Universidad de las Ciencias Informáticas

Facultad 5

Título: Manejador ODBC para Sistemas de Supervisión y Control de Procesos Automatizados.

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Yenisleydi Pérez González

Tutor: Yolier Galán Tasse

La Habana, 2011

Declaración de Autoría

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.
Para que así conste firmo la presente a los 18 días del mes de Mayo del año 2010.

Firma del Autor	Firma del Tutor
(Yenisleydi Pérez González)	(Ing. Yolier Galán Tasse)

Datos de Contacto

Ing. Yolier Galán Tasse

Ingeniero en Ciencias Informáticas desde el 2010. Ejerce en el Centro de Desarrollo de Informática Industrial de la Universidad de las Ciencias Informáticas.

Agradecimientos

A mis padres, las personas a quien más amo en este mundo. A ellos que me han dado siempre el mejor ejemplo, el más puro amor y el más incondicional de los apoyos para poder lograr todos mis sueños. Gracias por confiar tanto en mí.

A mi abuela querida por todo su amor, por su gran ejemplo de mujer, de madre y de persona. A ti gracias por estar a mi lado y ayudarme a seguir adelante en los peores momentos.

A mi tía Mirky por tanto amor, apoyo y comprensión en todos estos años. Gracias por estar siempre ahí para mí.

A toda la familia por estar siempre pendiente de mis estudios y por su apoyo.

A Leonel, por confiar siempre en mi y ser un apoyo en los momentos difíciles.

A mi viejito Lázaro que siempre me ha ayudado.

A mis amigos y compañeros de estudio por ayudarme y soportarme todos estos años. Especialmente gracias a Edith, Ariel y Lili por su amistad, ayuda y por no dejarme renunciar jamás.

A todos mis profesores por su empeño y apoyo. Gracias especialmente a los profes Roberto Desagües y Humberto Rivero quienes me apoyaron en los momentos más difíciles.

A mi tutor por toda su ayuda en el desarrollo de este trabajo de diploma y sobre todo por su amistad a lo largo de estos 5 años.

A Manuel, que desde el inicio de este trabajo de diploma me guió y me apoyó en todo.

A la revolución por permitirme estudiar en esta universidad y poder realizar el sueño, no solo mío, sino también de mis padres.

Dedicatoria

A mi Chucha y Gordito, los mejores padres del mundo.

A mis primos queridos, Melissa y Jeikel

A la memoria de mi abuela Mireya.

Resumen

Las nuevas tecnologías están haciendo que las aplicaciones de automatización industrial, como SCADA, sean cada vez más utilizadas. La automatización industrial comprende una serie de soluciones destinadas a la captura de información de un proceso o planta industrial empleando, para este objetivo, dispositivos como pueden ser controladores lógicos programables (PLC), autómatas programables y sensores. Con el objetivo de realizar un tratamiento histórico de la información obtenida la misma es almacenada en bases de datos.

En el presente trabajo de diploma se abordan aspectos relacionados con el desarrollo del manejador basado en el estándar ODBC (Open DataBase Connectivity),en español Conectividad Abierta a Bases de Datos, que permite el acceso a la información almacenada en diferentes Sistemas Gestores de Base de Datos y que será utilizada por el sistema SCADA “Guardián del Alba” desarrollado en el Centro de Desarrollo de Informática Industrial, de la Universidad de las Ciencias Informáticas.

También se exponen en este trabajo las clases implementadas y las tecnologías utilizadas para el desarrollo del manejador, así como un estudio sobre los sistemas de supervisión y control de procesos automatizados y el estándar de acceso a datos ODBC, el cual amplía la capacidad de dichas aplicaciones software.

Tabla de Contenidos

Introducción

Los sistemas SCADA son mecanismos de control de procesos industriales utilizados por el hombre para representar el control y gestión de soluciones en una amplia gama de industrias, algunas de ellas son Sistemas de Gestión de Agua, Energía Eléctrica, Sistemas de Tránsito. Se trata de una aplicación de software especialmente diseñada para funcionar sobre ordenadores en el control de producción, proporcionando comunicación con los dispositivos de campo y controlando el proceso de forma automática desde la pantalla del ordenador, proveyendo de toda la información que se genera en el proceso productivo a diversos usuarios. Estos sistemas actúan sobre los dispositivos instalados en la planta, como son los controladores, autómatas, sensores, actuadores y registradores. Además permiten controlar el proceso desde una estación remota, para ello el software brinda una interfaz gráfica que muestra el comportamiento del proceso en tiempo real. Estos sistemas presentan tres tareas críticas a ejecutar:

- Recolección periódica, procesamiento y monitoreo de información del sistema a controlar.
- Control remoto de dispositivos de campo.
- Visualización de alarmas.

En el Centro de Desarrollo de Informática Industrial (CEDIN) ubicado en la Universidad de Ciencias Informáticas (UCI) se desarrollan productos para la automatización de las industrias. Actualmente se trabaja con empresas de meteorología, telecomunicaciones y medicina para automatizar sus procesos y además con la industria petrolera de Venezuela, para la cual se desarrolla un sistema SCADA llamado Guardián del Alba.
Este sistema necesita intercambiar información, mediante una interfaz única, con los distintos Sistemas Gestores de Bases de Datos (SGBD) y así extraer la información necesaria de estos, para ello se requiere de un manejador basado en el estándar Open DataBase Connectivity (OBDC) o lo que es lo mismo, conectividad abierta de bases de datos, este es un estándar de acceso a Bases de Datos (BD) que hace posible el acceso

a cualquier dato desde cualquier aplicación, sin importar qué SGBD almacene la información.

Como resultado de lo analizado anteriormente surge el siguiente **Problema Científico** a resolver: ¿Cómo intercambiar información entre el Sistema de Supervisión y Control desarrollado en el CEDIN de la Universidad de las Ciencias Informáticas y Gestores Bases de Datos?

A partir del problema científico se puede definir como **Objeto de Estudio** para la investigación: Estudio del estándar ODBC, teniendo como **Campo de Acción**: Manejo del estándar ODBC en sistemas SCADA.

Para dar solución al problema planteado anteriormente se propone como **Objetivo general**: Desarrollar un manejador para el sistema de Supervisión y Control realizado en el CEDIN que permita la comunicación con Gestores Bases de Datos mediante el estándar ODBC.

Para desarrollar satisfactoriamente la investigación se han trazado las siguientes Tareas Investigativas:

- Estudio y análisis de la documentación referente a manejadores para el establecimiento de los conceptos básicos asociados a su implementación.
- Estudio del estándar ODBC para identificar las características y los requerimientos a tener en cuenta en el desarrollo del manejador.
- Estudio de las bibliotecas de acceso a datos para la posterior selección de la biblioteca a utilizar en el desarrollo del manejador.
- Análisis y selección de estructuras de programación que no limiten la capacidad multiplataforma de la solución final.
- Diseño e implementación del manejador según las características y requisitos identificados y las estructuras de programación seleccionadas.
- Validar el funcionamiento del manejador desarrollado así como su portabilidad.

Se utilizarán varios métodos científicos de investigación como: Analítico-Sintético, mediante el cual se podrán conocer los principales fundamentos y teorías relacionadas con el desarrollo de manejadores y Modelación ya que se realizarán diagramas y modelos que permitan estructurar teóricamente el sistema.

Capítulo 1: Fundamentación Teórica

1.1 Introducción.

Actualmente el principal objetivo de la automatización industrial es gobernar la actividad y la evolución de los procesos sin que sea necesaria la intervención continua del operador humano. En este capítulo se presenta un estudio sobre los sistemas de control de procesos industriales SCADA, sus características y funciones principales. Por otra parte se hace referencia al estándar ODBC exponiendo elementos de su historia, funcionamiento y características generales. También se abordan las tecnologías y herramientas seleccionadas para llevar a cabo el desarrollo del manejador.

1.2 Sistemas SCADA.

Los sistemas SCADA son aplicaciones de software, diseñadas con la finalidad de controlar y supervisar procesos a distancia, que cuentan con módulos o bloques de software como son:

- Configuración.
- Interfaz gráfico del operador.
- Módulo de proceso.
- Gestión y archivo de datos.
- Comunicaciones. [1]

En estos sistemas la adquisición de los datos puede estar a cargo de un PLC (Controlador Lógico Programable) el cual toma las señales y las envía a las estaciones remotas usando un protocolo determinado. Por otra parte para la realización de las tareas de supervisión y control el software SCADA permite que el operador pueda visualizar en la pantalla del computador de cada una de las estaciones remotas que conforman el sistema los estados de los procesos, las situaciones de alarma y que pueda tomar acciones físicas sobre algún equipo lejano, la comunicación se realiza mediante redes LAN o buses de campo como: [2]

• Profibus DP

• Interbus-S

• Profibus FMS

• Genius I/O

• AS-I

1.2.1 Prestaciones.

Un software SCADA ofrece:

- Posibilidad de crear paneles de alarma, que exigen la presencia del operador para reconocer una parada o situación de alarma, con registro de incidencias.
- Generación de datos históricos de las señales de planta, que pueden ser volcados para su proceso sobre una hoja de cálculo.
- Ejecución de programas, que modifican la ley de control, o incluso anular o modificar las tareas asociadas al autómata, bajo ciertas condiciones.
- Posibilidad de programación numérica, que permite realizar cálculos aritméticos de elevada resolución sobre la CPU del ordenador.[2]

1.2.2 Requisitos.

Para que la instalación de un sistema SCADA, en una empresa, sea perfectamente aprovechada el mismo debe cumplir los siguientes requisitos:

- Deben ser sistemas de arquitectura abierta, capaces de crecer o adaptarse según las necesidades cambiantes de la empresa.
- Deben comunicarse con total facilidad y de forma transparente al usuario con el equipo de planta y con el resto de la empresa (redes locales y de gestión).
- Deben ser programas sencillos de instalar, sin excesivas exigencias de hardware, y fáciles de utilizar, con interfaces amigables con el usuario.[1]

1.2.3 Funciones Principales de un Sistema SCADA:

- *Supervisión remota de instalaciones y equipos:* Permite al operador conocer el estado de desempeño de las instalaciones y los equipos alojados en la planta, permitiendo 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 de manera automática y también manual.
- *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 dentro de una 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 se 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:* Brinda la posibilidad de almacenar los datos adquiridos, el tiempo de almacenamiento dependerá del operador o del autor del programa, pudiendo ser analizada esta información posteriormente.
- *Programación de eventos:* Brinda la posibilidad de programar subprogramas que generen automáticamente reportes, estadísticas, gráfica de curvas y activación de tareas automáticas.[2]

1.3 Interfaz Genérica de Driver.

En las tareas de adquisición de datos de un SCADA pueden verse involucrados diferentes dispositivos de campo. Resulta engorroso que este contemple en su código los protocolos que corresponden a cada uno de estos dispositivos. Por lo tanto lo más común es que se realice la implementación de un protocolo general y los drivers se encarguen de traducirlo a los protocolos específicos de los dispositivos. La Línea de Manejadores diseñó e implementó una Interfaz Genérica para el acceso a los dispositivos, la cual está formada por un conjunto de funciones definidas en el lenguaje de programación ANSI C, que brindan la posibilidad del acceso a dispositivos industriales. El componente de software denominado “DriverCore” es la implementación de la IGD que ha servido como base para

el desarrollo de varios manejadores. Entre estos se encuentran Modbus TCP, Modbus RTU, Modbus ASCII, DNP3, Ethernet/IP, ABInterchange, DF1 de Allen Bradley, BSAP de Bristol para equipos industriales y el Cobas b121 para equipos de diagnóstico. Algunos de los aspectos que se deben especificar para cada manejador son los siguientes:

- Establecer la forma de validar tanto las direcciones de las variables como las de los dispositivos.
- Establecer los criterios de comparación para las direcciones de las variables.
- Especificar el concepto de bloque de variables para el manejador que se esté desarrollando.
- Determinar la forma de establecer la comunicación con los dispositivos.
- Identificar los parámetros de configuración necesarios.
- Colocar el nombre que identifica a los manejadores en la biblioteca resultante.

El componente DriverCore ha sido utilizado como base para el desarrollo del manejador ODBC.

1.4 Manejadores.

Un manejador de dispositivo es un conjunto de rutinas para acceder a un periférico o dispositivo. Constituye la interfaz software con el dispositivo, ocultando al sistema operativo las particularidades del hardware permitiendo que éste pueda acceder al dispositivo a través de una interfaz estandarizada, pudiendo ser compartido simultáneamente por varias aplicaciones. Por otra parte, un manejador para el acceso a base de datos, constituye la interfaz entre los datos de bajo nivel almacenados en la base de datos y los programas de aplicaciones y las consultas realizadas al sistema. El sistema SCADA emplea a los manejadores para comunicarse e intercambiar información tanto con dispositivos de campo, como con BD.

1.4.1 Manejador ODBC.

Un manejador basado en el estándar ODBC, es decir, de conectividad abierta de bases de datos constituye un mecanismo para acceder a datos contenidos y manejados por SGBD , utilizando el SQL (Structured Query Language) en español lenguaje estructurado de consultas, como lenguaje estándar de acceso a datos.

1.5 Historia de ODBC.

En 1992, con el propósito de facilitar el trabajo con los SGBD Microsoft crea el primer driver o manejador ODBC, diseñado especialmente para unificar el proceso de acceso a bases de datos. Tras el éxito de ODBC Microsoft creó OLE DB, un estándar de acceso a datos que se extendió a cualquier fuente de datos que proporcionara datos en formato tabular. El plan de Microsoft era que OLE DB suplantara a ODBC como el estándar de acceso a datos más común. Con este mismo objetivo surge ADO, otro estándar para el acceso a datos creado por Windows el cual amplió las funcionalidades de OLE DB. Sin embargo ODBC ha seguido siendo el estándar más utilizado para el acceso a datos, tanto para los SGDB relacionales como en los no relacionales puesto que brinda la máxima interoperabilidad a los desarrolladores de aplicaciones al permitirles escribir una única aplicación para acceder a fuentes de datos de diferentes proveedores. Desde su surgimiento ha sido incluido en cada copia de Windows y hoy en día hay versiones para casi cualquier sistema operativo en uso además es independiente de los SGBD y lenguajes de programación.

1.5.1 Características.

El estándar ODBC presenta las siguientes características:

- ODBC es una interfaz de programación de aplicaciones estándar que utiliza SQL como lenguaje estándar de acceso a datos.
- Oculta al programador la complejidad a la hora de conectarse a un origen de datos: por ejemplo, el acceso a los datos a través de redes de comunicación es transparente.
- Permite a múltiples aplicaciones acceder a múltiples orígenes de datos.
- Proporciona un modelo de programación homogéneo, es decir, bases de datos muy diferentes se manejan, vía ODBC, como si fueran idénticas, siendo ODBC el encargado de realizar las adaptaciones necesarias. [3]

1.5.2 Arquitectura.

La arquitectura de ODBC está basada en cuatro componentes:

- Aplicaciones: Son las responsables de interactuar con el usuario y de llamar a las funciones ODBC para ejecutar sentencias SQL y obtener los resultados.
- El Driver Manager (Administrador de Manejadores): Se encarga de cargar y llamar a los drivers necesarios, según lo demanden las aplicaciones.
- Drivers (Manejadores): Procesan las llamadas a las funciones ODBC, ejecutan sentencias SQL y devuelven los resultados a las aplicaciones.
- Orígenes de datos: Consisten en conjuntos de datos, más todo lo que pueda ser necesario para llegar hasta ellos; sistemas operativos, gestores de bases de datos y redes de comunicación. [3]

1.5.1.2 Funcionamiento.

Utilizando ODBC las aplicaciones pueden acceder a datos almacenados en una gran variedad de ordenadores personales, miniordenadores y grandes ordenadores, incluso aunque el SGBD que posea cada uno de ellos utilice un formato diferente para guardar la información. Esto se logra al insertar una capa intermedia llamada manejador de base de datos entre la aplicación y el SGBD. El propósito de esta capa es traducir las consultas de datos de la aplicación en comandos que el SGBD entienda. Cada SGBD necesita su controlador (driver ODBC) específico para transformar las llamadas de la aplicación en llamadas específicas del SGBD, de forma que este pueda responder.

Para realizar la conexión a una BD a través de ODBC es necesario que se configure un DSN (Data Source Name) en español Nombre de origen de datos, en donde se define la ruta y parámetros de la conexión a partir de los datos que solicite el fabricante, siendo este el identificador de cada una de las conexiones.

El DSN se crea mediante el Administrador de orígenes de datos de ODBC que proporciona Windows. En el caso de los sistemas Linux/Unix para realizar la conexión a una BD vía ODBC es necesario que se encuentren instalados en la computadora el paquete unixODBC, el cual no es más que la implementación del estándar ODBC para plataformas libres, el paquete unixodbc-bin que es el que proporciona aplicaciones gráficas para el trabajo con unixODBC tales como: ODBCConfig, que es la herramienta que se utiliza para la configuración del DSN y el DataManager, una herramienta de consulta para el acceso a bases de datos a través de ODBC. Otro paquete que debe ser instalado es el unixodbc-dev, este contiene los archivos de desarrollo (cabeceras y

bibliotecas) para unixODBC. Este paquete solo se instala cuando se desea desarrollar aplicaciones que utilicen el estándar ODBC, como es el caso del Manejador ODBC.

Independientemente del tipo de BD a la que se quiera acceder, en la creación del DSN se requieren algunos atributos como son:

- Data Source: Nombre que se le asigna a la conexión para referirnos a ella desde las aplicaciones o programas que deban conectarse con la BD. Este atributo, específicamente, constituye el identificador de las conexiones.
- DataBase: Nombre de la fuente de datos a la cual se desea acceder.
  Driver: Driver ODBC correspondiente al SGBD que se desea utilizar. La selección de este driver constituye el primer paso para la configuración del DSN.
- User Name / Password: Nombre de usuario y la contraseña que la BD requiera para la autenticación.

Una vez configurado el DSN los pasos que se realizan en una aplicación que interacciona con una fuente de datos a través de ODBC son:

- Se selecciona una fuente de datos (DSN).
- Se carga el driver o controlador correspondiente.
- Se establece la conexión.
- Se realizan las inserciones, modificaciones, actualizaciones, borrados, etc. sobre la BD a través de sentencias SQL.
- La aplicación se desconecta de la fuente de datos para terminar la interacción.

1.6. Tecnologías.

1.6.1 Selección de la biblioteca para el acceso a datos vía ODBC.

Para el desarrollo del manejador ODBC se hizo necesario realizar un proceso de selección de la biblioteca para acceso a datos a utilizar, las más destacadas por su rendimiento actualmente son: Oracle Template Library (OTL), Simple Oracle Call Interface (SOCI) y BOOST.

Criterios	OTL	BOOST	SOCI
Descripción	Biblioteca basada en plantillas, se integra con la Biblioteca de plantillas estándar (STL). Su código fuente es portátil y compatible con versiones de compiladores de C++ tanto de 32 como de 64 bits. OTL en su versión 4 cubre todas las funcionalidades de una biblioteca de C++ para el acceso a base de datos con sólo un puñado de clases, las cuales son: otl_stream, otl_connect, otl_exception, otl_long_string.	Es básicamente un repositorio de bibliotecas de C++ formada por más de 80 bibliotecas individuales incluidas las de álgebra lineal, la generación de números pseudoaleatorios, multihilos, procesamiento de imágenes, expresiones regulares, pruebas unitarias, entre otros. Posee su propia licencia: Boost Software License, la cual es mucho más permisiva que la GPL y permite que las bibliotecas sean utilizadas en programas comerciales sin problemas.	Es una biblioteca de acceso a base de datos que hace posible las consultas de SQL embebido en el código de C + +, manteniéndose totalmente dentro del estándar de este lenguaje, proporcionándole a los programadores una forma de acceder a bases de datos SQL de la forma más natural e intuitiva. SOCI está desarrollada bajo la licencia de Boost.

Lenguaje	C++	C++	C++
Capacidad Multiplataforma	OTL trabaja tanto en las versiones de Windows como en Unix.	Boost trabaja en casi cualquier sistema operativo moderno, incluyendo variantes de UNIX y Windows. Algunas de las distribuciones más populares de Linux y Unix como Fedora, Debian, y NetBSD incluyen paquetes de Boost.	Windows ,Unix
SGBD que Soporta	Oracle 7 Oracle 8 Oracle 8i Oracle 9i DB2 MS SQL Server Sybase MySQL Interbase PostgreSQL	Oracle PostgreSQL MySQL	Oracle PostgreSQL MySQL

Luego de realizar un estudio sobre las referidas bibliotecas y teniendo en cuenta los criterios de comparación anteriormente expuestos se decide utilizar, para el desarrollo del manejador ODBC, la biblioteca OTL en su versión 4. Se trata de un único archivo de cabecera que además de Oracle es compatible con diversos SGBD, además posee una amplia documentación disponible lo cual facilita su estudio y utilización. Por otra parte es compatible con un gran número de compiladores de C++, los principales son:

- Sun C++ Workshop 6.x y versiones siguientes.
- GNU C++ 3.3.x y versiones siguientes.
- VC++ 6.0 y versiones siguientes.
- HP ANSI C++ (aCC) 1.x.
- AIX, Visual Age C++ 6.x y versiones siguientes.
- Borland C++ 5.x.
- Intel C++ 7.0, 8.0, 9.x (Windows, Linux).

1.6.2 IDE: Eclipse

Para el desarrollo del manejador se hace necesaria la correcta selección tanto del lenguaje de programación como del IDE que se utilizará. Un IDE (Integrated Development Environment) en español Entorno de Desarrollo Integrado es un programa de aplicación, utilizado para el desarrollo de otras aplicaciones, compuesto por un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI), que provee un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Python, Java, C#, Delphi y Visual Basic. Eclipse, el IDE seleccionado para el desarrollo del manejador ODBC, es un entorno de desarrollo multiplataforma de código abierto. Fue originalmente desarrollado por IBM, sin embargo en la actualidad está siendo desarrollada por la Fundación Eclipse. Esta herramienta ofrece las siguientes características: editor de texto, resaltado de sintaxis, compilación en tiempo real, pruebas unitarias con JUnit control de versiones con CVS y soporte a SGBD.

1.6.3 Lenguaje de Programación: C++

Este lenguaje surge en 1980 por creación de Bjarne Stroustrup con el objetivo de añadir a C nuevas características: clases y funciones, tipos genéricos, expresiones así como la posibilidad de declarar variables en cualquier punto del programa. Es multiplataforma, o

sea se puede usar en diferentes sistemas operativos. Algunas de las características que lo distinguen de los demás lenguajes de programación son:

- *Programación orientada a objetos*: La posibilidad de orientar la programación a objetos permite al programador diseñar aplicaciones desde un punto de vista más cercano a la vida real. Además, permite la reutilización del código de una manera más lógica y productiva.
- *Portabilidad*: Un código escrito en C++ puede ser compilado en casi todo tipo de ordenadores y sistemas operativos sin hacer apenas cambios.
- *Brevedad*: El código escrito en C++ es muy corto en comparación con otros lenguajes, sobre todo porque en este lenguaje es preferible el uso de caracteres especiales que las "palabras clave".
- *Programación modular*: Un cuerpo de aplicación en C++ puede estar hecho con varios ficheros de código fuente que son compilados por separado y después unidos. Además, esta característica permite unir código en C++ con código producido en otros lenguajes de programación como Ensamblador o el propio C.
- *Velocidad*: El código resultante de una compilación en C++ es muy eficiente, gracias a su capacidad de actuar como lenguaje de alto y bajo nivel y a la reducida medida del lenguaje.

1.6.4 Metodología: RUP

RUP es un Proceso Unificado propuesto por IBM actualmente considerado como un estándar en el desarrollo de software en las empresas. Es un modelo que permite el desarrollo de software a gran escala, mediante un proceso continuo de pruebas y retroalimentación, garantizando el cumplimiento de ciertos estándares de calidad. RUP define claramente quien, cómo, cuándo y qué debe hacerse en un proyecto. [5]

Posee tres características esenciales:

- Está dirigido por los Casos de Uso: que orientan el proyecto a la importancia para el usuario y lo que este quiere.
- Está centrado en la arquitectura: que relaciona la toma de decisiones que indican cómo tiene que ser construido el sistema y en qué orden.

- Es iterativo e incremental: donde divide el proyecto en miniproyectos donde los casos de uso y la arquitectura cumplen sus objetivos de manera más depurada.

RUP divide el proceso en 4 fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en los distintas actividades.

- Inicio: Se hace un plan de fases, se identifican los principales casos de uso y se identifican los riesgos. Se define el alcance del proyecto.
- Elaboración: se hace un plan de proyecto, se completan los casos de uso y se eliminan los riesgos.
- Construcción: se concentra en la elaboración de un producto totalmente operativo y eficiente y el manual de usuario.
- Transición: se instala el producto en el cliente y se entrena a los usuarios.

1.6.5 Lenguaje de Modelado: UML

UML (Unified Modeling Language) o lo que es lo mismo Lenguaje Unificado de Modelado es, precisamente, un lenguaje de modelado que permite la representación conceptual y física de un sistema. Las características principales de este lenguaje son:

- Aporta una notación estándar orientada a objetos.
- Permite describir un sistema en diferentes niveles de abstracción.
- Divide cada proyecto en un número de diagramas que representan diferentes vistas del proyecto.
- UML se puede aplicar tanto a sistemas informáticos como a sistemas que no son informáticos.

Con la utilización de UML es posible llevar a cabo las siguientes funciones:

- Visualizar: UML permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.
- Especificar: UML permite especificar cuáles son las características de un sistema antes de su construcción.

- Construir: A partir de los modelos especificados se pueden construir los sistemas diseñados.
- Documentar: Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

1.6.6 Herramienta Case: Visual Paradigm

Las herramientas CASE (Computer Aided Software Engineering), en español Ingeniería de Software Asistida por Computadora, son aplicaciones informáticas utilizadas en el proceso de desarrollo de software en tareas como: realizar un diseño del proyecto, cálculo de costos, implementación de parte del código automáticamente a partir del diseño, compilación automática, documentación o detección de errores entre otras.

Visual Paradigm es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Este software ayuda a una, más rápida, construcción de aplicaciones de calidad y a un menor coste. [4]

Las características principales de esta herramienta son las que a continuación se enuncian:

- Soporte de UML.
- Ingeniería inversa - Código a modelo, código a diagrama.
- Generación de código - Modelo a código, diagrama a código.
- Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Ingeniería inversa de bases de datos - Desde Sistemas Gestores de Bases de Datos existentes a diagramas de Entidad-Relación.
- Editor de figuras.

1.7 Conclusiones.

Luego de analizar las características y potencialidades tanto de un sistema SCADA como del manejador ODBC se puede concluir que la utilización de dicho estándar para el acceso a datos, en aplicaciones como SCADA, supone una ventaja en cuanto al proceso de intercambio de datos tan necesario para este tipo de sistemas ya que hace posible

que desde la aplicación se acceda a información almacenada en cualquier SGBD, que posea un driver ODBC, sin realizar ningún cambio en la codificación del programa. Por otra parte, teniendo en cuenta que en la calidad de cualquier producto de software que se desarrolle influye la acertada selección de las tecnologías y las herramientas a utilizar, para el desarrollo del manejador ODBC como se ha apreciado a lo largo de este capítulo se realizó un estudio detallado de cada una de las tecnologías y herramientas utilizadas, justificando la selección de cada una de ellas.

Capítulo 2: Diseño e Implementación

Introducción.

En este capítulo, referido al diseño e implementación del manejador ODBC, se abordan las características del manejador, desde el punto de vista de su arquitectura y clases definidas e implementadas en el proceso de desarrollo del mismo.

2.1 Implementación de la IGD.

A partir de la IGD se puede desarrollar manejadores utilizando las clases que proporciona la biblioteca DriverCore. Esta, además, carga opcionalmente una biblioteca de transporte asíncrona (basada en ASIO 1.4.1) que contiene fábricas de transportes TCP y serie. También provee un conjunto de funcionalidades como: Introspección de manejadores y dispositivos, manejo de las direcciones y clasificación de las variables en bloques de lectura o escritura. La Introspección se refiere a la capacidad de los manejadores y dispositivos de mostrar las propiedades que exponen y la capacidad de obtener el valor de esas propiedades a partir del nombre de la misma y modificar ese valor igualmente a partir del nombre.

3.2 Arquitectura.

La Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema. La importancia de esta radica en que sus diferentes elementos garanticen el cumplimiento de las características tanto en la funcionalidad que el sistema debe ofrecer como en los requerimientos de calidad que debe satisfacer.

En el caso del manejador ODBC se definió una arquitectura en capas, este tipo de arquitectura permite probar los componentes por separado, que se lleve a cabo un desarrollo paralelo (en cada capa), mantenimiento y soporte más sencillo, mayor flexibilidad y también provee una alta escalabilidad. Las capas definidas son: Capa de Driver, Capa de Protocolo y Capa de Transporte. A continuación se realiza una descripción de las mismas.

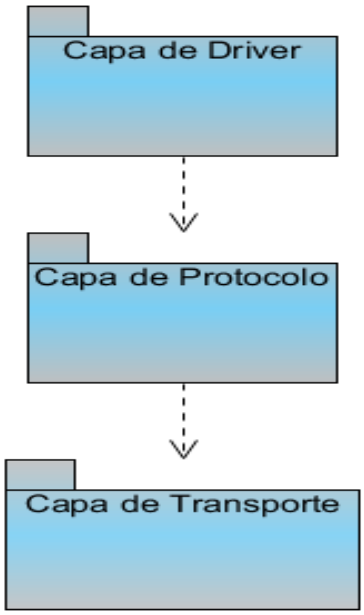


Figura 1. Arquitectura 3 capas del manejador ODBC.

3.2.1 Capa de Driver.

En la capa de Driver se implementan todas las clases que brinda el DriverCore para el desarrollo de los manejadores. Para esto es necesario heredar de las clases Device, Driver, DriverMetaClass y DeviceMetaClass. Otras clases que también deben ser implementadas son las clases Block y la IProtocolAddress.

3.2.2 Capa de Protocolo.

En esta capa se implementa la lógica de comunicación entre el manejador y los SGBD. Para esto se define la clase EndPoint que implementa las funcionalidades que permiten llevar a cabo el intercambio de información con los SGBD. También se definen otras clases como ODBC Connect la cual es la encargada de gestionar las consultas SQL y la conexión mediante ODBC al SGBD.

3.2.3 Capa de Transporte.

En la capa de Transporte se utiliza la biblioteca TransportProvider la cual está formada por un conjunto de clases e interfaces que permiten la definición de transportes TCP y Serie. Algunas de ellas son ITransport, de la que heredan ITCPTransport, ISerialTransport y de estas heredan TCPTransport y SerialTransport respectivamente. Se implementa la

clase ODBCTransport que hereda también de ITransport y es la clase que implementa el concepto de capa de transporte entre el manejador y la base de datos.

3.3 Decisiones de diseño.

Para la implementación del manejador ODBC se tomaron una serie de decisiones con el fin de satisfacer las necesidades del cliente y teniendo en cuenta una serie de buenas prácticas de aplicación recomendable en el diseño de software, como son, el uso de patrones GRASP (General Responsibility Assignment Software Patterns) en español patrones generales de software para la asignación de responsabilidades. Los patrones aplicados son los que a continuación se presentan:

- Experto: Este patrón responde a la interrogante de cómo asignar responsabilidades indicando que la responsabilidad de la creación de un objeto o la implementación de un método debe recaer sobre la clase que conoce toda la información necesaria para crearlo.
- Creador: Este patrón identifica quién debe ser el responsable de la creación, o instanciación de nuevos objetos de alguna clase.
- Bajo acoplamiento: Este patrón consiste en tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.
- Alta cohesión: Este patrón consiste en que la información que almacena una clase debe de ser coherente y debe estar, en la medida de lo posible, relacionada con la clase.

3.4 Diagrama de Clases del Diseño.

Un diagrama de clases del Diseño describe gráficamente las especificaciones de las clases en una aplicación. Puede contener la siguiente información:

- Clases, asociaciones y atributos.
- Interfaces, con sus operaciones y constantes.
- Métodos.
- Información sobre los tipos de los atributos.
- Dependencias.

En este diagrama se puede observar el componente de software DriverCore y las clases que este brinda para la implementación de manejadores y de las cuales heredan: ODBCProtocolAdress, ODBCDevice, ODBCDeviceMetaclass, ODBCDriverMetaclass, ODBCDriver y ODBCBlock.

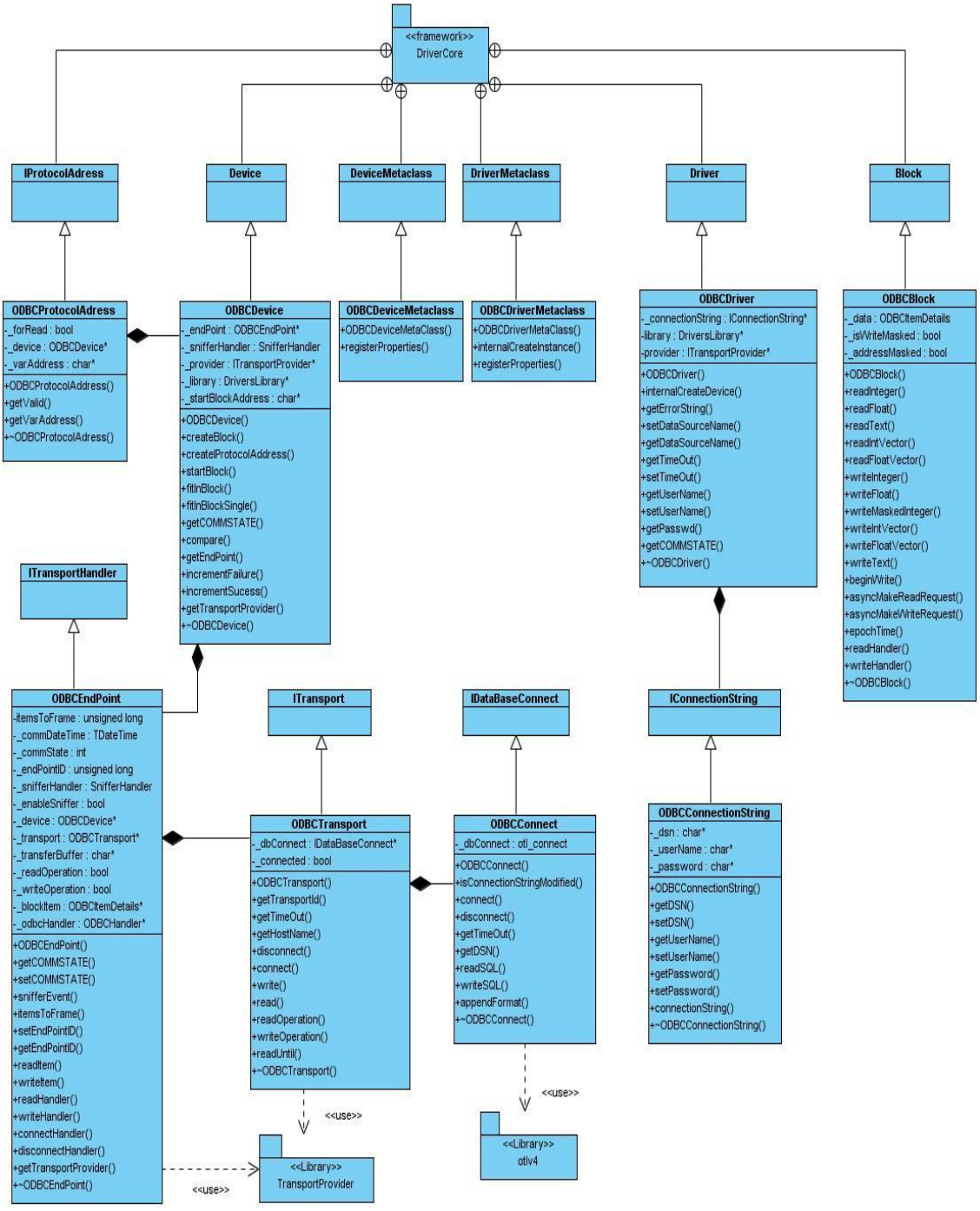


Figura 2. Diagrama de Clases del Diseño.

3.5 Descripción de Clases del Diseño.

A continuación se presenta una descripción de algunas de las clases más importantes implementadas para el desarrollo del manejador ODBC. Se muestra para cada una de ellas una breve descripción a fin de entender la función que realiza.

3.5.1 Descripción de la clase ODBCEndPoint.

La clase ODBCEndPoint posee las funcionalidades necesarias para establecer la comunicación con los SGBD. Esta clase brinda una interfaz de comunicación para realizar lecturas y escrituras en campos sobre la BD.

Tabla 1. ***Descripción de la clase ODBCEndPoint.***

Nombre: ODBCEndPoint	
Tipo de Clase: Controladora	
Atributo	**Tipo**
itemsToFrame	unsigned long
_commDateTime	int
_commState	unsigned long
_endPointID	SnifferHandler
_snifferHandler	bool
_enableSniffer	ODBCDevice*
_device	ODBCTransport*
_transport	char*
_transferBuffer	bool
_readOperation	bool
_writeOperation	bool
_blockItem	ODBCItemDetails*
_odbcHandler	ODBCHandler*
Funcionalidades:	
Nombre	ODBCEndPoint (ODBCDevice* pdevice, SnifferHandler psnifferHandler, IConnectionString* &connStr)
Descripción	Constructor de la clase ODBCEndPoint.
Nombre	~ODBCEndPoint()
Descripción	Destructor de la clase ODBCEndPoint.

Nombre	getCOMMSTATE(TDateTime* dateTime)
Descripción	Devuelve el estado de la calidad de la comunicación con el SGBD.
Nombre	setCOMMSTATE(intstate)
Descripción	Cambia el valor del estado de la comunicación.
Nombre	snifferEvent (unsigned long size, unsigned char* buffer, ConnectionEvents event)
Descripción	Evento del sniffer que se lanza en las operaciones realizadas en la transacción.
Nombre	itemsToFrame (short int command)
Descripción	Empaqueta los datos de los elementos en una trama.
Nombre	getEndPointID()
Descripción	Devuelve el identificador del EndPoint.
Nombre	setEndPointID(unsignedlongnewEndPointID)
Descripción	Cambia el identificador del EndPoint.
Nombre	readItem (ODBCItemDetails* item, ODBCHandler * handler)
Descripción	Permite leer un conjunto de valores a partir de los nombres de los elementos.
Nombre	writeItem (ODBCItemDetails * item,ODBCHandler* handler)
Descripción	Permite escribir un conjunto de valores a partir de los nombres de los elementos.
Nombre	ReadHandler (unsigned char* buffer, unsigned long size, unsigned long error)
Descripción	Handler invocado cuando finaliza la lectura.
Nombre	writeHandler(unsigned long bytesTransfered, unsigned long error)
Descripción	Handler invocado cuando finaliza la escritura.
Nombre	connectHandler(unsignedlong error)
Descripción	Handler invocado cuando finaliza la conexión.
Nombre	disconnectHandler(unsignedlong)
Descripción	Handler invocado cuando finaliza la desconexión.

3.5.2 Descripción de la clase ODBCBlock.

Esta clase hereda de la clase Block que proporciona la biblioteca DriverCore.

Tabla 2. ***Descripción de la clase ODBCBlock***

Nombre: ODBCBlock	
Tipo de Clase: Controladora	
Atributo	**Tipo**
_data	ODBCItemDetails
_isWriteMasked	bool
_addressMasked	bool
Funcionalidades:	
Nombre	ODBCBlock(Device* device, unsigned long blockStart, unsigned long blockEnd, bool forRead, VarLinkVector* container)
Descripción	Constructor de la clase ODBCBlock.
Nombre	~ODBCBlock()
Descripción	Destructor de la clase ODBCBlock.
Nombre	readInteger(IProtocolAddress* address, DeviceVarType varType, TDateTime* deviceTimeStamp, unsigned short* quality)
Descripción	Lee un valor entero del bloque a partir de un IProtocolAddress y de un tipo en el dispositivo.
Nombre	readFloat(IProtocolAddress* address, DeviceVarType varType, TDateTime* deviceTimeStamp, unsigned short* quality)
Descripción	Lee un valor flotante del bloque a partir de un IProtocolAddress y de un tipo en el dispositivo.
Nombre	readText(IProtocolAddress* address, unsigned long arraySize, char* value, TDateTime* deviceTimeStamp, unsigned short* quality)
Descripción	El método escribe una cadena en el bloque a partir de un IProtocolAddress.
Nombre	readIntVector(IProtocolAddress* address, DeviceVarType varType, unsigned long arraySize, long long* value, TDateTime* deviceTimeStamp, unsigned short* quality)
Descripción	Lee un arreglo de ordinales a partir de un IProtocolAddress.
Nombre	readFloatVector(IProtocolAddress* address, DeviceVarType

	varType, unsigned long arraySize, double* value, TDateTime* deviceTimeStamp, unsigned short* quality)
Descripción	Lee un arreglo de flotantes a partir de un IProtocolAddress.
Nombre	writeInteger(IProtocolAddress* address, DeviceVarType varType, long long value)
Descripción	Escribe un valor entero en el bloque a partir de un IProtocolAddress y de un tipo en el dispositivo.
Nombre	writeFloat(IProtocolAddress* address, DeviceVarType varType, double value)
Descripción	Escribe un valor flotante en el bloque a partir de un IProtocolAddress y de un tipo en el dispositivo.
Nombre	writeMaskedInteger(IProtocolAddress* address, DeviceVarType varType, long long value, long long mask)
Descripción	Permite modificar bits específicos en el bloque a partir de un ProtocolAddress y de un tipo en el dispositivo. El parámetro mask determina cuales bits serán modificados.
Nombre	writeIntVector(IProtocolAddress* address, DeviceVarType varType, unsigned long arraySize, long long* value)
Descripción	Permite modificar un arreglo de ordinales en el bloque a partir de un IProtocolAddress.
Nombre	writeFloatVector(IProtocolAddress* address, DeviceVarType varType, unsigned long arraySize, double* value)
Descripción	Permite modificar un arreglo de flotantes en el bloque a partirde un IProtocolAddress.
Nombre	writeText(IProtocolAddress* address, unsigned long arraySize, char* value)
Descripción	Escribe una cadena en el bloque a partir de un IProtocolAddress.
Nombre	beginWrite()
Descripción	Escribe las variables en el buffer del bloque y despacha la escritura asíncrona.
Nombre	asyncMakeReadRequest()

Descripción	Responde por el despacho asíncrono de la solicitud de lectura. Debe enviar el mensaje de solicitud y retornar inmediatamente y cuando la respuesta a la lectura este lista debe actualizar el valor de readResult y de forma asíncrona llamar a asyncReadCompleted.
Nombre	asyncMakeWriteRequest()
Descripción	Responde por el despacho asíncrono de la solicitud de escritura. Debe enviar el mensaje de escritura y retornar inmediatamente y cuando la respuesta a la escritura este lista debe actualizar el valor de writeResult y de forma asíncrona llamar a asyncWriteCompleted.
Nombre	epochTime()
Descripción	Devuelve la cantidad de milisegundos transcurridos desde el comienzo de la época Unix hasta el instante de tiempo en que se invoca al método.
Nombre	readHandler(unsigned long error)
Descripción	Handler invocado cuando concluye la operación de lectura.
Nombre	writeHandler(unsigned long error)
Descripción	Handler invocado cuando concluye la operación de escritura.

3.5.3 Descripción de la clase ODBCDevice.

Esta es la clase que permite modelar las características comunes de los dispositivos. Esta clase hereda de la clase Device que brinda la biblioteca DriverCore.

Tabla 3. ***Descripción de la clase ODBCDevice.***

Nombre: ODBCDevice	
Tipo de Clase: Controladora	
Atributo	**Tipo**
_endPoint	ODBCEndPoint *
_snifferHandler	SnifferHandler
_provider	ITransportProvider*
_library	DriversLibrary*
_startBlockAddress	char*
Funcionalidades:	

Nombre	ODBCDevice(unsigned long devScadaId, char* transferBuffer, unsigned long bufferSize, ITransportProvider*provider, SnifferHandler nsnifferHandler, IConnectionString *&cnxStr)
Descripción	Constructor de la clase ODBCDevice.
Nombre	~ODBCDevice()
Descripción	Destructor de la clase ODBCDevice
Nombre	createBlock(int blockStart, int blockEnd, bool forRead, VarLinkVector* varsVector)
Descripción	Responde por la creación de un Bloque de variables. Se invoca cuando es necesario en makeBlocks(). Devuelve un puntero a la instancia de Block creada a partir de los parámetros.
Nombre	createIProtocolAddress(const char* address, bool forRead)
Descripción	Crea una intancia de IProtocolAddress a partir de la representación textual de la dirección dada por el parámetro address. Este método se llama para las direcciones de lectura y de escritura de cada variable cada vez que se enlazan las variables.
Nombre	startBlock(IProtocolAddress* address1, DeviceVarType varType,unsigned long arraySize, bool forRead, unsigned long addressSize)
Descripción	Establece una dirección como dirección de comienzo de un bloque.
Nombre	fitInBlock(IProtocolAddress* address2, DeviceVarType varType, unsigned long arraySize, bool forRead, unsigned long addressSize)
Descripción	Determina si el rango de direcciones definido entre la dirección establecida por startBlock y la dirección que se define por los parámetros del método, puede alojarse en un bloque del protocolo, o sea puede ser recuperado o modificado con un solo mensaje del protocolo.
Nombre	fitInBlockSingle(IProtocolAddress* address1, IProtocolAddress* address2, DeviceVarType varType, unsigned long arraySize, bool forRead, unsigned long addressSize)
Descripción	Determina si una variable puede alojarse en un bloque. Esta determinación se hace a través de las direcciones address1 y address2 que representan las direcciones de protocolo de la menor

	y mayor de las direcciones simples de la variable respectivamente.
Nombre	getCOMMSTATE(TDateTime* dateTime)
Descripción	Devuelve el estado del dispositivo de forma individual.
Nombre	compare(IProtocolAddress* address1, IProtocolAddress* address2)
Descripción	Compara dos direcciones de protocolo de acuerdo a las reglas del protocolo.
Nombre	getEndPoint()
Descripción	Devuelve un apuntador al EndPoint.
Nombre	getTransportProvider()
Descripción	Devuelve el transportProvider

3.5.4 Descripción de la clase ODBCDriver.

Esta clase hereda de la clase Driver que brinda la biblioteca DriverCore la cual posee las características y comportamientos comunes de los manejadores. Implementa el concepto de Manejador SQL.

Tabla 4. ***Descripción de la clase ODBCDriver.***

Nombre: ODBCDriver	
Tipo de Clase: Controladora	
Atributo	**Tipo**
_connectionString	IConnectionString *
library	DriversLibrary*
provider	ITransportProvider*
Funcionalidades:	
Nombre	ODBCDriver(SCADA::Driver::ReadHandler readHandler,SCADA::Driver::WriteHandler writeHandler, SCADA::Driver::SnifferHandler nsnifferHandler)
Descripción	Constructor de la clase ODBCDriver.
Nombre	~ODBCDriver()
Descripción	Destructor de la clase ODBCDriver

Nombre	internalCreateDevice(unsigned long devScadaId,char* transferBuffer, unsigned long bufferSize)
Descripción	Responde por la creación de una instancia de Device.
Nombre	getErrorString(unsigned long , const char**)
Descripción	Permite obtener informacion textual comprensible a partir de un código de error del manejador.
Nombre	setDataSourceName(BaseClass* base, std::string dsn)
Descripción	Establece el valor del DSN(Data Sorce Name).
Nombre	getDataSourceName(BaseClass* base)
Descripción	Obtiene el valor del DSN(Data Sorce Name).
Nombre	getTimeOut(BaseClass* base)
Descripción	Devuelve el valor del Timeout.
Nombre	setTimeOut(BaseClass* base, long long loginTimeout)
Descripción	Establece el valor de loginTimeout.
Nombre	getUserName(BaseClass* base)
Descripción	Devuelve el valor del userName.
Nombre	setUserName(BaseClass* base, std::string userName)
Descripción	Establece el valor del userName.
Nombre	getPasswd(BaseClass* base)
Descripción	Devuelve el valor de la contraseña.
Nombre	getCOMMSTATE(TDateTime* dateTime)
Descripción	Devuelve el estado de la comunicación.

3.5.5 Descripción de la clase ODBCProtocolAdress.

La clase ODBCProtocolAddress encapsula el concepto de dirección de protocolo. Hereda de la clase IProtocolAddress que brinda la biblioteca DriverCore para el desarrollo de manejadores. En esta se define la validez de las direcciones de lectura.

Tabla 5. ***Descripción de la clase ODBCProtocolAdress.***

Nombre: ODBCProtocolAdress

Tipo de Clase: Controladora	
Atributo	**Tipo**
_forRead	bool
_device	ODBCDevice*
_varAddress	char*
Funcionalidades:	
Nombre	ODBCProtocolAddress(ODBCDevice* dev, const char* address, bool forRead)
Descripción	Constructor de la clase ODBCProtocolAdress.
Nombre	~ODBCProtocolAddress()
Descripción	Destructor de la clase ODBCProtocolAdress.
Nombre	getValid()
Descripción	Retorna un valor booleano que expresa si la dirección es válida o no de acuerdo a las reglas específicas del protocolo.
Nombre	getVarAddress()
Descripción	Retorna la dirección de la variable.

3.5.6 Descripción de la clase ODBCConnect.

La clase ODBCConnect es la clase encargada de gestionar las consultas SQL y la conexión mediante ODBC al gestor de bases de datos. Hereda de la clase IDataBaseConnect y es la clase que utiliza directamente la biblioteca otl.

Tabla 6. ***Descripción de la clase ODBCConnect.***

Nombre: ODBCConnect	
Tipo de Clase: Controladora	
Atributo	**Tipo**
_dbConnect	otl_connect
Funcionalidades:	
Nombre	ODBCConnect(IConnectionString *& connectStr)
Descripción	Constructor de la clase ODBCConnect.
Nombre	~ODBCConnect()
Descripción	Destructor de la clase ODBCConnect

Nombre	isConnectionStringModified()
Descripción	Establece el connectionString.
Nombre	connect()
Descripción	Conexión al Driver ODBC.
Nombre	disconnect()
Descripción	Desconexión al Driver ODBC.
Nombre	getTimeOut()
Descripción	Función utilizada para saber el tiempo de espera antes de notificar un errorde connexion.
Nombre	getDSN()
Descripción	Posibilita obtener el nombre del DSN
Nombre	readSQL(const char* query, char *& strValue, unsigned buffer) throw(exception::ReadError)
Descripción	Implementa la lectura de un valor de tipo cadena dada la consulta SQL.
Nombre	readSQL(const char* query, long long & iValue, unsigned buffer) throw(exception::ReadError)
Descripción	Implementa la lectura de un valor de tipo int dada la consulta SQL
Nombre	readSQL(const char* query, double & dValue, unsigned buffer) throw(exception::ReadError)
Descripción	Implementa la lectura de un valor de tipo double dada la consulta SQL.
Nombre	writeSQL(const char* query, const char *strValue)throw (exception::WriteError)
Descripción	Implementa la escritura dada la consulta y el valor a escribir.
Nombre	writeSQL(const char* query, long long llValue) throw (exception::WriteError)
Descripción	Implementa la escritura dada la consulta y el valor a escribir.
Nombre	writeSQL(const char* query, double *dValue) throw (exception::WriteError)
Descripción	Implementa la escritura dada la consulta y el valor a escribir.

Nombre	appendFormat(const char* query, const char* typeFormat)
Descripción	Le adiciona a la sentencia sql update el formato del valor adicionar.

3.5.7 Descripción de la clase ODBCTransport.

La clase ODBCTransport implementa el concepto de capa de transporte entre el manejador y la base de datos. Esta clase hereda de ITransport.

Tabla 7. ***Descripción de la clase ODBCTransport.***

Nombre: ODBCTransport	
Tipo de Clase: Controladora	
Atributo	**Tipo**
_dbConnect	IDataBaseConnect *
_connected	bool
Funcionalidades:	
Nombre	ODBCTransport(IConnectionString *&connectStr)
Descripción	Constructor de la clase ODBCTransport.
Nombre	~ODBCTransport()
Descripción	Destructor de la clase ODBCTransport.
Nombre	getTransportId()
Descripción	Retorna un código numérico que identifica el tipo de transporte implementado.
Nombre	setHostName(const char* hostName)
Descripción	Establece la dirección a conectarse.
Nombre	getHostName()
Descripción	Devuelve la dirección utilizada en la conexión.
Nombre	getTimeOut()
Descripción	Función utilizada para saber el tiempo de espera para recibir el mensaje.
Nombre	disconnect(ITransportHandler* handler)
Descripción	Permite desconectarse del dispositivo.
Nombre	connect(unsigned long timeOut, ITransportHandler* handler)
Descripción	Establece la conexiónasíncrona con el dispositivo.

Nombre	write(ODBCItemDetails* buffer, ITransportHandler* handler)
Descripción	La función escribe de forma asíncronacierta cantidad de bytes al transporte.
Nombre	read (ODBCItemDetails* buffer, unsigned long timeOut, ITransportHandler* handler)
Descripción	Lee de forma asíncrona cierta cantidad de bytes desde un servidor.
Nombre	readOperation(void *callback)
Descripción	Método a ser ejecutado en otro contexto para la realización de lecturas asíncronamente.
Nombre	writeOperation(void *callback)
Descripción	Método a ser ejecutado en otro contexto para la realización de escrituras asíncronamente.
Nombre	read(unsigned char* buffer, unsigned long size,unsigned long timeOut, ITransportHandler* handler)
Descripción	Lee de forma asíncrona cierta cantidad de bytes del transporte concreto (serial , tcp o udp).
Nombre	readUntil(unsigned char* buffer,unsigned long bytesAdditional, unsigned long timeOut,ITransportHandler* handler)
Descripción	Lee asíncronamente bytes disponibles, hasta que se cumpla la condición de parada definida por matchCondition en la instancia del handler de lectura. Luego de finalizada la lectura es invocado el handler de lectura correspondiente a la instancia pasada por parámetros.
Nombre	write(unsigned char* buffer,unsigned long size, ITransportHandler* handler)
Descripción	Escribe de forma asíncrona cierta cantidad de bytes al transporte.

3.6 Estándar de Codificación.

El uso de un estándar de codificación permite establecer una serie de reglas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura y comprensión. Se definen estándares de codificación porque un estilo de programación homogéneo en un proyecto desarrollado permite que otros desarrolladores lo puedan entender en menos tiempo y que el código

en consecuencia sea mantenible. El manejador ODBC está desarrollado teniendo en cuenta estos aspectos. A continuación se describen algunos de los elementos fundamentales que componen el estándar utilizado:

- **Nombre de los ficheros.**

  Todos los ficheros .h y .cpp se nombrarán comenzando con mayúscula. Si el nombre está formado por más de una palabra cada una debe comenzar con mayúscula. Todas las demás letras deben escribirse en minúscula.

- **Clases.**

  Todas las clases se nombrarán comenzando con mayúscula. Si el nombre está formado por más de una palabra cada una debe comenzar con mayúscula. Todas las demás letras deben escribirse en minúscula. En un fichero .h solo debe ser definida una clase en caso de que sea necesario y a dicha definición le debe corresponder un fichero .cpp donde se implementen las funcionalidades de la clase definida.

- **Métodos.**

  A excepción de los constructores y destructores todos los métodos deben empezar con minúscula. Cada palabra que forme parte del nombre se escribirá comenzando con mayúscula, por ejemplo:
  nuevoMetodo.

- **Condiciónales y ciclos.**

  Todas las expresiones condiciónales y los ciclos poseerán las llaves de inicio y cierre del campo de acción de la expresión, aunque la misma posea una sola línea de código.

3.7 Direcciones de Variables.

Las direcciones de variables del manejador ODBC están constituidas por sentencias SQL a partir de las cuales solo se espera obtener un único valor. En el caso de las direcciones de lectura la sentencia SQL a ejecutar es un Select_____From____Where_____, mientras que las direcciones de escritura deben ser una sentencia de tipo Update_____ Set____Where_____.

3.8 Bloques de Direcciones.

Los bloques se construyen dinámicamente por los manejadores cada vez que se asocian variables y cada vez que se efectúan operaciones de escritura. En este caso los bloques de direcciones estarán compuestos por solo una variable de dirección.

3.9 Diagrama de despliegue.

El Diagrama de Despliegue es un tipo de diagrama que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. Modela la topología del hardware donde se ejecuta el sistema.

El siguiente diagrama representa como será desplegado el manejador ODBC en términos de nodos. El nodo Manejador ODBC, representa el ordenador donde se encuentra funcionando el sistema de supervisión y control con el manejador ODBC, mientras que el nodo Servidor de Base de Datos representa el ordenador donde se encuentran el servidor de BD al cual el sistema accede utilizando el manejador, a través del protocolo TCP/IP.

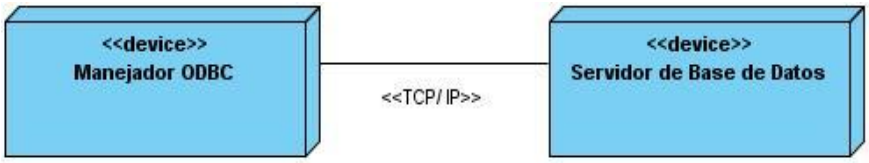


Figura 3. Diagrama de Despliegue.

3.10 Diagrama de componentes.

El diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos. Los componentes representan todos los tipos de elementos software que se utilizan en la fabricación de aplicaciones informáticas. En él se pueden situar librerías, tablas, archivos, ejecutables y documentos que formen parte del sistema.

En el siguiente diagrama se puede apreciar como el manejador ODBC utiliza las bibliotecas DriverCore, TransportProvider y otl versión 4 en su implementación.

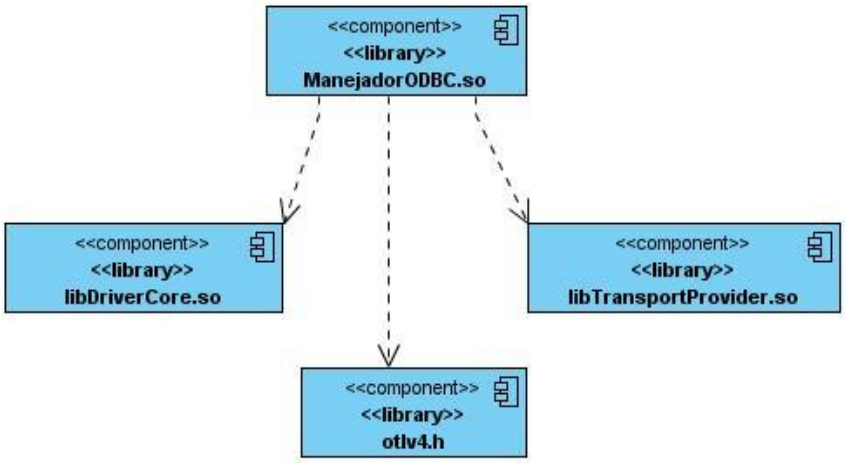


Figura 4. Diagrama de Componentes.

3.11 Conclusiones.

En este capítulo se abordó sobre las características del manejador referentes a su diseño e implementación, presentando una detallada descripción de cada una de las clases más importantes implementadas a fin de lograr el objetivo trazado inicialmente. Aspecto que contribuye, indudablemente, a una mejor comprensión, utilización y posterior mantenimiento del código del manejador desarrollado.

Capítulo 3: Pruebas del Sistema

Introducción

Las pruebas de software son procesos que permiten verificar la calidad de un producto. Se utilizan para identificar errores de implementación o usabilidad de los programas computacionales. Es un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones del diseño y de la codificación.

3.1 Pruebas

Las Pruebas que se le realizan a los productos de software tienen como objetivo:

- Verificar la interacción entre objetos.
- Verificar la interacción apropiada de todos los componentes del software.
- Verificar que todos los requerimientos hayan sido implementados correctamente.
- Identificar y asegurar que los defectos se hayan atendido y resuelto antes del despliegue del software.

Cualquier producto software puede ser probado de dos formas fundamentales, a través de:

- Pruebas de Caja Negra: En la prueba de la caja negra, se hace uso de los casos de prueba con el objetivo de demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta. Este tipo de prueba se realiza con el objetivo de detectar errores tales como:
    - Funciones incorrectas o ausentes.
    - Errores en estructuras de datos o en accesos a bases de datos externas.
    - Errores de rendimiento.
    - Errores de inicialización y de terminación.

- Pruebas de Caja Blanca: En la prueba de caja blanca se realiza un examen minucioso de los detalles procedimentales, comprobando los caminos lógicos del

programa, comprobando los bucles y condiciones, y examinado el estado del programa en varios puntos. Esta prueba supone un estudio más exhaustivo del software.

Al manejador ODBC se le han realizado pruebas de Caja Negra para comprobar su operatividad, obviando el comportamiento interno y la estructura del programa.

3.2 Herramientas de Prueba.

Para llevar a cabo el proceso de pruebas al Manejador ODBC, con el fin de comprobar que este cumpla con las funcionalidades previstas, se hizo uso de herramientas tales como:

- Recolector Gráfico. Este recolector es un componente que brinda la posibilidad de cargar las bibliotecas, crear Manejador, crear Dispositivo, crear, modificar y eliminar variables tanto de lectura como de escritura, configurar manejador, configurar dispositivo. Muestra la calidad de las variables, el bloque al que pertenecen y el momento en que se realizó la lectura de las mismas, de forma gráfica.
- Gestor de BD PostgreSQL y BD SQLite, con el fin de comprobar la capacidad de conexión del manejador, así como la correcta escritura y lectura de valores a partir de las direcciones de variables.

Las pruebas se realizan tanto en Windows como en Linux, con el propósito de comprobar la portabilidad del manejador.

3.3 Tipos de Datos de ODBC

Para la realización de las pruebas un elemento fundamental a tener en cuenta son los tipos de datos que soporta el estándar ODBC, de modo que pueda definirse los tipos de datos que pueden o no ser reconocidos en el proceso de recolección de datos.

Los tipos de datos que soporta son:

- CHAR
- DATE
- DOUBLE
- FLOAT
- INTEGER
- NUMERIC
- DECIMAL
- REAL

- SERIAL
- SMALLINT
- TIME
- TIMESTAMP
- TINYINT
- VARCHAR

3.4 Resultado de las pruebas.

Se realizan pruebas para comprobar la capacidad del manejador de recolectar el valor de los campos de las BD conectadas al Sistema de Supervisión, para ello se realiza la lectura de 100 variables para cada una de las BD, en cada uno de los Sistemas Operativos.

Las pruebas ejecutadas son las siguientes:

- Leer tipo de dato Entero
- Leer tipo de dato Real
- Leer tipo de dato String

3.4.1 SQLite.

Caso de Prueba 1: Leer Tipo de Dato Entero

Descripción: Con la ejecución de este caso de prueba se puede obtener el valor de una variable entera y la calidad de la lectura con un valor de 192, lo cual significa que la lectura fue exitosa.

Flujo Central:

1. En la ventana “Capturas” del Recolector Gráfico ejecutar clic derecho y seleccionar la opción de Crear Variable.
2. En Nombre definir un nombre para la variable, en Periodo definir como valor 1000, en Dirección (lectura) escribir una sentencia SQL de tipo Select, en tipo de dato seleccionar INT. Seleccionar aceptar.
3. En el menú Operaciones seleccionar la opción Comenzar Recolección.

Condiciones de Ejecución:

Debe estar instalado el driver ODBC para SQLite, creados el manejador y el dispositivo y establecido correctamente el Nombre de la Fuente de Datos.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de las Pruebas	Obser-vaciones
Se pide leer el valor de la variable con dirección SELECT num From yeni WHERE rowid=2		Se espera obtener el valor de la variable “num” y que la calidad sea de 192	Se visualiza un valor y la calidad es de 192	Se verifica en la BD que el valor devuelto corresponde a la variable “num” donde rowid tiene valor 2.
	Se pide leer el valor de la variable con dirección SELECT “num” From yeni WHERE rowid=21	Se espera obtener el valor de la variable “num” y que la calidad sea de 192	Devuelve como valor de la variable 0	En la BD el campo rowid no tiene valor 21 por lo que se muestra como valor 0 indicando que no se encuentra un valor para “num” que corresponda a rowid= 21

Caso de Prueba 2: Leer Tipo de Dato Real

Descripción: Con la ejecución de este caso de prueba se puede obtener el valor de una variable real y la calidad de la lectura con un valor de 192, lo cual significa que la lectura fue exitosa.

Flujo Central:

1. En la ventana “Capturas” del Recolector Gráfico ejecutar clic derecho y seleccionar la opción de Crear Variable.

2. En Nombre definir un nombre para la variable, en Periodo definir como valor 1000, en Dirección (lectura) escribir una sentencia SQL de tipo Select, en tipo de dato seleccionar REAL. Seleccionar aceptar.

3. En el menú Operaciones seleccionar la opción Comenzar Recolección.

Condiciones de Ejecución:

Debe estar instalado el driver ODBC para SQLite, creados el manejador y el dispositivo y establecido correctamente el Nombre de la Fuente de Datos.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de las Pruebas	Obser-vaciones
Se pide leer el valor de la variable con dirección SELECT coma From yeni WHERE rowid=7		Se espera obtener el valor de la variable "coma" y que la calidad sea de 192	Se visualiza un valor y la calidad es de 192	Se verifica en la BD que el valor devuelto corresponde a la variable "coma" donde rowid tiene valor 2.
	Se pide leer el valor de la variable con dirección SELECT coma From yeni WHERE rowid=21	Se espera obtener el valor de la variable "coma" y que la calidad sea de 192	Devuelve como valor de la variable 0	En la BD el campo rowid no tiene valor 21 por lo que se muestra como valor 0 indicando que no se encuentra un valor para "coma" que corresponda a rowid= 21

Caso de Prueba 3: Leer Tipo de Dato String

Descripción: Con la ejecución de este caso de prueba se puede obtener el valor de una variable string y la calidad de la lectura con un valor de 192, lo cual significa que la lectura fue exitosa.

Flujo Central:

1. En la ventana “Capturas” del Recolector Gráfico ejecutar clic derecho y seleccionar la opción de Crear Variable.
2. En Nombre definir un nombre para la variable, en Periodo definir como valor 1000, en Dirección (lectura) escribir una sentencia SQL de tipo Select, en tipo de dato seleccionar STRING. Seleccionar aceptar.
3. En el menú Operaciones seleccionar la opción Comenzar Recolección.

Condiciones de Ejecución:

Debe estar instalado el driver ODBC para SQLite, creados el manejador y el dispositivo y establecido correctamente el Nombre de la Fuente de Datos.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de las Pruebas	Obser-vaciones
Se pide leer el valor de la variable con dirección SELECT caract From yeni WHERE rowid=7		Se espera obtener el valor de la variable “caract” y que la calidad sea de 192	Se visualiza un valor que corresponde con el de la variable “caract” y la calidad es de 192	Se verifica en la BD que el valor devuelto corresponde a la variable “caract” donde rowid tiene valor 2.
	Se pide leer el valor de la variable con dirección SELECT caract From yeni WHERE	Se espera obtener el valor de la variable “caract” y que la calidad sea de 192	Devuelve como valor de la variable 0	En la BD el campo rowid no tiene valor 21 por lo que se muestra como valor 0

	rowid=21			indicando que no se encuentra un valor para “caract” que corresponda a rowid= 21

3.4.2 PostgreSQL.

Caso de Prueba 1: Leer Tipo de Dato Entero

Descripción: Con la ejecución de este caso de prueba se puede obtener el valor de una variable entera y la calidad de la lectura con un valor de 192, lo cual significa que la lectura fue exitosa.

Flujo Central:

1. En la ventana “Capturas” del Recolector Gráfico ejecutar clic derecho y seleccionar la opción de Crear Variable.
2. En Nombre definir un nombre para la variable, en Periodo definir como valor 1000, en Dirección (lectura) escribir una sentencia SQL de tipo Select, en tipo de dato seleccionar INT. Seleccionar aceptar.
3. En el menú Operaciones seleccionar la opción Comenzar Recolección.

Condiciones de Ejecución:

Debe estar instalado el driver ODBC para PostgreSQL, creados el manejador y el dispositivo y establecido correctamente el Nombre de la Fuente de Datos.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de las Pruebas	Obser-vaciones
Se pide leer el valor de la variable con dirección SELECT entero From yeni WHERE char=’m’		Se espera obtener el valor de la variable “entero” y que la calidad sea de 192	Se visualiza un valor y la calidad es de 192	Se verifica en la BD que el valor devuelto corresponde a la variable “entero” donde char

				tiene valor 'm'.
	Se pide leer el valor de la variable con dirección SELECT entero From yeni WHERE char='c'	Se espera obtener el valor de la variable "entero" y que la calidad sea de 192	Devuelve como valor de la variable 0	En la BD el campo char no tiene valor 'c' por lo que se muestra como valor 0 indicando que no se encuentra un valor para "entero" que corresponda a char='c'

Caso de Prueba 2: Leer Tipo de Dato Real

Descripción: Con la ejecución de este caso de prueba se puede obtener el valor de una variable real y la calidad de la lectura con un valor de 192, lo cual significa que la lectura fue exitosa.

Flujo Central:

1. En la ventana "Capturas" del Recolector Gráfico ejecutar clic derecho y seleccionar la opción de Crear Variable.
2. En Nombre definir un nombre para la variable, en Periodo definir como valor 1000, en Dirección (lectura) escribir una sentencia SQL de tipo Select, en tipo de dato seleccionar REAL. Seleccionar aceptar.
3. En el menú Operaciones seleccionar la opción Comenzar Recolección.

Condiciones de Ejecución:

Debe estar instalado el driver ODBC para PostgreSQL, creados el manejador y el dispositivo y establecido correctamente el Nombre de la Fuente de Datos.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de las Pruebas	Obser-vaciones
Se pide leer el valor de la variable con dirección SELECT double From yeni WHERE char='m'		Se espera obtener el valor de la variable "double" y que la calidad sea de 192	Se visualiza un valor que corresponde con el de la variable "double" y la calidad es de 192	Se verifica en la BD que el valor devuelto corresponde a la variable "double" donde char tiene valor 'm'.
	Se pide leer el valor de la variable con dirección SELECT double From yeni WHERE char='c'	Se espera obtener el valor de la variable "double" y que la calidad sea de 192	Devuelve como valor de la variable 0	En la BD el campo char no tiene valor 'c' por lo que se muestra como valor 0 indicando que no se encuentra un valor para "double" que corresponda a char='c'

Caso de Prueba 3: Leer Tipo de Dato String

Descripción: Con la ejecución de este caso de prueba se puede obtener el valor de una variable string y la calidad de la lectura con un valor de 192, lo cual significa que la lectura fue exitosa.

Flujo Central:

1. En la ventana "Capturas" del Recolector Gráfico ejecutar clic derecho y seleccionar la opción de Crear Variable.

2. En Nombre definir un nombre para la variable, en Periodo definir como valor 1000, en Dirección (lectura) escribir una sentencia SQL de tipo Select, en tipo de dato seleccionar STRING. Seleccionar aceptar.

3. En el menú Operaciones seleccionar la opción Comenzar Recolección.

Condiciones de Ejecución:

Debe estar instalado el driver ODBC para PostgreSQL, creados el manejador y el dispositivo y establecido correctamente el Nombre de la Fuente de Datos.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de las Pruebas	Obser-vaciones
Se pide leer el valor de la variable con dirección SELECT otra From yeni WHERE char='c'		Se espera obtener el valor de la variable "otra" y que la calidad sea de 192	Se visualiza un valor que corresponde con el de la variable "otra" y la calidad es de 192	Se verifica en la BD que el valor devuelto corresponde a la variable "otra" donde char tiene valor 'm'.
	Se pide leer el valor de la variable con dirección SELECT otra From yeni WHERE char='c'	Se espera obtener el valor de la variable "otra" y que la calidad sea de 192	Devuelve como valor de la variable 0	En la BD el campo char no tiene valor 'c' por lo que se muestra como valor 0 indicando que no se encuentra un valor para "otra" que corresponda a char='c'

3.5 Conclusiones.

Al concluir este capítulo se puede afirmar que el manejador desarrollado cumple con los requisitos previstos, ya que al realizársele pruebas de funcionalidad se han obtenido resultados satisfactorios que demuestran el correcto funcionamiento del mismo.

Conclusiones

Al concluir el trabajo de diploma “Manejador ODBC para Sistema de Supervisión y Control de Procesos Automatizados”, se da cumplimiento a las tareas de investigación definidas así como al objetivo general propuesto para la investigación, desarrollándose un manejador basado en el estándar ODBC que permite el acceso a datos contenidos en diferentes Sistemas Gestores de Base de Datos.

Se cumple también con otros aspectos definidos para el desarrollo de dicho manejador, entre los cuales sobresalen:

- El manejador obtenido cumple con los requisitos planteados de intercambio de información entre el SCADA y Sistemas Gestores de Base de Datos.
- El proceso de desarrollo de este manejador ODBC permite la reutilización de código ya existente.
- La utilización de bibliotecas de software libre.
- El manejador desarrollado es portable.

El desarrollo del manejador y su utilización en aplicaciones como SCADA, supone una ventaja en cuanto al proceso de intercambio de datos tan necesario en este tipo de sistemas de automatización industrial ya que hace posible que se acceda a información almacenada en varios Sistemas Gestores de Base de Datos sin realizar ningún cambio en la codificación del programa.

Recomendaciones

Al concluir este trabajo de diploma se recomienda, para el desarrollo de una nueva versión del manejador basado en el estándar OBDC, que se tenga en cuenta aspectos tales como:

- Para los SGBD que requieran autenticación, esta se pueda controlar desde la aplicación, y no únicamente en la configuración del DSN en el sistema.
- Ampliar las funcionalidades del driver de manera que permita la lectura de arreglos de variables.

Referencias Bibliográficas

[1].**Autómatas Industriales.** [En línea] 2 de Marzo de 2006. [Citado el: 10 de Enero de 2010.] http://www.automatas.org/redes/scadas.htm.

[2].**Díaz, Ing. Henry Mendiburu.** [En línea] 2008. [Citado el: 14 de Enero de 2011.] www.galeon.com/hamd/pdf/scada.pdf.

[3]**Casas Guijarro, Alejandro.** *Sistema Telefónico Multilínea con Reconocimiento de voz y acceso a Base de Datos Remota.* [En línea] Septiembre de 1997. [Citado el: 5 de Febrero de 2011.] http://lorien.die.upm.es/juancho/pfcs/ACG/.

[4]**Sitio Web de Visual Paradigm.** [En línea] [Citado el: 19 de Febrero de 2010.] http://www.visual-paradigm.com/product/vpuml/.

[5] **Gómez Gallego, Juan Pablo.** *Fundamentos de la Metodología RUP.* [En línea] 16 de Septiembre de 2007. [Citado el: 25 de Febrero de 2011.] http://www.scribd.com/doc/297224/RUP.

Bibliografía

Lozano, Carlos de Castro y Romero Morales, Cristóbal. *Introducción a SCADA.* [En línea] [Citado el: 12 de Enero de 2011.] www.uco.es/grupos/eatco/automatica/ihm/descargar/scada.pdf.

Sistemas Operativos .Manejadores de Dispositivos [En línea] [Citado el: 15 de Enero de 2011.] http://serdis.dis.ulpgc.es/~a013775/asignaturas/itig-ps/www/doc/apuntes/tema10.pdf

SIMBA. ODBC. [En línea] [Citado el: 20 de Enero de 2011.] http://www.simba.com/odbc.htm.

Rodríguez, Penin Aquilino. *Sistemas SCADA.2º Edición.2007.*

Cerverón Vicente y De Ves Esther. *Desarrollo de Aplicaciones de Bases de Datos.*[En línea] 2006. [Citado el: 9 de Febrero de 2011.] http://informatica.uv.es/iiguia/2000/BD2/2_0_BD2Tema2_06.pdf

Tutorial, ODBC from C.[En línea] 2010. [Citado el: 12 de Febrero de 2011.] http://www.easysoft.com/developer/languages/c/odbc_tutorial.html.

Guide, User´s.[En línea]2004. [Citado el: 13 de Febrero de 2011.] http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.help.sdk_12.5.1.aseodbc/html/aseodbc/Odfund.htm.

UML, Lenguaje. Lenguaje UML. *Lenguaje UML.* [En línea] 23 de Abril de2010. [Citado el: 22 de Febrero de 2011.] http://techfico.blogspot.com/2010/04/lenguaje-uml_23.html.

Integrado, Entornos de Desarrollo. Desarrollo de Entornos Integrados. *Desarrollo de Entornos Integrados.* [En línea] [Citado el: 20 de Febrero de 2011.]http://petra.euitio.uniovi.es/~i1667065/HD/documentos/Entornos%20de%20Desarrollo%20Integrado.pdf

Systems, Zator. [En línea] 2010. [Citado el: 22 de Febrero de 2011.] http://www.zator.com/Cpp/E1_2.htm.

Características de C++. [En línea] [Citado el: 22 de Febrero de 2011.]https://belenus.unirioja.es/~creguia/introduccion.html.

Didact, SL. 2005. *Manual de programación del Lenguaje C++.* [En línea] 2005. [Citado el: 24 de Febrero de 2011.]http://books.google.com.cu/books?id=py9NgRWWZE4C&pg=SL26-

PA11&dq=historia+de+C%2B%2B&hl=es-419&ei=atBmTcTDCo3BtgecqL3mAw&sa=X&oi=book_result&ct=result&resnum=5&ved=0CD8Q6AEwBA#v=onepage&q=historia%20de%20C%2B%2B&f=false

Kuchin, Sergei. *Oracle, Odbc and DB2-CLI Template Library Programmer's Guide.*[En línea] 2010.[Citado el: 2 de Marzo de 2011.] http://otl.sourceforge.net/.

Menéndez Alonso, Evelyn. *Herramientas CASE para el proceso de desarrollo de Software.* [En línea]. [Citado el: 27 de Febrero de 2011.]http://www.monografias.com/trabajos73/herramientas-case-proceso-desarrollo-software/herramientas-case-proceso-desarrollo-software2.shtml

Linux/UNIX ODBC.[En Línea].[Citado el: 1 de marzo del 2011.] http://www.easysoft.com/developer/interfaces/odbc/linux.html

Valencia, María Eugenia. [En línea]. [Citado el: 26 de febrero del 2011.]http://eisc.univalle.edu.co/materias/Material_Desarrollo_Software/DISCLASES_A12.pdf

Anexos

Anexo 1 Descripción de la Clase ODBCConnectionString

Esta clase es la encargada de construir la cadena de conexión al Gestor de Bases de Datos mediante ODBC. Hereda de la Clase IConnectionString.

Tabla 8. ***Descripción de la clase ODBCConnectionString.***

Nombre: ODBCConnectionString	
Tipo de Clase: Controladora	
Atributo	**Tipo**
_dsn	Char *
_userName	Char *
_password	Char *
Funcionalidades:	
Nombre	ODBCConnectionString()
Descripción	Constructor de la clase ODBCConnectionString.
Nombre	ODBCConnectionString(constchar *dsn,const char *userName, const char *password)
Descripción	Constructor Parametrizado de la clase ODBCConnectionString.
Nombre	~ODBCConnectionString()
Descripción	Destructor de la clase ODBCConnectionString.
Nombre	getDSN()
Descripción	Método utilizado para obtener el identificador unico para el acceso a la fuente de datos
Nombre	setDSN(const char* dsn)
Descripción	Método utilizado para modificar el identificador para el acceso a la fuente de datos.
Nombre	getUserName()
Descripción	Método utilizado para obtener el nombre del usuario para el acceso a la

	base de datos.
Nombre	setUserName(const char* userName)
Descripción	Método utilizado para modificar el nombre del usuario para el acceso a la base de datos.
Nombre	getPassword()
Descripción	Método utilizado para obtener la contraseña para el acceso a la base de datos.
Nombre	setPassword(const char* passwd)
Descripción	Método utilizado para modificar la contraseña para el acceso a la base de datos.
Nombre	connectionString()
Descripción	Este método devuelve la cadena de conexión que el mandejador ODBC necesita para acceder a los datos del Gestor de Bases de Datos ya especificado.

Anexo 2 Descripción de la Clase CallbackObject

Clase utilizada para llevar a cabo las operaciones asíncronas con la base de datos.

Tabla 9. ***Descripción de la clase CallbackObject***

Nombre: CallbackObject	
Tipo de Clase: Controladora	
Atributo	**Tipo**
_buffer	ODBCItemDetails*
_handle	ITransportHandler*
_dbConnect	IDataBaseConnect*
Funcionalidades:	
Nombre	CallbackObject()
Descripción	Constructor de la clase CallbackObject
Nombre	CallbackObject(ODBCItemDetails* vBuffer, IDataBaseConnect* connection,ITransportHandler* pitHandle)
Descripción	Constructor Parametrizado de la clase CallbackObject
Nombre	~CallbackObject()

Descripción	Destructor de la clase CallbackObject
Nombre	read()
Descripción	Función utilizada para realizar la lectura asíncrona
Nombre	write()
Descripción	Función utilizada para realizar la escritura asíncrona

Anexo 3 Manual de usuario Manejador ODBC para Windows.

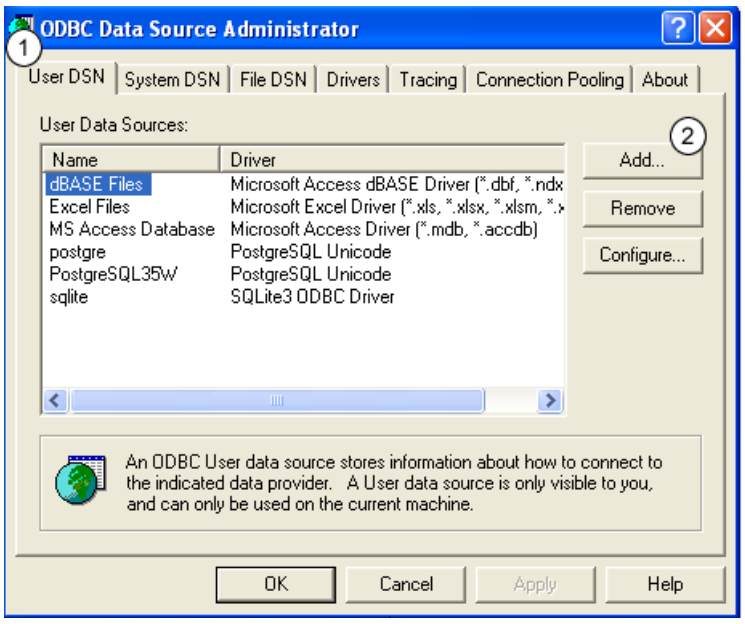
Para que sea posible el acceso desde el Sistema SCADA a datos almacenados en BD de diferentes SGBD, es necesario poseer el driver ODBC correspondiente a cada uno de esos SGBD y a la versión que se esté utilizando.

Por ejemplo, si se desea acceder a datos contenidos en una BD PostgreSQL y la versión de este SGBD que está utilizando es la 8.4 es necesario obtener e instalar el driver ODBC para PostgreSQL correspondiente a la versión 8.4 o mayor que esta. Una vez instalado el driver es necesario la creación de un DSN donde se establece la ruta y características de la conexión que se desea establecer a la BD.

Para la creación del DSN se debe acceder al Administrador de Origen de Datos de ODBC siguiendo los siguientes pasos: *Inicio > Panel de Control > Herramientas Administrativas > Orígenes de Datos (ODBC).*

Una vez ubicados en el Administrador de Origen de Datos de ODBC se llevarán a cabo los siguientes pasos:

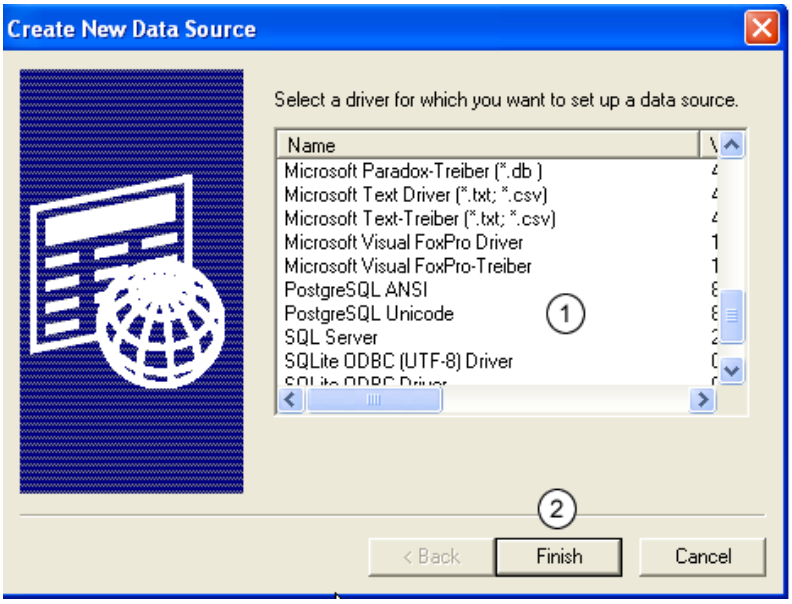
Interfaz Administrador de Origen de Datos de ODBC.



1-Acceder a la opción User DSN.

2-Acceder a la opción Add.

Interfaz para seleccionar Driver OBDC.

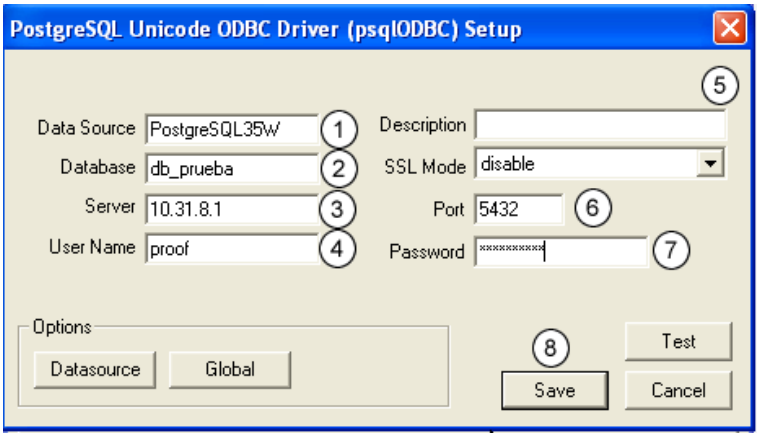


1- Seleccionar de la lista de drivers, el que se instaló y corresponde al SGBD al cual se desea acceder.

2- Acceder a la opción Finish (Finalizar).

A partir de este paso comienza la configuración del DSN teniendo en cuenta los parámetros que solicite el fabricante de cada unos de los SGBD. A continuación se muestra como ejemplo la configuración para una BD PostgreSQL.

Interfaz Configuración del DSN.



1-Opción para establecer el nombre de la conexión a la BD PostgreSQL.

2-Opción para establecer el nombre de la BD a la cual se desea acceder.

3- Opción para establecer la dirección IP donde se encuentra montado el servidor PostgreSQL.

4- Opción para definir el nombre de usuario necesario para acceder a la BD.

5- Opción para introducir una breve descripción de la conexión a realizar. (Este parámetro no es obligatorio).

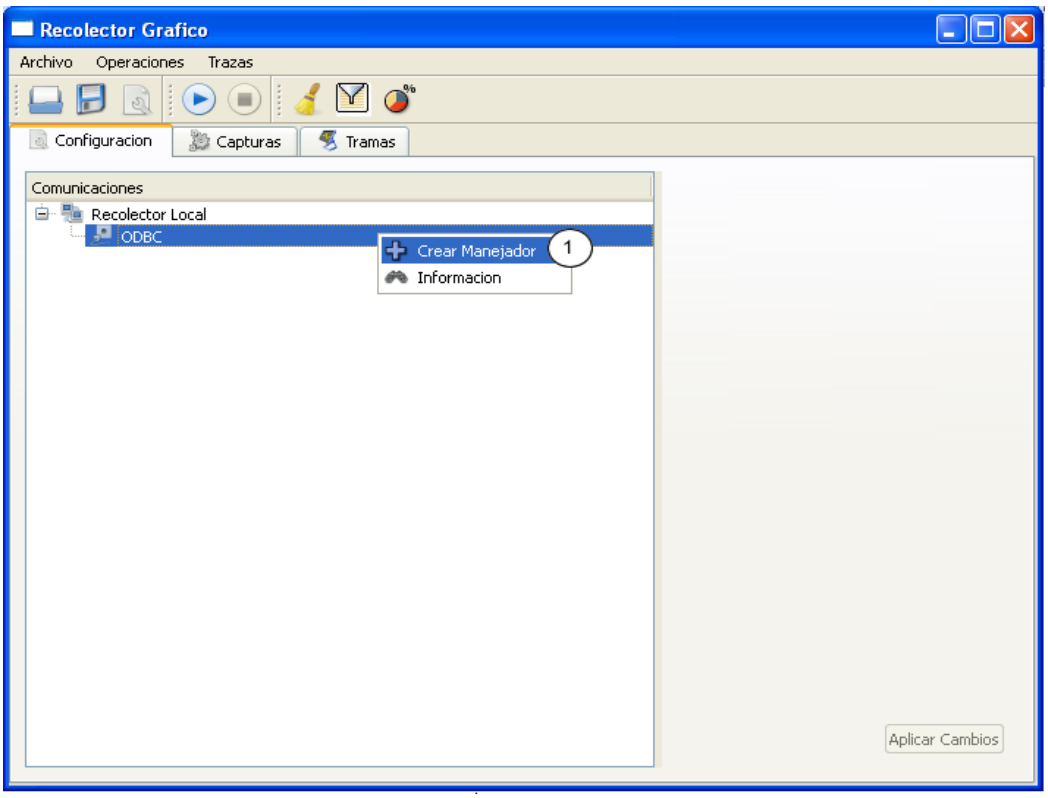
6- Opción para establecer el puerto por el cual se podrá conectar al servidor.

7- Opción para introducir la contraseña requerida para el acceso a la BD.

8- Opción para concluir y guardar la configuración del DSN creado.

Para comprobar que la instalación del driver ODBC para el SGBD seleccionado y que la configuración del DSN se realizó de forma satisfactoria es necesario acceder a la BD, realizar consultas y verificar que el resultado de estas es correcto. En este caso la recolección de datos se realizará utilizando el Recolector Gráfico. A continuación se muestra como ejemplo la configuración requerida en el Recolector para acceder a datos contenidos en una BD, en este caso PostgreSQL.

Interfaz Recolector Gráfico. Crear Manejador.



1- Opción para crear un manejador. Se llega a esta opción después de ejecutar clic derecho encima de “ODBC”.

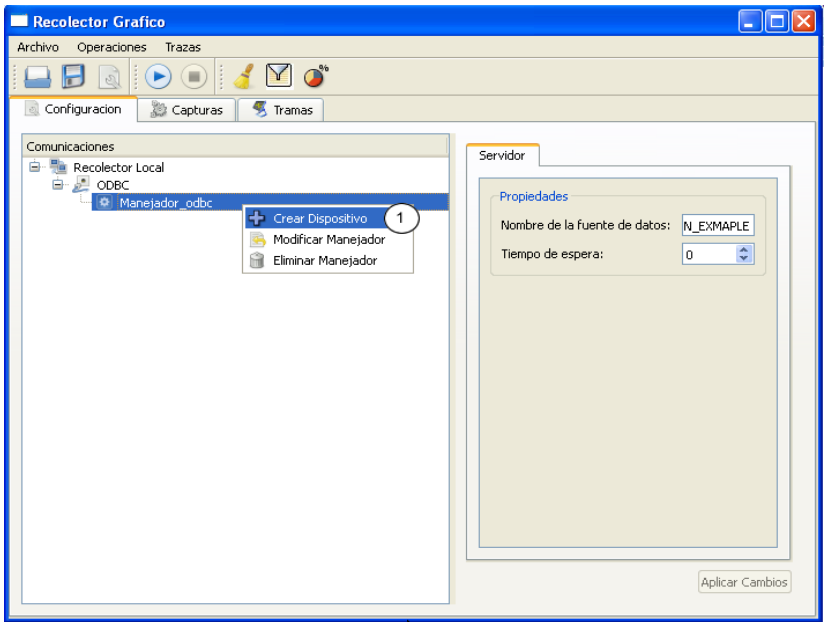
Interfaz para la creación del Manejador.

1- Opción para introducir el nombre del Manejador que se está creando.

2- Opción para introducir una breve descripción sobre el manejador. (Este campo no es obligatorio).

3- Acceder al botón Aceptar para guardar y cerrar la ventana.

Interfaz Recolector Gráfico. Crear Dispositivo.



1-Opción para crear un dispositivo. Se llega a esta opción después de ejecutar clic derecho encima del Manejador creado.

Interfaz para la creación del Dispositivo.

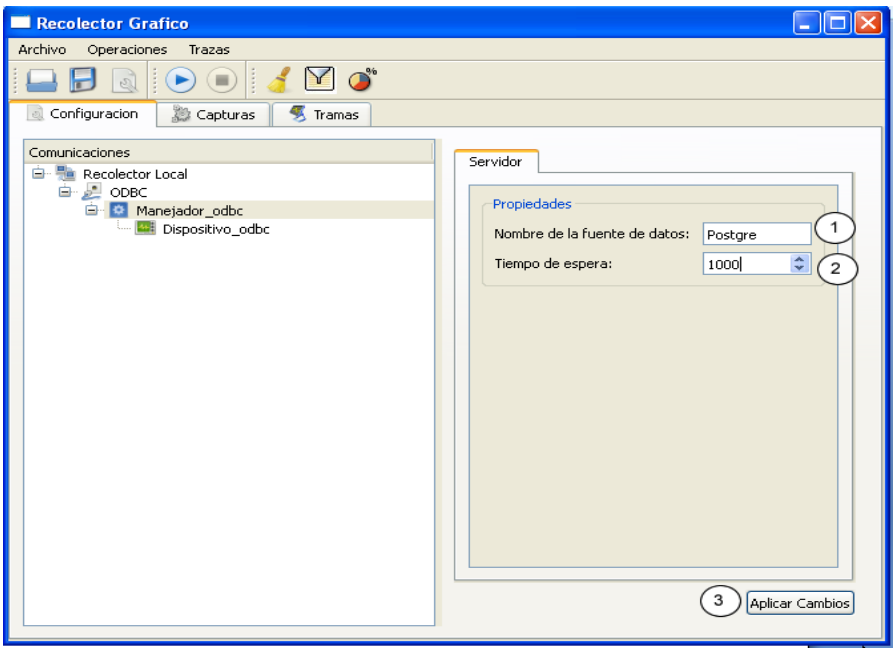
1-Opción para introducir el nombre del Dispositivo que se está creando.

2-Opcion para establecer un período. Se recomienda que el valor mínimo sea 1000.

3-Opción para introducir una breve descripción sobre el dispositivo. (Este campo no es obligatorio).

4-Acceder al botón Aceptar para guardar y cerrar la ventana.

Interfaz Recolector Gráfico. Establecer Conexión.

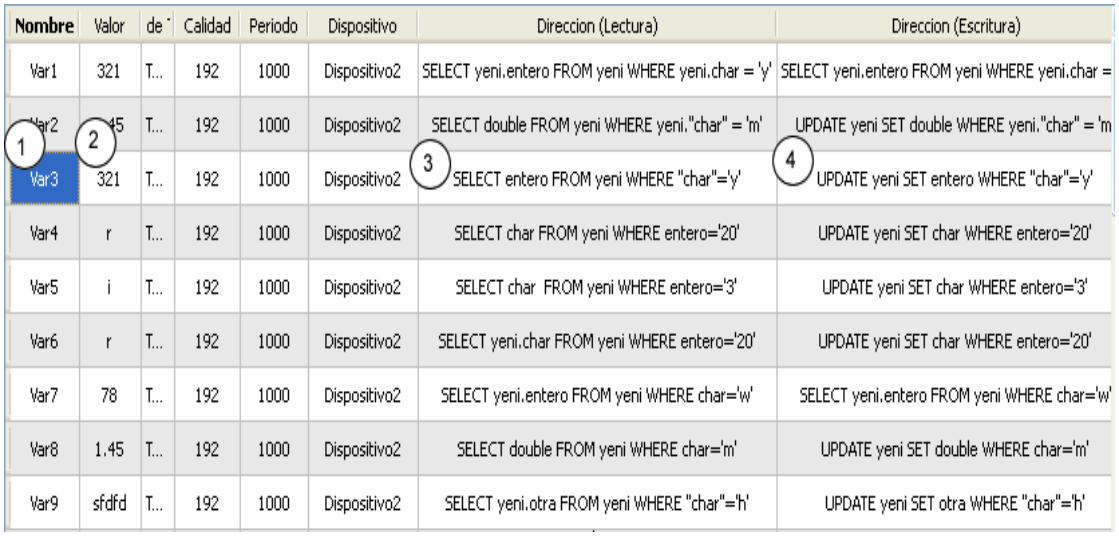


1- Opción para introducir el nombre de la fuente de datos a la que se desea conectar. Este nombre debe coincidir exactamente con el nombre introducido en el campo Data Source en la configuración del DSN. Ver ***Interfaz Configuración del DSN para una conexión a PostgreSQL***.
2- Opción para establecer un tiempo de espera de los resultados de las consultas que se realizarán.
3- Acceder al botón Aplicar Cambios para cerrar y guardar la configuración

Para verificar que la conexión a la BD se estableció correctamente se debe acceder a la pestaña “Capturas” donde, a partir de las direcciones de variables, se comprobará la validez de la conexión. En el caso del Manejador ODBC las direcciones de variables consisten en sentencias SQL ya que esta es la forma en que se puede acceder y obtener información de cualquier BD. La sintaxis SQL a utilizar es independiente del SGBD al que se desea acceder. La Dirección (Lectura) debe consistir en un SELECT, mientras que la Dirección (Escritura) en un UPDATE.

A continuación se muestra un ejemplo de direcciones de variables utilizadas en la conexión a una BD.

Interfaz Recolector Gráfico.



Nombre	Valor	de	Calidad	Periodo	Dispositivo	Direccion (Lectura)	Direccion (Escritura)
Var1	321	T...	192	1000	Dispositivo2	SELECT yeni.entero FROM yeni WHERE yeni.char = 'y'	SELECT yeni.entero FROM yeni WHERE yeni.char =
Var2	45	T...	192	1000	Dispositivo2	SELECT double FROM yeni WHERE yeni."char" = 'm'	UPDATE yeni SET double WHERE yeni."char" = 'm
Var3	321	T...	192	1000	Dispositivo2	SELECT entero FROM yeni WHERE "char"='y'	UPDATE yeni SET entero WHERE "char"='y'
Var4	r	T...	192	1000	Dispositivo2	SELECT char FROM yeni WHERE entero='20'	UPDATE yeni SET char WHERE entero='20'
Var5	i	T...	192	1000	Dispositivo2	SELECT char FROM yeni WHERE entero='3'	UPDATE yeni SET char WHERE entero='3'
Var6	r	T...	192	1000	Dispositivo2	SELECT yeni.char FROM yeni WHERE entero='20'	UPDATE yeni SET char WHERE entero='20'
Var7	78	T...	192	1000	Dispositivo2	SELECT yeni.entero FROM yeni WHERE char='w'	SELECT yeni.entero FROM yeni WHERE char='w'
Var8	1.45	T...	192	1000	Dispositivo2	SELECT double FROM yeni WHERE char='m'	UPDATE yeni SET double WHERE char='m'
Var9	sfdfd	T...	192	1000	Dispositivo2	SELECT yeni.otra FROM yeni WHERE "char"='h'	UPDATE yeni SET otra WHERE "char"='h'

1- Nombre de la variable a la que corresponde el valor obtenido de la consulta SQL realizada.
2- Valor obtenido de la consulta SQL realizada a la BD.
3- Dirección de Lectura. Consulta SQL que retorna un valor.
4- Dirección de Escritura. Sentencia SQL que modifica un valor.

De manera general si la conexión a la BD no se realizó de forma correcta, o los campos consultados no se encuentran en la BD, o el tipo de dato del campo consultado no es soportado por el estándar ODBC se retornará como valor 0 y la calidad no será de 192. Si el nombre de la variable que está creando coincide con el de alguna que fue creada anteriormente se mostrará el mensaje de error: *"Nombre no válido".* Si, como Dirección de lectura, se establece una sentencia UPDATE y como Dirección de Escritura un SELECT se mostrará la fila de color rojo, indicando el error cometido.

Anexo 4 Manual de usuario Manejador ODBC para Linux.

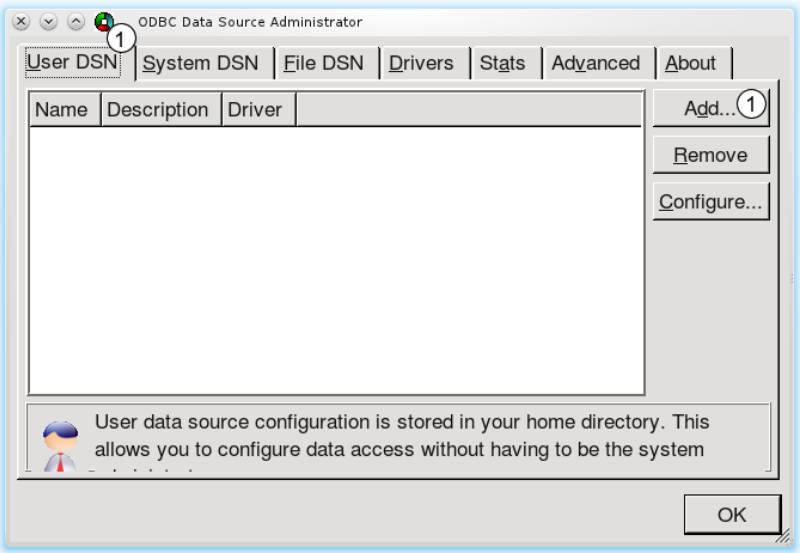
En Linux al igual que en Windows para que sea posible el acceso desde el Sistema SCADA a datos almacenados en BD de diferentes SGBD, es necesario poseer el driver

ODBC correspondiente a cada uno de estos SGBD y a la versión que se esté utilizando. Por ejemplo, si se desea acceder a datos contenidos en una BD PostgreSQL y la versión de este SGBD que está utilizando es la 8.4 es necesario obtener e instalar el driver ODBC para PostgreSQL correspondiente a la versión 8.4 o mayor que esta. Una vez instalado el driver es necesario la creación de un DSN donde se establece la ruta y característica de la conexión que se desea establecer a la BD. Por otra parte es necesario que se encuentre instalado, además de unixODBC, el paquete unixodbc-bin el cual se puede obtener del repositorio que posee cada una de las distribuciones de Linux.

Para la creación del DSN se debe acceder al ODBC Data Source Administrator, para ello es necesario ejecutar ODBCConfig.

Una vez ubicados en el ODBC Data Source Administrator se llevarán a cabo los siguientes pasos:

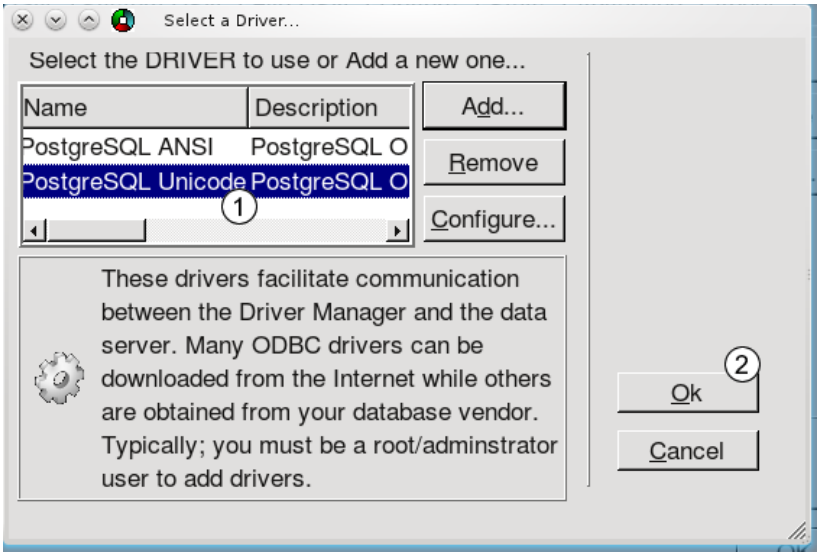
Interfaz Administrador de Orígenes de Datos de ODBC.



1-Acceder a la opción User DSN.

2-Acceder a la opción Add.

Interfaz para seleccionar Driver ODBC.



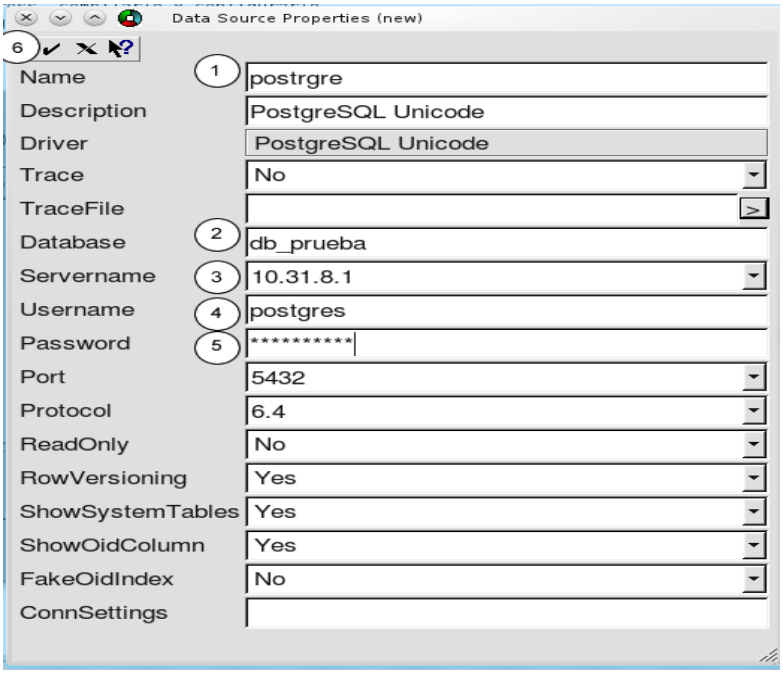
1-Seleccionar de la lista de drivers, el que se instaló y corresponde al SGBD al cual se desea acceder.

2-Acceder al botón OK.

A partir de este paso comienza la configuración del DSN teniendo en cuenta los parámetros que solicite el fabricante de cada unos de los SGBD.

A continuación se muestra como ejemplo la configuración para una BD PostgreSQL.

Interfaz Configuración del DSN.



1-Opción para establecer el nombre de la conexión a la BD PostgreSQL.

2-Opción para establecer el nombre de la BD a la cual se desea acceder.

3-Opción para establecer la dirección IP donde se encuentra montado el servidor PostgreSQL.

4- Opción para definir el nombre de usuario necesario para acceder a la BD.

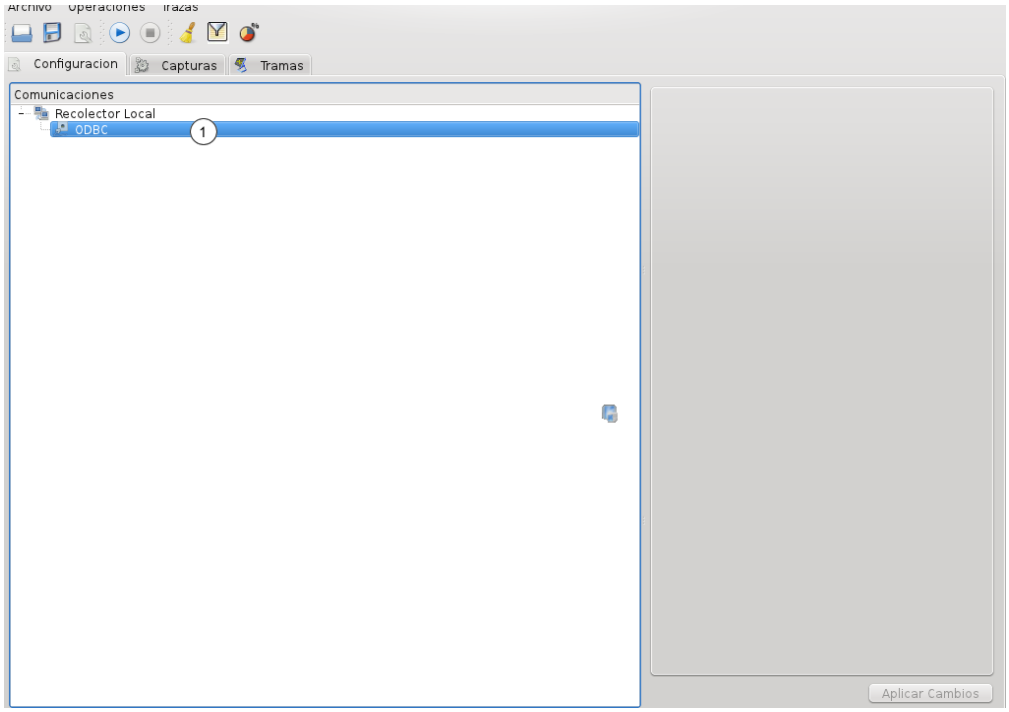
5- Opción para introducir la contraseña requerida para el acceso a la BD.

6- Opción para concluir y guardar la configuración del DSN creado.

Para comprobar que la instalación del driver ODBC para el SGBD seleccionado y que la configuración del DSN se realizó de forma satisfactoria es necesario acceder a la BD, realizar consultas y verificar que el resultado de estas es correcto. En este caso la recolección de datos se realizará utilizando el Recolector Gráfico.

A continuación se muestra como ejemplo la configuración requerida en el Recolector para acceder a datos contenidos en una BD, en este caso PostgreSQL.

Interfaz Recolector Gráfico. Crear Manejador.



1- Opción para crear Manejador ejecutando clic derecho encima de ODBC.

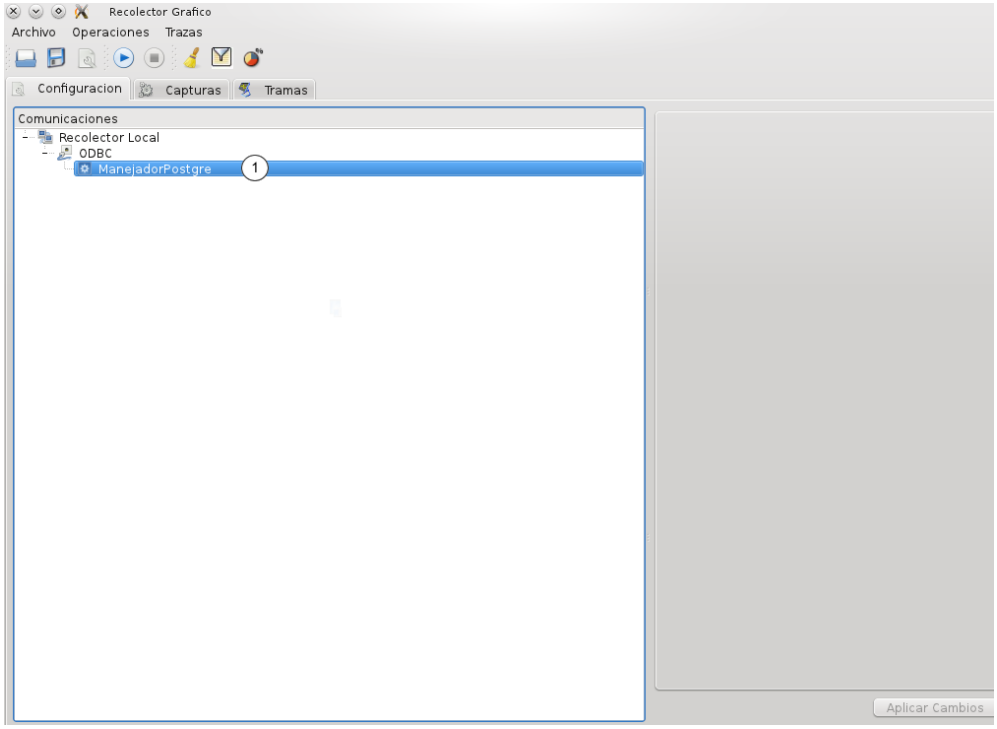
Interfaz para la creación del Manejador.

1- Opción para introducir el nombre del Manejador que se está creando.

2- Opción para introducir una breve descripción sobre el manejador. (Este campo no es obligatorio).

3- Acceder al botón Aceptar para guardar y cerrar la ventana.

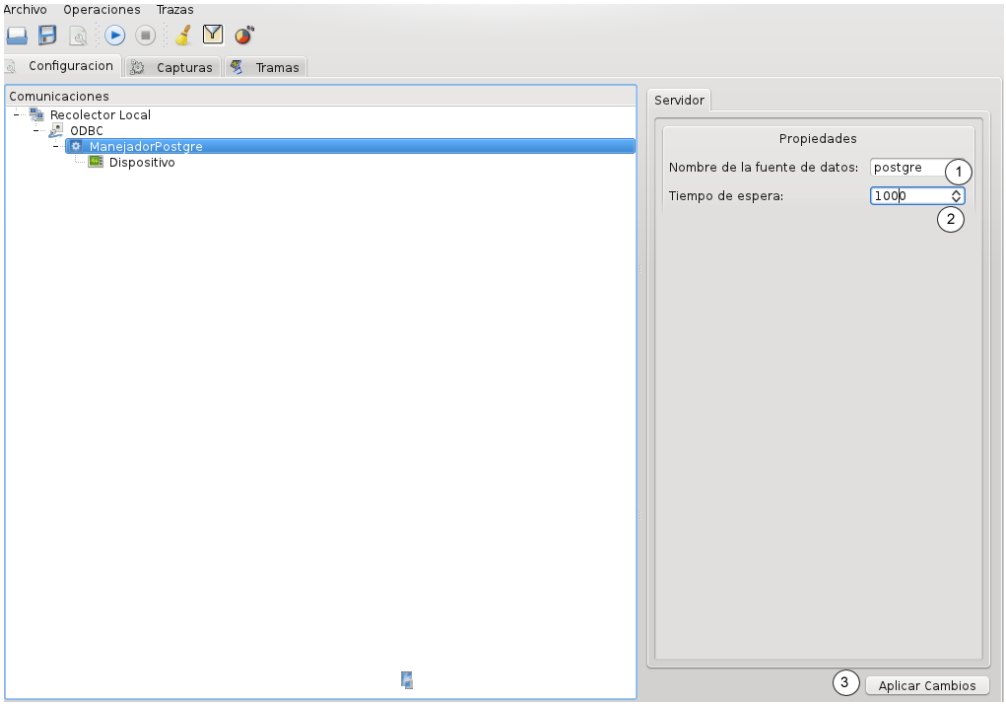
Interfaz Recolector Gráfico. Crear Dispositivo.



1- Opción para crear Dispositivo ejecutando clic derecho encima del Manejador creado.

(El Dispositivo se crea de igual forma que el Manejador).

Interfaz Recolector Gráfico.Establecer Conexión.

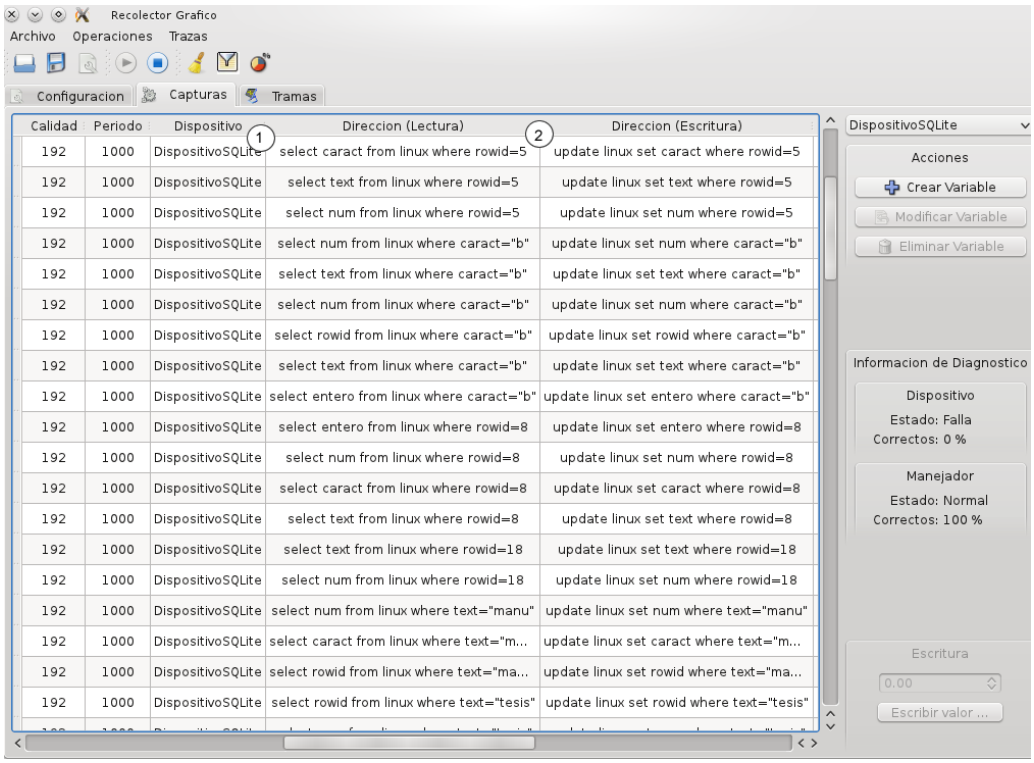


1- Opción para introducir el nombre de la fuente de datos a la que se desea conectar. Este nombre debe coincidir exactamente con el nombre introducido en el campo Name en la configuración del DSN. Ver ***Interfaz Configuración del DSN.***
2- Opción para establecer un tiempo de espera de los resultados de las consultas que se realizarán.
3- Acceder al botón Aplicar Cambios para cerrar y guardar la configuración

Para verificar que la conexión se estableció correctamente se debe acceder a la pestaña “Capturas” donde, a través de las direcciones de variables, se comprobará la validez de la misma. En el caso del Manejador ODBC las direcciones de variables consisten en sentencias SQL ya que es la forma en que se puede acceder y obtener información de cualquier BD. La sintaxis SQL a utilizar es independiente del SGBD al que se desea acceder. La Dirección (Lectura) debe consistir en un SELECT, mientras que la Dirección (Escritura) en un UPDATE.

A continuación se muestra un ejemplo de direcciones de variables utilizadas en la conexión a una BD.

Interfaz Recolector Gráfico.



1- Dirección de Lectura. Consulta SQL que retorna un valor.
2- Dirección de Escritura. Sentencia SQL que modifica un valor.

De manera general si la conexión a la BD no se realizó de forma correcta, o los campos consultados no se encuentran en la BD, o el tipo de dato del campo consultado no es soportado por el estándar ODBC se retornará como valor 0 y la calidad no será de 192. Si el nombre de la variable que está creando coincide con el de alguna que fue creada anteriormente se mostrará el mensaje de error: *"Nombre no válido".* Si, como Dirección de lectura, se establece una sentencia UPDATE y como Dirección de Escritura un SELECT se mostrará la fila de color rojo, indicando el error cometido.

Glosario de Términos

Bus de campo: Es un solo enlace de comunicaciones, al cual se conectan directamente todos los dispositivos. Existen dos formas de comunicación en esta topología, colisión y maestro/esclavo.

Controlador Lógico Programable (PLC): Dispositivos electrónicos usados en automatización industrial para realizar estrategias de control básicas. Por su robustez y características sencillas de control, están cercanos al proceso, permitiendo ejecutar las tareas básicas del control, aun cuando no tenga conexión a las capas superiores del control.

Dispositivo: Se denomina dispositivo a un elemento de hardware que alberga información de proceso o estado de sí mismo, que forma parte de un sistema automatizado. Comúnmente los dispositivos son Controladores Lógicos Programables, Computadoras Industriales, Sistemas de Control Distribuidos, sensores o actuadores inteligentes con capacidad de comunicación.

Sistemas de Control, Supervisión y Adquisición de Datos (SCADA): Aplicación de software especialmente diseñada para funcionar sobre computadores en el control de producción, proporcionando comunicación con los dispositivos de campo y controlando el proceso de forma automática desde la pantalla del operador, proveyendo toda la información asociada al proceso.

Asíncrona: Transmisión no relacionada con ningún tipo de sincronización temporal entre el emisor y el receptor.

Índice de Figuras

Índice de Tablas