliza estos datos en el procesamiento de la información propia.

Proceso de lectura:

Para la lectura del archivo de texto ya creado se utilizan las funciones *Read, ReadLine* o *ReadAll* en consecuencia de la magnitud en la que se quiera leer.

Las mismas se utilizan de la forma:

- Para leer un número específicos de caracteres: ***object.Read(characters)***
- Para la lectura de una línea del texto: ***object.ReadLine()***

Para la lectura del texto completo: ***object.ReadAll()***

Ver código en el Anexo 4.

2.7. Almacenamiento de valores de proceso y configuración en una base de datos de propósito general.

Este script se realiza con la intención de almacenar en una base de datos, variables y otros indicativos que sean necesarios para la comunicación con otra aplicación. Las plataformas SCADA presentan la opción de crear una base de datos para el almacenamiento y chequeo de los datos, sin embargo estas bases de datos tienen una alta seguridad y se hace altamente complejo el acceso a estas desde otra aplicación.

El trabajo con este módulo consiste en, desde el mismo sistema SCADA, conectarse con una base de datos de propósito general, ejecutar consultas para almacenar los datos que se quiera y de esta forma otra aplicación podrá obtener los datos requeridos.

La Base de Datos se puede crear en cualquier gestor de propósito general (en el caso de las pruebas realizadas como parte del capítulo 3 se utilizó Microsoft Office Access) en una dirección determinada y con el nombre que se estime conveniente. Aquí serán guardados los datos y podrán ser obtenidos por otros programas.

Para la realización del *script* se necesita del auxilio de la plataforma .NET por lo que la plataforma SCADA que se utilice para ejecutarlo debe ser totalmente compatible con .NET.

Para el acceso al *framework* .NET se introduce la siguiente línea:

'#Language "WWB.NET"

La clase que contiene todos los comandos que serán utilizados es:

Imports System

Y de ella se derivan para el acceso y configuración de base de datos:

- *Imports System.Data*
- *Imports System.Data.OleDb*

- *Imports System.Data.Odbc*

Para la conexión y desconexión se crea una variable de tipo *ADODB.ConnectionClass* y para la ejecución de las consultas en SQL se crea la variable de tipo *ADODB.RecordClass*.

El programa principal (*main*) consta solo de la llamada de las sentencias de conexión y desconexión.

La sentencia de conexión de divide en los siguientes puntos:

- Se crea la conexión.
- Apertura de la conexión.
- Se chequea si hubo errores al abrir.
- Se almacena la sentencia de datos a guardar en la base de datos.
- Se ejecuta la sentencia para insertar los datos.

La sentencia de desconexión solo se basa en cerrar la conexión antes abierta.

Ver código en el Anexo 5.

2.8. Almacenamiento de valores de proceso y configuración en un archivo XML.

Un archivo XML, *e**X**tensible **M**arkup **L**anguage* deriva del lenguaje SGML y permite definir la gramática de lenguajes específicos para estructurar documentos grandes. A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones se deben comunicar entre sí o integrar información. Es conocido como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

Los archivos XML presentan las siguientes ventajas:

- Es extensible: Después de diseñado y puesto en producción, es posible extender XML con la adición de nuevas etiquetas, de modo que se pueda continuar utilizando sin complicación alguna.

- El analizador es un componente estándar, no es necesario crear un analizador específico para cada versión de lenguaje XML. Esto posibilita el empleo de cualquiera de los analizadores disponibles. De esta manera se evitan *bugs* y se acelera el desarrollo de aplicaciones.
- Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarla. Mejora la compatibilidad entre aplicaciones. Podemos comunicar aplicaciones de distintas plataformas, sin que importe el origen de los datos, es decir, podríamos tener una aplicación en Linux con una base de datos *Postgres* y comunicarla con otra aplicación en Windows y Base de Datos *MS-SQL Server*.
- Transformamos datos en información, pues se le añade un significado concreto y los asociamos a un contexto, con lo cual tenemos flexibilidad para estructurar documentos.

Para el desarrollo de este *script* se utiliza la plataforma .NET, esta es necesaria para el acceso a las clases:

- *Imports System.Xml*
- *Imports System.IO*
- *Imports System.Text*

Para la conversión de los valores de las variables a tipo cadena (*string*) se le asignan los valores de las variables reales a variables auxiliares.

Para iniciar el proceso de escritura se crea el objeto:

XmlWriterSettings

Seguidamente se crea el objeto (a este se le pasa la dirección en la que se guardará el archivo XML):

XmlWriter

El proceso de escritura se programa en los siguientes pasos:

- Abrir el documento creado.

- Inicializar escritura en el documento.
- Llamar individualmente a cada una de las variables que se escribirán en el texto y escribirlas convirtiéndolas a cadena (*string*).
- Finalizar escritura en el documento.
- Cerrar documento antes abierto.

Ver código en el Anexo 6.

2.9. Consideraciones parciales del capítulo.

En el capítulo que concluye, se realiza el análisis y la necesidad de la funcionalidad de las funciones de software diseñados y programados basados en necesidades reales surgidas anteriormente. Se muestra una descripción de cómo realizar la programación de los *scripts* para el desarrollo de sistemas SCADA. En la descripción se incluyen los métodos utilizados, desarrollo matemático y los algoritmos creados. La tarea antes propuesta de diseño y codificación de las funciones para scripts de aplicaciones SCADA se cumplen satisfactoriamente.

Capítulo 3. PRUEBAS Y RESULTADOS DE LOS *SCRIPTS*

3.1. Introducción al capítulo.

En el siguiente capítulo se describen las pruebas y resultados de la validación de los códigos creados en la plataforma SCADA seleccionada para ello, cumpliendo de esta forma con los objetivos trazados. Para estas pruebas también fue necesario utilizar otros programas auxiliares en función de adquirir datos generados por el sistema SCADA, cada uno de estos procesos son detallados en el transcurso del capítulo. Por último se realiza un análisis económico en términos aproximados lo cual permite redundar la importancia del trabajo desarrollado.

3.2. Contexto para probar los códigos (*scripts*).

Para las funcionalidades operativas de los *scripts* realizados como parte del trabajo, fue necesario crear un ambiente adecuado y técnicamente correcto en cuanto a hardware y software. Para ello se utilizaron las siguientes condiciones y equipos:

- Computadora Personal (procesador Dual Core a 2.2 GHz y 1MB de Cache; 2 GB de memoria RAM)
- Sistema Operativo Windows 8 Professional.
- Plataforma SCADA Movicon v.11.3, (con lenguaje nativo Visual Basic Script y totalmente compatible con la tecnología .NET).
- *Framework* .NET v.3.5.
- Programas creados en la plataforma LabVIEW versión 7.1.

3.3. Diseño de pruebas.

Para cada *script* creado se ha realizado un diseño de pruebas con el fin de evaluar su desempeño. Los diseños se describen de acuerdo al orden de los scripts y pueden ser más de uno por cada código.

3.3.1. Diseño para la separación en bits individuales de una palabra (16 bits).

- Crear un módulo *script* en un proyecto Movicon.
- Copiar el *script* para la lectura de una palabra de 16 bits.
- Crear un sinóptico para visualizar los resultados obtenidos.
- Escoger un número entero positivo cuya representación no exceda de 16 bits.
- Realizar el cálculo en papel y anotar el resultado esperado.
- Insertar el número escogido en la variable de entrada.
- Ejecutar el programa.
- Verificar que el resultado mostrado en Movicon es igual al resultado esperado.

3.3.2. Diseño para la traducción de formato de un número y conversión del mismo, representado en Complemento a 2.

- Crear un módulo de *script* en un proyecto Movicon
- Copiar el *script* del complemento a 2 para el *script* creado.
- Poner un valor entero cualquiera en la variable de entrada.
- Realizar el cálculo en papel y anotar el resultado con el mismo número de entrada.
- Ejecutar el *script*.
- Verificar que el resultado mostrado por el Movicon es el mismo que el del papel.

3.3.3. Diseño para el almacenamiento de datos de proceso o configuración en un archivo de texto.

- Crear un módulo *script* en un proyecto Movicon.
- Copiar el *script* para creación y escritura de un archivo de texto.
- Crear una pantalla donde se inserte el valor de la variable que se exportará hacia el archivo de texto.
- Definir una variable de tipo palabra (16 bits).
- Crear un programa en LabVIEW para leer datos de ficheros texto.
- Ejecutar el programa en Movicon.
- Abrir el archivo de texto y verificar que la variable o dato se guardó correctamente.
- Ejecutar el programa creado en LabView y verificar si la variable que se leyó es la correcta.

3.3.4. Diseño para la lectura de datos de proceso o configuración desde un archivo de texto.

- Crear un módulo *script* en un proyecto Movicon.
- Copiar en este módulo el *script* para la lectura de un archivo de texto.
- Crear una pantalla para la visualización de la variable que se importará al SCADA.
- Crear un programa en LabVIEW para escribir un valor numérico en un fichero texto.
- Ejecutar el programa creado en LabVIEW.
- Definir las variables o datos que se importarán al SCADA por medio del archivo de texto.
- Abrir el archivo de texto creado y verificar si la variable es la correcta.
- Ejecutar el programa en Movicon.
- Verificar en la pantalla si la variable se importó correctamente.

3.3.5. Diseño para el almacenamiento de valores de proceso y configuración en una base de datos de propósito general.

- Crear un módulo *script* en un proyecto Movicon.
- Copiar el *script* que guarda información en una base de datos.
- Crear una base de datos en Microsoft Office Access con una tabla de 5 columnas para datos enteros.
- Se crea un *script* adicional el cual va generando cada un segundo 5 variables aleatorias que serán guardadas en la base de datos.
- Crear un programa en LabVIEW que grafique los valores de las 5 variables, obtenidos a través de una consulta SQL a la base de datos.
- Ejecutar el programa en Movicon.
- Ejecutar el programa en LabVIEW.
- Verificar si en la base de datos se guardan las variables.
- Verificar si la gráfica está respondiendo correctamente a los cambios de las variables.

3.3.6. Diseño para el almacenamiento de valores de proceso y configuración en un archivo XML.

- Crear un módulo *script* en un proyecto Movicon.
- Copiar el *script* que guarda información en formato XML.
- Crear una pantalla para la visualización de las variables.
- Ejecutar el programa en Movicon.
- Abrir el fichero XML y verificar que las variables se guardaron correctamente.

3.4.1. Resultados para la separacion en bits individuales de una palabra (16bits).

Basado en el diseño de pruebas del sub-epígrafe **3.3.2** se realiza un nuevo proyecto en el cual se crea una nueva pantalla en Movicon que muestra un cuadro de texto para introducir el número entero positivo con representación en 16 bits, un botón para ejecutar el *script* y 16 cuadros de texto para mostrar cada uno de los valores (bits) obtenidos (Fig.3.2).

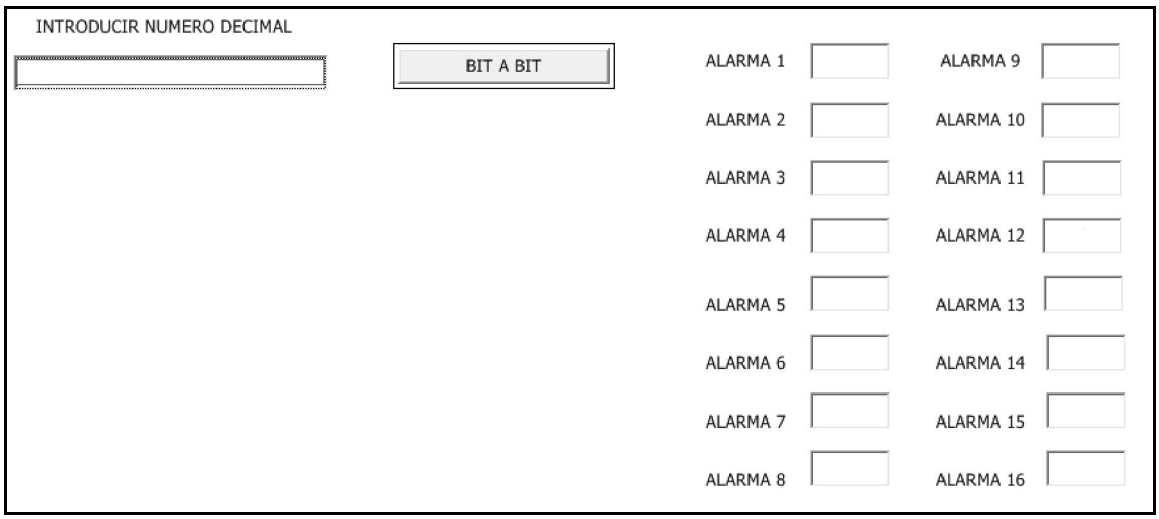


Fig.3.2 Pantalla de Movicon para la separación de una palabra (16 bits) en bits individuales.

El número escogido para la prueba es el 170. En el cálculo en el papel se lleva este número a binario de la forma:

170_d à 10101010_b

Los bits esperados para cada una de las alarmas se representan en la tabla 3.1.

Alarma 1 à 0	Alarma 9 à 0
Alarma 2 à 1	Alarma 10 à 0
Alarma 3 à 0	Alarma 11 à 0
Alarma 4 à 1	Alarma 12 à 0
Alarma 5 à 0	Alarma 13 à 0

Alarma 6 à 1	Alarma 14 à 0
Alarma 7 à 0	Alarma 15 à 0
Alarma 8 à 1	Alarma 16 à 0

Tabla.3.1 Representación de los bits esperados en la prueba de separación de una palabra (16 bits) en bits individuales.

Al ejecutar el programa en Movicon (Fig.3.3) se puede observar que el resultado de la prueba es satisfactorio, pues cada bit concuerda perfectamente con el esperado.

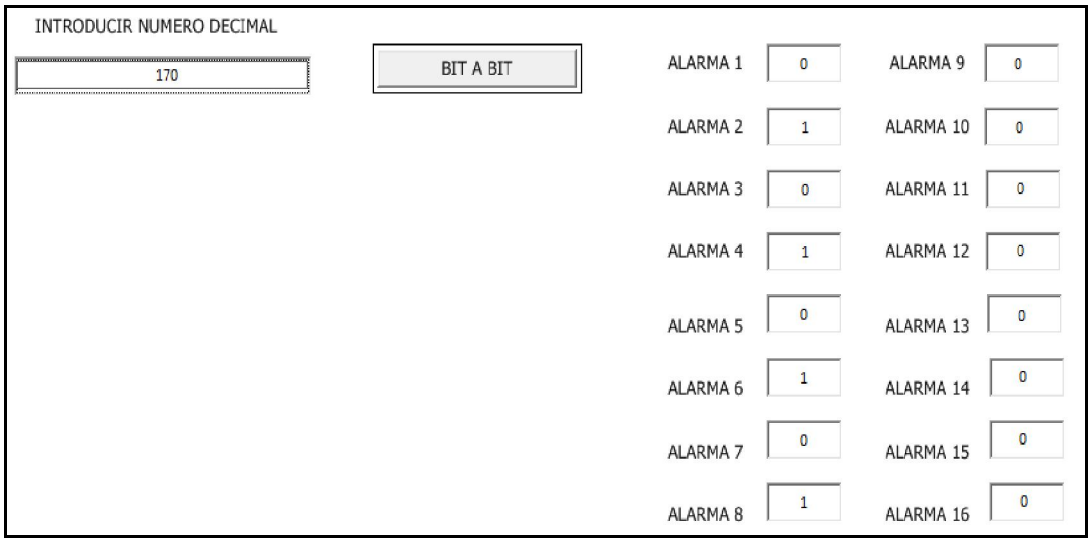


Fig.3.3 Ejecución del programa en Movicon

3.4.2. Resultados para la traducción de formato de un número y conversión del mismo, representado en Complemento a 2.

Basado en el diseño de pruebas del sub-epígrafe **3.3.1** para la prueba se toma el número 246, que corresponde en binario con 8 bits, al complemento a 2 del número 10, o sea es el 10 en negativo.

El cálculo en papel se desarrolló de la siguiente forma:

- Se toma el número inicial y se lleva a binario.

  246_d à $1111\ 0110_b$

- Se halla el complemento A1.

  $1111\ 0110_b$ à $0000\ 1001_b$

- Al resultado obtenido se le suma el número 1.

  $0000\ 1001_b$

  $+ \qquad 1$

  $0000\ 1010_b$

- El número obtenido se lleva a decimal para obtener el número esperado.

  $0000\ 1010_b$ à 10_d

El número esperado es 10. Al ejecutar el *script* en el Movicon, se observa mediante una instrucción *MsgBox*, en el cuadro de texto que el resultado mostrado es 10 (Fig.3.1). Por lo cual el resultado de la prueba es satisfactorio.

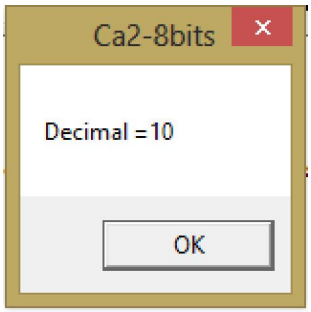


Fig.3.1 Cuadro de texto que muestra el resultado.

3.4.3. Resultados para el almacenamiento de datos de proceso o configuración en un archivo de texto.

Basado en el diseño de pruebas del sub-epígrafe **3.3.3** se creó una pantalla la cual contiene un botón, un cuadro de texto y un *slider*. Se modifica el *slider* para obtener el número deseado, en esta prueba el 33 (Fig.3.4). Se presiona el botón de envío y se comprueba si se creó el archivo de texto en la dirección que tiene definida el *script* y se verifica si se guarda correctamente el número 33.

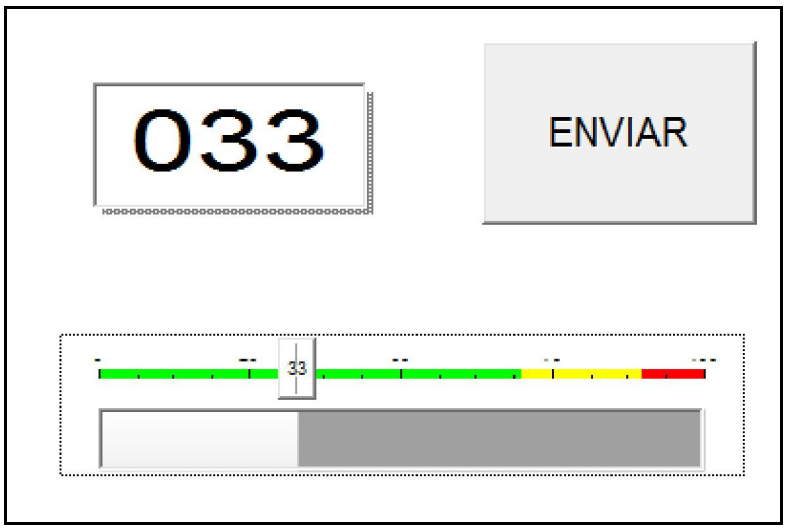


Fig.3.4 Pantalla de Movicon para la prueba de creación y escritura en archivo de texto.

Cuando se ejecuta el programa creado en LabVIEW (Fig.3.5) para leer desde archivos de texto se observa que la variable que muestra es la correcta. Por lo que se arriba a la conclusión de que la prueba realizada fue exitosa.

Fig.3.5 Programa de LabVIEW para leer archivos de texto

3.4.4. Resultados para la lectura de datos de proceso o configuración desde un archivo de texto.

Basado en el diseño de pruebas del sub-epígrafe **3.3.4** se crea un programa en LabVIEW donde se introduce la variable que se exporta al Movicon mediante un archivo de texto. En esta prueba 234 (Fig.3.6) es el valor de la variable que se inserta en el programa.

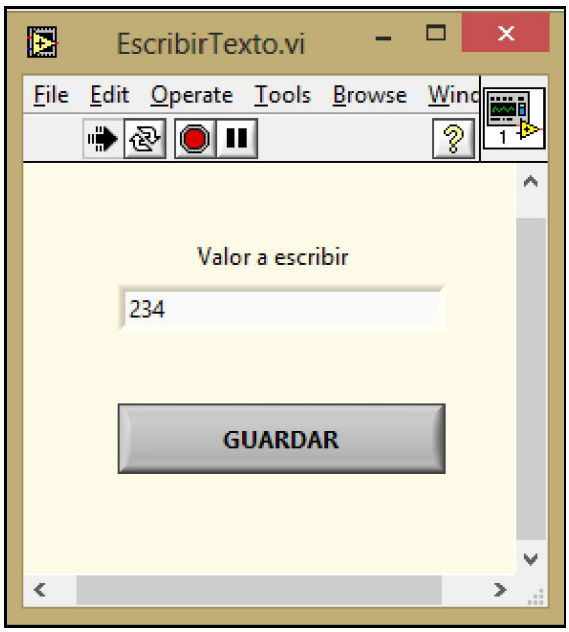


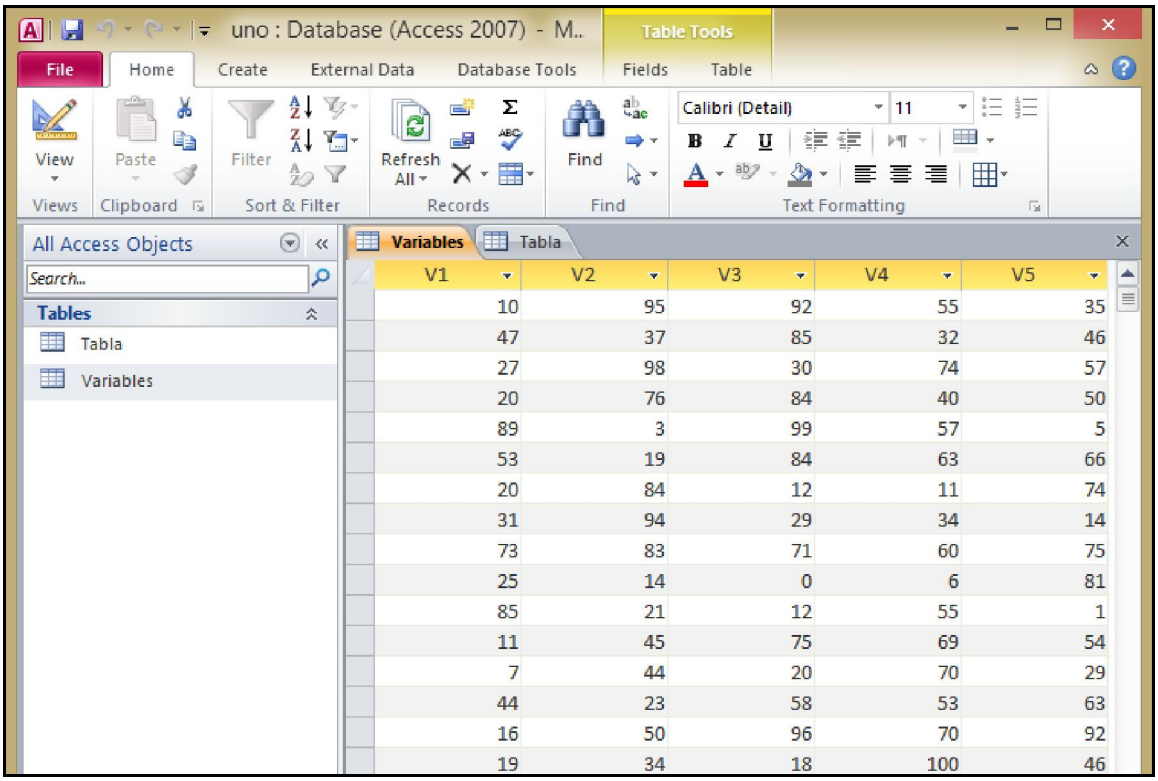
Fig.3.6 Programa para escribir en ficheros de texto

Se abre el archivo de texto y se observa que la variable que aparece en el fichero de texto es efectivamente 234. Se ejecuta el programa y en la pantalla (Fig.3.7) se verifica que la variable que se importa al Movicon es tiene un valor de 234. Se concluye que la prueba realizada obtuvo resultados satisfactorios.

Fig.3.7 Pantalla de Movicon que visualiza la variable importada desde el archivo de texto.

3.4.5. Resultados para el almacenamiento de valores de proceso y configuración en una base de datos de propósito general.

Basado en el diseño de pruebas del sub-epígrafe **3.3.5** se crea una base de datos en Microsoft Office Access 2007 (Fig.3.8), esta será la encargada de guardar los datos provenientes desde el SCADA y a su vez se podrá acceder a estos datos desde un pequeño VI creado con LabVIEW.



V1	V2	V3	V4	V5
10	95	92	55	35
47	37	85	32	46
27	98	30	74	57
20	76	84	40	50
89	3	99	57	5
53	19	84	63	66
20	84	12	11	74
31	94	29	34	14
73	83	71	60	75
25	14	0	6	81
85	21	12	55	1
11	45	75	69	54
7	44	20	70	29
44	23	58	53	63
16	50	96	70	92
19	34	18	100	46

Fig.3.8 Base de datos en Access

Para el desarrollo de esta prueba se realiza una interfaz (Fig.3.9) la cual muestra 5 variables que cada 1 segundo estarán variando constantemente y las mismas cada 1 segundo igualmente se guardan en la base de datos.

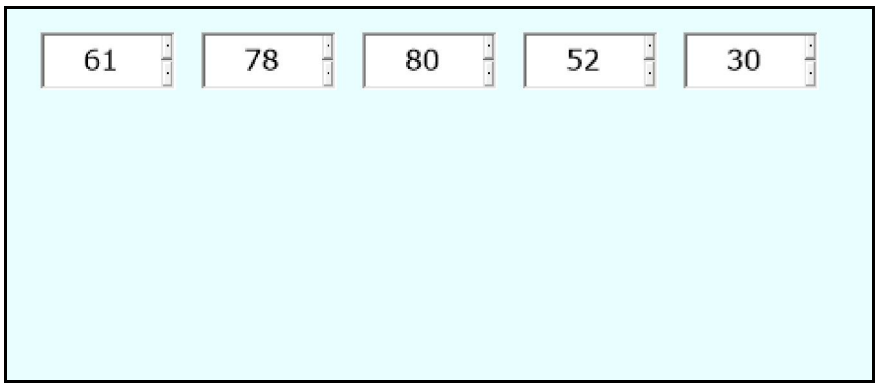


Fig.3.9 Pantalla de Movicon

Se abre la base de datos y se observa que las variables están siendo guardadas correctamente. Al ejecutar el programa creado en LabVIEW (Fig.3.10) se comprueba que las variables están siendo graficadas de forma satisfactoria.

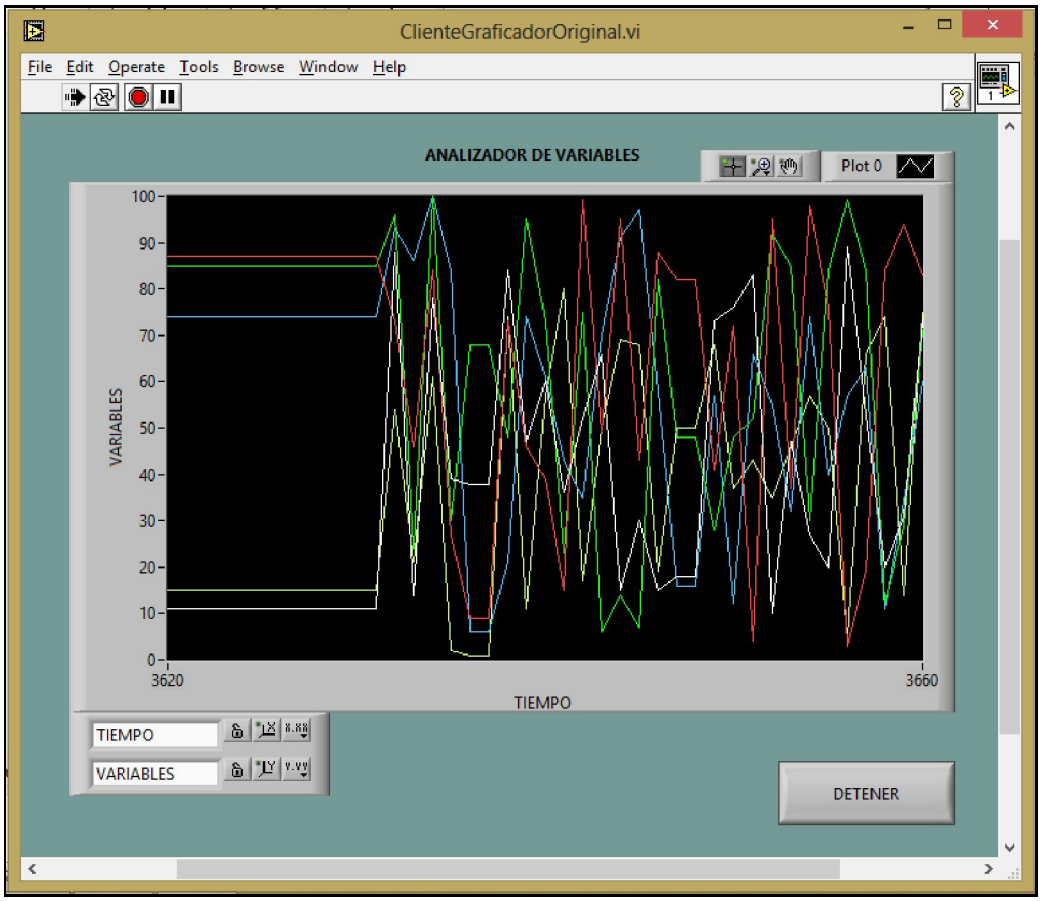


Fig.3.10 Programa creado en LabVIEW para graficar las variables obtenidas de la base de datos.

3.4.6. Resultados para el almacenamiento de valores de proceso y configuración en un archivo XML.

Basado en el diseño de pruebas del sub-epígrafe **3.3.6** se creó un sinóptico que contiene la representación de las 5 variables en cuadros de texto que están variando aleatoriamente cada 1 segundo, además presenta un botón que al activarlo se guardan las variables que están visibles actualmente en la pantalla en un archivo XML.

Se ejecuta el programa donde se visualizan las variables (Fig.3.11), en este mismo momento se activa el botón el cual se espera ejecute el *script* que envíe las variables al archivo XML.

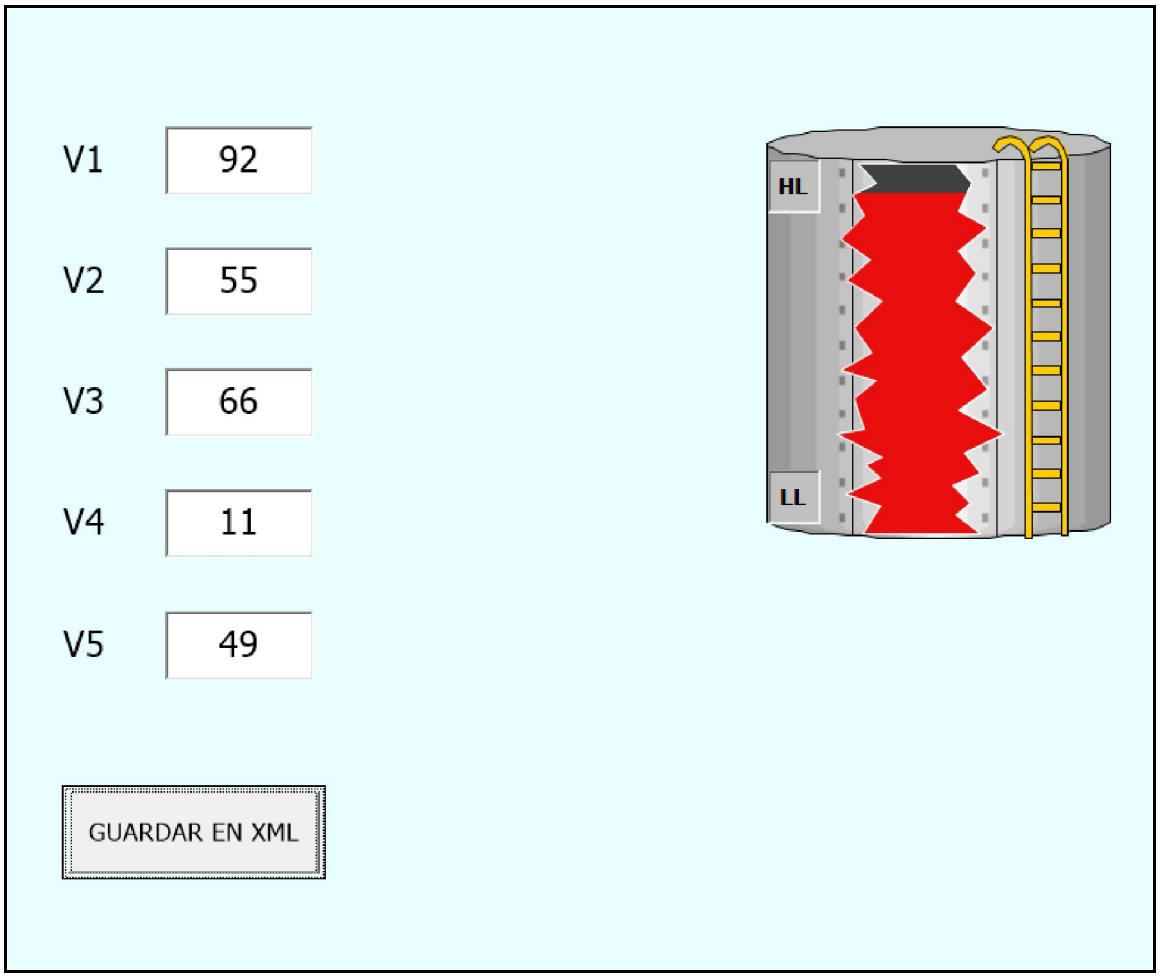


Fig.3.11 Pantalla de Movicon para el almacén de datos en formato XML

Se abre el archivo XML y se observa que las variables fueron almacenadas correctamente en el archivo XML (Fig.3.12). Se concluye que la prueba realizada fue exitosa.

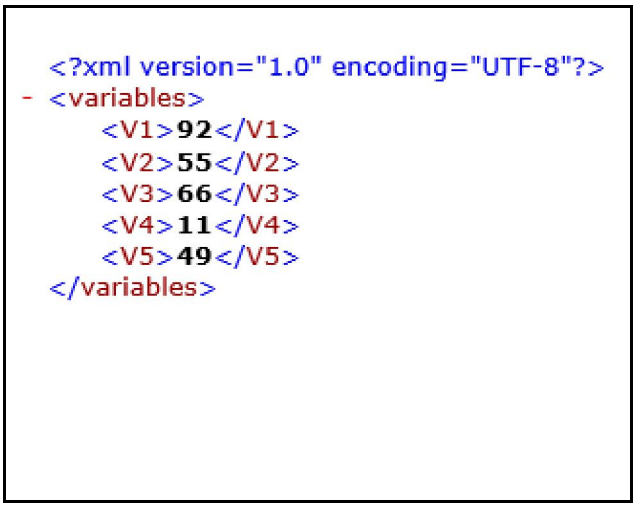


Fig.3.12 Archivo XML en el que se almacenan las variables.

3.1. Análisis económico.

El trabajo realizado tiene una gran importancia económica. Los beneficios económicos se derivan de la cantidad de horas que ahorra un programador de aplicaciones SCADA al no tener que investigar la forma de hacer y posteriormente codificar desde cero estos *scripts*. A continuación se muestra un ejemplo del costo que puede ahorrar un proyecto donde se suponga haya que realizar los scripts presentados en este trabajo.

Ejemplo:

A un desarrollador avezado de aplicaciones SCADA se le encarga la supervisión de un proceso que consta de 500 variables. Para ello estima un plazo de 60 días, trabajando 8 horas diarias para la terminación del mismo. El salario aproximado de un programador de esta índole es de $60.00 por hora. Suponiendo que para la aplicación que le fue encargada tuviera que crear los scripts presentados en este trabajo, de los 60 días de desarrollo, aproximadamente 10 días los invertiría en investigar, diseñar los algoritmos y codificar los scripts, para un total de 80 horas para la creación de los códigos.

En el caso que el programador cuente con un trabajo similar a este le tomaría a lo sumo 2 horas para la copia y modificación de todos los scripts, lo que le ahorraría 78 horas de trabajo, lo cual representa en dinero:

78 horas * $60 = $4680

3.2. Consideraciones parciales del capítulo.

Para la etapa de pruebas se hizo necesario crear un ambiente adecuado en cuanto a hardware y software, por lo que en el capítulo se realizó un contexto de las herramientas utilizadas. Se realiza el diseño de pruebas muy bien estructurado para evitar la mayor cantidad de errores posibles. Se realizó una descripción detallada de los resultados de las pruebas. Los resultados de la validación de los códigos creados fueron satisfactorios, cumpliendo de esta forma con los objetivos trazados anteriormente en el trabajo. El análisis económico realizado, en términos aproximados, brinda una buena idea de la importancia del trabajo desarrollado.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones.

Finalizado el trabajo y el estudio de los temas relacionados con los sistemas SCADA y evaluando los resultados alcanzados, se puede arribar a las siguientes conclusiones:

1. De acuerdo a la bibliografía estudiada se concluye que no existe alguna biblioteca compartida que pueda ser de uso público por programadores de aplicaciones SCADA
2. Con el trabajo se puede contar con un conjunto de funciones genéricas que se pueden utilizar para el desarrollo de aplicaciones SCADA.
3. Las pruebas validaron la correcta funcionalidad de los códigos creados..
4. Se comprobó que el lenguaje Visual Basic Scripts es potente, cómodo y funcional para la programación de *scripts* para aplicaciones SCADA.
5. Movicon v.11.3 es una plataforma SCADA que ofrece la posibilidad de realizar potentes y compactos sistemas de supervisión y control con agradables interfaces HMI. Como plataforma de pruebas cumplió a cabalidad sus objetivos.
6. El soporte para .NET es vital para crear códigos multifuncionales y potentes que complementan el funcionamiento de un sistema SCADA.

Recomendaciones.

1. Agrupar de alguna forma las funciones creadas en una biblioteca general (.h) o una biblioteca de enlace dinámico (dll).
2. Continuar con la investigación de las necesidades de módulos de funciones que pueda necesitar un sistema SCADA, así como la programación de las mismas para completar una biblioteca de funciones genéricas.

REFERENCIAS BIBLIOGRÁFICAS

Bailey, D. and E. Wright (2006). SCADA for Industry.

Castellanos, E. I. (2008). Sistemas de automatización. Santa Clara, Samuel Feijóo.

Castellanos, E. I. (2010). Sistemas de Automatización. Santa Clara, Universidad Central "Marta Abreu" de Las Villas.

Chacon, D., O. Dijort, et al. (2001). Supervisión y Control de Procesos.

Childs, M., P. Lomax, et al. (2003). VBScript in a Nutshell, O'Reilly.

Hernández, J. M. (2013). Sistemas SCADA.

Janus. (2006). "SCADA. Janus, Experto en Productos Dedicados." from http://www.ejanus.com.ar/software_jako_scada.php 10-2-2013.

Jones, D. (2004). Managing Windows with VBScript and WMI, Addison Wesley.

Lakhoua, M. N. (2009). "Application of functional analysis on a SCADA system of a thermal power plant." Advances in Electrical and Computer Engineering **Vol 9, No. 2**.

Lozano, C. D. C. M., C. R. (2009). Introducción a SCADA.

Meza, L. E. C. (2007). SCADA System's & Telemetry. México, Atlantic International University.

Microsoft (2013). "Versiones y dependencias de .NET Framework."

Montero, D., D. Barrantes, et al. (2004). Introducción a los Sistemas de Control y de Adquisición de Datos.

Penin, A. R. (2007). Sistemas SCADA. México, Editorial Marcombo.

Progea (2007). Movicon Programmer Guide.

Progea (2012). Movicon 11 Programmer Guide

Schneider-Electric. (2012). "Citect is now Schneider Electric." from http://www2.schneider-electric.com/sites/corporate/en/products-services/former-brands/citect.

Siemens (2005). WinCC Flexible. Comunication Part One.

Vidal., P. (2002). OPC: Un estándar en las redes industriales y buses de campo.

Wonderware (2013). "Wonderware InTouch HMI."

ANEXOS

ANEXO I

```
Sub Main()
  Dim Arr(16)
  Dim r,c,i,j
  Dim n,s
  Dim Pos01, Pos02, Pos03, Pos04, Pos05, Pos06, Pos07,
Pos08, Pos09, Pos10, Pos11, Pos12, Pos13, Pos14, Pos15,
Pos16


  numero = n


  i = 0
  s = 0
  c = 0


    If n = 0 Then
      For j = 0 To 15 Step 1
         Arr(j) = 0
      Next
    Else
      Do While c <> 1
        r = n Mod 2
        c = Int n / 2
        n = c
        Arr(i) = Int r
        i = i + 1
```

```
Loop
Arr(i) = c
    End If
       Bit1 = Arr(0)
       Bit2 = Arr(1)
       Bit3 = Arr(2)
       Bit4 = Arr(3)
       Bit5 = Arr(4)
       Bit6 = Arr(5)
       Bit7 = Arr(6)
       Bit8 = Arr(7)
       Bit9 = Arr(8)
       Bit10 = Arr(9)
       Bit11 = Arr(10)
       Bit12 = Arr(11)
       Bit13 = Arr(12)
       Bit14 = Arr(13)
       Bit15 = Arr(14)
       Bit16 = Arr(15)
End Sub
```

ANEXO II

```
Sub Main()
   Dim Arr(8)
   Dim r,c,i,j
   Dim n,s,temp
   Dim Pos01, Pos02, Pos03, Pos04, Pos05, Pos06, Pos07, Pos08
   n = 10
   i = 0
   s = 0
   c = 0

   If n = 0 Then
      For j = 0 To 7 Step 1
         Arr(j) = 0
      Next
   Else

      Do While c <> 1
        r = n Mod 2
        c = Int n / 2
        n = c
        Arr(i) = Int r
        I = i + 1
      Loop
      Arr(i) = c
   End If

       Pos01 = Arr(0)
       Pos02 = Arr(1)
       Pos03 = Arr(2)
       Pos04 = Arr(3)
       Pos05 = Arr(4)
       Pos06 = Arr(5)
       Pos07 = Arr(6)
       Pos08 = Arr(7)
```

```
If Pos08 = 0 Then
   If Pos01 = 1 Then
   a = 2^0
   End If
   If Pos02 = 1 Then
   b = 2^1
   End If
   If Pos03 = 1 Then
   c = 2^2
   End If
   If Pos04 = 1 Then
   d = 2^3
   End If
   If Pos05 = 1 Then
   e = 2^4
   End If
   If Pos06 = 1 Then
   f = 2^5
   End If

   If Pos07 = 1 Then
   g = 2^6
   End If
   x = a + b + c + d + e + f + g
Else
   If Pos01 = 0 Then
   a = 2^0
   End If
   If Pos02 = 0 Then
   b = 2^1
   End If
   If Pos03 = 0 Then
   c = 2^2
   End If
   If Pos04 = 0 Then
   d = 2^3
   End If
   If Pos05 = 0 Then
   e = 2^4
   End If
   If Pos06 = 0 Then
   f = 2^5
   End If
   If Pos07 = 0 Then
   g = 2^6
   End If
x = a + b + c + d + e + f + g + 1
End If
MsgBox "Decimal ="&x

End Sub
```

ANEXO III

```
Sub Main


Dim x,y



Const escribir=2
Set x = CreateObject("Scripting.FileSystemObject")
'crea archivo.txt
Set y = x.OpenTextFile("C:\Users\Joseo\Desktop\prueba2.txt",
escribir, True)'abre el archivo
y.WriteLine(v1) 'escribe en el archivo
'y.WriteBlankLines(3)
'Escribir tres caracteres de nueva línea al archivo.
'y.Write ("Esto es una prueba.")
'Escribir una línea.
y.Close


End Sub
```

ANEXO IV

```
Sub Main
     Dim fso, f1, ts, s
     Const ForReading = 1
     Set fso = CreateObject("Scripting.FileSystemObject")
     Set ts = fso.OpenTextFile("C:\Users\Joseo\Desktop\prueba2.txt",
ForReading)


        s = ts.ReadLine
        v2 = s
        'MsgBox(s)
        ts.Close()
End Sub
```

ANEXO V

```
'#Language "WWB.NET"

Imports System
Imports System.Data
Imports System.Data.OleDb
Imports System.Data.Odbc

Dim cn As New ADODB.ConnectionClass
Dim rs As New ADODB.RecordClass

Sub Main()
    conectar()
    desconectar()
End Sub

Public Sub conectar()

cn.ConnectionString = "DSN=cxn"
cn.Open
If cn.State = 0 Then
MsgBox "error"
Else
MsgBox "Conected"
End If
Dim x=20
Dim cad As String
cad = "insert into Variables values(" & V1 & "," & V2 & "," & V3 & ","
& V4 & "," & V5 & ")"
cn.Execute(cad)
End Sub

Public Sub desconectar()
   cn.Close
End Sub
```

ANEXO VI

```
Sub Main()
     'Se le asignan los valores de las variables reales a las
variables auxiliares
     'Esto se hace para poder convertir a string el valor
    x1 = V1
    x2 = V2
    x3 = V3
    x4 = V4
    x5 = V5


     ' Crea el objeto XmlWriterSettings.
     Dim settings As XmlWriterSettings = New XmlWriterSettings()
     settings.Indent = True
     ' Crea el objeto XmlWriter.
     Using writer As XmlWriter =
XmlWriter.Create("C:\Users\Joseo\Desktop\Variables.xml", settings)


         ' Comienza a escribir
         writer.WriteStartDocument()
         writer.WriteStartElement("variables") ' Raiz.


         writer.WriteElementString("V1", x1.ToString)
          writer.WriteElementString("V2", x2.ToString)
          writer.WriteElementString("V3", x3.ToString)
          writer.WriteElementString("V4", x4.ToString)
          writer.WriteElementString("V5", x5.ToString)


        writer.WriteEndElement()  ' Finalizar la escritura
         writer.WriteEndDocument()  ' Escribir fin de documento
     End Using
```