

ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA



PROYECTO FIN DE CARRERA

**Desarrollo de un entorno de test de una red
comunicaciones de Dispositivos Electrónicos
Inteligentes**

Autor: Jesús Galván Ruiz
Tutor: Juan Cerezo Sánchez
Fecha: Octubre de 2016

ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA



PROYECTO FIN DE CARRERA

**Desarrollo de un entorno de test de una red
comunicaciones de Dispositivos Electrónicos
Inteligentes**

HOJA DE FIRMAS

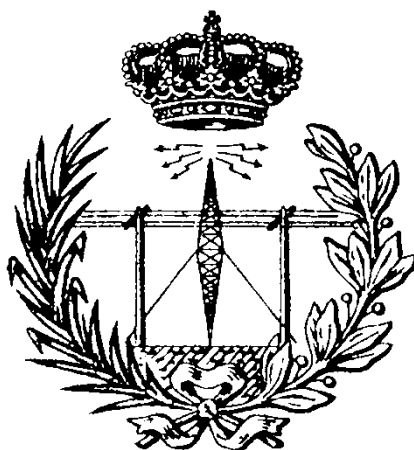
Alumno/a

Fdo.: Jesús Galván Ruiz

Tutor/a

Fdo.: Juan Cerezo Sánchez
Fecha: Octubre de 2016

ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA



PROYECTO FIN DE CARRERA

**Desarrollo de un entorno de test de una red
comunicaciones de Dispositivos Electrónicos
Inteligentes**

HOJA DE EVALUACIÓN

Calificación: _____

Presidente

Fdo.: Juan Luis Navarro Mesa

Vocal

Fdo.: Sunil Lalchand Khemchandani
Fecha: Octubre de 2016

Secretario/a

Fdo.: Aurelio Vega Martínez

Dedicatoria

*Este Proyecto está dedicado a mi mujer
Sonia y a mis hijos Cloe y Miguel por su
apoyo y aguante.*

*Mención especial a mi madre “por fin llegué”
que Dios te aguarde donde estés.*

A mi Padre y hermanos.

*Por último dar las gracias a mi tutor por su
acompañamiento y sabiduría.*

ÍNDICE

BLOQUE 1	14
CAPÍTULO 1: INTRODUCCIÓN	14
1.1.- PETICIONARIO.....	14
1.2.- ANTECEDENTES.....	14
1.2.1 POSIBLES SOLUCIONES A LOS RETARDOS	14
1.3.- OBJETIVOS	16
1.4.- PROBLEMÁTICA DETECTADA	16
1.5.- SOLUCIÓN PROPUESTA	17
1.6.- ESTRUCTURA DE LA MEMORIA.....	18
BLOQUE 2	20
CAPÍTULO 2: PROTOCOLO OPCUA	20
2.1.- INTRODUCCIÓN	20
2.2.- OPC CLÁSICO.....	21
2.2.1.- OPC DA (Data Access).....	22
2.2.2.- OPC A&E (Alarm and Events).....	23
2.2.3.- OPC HDA (Historical Data Access)	24
2.2.4.- OTRAS INTERFACES ESTÁNDARES OPC.....	25
2.2.5.- OPC XML-DA	26
2.3.- MOTIVOS PARA OPC UA.....	27
2.3.1.- VISIÓN GENERAL DE OPC-UA.....	30
2.3.2.- ESPECIFICACIONES OPC-UA	31
2.3.3.- CAPAS SOFTWARE EN OPC UA.....	35
2.4.- CONCLUSIONES.....	37
CAPÍTULO 3: PROTOCOLO IEC-60780	40
3.1.- INTRODUCCIÓN	40
3.2.- DEFINICIONES	40
3.3.- ESTRUCTURA DEL PROTOCOLO	41
3.3.1.- CAPA FÍSICA.....	42
3.3.2.- CAPA DE ENLACE	43
3.3.3.- CAPA DE APLICACIÓN	43

3.4.- CAPA FÍSICA	44
3.4.1 NIVEL FÍSICO TCP/IP.....	44
3.4.2 NIVEL DE ENLACE TCP/IP	45
3.5.- CAPA DE ENLACE IEC-60870	47
3.5.1.- FORMATOS DE LA TRAMA DE TRANSMISIÓN	47
3.5.2.- IEC 60870-5-2: PROCEDIMIENTOS DE TRANSMISIÓN DE ENLACE	47
3.5.3.- DIAGRAMAS DE TRANSICIÓN DE ESTADO	48
3.5.4.- PROCEDIMIENTOS DE TRANSMISIÓN NO BALANCEADA.....	49
CAPÍTULO 4: WIRESHARK	59
4.1.- ¿QUÉ ES WIRESHARK?.....	59
4.2.- UTILIZACIÓN DE WIRESHARK.....	59
4.3.- CARACTERÍSTICAS	59
4.4.- REQUISITOS DEL SISTEMA	60
4.5.- INSTALACIÓN	61
4.5.1.- COMPONENTES DE INSTALACIÓN	61
4.6.- INTERFAZ DE USUARIO	62
4.6.1.- LA VENTANA PRINCIPAL	62
4.6.2.- EL MENÚ	63
4.7.- FUNCIONAMIENTO	65
4.7.1.- FICHERO XML (detallado)	68
4.8.- CAMPOS SELECCIONADOS.....	68
4.8.1.- OPCUA	68
4.8.2.- IEC-60870.....	69
CAPÍTULO 5: SIMULADORES DE PROTOCOLOS.....	72
5.1.- SIMULADOR OPC UA	72
5.1.1.- WIRESHARK EN LINUX	72
5.1.2.- CARPETAS Y ARCHIVOS.....	73
5.1.3.- CLIENTE OPC UA.....	74
5.1.4.- SERVIDOR OPC UA.....	76
5.1.5.- EJEMPLO	77
5.2.- SIMULADOR IEC – 60870	78
BLOQUE 3	86

CAPÍTULO 6: SOFTWARE DE ANÁLISIS	86
6.1.- JUSTIFICACIÓN DE MYSQL	86
6.2.- INSTALACIÓN DE MYSQL	87
6.3.- CONFIGURACIÓN MySQL.....	89
6.3.1.- CAMPOS DE LA TABLA OPC UA	92
6.3.2.- CAMPOS DE LA TABLA IEC – 60870	93
6.4.- SOFTWARE.....	94
6.4.1.- MENÚ ARCHIVO	94
6.4.2.- MENÚ MOSTRAR.....	97
6.4.3.- MENÚ CALCULAR	98
6.5 EJEMPLOS DE CÁLCULO CON DIFERENTES CAPTURAS.....	100
CAPÍTULO 7: IMPLEMENTACIÓN	106
7.1.- INTRODUCCIÓN	106
7.2.- ENTORNO	106
7.3.- IMPLEMENTACIÓN.....	106
7.3.1.- PAQUETE CHARTS	107
7.3.2.- PAQUETE FILES.....	109
7.3.3.- PAQUETE GRÁFICOS	127
7.3.4.- PAQUETE OPERACIONES.....	141
BLOQUE 4	154
BIBLIOGRAFÍA	154
ANEXOS	158
A.1- Código fuente archivo PanelChart.java.....	158
A.2- Código fuente archivo Archivo.java.....	160
A.3- Código fuente archivo Paquete.java.....	163
A.4- Código fuente archivo Diálogo1.java.....	165
A.5- Código fuente archivo DiálogoCalcular.java	167
A.6- Código fuente archivo VentanaInterna.java	170
A.7- Código fuente archivo VentanaInternaCalcular.java	172
A.8- Código fuente archivo VentanaInternaMostrar.java.....	173
A.9- Código fuente archivo VentanaInternaMostrar.java.....	174
A.10- Código fuente archivo Operaciones.java	180

BLOQUE 5	190
PLANOS Y PROGRAMAS	190
PP.1- INTRODUCCIÓN	190
PP.2- DISTRIBUCIÓN DE ALGORITMOS.....	190
PLIEGO DE CONDICIONES.....	194
PRESUPUESTO	198
P.1- DESGLOSE DEL PRESUPUESTO	198
P.2- RECURSOS MATERIALES	199
P.2.1- RECURSOS SOFTWARE.....	199
P.2.2- RECURSOS HARDWARE	200
P.3- TRABAJO TARIFADO POR TIEMPO EMPLEADO	201
P.4- COSTES DE REDACCIÓN DEL PROYECTO.....	202
P.5- MATERIAL FUNGIBLE.....	203
P.6- DERECHOS DE VISADO DEL COIT	203
P.7- GASTOS DE TRAMITACIÓN Y ENVÍO.....	204
P.8- APLICACIÓN DE IMPUESTOS.....	204

CAPÍTULO 1

INTRODUCCIÓN

CAPÍTULO 1: INTRODUCCIÓN

1.1.- PETICIONARIO

Este proyecto fin de carrera ha sido realizado por el alumno Jesús Galván Ruiz, para la obtención del título de Ingeniero en electrónica. Se ha llevado a cabo a petición de Escuela Ingeniería de Telecomunicaciones y Electrónica, perteneciente a la Universidad de Las Palmas de Gran Canaria.

1.2.- ANTECEDENTES

En los entornos industriales las redes de comunicaciones son de vital importancia para el control de todos los procesos de producción. El parámetro más importante en los procesos de producción es el tiempo. Si nos imaginamos una fábrica de cerveza donde las máquinas de producción son capaces de llenar 1000 botellas en un minuto, y se necesita controlar dicha máquina en tiempo real para poder gestionar correctamente todos los procesos, nos encontramos con un problema si los retardos son excesivamente elevados. Este proyecto trata de calcular dichos retardos y así poder determinar diferentes soluciones para que puedan ser corregidos.

1.2.1 POSIBLES SOLUCIONES A LOS RETARDOS

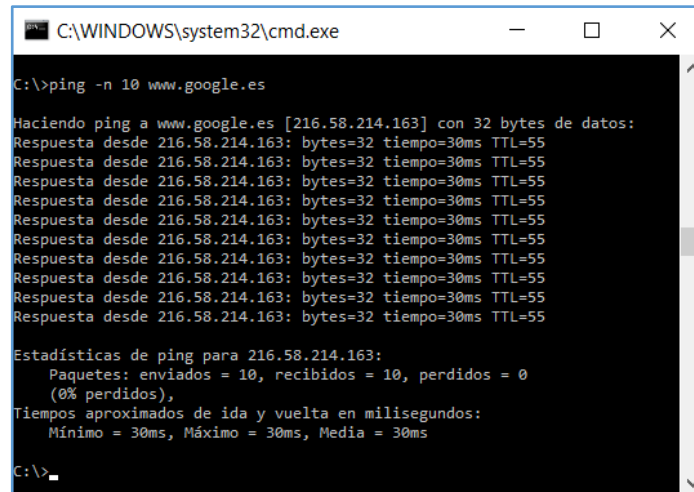
Hoy en día existen en la formas de calcular retardos. Aquí se enumeran algunas de ellas:

Equipos reflectómetros

El uso de equipos reflectómetros para averiguar los retardos de una red. Estos equipos ofrecen de forma exacta la distancia de los cables de conexión y su retardo con respecto a una frecuencia determinada. El problema que existe con estos equipos a la hora de calcular los retardos, es que son incapaces de calcular dicho parámetro sobre cualquier protocolo.

Comando Ping

Existen distintos comandos que se pueden usar para averiguar el tiempo de respuesta de envío de determinados paquetes TCP/IP. Un ejemplo puede ser el comando *ping*. Dicho comando es capaz, introduciendo distintos parámetros, darnos el tiempo de respuesta de un paquete. En este gráfico se muestra un ejemplo de cómo funciona dicho comando:



```
C:\WINDOWS\system32\cmd.exe
C:\>ping -n 10 www.google.es

Haciendo ping a www.google.es [216.58.214.163] con 32 bytes de datos:
Respuesta desde 216.58.214.163: bytes=32 tiempo=30ms TTL=55
Respuesta desde 216.58.214.163: bytes=32 tiempo=30ms TTL=55
Respuesta desde 216.58.214.163: bytes=32 tiempo=30ms TTL=55
Respuesta desde 216.58.214.163: bytes=32 tiempo=30ms TTL=55
Respuesta desde 216.58.214.163: bytes=32 tiempo=30ms TTL=55
Respuesta desde 216.58.214.163: bytes=32 tiempo=30ms TTL=55
Respuesta desde 216.58.214.163: bytes=32 tiempo=30ms TTL=55
Respuesta desde 216.58.214.163: bytes=32 tiempo=30ms TTL=55
Respuesta desde 216.58.214.163: bytes=32 tiempo=30ms TTL=55
Respuesta desde 216.58.214.163: bytes=32 tiempo=30ms TTL=55

Estadísticas de ping para 216.58.214.163:
    Paquetes: enviados = 10, recibidos = 10, perdidos = 0
    (0% perdidos),
    Tiempos aproximados de ida y vuelta en milisegundos:
        Mínimo = 30ms, Máximo = 30ms, Media = 30ms

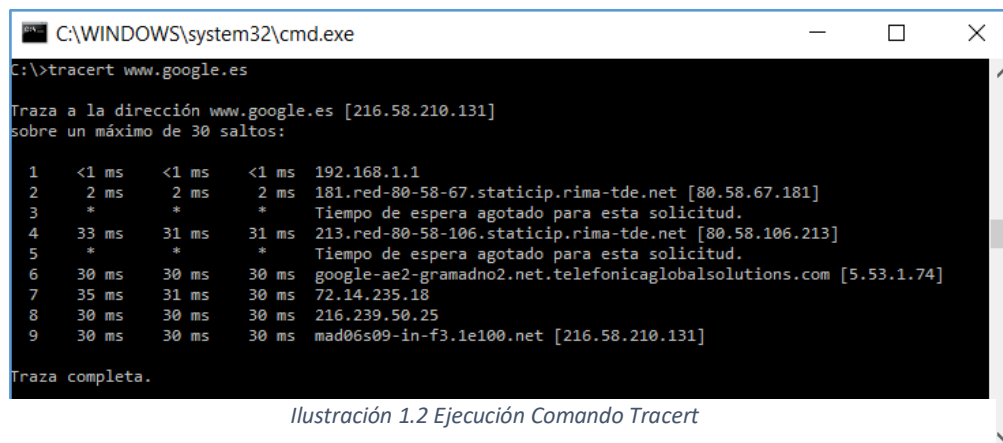
C:\>
```

Ilustración 1.1 Ejecución Comando Ping

En la *ilustración 1.1* se puede comprobar, como el comando envía 10 paquetes a una dirección, y ésta le devuelve todos los paquetes, al mismo tiempo que ofrece una estadística de los distintos elementos enviados. Este comando tiene una exactitud relativa, debido a que funciona como un eco sobre un paquete estándar de 32 bytes, cuyo valor es demasiado pequeño y no introduce ningún dato que el equipo necesite gestionar. Además no todos los equipos que se usan en un entorno industrial, ofrecen la capacidad de responder a un *ping*.

Comando tracert

Otro comando que también devuelve tiempos de respuesta es el comando *tracert*. Este comando suele ser útil en entornos de redes complejas, para averiguar la ruta que



```
C:\WINDOWS\system32\cmd.exe
C:\>tracert www.google.es

Traza a la dirección www.google.es [216.58.214.131]
sobre un máximo de 30 saltos:

 1  <1 ms  <1 ms  <1 ms  192.168.1.1
 2  2 ms   2 ms   2 ms   181.red-80-58-67.staticip.rima-tde.net [80.58.67.181]
 3  *      *      *      Tiempo de espera agotado para esta solicitud.
 4  33 ms  31 ms  31 ms  213.red-80-58-106.staticip.rima-tde.net [80.58.106.213]
 5  *      *      *      Tiempo de espera agotado para esta solicitud.
 6  30 ms  30 ms  30 ms  google-ae2-gramadno2.net.telefonicaglobalsolutions.com [5.53.1.74]
 7  35 ms  31 ms  30 ms  72.14.235.18
 8  30 ms  30 ms  30 ms  216.239.50.25
 9  30 ms  30 ms  30 ms  mad06s09-in-f3.1e100.net [216.58.214.131]

Traza completa.
```

Ilustración 1.2 Ejecución Comando Tracert

sigue un determinado paquete y los tiempos de respuesta. En la *ilustración 1.2* se puede observar su funcionamiento:

Este comando no es útil a la hora de averiguar los retardos, debido a que su función principal es la de establecer los distintos tiempos de respuesta, sobre cada uno de los nodos de la ruta que sigue hasta llegar al final. El objetivo de este trabajo es calcular el tiempo de respuesta entre el equipo local y el remoto, obviando los tiempos de respuesta intermedios, por lo que el uso de este comando no ayuda al propósito de obtener un tiempo de respuesta final.

Equipos parametrizados

Otra posible solución para la obtención de tiempos en una red, es utilizar hardware que ya esté parametrizado. Existen empresas que fabrican equipos y material de redes que ya está evaluado y preparado para entornos industriales. Este método de trabajo no es exacto, debido a que se sabe realmente el retardo físico del hardware, pero no los tiempos con tráfico de paquetes TCP/IP.

1.3.- OBJETIVOS

- Obtener el título de ingeniero en electrónica.
- Estudiar las distintas dificultades que existen en la ingeniería de control sobre los tiempos en los procesos industriales.
- Calcular los distintos retardos que suceden dentro de una red de comunicaciones en un entorno industrial a través de los protocolos IEC-60870 y OPC UA.
- Crear un entorno de pruebas simulando una red de comunicaciones industriales para su posterior análisis.
- Trabajar con distintas herramientas de simulación y captura de datos de red.
- Crear una base de datos con los distintos tiempos entre equipos.
- Visualizar de forma gráfica distintos cálculos sobre los tiempos de equipos.

1.4.- PROBLEMÁTICA DETECTADA

- Búsqueda de información específica sobre retardo en redes de comunicaciones TCP/IP.

- Simular una red con servidores y clientes, de los distintos protocolos.
- Encontrar información específica sobre los 2 protocolos.
- Trabajar con herramientas que no supongan un coste excesivo.

1.5.- SOLUCIÓN PROPUESTA

Para cumplir con los objetivos marcados en primer lugar, se buscará toda la información libre sobre cada uno de los protocolos y cada una de las herramientas que se han usado para la adquisición de datos. En la *ilustración 1.3* se puede comprobar como se va a realizar el procedimiento para la adquisición de los paquetes de datos, generando lo que se denomina **Entorno de Pruebas**. En primer lugar tenemos los simuladores conectados a la red de datos donde van a generar los distintos paquetes de cada uno de los protocolos. Ambos simuladores van a funcionar de forma independiente dentro de la red de datos. El wireshark va a capturar todos los paquetes de ambos protocolos, y posteriormente generará un archivo **XML** como se explicará en el *capítulo 4*. Seguidamente se desarrollará el software de análisis que tratará el archivo volcado por el wireshark, y gestionará la información. Este programa, trabajará con la base de datos MySQL para guardar y extraer los distintos parámetros que se explicarán en los *capítulos 6 y 7*. Se visualizarán los distintos resultados que genera el software de análisis en el *capítulo 6* y se explicará con detalle el código en el *capítulo 7*.

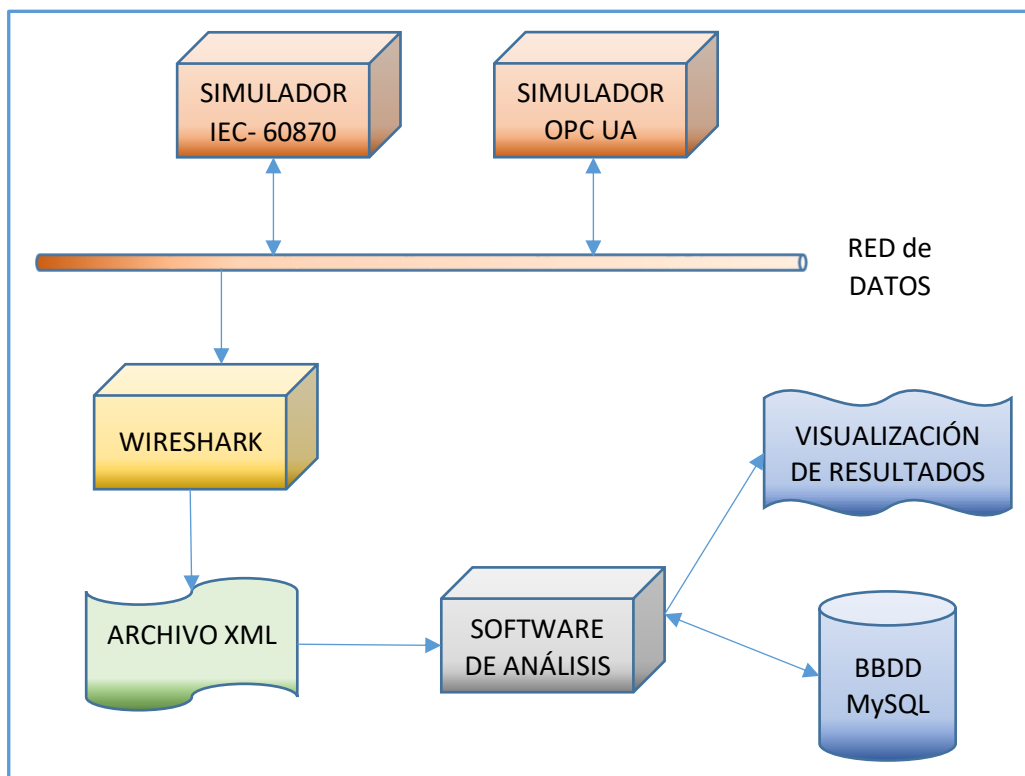


Ilustración 1.3 Entorno de Pruebas

1.6.- ESTRUCTURA DE LA MEMORIA

La memoria está distribuida en cinco grandes bloques. En el primer bloque (*Capítulo 1*), se encuentra un resumen, donde se detallan los objetivos, antecedentes, problemática detectada y soluciones. El segundo bloque (*Capítulos 2 al 5*), comprende los distintos estudios previos de los protocolos, donde están incluidos los simuladores y herramientas de captura. Cada uno de estos elementos estará dentro de su propio capítulo. En el tercer bloque (*Capítulos 6 y 7*), se introduce la solución y el desarrollo del proyecto, con parte del código de programación y los diagramas de flujo, que ayudan a comprender la estructura del software desarrollado.

En el cuarto bloque se desarrollan los distintos anexos correspondientes al proyecto como son la bibliografía y anexos. Por último están el pliego de condiciones, planos y programas y el presupuesto que no se han incluido en el desarrollo de la memoria.

CAPÍTULO 2

PROTOCOLO OPC UA

CAPÍTULO 2: PROTOCOLO OPCUA

2.1.- INTRODUCCIÓN

A principios de los 90 la automatización basados en PCs aumentó de forma rápida para la automatización industrial. Cabe destacar especialmente los sistemas basados en Windows para la visualización y control. El mayor esfuerzo para que se pudiera desarrollar un software estandarizado, ha sido el acceso a los datos por distintos dispositivos, protocolos, interfaces y redes.

Un ejemplo parecido lo teníamos en las aplicaciones software y los accesos a las impresoras en el antiguo sistema operativo MS-DOS, donde cada una de las aplicaciones, tenía que tener su propio driver para la impresora que quería soportar.

Windows fue capaz de resolver el problema del driver, he introdujo los soportes de impresora dentro del SO (Sistema Operativo). Así las aplicaciones que querían trabajar bajo Windows, solo tenían que tener la interfaz del driver de la impresora del propio SO. De esta forma los fabricantes de impresoras programaban y ofrecían los controladores de las impresoras para el mismo SO.

Tanto los fabricantes de Interfaces Hombre-Máquina (*Human-Machine Intefaces – HMI*) y software de Supervisión y Control (*Supervision Control and Data Acquisition – SCADA*) tenían problemas parecidos, en 1995 se formó un grupo de trabajo formado por las empresas *Fischer-Rosemount, Rockwell Software, Opto 22, Intellution e Intuitive Technology*. El entorno de trabajo definió un estándar *Plug & Play* para drivers de dispositivos, creando un sistema de acceso estandarizado a datos de automatización basados en el **SO** de Windows.

Toda la labor realizada por el grupo de trabajo, dio como resultado la especificación OPC Data Access (**OPC-DA**), donde vio la luz en agosto del año 1996. Todas las características sobre OPC, está sostenida por una organización sin ánimo de lucro denominada OPC Foundation y prácticamente en ella se incluyen la mayoría de los proveedores de sistemas de automatización industrial. Esta fundación define y adopta estándares de forma más rápida que otras organizaciones, usando APIs basadas en tecnologías de Microsoft Windows como (*Component Object Model – COM*) y (*Distributed COM – DCOM*).

Los sistemas **SCADA** y **HMI**, la gestión de procesos y Sistemas de Control Distribuidos (*Distributed Control Systems – DCS*) y Sistemas de Ejecución Manufacturera (*Manufacturing Execution System – MES*) deben soportar interfaces OPC. El mismo está aceptado de forma prácticamente universal, donde ofrece la posibilidad de combinar datos entre sistemas de automatización industrial.

Para asegurar que los productos OPC sean operativos, existen dos niveles de certificación o conformidad OPC. En el primer nivel de OPC se ofrecen distintos tests de conformidad, donde los resultados obtenidos son enviados de forma encriptada a la OPC Foundation. Una vez se validan los componentes se les permite el uso del logo “*Self-Tested*”.

Para el segundo nivel se realizan test funcionales de calidad en laboratorios con una certificación independiente. Todos los productos que pasen estos tests funcionales de carga y de stress, pueden usar el logo “*OPC Certified*”, el cual garantiza un alto nivel de calidad y conformidad con los estándares OPC. Se da por hecho que a los usuarios finales siempre se les recomienda, para garantizar los problemas de operatividad y asegurar la viabilidad y el rendimiento de los productos, soluciones basadas en “*OPC Certified*”.

2.2.- OPC CLÁSICO

Según las necesidades de las distintas aplicaciones industriales, se desarrollan tres especificaciones principales OPC:

- Acceso a Datos (**OPC-DA**): Acceso a datos de proceso actuales.
- Alarmas y Eventos (**OPC-A&E**): Información basada en eventos, incluyendo reconocimiento de alarmas.
- Acceso a Datos Históricos (**OPC-HDA**): Acceso a datos archivados.

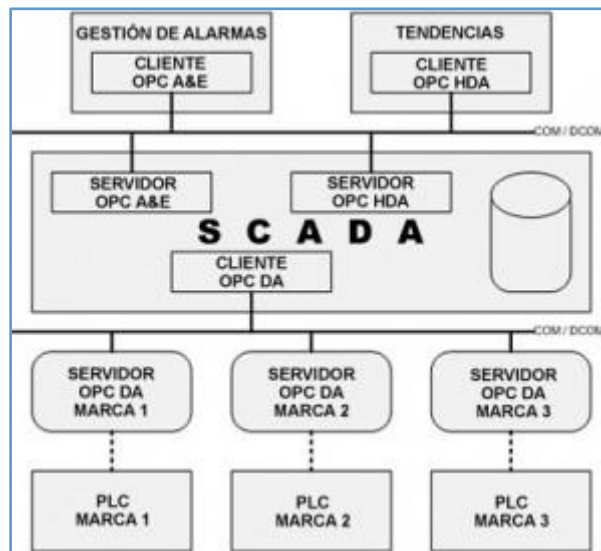


Ilustración 2.1 Caso de uso en Clientes y Servidores OPC

Para el intercambio de información OPC utiliza un sistema cliente-servidor (*Ilustración 2.1*), siendo un servidor una fuente de datos encapsulada, donde la información está disponible a través de su interfaz. El cliente a su vez accede y consume la información ofrecida y tanto las aplicaciones que consumen y ofrecen datos, pueden ser al mismo tiempo cliente y servidor. Las interfaces clásicas de OPC están basadas en la tecnología **COM** (*Component Object Model*) y **DCOM** (*Distributed Component Object Model*) de Microsoft.

La principal ventaja de este sistema, era que reducía el trabajo en las especificaciones de definición de las diferentes **APIs** para distintas necesidades especializadas, sin tener que definir un protocolo de red o un mecanismo para comunicarse entre procesos. De esta forma esta tecnología ofrece un mecanismo transparente para llamar a métodos de un objeto, en procesos que se puedan estar ejecutando tanto en el mismo nodo como en otro nodo de la red.

Las principales desventajas son:

- La dependencia al **SO** Windows.
- La utilización de **DCOM** tiene muchos problemas en comunicaciones remotas. Es un sistema difícil de configurar en tiempos de espera largos y no se puede usar en comunicaciones a través de Internet.

2.2.1.- OPC DA (Data Access)

Este interfaz permite leer, escribir y monitorizar distintas variables que pueden contener datos de procesos actuales. El objetivo principal es transmitir datos en tiempo real de **PLCs**, otros dispositivos similares a **HMIs** y otros clientes. Es el interfaz más importante y el que está más implementado, siendo prácticamente un 99% de los productos los que usan este tipo de tecnología. El resto de interfaces OPC se implementan principalmente como complemento a **OPC-DA**.

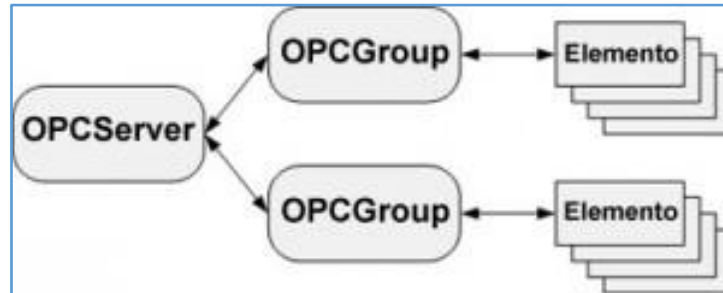


Ilustración 2.2 Objetos creados por un cliente OPC para acceder a datos

Los clientes son los encargados de seleccionar la variable que quieren leer y/o monitorizar en el servidor. Normalmente el cliente establece una conexión al servidor generando un objeto *OPCServer*. El objeto es capaz de ofrecer métodos para navegar por un rango de direcciones que buscan elementos OPC y sus propiedades, cómo son el tipo de dato o los permisos de acceso. Normalmente en el acceso a los datos, el cliente agrupa distintos OPCs que tengan propiedades semejantes, entre las que se encuentra por ejemplo, el tiempo de actualización.

Una vez añadidos a un grupo, el cliente escribe o lee los elementos que considere. Normalmente para una lectura cíclica de los datos por parte del cliente, se utiliza la monitorización de cambio de valores en el servidor, por ejemplo: El cliente define un tiempo de actualización. Este tiempo es utilizado por el servidor para comprobar que dichos valores han cambiado de forma cíclica. En cada ciclo el servidor envía solo los valores cambiados al cliente.

Esta tecnología es capaz de ofrecer datos en tiempo real y no tiene por qué ser de forma permanente. Por ejemplo, si se interrumpe de forma temporal la comunicación con un dispositivo. Este problema se gestiona ofreciendo el instante de muestreo (*timestamp*) y la calidad a los datos enviados. Esta calidad comprueba el dato correcto (bueno) no disponible (malo), o desconocido (dudoso).

2.2.2.- OPC A&E (Alarm and Events)

Este interfaz (*Ilustración 2.3*), es capaz de recibir notificaciones de eventos y alarmas. Los eventos son notificaciones singulares que informa al cliente sobre algo que ha ocurrido. Las alarmas notifican e informan al cliente, sobre el cambio de alguna condición en el proceso. Por ejemplo, si se tiene una condición en la cual el nivel de un depósito supera el máximo permitido, en ese caso se tiene una alarma. En cambio si se observa cómo cambian los niveles de un depósito, se está hablando de un evento. Muchas de estas alarmas necesitan un reconocimiento, que se pueden realizar a través de la interfaz **OPC A&E**.

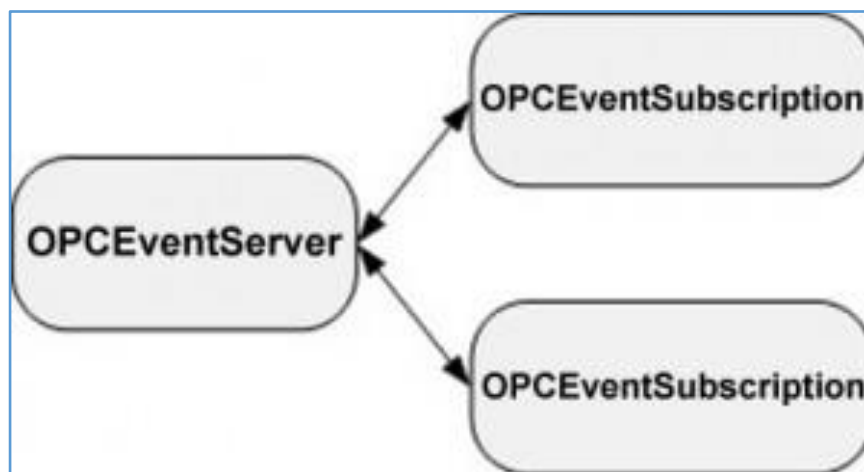


Ilustración 2.3 Objetos creados por un cliente OPC para recibir eventos

Para que un cliente pueda recibir estas notificaciones debe suscribirse a éstas, de forma que todas las que sean lanzadas por el servidor, serán recibidas por el cliente. Se podrían limitar el número de notificaciones usando algún tipo de criterio como filtro.

El primer paso de un cliente OPC es conectarse creando un objeto *OPCEventServer* en el servidor **OPC-A&E**. El siguiente paso es generar un *OPCEventSubscription*, utilizado para recibir los mensajes de eventos. Los filtros para estos mensajes se pueden configurar según cada suscripción. En la *ilustración 2.3* se muestran los distintos objetos que puede crear un cliente OPC en el servidor.

La diferencia con el **OPC-DA**, es que no se pueden hacer peticiones concretas de información explícita, como por ejemplo la lectura de valores. En cambio gracias a este interfaz, se tiene acceso a todos los eventos del proceso, de forma que el cliente siempre puede limitar dichos eventos usando algún tipo de criterio, por ejemplo la prioridad.

2.2.3.- OPC HDA (Historical Data Access)

Este interfaz permite el acceso a datos que ya hayan sido almacenados. Estos archivos se recuperan de manera uniforme desde un simple sistema de registro de datos, a un complejo sistema **SCADA**.

El cliente se conecta creando un objeto *OPCHDAServer* en el servidor **HDA**. Este objeto es capaz de ofrecer interfaces y métodos para leer y actualizar datos históricos. Otro objeto llamado *OPCHDABrowser*, se define para navegar en el rango de direcciones de un servidor **HDA**.

Esta funcionalidad permite leer datos históricos desde tres formas diferentes:

- En la primera forma de lectura, el cliente puede definir las variables y la ventana temporal que quiere leer, y el sistema le devuelve datos “crudos” del archivo.
- Otra forma de leer los valores es a través de una variable específica, para determinados tiempos de muestreo.
- La tercera forma permite operar con variables de datos, para determinados elementos en la ventana temporal.

Estos valores siempre incluyen la calidad y el instante de muestreo. Además este interfaz define métodos para poder insertar, reemplazar y borrar elementos de la base de datos históricos.

2.2.4.- OTRAS INTERFACES ESTÁNDARES OPC

OPC especifica muchos estándares como característica básica o para necesidades específicas (*Ilustración 2.4*). Por ejemplo *OPC Overview* y *OPC Common*, son comunes a todas las especificaciones OPC basadas en **COM**.

Para controlar el acceso de los clientes y proteger toda la información sensible, al mismo tiempo que las modificaciones de distintos parámetros sin autorización, se especifica la interfaz *OPC Security*.

OPC Complex Data define cómo describir y transportar valores con tipos de datos estructurados y complejos. Para el intercambio de datos entre servidores **OPC-DA**, se especifica el **OPC-DX**, definiendo el funcionamiento del cliente y la interfaces de configuración para el cliente interno del servidor. Para necesidades específicas de procesos por lotes, se especifica *OPC Batch*.

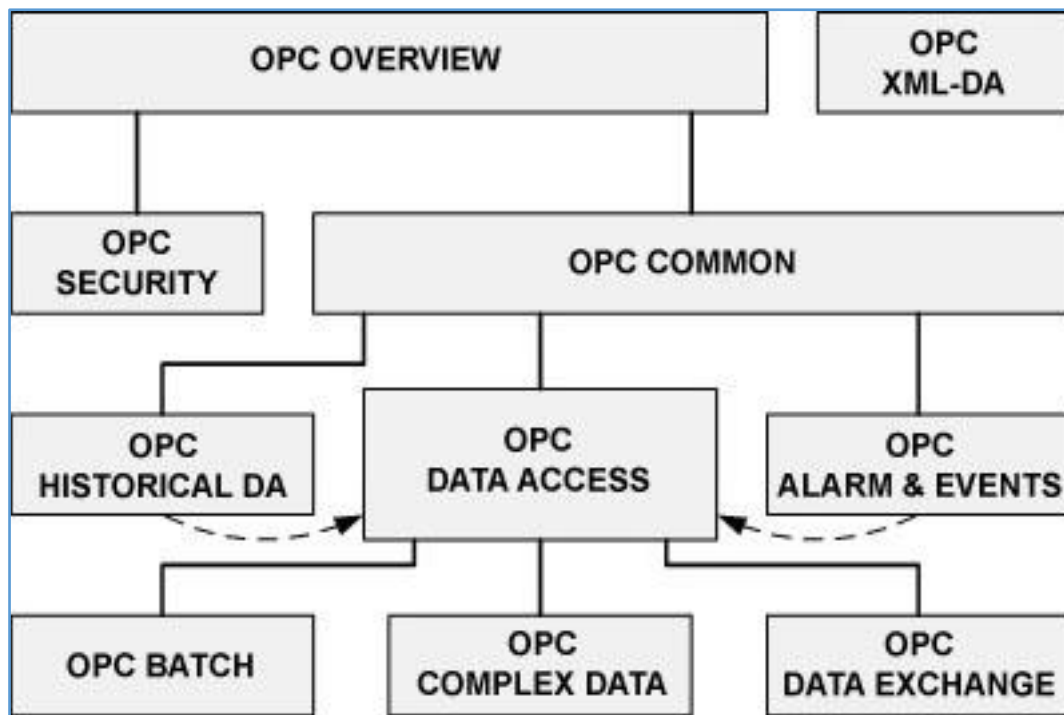


Ilustración 2.4 Interfaces estándar de OPC Clásico

2.2.5.- OPC XML-DA

Fue la primera especificación OPC independiente de plataforma que reemplazaba **COM/DCOM** por **HTTP/SOAP** y tecnologías de Servicio Web. Esta infraestructura de comunicaciones es neutra en cuanto a la plataforma del fabricante. Fue aceptada forma amplia y al mismo tiempo mantenía la funcionalidad de **OPC-DA**.

El problema de esta plataforma es que los servicios web no tienen estado, por ello introdujeron un conjunto mínimo de métodos para que se pudiera intercambiar información **OPC-DA**. Para esta plataforma se añadieron ocho métodos, que cumplen con las características principales de **OPC-DA**:

- **GetStatus:** verifica el estado del servidor.
- **Read:** Lee uno o más valores.
- **Write:** Escribe uno a más valores.
- **Browse y GetProperties:** Obtiene información de los elementos disponibles.
- **Subscribe:** Crea una suscripción para una lista de elementos.
- **SubscriptionPolledRefresh:** Intercambia valores modificados de una suscripción.

- **SubscriptionCancel:** Elimina una suscripción.

Este módulo se diseñó para poder acceder a Internet y para poder ser integrado en la empresa. Aunque este módulo intentó independizarse de la plataforma de Windows, sólo pudo ser implementado en sistemas embebidos. Aun así, debido a que el consumo de estos recursos y de que su rendimiento era limitado, no tuvo el éxito que se esperaba.

2.3.- MOTIVOS PARA OPC UA

El OPC UA, es un protocolo Unificado que independientemente de la plataforma que se utilice en los sistemas de control, la conexión es única para todos los dispositivos. La norma sólo tiene en cuenta los objetos y los métodos de los sistemas de conexión, dejando la implementación de los mismos en manos de los desarrolladores. A partir de aquí se diseñaron distintos módulos como los que se especifican a partir de aquí.

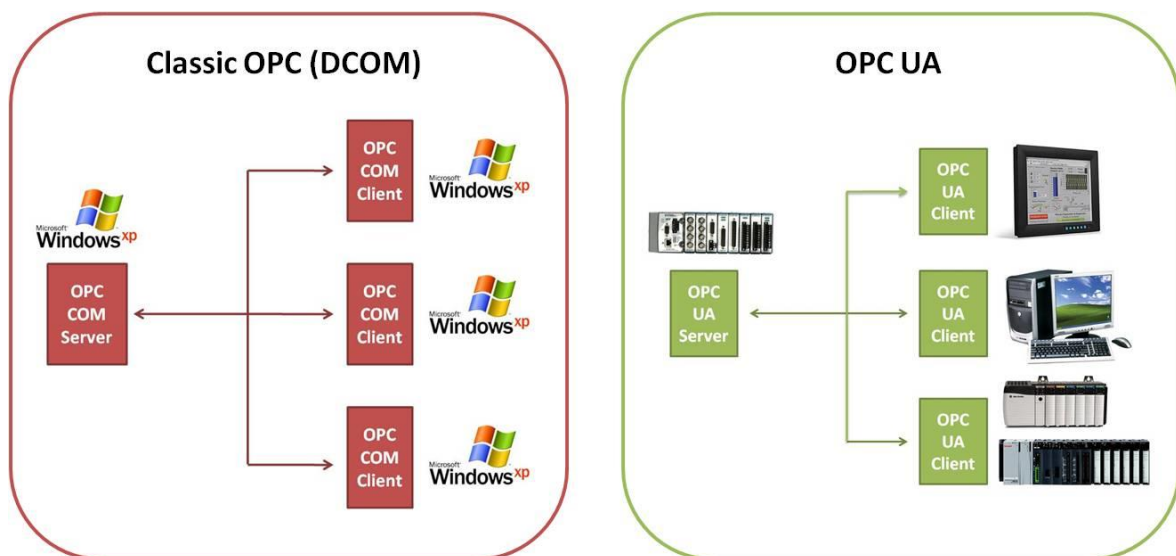


Ilustración 2.6 Comparativa entre OPC y OPC-UA

El módulo **OPC-DA** fue diseñado como un interfaz de drivers de comunicaciones, permitiendo un acceso estandarizado, tanto de lectura como de escritura con datos actuales en dispositivos de automatización. Los casos más destacables eran los accesos a sistemas **HMI** y **SCADA**, accediendo a diferentes datos de distintos dispositivos y fabricantes, utilizando la interfaz ofrecida por el propio proveedor.

Hoy por hoy OPC se utiliza como un interfaz estándar entre sistemas de automatización en distintos niveles de la propia pirámide (ilustración 2.7). Se usa incluso en ámbitos para los que no fue desarrollado. Muchos productores también le gustaría usar

un estándar como OPC en otros entornos, pero la dependencia de COM de OPC o las limitaciones de acceso remoto, les hacen rechazar esta idea.

El primer intento de la OPC Foundation para poder mantener las características tan exitosas, fue la **OPC XML-DA**, comentadas anteriormente. Este fue el primer proceso para utilizar una infraestructura neutra en cuanto a la plataforma y el fabricante. Las razones principales para desechar el **OPC XML-DA** fueron, el bajo rendimiento de los Servicios Web **XML** comparado con las versiones originales desarrolladas en **COM** y los problemas de operatividad al usar distintas capas de Servicio Web **XML**. Estas circunstancias hicieron que esos requisitos no se cumplieran en la nueva generación de OPC.

Además de la propia independencia de la plataforma, las compañías que pertenecen a la OPC Foundation, adelantaron requisitos para poder usar datos y sistemas complejos, eliminando todas las limitaciones del OPC clásico.

La OPC Unified Architecture (OPC-UA), se desarrolla para crear un recambio a todas las especificaciones basadas en **COM**, sin perder las principales características y el rendimiento aportado por el OPC. Además intenta cubrir todos los requisitos del sistema, independientemente de la plataforma usada y es capaz de describir sistemas complejos. Esto requiere una escalabilidad que va desde los sistemas embebidos, pasando por los sistemas **SCADA**, hasta los sistemas **MES** y **ERP**. Los requisitos más importantes están incorporados dentro de la *tabla 2.1*:

Comunicación entre sistemas distribuidos	Modelo de datos
Fiabilidad	Modelo común para todos los datos OPC
Robustez y tolerancia a fallos	Orientado a objetos
Redundancia	Sistema de tipos ampliable
Independencia de plataforma	Meta-información
Escalabilidad	Datos y métodos complejos
Alto rendimiento	Escalabilidad de modelos desde simples a complejos
Internet y cortafuegos	Modelo básico abstracto
Seguridad y control de acceso	Base para otro modelo de datos estándar
Interoperabilidad	

Tabla 2.1 Requisitos importantes para OPC-UA

El OPC clásico se diseñó como una interfaz de drivers a dispositivos. Es muy importante que la comunicación entre distintos sistemas distribuidos sea fiable. Dado que la comunicación en red por definición no es fiable, es necesario que existan requisitos de robustez, tolerancia a fallos, redundancia y alta disponibilidad. También es muy importante la independencia de la propia plataforma y la escalabilidad para poder integrar distintos interfaces OPC, funcionando en múltiples plataformas diferentes.

Es fundamental la posibilidad de la comunicación en Internet a través de cortafuegos, haciendo que la seguridad y el control de accesos sean también requisitos importantes. Hay que añadir por último, la interoperabilidad de distintos fabricantes.

En el OPC clásico el propio modelado de datos era limitado y se necesitaban distintas mejoras que ofreciera un modelo común orientado a objetos para todos los datos. El OPC UA, debe incluir un sistema de tipos que sea ampliable y capaz de ofrecer meta-información describiendo sistemas complejos (*Ilustración 2.6*).

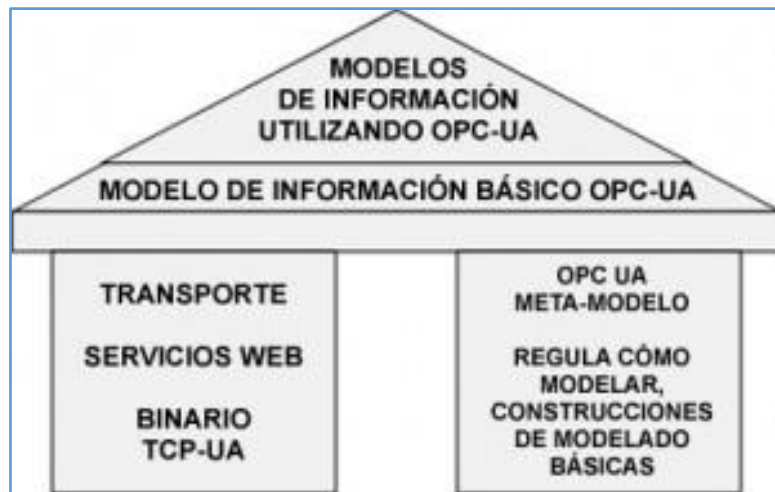


Ilustración 2.7 Fundamentos de OPC-UA

Estos datos complejos son fundamentales para soportar la descripción y transporte de estructuras de datos complejas. Evidentemente es un requisito fundamental para la mejora de las capacidades de modelado, e igual de necesario para soportar modelos simples con conceptos simples. Es por esta razón, por la que es importante disponer de un modelo básico, simple y abstracto que sea capaz de ampliarse para así escalar de modelos simples a complejos.

El grupo inicial de más de 40 representantes que estaban definiendo los requisitos para el sistema de Arquitectura Unificada, no estaba solamente compuesto por miembros de la OPC Foundation, también existen otras organizaciones como la IEC e ISA, interesadas en usar el OPC como sistema de transporte de la información en el diseño inicial. En este grupo inicial de representantes, se detalla **cómo** describir y transportar los datos y las organizaciones colaboradoras definen el tipo de datos **qué** quieren describir y transportar dependiendo de su modelo de información.

Por último era necesario en el diseño que se permitiera una migración sencilla del OPC clásico a la Arquitectura Unificada.

2.3.1.- VISIÓN GENERAL DE OPC-UA

Para poderse cumplir los objetivos definidos en el OPC-UA, se construyen distintas capas como se muestra en la *ilustración 2.7*, siendo componentes fundamentales los mecanismos de transporte y modelo de datos.

El transporte define distintos mecanismos para diferentes casos de uso. En la primera versión se define un protocolo **TCP** binario de alto rendimiento de comunicaciones intranet, así como los estándares aceptados en Internet de Servicios Web, **XML** y **HTTP**, usando cortafuegos. Ambos transportes usan un sistema de seguridad basados en mensajes, siendo el modelo de comunicaciones abstracto no dependiente de un protocolo específico y que además permite añadir más protocolos en un futuro.

En el modelo de datos se definen las reglas y bloques constructivos necesarios para extrapolar un modelo de información con la Arquitectura Unificada, definiendo los puntos de entrada al espacio direcciones y los tipos básicos para construir una jerarquía de tipos. Entre ellos los servicios web con la interfaz entre servidor, proveedores de modelo de información y clientes que consumen dicho modelo. Los servicios se definen de forma abstracta usando mecanismos de transporte para intercambiar datos entre cliente y servidor.

El concepto básico UA permite a un cliente acceder a una pieza pequeña de datos sin entender el modelo completo. Los clientes que trabajen con modelos específicos, pueden usar características mejoradas ya definidas para dominios concretos.

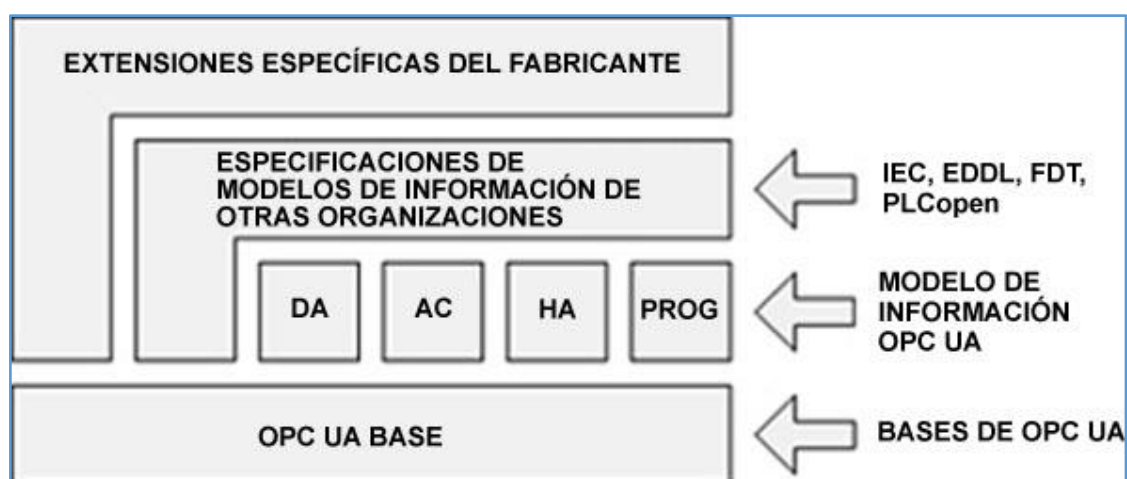


Ilustración 2.8 Capas de la Arquitectura OPC-UA

En la *ilustración 2.8*, se muestran las distintas capas del modelo de formación definidos tanto por OPC como por otras organizaciones o fabricantes. Al intentar cubrir

todas las especificaciones conocidas del OPC clásico, el modelo de información de procesos se define sobre las especificaciones base de OPC-UA.

- **Data Access (DA)**, → tiene tanto las extensiones específicas de la automatización, como el modelado de datos discretos o analógicos. A su vez es capaz de exponer la calidad de servicio, estando el resto de características DA ya cubiertas por la base.
- **Alarm & Conditions (A&C)** → especifica un sistema avanzado para la alarma y monitorización de condiciones de procesos.
- **Historical Access (HA)** → define la forma de acceder a datos y eventos históricos.
- **Programs (Prog)** → especifica un mecanismo que es capaz de iniciar, manipular y monitorizar la ejecución de programas.

Existen ya distintos tipos de estándares trabajando sobre OPC-UA por ejemplo, Field Device Integration (**FDI**), Electronic Device Description Language (**EDDL**), Field Device Tool (**FDT**) y PLCopen. Además se pueden definir otros modelos de información específica usando de forma directa la base UA, los modelos OPC, u otros modelos de información basados en OPC-UA.

2.3.2.- ESPECIFICACIONES OPC-UA

Todas las especificaciones de la arquitectura OPC-UA se dividen en distintas partes, como requisito de la estandarización IEC. Éste será conocido como el **IEC 62541**. En la *ilustración 2.9*, se muestra una vista de cómo están divididas al menos en 2 partes. La primera corresponde al núcleo (*Core*), definiendo en el mismo la base para OPC UA. La



Ilustración 2.9 Especificaciones OPC-UA

segunda las especificaciones de tipo de acceso, donde se tratan principalmente los modelos de información.

Parte 1. Descripción y conceptos. En esta especificación se proporciona una introducción importante a la tecnología UA:

- Introduce cada una de las especificaciones de OPC UA.
- Vista general de los objetivos de diseño, escalabilidad y seguridad.
- Metodología usada por OPC. Conceptos de los sistemas OPC tales como:
 - Espacio de direcciones.
 - Suscripciones.
 - Eventos.

Parte 2. Modelo de seguridad. En esta especificación se describe los elementos de seguridad.

- Introducción de los objetivos de seguridad y la identificación de las amenazas de los sistemas basados en OPC.
- Como mitigar las amenazas identificadas.
- Visión general de sobre la autenticación de usuarios y los derechos de control de acceso. También están incluidos la identificación de la aplicación utilizando certificados digitales, las actividades del usuario y del sistema de auditoría, la disponibilidad de los sistemas OPC (redundancia) y la transmisión de mensajes seguros / cifrado.

Parte 3. Modelo de espacio de direcciones. En esta especificación se proporciona una descripción detallada de un espacio de direcciones dentro de un servidor OPC UA, para el consumo de clientes OPC UA, que abarca:

- Los conceptos de un espacio de Direcciones, Nodos y Vistas.
- Descripción detallada de los Nodos y Referencias y la forma en que se utiliza la organización lógica del espacio de direcciones.
- Las descripciones detalladas de los tipos de nodos, los tipos de referencia, tipos de datos, tipos de evento y la forma en la que se pueden utilizar para el modelado de información.

Parte 4. Servicios. Esta especificación es el más importante de todas las especificaciones OPC UA, cubriendo:

- Los Servicios UA (interfaces) que los servidores y clientes deben utilizar.
- El comportamiento previsto del servidor y cliente.
- Tipos de datos comunes utilizados por los parámetros de servicio.

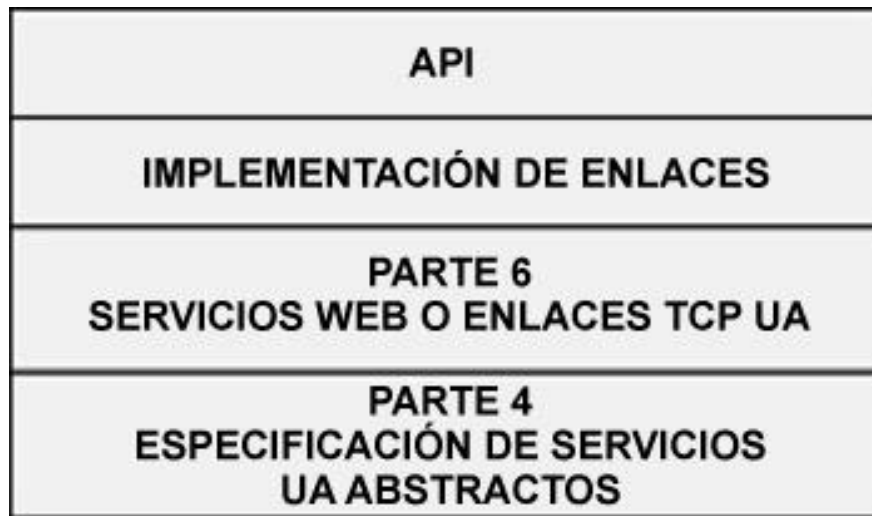


Ilustración 2.10 Capas de la arquitectura de comunicación OPC-UA

Parte 5. Modelo de información. Esta especificación proporciona una descripción detallada de cómo se utilizan los espacios de direcciones OPC UA, nodos y referencias para definir un modelo de información, abarcando dicha especificación:

- Detalles de las reglas y descripciones para las Referencias y los Nodos.
- Detalles de las reglas y descripciones para los tipos estándar de objetos, tipos de variables, métodos, tipos de eventos y tipos de datos estándar.
- Información sobre las reglas de modelado, incluyendo el modelado de subtipos.
- Máquinas-Estado y descripciones de transferencia de archivos.

Parte 6. Mapa de servicios. Esta especificación describe cómo los datos y la información se transfieren entre servidores y clientes OPC UA, incluyendo:

- Codificación/descodificación en la vista de datos y reglas para tipos de datos estándar, datos complejos y objetos.
- Protección de mensajes de OPC UA para conversaciones seguras.
- Reglas de validación de seguridad.
- Asignaciones del protocolo de transporte: **TCP UA, SOAP/HTTP y HTTPS.**

Parte 7. Perfiles. Esta especificación describe las categorías de comportamientos que servidores y clientes OPC UA pueden implementar, incluyendo:

- Los conceptos de perfiles y unidades de conformidad.
- Categorías de servidores y clientes UA para el comportamiento, soporte de funcionalidad y apoyo a la seguridad.
- En las descripciones se detallan los comportamientos requeridos y comportamientos opcionales de cada perfil, incluyendo perfiles anidados.

Parte 8. Acceso a datos. Aquí se especifican las aplicaciones con acceso a datos, añadiendo:

- Descripción y conceptos de acceso a datos y su evolución a partir de las especificaciones OPC Data Access clásico.
- Descripción de modelo de información y reglas de comportamiento de tipos de datos.
- Organización del espacio de direcciones.
- Descripción y reglas del comportamiento *PercentDeadband*.
- Descripción detallada de los códigos de error que son propios para esta especificación

Parte 9. Alarmas y condiciones. Esta especificación describe las Alarmas y Condiciones de las aplicaciones, introduciendo:

- Descripción y conceptos de Alarmas y Condiciones y su evolución a partir de la especificación del OPC clásico de Alarmas y Eventos.
- Descripción del modelo de información y reglas de comportamiento, de cada tipo de datos y comportamientos esperados.
- Organización del espacio de direcciones.

Parte 10. Programas. Esta especificación describe los programas y la forma en que se puede utilizar en aplicaciones OPC UA:

- Conceptos de un programa, y donde se pueden utilizar, tiempos de vida y estado.
- Descripción de modelo de información y reglas de comportamiento que incluyen tipos de programas, causas y efectos, parámetros y códigos de retorno.
- Aplicación de diagnóstico.
- Implementaciones de programas de ejemplo.

Parte 11. Acceso a Datos Históricos. Estas especificaciones se describe cómo los datos pueden ser archivados y recuperados de una base de datos Histórica:

- Descripción y conceptos de datos y eventos históricos y su evolución a partir de *OPC Classic Historical Data Access*.
- Descripción del modelo de información y reglas de comportamiento para los nodos y cada uno de los tipos de datos y eventos.
- Descripción detallada del comportamiento para crear, recuperar, actualizar y borrar datos archivados y/o anotaciones/notas.

Parte 12. Descubrimiento. Esta especificación describe cómo los productos UA pueden ser detectados y administrados en un equipo, infraestructura de red, o de toda la empresa. Los temas incluyen:

- El proceso de detección.
- Descripción del *Local Discovery Server (LDS)*.
- Descripción del *Global Discovery Server (GDS)*.
- Gestión de certificados para métodos *Push & Pull*.
- Configuración y despliegue.

Parte 13. Agregados. Esta especificación describe el uso de funciones Agregadas para aplicaciones UA:

- Los conceptos de agregados y donde pueden y deben ser incluidos en aplicaciones genéricas e Históricos.
- Las descripciones detalladas y los requisitos de comportamiento de las 37 funciones agregados.
- Descripción del modelo de información y reglas de comportamiento de cada uno de los tipos de datos.
- Extensa biblioteca de material de referencia que muestra ejemplos de consultas y resultados de los distintos agregados.

2.3.3.- CAPAS SOFTWARE EN OPC UA

El concepto cliente-servidor que utiliza OPC-UA es similar al OPC Clásico. Si una aplicación quiere exponer su propia información para otras aplicaciones se llama Servidor de UA en cambio, si una aplicación quiere obtener información de otras aplicaciones se le

Capítulo 2: PROTOCOLO OPC UA

llama Cliente UA. Pueden existir aplicaciones que sean al mismo tiempo cliente y servidor. Gracias a la Arquitectura Unificada, cada vez habrá más Servidores que estén integrados directamente en los dispositivos. Además podrán ser configurados vía OPC UA.

Una aplicación típica OPC UA está compuesta por tres capas de software. En la *ilustración 2.11* se puede ver cómo está formada.

Normalmente la pila de software se puede desarrollar con **C/C++**, **.NET** o **JAVA**. La arquitectura unificada no está limitada a estos lenguajes de programación, ni a determinadas plataformas de desarrollo. Es cierto que actualmente se utiliza este tipo de entornos de trabajo para la implementación de los módulos de la pila UA que ofrece la OPC Foundation.

Cualquier aplicación que quiera trabajar en OPC UA, debe contener la funcionalidad específica para la aplicación empleando una pila OPC UA y un *Software Development Kit (SDK)* de OPC-UA.

La pila implementa diferentes asociaciones de transporte definidas en la *parte 6*. Esta pila se usa para llamar a Servicios UA a través de procesos de redes. La Arquitectura Unificada define tres capas de pila y perfiles diferentes para cada capa.



Ilustración 2.11 Capas Software OPC-UA

Dentro de lo que es la capa de codificación de mensajes se define la señalización de parámetros de servicio en formato binario y **XML**. En la capa de seguridad de mensaje se aseguran los mensajes usando estándares de seguridad basados en Servicios Web o una versión binaria UA de los mismos.

La capa de transporte define los protocolos de red que se usa que pueden ser **UA-TCP**, **HTTP** y **SOAP** para Servicios web. En la *ilustración 2.12*, se pueden comprobar los distintos tipos de capas que están contenidos en la pila de comunicaciones UA. La propia implementación de las capas de la pila y los **APIs** que resulten de la aplicación, no son parte de las especificaciones.

La pila ofrece **APIs** dependientes del lenguaje de programación para aplicaciones clientes y servidores UA, aunque los servicios y sus parámetros son parecidos, ambos están basados en servicios abstractos de la parte cuatro.

Con la implementación de en **ANSI**, **C/C++**, **.NET** y **JAVA**, las pilas cubren los principales entornos de implementación y lenguajes de programación, desarrollados y mantenidos por la OPC Foundation.



Ilustración 2.12 Capas de la pila de comunicación UA definida en la parte 6

2.4.- CONCLUSIONES

El OPC-UA es más flexible y tiene más características en las especificaciones que el OPC clásico. Incorpora conceptos muy importantes que tuvieron buena acogida de la

especificación OPC existente. Además arregla problemas conocidos del estándar existente y añade nuevas estandarizaciones para nuevos casos.

Permite una asociación simple y ofrece nuevas estrategias para poder migrar e integrar productos OPC clásicos. Además hay una parte de la migración que ni siquiera requiere un cambio de los productos existentes. La OPC Foundation ofrece “*wrappers*” y “*proxies*” capaces de traducir las diferentes interfaces OPC Clásicas a OPC-UA y viceversa.

Es importante que para todos los usuarios finales se disponga de los “*wrappers*” y “*proxies*” para hacer de túnel al OPC Clásico a través de cortafuegos. Además incluyen las transmisiones seguras por Internet con acceso autenticado, de forma que añaden valor a soluciones industriales ya probadas.

OPC-UA ofrece conceptos básicos abstractos, modulares y simples en todas las áreas del estándar. La potencia de estos conceptos permite a los fabricantes exponer más características de sus sistemas a través del OPC-UA.

CAPÍTULO 3
PROTOCOLO
IEC-60870

CAPÍTULO 3: PROTOCOLO IEC-60780

3.1.- INTRODUCCIÓN

El protocolo IEC 60870-5 se aplica a equipos y sistemas de telecontrol para la transmisión de datos con bits codificados en serie, para controlar y supervisar procesos en múltiples lugares separados geográficamente. En él se define un sistema de telecontrol estándar que permite la interoperabilidad entre equipos de telecontrol compatibles. Las especificaciones de esta norma presentan un perfil funcional para tareas básicas de telecontrol.

Este estándar define **ASDUs** (*Application Service Data Unit*) con las etiquetas de tiempo *Time2CP24a*, incluye tres octetos binarios de tiempo desde milisegundos a minutos. Además de estas especificaciones, **ASDUs** con las etiquetas de tiempo *CP56Time2a*, incluye siete octetos binarios de tiempo desde milisegundos a años.

Aunque este estándar define las funciones de usuario más importantes, aparte de las funciones de comunicación real, no puede garantizar la plena compatibilidad e interoperabilidad entre equipos de diferentes proveedores. En suma, se requiere normalmente un acuerdo mutuo entre las partes interesadas sobre los métodos de uso de las funciones de comunicación definidos, teniendo en cuenta el funcionamiento de todo el equipo de telecontrol.

3.2.- DEFINICIONES

La dirección de control: Dirección de transmisión desde la estación de control a una estación controlada.

Dirección de monitor: Dirección de transmisión desde una estación controlada a la estación de control.

El parámetro del sistema: Un parámetro del sistema (o parámetros específicos del sistema) incluye todo el sistema de telecontrol que usa este estándar. El sistema de telecontrol consta, de todos los elementos controlados y estaciones de control que pueden ser conectados a través de diferentes configuraciones de red.

Parámetros específicos de la red: Un parámetro específico de la red incluye todas las estaciones que se conectan a través de una configuración de red en particular.

Parámetros específicos de la estación: Un parámetro específico de la estación que incluye estaciones particulares.

Parámetros específicos de objetos: Los parámetros específicos de un objeto se incluyen para un determinado objeto de información o de un grupo específico de objetos informativos.

3.3.- ESTRUCTURA DEL PROTOCOLO

El protocolo IEC 60870-5 Series, se basa en el modelo de referencia de tres capas llamado **EPA** (*Enhanced Performance Architecture*), "Arquitectura Mejorada de Rendimiento", tal como se especifica en la cláusula 4 de la IEC 60870-5-3.

La capa física utiliza las recomendaciones **UIT-T** que proporcionan una transmisión simétrica, binaria y sin memoria, sobre cualquier medio de transmisión, para preservar el alto nivel de integridad de los datos del bloque codificado en la capa de enlace.

La capa de enlace consiste en una serie de procedimientos de transmisión de enlace explícito que usa el **LPCI** (*Link Protocol Control Information*), "Control de Información del Protocolo de Enlace", que son capaces de transportar **ASDUS** como enlace de datos de usuario. La capa de enlace utiliza una selección de formatos de trama para proporcionar la integridad, eficacia y comodidad de la transmisión. Además la aplicación de la capa de usuario contiene una serie de "Funciones de aplicación" que incluyen la transmisión de ASDUs entre el origen y el destino.

La capa de aplicación de este estándar no usa explícitamente el (**APCI**), "*Application Protocol Control Information*" el Protocolo de Aplicación de Control de la Información. Esto está contextualizado en el contenido del campo Identificador de la unidad de datos **ASDU** y en el tipo de servicio de enlace utilizado.

Funciones de la aplicación IEC 60870-5-5		El proceso de usuario
Elementos de la aplicación de la información IEC 60870-5-4		Aplicación (capa 7)
Unidades de datos de servicio de la aplicación IEC 60870-5-3		

Procedimientos de transmisión de enlace IEC 60870-5-2	Enlace (capa 2).
Formatos de tramas de transmisión IEC 60870-5-1	
Recomendaciones UIT-T	Capa física (1)

Tabla 3.1 Capas de la Arquitectura IEC-60870

En la *tabla 3.1*, se puede comprobar cuales son las disposiciones del estándar de telecontrol.

3.3.1.- CAPA FÍSICA

El estándar especifica las recomendaciones UIT-T que definen las interfaces entre el equipo de terminación de circuito de datos (**DCE**) y equipo de terminación de datos (**DTE**), del control y la estación controlada. Esto se puede observar en la *ilustración 3.1*:

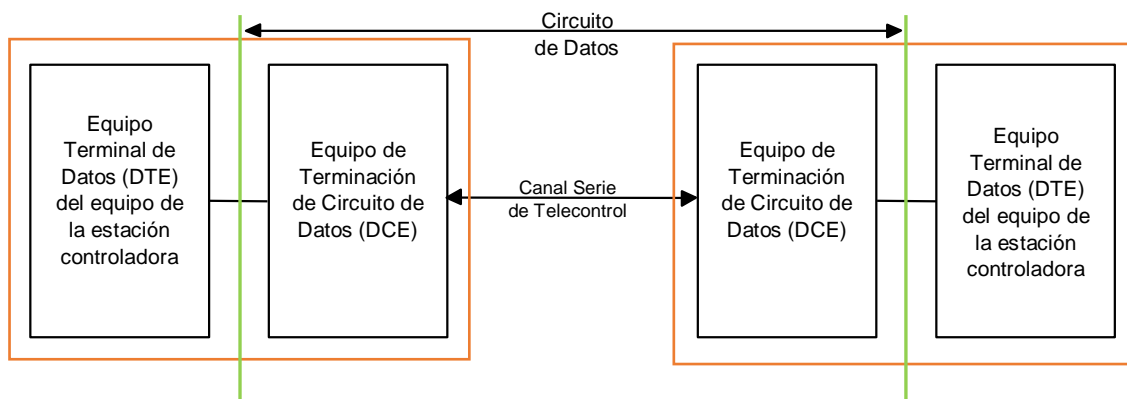


Ilustración 3.1 Interfaces y conexiones de control de las estaciones esclavas.

La interfaz estándar entre el **DTE** y el **DCE** es la interfaz asincrónica **ITU-T V.24/ITU-T V.28**. El uso de las señales de interfaz solicitadas depende del modo de funcionamiento del canal de transmisión utilizado. Por lo tanto el estándar define un conjunto de circuitos de intercambio (señales) que pueden, pero no es necesario, que se usen.

Normalmente los métodos de transmisión de datos que se utilizan para aumentar el aprovechamiento del ancho de banda de un canal de transmisión, deben ser evitados a menos que pueda probarse que el método empleado (que generalmente viola el principio de codificación de canal solicitado sin memoria), no reduce la integridad de los datos del mismo bloque codificado, dentro del formato de trama solicitada en la capa de enlace.

3.3.2.- CAPA DE ENLACE

IEC 60870-5-2 ofrece un conjunto de procedimientos de transmisión de enlace mediante un campo de control y un campo de dirección opcional. La unión entre dos estaciones puede transmitir en modo balanceado y no balanceado. Para ambos modos de operación, se especifican los códigos de función para el campo de control.

Si los enlaces desde una estación central de control (control de la estación), a varias estaciones remotas (estaciones controladas), comparten un canal físico, entonces estos enlaces deben funcionar en un modo no balanceado, para evitar la posibilidad de que más de una estación remota intente transmitir por el mismo canal al mismo tiempo. La secuencia en que las distintas estaciones remotas realizan el acceso para la transmisión en el canal, está determinado por un procedimiento en la capa de aplicación de la estación controlada.

El estándar especifica si el modo de transmisión es o no balanceada, junto con los procedimientos de enlace (y sus correspondientes códigos de función), que serán usados en dicha transmisión.

El estándar especifica una dirección unívoca (número) para cada enlace. Cada dirección puede ser única dentro de un sistema específico, o puede ser única dentro de un grupo de enlaces que comparten un canal común. Este último, necesita un campo de dirección más pequeño pero requiere que la estación de control asigne direcciones por número de canal.

El estándar debe especificar el formato elegido de trama, en este caso, incluidos en IEC 60870-5-1. El formato elegido deberá proporcionar la integridad de los datos junto con la máxima eficiencia para un nivel aceptable de comodidad e implementación. Además, el estándar especifica el tiempo time-out (T_0) de la estación primaria y el tiempo de respuesta máximo permitido (T_r) en la estación secundaria de todos los enlaces.

3.3.3.- CAPA DE APLICACIÓN

El estándar define los **ASDUs** adecuado desde una determinada estructura general en la norma IEC 60870-5-3. Estos **ASDUs** están contruidos utilizando la definición y especificaciones de codificación, para obtener información de la aplicación de los elementos seleccionados en IEC 60870-5-4.

Esta capa especifica la clasificación de transporte elegida para la aplicación de los campos de datos. La clasificación (modo 1 o modo 2) pueden ser elegidos para

proporcionar la máxima comodidad de programación, para los distintos equipos en el sistema de telecontrol de algunas estaciones.

Proceso de usuario

IEC 60870-5-5 ofrece un conjunto de funciones básicas de aplicaciones. El estándar contiene una o más instancias de estas funciones elegidas, para proporcionar el conjunto de los procedimientos de solicitud de entrada/salida, adaptándose a las especificaciones del sistema de telecontrol.

3.4.- CAPA FÍSICA

ISO y los estándares ITU-T

Son compatibles las siguientes configuraciones de red para este protocolo:

- Punto a punto.
- Punto a punto múltiple.
- Multipunto en estrella
- Multipunto en anillo

El caso que nos ocupa este proyecto, no se va explicar la capa física del protocolo IEC-60870, debido a que no se va a utilizar para el desarrollo del mismo. En los siguientes apartados se explicará el nivel físico y de enlace del protocolo TCP/IP, dado que el IEC-60870 va a estar incorporado dentro del TCP/IP.

3.4.1 NIVEL FÍSICO TCP/IP

En el nivel físico, nivel 1 (**TCP/IP**) se agrupan todas las normas que se refieren a los aspectos eléctricos y electrónicos de la comunicación entre ordenadores. Por ejemplo:

- **Nivel eléctrico del 0 y del 1:** Código binario de.
- **Tipos de conectores que se pueden usar y uso de cada hilo:** El tipo de conector usado es el RJ-45 y el cable puede ser de CAT5e (100Mbs) y CAT6 (1Gbs), existiendo categorías superiores y tasas con altas tasas de velocidad. También se incluye la fibra óptica.

La capa 1 (nivel 1), tiene una serie de limitaciones con respecto a la capa 2 de enlace.

- No se puede comunicar con las capas superiores.

- No pueden identificarse los dispositivos.
- Sólo reconoce stremas de bits.
- No puede determinar la fuente de la transmisión cuando transmiten múltiples dispositivos.

3.4.2 NIVEL DE ENLACE TCP/IP

El nivel de enlace (nivel 2 TCP/IP), regula la comunicación entre dos equipos que están en la misma red local.

Se encarga por ejemplo de:

- Establecer la forma de acceder al medio.
- Forma de dividir la información a transmitir en “paquetes”.
- Saber si un paquete se ha recibido correctamente o no (detección y corrección de errores por CRC).
- Control de flujo entre PCs. (desbordamiento).

La estructura de la trama de Ethernet agrega encabezados y *tráilers* a la PDU de *Capa 3* para encapsular el mensaje que se envía. Hay dos estilos de tramas de Ethernet: el estándar DIX Ethernet, que ahora es Ethernet II, y el estándar IEEE 802.3, que ha sido actualizado varias veces para incluir nuevas tecnologías. Las diferencias entre los estilos de tramas son mínimas.

Tamaño de la trama de Ethernet

Tanto el estándar Ethernet II como el IEEE 802.3 definen el tamaño mínimo de trama en 64 bytes y el tamaño máximo de trama en 1518 bytes. Esto incluye todos los bytes del campo Dirección **MAC** de destino a través del campo Secuencia de verificación de trama (**FCS**). Los campos Preámbulo y Delimitador de inicio de trama no se incluyen en la descripción del tamaño de una trama. El estándar IEEE 802.3ac, publicado en 1998, amplió el tamaño de trama máximo permitido a 1522 bytes. Se aumentó el tamaño de la trama para que se adapte a una tecnología denominada Red de área local virtual (VLAN).

Si el tamaño de una trama transmitida es menor que el mínimo o mayor que el máximo, el dispositivo receptor descarta la trama. Es posible que las tramas descartadas se originen en colisiones u otras señales no deseadas y, por lo tanto, se consideran no válidas.



Ilustración 3.2 Comparativa de los campos de la trama Ethernet y IEEE 802.

CAMPO	SIGNIFICADO
Campos Preámbulo y Delimitador de inicio de trama	Los campos Preámbulo (7 bytes) y Delimitador de inicio de trama (SFD) (1 byte) se utilizan para la sincronización entre los dispositivos emisores y receptores. Estos ocho primeros bytes de la trama se utilizan para captar la atención de los nodos receptores. Básicamente, los primeros bytes le indican al receptor que se prepare para recibir una trama nueva.
Campo Dirección MAC de destino	El campo Dirección MAC de destino (6 bytes) es el identificador del receptor deseado. La dirección de la trama se compara con la dirección MAC del dispositivo. Si coinciden, el dispositivo acepta la trama.
Campo Dirección MAC de origen	El campo Dirección MAC de origen (6 bytes) identifica la NIC o interfaz de origen de la trama
Campo Longitud/tipo	Para todos los estándares IEEE 802.3 anteriores a 1997. El campo Longitud define la longitud exacta del campo de datos de la trama. Esto se utiliza posteriormente como parte de la FCS para garantizar que el mensaje se reciba adecuadamente. Si el objetivo de un campo es designar un tipo como en Ethernet II, el campo Tipo describe cuál es el protocolo que se implementa.
Campos Datos y Pad.	Los campos Datos y Pad (de 46 a 1500 bytes) contienen los datos encapsulados de una capa superior, que es una PDU de Capa 3 genérica o, con mayor frecuencia, un paquete IPv4 . Todas las tramas deben tener al menos 64 bytes de longitud. Si se encapsula un paquete pequeño, el Pad se utiliza

	para incrementar el tamaño de la trama hasta alcanzar el tamaño mínimo
Campo Secuencia de verificación de trama	El campo Secuencia de verificación de trama (FCS) (4 bytes) se utiliza para detectar errores en la trama. Utiliza una comprobación cíclica de redundancia (CRC). El dispositivo emisor incluye los resultados de una CRC en el campo FCS de la trama.

Tabla 3.2 Campos de la trama Ethernet

La capa de enlace del IEC-60780, está encapsulada en el campo **Campos Datos y PAD.**, del TCP/IP.

3.5.- CAPA DE ENLACE IEC-60870

3.5.1.- FORMATOS DE LA TRAMA DE TRANSMISIÓN

Esta especificación admite exclusivamente el formato de trama estándar **FT1.2** que está definido en la *sub-cláusula 6.2.4.2 de la IEC 60870-5-1*. Son admitidos formatos con longitud de bloque fija y variable. También está admitido el carácter de control I transmisión único. Una trama con longitud variable tiene que ser usada cuando una **ASDU** va a ser transmitida. Cuando una **ASDU** no va a ser transmitida, deben ser usadas o tramas con longitud fija o el carácter único.

La trama es básicamente asíncrona, con la distribución de cada uno de los caracteres que lo constituyen de 11 bits, comenzando con su primer bit y parando con su último bit. Sin embargo, cuando se utiliza con la interfaz sincrónica, definida anteriormente, la señal de sincronización de elementos se deriva de la **DCE** y se ejecuta de forma continua. En este caso, la trama debe ser transmitida y recibida de forma sincrónica.

La regla de transmisión **R3** establece, que no son admitidas ninguna línea inactiva entre caracteres. Esto puede que no sea posible conseguirlo en algunas implementaciones prácticas, especialmente con alta tasa de bits de transmisión, debido a los inevitables retrasos de hardware o software.

Aun así se demuestra que en tiempos inferiores a una hora, no existe el problema de que la integridad del mensaje sufra algún tipo de percance. Por lo tanto, las reglas de transmisión **R3** pueden ser flexibles, permitiendo la línea inactiva en intervalos de hasta una hora de duración de bits transmitidos entre caracteres. El mayor problema surge con la sincronización de los relojes, reduciendo su precisión en las estaciones controladas.

3.5.2.- IEC 60870-5-2: PROCEDIMIENTOS DE TRANSMISIÓN DE ENLACE

La longitud máxima de tramas de enlace, se establece de forma fija (parámetro específico de la red). Puede ser diferente, si se requiere la máxima longitud para cada dirección. La trama con longitud fija no tiene ningún vínculo con datos del usuario.

Los procedimientos de transmisión **SEND/NO REPLY**, **SEND/CONFIRM** and **REQUEST/RESPOND**, han de utilizarse cuando sea necesario. La interfaz entre la capa de enlace y el usuario de servicio no está definida en este estándar.

3.5.3.- DIAGRAMAS DE TRANSICIÓN DE ESTADO

Esta subclase añade más detalles a las definiciones base de los procedimientos de enlace incluidos en IEC 60870-5-2. Los diagramas de transición de estados se utilizan para definir los procedimientos de forma más exacta, así como enlazar capas aplicadas por diferentes fabricantes para que puedan ser completamente interoperables. Los diagramas de transición de estados representan los estados (en este caso la capa de enlace se define en IEC 60870-5-2) y las transiciones de un estado a otro. Están incluidas las acciones (enviar una trama Tx y recibir una trama y Rx). Además de los estados, se describen los procesos internos importantes.

Los diagramas de transición de estados se presentan en el formato definido por *Grady Booch/Harel*. La explicación de los elementos particulares se muestra en la *ilustración 3.3*.

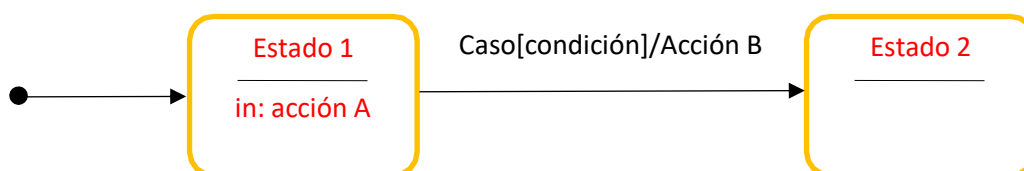


Ilustración 3.3 Diagrama de Transición de Grady Booch/Harel

La palabra “*in*” describe una acción que se desencadena cuando se produce una transición hacia un nuevo estado. La transición al siguiente estado puede ser desencadenada por la terminación del actual estado, en el caso donde no se haya definido ningún evento a causa de la transición.

La notación utilizada en los siguientes diagramas de transición de estado es:

- **FC0** a **FC15** = Número de código función de 0 a 15, habría que consultar las Tablas 1 a 4 de la especificación IEC 60870-5-2.
- **FCB** = bit cuenta de tramas.

- **FCV** = número válido de tramas de bit.
- **DFC** = control de flujo de datos.
- **ACD** = demanda de acceso.
- **PRM** = mensaje principal.
- **SC** = carácter único.

3.5.4.- PROCEDIMIENTOS DE TRANSMISIÓN NO BALANCEADA

En el sistema de transmisión no balanceada, las estaciones remotas son siempre secundarias (esclavas). El centro de control es la primaria (master). En un sistema jerárquico, todos modos intermedios en dirección a las estaciones remotas son primarios, y en dirección al centro de control son secundarios.

No se utiliza en el campo de control del bit RES, y se ajusta a cero. El campo de dirección A de la capa de enlace es uno de los 2 octetos que está determinado por un parámetro de sistema fijo. El número de la dirección de broadcast (servicio **SEND/NO REPLY**) es de 255, (correspondiente a un octeto) o 65535 (correspondiente a 2 octetos). El **SEND/NO REPLY** utiliza el servicio de respuesta al emitir un mensaje a todos los datos del usuario de destino (dirección de difusión).

En este estándar no existen direcciones de grupo.

El procedimiento de transmisión básica utiliza la función **SEND/RESPOND** de servicio código 11 (solicitar datos de usuario clase 2). Los datos de clase 1 se indican a través del bit **ACD** definido en IEC 60870-5-2. La asignación de las causas de la transmisión para las dos clases se define en el estándar. Las estaciones controladas que no tienen datos de Clase 2 disponibles, pueden responder a la solicitud de clase 2 con datos de clase 1.

Los códigos de función y servicios en la dirección principal	Los códigos de función y servicios permitidos en el sentido secundario
<0> Reinicio del enlace remoto	<0> Confirmar: ACK o <1> Confirmar: NACK
<1> Reinicio del proceso de usuario	<0> Confirmar: ACK o <1> Confirmar: NACK
<3> SEND/CONF datos de usuario	<0> Confirmar: ACK o <1> Confirmar: NACK
<4> SEND/NO REPLY Enviar datos de usuario	Sin Respuesta.
<8> REQUEST Solicitud de demanda de acceso	<11> RESPOND: Estado de enlace
<9> REQUEST/RESP Solicitar estado de enlace	<11> RESPOND: Estado de enlace
<10> REQUEST/RESP Solicitar datos de usuario clase 1	<8> RESPOND: Datos de usuario o <9> RESPOND: Datos solicitados no disponible
<11> REQUEST/RESP Solicitar datos de usuario clase 2	<8> RESPOND: Datos de usuario o <9> RESPOND: Datos solicitados no disponible

Tabla 3.3 Combinaciones permitidas de la capa de servicios de enlaces no balanceados

La *tabla 3.3* muestra las combinaciones permisibles de los procedimientos de la capa de enlace en transmisiones no balanceadas.

Otros códigos como **<14>** significa que el servicio de enlace no funciona. **<15>** Servicio de enlace no implementado. El carácter de control único **E5** puede usarse en lugar de una longitud fija **CONFIRM ACK** (código de función secundaria **<0>**) o longitud fija **RESPOND NACK** (código de función secundaria **<9>**), excepto cuando hay una demanda de acceso para datos de clase 1 (**ACD = 1**) u otros mensajes, pueden provocar un desbordamiento (**DFC = 1**). El carácter único A2 no debe ser usado.

Para los procedimientos de transmisión no balanceada, la estación principal contiene sólo la capa de enlace principal y la estación secundaria sólo contiene una capa de enlace secundario como se puede comprobar en la figura anterior. Se puede conectar a una estación primaria más de una estación secundaria. La compatibilidad de la comunicación entre la estación principal y una estación secundaria se basa únicamente en estas dos estaciones. El procedimiento de sondeo para solicitar datos de múltiples estaciones secundarias locales, es una función interna de la estación principal y no necesita ser mostradas. Estos diagramas (*ilustraciones 3.4, 3.5 y 3.6*) sólo muestran la estación primaria y una única estación secundaria. En el caso de más de una estación secundaria, la estación principal tiene que recordar el estado actual de cada estación secundaria.

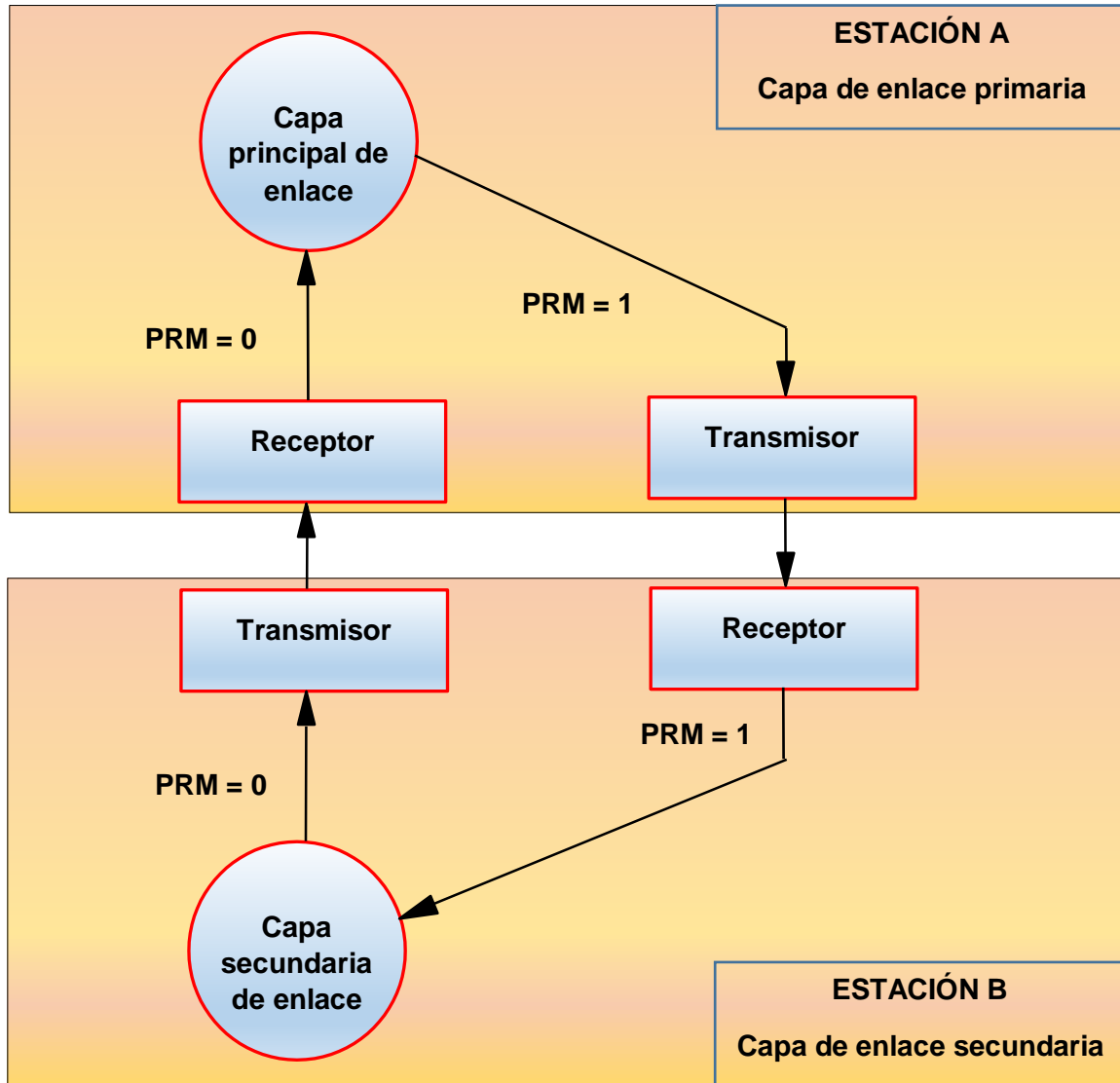


Ilustración 3.4 Procedimiento de transmisión no balanceada, estaciones primaria y secundaria

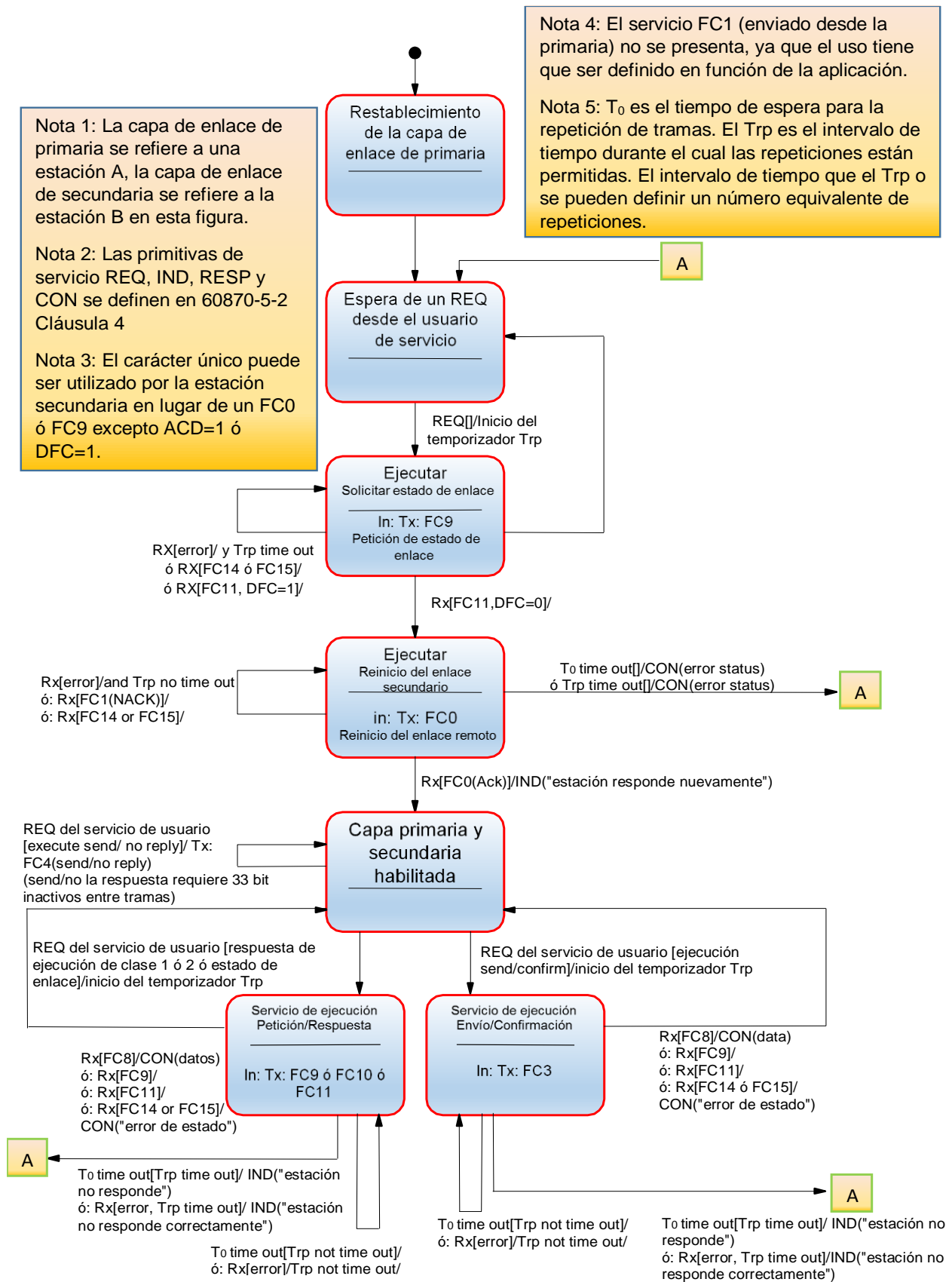


Ilustración 3.5 Diagramas de estados de transición de la estaciones primaria y secundaria para transmisiones no balanceadas

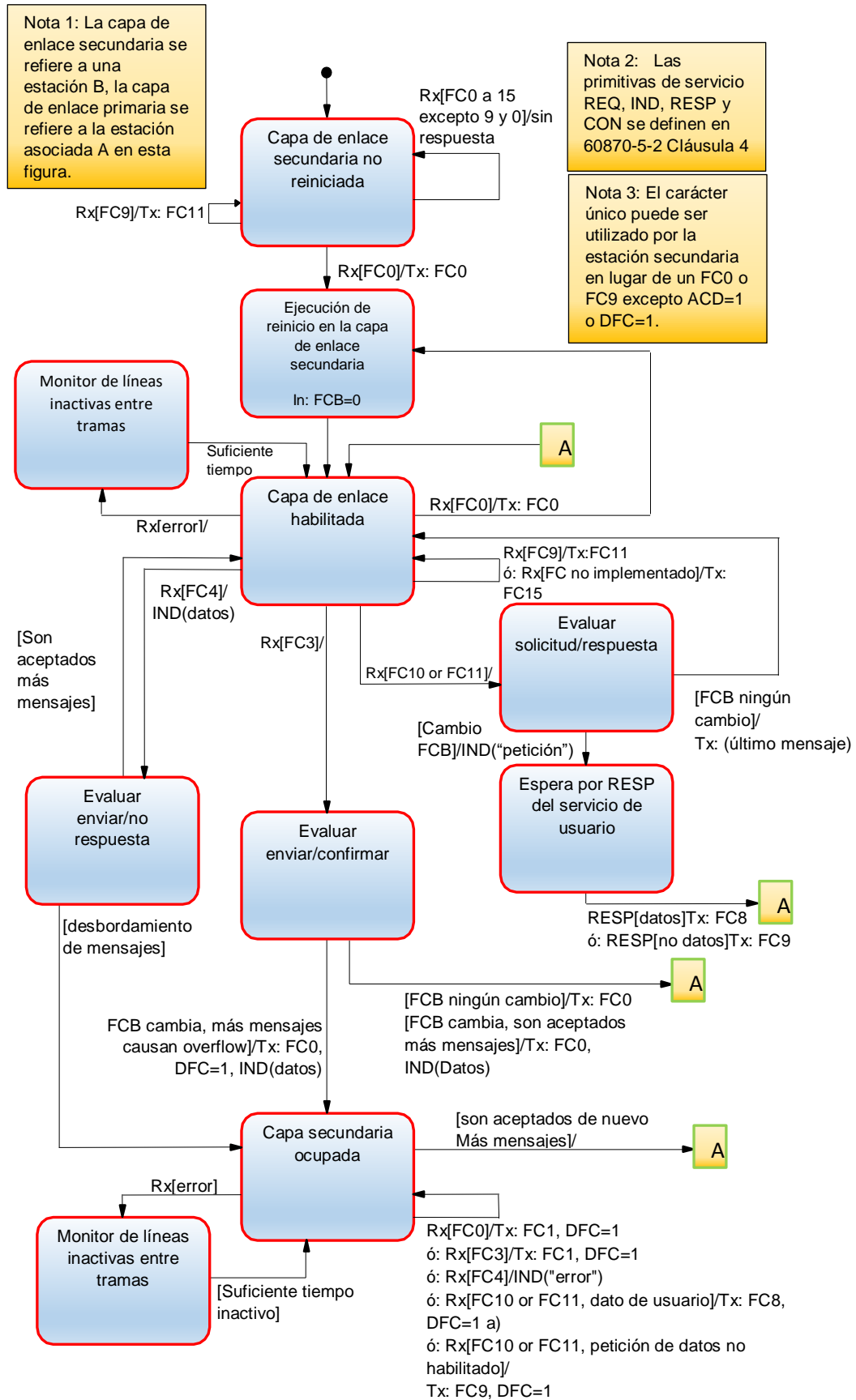


Ilustración 3.6 Diagramas de estados de transición de las estaciones primaria y secundaria para transmisiones no balanceadas. Procedimientos para transmisiones balanceadas

La solicitud de todos los códigos de función estandarizada con dirección primaria (0 hasta 4 y 9), han de recibir respuestas positivas y negativas. En el caso de un servicio no implementado, la estación secundaria tiene que responder con el código de función 15 para vincular el servicio no implementado. La *tabla 3.4* muestra las combinaciones permitidas de los procedimientos balanceados de la capa de enlace.

Los códigos de función y servicios en la dirección principal	Los códigos de función y servicios permitidos en el secundario
<0> Reinicio de comunicación remota	<0> CONFIRM: ACK o <1> CONFIRM: NACK
<1> Reinicio del proceso de usuario	<0> CONFIRM: ACK o <1> CONFIRM: NACK
<2>SEND/CONF Función de prueba de enlace	<0> CONFIRM: ACK o <1> CONFIRM: NACK
<3> SEND/CONF datos de usuario	<0> CONFIRM: ACK o <1> CONFIRM: NACK
<4> SEND/NO REPLY datos de usuario	Sin respuesta
<9> REQUEST/RESP solicitar estado de enlace	<11>RESPOND: estado de enlace

Tabla 3.4 Combinaciones permitidas de transmisión balanceada en la capa de enlace

Otros códigos como **<14>** significa que el servicio de enlace no funciona. **<15>** Servicio de enlace no implementado. El carácter de control único **E5** puede usarse en lugar de una longitud fija **CONFIRM ACK** (código de función secundaria **<0>**) excepto cuando otros mensajes pueden provocar un desbordamiento (**DFC = 1**).

El campo de dirección **A** es opcional. Si se define, consiste en uno o dos octetos especificados por el sistema. En sistemas de transmisiones balanceadas, no está definido el comando de difusión. El bit **RES** en el campo de control no se usa y se coloca a cero.

La capa de enlace para una transmisión balanceada, consiste en dos procesos lógicos desacoplados, un proceso lógico se representa como estación primaria a la estación A y como estación secundaria a la estación B y en otro proceso lógico representa como estación primaria a la estación B y como estación secundaria a la estación A (cada estación es una estación combinada). Por lo tanto, existen dos procesos independientes en cada estación para el control de la capa de enlace lógico, la dirección primaria y la secundaria. En la *figura 3.6* se muestra la disposición de la capa de enlace usando transmisión balanceada.

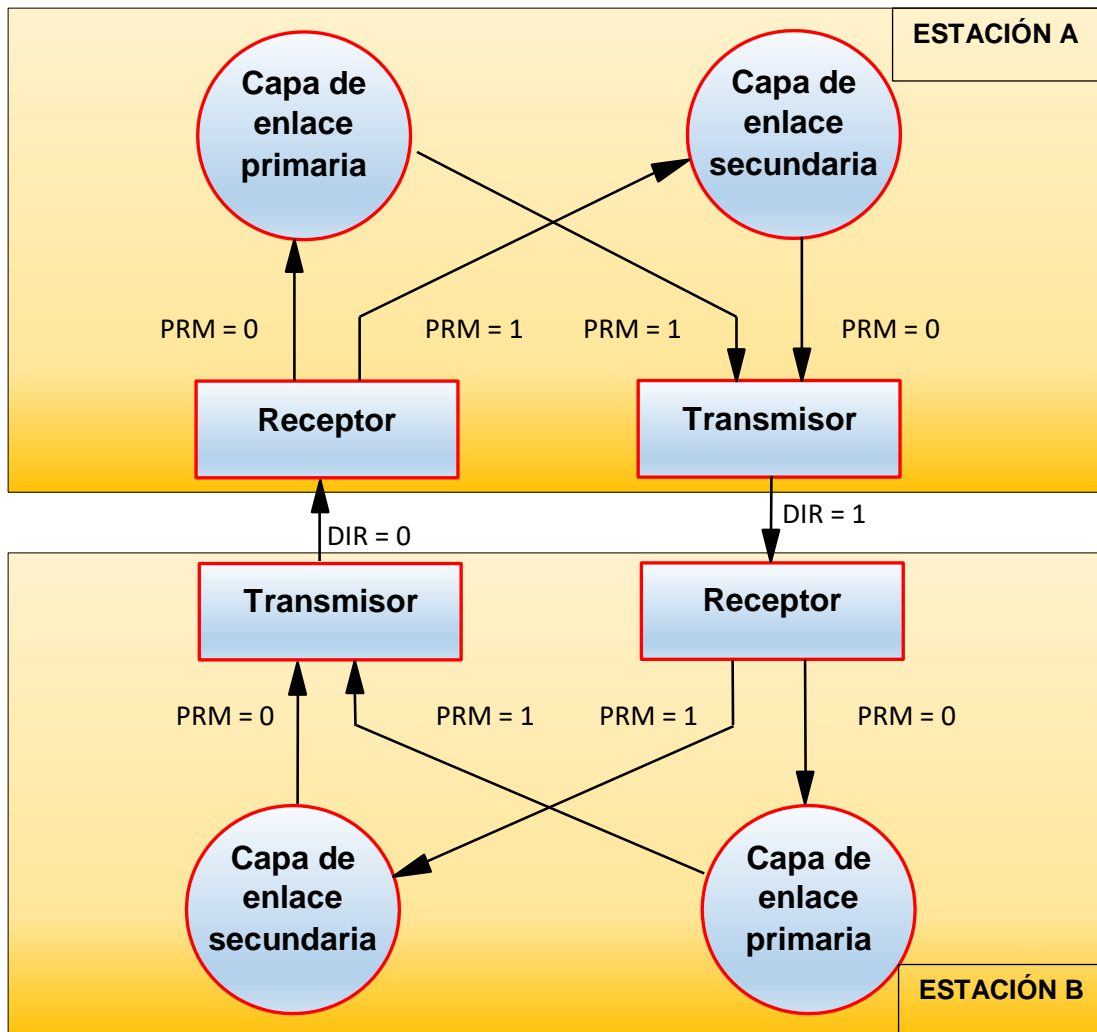


Ilustración 3.7 Procedimientos de transmisión balanceada. Capa de enlace del primario y secundario

En las figuras siguientes no se muestran las reacciones de la capa de enlace en el caso de recibir tramas dañadas. Estas tramas son rechazadas por un proceso que no se ve en las *ilustraciones 3.8 y 3.9*. Este proceso es además es responsable del control del intervalo *time out*.

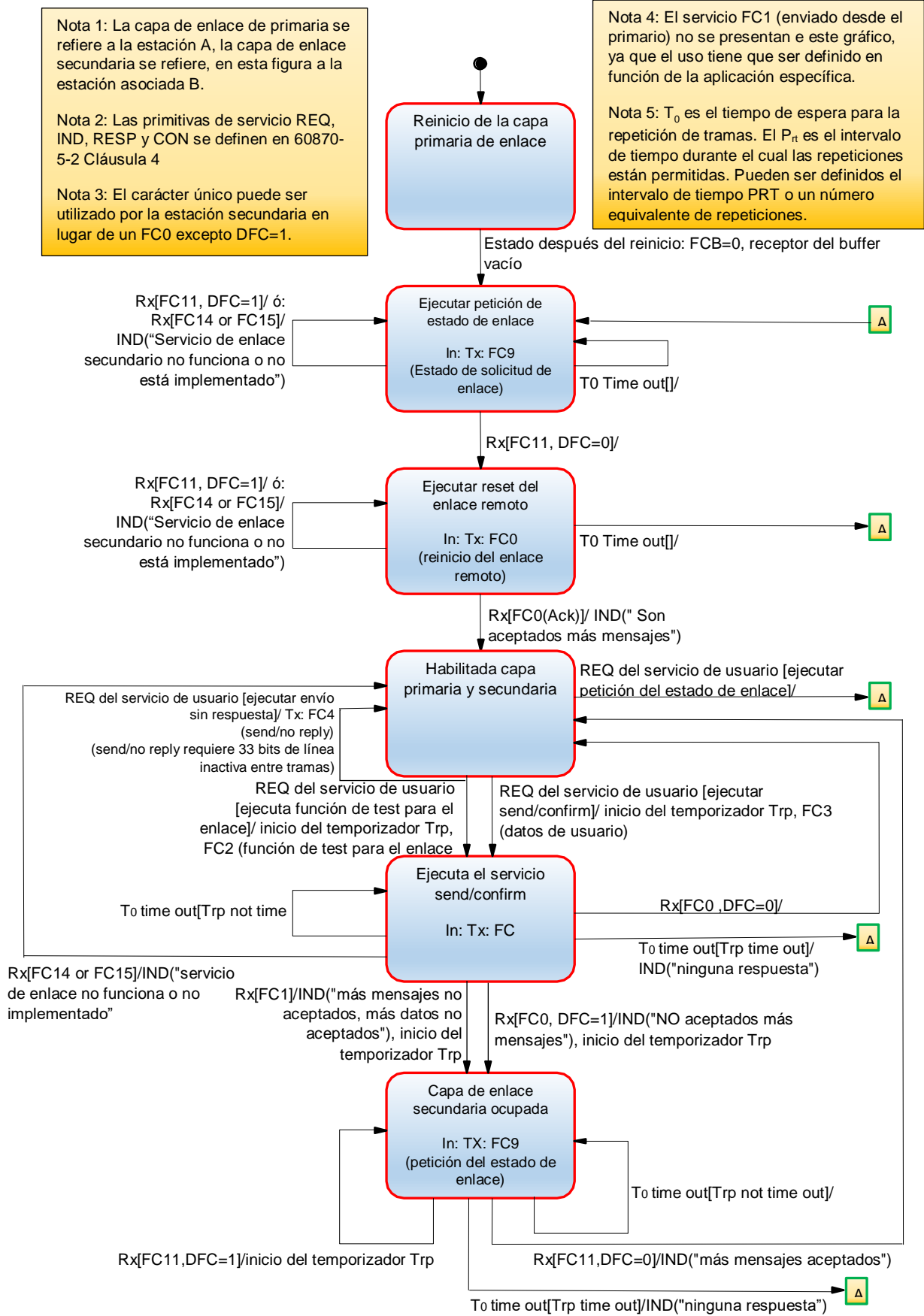


Ilustración 3.8 Diagramas de estado de transición del primario y secundario para transmisiones balanceadas

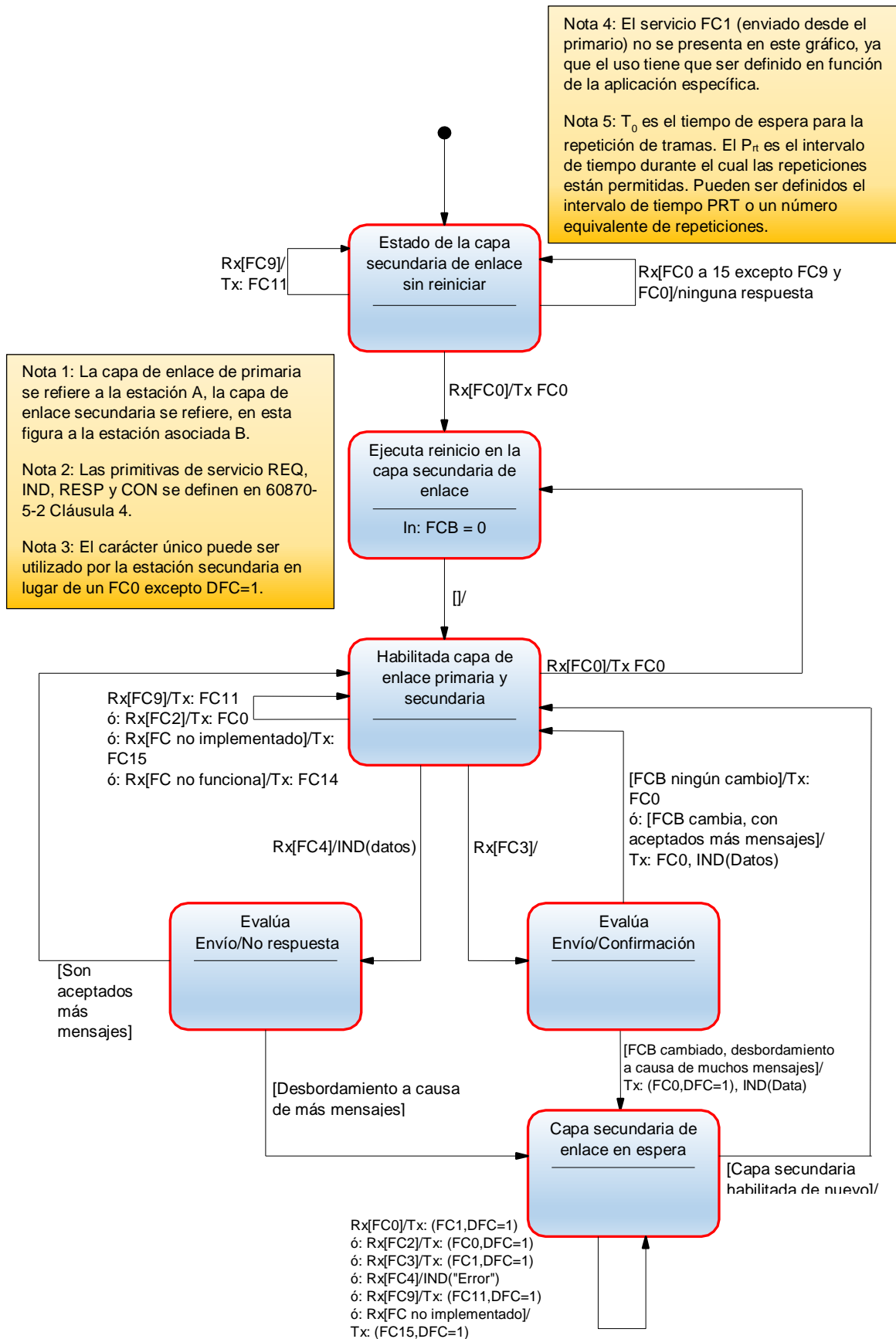


Ilustración 3.9 Diagramas de estado de transición del primario y secundario para transmisiones balanceadas

CAPÍTULO 4

WIRESHARK

CAPÍTULO 4: WIRESHARK

4.1.- ¿QUÉ ES WIRESHARK?

Wireshark es un analizador de paquetes de red. Un analizador de paquetes de red que intenta capturar paquetes de red e intenta mostrar los datos de la forma más detallada posible.

Un analizador de paquetes de red es un software de medición utilizado para examinar lo que ocurre en el interior de un cable de red. Wireshark tiene como principal característica el hecho de ser un programa de código abierto y a día de hoy un referente como software.

4.2.- UTILIZACIÓN DE WIRESHARK

- Los administradores de red utilizan para resolver problemas de red.
- Los ingenieros de seguridad de red lo usan para examinar los problemas de seguridad.
- Los desarrolladores lo usan para depurar las implementaciones de protocolos.
- Las personas (en general) lo utilizan para aprender el protocolo de red internos.
- Junto a estos ejemplos Wireshark puede ser útil en otras situaciones.

4.3.- CARACTERÍSTICAS

- Disponible para **UNIX** y **Windows**.
- Captura paquetes de datos en tiempo real desde una interfaz de red.
- Abre archivos que contengan datos de paquetes capturados con *tcpdump/WinDump*, Wireshark, y de otros programas de captura de paquetes.
- Importa paquetes de archivos de texto que contienen datos hexadecimales de paquetes de datos.
- Genera una pantalla de paquetes con información de protocolos muy detallado.
- Guarda datos de paquetes capturados.
- Exportar algunos o todos los paquetes en diferentes formatos de archivo de captura.
- Filtra paquetes bajo múltiples criterios.

- Busca paquetes con diferentes criterios.
- Colorea la pantalla de paquetes según los filtros establecidos.
- Crea diversas estadísticas.

Wireshark puede capturar tráfico de diferentes tipos de medios de red incluyendo acceso inalámbrico a internet. Los tipos de medios compatibles dependen de muchas cuestiones como, por ejemplo, el sistema operativo que se está utilizando. Una descripción general de los tipos de archivos multimedia compatibles pueden encontrarse en su [página web](#).

Este programa es capaz de importar y/o exportar paquetes capturados tanto en diferentes formatos como de otros programas.

Este programa es un proyecto de software de código abierto y está liberado bajo la [GNU General Public License](#) **GPL**. Se puede usar libremente Wireshark en cualquier número de ordenadores que se quieran, sin preocuparse de las claves de licencia o comisiones. Además, todo el código fuente está disponible libremente bajo licencia **GPL**. Por eso, es muy fácil para el público en general, añadir nuevos protocolos, plugins, o cambios en el código fuente.

4.4.- REQUISITOS DEL SISTEMA

La cantidad de recursos que Wireshark necesita depende de su entorno y del tamaño del archivo de captura que está analizando. Los valores sobre unos pocos cientos de **MB**, están bien para medianos y pequeños archivos de captura. Los archivos de captura de mayor tamaño requerirán más memoria y espacio en disco.

Si se trabaja con una red con tráfico intenso, fácilmente puede producir enormes archivos de captura. La captura de datos en un corto período de tiempo, puede producir de 1 **Gigabit** o incluso cientos de **Megabits**. Siempre es aconsejable un procesador rápido, mucha memoria y espacio en disco. Si se agota la memoria de Wireshark, éste se bloqueará.

Wireshark captura los paquetes mediante un proceso independiente de la interfaz principal, tiene un único subproceso. Esto hace que los sistemas multi-núcleos no se beneficien de este software.

4.5.- INSTALACIÓN

En primer lugar para poder ser instalado este programa hace falta descargarse el instalador desde la página [web](#) del mismo. En dicha página web, se encuentran varios instaladores según las características que tenga el sistema operativo. En el caso de este proyecto se ha instalado la versión de 64 bits, y posteriormente se instaló la versión de este software para **Linux**. La instalación en **Linux** se explicará en el *capítulo 6* y el por qué ha sido necesaria para este proyecto.

4.5.1.- COMPONENTES DE INSTALACIÓN

- **Wireshark** → El analizador de protocolos.
- **TShark** → Línea de comandos analizador de protocolos de red.
- **Wireshark 1 Legacy** → El viejo interfaz de usuario (GTK+).
- **Plugins y extensiones** → Extras para el Wireshark y TShark.
 - **Dissector Plugins** - Plugins con algunas opciones de análisis detallados.
 - **Tree Statistics Plugins** – Extensión de estadísticas.
 - **Mate - Meta Analysis and Tracing Engine** – Extensión configurable de la pantalla de filtros.
 - **SNMP MIBs** – Estudio detallado de paquetes SNMP MIBs.
- **Herramientas** → Otras herramientas de línea de comandos para trabajar con archivos de captura.
 - **Editcap** → Lee un archivo de captura y escribe algunos o todos los paquetes en otro archivo de captura.
 - **Text2Pcap** → Lee un carácter ASCII hexadecimal y escribe los datos en un archivo de captura pcap.
 - **Reordercap** → Reordena un archivo de captura por *timestamp*.
 - **Mergecap** → Combina varios archivos de captura guardado en un único archivo de salida.
 - **Capinfos** → Proporciona información sobre archivos de captura.
 - **Rawshark** → Filtro de paquetes Raw.
- **Guía del usuario** → La instalación local de la Guía del usuario. Los botones de Ayuda en la mayoría de diálogos, requiere de una conexión a internet para mostrar páginas de ayuda, si la Guía del usuario no está instalada localmente.

El instalador de Wireshark se encarga de la instalación de *WinPcap*. Sólo es necesario si se desea utilizar una versión diferente a la que se incluye en el instalador de

Wireshark por ejemplo, debido a una nueva versión liberada de *WinPcap*. Este software es necesario para que funcione y puede ser descargado de <https://www.winpcap.org/>.

4.6.- INTERFAZ DE USUARIO

4.6.1.- LA VENTANA PRINCIPAL

Un vez se accede al programa aparece la ventana de la *ilustración 4.1*. En ella se muestran los nombres de distintos ficheros que se han abierto o han sido usados anteriormente. En la parte inferior están los distintos dispositivos de captura que reconoce el wireshark en el equipo.

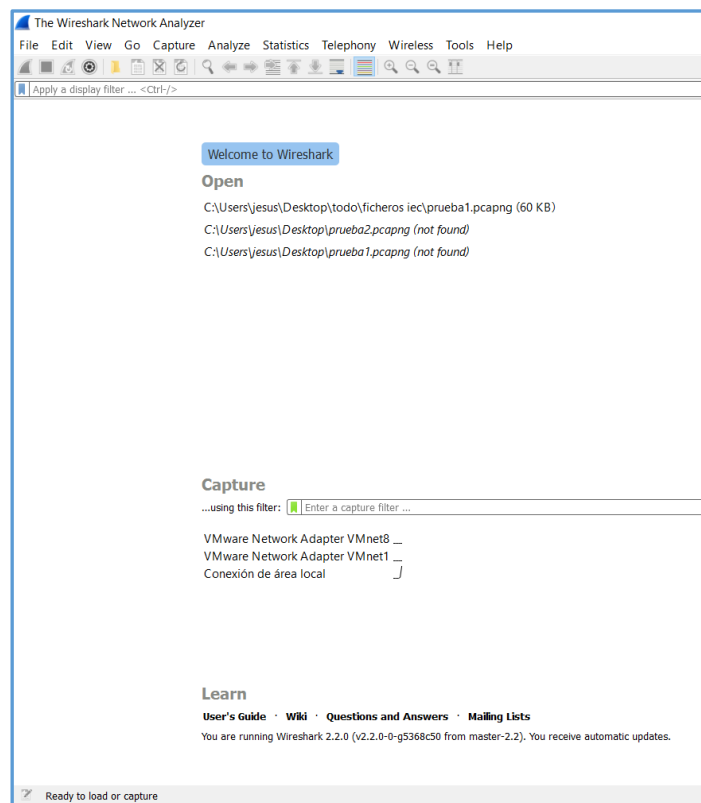


Ilustración 4.1 Ventana de arranque de Wireshark

La ventana principal consta de elementos que son comúnmente conocidos por otros programas GUI, como se muestra en la *ilustración 4.2*.

1. El **menú** se utiliza para iniciar acciones.
2. La **barra de herramientas principal** proporciona un acceso rápido a elementos utilizados con frecuencia en el menú.
3. La **barra de herramientas Filtro** proporciona una manera de manipular directamente el filtro de visualización utilizado en el momento de la captura.
4. El **panel Lista de paquetes**, el panel muestra un resumen de todos los paquetes capturados. Haciendo clic en los paquetes en este panel, permite controlar lo que aparece en los otros dos paneles.
5. El **panel de detalles del paquete** muestra el paquete seleccionado en el panel de lista de paquetes con más detalle.
6. El **panel packet bytes** muestra los datos del paquete seleccionado en el panel de lista de paquetes, y resalta el campo seleccionado en el panel de detalles del paquete.
7. La **barra de estado** muestra información detallada sobre el estado actual del programa y los datos capturados.

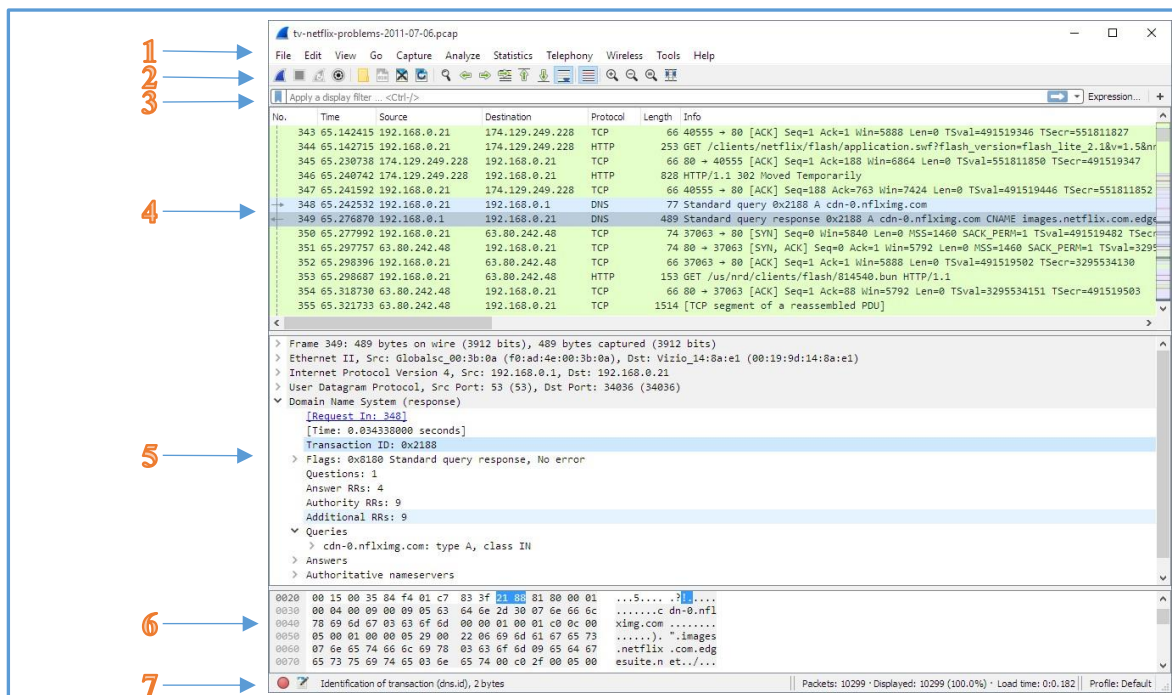


Ilustración 4.2 Ventana general de Captura de Wireshark

4.6.2.- EL MENÚ

El menú principal (*ilustración 4.3*), está situado en la parte superior de la ventana principal (**Windows, Linux**).

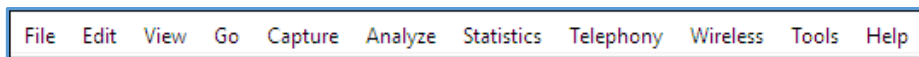


Ilustración 4.3 Barra de menú principal del Wireshark


El menú principal contiene los siguientes elementos:

- **File** → Este menú contiene opciones para abrir y combinar archivos de captura, guardar, imprimir o exportar archivos de captura en todo o en parte, y para cerrar la aplicación Wireshark.
- **Edit** → Este menú contiene opciones para encontrar un paquete, el tiempo de referencia o marcaje de uno o más paquetes, manejar perfiles de configuración y configurar sus preferencias; (cortar, copiar y pegar no se aplica actualmente).
- **View** → Este menú controla la visualización de los datos capturados, incluyendo la colorización de paquetes, zoom de la fuente, mostrando un paquete en una ventana separada, la expansión y contracción de los árboles en los detalles del paquete.
- **Go** → Este menú contiene opciones para ir a un determinado paquete.
- **Capture** → Este menú le permite iniciar y detener la captura y editar filtros de captura.
- **Analyze** → Este menú contiene opciones para manipular, mostrar filtros, habilitar o deshabilitar la disección de protocolos, configura el usuario especificado descodifica y sigue un flujo TCP.
- **Statistics** → Este menú contiene opciones para mostrar varias ventanas de estadísticas, incluyendo un resumen de los paquetes que han sido capturados, el protocolo de visualización jerarquía estadística y más opciones.
- **Telephony** → Este menú contiene opciones para mostrar diversas estadísticas relacionadas con telefonía de Windows, incluyendo un análisis de los medios, diagramas de flujo, el protocolo de visualización jerarquía estadística y más opciones.
- **Wireless** → Los elementos de este menú muestran el Bluetooth inalámbrico IEEE 802.11.
- **Tools** → Este menú contiene diversas herramientas disponibles en Wireshark, como la creación de cortafuegos, reglas ACL, etc...
- **Help** → Este menú contiene opciones para ayudar al usuario, por ejemplo, el acceso a algunas páginas del manual de ayuda básica, de las diversas herramientas de línea de comandos, el acceso en línea a algunas de las páginas web, y el habitual diálogo Acerca de.

4.7.- FUNCIONAMIENTO

En primer lugar buscamos el filtro necesario para que el programa seleccione solamente los paquetes para cada protocolo. En el caso que se está desarrollando en este proyecto, los 2 filtros que se utilizan son:

- **Para el OPCUA** → opcua
- **Para el IEC-60870** → 104asdu

Una vez seleccionado cada uno de los filtros hay pulsar el botón . Éste está situado en la parte superior derecha.

Para la captura de paquetes, hay que dirigirse al menú **Capture** y seleccionar la opción de **Options** (*ilustración 4.5*).

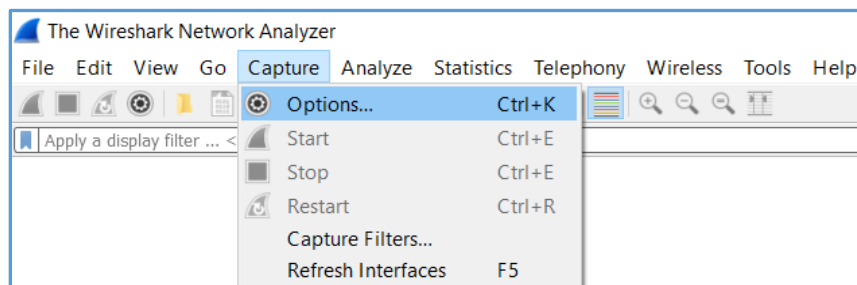


Ilustración 4.5 Menú desplegable de la opción Capture

Una vez se selecciona esta opción aparece la ventana (*ilustración 4.6*). En ésta aparecen los distintos interfaces que ha detectado. En este caso se selecciona la **Conexión de área local** y seguidamente se pulsa el botón **Start**.

Entonces el sistema empieza a capturar los paquetes según el filtro que se la haya puesto. Si no se introduce ningún filtro, el software captura todos los paquetes que circulan por el dispositivo elegido. En la *ilustración 4.8*, se puede comprobar como se ha filtrado los paquetes para el IEC-60870.

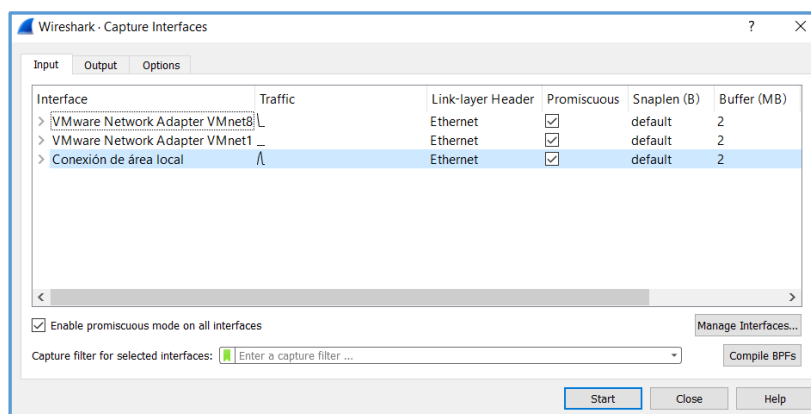


Ilustración 4.6 Ventana de selección del dispositivo de captura

Capítulo 4: WIRESHARK

Una vez se haya hecho la captura, primero hay que salvar el fichero (**SAVE**) y seguidamente exportar los datos seleccionando **File** → **Export Packet Dissections** → **As PDML XML**, como se indica en la *ilustración 4.7*.

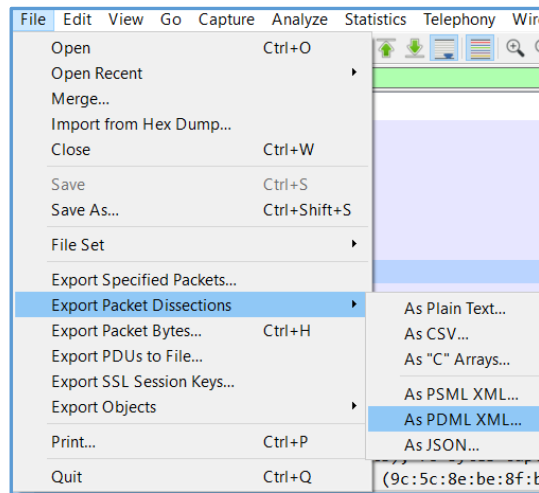


Ilustración 4.7 Ventana guardar datos en XML

The screenshot shows the Wireshark interface with a packet capture from 'Conexión de área local'. The main pane displays a list of 16 packets, all of type '104asdu'. The selected packet (No. 140) is expanded to show the 'Frame 38: 79 bytes on wire (632 bits), 79 bytes captured (632 bits) on interface 0'. The details pane shows the following structure:

- Ethernet II, Src: Asusstek_b... (9c:5c:8e:be:8f:be), Dst: VMware_37:f0:b5 (00:0c:29:37:f0:b5)
- Internet Protocol Version 4, Src: 192.168.1.50, Dst: 192.168.1.52
- Transmission Control Protocol, Src Port: 2404, Dst Port: 2404, Seq: 1, Ack: 1, Len: 25
- IEC 60870-5-104-Asdu: ASDU=0 M_ME_TD_1 Sport 10A=0 "measured value, normalized value with time tag CP56Time2a"

The hex and ASCII panes show the raw data for the selected packet:

```

0000  00 0c 29 37 f0 b5 9c 5c 8e be 8f be 08 00 45 00  ..J... \.....E.
0010  00 41 56 c3 40 00 80 06 00 00 c0 a8 01 32 c0 a8  .AV@... ..2..
0020  01 34 09 64 09 64 23 75 5c 3b 40 09 8b 35 50 18  .4.d#u \;.5P.
0030  08 05 83 ea 00 00 68 17 02 00 00 00 22 01 03 00  .....h....."
0040  00 00 00 00 00 01 00 00 5b 02 1b 09 01 0a 10  ..... [.....
  
```

At the bottom, the status bar indicates 'Packets: 390 · Displayed: 16 (4.1%)' and 'Profile: Default'.

Ilustración 4.8 Ejemplo de Captura del protocolo IEC-60870

4.7.1.- FICHERO XML (detallado)

El fichero que se genera está etiquetado según el formato XML. En él se detallan cada uno de los paquetes y de los campos que se han capturado según cada protocolo. En la siguiente figura se muestra parte del contenido que genera el wireshark tipo XML para el protocolo IEC-60780. Hay que recordar que este fichero es de texto plano etiquetado.

```
<field name="104asdu.cp56time" showname="CP56Time: Jun 23, 2016
18:52:39.810000000 Hora de verano GMT" size="7" pos="72" show="Jun
23, 2016 18:52:39.810000000 Hora de verano GMT"
value="829b3412170610">
  <field name="104asdu.cp56time.ms" showname="1001 1011 1000
0010 = MS: 39810" size="2" pos="72" show="39810" value="9B82"
unmaskedvalue="829b"/>
  <field name="104asdu.cp56time.min" showname="..11 0100 =
Min: 52" size="1" pos="74" show="52" value="34"
unmaskedvalue="34"/>
  <field name="104asdu.cp56time.iv" showname="0... .. =
IV: Valid" size="1" pos="74" show="0" value="0"
unmaskedvalue="34"/>
  <field name="104asdu.cp56time.hour" showname="...1 0010 =
Hour: 18" size="1" pos="75" show="18" value="12"
unmaskedvalue="12"/>
```

En la siguiente figura se muestra parte del contenido el fichero generado para OPCUA.

```
<field name="opcua.Timestamp" showname="Timestamp: Jun 4, 2016
18:27:33.356941800 WEST" size="8" pos="151" show="Jun 4, 2016
18:27:33.356941800" value="8aa70c6186bed101"/>
  <field name="opcua.RequestHandle"
showname="RequestHandle: 0" size="4" pos="159" show="0"
value="00000000"/>
  <field name="opcua.ReturnDiagnostics"
showname="ReturnDiagnostics: 0" size="4" pos="163" show="0"
value="00000000"/>
  <field name="opcua.AuditEntryId" showname="AuditEntryId:
[OpCua Null String]" size="0" pos="171" show=""/>
  <field name="opcua.TimeoutHint" showname="TimeoutHint:
0" size="4" pos="171" show="0" value="00000000"/>
```

4.8.- CAMPOS SELECCIONADOS

4.8.1.- OPCUA

Aquí se describen los distintos campos capturados por el software que se ha implementado para este proyecto relativo al protocolo OPCUA.

CAMPO	SIGNIFICADO
timestamp	Tiempo de captura del paquete (tiempo proporcionado por la propia tarjeta) en microsegundos
frame.time_delta	Tiempo de respuesta (confirmación ACK WireShark)
eth.dst	Dirección física de destino
eth.src	Dirección física del origen
ip.src	Dirección lógica de origen
ip.dst	Dirección lógica de destino
transport.endpoint	Dirección final del paquete (puede ser un enlace por ejemplo opc.tcp://localhost:48010)
frame.number	Número de paquete
tcp.analysis.acks_frame	Número de trama confirmada en este paquete
opcua.AuditEntryId	Identifica el cliente o el usuario que ha iniciado la acción; utilizado en caso de auditoría
opcua.TimeoutHint	Tiempo de espera para la llamada de servicio de cliente en 'UA Pila; cuando termina, el servidor cancela la llamada entrante
opcua.RequestedLifetime	Indica cuánto es el intervalo sin que transcurra ninguna actividad supervisada por el cliente. Pasado este tiempo, el servidor suprime los recursos de suscripción y se libera de la misma. Este parámetro debe ser por lo menos 3 veces más grande que los posibles retardos que pudieran surgir. Es negociado por el cliente y el servidor
opcua.RequestHandle	Identificador asignado a la llamada de servicio, que se define por el cliente
opcua.ReturnDiagnostics	Indica si el cliente solicita el servidor para proporcionar, en caso de error, datos de diagnóstico en lugar de sólo un código de estado simple
opcua.String	Cadena de caracteres con información diversa

Tabla 4.1 Campos capturados OPC-UA

4.8.2.- IEC-60870

En este apartado se describen los distintos campos capturados por el software que se ha implementado para este proyecto relativo al protocolo IEC-60870.

CAMPO	SIGNIFICADO
timestamp	Tiempo de captura del paquete (tiempo proporcionado por la propia tarjeta) en microsegundos
frame.time_delta	Tiempo de respuesta (confirmación ACK WireShark)
eth.dst	Dirección física de destino
eth.src	Dirección física del origen

ip.src	Dirección lógica de origen
ip.dst	Dirección lógica de destino
transport.endpoint	Dirección final del paquete (puede ser un enlace por ejemplo opc.tcp://localhost:48010)
frame.number	Número de paquete
tcp.analysis.acks_frame	Número de trama confirmada en este paquete
104asdu.cp56time.ms	Campo CP56Time2a del estándar IEC-60870 tiempo binario (segundos). Se utiliza para la sincronización del reloj.
104asdu.cp56time.min	Campo CP56Time2a del estándar IEC-60870 tiempo binario (minutos). Se utiliza para la sincronización del reloj.
104asdu.cp56time.hour	Campo CP56Time2a del estándar IEC-60870 tiempo binario (horas). Se utiliza para la sincronización del reloj.
104asdu.cp56time.day	Campo CP56Time2a del estándar IEC-60870 tiempo binario (día). Se utiliza para la sincronización del reloj.
104asdu.cp56time.month	Campo CP56Time2a del estándar IEC-60870 tiempo binario (mes). Se utiliza para la sincronización del reloj.
104asdu.cp56time.year	Campo CP56Time2a del estándar IEC-60870 tiempo binario (año). Se utiliza para la sincronización del reloj.

Tabla 4.2 Campos capturados IEC-60870

Como de se ha explicado anteriormente en el *capítulo 3*, el IEC-60870 trabaja con **ASDUs**. Entre ellos está, como se comenta en ese *capítulo 3*, el *CP56Time2a*. En la *ilustración 4.9*, se especifica como está configurada la trama de ese campo en forma de octetos según las especificaciones del estándar.

7	6	5	4	3	2	1	0	Octeto	Rango
								1	0 .. 59999 ms
								2	
		Minutos						3	0 .. 59 min
			Horas					4	0 .. 23 h
Día de la semana=0			Día de Mes					5	0, 1 .. 31
				Mes				6	1 .. 12
Año								7	0 .. 99

Ilustración 4.9 Trama de datos de captura del IEC-60780

CAPÍTULO 5
SIMULADORES DE
PROTOCOLOS

CAPÍTULO 5: SIMULADORES DE PROTOCOLOS

5.1.- SIMULADOR OPC UA

El simulador para este tipo de protocolo, ha sido desarrollado por la compañía *Unifed Automation*. Esta compañía de origen alemán permite descargar los simuladores desde su propia página [web](#). Este software tanto para el servidor como para el cliente, se han descargado una vez se registró un usuario válido. Con la descarga no sólo se ha bajado los programas ejecutables, sino también el código fuente de dicho software. Debido al funcionamiento del programa wireshark, se ha tenido que descargar estos paquetes para trabajar bajo Linux.

5.1.1.- WIRESHARK EN LINUX

En el *capítulo 4* ya se ha explicado el funcionamiento de este software de licencia libre y cuál ha sido su función en este proyecto. Este software tiene unas circunstancias muy particulares a la hora de trabajar sobre una misma máquina. En modo **Windows** este programa no permite extraer datos bajo una dirección “*loopback*”, es decir, no se puede extraer información que circule dentro de la propia máquina, bajo una dirección **IP 127.0.0.1**. Los simuladores que trabajan tanto en entorno Windows como Linux sólo pueden ejecutarse bajo una misma dirección, es decir, tanto el cliente como el servidor están en el mismo entorno (misma máquina) con la misma dirección **127.0.1.1**.

Hay que destacar que estos simuladores se han podido descargar de forma gratuita y libre, y que por lo tanto no permite ningún cambio en cuanto a las direcciones tanto del cliente como del servidor.

En la *ilustración 5.1* se puede comprobar como el WireShark para **Linux**, es capaz de capturar paquetes de la dirección “*loopback*”, algo no que puede hacer este mismo software en Windows.

Para poder instalar el Wireshark en **Linux**, se debe ejecutar el siguiente comando:

```
# apt-get install wireshark
```

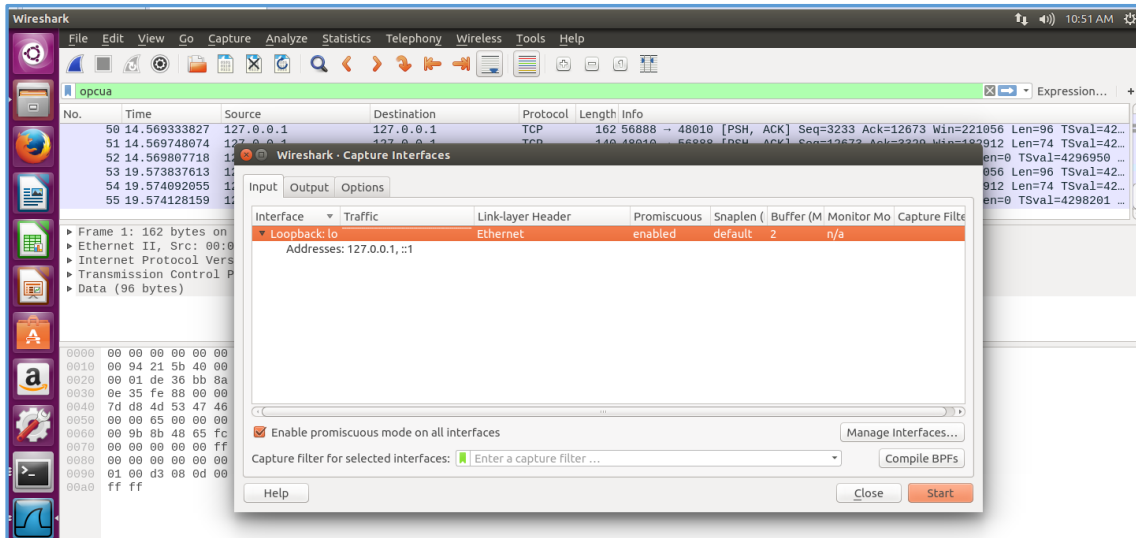



Ilustración 5.1 Ventana de Selección de dispositivo de Wireshark para Linux

Una vez se haya instalado el software, habría que ejecutarlo en modo *root*, para que pudiera funcionar de forma adecuada. Esto es debido a que el capturador de paquetes necesita usar algunas librerías y herramientas que sólo le está permitido al superusuario, como puede ser el uso de la tarjeta de red.

5.1.2.- CARPETAS Y ARCHIVOS

Cuando se realiza la descarga de los simuladores, el contenido viene comprimido en un archivo **tar.gz**. Una vez se descomprime dicha carpeta en ella se puede encontrar varias sub-carpetas y archivos, tal y como se muestra en la *ilustración 5.2*.

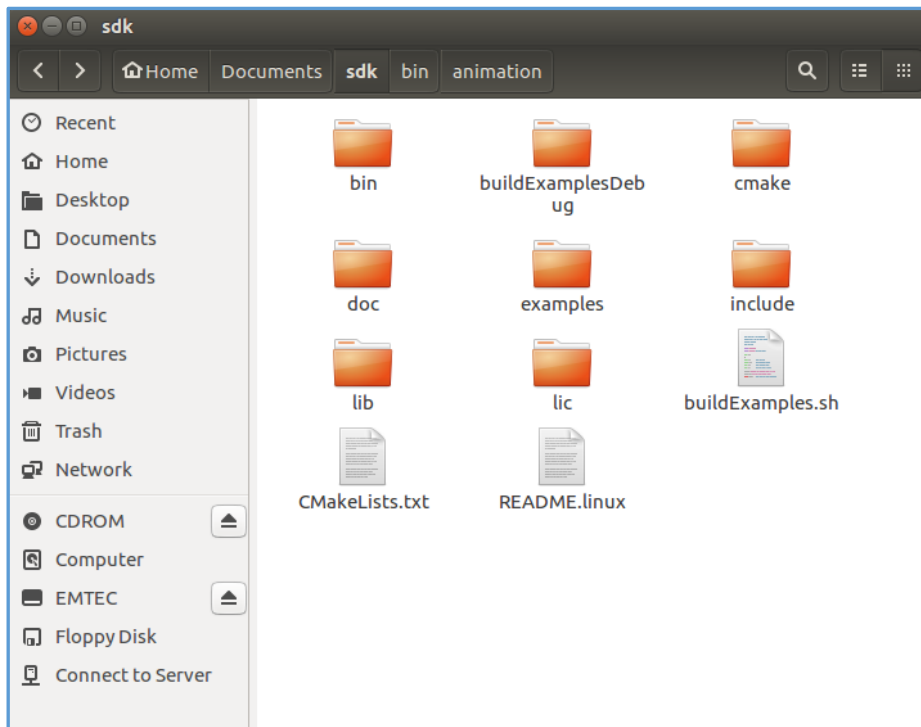


Ilustración 5.2 Carpeta de contenido del simulador OPC-UA de Unified Automation

El contenido de cada carpeta sería el que se adjunta en la *tabla 5.1*:

CARPETA	CONTENIDO
bin	Los ficheros ejecutables tanto del servidor como del cliente
buildExamplesDebug	Ejemplos depurados de los servidores y clientes
cmake	Ficheros para poder ejecutar la compilación
doc	Tutorial en html de las especificaciones de cada módulo junto con las especificaciones del OPC UA
examples	Código fuente de distintos ejemplos de servidores y clientes
include	Código fuente de los módulos base de los servidores y clientes
lib	Librerías necesarias para la compilación del código fuente
lic	Las distintas licencias de uso

Tabla 5.1 Contenido de las carpetas del simulador OPC UA

5.1.3.- CLIENTE OPC UA

El cliente OP UA, tiene una serie de requisitos que cumplir según el estándar unificado. Estos requisitos están especificados en el *capítulo 2* de esta memoria.

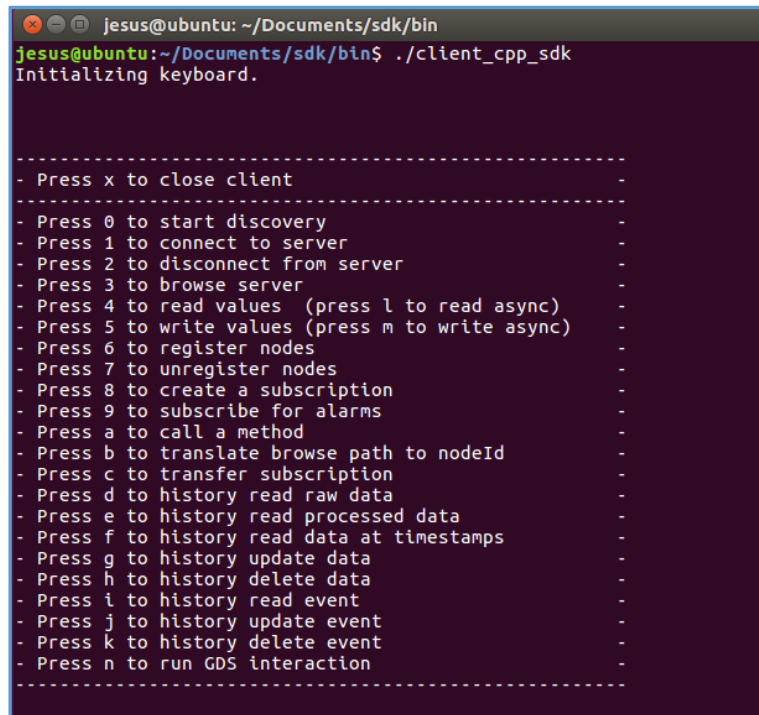
Para poder ser ejecutado en **Linux** el cliente debe ir a la carpeta **bin**. Para ello se puede ejecutar en un terminal, el comando:

```
# cd sdk/bin
```

Una vez se haya situado en esta carpeta se debe ejecutar el siguiente comando:

```
# ./client_cpp_sdk
```

Entonces aparecerá la siguiente ventana:



```
jesus@ubuntu: ~/Documents/sdk/bin
jesus@ubuntu:~/Documents/sdk/bin$ ./client_cpp_sdk
Initializing keyboard.

-----
- Press x to close client -
-----
- Press 0 to start discovery -
- Press 1 to connect to server -
- Press 2 to disconnect from server -
- Press 3 to browse server -
- Press 4 to read values (press l to read async) -
- Press 5 to write values (press m to write async) -
- Press 6 to register nodes -
- Press 7 to unregister nodes -
- Press 8 to create a subscription -
- Press 9 to subscribe for alarms -
- Press a to call a method -
- Press b to translate browse path to nodeId -
- Press c to transfer subscription -
- Press d to history read raw data -
- Press e to history read processed data -
- Press f to history read data at timestamps -
- Press g to history update data -
- Press h to history delete data -
- Press i to history read event -
- Press j to history update event -
- Press k to history delete event -
- Press n to run GDS interaction -
-----
```

Ilustración 5.3 Entorno de simulación del cliente OPC-UA

Una vez aparece la ventana anterior, simplemente se debe teclear cada una de las opciones que aparecen en este menú. Estas opciones son elementos estándares del OPC UA. Para la generación de paquetes necesarios y poder ser capturados y posteriormente analizados, primeramente se debe conectar al servidor pulsando la **tecla 1**. Seguidamente para la generación de tramas más seguidas se ha utilizado la **opción 4 y 5**. El resto de funciones trabajan correctamente ya que dicho software ha sido certificado por la OPC Foundation, y por lo tanto cumple con todos los requisitos del estándar.

Para tener la certeza de que la conexión se ha establecido nos tiene que aparecer esta ventana (*ilustración 5.4*).

En ella se puede comprobar cómo el cliente está conectado con el servidor. A partir de aquí se pueden ejecutar cualquiera de las opciones del menú. Para cerrar dicha ventana pulsaremos la **tecla x**.

```
jesus@ubuntu: ~/Documents/sdk/bin
clientConnectionId 0
serverStatus Connected
-----
** Connected to Server opc.tcp://localhost:48010 with security mode:
http://opcfoundation.org/UA/SecurityPolicy#None ****
*****
-----
- Press x to close client -
-----
- Press 0 to start discovery -
- Press 1 to connect to server -
- Press 2 to disconnect from server -
- Press 3 to browse server -
- Press 4 to read values (press l to read async) -
- Press 5 to write values (press m to write async) -
- Press 6 to register nodes -
- Press 7 to unregister nodes -
- Press 8 to create a subscription -
- Press 9 to subscribe for alarms -
- Press a to call a method -
- Press b to translate browse path to nodeId -
- Press c to transfer subscription -
- Press d to history read raw data -
- Press e to history read processed data -
- Press f to history read data at timestamps -
- Press g to history update data -
- Press h to history delete data -
- Press i to history read event -
- Press j to history update event -
```

Ilustración 5.4 Ventana de confirmación establecida del cliente OPC-UA

5.1.4.- SERVIDOR OPC UA

Al igual que pasa con el cliente, este servidor cumple con todos los requisitos del estándar OPC UA. Para poder ser ejecutado también hay que seguir los siguientes pasos:

El usuario se debe ir a la carpeta bin ejecutando en un terminal, el comando:

```
# cd sdk/bin
```

Una vez se haya situado en esta carpeta se debe ejecutar el siguiente comando:

```
# ./uaservercpp
```

Entonces aparecerá la siguiente ventana (*ilustración 5.5*):

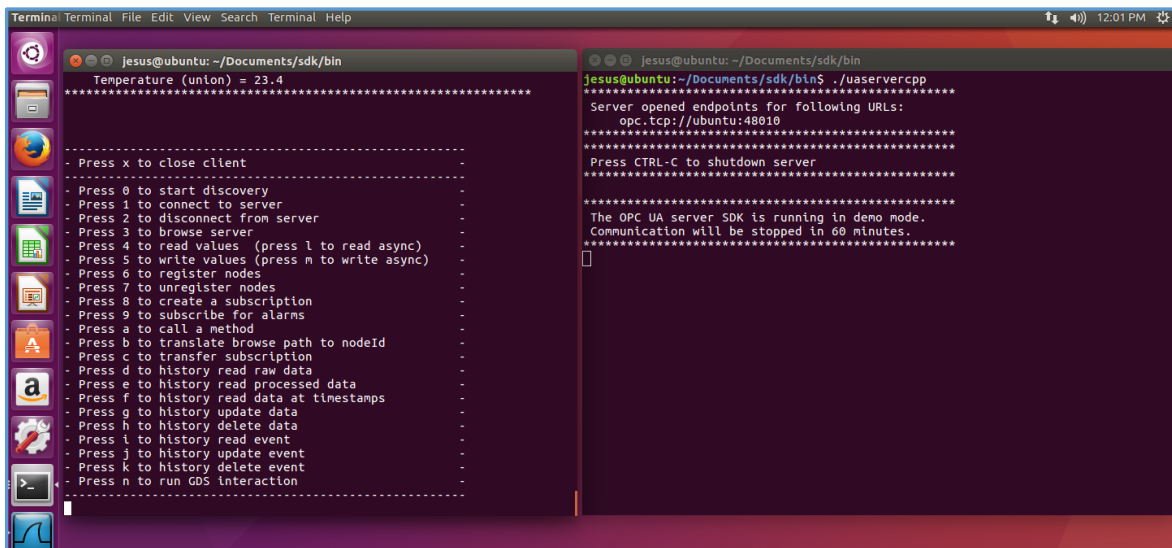
```
Jesus@ubuntu: ~/Documents/sdk/bin
Jesus@ubuntu:~/Documents/sdk/bin$ ./uaservercpp
*****
Server opened endpoints for following URLs:
opc.tcp://ubuntu:48010
*****
Press CTRL-C to shutdown server
*****

The OPC UA server SDK is running in demo mode.
Communication will be stopped in 60 minutes.
*****
```

Ilustración 5.5 Entorno de trabajo del servidor OPC-UA

Esto significa que el servidor está marcha y que al ser una versión demo solo se puede usar durante **60 minutos**. Si se quiere apagar el servidor debemos pulsar las teclas **Ctrl+c**.

El sistema completo quedaría de la siguiente forma (ilustración 5.6):



The image shows two terminal windows side-by-side. The left window displays the output of a client application, showing a temperature reading of 23.4 and a list of keyboard shortcuts for various OPC-UA operations. The right window shows the server application output, indicating that the server has started and is running in demo mode for 60 minutes.

```
Termina Terminal File Edit View Search Terminal Help 12:01 PM
Jesus@ubuntu: ~/Documents/sdk/bin
Temperature (untion) = 23.4
*****
-----
- Press x to close client
-----
- Press 0 to start discovery
- Press 1 to connect to server
- Press 2 to disconnect from server
- Press 3 to browse server
- Press 4 to read values (press l to read async)
- Press 5 to write values (press m to write async)
- Press 6 to register nodes
- Press 7 to unregister nodes
- Press 8 to create a subscription
- Press 9 to subscribe for alarms
- Press a to call a method
- Press b to translate browse path to nodeId
- Press c to transfer subscription
- Press d to history read raw data
- Press e to history read processed data
- Press f to history read data at timestamps
- Press g to history update data
- Press h to history delete data
- Press i to history read event
- Press j to history update event
- Press k to history delete event
- Press n to run GDS interaction
-----

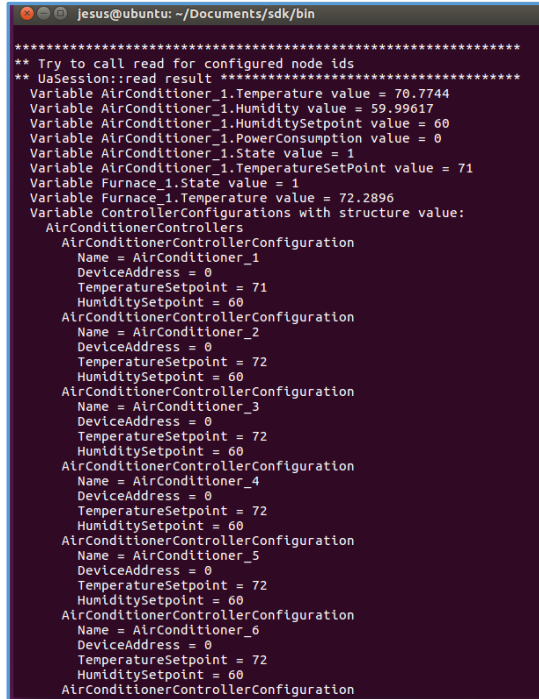
Jesus@ubuntu:~/Documents/sdk/bin$ ./uaservercpp
*****
Server opened endpoints for following URLs:
opc.tcp://ubuntu:48010
*****
Press CTRL-C to shutdown server
*****

The OPC UA server SDK is running in demo mode.
Communication will be stopped in 60 minutes.
*****
```

Ilustración 5.6 Conjunto de ventanas de trabajo cliente y servidor OPC-UA

5.1.5.- EJEMPLO

Para el desarrollo final de este proyecto se han hecho diferentes simulaciones con distintas capturas. Un ejemplo básico que se ha seguido, ha sido la lectura de datos del servidor. Éste para poder ser simulado correctamente, tiene una serie de valores preestablecidos en su memoria, que mediante la lectura (**opción 4 “read values”**), se obtendrían los valores aparecidos en la siguiente figura (*ilustración 5.7*):



```
jesus@ubuntu: ~/Documents/sdk/bin
*****
** Try to call read for configured node ids
** UaSession::read result *****
Variable AirConditioner_1.Temperature value = 70.7744
Variable AirConditioner_1.Humidity value = 59.99617
Variable AirConditioner_1.HumiditySetpoint value = 60
Variable AirConditioner_1.PowerConsumption value = 0
Variable AirConditioner_1.State value = 1
Variable AirConditioner_1.TemperatureSetPoint value = 71
Variable Furnace_1.State value = 1
Variable Furnace_1.Temperature value = 72.2896
Variable ControllerConfigurations with structure value:
AirConditionerControllers
AirConditionerControllerConfiguration
Name = AirConditioner_1
DeviceAddress = 0
TemperatureSetpoint = 71
HumiditySetpoint = 60
AirConditionerControllerConfiguration
Name = AirConditioner_2
DeviceAddress = 0
TemperatureSetpoint = 72
HumiditySetpoint = 60
AirConditionerControllerConfiguration
Name = AirConditioner_3
DeviceAddress = 0
TemperatureSetpoint = 72
HumiditySetpoint = 60
AirConditionerControllerConfiguration
Name = AirConditioner_4
DeviceAddress = 0
TemperatureSetpoint = 72
HumiditySetpoint = 60
AirConditionerControllerConfiguration
Name = AirConditioner_5
DeviceAddress = 0
TemperatureSetpoint = 72
HumiditySetpoint = 60
AirConditionerControllerConfiguration
Name = AirConditioner_6
DeviceAddress = 0
TemperatureSetpoint = 72
HumiditySetpoint = 60
AirConditionerControllerConfiguration
```

Ilustración 5.7 Transferencia de datos del Servidor al Cliente

De esta forma la simulación se asemeja más a la realidad de cualquier tipo de servidor, que esté enviando distintos parámetros para poder ser tratados.

5.2.- SIMULADOR IEC – 60870

Este simulador también ha sido descargado con una licencia demo. Se trata del programa “WinPP104”. Este software ha sido diseñado por *Reinhard Fink* un ingeniero alemán que tiene desarrollado distintos tipos de productos para venderlos a diferentes empresas. Como ha pasado anteriormente con los simuladores OPC UA, es suficiente este simulador para el objetivo de este trabajo fin de carrera.

En primer lugar, hay que descargarse el software directamente de la página [web](#) de este ingeniero. El programa necesario para poder simular correctamente este protocolo es el “WinPP104”. El proceso de instalación es bastante sencillo, simplemente hay que

ejecutar el archivo que se descarga y seguir los pasos estándares para este tipo de software.

Para poder trabajar correctamente con este programa, se ha instalado una máquina virtual **Windows**, para que funcione entre dos equipos diferentes (dos direcciones IP distintas). En ambas máquinas se ha instalado dicho software y se ha configurado de la siguiente forma:

En la máquina principal una vez instalado el software aparecerá esta ventana (*ilustración 5.8*):

	Received	Error	Transmitted	Error	Status	IP Partner	Cl-,Se-Port	Function
Rec/Tr 1	0	0	0	0	-	192.168.0.30	-,2404	Master
Rec/Tr 2	0	0	0	0	-	192.168.0.30	-,2404	Off

Online Messages, logical, with time, with link

Ilustración 5.8 Ventana de Log cuando se arranca el simulador

Seguidamente se debe configurar los distintos parámetros del transmisor y receptor. Para ello se debe ir a la pestaña **Parameterize** → **Receiver/Transmitter 1...** como se indica en la *ilustración 5.9*:

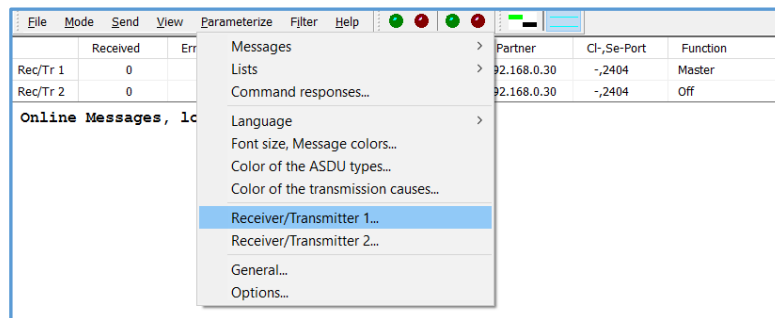


Ilustración 5.9 Ventana de selección para la configuración del Emisor/Receptor

Una vez ha seleccionado dicha opción se introduce los siguientes parámetros:

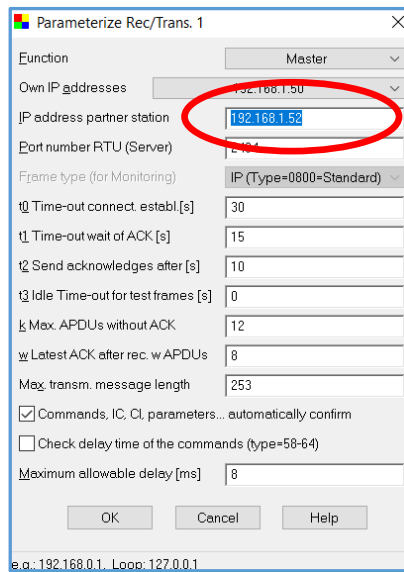


Ilustración 5.10 Ventana de configuración de parámetros

En la opción marcada se debe introducir la dirección IP de la estación a la que se quiere asociar dicha máquina. En la primera etiqueta se debe señalar la opción de “Master”. Con esto se consigue que la máquina principal pueda trabajar como servidor.

Los simuladores deben tener la siguiente configuración tanto en la máquina principal como en la virtual (*ilustración 5.11 y 5.12*).

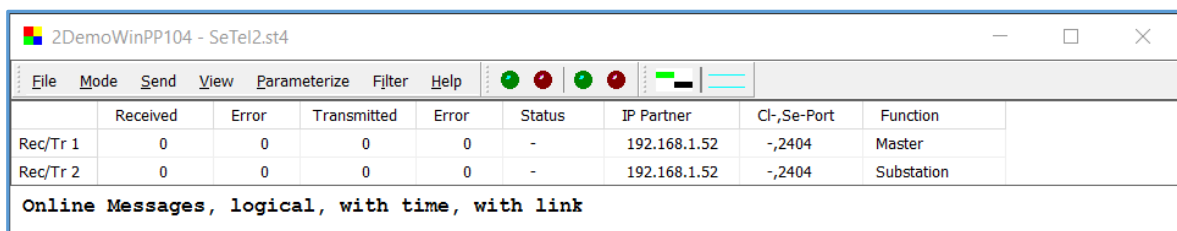


Ilustración 5.11 Máquina principal

	Received	Error	Transmitted	Error	Status	IP Partner	Cl-,Se-Port	Function
Rec/Tr 1	0	0	0	0	-	192.168.1.50	-,2404	Master
Rec/Tr 2	0	0	0	0	-	192.168.1.50	-,2404	Substation

Online Messages, logical, with time, with link

Ilustración 5.12 Máquina Virtual

Una vez se han configurado ambas máquinas, hay que ir a la pestaña **Mode** → **Online**, para que ambas máquinas se intenten comunicar una con otra. Cuando ambas máquinas se reconocen se debe visualizar esta ventana (*ilustración 5.13 y 5.14*):

	Received	Error	Transmitted	Error	Status	IP Partner	Cl-,Se-Port	Function
Rec/Tr 1	1	0	1	0	Connected	192.168.1.52	53209,2404	Master
Rec/Tr 2	1	0	1	0	Connected	192.168.1.52	49180,2404	Substation

Online Messages, logical, with time, with link

```

1      PI  21:05:22,267  Opened Master  t1=15 t2=10 t3=0  k=12 w=8  R1: 0/0  T1: 0/0
2      PI  21:05:22,268  Opened Substation  t1=15 t2=10 t3=0  k=13 w=8  R2: 0/0  T2: 0/0
3      PI  21:05:23,265  Substation: 192.168.1.50:2404  connected with: 192.168.1.52:49180
4      R2  21:05:23,906
IP:Port : 192.168.1.52:49180  >  192.168.1.50: 2404
Ctrl   : StartDT: act
5      T2  21:05:23,959  d=0,053s
IP:Port : 192.168.1.52:49180  <  192.168.1.50: 2404
Ctrl   : StartDT: con
6      PI  21:05:25,268  Master: 192.168.1.50:53209  connected with: 192.168.1.52:2404
7      T1  21:05:25,956  d=1,997s
IP:Port : 192.168.1.50:53209  >  192.168.1.52: 2404
Ctrl   : StartDT: act
8      R1  21:05:25,967  d=0,011s
IP:Port : 192.168.1.50:53209  <  192.168.1.52: 2404
Ctrl   : StartDT: con
    
```

Ilustración 5.13 Log Máquina Principal

	Received	Error	Transmitted	Error	Status	IP Partner	Cl-,Se-Port	Function
Rec/Tr 1	1	1	1	0	Connected	192.168.1.50	49180,2404	Master
Rec/Tr 2	1	0	1	0	Connected	192.168.1.50	53209,2404	Substation

Online Messages, logical, with time, with link

```

1      PI  21:05:23,241  Opened Master  t1=15 t2=10 t3=0  k=12 w=8  R1: 0/0  T1: 0/0
2      PI  21:05:23,241  192.168.1.52  Error at bind, WSA flag: #00002740
3      PI  21:05:23,242  Opened Substation  t1=15 t2=10 t3=0  k=13 w=8  R2: 0/0  T2: 0/0
4      PI  21:05:23,242  Master: 192.168.1.52:49180  connected with: 192.168.1.50:2404
5      T1  21:05:23,887
IP:Port : 192.168.1.52:49180  >  192.168.1.50: 2404
Ctrl   : StartDT: act
6      R1  21:05:23,935  d=0,048s
IP:Port : 192.168.1.52:49180  <  192.168.1.50: 2404
Ctrl   : StartDT: con
7      PI  21:05:25,255  Substation: 192.168.1.52:2404  connected with: 192.168.1.50:53209
8      R2  21:05:25,933  d=1,998s
IP:Port : 192.168.1.50:53209  >  192.168.1.52: 2404
Ctrl   : StartDT: act
9      T2  21:05:25,946  d=0,013s
IP:Port : 192.168.1.50:53209  <  192.168.1.52: 2404
Ctrl   : StartDT: con
    
```

Ilustración 5.14 Log Máquina Virtual

En ambas ventanas se puede comprobar que están conectadas entre sí y están preparadas para recibir y/o enviar tramas de información. Si se quiere enviar distintos mensajes hay que ir a la pestaña **Send** → **Messages** → “Elegir una opción”, (*ilustración 5.15*). Estas opciones están en alemán, pero no perjudica el objetivo que se busca en este proyecto.

Para el desarrollo de este proyecto se ha evaluado todas las opciones de la pestaña Send. Se hecho la simulación completa de todos los mensajes que puede generar este simulador, y se ha capturado con el WireShark dicha comunicación, como se explicado en el capítulo 4 sobre esta herramienta. En la *ilustración 5.16* se puede comprobar cómo ante el envío de distintos mensajes, el software va generando un archivo log, donde se va explicando los distintos envíos y recepciones.

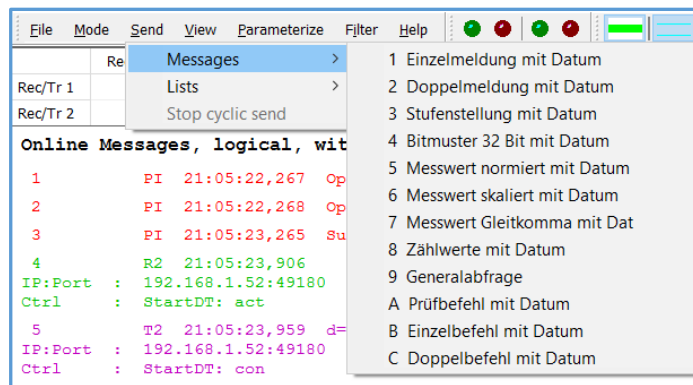


Ilustración 5.15 Selección de mensajes de envío entre máquinas

Ilustración 5.16 Ventana de Logs de envío de mensajes entre máquinas

CAPÍTULO 6
SOFTWARE DE
ANÁLISIS

CAPÍTULO 6: SOFTWARE DE ANÁLISIS

En los capítulos anteriores se ha explicado ampliamente, cada uno de los protocolos y cada una de las herramientas que se han usado para la adquisición de datos. En la *ilustración 6.1* se puede comprobar cómo ha sido el procedimiento para la adquisición de los paquetes de datos. En primer lugar tenemos los simuladores conectados a la red de datos donde van a generar los distintos paquetes de cada uno de los protocolos. Ambos simuladores van a funcionar de forma independiente dentro de la red de datos. El wireshark va a capturar todos los paquetes de ambos protocolos, y posteriormente generará un archivo **XML** como se ha explicado en el *capítulo 4*. Seguidamente el software de análisis desarrollado tratará el archivo volcado por el wireshark, y gestionará la información. Este programa, trabajará con la base de datos MySQL para guardar y extraer los distintos parámetros que se explicarán en este capítulo. Por último, se visualizarán los distintos resultados que genera el software de análisis.

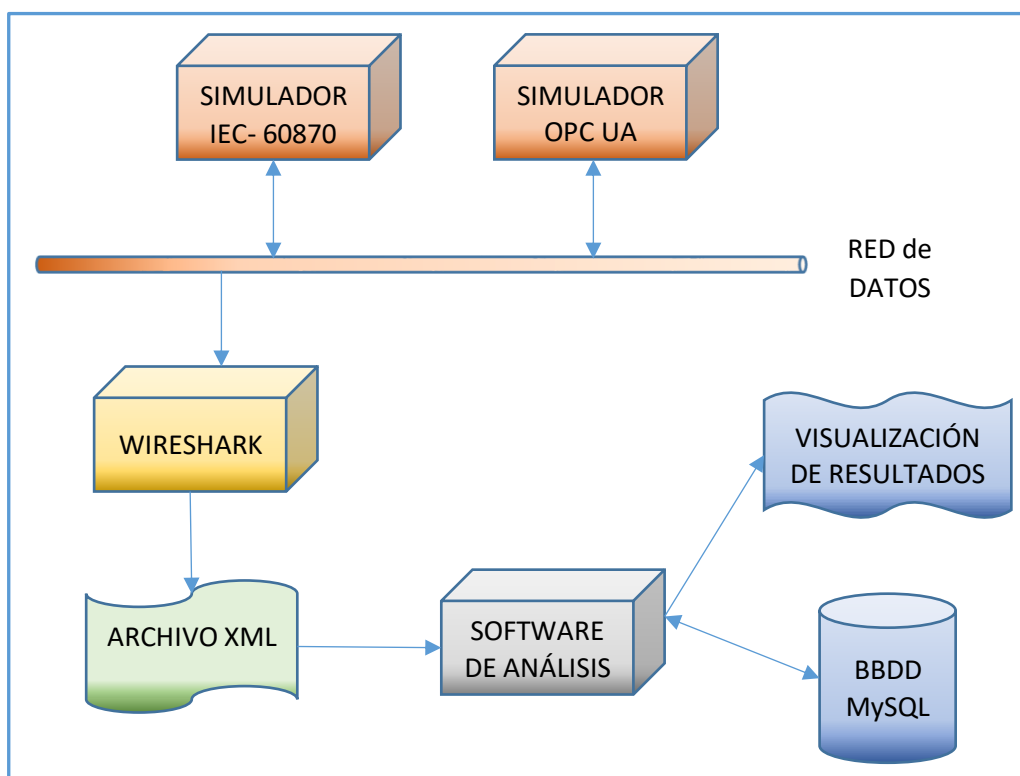


Ilustración 6.1 Entorno de pruebas

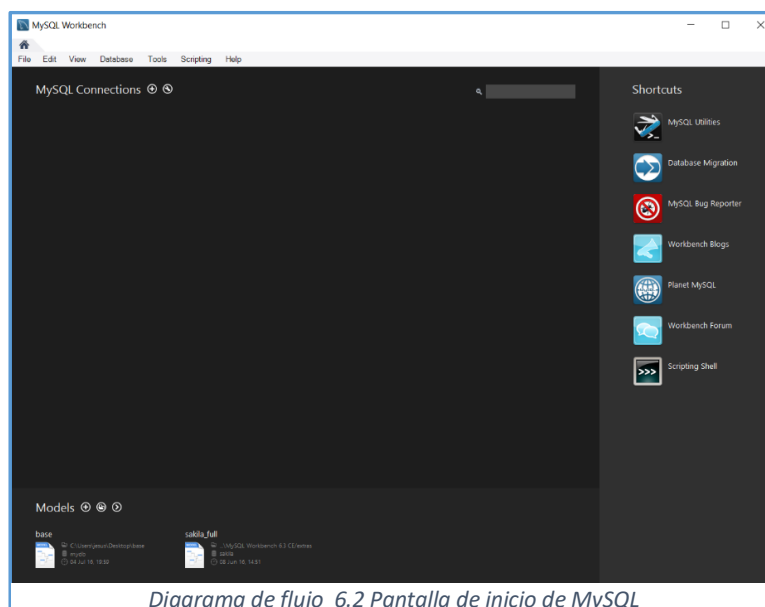
6.1.- JUSTIFICACIÓN DE MYSQL

En este proyecto el uso de una base de datos es muy importante, a la hora de almacenar los distintos parámetros extraídos de los paquetes a través de la aplicación WireShark. Se ha optado por el uso de esta base de datos, en primer lugar, porque es de carácter libre y el segundo lugar porque la gestión de la misma es bastante simple, ya que tiene múltiples herramientas para poder ser tratada.

6.2.- INSTALACIÓN DE MYSQL

Para instalar la base de datos en primer lugar hay que ir a la página [web](#) y descargarse estos tres paquetes.

- **MySQL Community Server:** Es el servidor de la base de datos de código abierto.
- **MySQL Workbench:** es una aplicación de diseño visual de base de datos de ésta, que se puede utilizar para diseñar y gestionar documentos de esquemas de base de datos de manera eficiente. Está disponible como código abierto y edición comercial.
- **MySQL Connectors:** Este es el controlador de conectividad de base de datos estándar, para el uso de MySQL con aplicaciones y herramientas que son compatibles con los estándares de la industria ODBC y JDBC.



Estos tres paquetes son necesarios para trabajar de una forma satisfactoria con una base de datos. Una vez terminada la instalación se ejecuta el software MySQL Workbench. En primer lugar se podrá visualizar la siguiente ventana (*ilustración 6.2*).

Seguidamente, hay que crear una conexión, donde estará la base de datos. Para ello hay que pulsar el botón que se muestra en la *ilustración 6.3*:

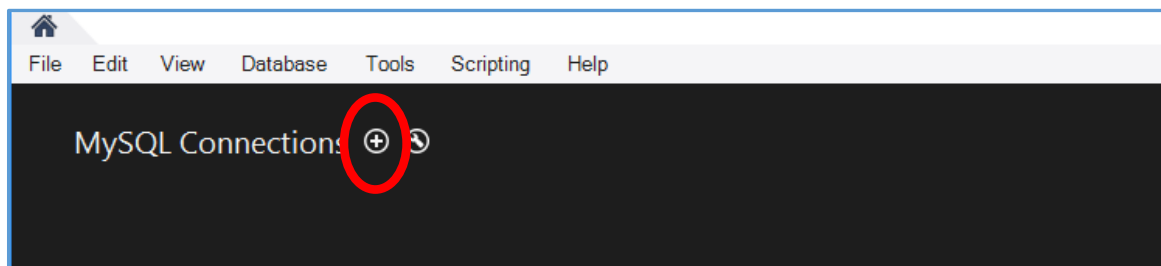


Ilustración 6.3 Ventana para crear una conexión nueva

Aparecerá la siguiente ventana y se le introducirá un nombre a la conexión, tal y como se indica en la *ilustración 6.4*. El resto de los parámetros se deben mantener tal y como están.

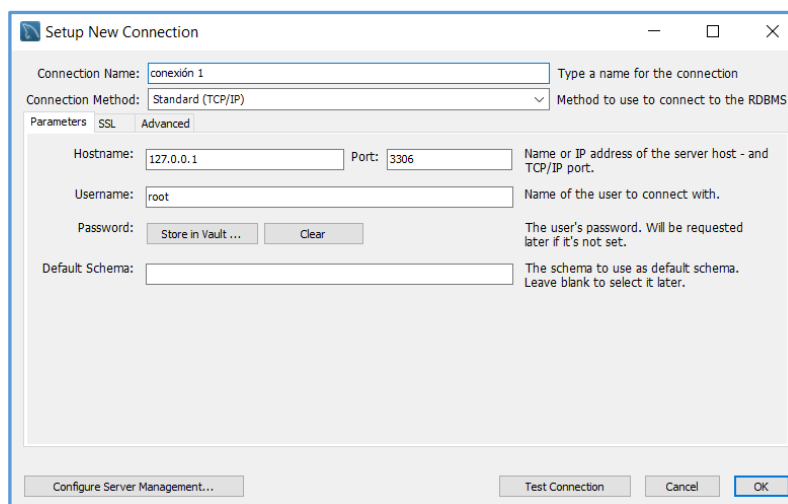


Ilustración 6.4 Ventana de configuración de la conexión

Se pulsa el botón **OK** y ya está la conexión creada. Se realiza un doble click sobre la conexión creada y entonces aparece la ventana que solicita la clave de *root* de MySQL, (*ilustración 6.4*) creada en la instalación. Una vez introducida dicha clave aparecerá la ventana siguiente (*ilustración 6.5*).

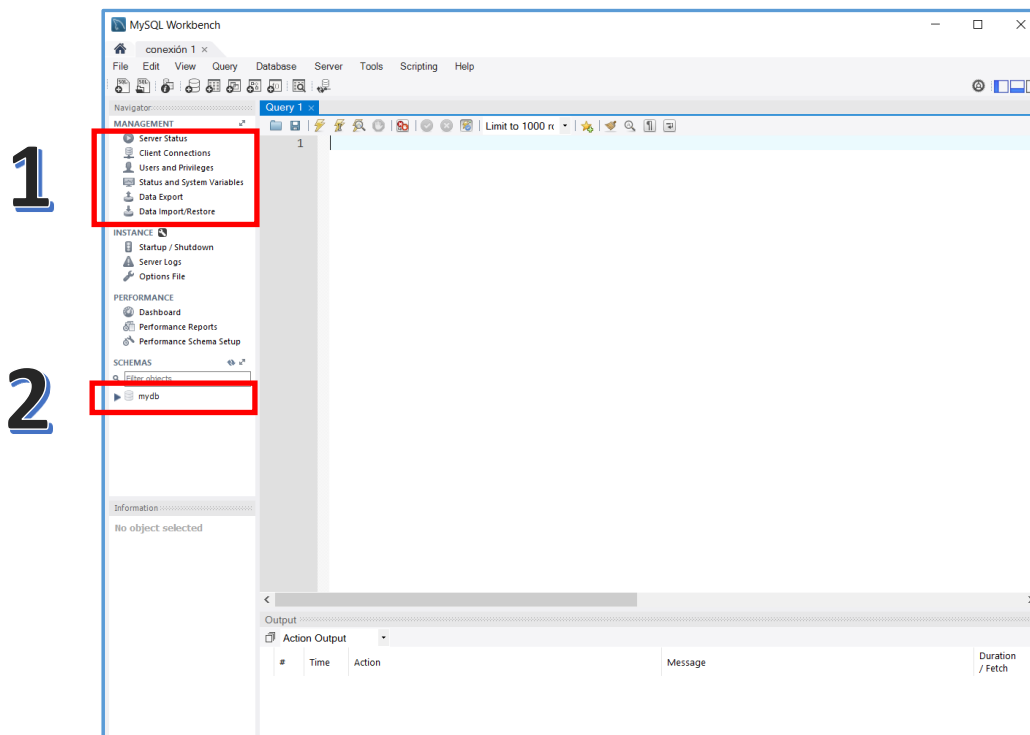


Ilustración 6.5 Ventana de gestión de MySQL

6.3.- CONFIGURACIÓN MySQL

En el **recuadro 1** de la *ilustración 6.5*, están los distintos botones de acceso rápido para la creación de bases de datos y de tablas. En el **recuadro 2** está la base de datos que el sistema crea por defecto y que se va a utilizar por parte de este programa. En primer lugar hay que pulsar con el botón derecho sobre el **recuadro 2** y seleccionar la primera opción “**Set as Default Schema**”, tal y como se indica en la *ilustración 6.6*.

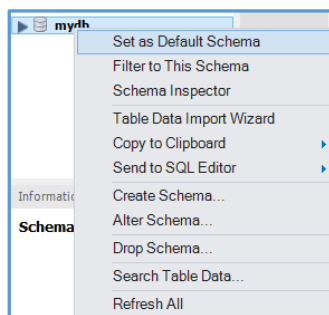


Ilustración 6.6 Selección por defecto la base de datos creada

Una vez se tiene seleccionada la base de datos hay que crear las distintas tablas que contendrán la base de datos. Para ello se pulsa la opción “**Data Import/Restore**” que se encuentra en el **recuadro 1**. Aparecerá la ventana que aparece en la siguiente figura

(ilustración 6.7), y donde se debe seleccionar la siguiente opción “**Import form Self-Contained file**”.

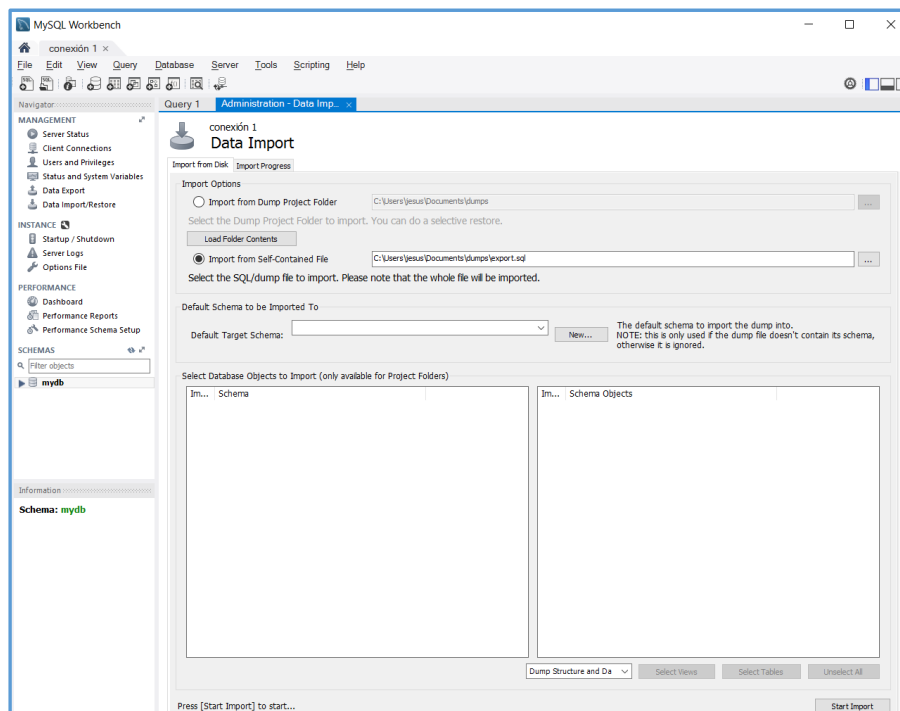


Ilustración 6.7 Importación de Datos del fichero script

Una vez seleccionada la opción se carga el fichero de texto plano, que contiene la instrucción de creación de tabla junto con los campos que se van a crear. Al mismo tiempo hay que seleccionar en la opción “**Default Target Schema**” la base de datos a la que se quiere añadir las **2 tablas**.

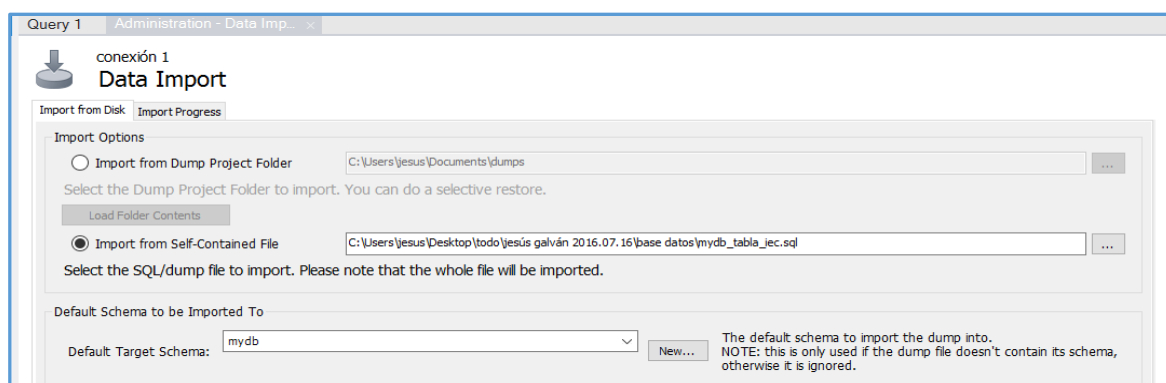


Ilustración 6.8 Selección de parámetros de importación de la tabla de datos

El contenido de ambos ficheros se muestran a continuación. Primero el contenido del fichero que crea la tabla para el protocolo IEC y después el contenido que crea la tabla para el OPC UA. Una vez cargado cada uno de los archivos se pulsa el botón “**Start Import**”. Entonces ya se puede comprobar en la base de datos que están ambas tablas incluidas (ilustración 6.9).

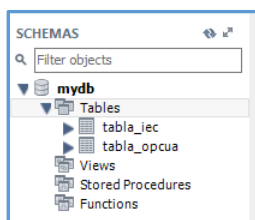


Ilustración 6.9 Tablas de la BBDD

Fichero “mydb_tabla_iec.sql”

```
DROP TABLE IF EXISTS `tabla_iec`;
CREATE TABLE `tabla_iec` (
  `paquete` int(11) NOT NULL AUTO_INCREMENT,
  `timestamp` varchar(45) DEFAULT NULL,
  `frame_time_delta` varchar(45) DEFAULT NULL,
  `eth_dst` varchar(45) DEFAULT NULL,
  `eth_src` varchar(45) DEFAULT NULL,
  `ip_dst` varchar(45) DEFAULT NULL,
  `ip_src` varchar(45) DEFAULT NULL,
  `frame` varchar(45) DEFAULT NULL,
  `ack_de_frame` varchar(45) DEFAULT NULL,
  `delta` double DEFAULT NULL,
  `time_ms` varchar(45) DEFAULT NULL,
  `time_min` varchar(45) DEFAULT NULL,
  `time_hour` varchar(45) DEFAULT NULL,
  `time_day` varchar(45) DEFAULT NULL,
  `time_month` varchar(45) DEFAULT NULL,
  `time_year` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`paquete`)
) ENGINE=InnoDB AUTO_INCREMENT=181 DEFAULT CHARSET=utf8;
```

Fichero “mydb_tabla_iec.sql”

```
DROP TABLE IF EXISTS `tabla_opcua`;
CREATE TABLE `tabla_opcua` (
  `paquete` int(11) NOT NULL AUTO_INCREMENT,
  `timestamp` varchar(45) DEFAULT NULL,
  `frame_time_delta` varchar(45) DEFAULT NULL,
  `eth_dst` varchar(45) DEFAULT NULL,
  `eth_src` varchar(45) DEFAULT NULL,
  `ip_dst` varchar(45) DEFAULT NULL,
  `ip_src` varchar(45) DEFAULT NULL,
  `transport_endpoint` varchar(45) DEFAULT NULL,
  `opcua_AuditEntryId` varchar(45) DEFAULT NULL,
  `opcua_TimeoutHint` varchar(45) DEFAULT NULL,
  `opcua_RequestedLifetime` varchar(45) DEFAULT NULL,
  `opcua_RequestHandle` varchar(45) DEFAULT NULL,
```

```

`opcua_ReturnDiagnostics` varchar(45) DEFAULT NULL,
`frame` varchar(45) DEFAULT NULL,
`ack_de_frame` varchar(45) DEFAULT NULL,
`delta` double DEFAULT NULL,
`cadena` varchar(100) DEFAULT NULL,
PRIMARY KEY (`paquete`)
) ENGINE=InnoDB AUTO_INCREMENT=705 DEFAULT CHARSET=utf8;

```

Una vez creada la base de datos junto con las tablas, manteniendo abierto el **MySQL Workbench**, el servidor de la base de datos está en marcha.

6.3.1.- CAMPOS DE LA TABLA OPC UA

En la siguiente tabla se muestran los campos de la “**tabla_opcua**”:

CAMPO	TIPO	SIGNIFICADO
paquete	INT	Número de paquete
timestamp	VARCHAR	Tiempo de captura del paquete (tiempo proporcionado por la propia tarjeta) en microsegundos
frame_time_delta	VARCHAR	Tiempo de respuesta (confirmación ACK WireShark)
eth_dst	VARCHAR	Dirección física del origen
eth_src	VARCHAR	Dirección física de destino
ip_src	VARCHAR	Dirección lógica de origen
ip_dst	VARCHAR	Dirección lógica de destino
transport_endpoint	VARCHAR	Dirección final del paquete (puede ser un enlace por ejemplo opc.tcp://localhost:48010)
frame	VARCHAR	Número de paquete proporcionado por la base de datos
ack_de_frame	VARCHAR	Número de trama confirmada en este paquete
delta	DOUBLE	Tiempo de respuesta (confirmación ACK). Tiempo calculado por el software programado, dando una precisión de nanosegundos
opcua_AuditEntryId	VARCHAR	Identifica el cliente o el usuario que ha iniciado la acción; utilizado en caso de auditoría
opcua_TimeoutHint	VARCHAR	Tiempo de espera para la llamada de servicio de cliente en 'UA Pila; cuando

		termina, el servidor cancela la llamada entrante
opcua_RequestedLifetime	VARCHAR	Indica cuánto es el intervalo sin que transcurra ninguna actividad supervisada por el cliente. Pasado este tiempo, el servidor suprime los recursos de suscripción y se libera de la misma. Este parámetro debe ser superior al menos 3 veces más que los posibles retardos que pudieran surgir. Es negociado por el cliente y el servidor
opcua_RequestHandle	VARCHAR	Identificador asignado a la llamada de servicio, que se define por el cliente
opcua_ReturnDiagnostics	VARCHAR	Indica si el cliente solicita el servidor para proporcionar, en caso de error, datos de diagnóstico en lugar de sólo un código de estado simple

Tabla 6.1 Campos de la tabla OPC-UA

6.3.2.- CAMPOS DE LA TABLA IEC – 60870

En la siguiente tabla se muestran los campos de la “**tabla_iec**”:

CAMPO	TIPO	SIGNIFICADO
paquete	INT	Número de paquete
timestamp	VARCHAR	Tiempo de captura del paquete (tiempo proporcionado por la propia tarjeta) en microsegundos
frame_time_delta	VARCHAR	Tiempo de respuesta (confirmación ACK WireShark)
eth_dst	VARCHAR	Dirección física del origen
eth_src	VARCHAR	Dirección física de destino
ip_src	VARCHAR	Dirección lógica de origen
ip_dst	VARCHAR	Dirección lógica de destino
transport_endpoint	VARCHAR	Dirección final del paquete (puede ser un enlace por ejemplo opc.tcp://localhost:48010)
frame	VARCHAR	Número de paquete proporcionado por la base de datos
ack_de_frame	VARCHAR	Número de trama confirmada en este paquete
delta	DOUBLE	Tiempo de respuesta (confirmación ACK). Tiempo calculado por el software programado, dando una precisión de nanosegundos

time_ms	VARCHAR	Campo CP56Time2a del estándar IEC-60870 tiempo binario (segundos). Se utiliza para la sincronización del reloj.
time_min	VARCHAR	Campo CP56Time2a del estándar IEC-60870 tiempo binario (minutos). Se utiliza para la sincronización del reloj.
time_hour	VARCHAR	Campo CP56Time2a del estándar IEC-60870 tiempo binario (horas). Se utiliza para la sincronización del reloj.
time_day	VARCHAR	Campo CP56Time2a del estándar IEC-60870 tiempo binario (días). Se utiliza para la sincronización del reloj.
time_month	VARCHAR	Campo CP56Time2a del estándar IEC-60870 tiempo binario (mes). Se utiliza para la sincronización del reloj.
time_year	VARCHAR	Campo CP56Time2a del estándar IEC-60870 tiempo binario (mes). Se utiliza para la sincronización del reloj.

Tabla 6.2 Campos de la tabla IEC

6.4.- SOFTWARE

6.4.1.- MENÚ ARCHIVO

La ventana principal del software tiene el siguiente aspecto (*ilustración 6.10*). Donde aparece un menú desplegable en la parte superior izquierda con tres opciones.

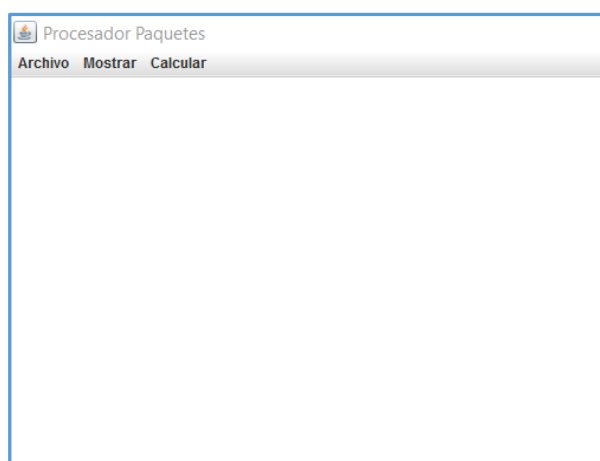


Ilustración 6.10 Ventana principal del programa

Cada uno de los tres menús desplegables tienen diferentes opciones. El menú **Archivo** tiene dos opciones explicados en la *tabla 6.3*:

MENU DESPLEGABLE	PROCESO QUE REALIZA
Abrir OPCUA	Una vez seleccionado sale un cuadro de diálogo para que seleccionar un archivo de Wireshark con la captura OPCUA
Abrir IEC	Una vez seleccionado sale un cuadro de diálogo para que seleccionar un archivo de Wireshark con la captura IEC-60870

Tabla 6.3 Menú desplegable Archivo

Una vez se haya seleccionado el archivo, el programa mostrará por pantalla, los distintos parámetros de cada uno de los protocolos. Los datos se mostrarán cómo aparece en la *ilustración 6.12*. Estos datos están extraídos directamente del fichero, pero no están guardados en la base de datos. Para poder ser incluido en la misma, hay que pulsar dentro de la ventana de datos el menú Archivo, como se indica en la siguiente ilustración.

timestamp	frame offset(ms)	eth_dst	eth_src	ip_src	ip_dst	transport_endpoint	opcu_audiEntryId	opcu_TimeoutId	opcu_Requested	opcu_RequestId	opcu_ReturnCsp	frame	ack_de_frame	cadena	delta (ms)
1485063727.8561	0	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	null	null	null	null	null	1	null	null	-1.0
1485063727.8562	0.024	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	null	null	null	null	null	2	1	null	0.024002794892
1485063727.8562	0.823	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	null	null	null	null	null	3	2	null	0.0238818359375
1485063727.8566	0.41800000000000	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	opc-udp.hubermu.de	null	null	null	null	null	4	null	null	1.0
1485063727.8566	0.30000000000000	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	null	null	null	null	null	5	4	null	0.00040700967
1485063727.8569	0.22999999999999	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	null	null	null	null	null	6	null	null	-1.0
1485063727.8569	0.202	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	null	null	null	null	null	7	6	null	0.0284412094384
1485063727.8572	0.358	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	8	null	null	-1.0
1485063727.8574	0.202	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	null	null	null	0	0	9	8	null	0.018400544992
1485063727.8585	1.649	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	10	9	null	1.049041748045875
1485063727.8588	0.12000000000000	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	11	10	null	0.1199245482880
1485063727.8591	0.517	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	12	null	null	-1.0
1485063727.8594	0.238	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	13	12	null	0.2390343833105
1485063727.8613	1.898	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	14	null	null	-1.0
1485063727.8681	8.538	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	15	14	String	6.83883267211914
1485063727.8688	0.733	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	16	15	null	0.7328987121582
1485063727.8690	0.211	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	17	16	String	0.2110004205048
1485063727.8699	0.833	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	18	17	null	0.833045153808
1485063727.8701	0.177	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	19	18	String	0.176895858933
1485063727.8676	17.55499999999999	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	20	19	null	17.555148841308
1485063712.8548	4847.015	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	21	null	null	-1.0
1485063712.8551	0.48700000000000	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	22	21	String	0.4885071882633
1485063732.8551	0.648	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	23	22	null	0.6481605529795
1485063733.54853	170.15800000000001	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	24	23	String	AirCondition
1485063733.54856	0.348	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	25	24	String	0.3478237099091
1485063733.54857	0.641	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	26	25	String	0.6415079956054
1485063733.54867	1.669	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	27	26	null	1.0
1485063733.54870	0.283	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	28	27	null	0.283028533835
1485063733.54881	1.108	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	29	28	String	AirCondition
1485063733.54885	0.37	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	30	29	null	0.3700263476626
1485063735.9668	2981.208	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	31	30	String	BuildingState
1485063735.9662	0.43000000000000	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	32	31	String	AirCondition
1485063735.9963	0.805	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	34	33	String	0.805675811787
1485063737.8845	1888.187	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	35	34	null	0.0548362719333
1485063737.8853	0.71999999999999	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	36	35	null	-1.0
1485063737.8853	0.81600000000000	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	37	36	null	0.8160265187753
1485063740.6114	2715.143	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	38	37	null	0.0448229528719
1485063740.6117	0.277	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	39	38	null	-1.0
1485063740.6118	0.602	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	40	39	String	0.2779423889160
1485063740.6128	0.98	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	41	40	null	0.91767020441
1485063740.6129	0.126	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	42	41	null	-1.0
1485063740.6129	0.805	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	43	42	String	0.1261234283447
1485063740.6130	0.81600000000000	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	44	43	String	0.8160362719333
1485063740.6130	0.81600000000000	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	null	AudiEntryId	00000000	16000000	0	0	44	43	String	0.816012054443

Ilustración 6.12 Datos capturados de un archivo

Una vez se haya pulsado **Guardar**, nos aparecerá un cuadro de diálogo en el cual nos muestra algunos campos ya completados (*ilustración 6.13*), que por lo general van a coincidir con el servidor de la base de los datos. En este caso solamente habría que introducir la contraseña de *root* del servidor de datos.

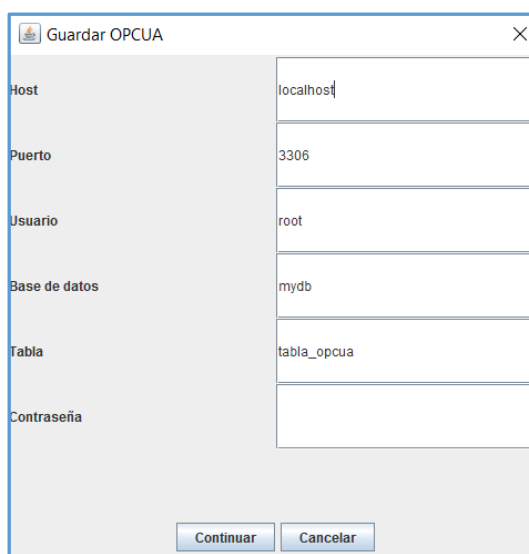


Ilustración 6.13 Ventana de parámetros para guardar el archivo en la base de datos

En la *tabla 6.3*, se indica cada uno de los parámetros:

CAMPO	SIGNIFICADO
Host	Dirección IP del servidor. En este caso la dirección es la propia máquina localhost o 127.0.0.1.
Puerto	Es el puerto que por defecto trabaja el servidor MySQL. En general utiliza el puerto 3306.
Usuario	Se establece cuando se instala y configura el servidor. Por defecto en este caso se usa el superusuario root.
Base de Datos	El nombre que se le haya dado a la base de datos. Por defecto y como se ha descrito anteriormente en este capítulo, se le ha nombrado como mydb.
Tabla	En este campo hay que introducir el nombre de la tabla. Concretamente va a aparecer por defecto el nombre de la tabla, según el archivo de datos que se haya elegido.
Contraseña	Es la contraseña de la base de datos. Este es el único campo que hay que introducir y aparece en blanco por defecto.

Tabla 6.3 Campos necesarios para el guardado en la base de datos

Una vez introducido todos los campos, hay que pulsar el botón de **Continuar** e inmediatamente se guarda en la base de datos todos los campos ya descritos anteriormente. Seguidamente saldrá una ventana de confirmación, *ilustración 6.14*, que nos pedirá que se **Acepte** la confirmación.

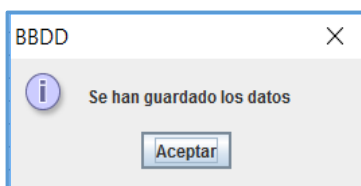


Ilustración 6.14 Ventana de confirmación de guardado de datos

Una vez se haya realizado esta operación, hay que **cerrar la ventana de datos** para poder seguir trabajando con el programa.

6.4.2.- MENÚ MOSTRAR

Seguidamente está el menú desplegable Mostar. Este menú tiene 4 opciones (ilustración 6.15). Estas opciones son explicadas en la *tabla 6.4*.

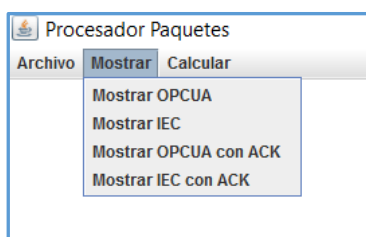


Ilustración 6.15 Menú desplegable Mostar

MENÚ DESPLEGABLE	PROCESO QUE REALIZA
Mostrar OPCUA	Si se selecciona esta opción, se mostrará una ventana con todos los campos guardados en la base de datos sin procesar es decir, aparecen los paquetes que no están confirmados del protocolo OPCUA
Mostrar IEC	Si se selecciona esta opción, se mostrará una ventana con todos los campos guardados en la base de datos sin procesar es decir, aparecen los paquetes que no están confirmados del protocolo IEC - 60870
Mostrar OPCUA con ACK	En esta opción se mostrarán los datos de la base de datos procesados. Se muestran solamente los paquetes confirmados para el protocolo OPCUA
Mostrar IEC con ACK	En esta opción se mostrarán los datos de la base de datos procesados. Se muestran solamente los paquetes confirmados para el protocolo IEC - 60870

Tabla 6.4 Menú desplegable Mostar

En la *ilustración 6.16*, se muestra como se verían los datos sin la confirmación de paquetes. En azul está señalado un paquete que no tiene confirmación.

Capítulo 6: SOFTWARE DE ANÁLISIS

Ilustración 6.16 Mostrar datos sin ACK

En esta otra figura (ilustración 6.17) se muestra como el programa visualiza los datos con los paquetes confirmados. Se puede comprobar que en el campo **ack_de_frame**, no hay ningún registro null.

Ilustración 6.16 Mostrar datos con ACK

6.4.3.- MENÚ CALCULAR

Este menú desplegable (ilustración 6.18), es la parte más importante del programa. Aquí se realizan los cálculos estadísticos relativos a los tiempos de respuesta, máximos, mínimos, media y desviación típica. Los tiempos son generados en *milisegundos*. Cada una de las selecciones se explican en la tabla 6.5.

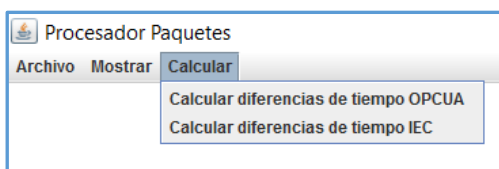


Ilustración 6.18 Menú desplegable Calculador

MENU DESPLEGABLE	PROCESO QUE REALIZA
Calcular diferencias de tiempo OPCUA	En esta opción se calculan las estadísticas relativas al protocolo OPCUA
Calcular diferencias de tiempo IEC	En esta opción se calculan las estadísticas relativas al protocolo IEC-60870

Tabla 6.5 Menú desplegable Calcular

En esta opción se calculan las estadísticas relativas al protocolo OPCUA.

Una vez se ha seleccionado alguna de las 2 opciones aparece este cuadro de diálogo. (Ilustración 6.19).

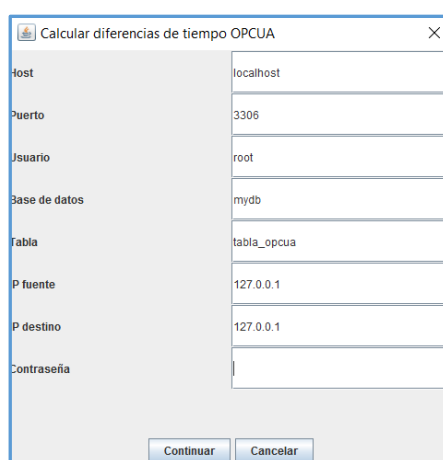


Ilustración 6.19 Ventana para introducir los parámetros de lectura de la base de datos

Las opciones que aparecen son las siguientes:

CAMPO	SIGNIFICADO
Host	Dirección IP del servidor. En este caso la dirección es la propia máquina localhost o 127.0.0.1.
Puerto	Es el puerto que por defecto trabaja el servidor MySQL. En general utiliza el puerto 3306.
Usuario	Se establece cuando se instala y configura el servidor. Por defecto en este caso se usa el superusuario root.
Base de Datos	El nombre que se le haya dado a la base de datos. Por defecto y como se ha descrito anteriormente en este capítulo, se le ha nombrado como mydb.
Tabla	En este campo hay que introducir el nombre de la tabla. Concretamente va a aparecer por defecto el nombre de la tabla, según el archivo de datos que se haya elegido.
IP fuente	Hay que introducir la dirección IP de la máquina que envía los paquetes. Viene por defecto la dirección de localhost 127.0.0.1. Ya se comentó en capítulo 4 sobre los simuladores utilizados, que el simulador OPCUA trabajaba en la misma máquina

IP destino	Hay que introducir la dirección IP de la máquina que recibe los paquetes. Viene por defecto la dirección de localhost 127.0.0.1. Ya se comentó en el capítulo 4 sobre los simuladores utilizados, que el simulador OPCUA trabajaba en la misma máquina.
Contraseña	Es la contraseña de la base de datos. Este es el único campo que hay que introducir y aparece en blanco por defecto.

Tabla 6.6 Parámetros necesarios para la lectura de la BBDD y su posterior visualización

Una vez introducidos estos parámetros, se obtendrá una gráfica con distintos datos estadísticos, como se muestra en la siguiente figura, (ilustración 6.20).

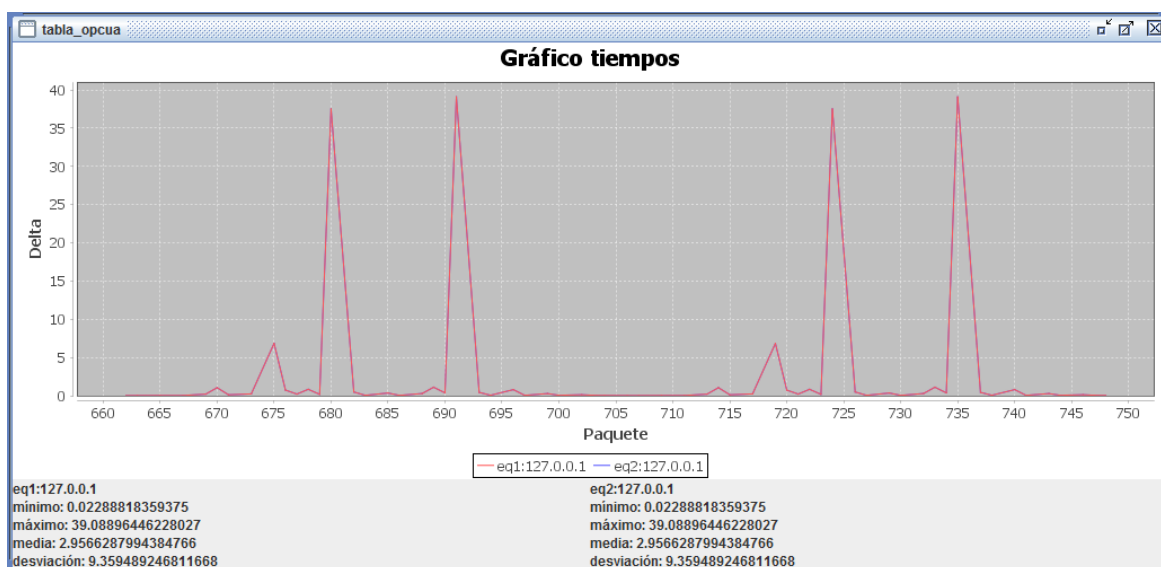


Ilustración 6.20 Gráfico de los resultados OPC-UA extraídos de la BBDD

Todos los valores establecidos anteriormente están en **milisegundos**. Ofreciendo los siguientes valores (tabla 6.7):

MENU DESPLEGABLE	PROCESO QUE REALIZA
mínimo	Tiempo de retardo mínimo entre todos los paquetes.
máximo	Tiempo de retardo máximo entre todos los paquetes.
media	Tiempo medio calculado con todos los paquetes.
desviación	Desviación típica. Fluctuación entre máximos y mínimos.

Tabla 6.7 Variables calculadas del archivo

6.5 EJEMPLOS DE CÁLCULO CON DIFERENTES CAPTURAS

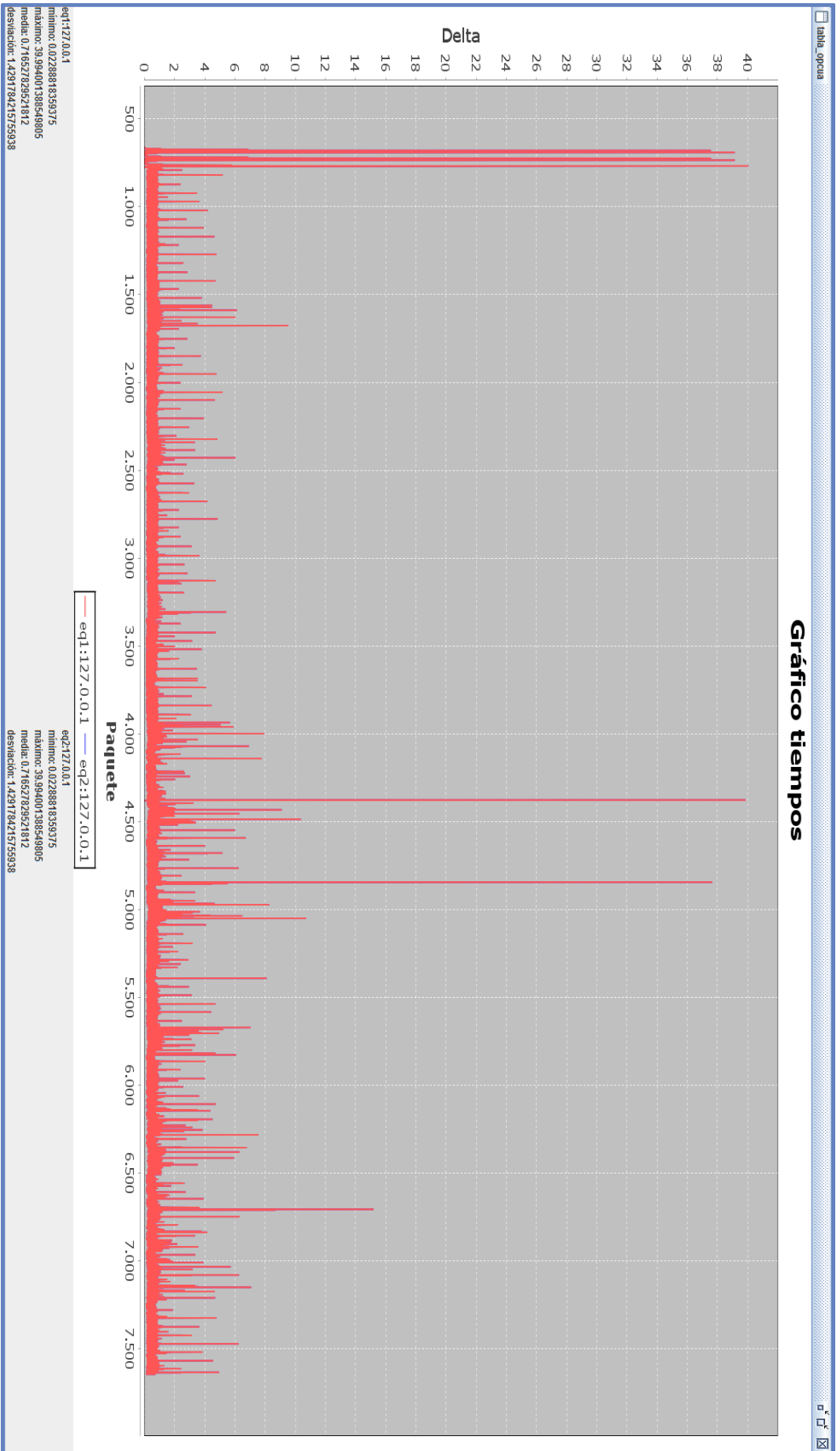


Ilustración 6.21 Cálculo de tiempos de alrededor de 7000 paquetes de OPC-UA

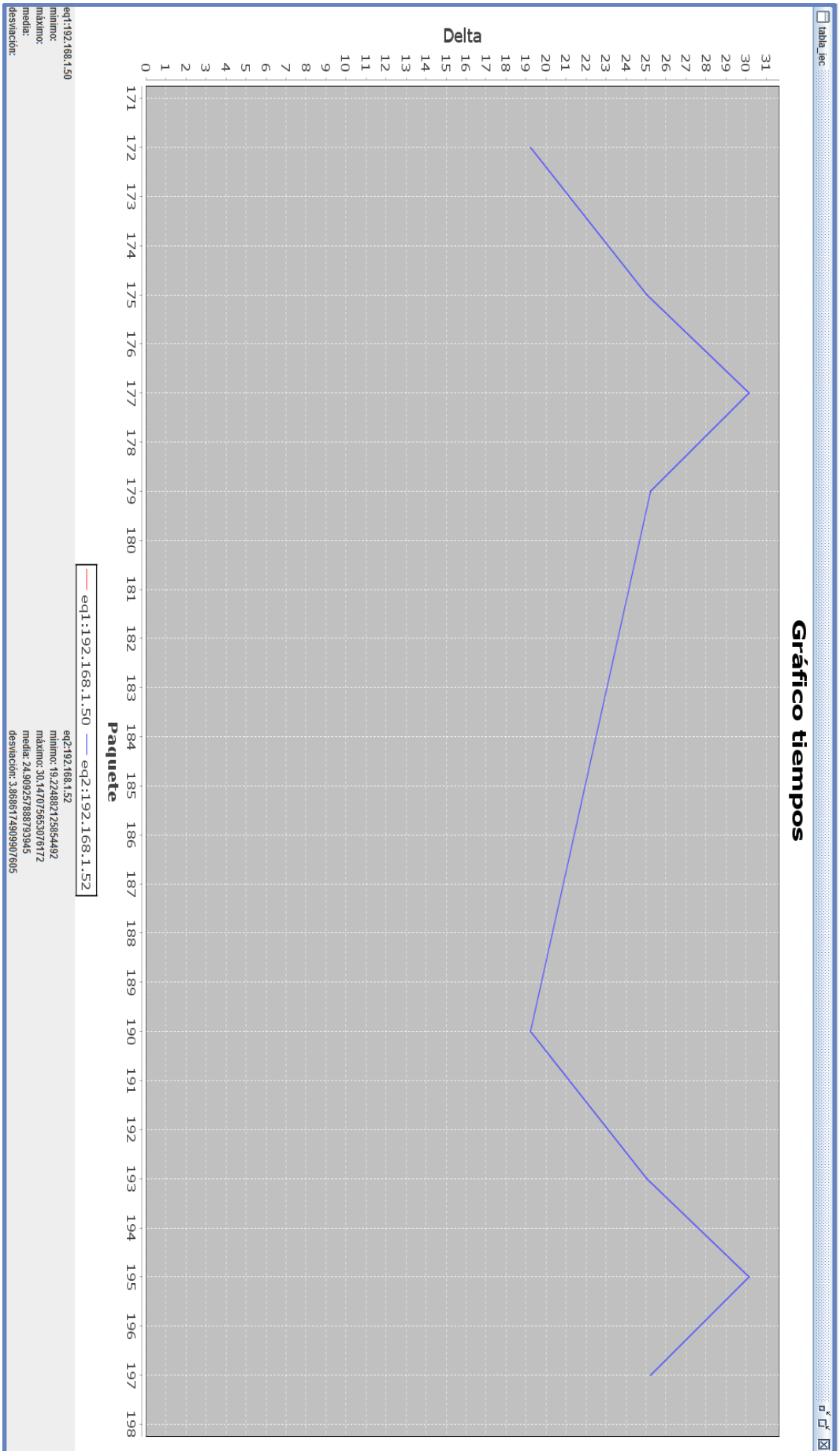


Ilustración 6.22 Cálculo de unos 20 paquetes de IEC-60870

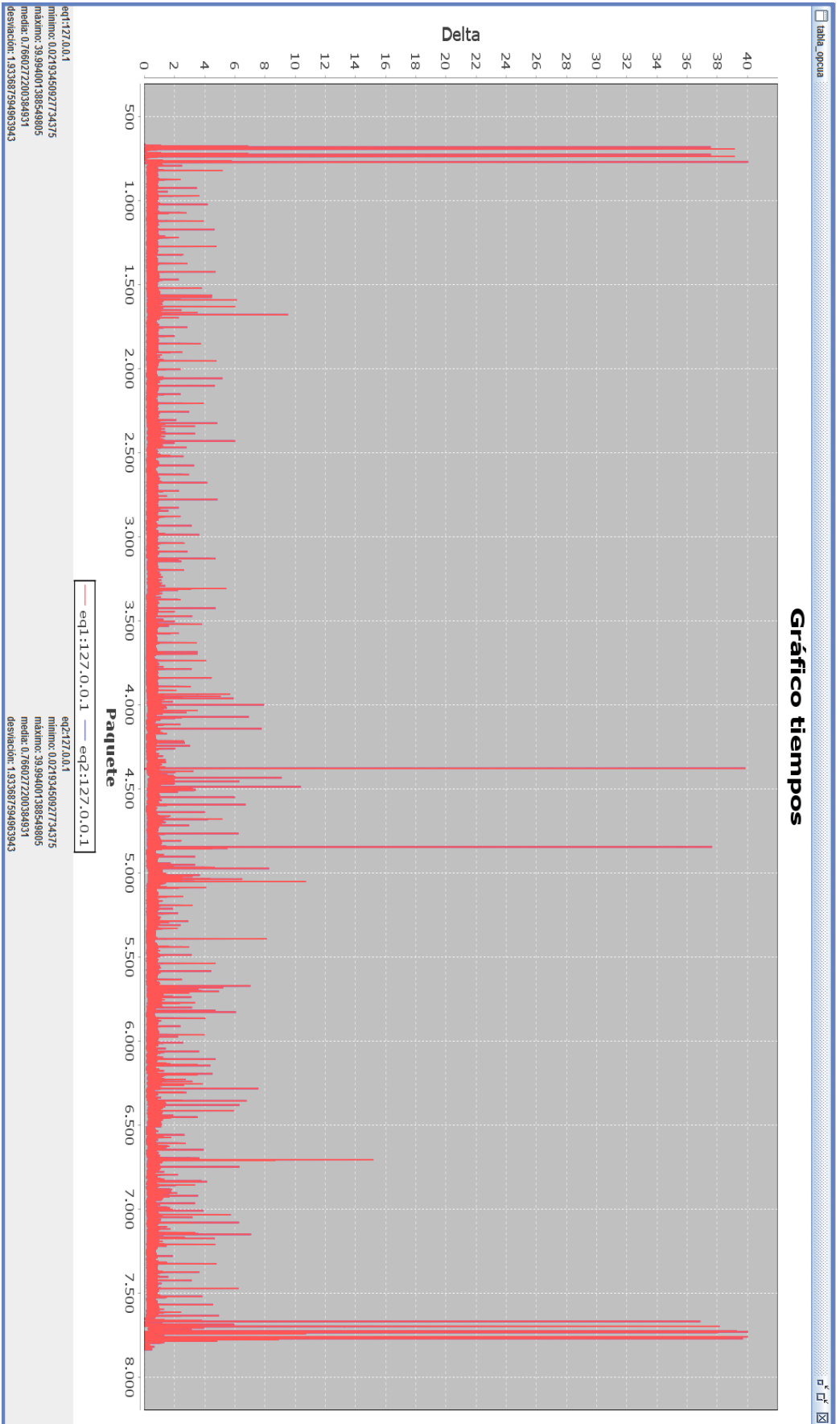


Ilustración 6.23 Cálculo de unos 6500 paquetes OPC-UA

CAPÍTULO 7

IMPLEMENTACIÓN

7.1.- INTRODUCCIÓN

Para la implementación de este proyecto se ha utilizado la herramienta Netbeans para la programación en Java. Este software es de carácter libre y lo distribuye Oracle. El uso de esta herramienta viene motivado por el manejo sencillo y la potencia tanto del Java como de Netbeans.

7.2.- ENTORNO

El interfaz de Netbeans tiene la misma similitud, en cuanto a su entorno de trabajo, que cualquier otra herramienta que trabaja en Windows. En la *ilustración 6.1*, se puede comprobar como es el entorno de trabajo, donde existe múltiple documentación para el estudio y manejo del programa.

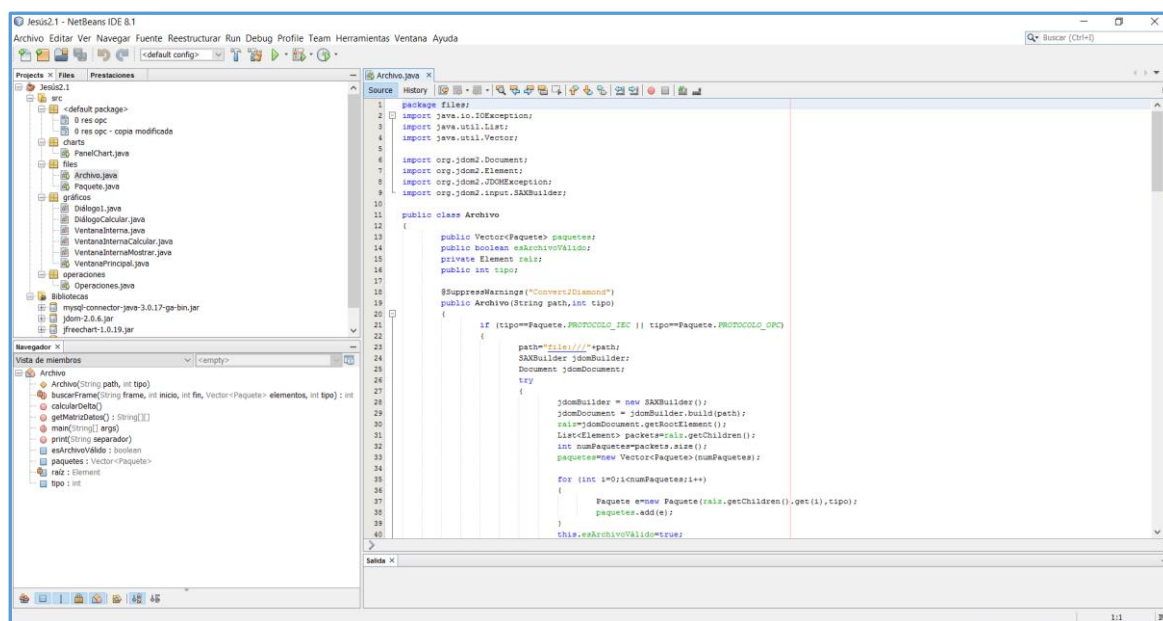


Ilustración 7.1 Ventana Principal del Netbeans

En el panel izquierdo se puede observar los distintos paquetes, bibliotecas y clases usadas para este proyecto. En los siguientes apartados se comentará el funcionamiento de cada una de las clases con sus diagramas de **flujo**.

7.3.- IMPLEMENTACIÓN

En la *ilustración 6.2*, se puede comprobar el árbol de dependencia de cada una de las clases y cada uno de los paquetes. En los siguientes apartados se describirán cada una de las clases con sus métodos. Cada método a su vez, tendrá un diagrama de flujo explicando su funcionamiento.

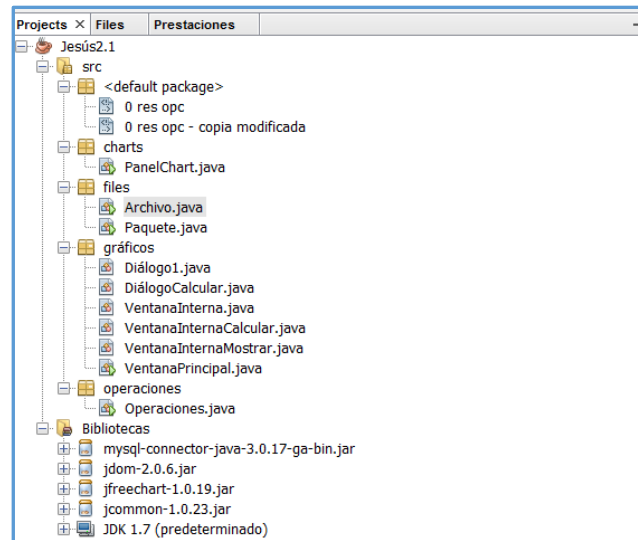


Ilustración 7.2 Árbol de Paquetes y clases del proyecto Netbeans

7.3.1.- PAQUETECHARTS

Clase charts.PanelChart

```
public static ChartPanel crearGráfico(int[] paquetesF1, double[] deltasF1, String nombreF1, int[] paquetesF2, double[] deltasF2, String nombreF2)
```

Descripción: crea un panel con un gráfico. Cada gráfico se compone de dos series. Las entradas son los números de paquetes (**índices**), datos (**deltas**) y el nombre de la serie.

```
(out) ChartPanel=crearGráfico (int[] númeroPaquetesSerie1, double[] deltasSerie1, String nombreSerie1, int[] númeroPaquetesSerie2, double[] deltasSerie2, String nombreSerie1)
{
    Comprobar nombres y tamaños de las series
    CrearGráfico
    DevolverGráfico
}
```

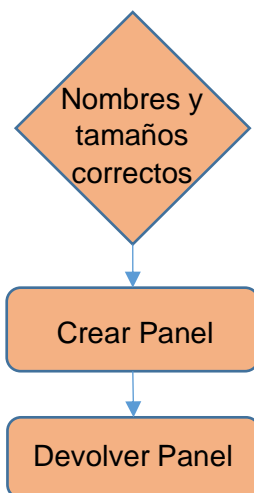


Diagrama de flujo 7.1

```
public static JPanel crearPanelDatos (String mínimo, String máximo, String media, String desviación, String ip)
```

Descripción: crea un panel con la información complementaria del gráfico (**máximo, mínimo, media, desviación y dirección IP**).

```
(out) JPanel=crearPanelDatos (String mínimo, String máximo, String media, String desviación, String ip)
{
    Crear panel
    Añadir información al panel
    Devolver panel
}
```

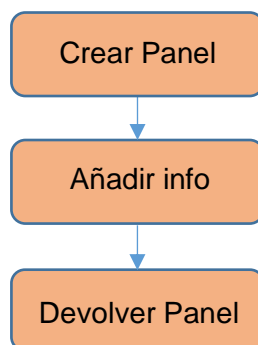


Diagrama de flujo 7.2

```
public static JPanel crearPanelGráfico (ChartPanel g, JPanel p, JPanel p2)
```

Descripción: crea un *ChartPanel* con el panel que contiene el gráfico (**g**) y con los *JPanel* que contiene la información adicional **p** y **p2**.

```
(out) JPanel=crearPanelDatos (ChartPanel g,JPanel p,JPanel p2)
{
    Crear JPanel
    Insertar panel gráfico y panel datos
    Devolver panel
}
```

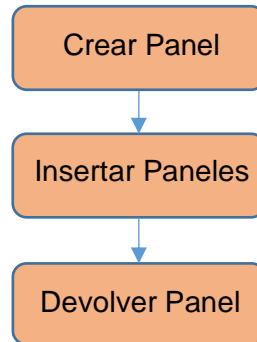


Diagrama de flujo 7.3

7.3.2.- PAQUETE FILES

Clase `files.Archivo`

```
public Vector<Paquete> paquetes;
public boolean esArchivoVálido;
private Element raíz;
public int tipo;

public Archivo (String path,int tipo)
```

Descripción: constructor de la clase. Crea un objeto de la clase `Archivo` leyendo el archivo indicado en el path y según el tipo seleccionado (**opc** o **iec**).

```
Archivo (String path,int tipo)
{
    comprobar tipo
    obtener nodo raíz
    obtener hijos
    llenar paquetes
    calcular deltas
}
```

```
public void print (String separador)
```

Descripción: imprime por pantalla el contenido del árbol usando como separador el indicado como parámetro de entrada.

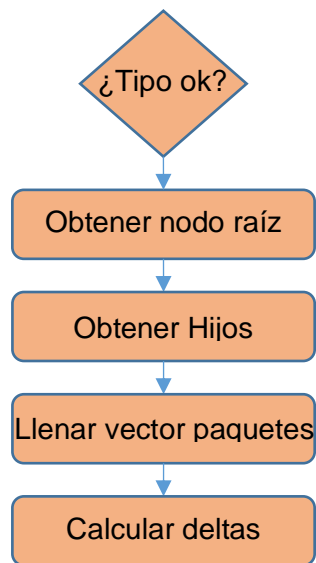
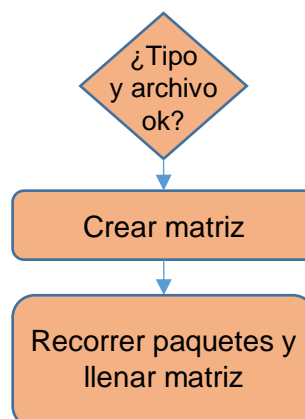


Diagrama de flujo 7.4

```
public String[][] getMatrizDatos ()
```

Descripción: llena una matriz con la información de los paquetes.

```
Archivo (String path, int tipo)
{
    comprobar tipo y archivo válido
    crear matriz
    recorrer paquetes y llenar matriz
}
```



```
public void calcularDelta ()
```

Descripción: calcula la diferencia de tiempo de los paquetes con **ACK**.

```
(out) void=CalcularDelta ()  
{  
  for i,i==número de paquetes  
    si paquete i tiene ack  
      buscar frame correspondiente  
      calcular y guardar delta  
}
```

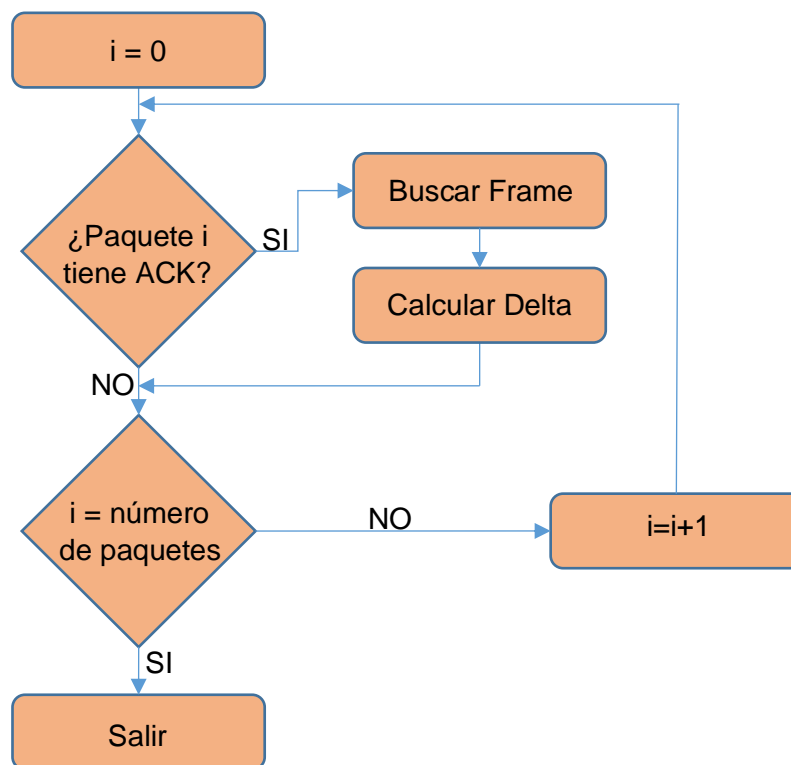


Diagrama de flujo 7.7

```
private static int buscarFrame (String frame,int inicio,int  
fin,Vector<Paquete> elementos,int tipo)
```

Descripción: busca un frame. Se usa en calcular delta para encontrar el frame al que hace referencia un **ACK**. Por ejemplo, si el frame 87 es **ACK** del frame 80, se busca el frame 80 para calcular la diferencia de tiempo entre el frame 80 y 87.

Capítulo: 7 IMPLEMENTACIÓN

Las variables de entrada son la cadena a *buscar (frame)*, los números de inicio y fin de la búsqueda (desde qué frame a qué frame se busca), los paquetes y el tipo (**opc** o **iec**).

```
(out) int=buscarFrame(String frame,int inicio,int fin,Vector<Paquete>
elementos,int tipo)
{
    for i=inicio,i==fin
        si paquete(i)==frame
            return i
}
```

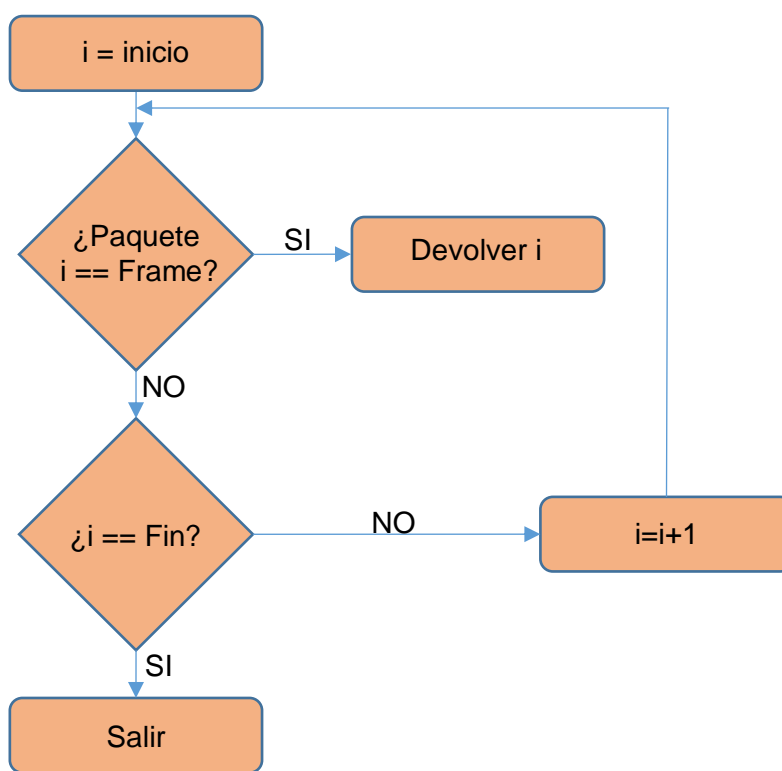


Diagrama de flujo 7.8

Clase files.Paquete

```
public Paquete(Element e,int tipo)
public void llenarInformación(Element e,int tipo)
private static String[] obtenerCampos(int tipo)
private static String[] obtenerAtributosBuscar(int tipo)
static String[] obtenerAtributosMostrar(int tipo)
public static String[] obtenerAtributosBaseDeDatos(int tipo)
public static void print(Element elemento)
public static String toString(Element elemento)
public static String getInformaciónPaquete(Element elemento,int tipo)
public static String getInformaciónPaquete(Element elemento,String
separador,int tipo)
```



```
public static String getInformaciónPaquete(Element elemento, String[]
atributos, String[] campos, String separador)
public static String getInformación(Element elemento, String nombreCampo,
String nombreAtributo)
public static String getInformación(Element elemento, String nombreCampo,
String[] nombreAtributos)
public static String[][] getMatrizDatos(Element raíz, int tipo)
public static String buscarInfo(Element elemento, String campo)
public static Element encontrarElemento(Element elemento, String valor)
public static boolean esPaquete(Element elemento)
public static boolean esProtocolo(Element elemento)
public static boolean contieneOPCUA(Element elemento)
public void print(String separador)

public Paquete(Element e, int tipo)
```

Descripción: constructor de la clase. Crea una instancia de tipo paquete a partir de un elemento del archivo *original (jdom2)* y según el tipo que sea (*opc* o *iec*).

```
Paquete(Element e, int tipo)
{
    Comprobar tipo
    Comprobar elemento válido
    Llenar información (llenarInformación)
}
```

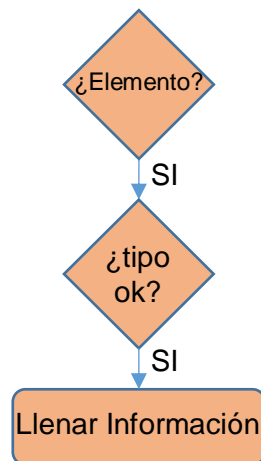


Diagrama de flujo 7.9

Capítulo: 7 IMPLEMENTACIÓN

```
public void llenarInformación(Element e,int tipo)
```

Descripción: Llena un vector con los elementos que se encuentran en el elemento **e**. Al inicializar los elementos, se crea un vector con los atributos y los campos que hay que buscar. Dichos elementos dependen del tipo.

```
(out) void=llenarInformación(Element e,int tipo)
{
  Inicializar elementos
  For i,elementos
    Atributos(i)=getInformación(e,campo(i),atributo(i))
}
```

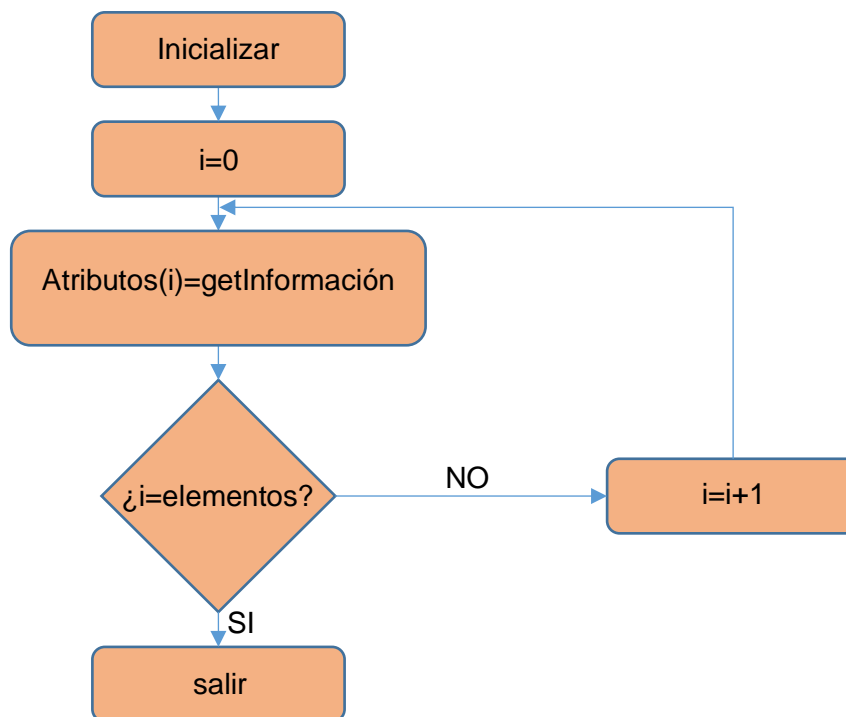


Diagrama de flujo 7.10

```
private static String[] obtenerCampos(int tipo)
```

Descripción: Método para obtener los campos que hay que buscar en el archivo original. Los campos dependen del tipo de archivo.

```
(out)String[]=obtenerCampos(int tipo)
{
  si tipo=opc
    return camposOpc
  si tipo=iec
    return camposIec
  else
    return vector vacío
}
```

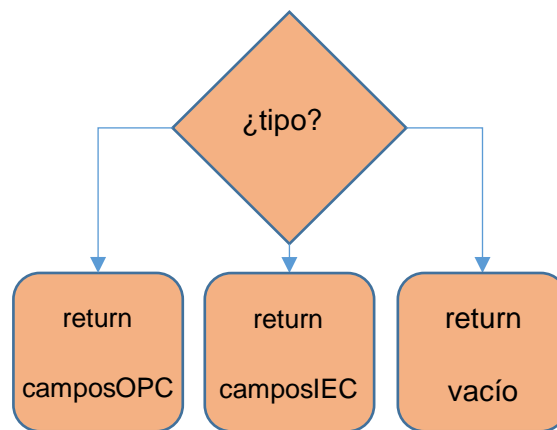


Diagrama de flujo 7.11

```
private static String[] obtenerAtributosBuscar(int tipo)
```

Descripción: Método para obtener los atributos que hay que buscar en el archivo original. Los atributos dependen del tipo de archivo.

```
(out)String[]=obtenerAtributosBuscar (int tipo)
{
    si tipo=opc
        return atributosOpc
    si tipo=iec
        return atributosIec
    else
        return vector vacío
}
```

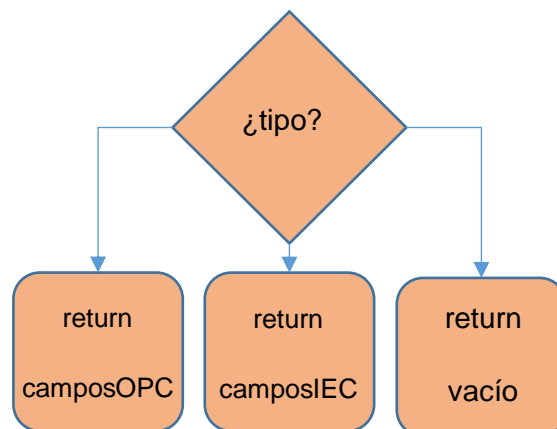


Diagrama de flujo 7.12

```
static String[] obtenerAtributosMostrar(int tipo)
```

Descripción: Método para obtener los atributos que se muestran en la **tabla**. Los atributos dependen del tipo de archivo.

```
(out)String[]=obtenerAtributosMostrar (int tipo)
{
    si tipo=opc
        return atributosOpc
    si tipo=iec
        return atributosIec
    else
        return vector vacío
}
```

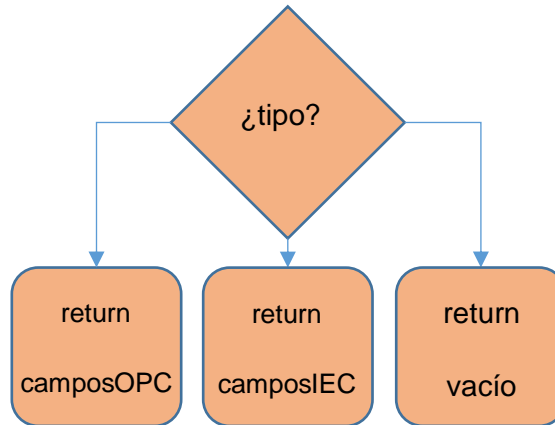


Diagrama de flujo 7.13

```
public static String[] obtenerAtributosBaseDeDatos (int tipo)
```

Descripción: Método para obtener los atributos que se usan en la base de datos.

Los atributos dependen del tipo de archivo.

```
(out)String[]=obtenerAtributosBaseDeDatos (int tipo)
{
    si tipo=opc
        return atributosOpc
    si tipo=iec
        return atributosIec
    else
        return vector vacío
}
```

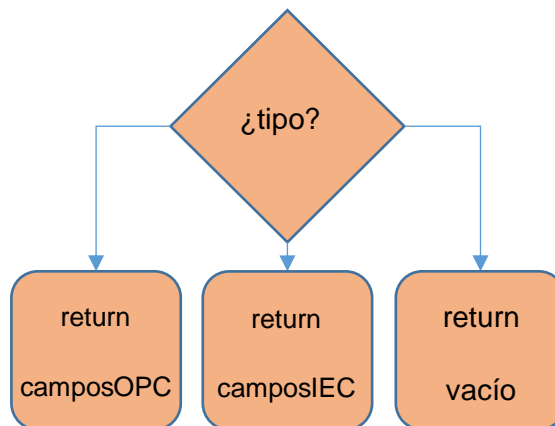


Diagrama de flujo 7.14

`public static void print`(Element elemento)

Descripción: Se imprime por pantalla el árbol entero a partir del elemento indicado.

```
(out) void=print(Element e)
{
    si elemento==null
        salir
    atributos=e.obtenerAtributos
    hijos=e.obtenerElementosHijos
    for i, atributos
        imprimir(atributos(i))
    for i, hijos
        elementoAux=e.get(i)
        elementoAux.print
}
```

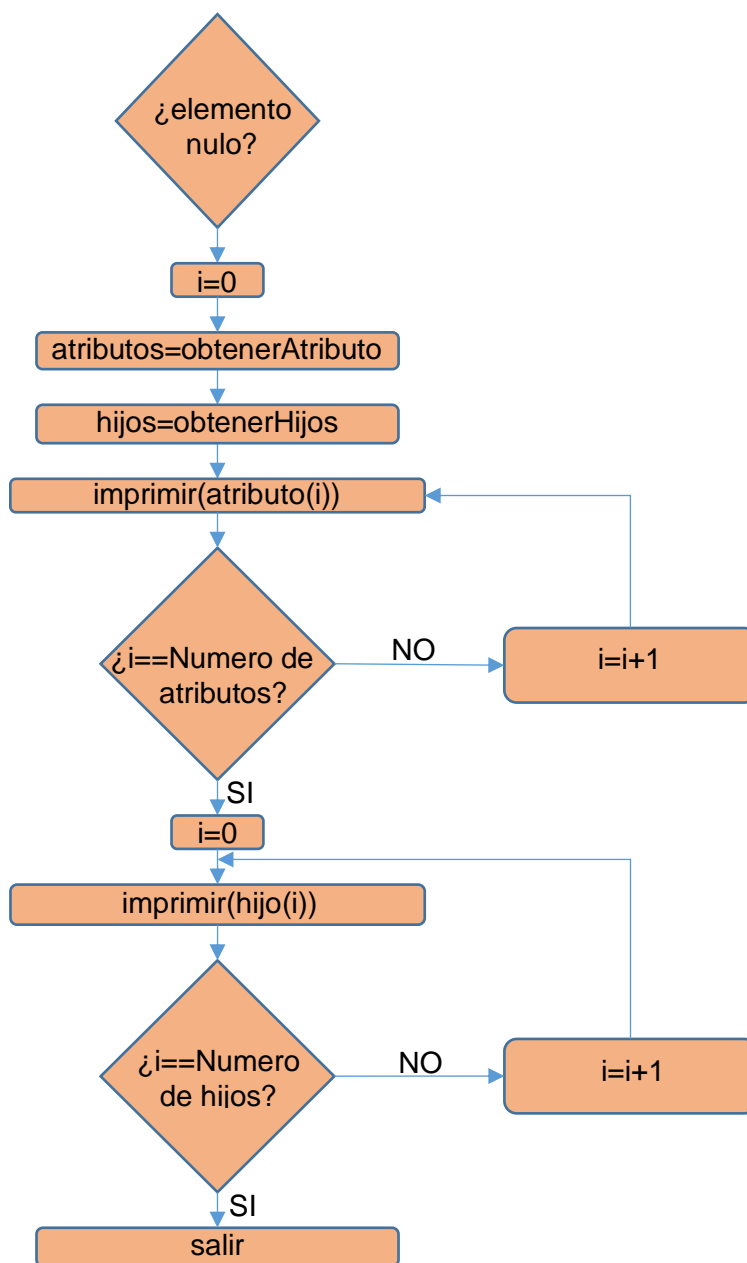


Diagrama de flujo 7.15

`public static String toString(Element elemento)`

Descripción: Crea una cadena con la información del **árbol** entero a partir del elemento indicado.

```

(out)String=toString(Element e)
{
  inicializarCadena
  si elemento==null
    salir
  atributos=e.obtenerAtributos
  hijos=e.obtenerElementosHijos
  for i,atributos
    cadena=cadena+(atributos(i))
  for i,hijos
    elementoAux=e.get(i)
    cadena=cadena+elementoAux.toString
}
  
```

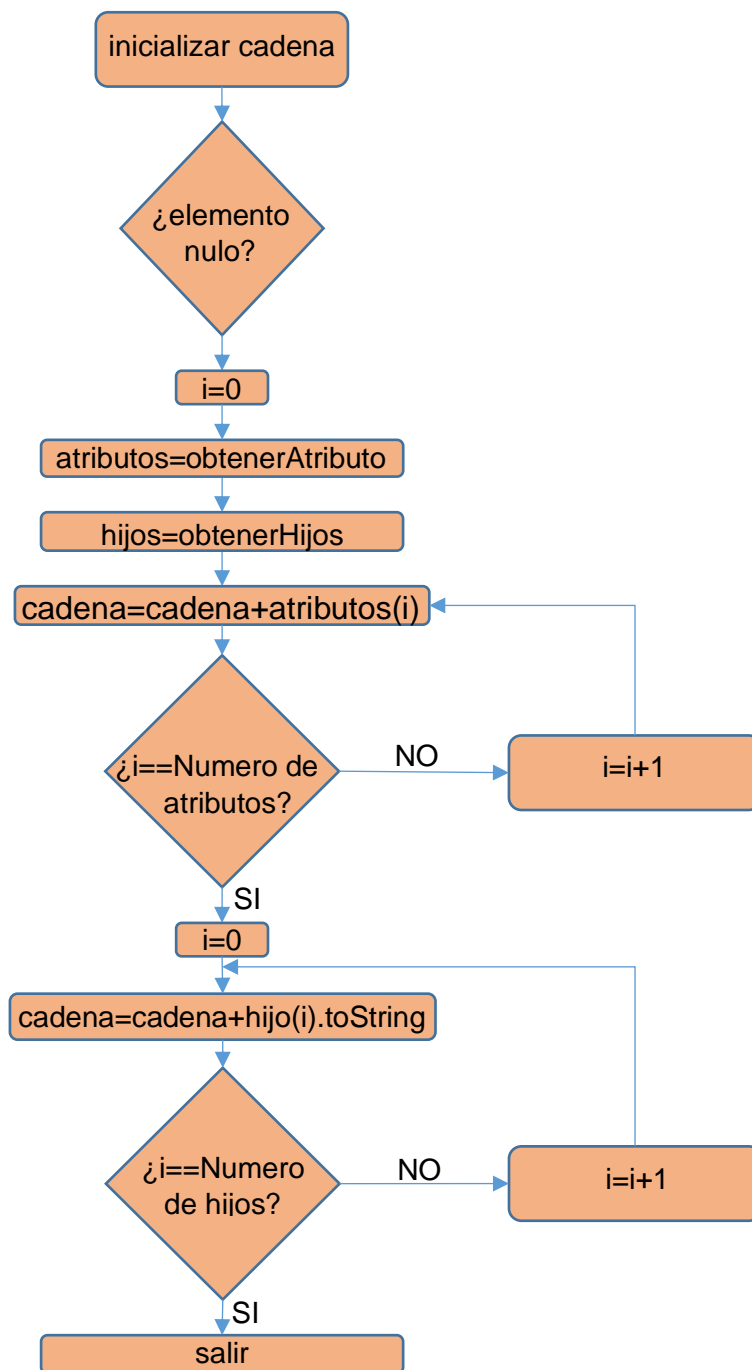


Diagrama de flujo 7.16

```

public static String getInformaciónPaquete (Element elemento, String
separador, int tipo)

```

Descripción: busca la información de un elemento y la devuelve en una cadena separada por el elemento separador. Llama a **getInformaciónPaquete** (método con el mismo nombre pero con los atributos y los campos como parámetros de entrada).

```
(out)String=getInformaciónPaquete(Element e)
{
    obtenerCtributos (tipo)
    obtenerCampos (tipo)
    llamar getInformaciónPaquete(atributos, campos)
    devolver cadena
}
```

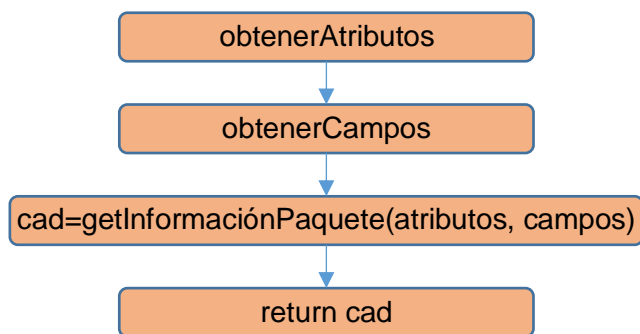


Diagrama de flujo 7.18

```
public static String getInformaciónPaquete(Element elemento,int tipo)
Descripción: igual que el anterior pero con el elemento separador automático.
```

```
public static String getInformaciónPaquete(Element elemento,String[]
atributos,String[]campos,String separador)
```

Descripción: Devuelve una cadena, separada con la cadena separador, con la información del elemento. En cada iteración del bucle se busca la información dada por el índice *i* y se concatena dicha información. La búsqueda se hace con el método **getInformación**.

```
(out)String= getInformaciónPaquete (Element elemento,String[]
atributos,String[]campos,String separador)
{
    comprobarElemento
    inicializarCadena
    for i,atributos
        cadena=cadena+(getInformación(elemento),atributos(i),campos(i))
}
```

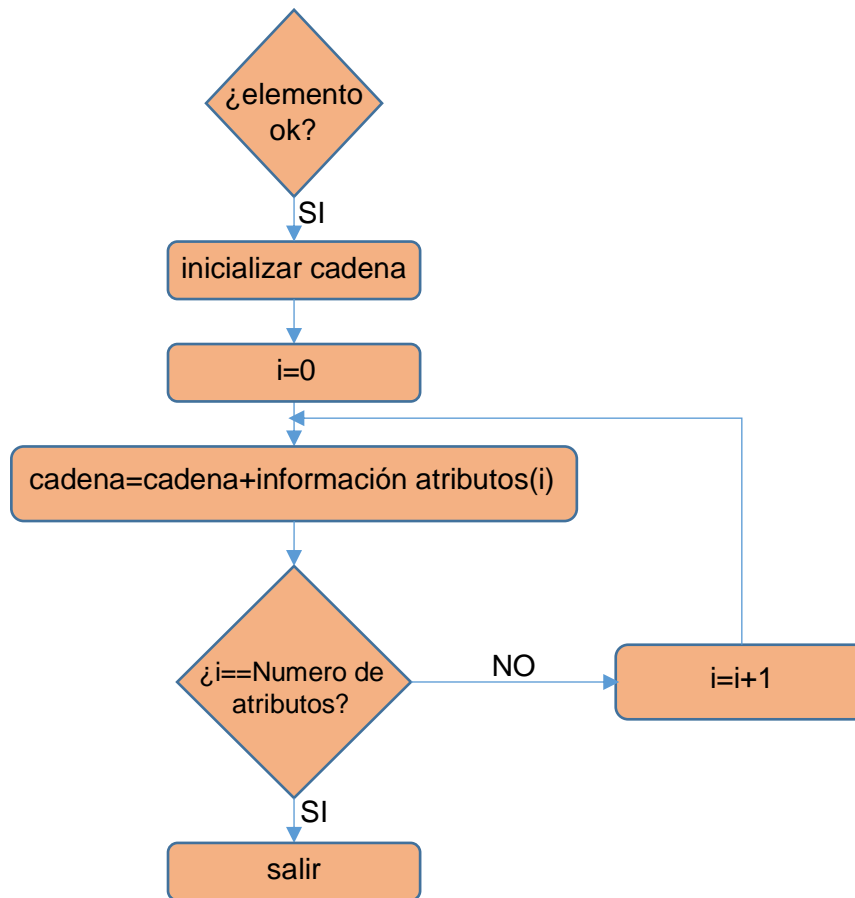



Diagrama de flujo 7.19

```

public static String getInformación(Element elemento, String nombreCampo,
String nombreAtributo)
    
```

Descripción: Devuelve una cadena con la información requerida (**campo** y **atributo**) del elemento.

```

(out)String= getInformación(Element elemento, String nombreCampo,
String nombreAtributo)
{
    campo=encontrarCampo (elemento, nombreCampo)
    valor=campo.encontrarAtributo (nombreAtributo)
    return valor
}
    
```

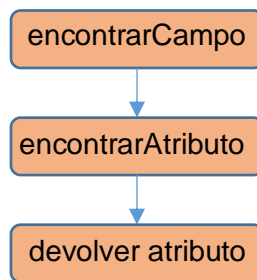


Diagrama de flujo 7.20

```
public static String getInformación(Element elemento, String nombreCampo, String[] nombreAtributos)
```

Descripción: parecido al anterior pero este se usa para el atributo opcua.cadena. Este atributo difiere del resto al ser una cadena que puede ser larga y que se encuentra en varios atributos. La diferencia radica en la concatenación de dicha cadena hasta que se llegue a la longitud máxima.

```
(out)String= getInformación(Element elemento, String nombreCampo, String[] nombreAtributos)
{
    campo=encontrarCampo(elemento, nombreCampo)
    for i, nombreAtributos
        concatenar valor=campo.valorAtributo(i)
    si tamaño>tamañoMáximo
        truncarTamaño
    return cadena
}
```

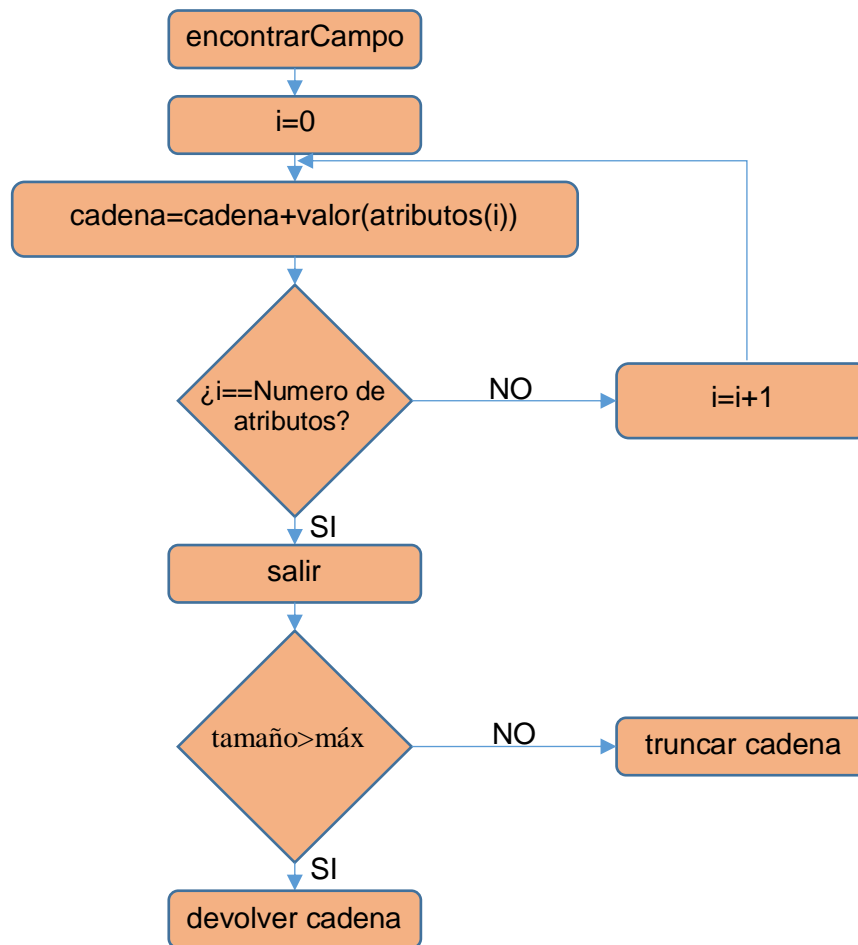


Diagrama de flujo 7.21

```
public static String[][] getMatrizDatos(Element raíz,int tipo)
```

Descripción: crea y devuelve una matriz con la información desde un nodo (**raíz**) y dado un tipo de *archivo* (**opc** o **iec**).

```
(out)String[][]=getMatrizDatos(Element raíz,int tipo)
{
    inicializarMatriz(tipo)
    for i,númeroDePaquetes
        e=paquete(i)
        for j,númeroAtributos
            matriz[i][j]=atributo
    return matriz
}
```

```
public static String buscarInfo(Element elemento, String campo)
```

Descripción: busca en el árbol, a partir de elemento, un campo y devuelve su valor. Se busca primero en los atributos del propio elemento y, si no se encuentra, se sigue buscando en los hijos.

```
(out) String=buscarInfo(Element elemento, String campo)
{
  comprobarElemento
  atributos=elemento.getAtributos
  for i,númeroAtributos
    if campo=nombreAtributo(i)
      return valorAtributo(i)

  hijos=elemento.getHijos
  for i,númeroHijos
    elementoAux=hijos(i)
    buscarInfo(elementoAux,campo)
}
```

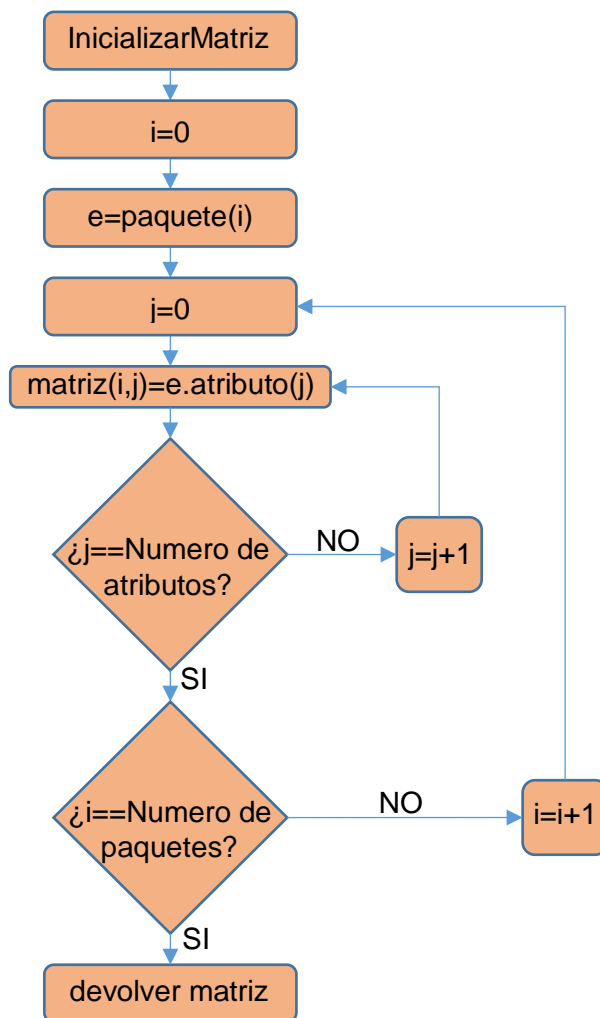


Diagrama de flujo 7.22

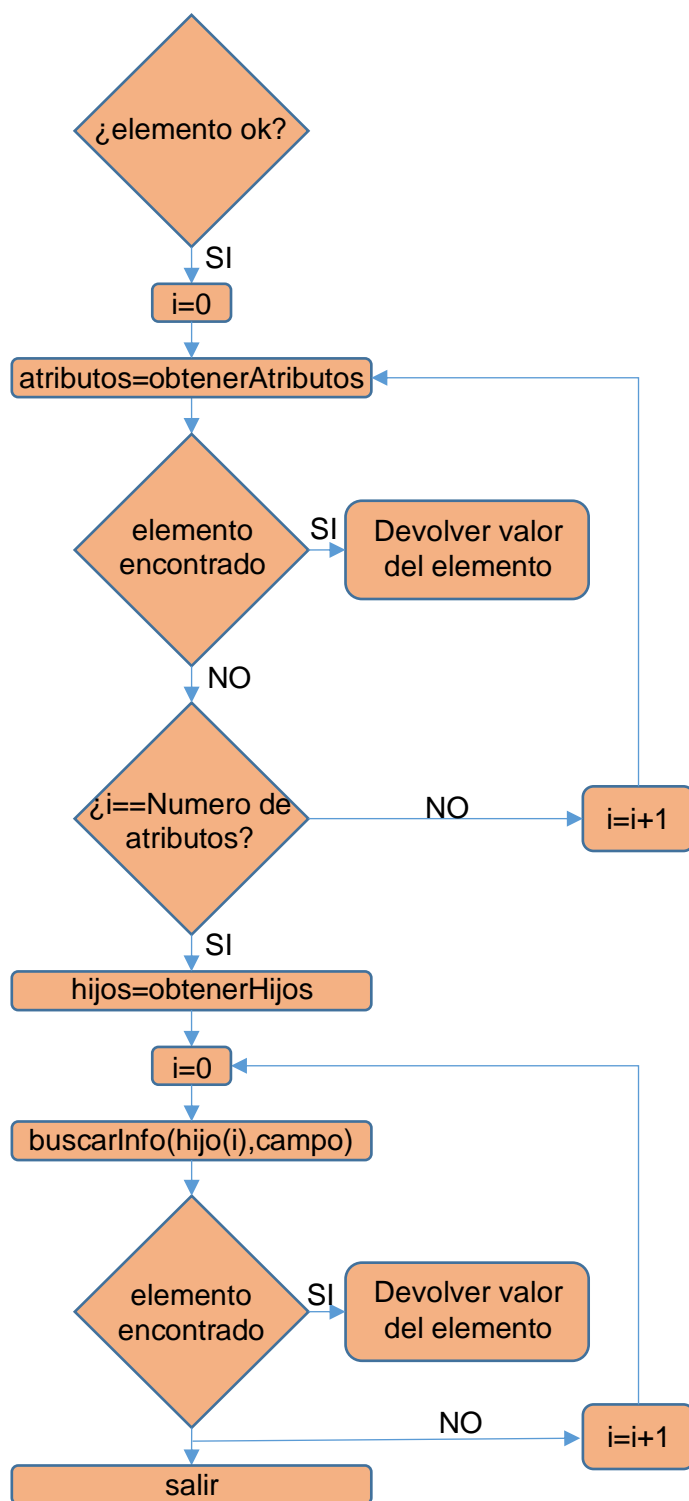


Diagrama de flujo 7.23

```
public static Element encontrarElemento(Element elemento, String valor)
```

Similar al anterior pero en lugar de buscar un campo para obtener su valor, busca el valor y devuelve el elemento. Se usa para buscar los paquetes ACK. Ejemplo: el paquete

Capítulo: 7 IMPLEMENTACIÓN

80 es el ack del paquete 77; se debe buscar el paquete 77 para averiguar la diferencia entre el paquete 77 y 80.

(files.Paquete)

```
public static boolean esPaquete (Element elemento)
```

Descripción: comprueba que el elemento es un paquete.

```
(out) Boolean=esPaquete (Element elemento)
{
    comprobarElemento
    return nombreElemento=="packet"
}
```

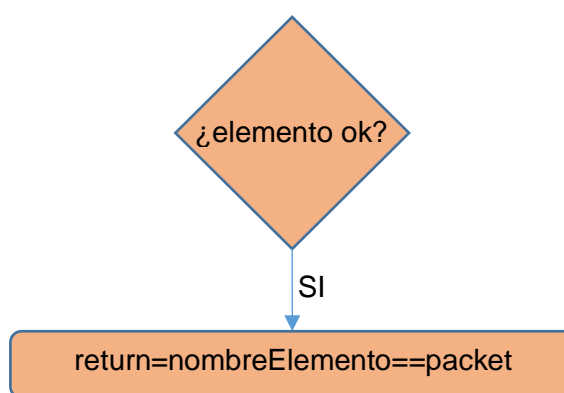


Diagrama de flujo 7.24

```
public static boolean esProtocolo (Element elemento)
```

Descripción: Similar al anterior pero se comprueba que sea protocolo (nombreElemento==proto).

```
public static boolean contieneOPCUA (Element elemento)
```

Descripción: similar a los anteriores pero ahora se comprueba el nombre del protocolo.

```
public String infoACadena (String separador)
```

Descripción: pasa la información contenida en el vector de atributos a una cadena.

La información se separa mediante la cadena separador.

```
(out) String=infoACadena (String separador)
{
    inicializar cadena
    for i,número de atributos
        cadena=cadena+atributo(i)
}
```

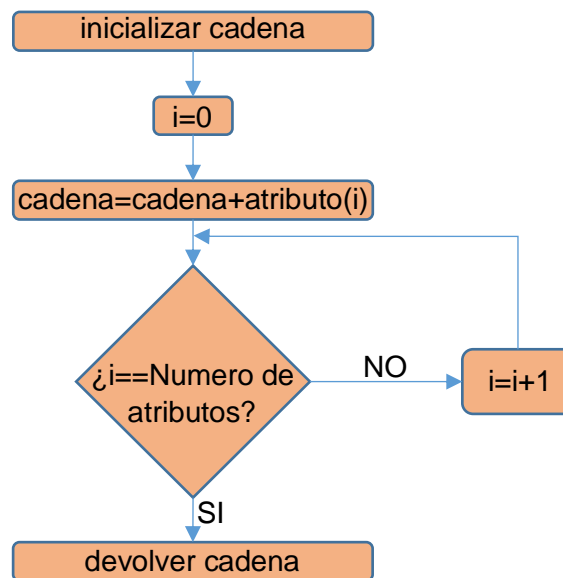


Diagrama de flujo 7.25

```
public void print(String separador)
```

Descripción: imprime por pantalla una cadena con los valores de los atributos.

```
(out) String=infoACadena (String separador)
{
    cadena=infoACadena
    devolver cadena
}
```

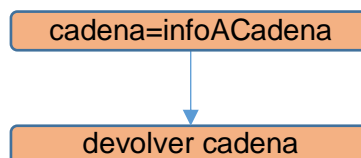


Diagrama de flujo 7.26

7.3.3.- PAQUETE GRÁFICOS

Clase gráficos.Diálogo1

Esta clase maneja el cuadro diálogo que se usa para guardar y mostrar los datos.

```
public Diálogo1(int tipo)
```

Descripción: constructor de la clase. Crea un **diálogo** según el tipo seleccionado (**opc** o **iec**).

```
(out) void=Diálogo1(int tipo)
{
    Crear y añadir botones
    Crear y añadir panelCentral
}
```

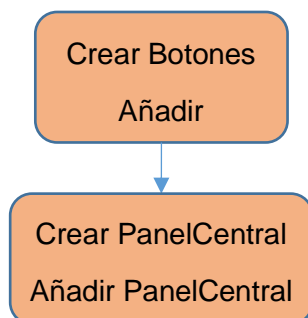


Diagrama de flujo 7.27

```
public class PanelCentralDialogo extends JPanel
{
```

Descripción: Clase interna en **Diálogo1**. Para el panel central con los cuadros de texto para introducir los datos.

```
public PanelCentralDiálogo(int tipo)
```

Descripción: constructor de la clase. Crea un panel central que depende del tipo (para la introducción automática de datos).

```
PanelCentralDiálogo(int tipo)
{
    Crear panel
    Añadir etiquetas
    Crear, rellenar y añadir los cuadros de texto
}
```

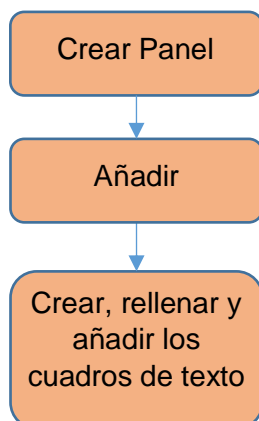


Diagrama de flujo 7.28


```
public String getHost () {return this.host.getText ();}
```

Descripción: devuelve el contenido del cuadro de texto host.

```
public String getPuerto () {return this.puerto.getText ();}
```

Descripción: devuelve el contenido del cuadro de texto puerto.

```
public String getUsuario () {return this.usuario.getText ();}
```

Descripción: devuelve el contenido del cuadro de texto usuario.

```
public String getTabla () {return this.tabla.getText ();}
```

Descripción: devuelve el contenido del cuadro de texto tabla.

```
public String getBaseDeDatos () {return this.baseDeDatos.getText ();}
```

Descripción: devuelve el contenido del cuadro de texto base de datos.

```
public String getContraseña ()
```

Descripción: devuelve el contenido del cuadro de texto contraseña. Atención: devuelve una cadena normal.

```
(out)String=getContraseña(int tipo)
{
    Obtener vector de caracteres que conforman la contraseña
    Devolver cadena de caracteres
}
```

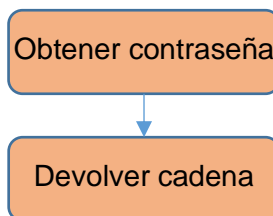


Diagrama de flujo 7.29

```
}//fin de la clase interna PanelCentralDiálogo
```

```
public class ContinuarListener implements ActionListener
```

Descripción: clase que implementa el listener para el botón continuar

```
{
public void actionPerformed (ActionEvent e)
```

Descripción: acción que se realiza al disparar este evento. Cierra el diálogo.

```
(out) void=actionPerformed(ActionEvent e)
{
    Continuar=true
    Obtener diálogo que hay que cerrar
    Cerrar diálogo
}
```

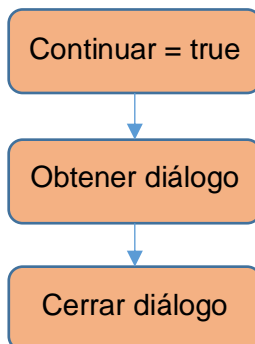


Diagrama de flujo 7.30

```
} //fin de la clase interna ContinuarListener
```

```
private class CancelarListener implements ActionListener
```

Descripción: clase que implementa el listener para el botón continuar.

```
{
public void actionPerformed(ActionEvent e)
```

Descripción: acción que se realiza al disparar este evento. Cierra el diálogo.

```
(out) void=actionPerformed(ActionEvent e)
{
    Continuar=false
    Obtener diálogo que hay que cerrar
    Cerrar diálogo
}
```

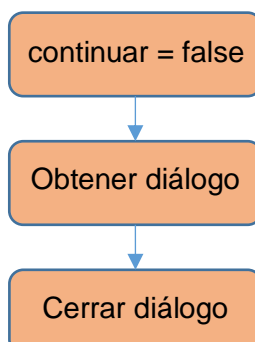


Diagrama de flujo 7.31

```
} //fin de la clase interna CancelarListener
```

Clase gráficas.DiálogoCalcular

Esta clase maneja el cuadro diálogo que se usa para calcular las deltas. Igual que el anterior pero con el añadido de las direcciones **IP** para el cálculo de las deltas.

```
public DiálogoCalcular (int tipo)
```

Descripción: constructor de la clase. Crea un diálogo según el tipo seleccionado (**opc** o **iec**).

```
(out) void=Diálogo1(int tipo)
{
    Crear y añadir botones
    Crear y añadir panelCentral
}
```

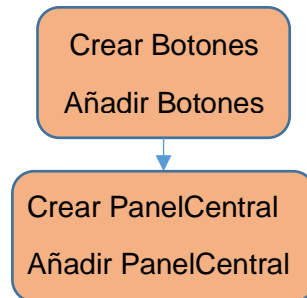


Diagrama de flujo 7.32

```
public class PanelCentralDiálogo extends JPanel{
```

Descripción: Clase interna en **Diálogo1**. Para el panel central con los cuadros de texto para introducir los datos.

```
public PanelCentralDiálogo (int tipo)
```

Descripción: constructor de la clase. Crea un panel central que depende del tipo (introducción automática de datos).

```
PanelCentralDiálogo(int tipo)
{
    Crear panel
    Añadir etiquetas
    Crear, rellenar y añadir los cuadros de texto
}
```

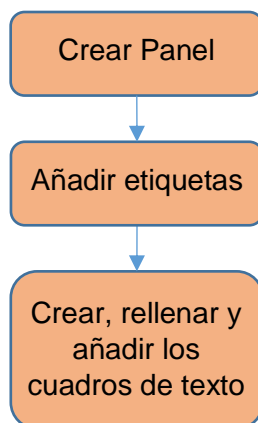


Diagrama de flujo 7.33

```
public String getHost () {return this.host.getText ();}
```

Descripción: devuelve el contenido del cuadro de texto host

```
public String getPuerto () {return this.puerto.getText ();}
```

Descripción: devuelve el contenido del cuadro de texto puerto

```
public String getUsuario () {return this.usuario.getText ();}
```

Descripción: devuelve el contenido del cuadro de texto usuario

```
public String getTabla () {return this.tabla.getText ();}
```

Descripción: devuelve el contenido del cuadro de texto tabla

```
public String getBaseDeDatos () {return this.baseDeDatos.getText ();}
```

Descripción: devuelve el contenido del cuadro de texto base de datos

```
public String getContraseña ()
```

Descripción: devuelve el contenido del cuadro de texto contraseña. Atención: devuelve una cadena normal.

```
(out)String=getContraseña ()
{
    Obtener vector de caracteres que conforman la contraseña
    Devolver cadena de caracteres
}
```

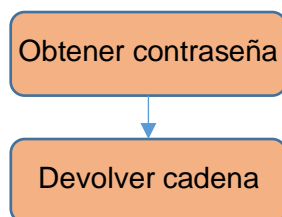


Diagrama de flujo 7.34

```
} //fin de la clase interna PanelCentralDiálogo
```

```
public class ContinuarListener implements ActionListener
Descripción: clase que implementa el listener para el botón continuar
{
public void actionPerformed(ActionEvent e)
```

Descripción: acción que se realiza al disparar este evento. Cierra el **diálogo**.

```
(out) void=actionPerformed(ActionEvent e)
{
    Continuar=true
    Obtener diálogo que hay que cerrar
    Cerrar diálogo
}
```

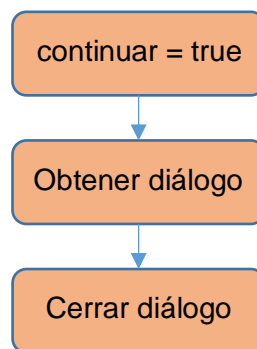


Diagrama de flujo 7.35

```
} //fin de la clase interna ContinuarListener
```

```
private class CancelarListener implements ActionListener
Descripción: clase que implementa el listener para el botón continuar.
{
public void actionPerformed(ActionEvent e)
```

Descripción: acción que se realiza al disparar este evento. Cierra el **diálogo**.

```
(out) void=actionPerformed(ActionEvent e)
{
    Continuar=false
    Obtener diálogo que hay que cerrar
    Cerrar diálogo
}
```

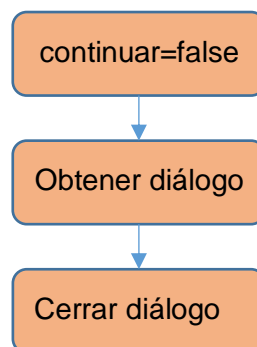


Diagrama de flujo 7.36

Capítulo: 7 IMPLEMENTACIÓN

```
}//fin de la clase interna CancelarListener
```

Clase graficos.VentanaInterna

Descripción: esta clase maneja la ventana que se crea cuando se selecciona un archivo. En ella se muestran los datos que se extraen de dicho archivo.

```
0public VentanaInterna (String nombre ,boolean b1,boolean b2,boolean  
b3,boolean b4,Archivo archivo,int tipo)
```

Descripción: constructor de la clase. Crea una ventana interna y la muestra dentro de la ventana principal. Parámetros de entrada:

- **nombre:** nombre de la ventana.
- **tipo:** OPC-UA o IEC.
- **b1,b2,b3,b4:** variables tipo Boolean (resizable, closable, maximizable, y iconifiable respectivamente).

```
VentanaInterna (String nombre ,boolean b1,boolean b2,boolean b3,boolean  
b4,Archivo archivo,int tipo)  
{  
    Indicar opciones a la ventana  
    Obtener datos del archivo  
    Crear tabla con los datos  
    Añadir menús a la ventana  
    Añadir tabla a la ventana  
}
```

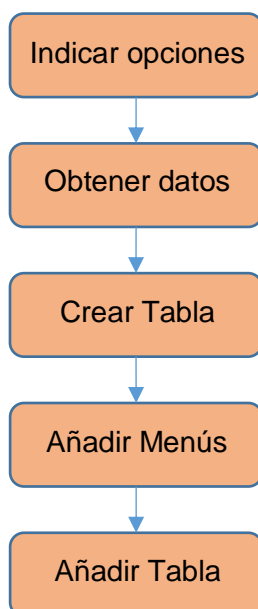


Diagrama de flujo 7.37

```
private class GuardarListener implements ActionListener  
{
```

Descripción: Clase interna que se ocupa de guardar los datos del archivo en la base de datos. Se activa al pulsar **guardar** en el **menú**.

```
public void actionPerformed(ActionEvent event1)
```

```
void=actionPerformed(ActionEvent event1)
{
    Crear Diálogo1 y mostrarlo
    si continuar==true //si se pulsa continuar
    leer los datos introducidos en los cuadros de texto del diálogo
    Crear cadena de conexión
    Guardar datos //guardarEnBaseDeDatos
    Comprobar resultado
    si ok
        mostrar ok
    si no
        mostrar error
}
```

```
}//fin clase privada GuardarListener
```

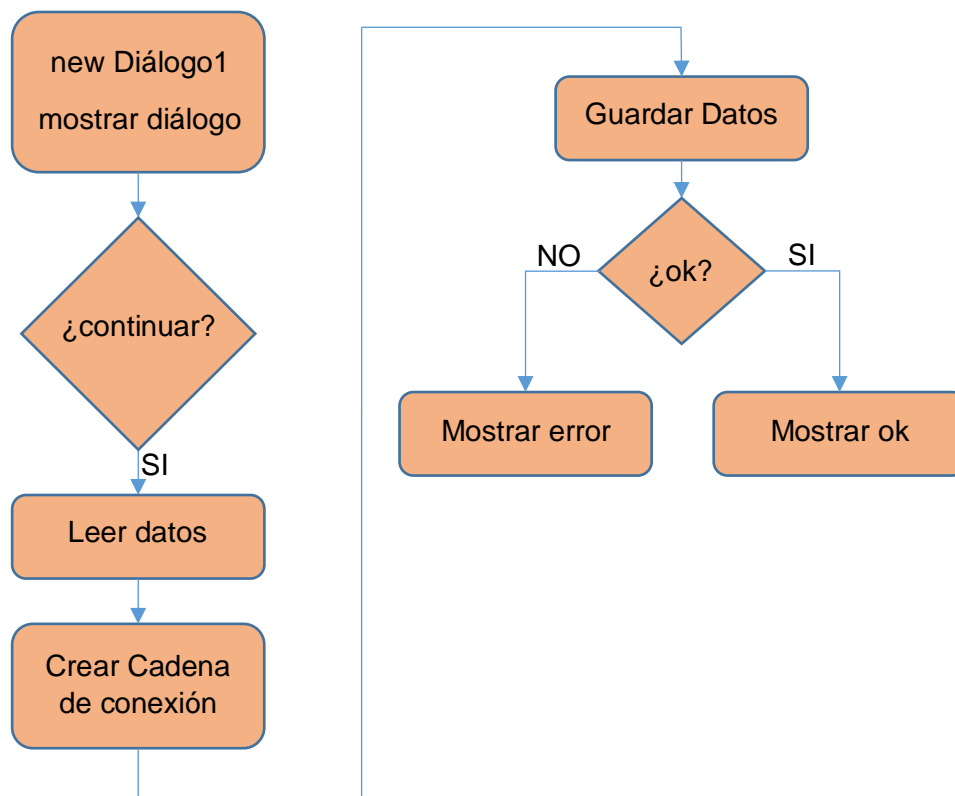


Diagrama de flujo 7.38

Clase graficos.VentanaInternaCalcular

```
public VentanaInternaCalcular(String nombre,int[] p1, double[]d1,String
s1,int[]p2,double[]d2,String s2)
```

Capítulo: 7 IMPLEMENTACIÓN

Descripción: constructor de la clase. Crea una ventana interna con los gráficos de los cálculos de las **deltas**.

VARIABLES DE ENTRADA:

- **nombre**: nombre de la ventana.
- **p1,p2**: índices de los paquetes de la serie 1 o 2.
- **d1,d2**: deltas de los paquetes de la clase 1 o 2.
- **s1,s2**: nombres de las series 1 o 2.

```
VentanaInternaCalcular(String nombre,int[] p1, double[]d1,String s1,int[]p2,double[]d2,String s2)  
{  
    Crear gráfico  
    Calcular datos complementarios  
    Crear panel con los datos complementarios  
    Crear panel e insertar el gráfico y el panel complementario  
    Visualizar ventana  
}
```

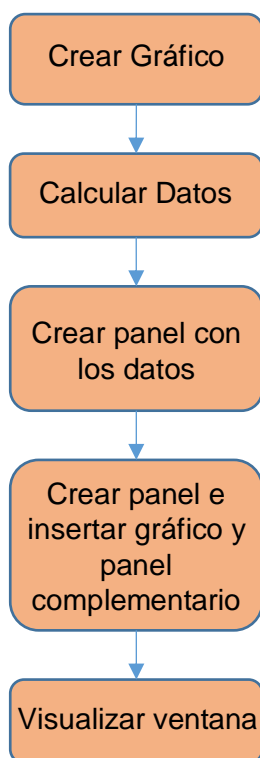


Diagrama de flujo 7.39

Clase graficos.VentanaInternaMostrar

Descripción: Clase que se encarga de mostrar los resultados leídos de la base de datos.

```
public VentanaInternaMostrar(String nombre,Object[][]datos,  
String[]nombresColumnas)
```


Descripción: constructor de la clase. Crea una ventana interna en la que se muestran los datos.

- **nombre**: nombre de la ventana.
- **datos**: matriz con los datos que se mostrarán.
- **nombresColumnas**: nombres de las columnas de la tabla.

```
VentanaInternaCalcular(String nombre, int[] p1, double[] d1, String s1, int[] p2, double[] d2, String s2)
{
    Crear tabla
    Añadir tabla
    Visualizar ventana
}
```

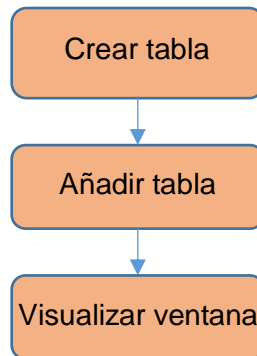


Diagrama de flujo 7.40

Clase Gráficos.VentanaPrincipal

Descripción: Clase en la que se describe la ventana principal de la aplicación. El método **main** se encuentra en esta clase.

```
public VentanaPrincipal ()
```

Descripción: constructor de la clase. Crea la ventana y añade los **menús** y las **opciones de la ventana**.

```
VentanaPrincipal ()
{
    Crear la ventana
    Crear y añadir menús
    Establecer opciones
}
```

```
public static File seleccionarArchivoYDevolverArchivo(String nombre)
```

Descripción: muestra un menú para seleccionar un archivo y devuelve dicho

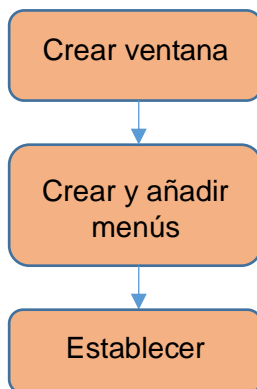


Diagrama de flujo 7.41

archivo.

- **nombre:** nombre del menu.

```
(out) File=seleccionarArchivoYDevolverArchivo(String nombre)
{
    Crear diálogo
    Mostrar diálogo
    Devolver archivo seleccionado
}
```

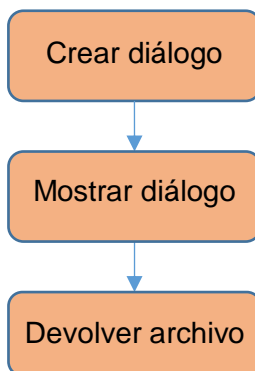


Diagrama de flujo 7.42

```
public class AbrirListener implements ActionListener{
Descripción: clase interna que maneja la apertura de un archivo.
```

```
public void actionPerformed(ActionEvent event1)
```

```
(out)void= actionPerformed(ActionEvent event1)
{
    seleccionarArchivoYDevolverArchivo
    Comprobar archivo
    si archivo ok
        abrir y mostrar nueva VentanaInterna
    si no
        mostrar error
}
```

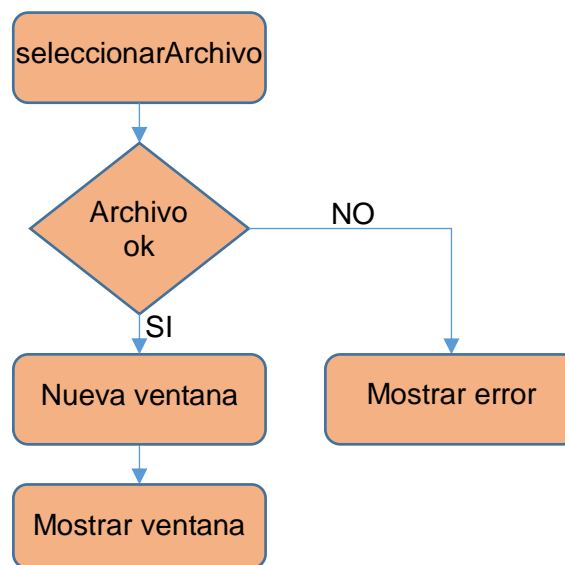


Diagrama de flujo 7.43

```
public class MostrarTablaListener implements ActionListener
{
```

Descripción: clase interna que se encarga de mostrar todos los datos procedentes de la base de datos.

```
Public void actionPerformed(ActionEvent e)
```

```
void=actionPerformed(ActionEvent event1)
{
    Crear Diálogo1 y mostrarlo
    si continuar==true //si se pulsa continuar
    leer los datos introducidos en los cuadros de texto del diálogo
    Crear cadena de conexión
    Leer datos
    Comprobar resultado
    si ok
        crear y mostrar nueva VentanaInternaMostrar
    si no
        mostrar error
}
```

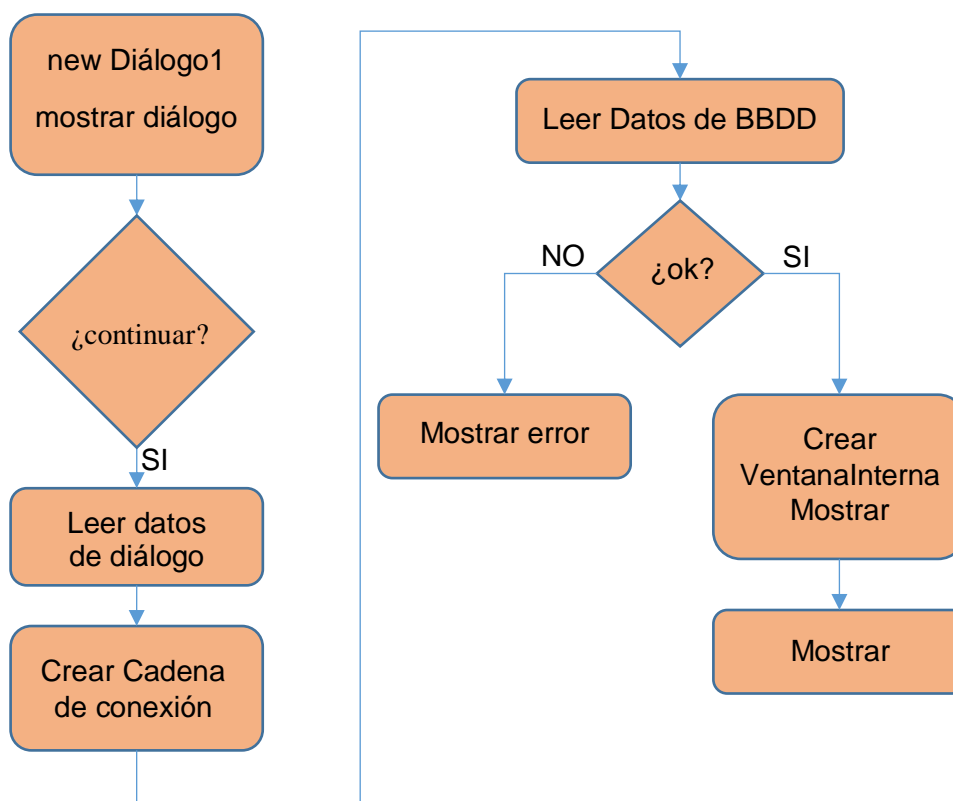


Diagrama de flujo 7.44

```
}//fin MostrarTablaListener
```

```
public class MostrarTablaListenerACK implements ActionListener
```

Descripción: Igual que la anterior pero solo con los **ACK**.

```
public class CalcularListener implements ActionListener
```

```
{
```

```
public void actionPerformed(ActionEvent e)
```

```
}
```

```
public class Listener implements ActionListener
```

Descripción: clase interna que se encarga de calcular y mostrar los datos de los deltas. Crea dos cadenas de conexión y lee dos veces la base de datos (dos direcciones IP, para calcular las diferencias de tiempo)

```
{
```

```
public void actionPerformed(ActionEvent e)
```

```

void=actionPerformed(ActionEvent event1)
{
    Crear DiálogoCalcular y mostrarlo
    si continuar==true //si se pulsa continuar
    leer los datos introducidos en los cuadros de texto del diálogo
    Crear cadena de conexión
    Leer datos
    Comprobar resultado
    si ok
        crear y mostrar nueva VentanaInternaMostrar
    si no
        mostrar error
}
    
```

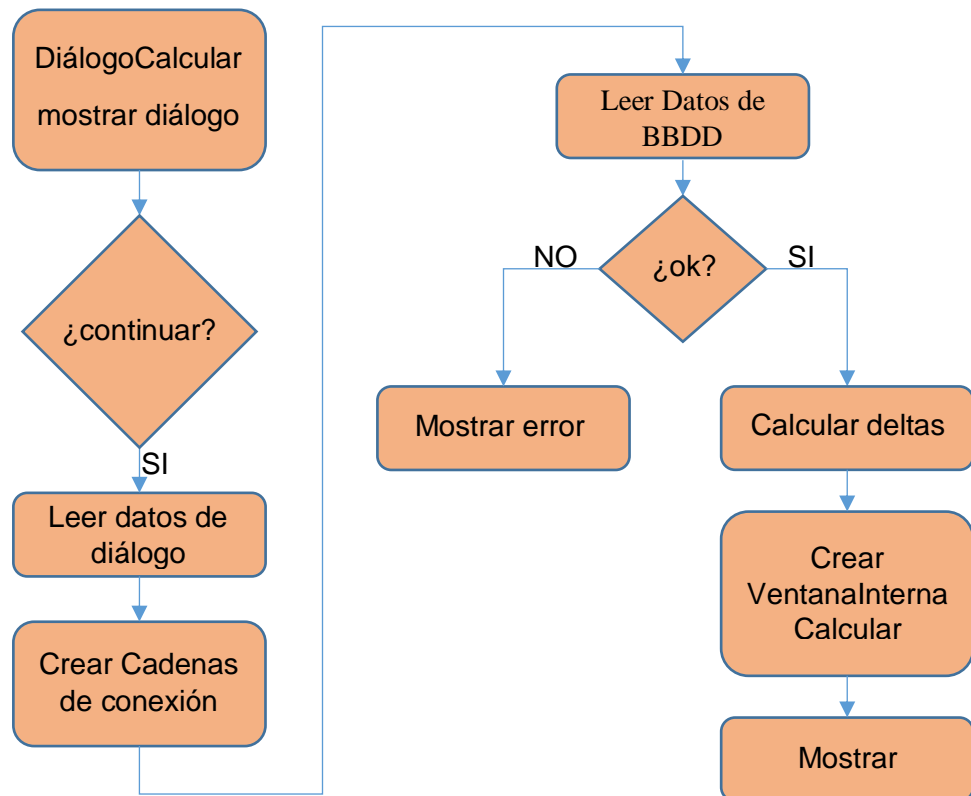


Diagrama de flujo 7.45

7.3.4.- PAQUETE OPERACIONES

Clase operaciones.Operaciones

Esta clase es una librería de funciones estáticas que se usan en diferentes partes del programa.

```
public static double media (double[]v)
```

Capítulo: 7 IMPLEMENTACIÓN

Descripción: realiza la media de un **vector** de *double*.

```
(out) double=media(double[] v)
{
    Hacer media
    Devolver valor
}
```

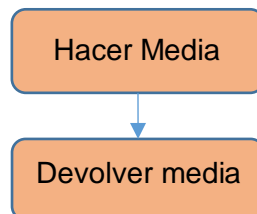


Diagrama de flujo 7.46

```
public static double varianza(double[] v)
```

Descripción: realiza la varianza de un vector de *double*.

```
(out) double=varianza(double[] v)
{
    Hacer varianza
    Devolver valor
}
```

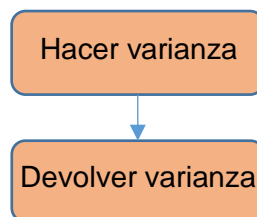


Diagrama de flujo 7.47

```
public static double desviaciónEstándar(double[] v)
```

Descripción: realiza la desviación estándar de un **vector** de *double*.

```
(out) double=varianza(double[] v)
{
    Hacer desviación estándar
    Devolver valor
}
```

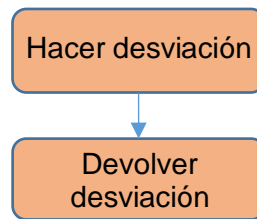


Diagrama de flujo 7.48

```
public static int encontrarMínimo(double[] v)
```

Descripción: busca el elemento más pequeño dentro de un vector de double.

```
(out) double=mínimo(double[]vector)
{
  Comprobar vector
  min= vector[0]
  pos=0
  for i=1 to i==longitudVector -1
    si vector [i]<min
      pos=i
      min= vector [i]
    end si
  end for
  Devolver i
}
```

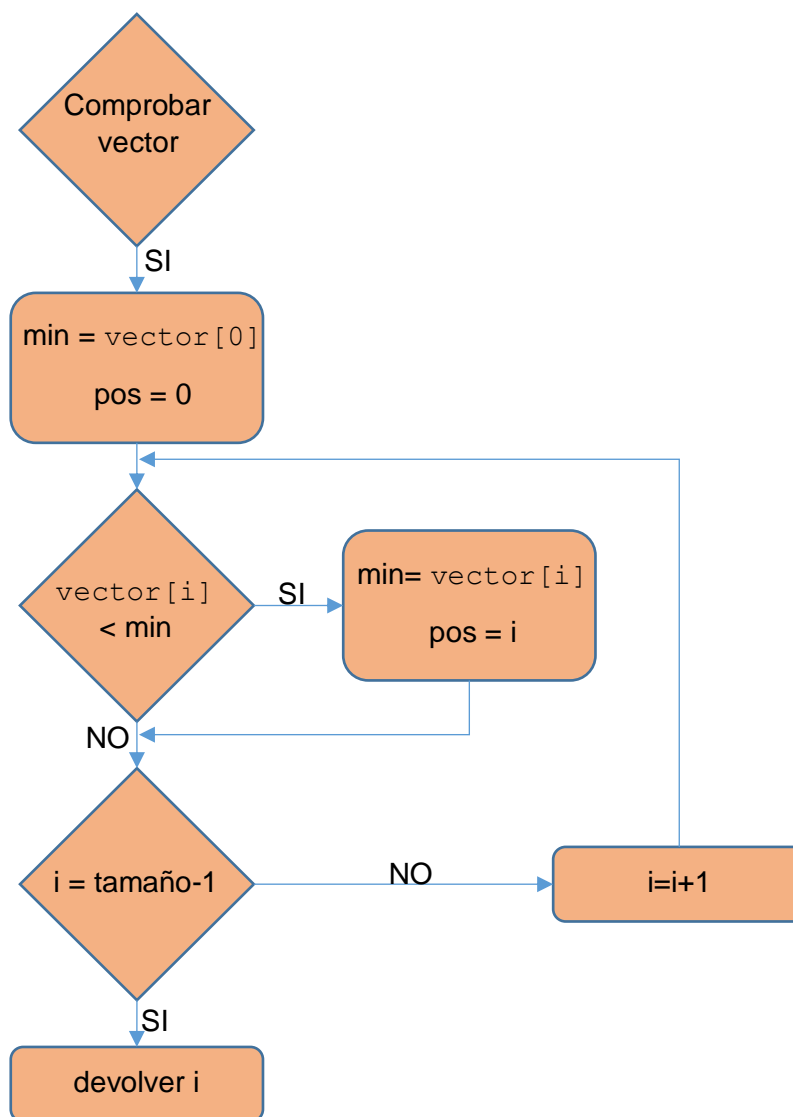


Diagrama de flujo 7.49

```
public static int encontrarMáximo(double[] v)
```

Descripción: busca el elemento más grande dentro de un vector de double.

```
(out) double=máximo(double[]vector)
{
  Comprobar vector
  máx= vector[0]
  pos=0
  for i=1 to i==longitudVector -1
    si vector [i]>máx
      pos=i
      máx= vector [i]
    end si
  end for
  Devolver i
}
```

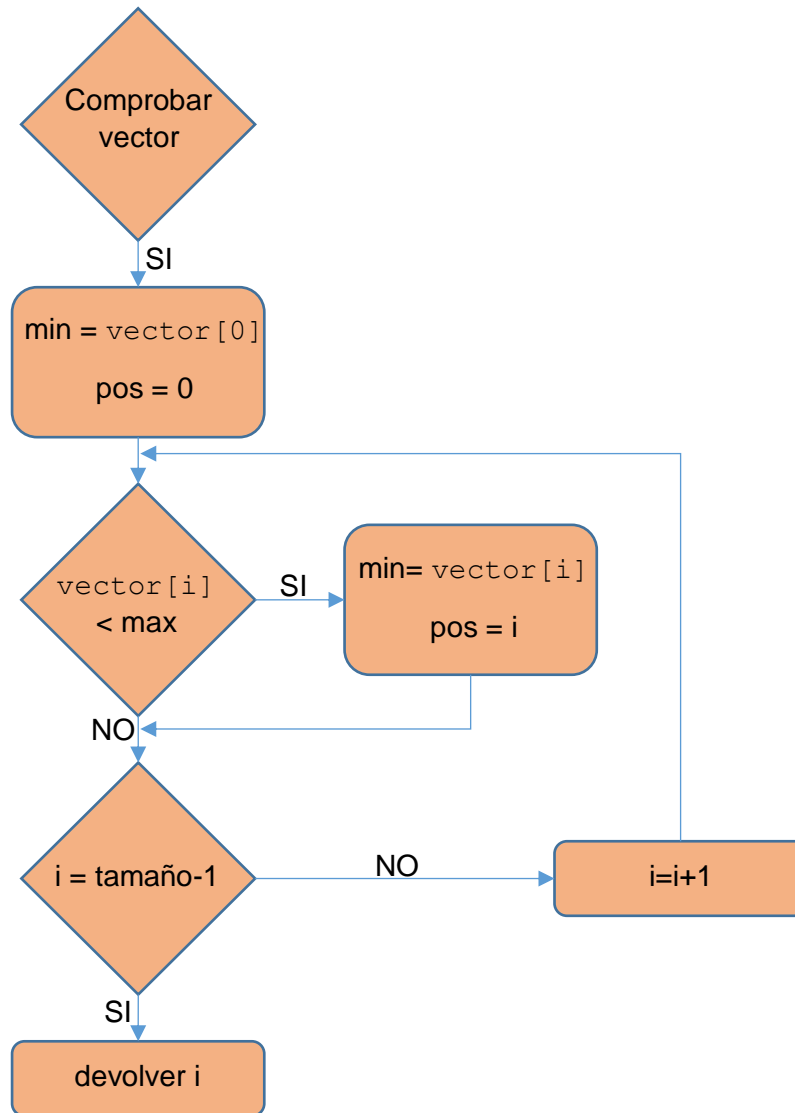



Diagrama de flujo 7.50

```
public static int getTipo(String nombreBotón)
```

Descripción: Devuelve el tipo del botón (constante entera) a partir de su nombre. Se usa para saber qué botón se pulsó y distinguir si se trata de **OPC-UA** o de **IEC**.

```
(out) int=getTipo (String nombreBotón)
{
    si nombreBotón==NombreBotónIEC
        tipo=IEC
    else if nombreBotón==NombreBotónOPC
        tipo=OPC
    Devolver tipo
}
```

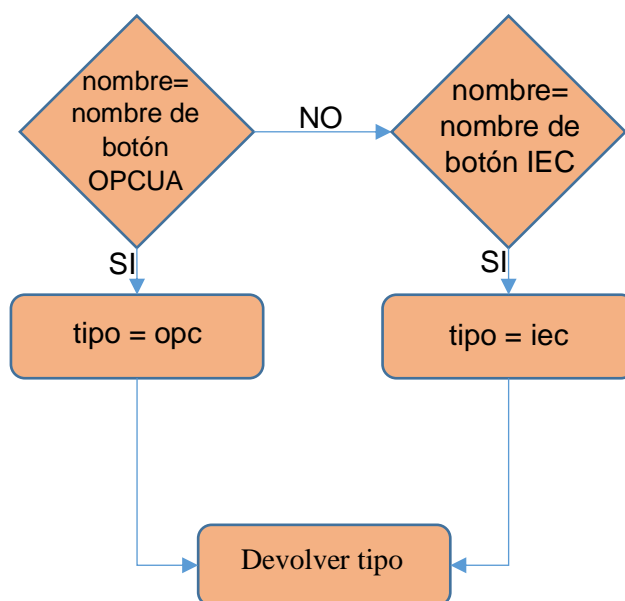


Diagrama de flujo 7.51

```
public static String[] getNombresBaseDeDatos (Vector<Object>v)
```

Descripción: Se devuelve el primer elemento del **vector** (que es el vector con los nombres extraído de la base de datos)

```
(out) String[]=getNombresBaseDeDatos (Vector<Object>)  
{  
    Devolver Vector(0)  
}
```

```
public static Vector<Object> leerTodoDeBaseDeDatos (String  
cadenaConexión,String tabla, String usuario,String contraseña)
```

Descripción: Lee la base de datos y devuelve un vector de objetos con los datos leídos. Si hubo error, hay nulo en **posición 0** y el mensaje de error está en la **posición 1**. Si no hay error el vector con los nombres está en la posición 0 y un vector con los datos de cada fila en cada una de las siguientes columnas.

Las variables de entrada son la cadena de conexión, el nombre de la **tabla**, nombre de usuario y contraseña para realizar la conexión con la base de datos.

```
(out) Vector<Object>=leerTodoDeBaseDeDatos(String
cadenaConexión,String tabla, String usuario,String contraseña)
{
    Conectar a base de datos
    Leer Base de datos
    Comprobar lectura
    Devolver resultado
}
```

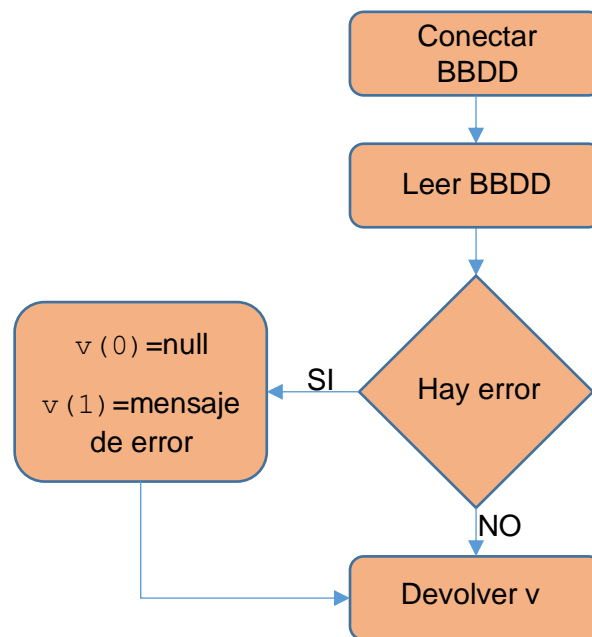


Diagrama de flujo 7.52

```
public static Vector<Object> leerTodoDeBaseDeDatosACK(String
cadenaConexión,String tabla, String usuario,String contraseña)
```

Descripción: Igual que el anterior pero lee únicamente los **ACK**.

```
(out) Vector<Object>=leerTodoDeBaseDeDatosACK(String
cadenaConexión,String tabla, String usuario,String contraseña)
{
    Conectar a base de datos
    Leer Base de datos
    Comprobar lectura
    Devolver resultado
}
```

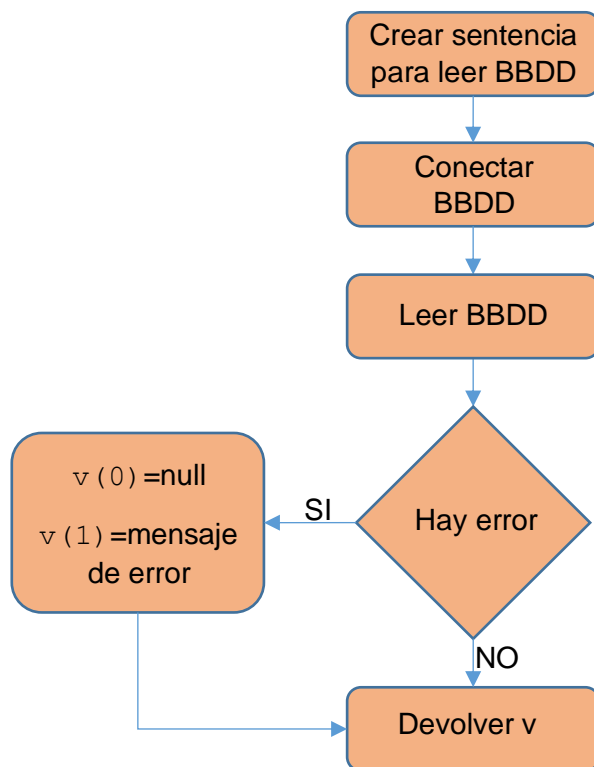


Diagrama de flujo 7.53

```
public static Vector<Object> leerDeBaseDeDatos (String cadenaConexión,String tabla, String usuario,String contraseña,String ipSrc,tring ipDst,int opción)
```

Descripción: Como los anteriores pero se le añade un parámetro para leer los deltas o todos los datos de la base de datos. También se añaden las direcciones **IP** para leer solo aquellos datos cuyas **IP** coincidan con las dadas.

```
(out) Vector<Object>=leerDeBaseDeDatos (String cadenaConexión,String tabla, String usuario,String contraseña)
{
    Conectar a base de datos
    Leer Base de datos
    Comprobar lectura
    Devolver resultado
}
```

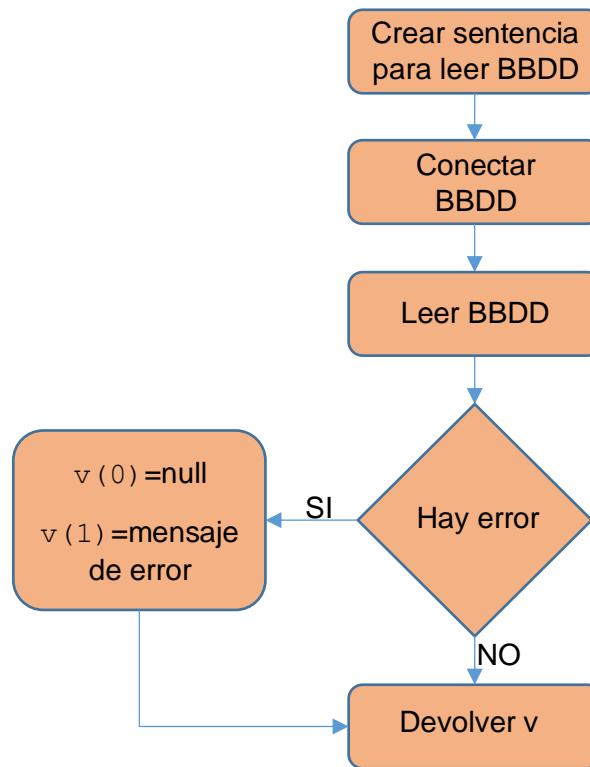


Diagrama de flujo 7.54

`public static String getErrorLectura (Vector<Object>v)`

Descripción: Para la comprobación de error de lectura de la base de datos.

```

(out) String=getErrorLectura (Vector<Object>v)
{
  Comprobar error
  Devolver resultado de comprobación
}
  
```

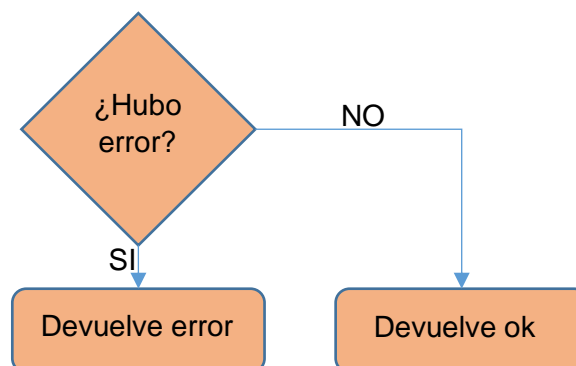


Diagrama de flujo 7.55

Capítulo: 7 IMPLEMENTACIÓN

```
public static Object[][] getDatosVectorBaseDeDatos (Vector<Object>v, int
columnas)
```

Descripción: A partir de la lectura de la base de datos, devuelve una **matriz** con dichos datos.

```
(out) Object[][]=getDatosVectorBaseDeDatos (Vector<Object>v, int
columnas)
{
    Comprobar tamaño vector
    Crear matriz
    Llenar Matriz
    Devolver matriz
}
```

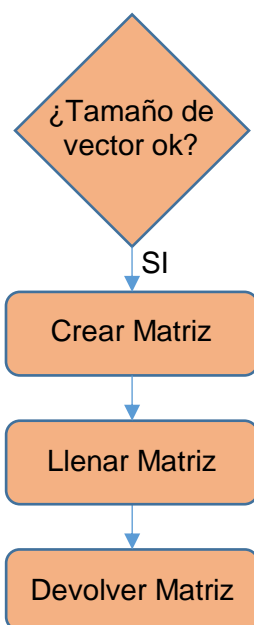


Diagrama de flujo 7.56

```
public static double[] getDelta (Vector<Object>v)
```

Descripción: A partir de la lectura de la base de datos, obtiene los datos de las **deltas**.

```
(out) double[]=getDelta (Vector<Object>v)
{
    Comprobar error lectura
    Crear vector de deltas
    Llenar vector de deltas
    Devolver vector de deltas
}
```

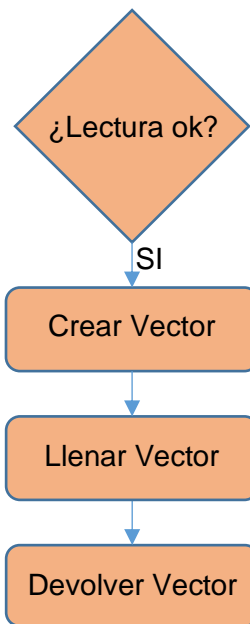


Diagrama de flujo 7.57

```
public static int[] getPaquetes (Vector<Object>v)
```

Descripción: A partir de la lectura de la base de datos, obtiene los índices de los **paquetes**.

```
(out) int[]=getPaquetes (Vector<Object>v)
{
    Comprobar error lectura
    Crear vector de índices
    Llenar vector de índices
    Devolver vector de índices
}
```

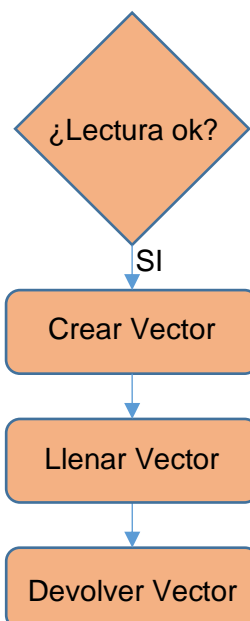


Diagrama de flujo 7.58

Capítulo: 7 IMPLEMENTACIÓN

```
public static String guardarEnBaseDeDatos (String cadenaConexión, String
tabla, String usuario, String contraseña, Archivo a, String[] columnas, int
tipo)
```

Descripción: **Guarda** los datos del archivo a en la base de datos.

Las variables de entrada son la cadena de conexión, el nombre de la tabla, nombre de usuario y contraseña para realizar la conexión con la base de datos; el archivo es el que se lee para obtener los datos que hay que guardar.

```
(out) String=crearGráfico guardarEnBaseDeDatos (String
cadenaConexión, String tabla, String usuario, String contraseña, Archivo
a, String[] columnas, int tipo)
{
    Crear sentencia
    Conectar BBDD
    Guardar datos (ejecutar sentencia)
}
```

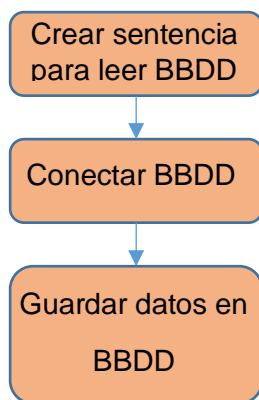


Diagrama de flujo 7.59

BIBLIOGRAFÍA

- Adriano da Acosta. (2015). *“Industry 4.0 - El Rol de OPC UA en una Fábrica Inteligente”*. [En Línea]. Recuperado de: <https://www.matrikonopc.com/downloads/1317/whitepapers/index.aspx>
- AGM – Larraioz Elektronika. (2015). *“OPC: Desde el clásico al nuevo OPC-UA”*. [En Línea]. Recuperado de: <http://larraioz.com/en/?p=739&lang=en>
- Axon Group. *“Teoría en Protocolo IEC 60870-5-104”*. [En Línea]. Recuperado de: http://www.axongroup.com.co/protocolo_iec_160870_5_104.php
- Carlos A. Castaño Gómez; Edgar F. Villamarín Meneses y Oscar A. Rojas Alvarado. (2012). *“PRINCIPIOS DE APLICACIÓN DEL ESTANDAR OPC UA EN UN CASO DE ESTUDIO”*. Cauca, Colombia. Universidad de Cauca.
- Eric Murphy. (2009). *“OPC UA: 5 Conceptos que Todos Necesitan Saber”*. Alberta, Canadá. MatrikonOPC.
- Departamento de Métodos y Medios. (2009). *“Protocolo de Comunicaciones entre los sistemas de Gestión del Operador del Sistema y Equipos de Control de Consumidores Interrumpibles y Empresas Distribuidoras (PCC-SCECI)”*. Madrid. Red Eléctrica de España.
- Gimeno, J. Manuel y González J. Luis. (2010-2011). *“Introducción a Netbeans”*. [En Línea]. Recuperado de: <http://ocw.udl.cat/enginyeria-i-arquitectura/programacio-2/continguts-1/1-introduccioi81n-a-netbeans.pdf>
- González Barnosa, H. Darío. (2006). *“Supervisión de las Unidades de Transmisión remota de la Electrificadora de Santander S.A. ESP Mediante Protocolo IEC-60870 5 101”*. Santander. Universidad Industrial de Santander, Facultad de Ingenierías Eléctrica, Electrónica y Telecomunicaciones.

- International Electrotechnical Commission (IEC). (2003). *“Telecontrol equipment and systems –Part 5-101: Transmission protocols – Companion standard for basic telecontrol tasks”*. Ginebra, Suiza. IEC Central Office.
- InfoPLC (2011), *“¿Por qué OPC-UA estará en cada sistema de control automatizado?”* [En línea]. Recuperado de: <http://www.infopl.net/documentacion/7-comunicaciones-industriales/1047-ipor-que-opc-ua-estara-en-cada-sistema-de-control-automatizado>
- Joyanes Aguilar, Luis y Zahonero Martínez, Ignacio. (2010). *“Programación en C, C++, Java y UML”*. México DF. Editorial McGrawHill.
- Lamping, ULF; Sharpe, Richard Sharpe y Warnicke, Ed. (2014). *“Wireshark User’s Guide*
- *For Wireshark 2.1”*. [En Línea]. Recuperado de: <https://www.wireshark.org/download/docs/user-guide-a4.pdf>
- MakitronOPC. (2016). *“Download: Arquitectura Unificada OPC (UA): Todo lo que necesitaba saber (parte dos)”* [En Línea]. Recuperado de: <https://www.matrikonopc.com/downloads/1008/webcasts/index.aspx>
- Manuel Rodríguez. (2016). *“Protocolo OPC UA. Características y aplicaciones en SCADA”*. [En Línea]. Recuperado de: <http://revistadigital.inesem.es/gestion-integrada/protocolo-opc-ua-caracteristicas-y-aplicaciones/>
- Mario del Pozo. (2015). *“OPC UA (OPC Unified Architecture)”*. [En Línea]. Recuperado de:
- <http://www.iacssec.com/2015/03/opc-ua.html>
- Moran Nahin. (2013). *“Tutorial de Netbeans IDE 7.2 Básico”*. [En Línea]. Recuperado de: <https://hocode.wordpress.com/tutorial-de-netbeans-7-2/>
- OPC Foundation. (2016). [En Línea]. Recuperado de: <https://opcfoundation.org/>
- Oracle. (2016). Documentación Netbeans 8. [En Línea]. Recuperado de: <https://netbeans.org/kb/docs/java/javase-jdk8.html>

BIBLIOGRAFÍA

- Reinhard Fink. (2016). [En Línea]. Recuperado de: <http://www.ppfink.de/>
- Rodolfo Cáliz. (2013). *“TUTORIAL ANALIZADOR DE PROTOCOLOS WIRESHARK”*. Perú. Universidad de los Andes.
- Rodríguez, Jhon Jairo.; Tangarife, Diego. y Cardona, Luis Fernando. (2008). *“Protocolos Industriales y de Telecontrol para Sistemas Distribuidos”*. Colombia. Universidad de San Buenaventura. Rodríguez.
- Samboni Núñez, Diana Marcela. (2012). *“Manual Básico de Wireshark”*. Medellín, Colombia. Universidad de Medellín.
- Schildt, Hervert. (2004). *“La Biblia Java 2 V5.0”*. Illinois, USA. Editorial Anaya Multimedia.
- Unifed Automation. (2016). [En Línea]. Recuperado de: <https://www.unified-automation.com/>
- Wikipedia (2016). *“OPC”*. [En Línea]. Recuperado de: <https://es.wikipedia.org/wiki/OPC>
- Wolfgang Mahnke y Stefan-Helmut Leitner. (2009). *“Arquitectura OPC Unificada”*. Ladenburg, Alemania. Revista ABB, 1, pp. 57-61.

ANEXOS

Árbol de archivos de la aplicación Software.

Paquete charts

- PanelChart.java

Paquete files

- Archivo.java
- Paquete.java

Paquete gráficos

- Diálogo1.java
- DiálogoCalcular.java
- VentanaInterna.java
- VentanaInternaCalcular.java
- VentanaInternaMostrar.java
- VentanaPrincipal.java

Paquete operaciones

- Operaciones.java

A.1- Código fuente archivo PanelChart.java

```
package charts;

import java.awt.BorderLayout;
import java.awt.GridLayout;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.data.category.DefaultCategoryDataset;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;

public class PanelChart
{

    public static ChartPanel crearGráfico(int[] paquetesF1,double[]
deltasF1,String nombreF1,int[]paquetesF2,double[]deltasF2,String nombreF2)
    {
        if (nombreF1.equals(nombreF2))//los nombres deben ser diferentes
            return null;
        if (paquetesF1.length!=deltasF1.length)//los tamaños deben ser
iguales
            return null;
        if (paquetesF2.length!=deltasF2.length)//los tamaños deben ser
iguales
            return null;

        XYSeries s1=new XYSeries(nombreF1);
```

```

        for (int i=0;i<paquetesF1.length;i++)
            s1.add(paquetesF1[i],deltasF1[i]);

        XYSeries s2=new XYSeries(nombreF2);
        for (int i=0;i<paquetesF2.length;i++)
            s2.add(paquetesF2[i],deltasF2[i]);

        XYSeriesCollection collection=new XYSeriesCollection();
        collection.addSeries(s1);
        collection.addSeries(s2);

        JFreeChart chart=ChartFactory.createXYLineChart("Gráfico tiempos",
"Paquete", "Delta", collection);
        ChartPanel chartPanel = new ChartPanel(chart);

        return chartPanel;
    }

    public static JPanel crearPanelDatos(String mínimo,String máximo,String
media,String desviación,String ip)
    {

        JLabel label1=new JLabel("mínimo: "+mínimo);
        JLabel label2=new JLabel("máximo: "+máximo);
        JLabel label3=new JLabel("media: "+media);
        JLabel label4=new JLabel("desviación: "+desviación);
        JPanel p=new JPanel();p.setLayout(new GridLayout(5,1));
        p.add(new
JLabel(ip));p.add(label1);p.add(label2);p.add(label3);p.add(label4);
        return p;
    }

    public static JPanel crearPanelGráfico(ChartPanel g,JPanel p,JPanel p2)
    {

        JPanel panel=new JPanel (new BorderLayout());
        panel.add(g,BorderLayout.CENTER);
        //panel.add(p,BorderLayout.SOUTH);

        JPanel south=new JPanel(new GridLayout(1,2));
        south.add(p);south.add(p2);
        panel.add(south,BorderLayout.SOUTH);

        return panel;
    }

    public static void main(String args[])
    {

        int[] p1={1,3,5,9};int[] p2={2,4,6,10};
        double[] d1={8,7,9,8};double[] d2={12,10,12,20};

        ChartPanel chartPanel=PanelChart.crearGráfico(p1, d1, "192.168.0.1",
p2,d2, "192.168.0.2");
        JPanel panelDatos=PanelChart.crearPanelDatos("0.01", "3", "1.5",
"1.2","192.168.1.1");
        JPanel panelDatos2=PanelChart.crearPanelDatos("2.01", "5", "3.5",
"3.2","192.168.1.5");

        JPanel panelGráfico=PanelChart.crearPanelGráfico(chartPanel,
panelDatos,panelDatos2);
        JFrame f=new
JFrame();f.add(panelGráfico);f.setVisible(true);f.setSize(500, 500);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}

```

A.2- Código fuente archivo Archivo.java

```
package files;
import java.io.IOException;
import java.util.List;
import java.util.Vector;

import org.jdom2.Document;
import org.jdom2.Element;
import org.jdom2.JDOMException;
import org.jdom2.input.SAXBuilder;

public class Archivo
{
    public Vector<Paquete> paquetes;
    public boolean esArchivoVálido;
    private Element raíz;
    public int tipo;

    @SuppressWarnings("Convert2Diamond")
    public Archivo(String path,int tipo)
    {
        if (tipo==Paquete.PROTOCOLO_IEC || tipo==Paquete.PROTOCOLO_OPC)
        {
            path="file:///"+path;
            SAXBuilder jdomBuilder;
            Document jdomDocument;
            try
            {
                jdomBuilder = new SAXBuilder();
                jdomDocument = jdomBuilder.build(path);
                raíz=jdomDocument.getRootElement();
                List<Element> packets=raíz.getChildren();
                int numPaquetes=packets.size();
                paquetes=new Vector<Paquete>(numPaquetes);

                for (int i=0;i<numPaquetes;i++)
                {
                    Paquete e=new
Paquete(raíz.getChildren().get(i),tipo);
                    paquetes.add(e);
                }
                this.esArchivoVálido=true;
                this.tipo=tipo;
                this.calcularDelta();//calcula las diferencias de
tiempo de los paquetes con ACK
            }
            catch(JDOMException | IOException
e){this.esArchivoVálido=false;System.out.println(e.getMessage());}
            finally{System.out.println("finally");}
        }
        else
        {
            this.tipo=Paquete.ERROR;
            this.esArchivoVálido=false;
        }
    }

    public void print(String separador)
    {
        if (this.esArchivoVálido==true)
        {
            for (int i=0;i<this.paquetes.size();i++)
                this.paquetes.get(i).print(separador);
        }
    }

    public String[][] getMatrizDatos()
    {
```



```

        if (this.esArchivoVálido==false || this.tipo==Paquete.ERROR)
            return null;
        int longitud;
        if (tipo==Paquete.PROTOCOLO_IEC)
            longitud=Paquete.ATRIBUTOS_MOSTRAR_IEC.length;
        else
            longitud=Paquete.ATRIBUTOS_MOSTRAR_OPC.length;

        int númeroPaquetes=raíz.getChildren().size();
        String[][] data=new String[númeroPaquetes][longitud];

        for (int fila=0;fila<númeroPaquetes;fila++)
        {
            Paquete e=this.paquetes.get(fila);
            System.arraycopy(e.atributos, 0, data[fila], 0, longitud);
        }
        return data;
    }

    public void calcularDelta()
    {
        if (this.esArchivoVálido==false)
            return;

        for (int i=1;i<this.paquetes.size();i++)
        {
            Paquete p=this.paquetes.get(i);
            String stringACK;
            int índice,índiceDelta;
            switch (this.tipo) {
                case Paquete.PROTOCOLO_IEC:
                    stringACK=p.atributos[Paquete.IEC_INDEX_ACK_FRAME];
                    índice=Paquete.IEC_INDEX_TIMESTAMP;
                    índiceDelta=Paquete.IEC_INDEX_DELTA;
                    break;
                case Paquete.PROTOCOLO_OPC:
                    stringACK=p.atributos[Paquete.OPC_INDEX_ACK_FRAME];
                    índice=Paquete.OPC_INDEX_TIMESTAMP;
                    índiceDelta=Paquete.OPC_INDEX_DELTA;
                    break;
                default:
                    stringACK="null";
                    índice=-1;
                    índiceDelta=-1;
                    break;
            }

            int índiceFrame=Archivo.buscarFrame(stringACK, 0, i,
            this.paquetes,tipo);
            if (índiceFrame>=0)
            {
                String
                tiempoCadena1=this.paquetes.get(índiceFrame).atributos[índice];
                Double d1=Double.valueOf(tiempoCadena1);

                String
                tiempoCadena2=this.paquetes.get(i).atributos[índice];
                Double d2=Double.valueOf(tiempoCadena2);
                if (d2!=null && d1!=null)
                {
                    double delta=d2-d1;
                    delta=delta*1000;//pasarlo a milisegundos
                    p.delta=delta;
                    p.atributos[índiceDelta]=String.valueOf(delta);
                }
            }
        }
    }

```

```
        }
    }
    private static int buscarFrame(String frame,int inicio,int
fin,Vector<Paquete> elementos,int tipo)
    {
        //busca en la lista el paquete que tenga el número de frame indicado
        //devuelve su posición en la lista si está y devuelve -1 si no se
encuentra
        //se busca desde inicio hasta final (no incluido)
        int índice;
        switch (tipo) {
            case Paquete.PROTOCOLO_IEC:
                índice=Paquete.IEC_INDEX_FRAME_NUMBER;
                break;
            case Paquete.PROTOCOLO_OPC:
                índice=Paquete.OPC_INDEX_FRAME_NUMBER;
                break;
            default:
                return -1;
        }

        if (frame!=null && elementos!=null && inicio>=0)
        {
            for (int i=inicio;i<fin && i<elementos.size() ;i++)
            {
                Paquete p=elementos.get(i);
                if (frame.equals(p.atributos[índice]))
                    return i;
            }
        }
        return -1;
    }

    public static void main(String[]args)
    {
        int opción=1;
        int tipo;
        String pathString;

        switch (opción) {
            //opc
            case 0:
                pathString="C:\\Users\\jesus\\resultados2\\r1";
                tipo=Paquete.PROTOCOLO_OPC;
                break;
            //iec
            case 1:
                pathString="C:\\Users\\jesus\\resultadosIEC\\prueba1b";
                tipo=Paquete.PROTOCOLO_IEC;
                break;
            default:
                pathString="C:\\Users\\jesus\\resultadosIEC\\prueba1b";
                tipo=3;
                break;
        }
        Archivo a=new Archivo(pathString,tipo);
        a.print("; ");

        a.calcularDelta();
        System.exit(0);
    }
}
```

A.3- Código fuente archivo Paquete.java

```

package charts;

import java.awt.BorderLayout;
import java.awt.GridLayout;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.data.category.DefaultCategoryDataset;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;

public class PanelChart
{
    public static ChartPanel crearGráfico(int[] paquetesF1,double[]
deltasF1,String nombreF1,int[]paquetesF2,double[]deltasF2,String nombreF2)
    {
        if (nombreF1.equals(nombreF2))//los nombres deben ser diferentes
            return null;
        if (paquetesF1.length!=deltasF1.length)//los tamaños deben ser
iguales
            return null;
        if (paquetesF2.length!=deltasF2.length)//los tamaños deben ser
iguales
            return null;

        XYSeries s1=new XYSeries(nombreF1);
        for (int i=0;i<paquetesF1.length;i++)
            s1.add(paquetesF1[i],deltasF1[i]);

        XYSeries s2=new XYSeries(nombreF2);
        for (int i=0;i<paquetesF2.length;i++)
            s2.add(paquetesF2[i],deltasF2[i]);

        XYSeriesCollection collection=new XYSeriesCollection();
        collection.addSeries(s1);
        collection.addSeries(s2);

        JFreeChart chart=ChartFactory.createXYLineChart("Gráfico tiempos",
"Paquete", "Delta", collection);
        ChartPanel chartPanel = new ChartPanel(chart);

        return chartPanel;
    }

    public static JPanel crearPanelDatos(String mínimo,String máximo,String
media,String desviación,String ip)
    {
        JLabel label1=new JLabel("mínimo: "+mínimo);
        JLabel label2=new JLabel("máximo: "+máximo);
        JLabel label3=new JLabel("media: "+media);
        JLabel label4=new JLabel("desviación: "+desviación);
        JPanel p=new JPanel();p.setLayout(new GridLayout(5,1));
        p.add(new
JLabel(ip));p.add(label1);p.add(label2);p.add(label3);p.add(label4);
        return p;
    }

    public static JPanel crearPanelGráfico(ChartPanel g,JPanel p,JPanel p2)

```

```
{
    JPanel panel=new JPanel (new BorderLayout());
    panel.add(g,BorderLayout.CENTER);
    //panel.add(p,BorderLayout.SOUTH);

    JPanel south=new JPanel(new GridLayout(1,2));
    south.add(p);south.add(p2);
    panel.add(south,BorderLayout.SOUTH);

    return panel;
}
public static void main(String args[])
{
    int[] p1={1,3,5,9};int[] p2={2,4,6,10};
    double[] d1={8,7,9,8};double[] d2={12,10,12,20};

    ChartPanel chartPanel=PanelChart.crearGráfico(p1, d1, "192.168.0.1",
p2,d2, "192.168.0.2");
    JPanel panelDatos=PanelChart.crearPanelDatos("0.01", "3", "1.5",
"1.2","192.168.1.1");
    JPanel panelDatos2=PanelChart.crearPanelDatos("2.01", "5", "3.5",
"3.2","192.168.1.5");

    JPanel panelGráfico=PanelChart.crearPanelGráfico(chartPanel,
panelDatos,panelDatos2);
    JFrame f=new
JFrame();f.add(panelGráfico);f.setVisible(true);f.setSize(500, 500);
    f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
}
```

A.4- Código fuente archivo Diálogo1.java

```

package gráficos;

import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.JTextField;

import files.Paquete;

public class Diálogo1 extends JDialog
{
    //es el diálogo que se usa para guardar y para mostrar
    boolean continuar=false;
    public PanelCentralDiálogo panelCentral;

    public String getHost(){return this.panelCentral.getHost();}
    public String getPuerto(){return this.panelCentral.getPuerto();}
    public String getUsuario(){return this.panelCentral.getUsuario();}
    public String getTabla(){return this.panelCentral.getTabla();}
    public String getBaseDeDatos(){return this.panelCentral.getBaseDeDatos();}
    public String getContraseña(){return this.panelCentral.getContraseña();}

    public Diálogo1(int tipo)
    {
        super();
        this.setModal(true);
        this.setLayout(new BorderLayout());
        this.panelCentral=new PanelCentralDiálogo(tipo);
        this.add(panelCentral, BorderLayout.CENTER);
        this.setSize(500,500);

        JButton botónContinuar=new JButton("Continuar");
        botónContinuar.addActionListener(new ContinuarListener());
        JButton botónCancelar=new JButton("Cancelar");
        botónCancelar.addActionListener(new CancelarListener());
        JPanel panelBotones=new JPanel();
        panelBotones.add(botónContinuar);
        panelBotones.add(botónCancelar);
        this.add(panelBotones, BorderLayout.PAGE_END);
    }
    public class PanelCentralDiálogo extends JPanel
    {
        JTextField host,puerto,usuario,tabla,baseDeDatos;
        JPasswordField contraseña;
        public String getHost(){return this.host.getText();}
        public String getPuerto(){return this.puerto.getText();}
        public String getUsuario(){return this.usuario.getText();}
        public String getTabla(){return this.tabla.getText();}
        public String getBaseDeDatos(){return this.baseDeDatos.getText();}
        public String getContraseña()
        {
            char[] c=this.contraseña.getPassword();
            return new String(c);
        }

        public PanelCentralDiálogo(int tipo)
        {

```

```
super();
String nombreTabla;
switch (tipo) {
    case Paquete.PROTOCOLO_IEC:
        nombreTabla=VentanaPrincipal.TABLA_IEC;
        break;
    case Paquete.PROTOCOLO_OPC:
        nombreTabla=VentanaPrincipal.TABLA_OPCUA;
        break;
    default:
        nombreTabla="";
        break;
}
GridLayout layout1=new GridLayout(7,2);
this.setLayout(layout1);
JLabel hostLabel=new JLabel("Host");
JLabel puertoLabel=new JLabel("Puerto");
JLabel usuarioLabel=new JLabel("Usuario");
JLabel tablaLabel=new JLabel("Tabla");
JLabel contraseñaLabel=new JLabel("Contraseña");
JLabel baseLabel=new JLabel("Base de datos");

host=new JTextField("Introduzca ip de host",20);
puerto=new JTextField("Introduzca puerto de servidor",20);
usuario=new JTextField("Introduzca su usuario",20);
tabla=new JTextField("Introduzca la tabla",20);
contraseña=new JPasswordField(20);
baseDeDatos=new JTextField("Introduzca la base de datos");

host=new JTextField("localhost",20);
puerto=new JTextField("3306",20);
usuario=new JTextField("root",20);
tabla=new JTextField(nombreTabla,20);
contraseña=new JPasswordField(20);
baseDeDatos=new JTextField("mydb");

this.add(hostLabel);this.add(host);
this.add(puertoLabel);this.add(puerto);
this.add(usuarioLabel);this.add(usuario);
this.add(baseLabel);this.add(baseDeDatos);
this.add(tablaLabel);this.add(tabla);
this.add(contraseñaLabel);this.add(contraseña);
}
}

public class ContinuarListener implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
        continuar=true;
        JButton b=(JButton)e.getSource();
        Diálogo1 diálogo=(Diálogo1)b.getTopLevelAncestor();
        diálogo.dispose();
    }
}

private class CancelarListener implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
        continuar=false;
        JButton b=(JButton)e.getSource();
        Diálogo1 diálogo=(Diálogo1)b.getTopLevelAncestor();
        diálogo.dispose();
    }
}
}}}
```

A.5- Código fuente archivo DiálogoCalcular.java

```

package gráficos;

import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.JTextField;

import files.Paquete;

public class DiálogoCalcular extends JDialog
{
    boolean continuar=false;
    public PanelCentralDiálogo panelCentral;

    public String getHost(){return this.panelCentral.getHost();}
    public String getPuerto(){return this.panelCentral.getPuerto();}
    public String getUsuario(){return this.panelCentral.getUsuario();}
    public String getTabla(){return this.panelCentral.getTabla();}
    public String getBaseDeDatos(){return this.panelCentral.getBaseDeDatos();}
    public String getContraseña(){return this.panelCentral.getContraseña();}
    public String getIP1(){return this.panelCentral.getIP1();}
    public String getIP2(){return this.panelCentral.getIP2();}

    public DiálogoCalcular(int tipo)
    {
        super();
        this.setModal(true);
        this.setLayout(new BorderLayout());
        this.panelCentral=new PanelCentralDiálogo(tipo);
        this.add(panelCentral, BorderLayout.CENTER);
        this.setSize(500,500);

        JButton botónContinuar=new JButton("Continuar");
        botónContinuar.addActionListener(new ContinuarListener());
        JButton botónCancelar=new JButton("Cancelar");
        botónCancelar.addActionListener(new CancelarListener());
        JPanel panelBotones=new JPanel();
        panelBotones.add(botónContinuar);
        panelBotones.add(botónCancelar);
        this.add(panelBotones, BorderLayout.PAGE_END);
    }
    public class PanelCentralDiálogo extends JPanel
    {
        JTextField host,puerto,usuario,tabela,baseDeDatos,ip1,ip2;
        JPasswordField contraseña;
        public String getHost(){return this.host.getText();}
        public String getPuerto(){return this.puerto.getText();}
        public String getUsuario(){return this.usuario.getText();}
        public String getTabla(){return this.tabela.getText();}
        public String getBaseDeDatos(){return this.baseDeDatos.getText();}
        public String getIP1(){return this.ip1.getText();}
        public String getIP2(){return this.ip2.getText();}

        public String getContraseña()
        {
            char[] c=this.contraseña.getPassword();
            return new String(c);
        }
    }
}

```

```
    }

    public PanelCentralDiálogo(int tipo)
    {
        super();
        String nombreTabla;
        switch (tipo) {
            case Paquete.PROTOCOLO_IEC:
                nombreTabla=VentanaPrincipal.TABLA_IEC;
                break;
            case Paquete.PROTOCOLO_OPC:
                nombreTabla=VentanaPrincipal.TABLA_OPCUA;
                break;
            default:
                nombreTabla=""; //nombreTabla="paquetes";
                break;
        }

        GridLayout layout1=new GridLayout(9,2);
        this.setLayout(layout1);
        JLabel hostLabel=new JLabel("Host");
        JLabel puertoLabel=new JLabel("Puerto");
        JLabel usuarioLabel=new JLabel("Usuario");
        JLabel tablaLabel=new JLabel("Tabla");
        JLabel contraseñaLabel=new JLabel("Contraseña");
        JLabel baseLabel=new JLabel("Base de datos");
        JLabel ip1Label=new JLabel("IP fuente");
        JLabel ip2Label=new JLabel("IP destino");

        host=new JTextField("Introduzca ip de host",20);
        puerto=new JTextField("Introduzca puerto de servidor",20);
        usuario=new JTextField("Introduzca su usuario",20);
        tabla=new JTextField("Introduzca la tabla",20);
        contraseña=new JPasswordField(20);
        baseDeDatos=new JTextField("Introduzca la base de datos");

        host=new JTextField("localhost",20);
        puerto=new JTextField("3306",20);
        usuario=new JTextField("root",20);
        tabla=new JTextField(nombreTabla,20);
        contraseña=new JPasswordField(20);
        baseDeDatos=new JTextField("mydb");
        ip1=new JTextField("127.0.0.1");
        ip2=new JTextField("127.0.0.1");

        this.add(hostLabel);this.add(host);
        this.add(puertoLabel);this.add(puerto);
        this.add(usuarioLabel);this.add(usuario);
        this.add(baseLabel);this.add(baseDeDatos);
        this.add(tablaLabel);this.add(tabla);
        this.add(ip1Label);this.add(ip1);
        this.add(ip2Label);this.add(ip2);
        this.add(contraseñaLabel);this.add(contraseña);
    }
}

public class ContinuarListener implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
        continuar=true;
        JButton b=(JButton)e.getSource();
        DiálogoCalcular
diálogo=(DiálogoCalcular)b.getTopLevelAncestor();
        diálogo.dispose();
    }
}
```



```
private class CancelarListener implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
        continuar=false;
        JButton b=(JButton)e.getSource();
        DiálogoCalcular
diálogo=(DiálogoCalcular)b.getTopLevelAncestor();
        diálogo.dispose();
    }
}
```

A.6- Código fuente archivo VentanaInterna.java

```
package gráficos;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JFrame;
import javax.swing.JInternalFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JTable;

import files.Archivo;
import files.Paquete;
import operaciones.Operaciones;

public class VentanaInterna extends JInternalFrame
{
    Archivo archivo;
    int tipo;

    public VentanaInterna(String nombre ,boolean b1,boolean b2,boolean
b3,boolean b4,Archivo archivo,int tipo)
    {
        super(nombre,b1,b2,b3,b4);
        this.tipo=tipo;
        this.archivo=archivo;
        String[][]data=archivo.getMatrizDatos();
        String[]names;
        switch (tipo) {
            case Paquete.PROTOCOLO_IEC:
                names=Paquete.ATRIBUTOS_MOSTRAR_IEC;
                break;
            case Paquete.PROTOCOLO_OPC:
                names=Paquete.ATRIBUTOS_MOSTRAR_OPC;
                break;
            default:
                names=new String[0];
                break;
        }
        JTable tabla=new JTable(data,names);
        JMenuBar barraMenú=new JMenuBar();
        JMenu menuArchivo=new JMenu("Archivo");
        JMenuItem archivoGuardar=new JMenuItem("Guardar");
        archivoGuardar.addActionListener(new GuardarListener());
        menuArchivo.add(archivoGuardar);
        barraMenú.add(menuArchivo);
        this.setJMenuBar(barraMenú);
        this.setSize(500,500);
        this.setVisible(true);
        JScrollPane p=new JScrollPane(tabla);
        this.add(p);
    }

    private class GuardarListener implements ActionListener
    {
        public void actionPerformed(ActionEvent event1)
        {
            //JOptionPane.showMessageDialog(desktop, "No se ha
implementado","Error",JOptionPane.WARNING_MESSAGE);
            //se crea un diálogo para obtener los datos de la conexión
            Diálogo1 diálogo=new Diálogo1(tipo);
            if (tipo==Paquete.PROTOCOLO_IEC)
                diálogo.setTitle("Guardar IEC");
        }
    }
}
```

```
else
    diálogo.setTitle("Guardar OPCUA");
diálogo.setVisible(true);
boolean opción=diálogo.continuar;

if (opción==true)//si se presiona continuar en el diálogo
{//base de datos
    diálogo.continuar=false;//se vuelve al estado inicial
para la próxima vez que se llame

        //se obtiene la información de la interfaz gráfica
        //estas cadenas se usan para la conexión a la base de
datos

        String hostCadena=diálogo.getHost();
        String puertoCadena=diálogo.getPuerto();
        String tabla=diálogo.getTabla();
        String usuario=diálogo.getUsuario();
        String contraseña=diálogo.getContraseña();
        String baseDatosCadena=diálogo.getBaseDeDatos();

        //se realiza la conexión
        Object o=evento1.getSource();
        System.out.println(o.getClass());
        System.out.println(o.hashCode());

        String
cadenaConexión="jdbc:mysql://" +hostCadena+": "+puertoCadena+"/"+baseDatosCadena;
        String[]names;
        if (tipo==Paquete.PROTOCOLO_IEC)
            names=Paquete.ATRIBUTOS_BUSCAR_IEC;
        else if (tipo==Paquete.PROTOCOLO_IEC)
            names=Paquete.ATRIBUTOS_BUSCAR_OPC;
        else
            names=new String[0];

        String
error=Operaciones.guardarEnBaseDeDatos(cadenaConexión, tabla, usuario,
contraseña, archivo, names,tipo);
        if (error.length()==0)
        {
            JOptionPane.showMessageDialog(new JFrame(), "Se
han guardado los datos", "BBDD",JOptionPane.INFORMATION_MESSAGE);
                //no error
            }
            else
            {
                JOptionPane.showMessageDialog(new JFrame(),
error, "BBDD",JOptionPane.WARNING_MESSAGE);
                    //error
            }
        }
    }
}
```

A.7- Código fuente archivo VentanaInternaCalcular.java

```
package gráficos;

import javax.swing.JInternalFrame;
import javax.swing.JPanel;

import org.jfree.chart.ChartPanel;

import charts.PanelChart;
import operaciones.Operaciones;

public class VentanaInternaCalcular extends JInternalFrame
{
    public VentanaInternaCalcular(String nombre,int[] p1, double[]d1,String
s1,int[]p2,double[]d2,String s2)
    {
        //p1,p2: vectores con la información de los paquetes; d1,d2:datos de
la tabla; s1,s2:cadenas con los nombres;
        super(nombre,true,true,true,true);
        ChartPanel chartPanel=PanelChart.crearGráfico(p1, d1, s1, p2,d2,
s2);
        int índice=Operaciones.encontrarMáximo(d1);
        String máx1= (índice>=0) ? String.valueOf(d1[índice]) : "";
        índice=Operaciones.encontrarMínimo(d1);
        String mín1=(índice>=0) ? String.valueOf(d1[índice]) : "";
        String desv1=(índice>=0) ?
String.valueOf(Operaciones.desviaciónEstándar(d1)) : "";
        String media1=(índice>=0) ? String.valueOf(Operaciones.media(d1)) :
"";

        índice=Operaciones.encontrarMáximo(d2);
        String máx2= (índice>=0) ? String.valueOf(d2[índice]) : "";
        índice=Operaciones.encontrarMínimo(d2);
        String mín2=(índice>=0) ? String.valueOf(d2[índice]) : "";
        String desv2=(índice>=0) ?
String.valueOf(Operaciones.desviaciónEstándar(d2)) : "";
        String media2=(índice>=0) ? String.valueOf(Operaciones.media(d2)) :
"";

        JPanel panelDatos1=PanelChart.crearPanelDatos(mín1, máx1, media1,
desv1,s1);
        JPanel panelDatos2=PanelChart.crearPanelDatos(mín2, máx2, media2,
desv2,s2);

        JPanel panelGráfico1=PanelChart.crearPanelGráfico(chartPanel,
panelDatos1,panelDatos2);
        this.add(panelGráfico1);
        this.setSize(500,500);
        this.setVisible(true);
    }
}
```

A.8- Código fuente archivo VentanaInternaMostrar.java

```
package gráficos;

import javax.swing.JInternalFrame;
import javax.swing.JScrollPane;
import javax.swing.JTable;

public class VentanaInternaMostrar extends JInternalFrame
{
    public VentanaInternaMostrar(String nombre, Object[][] datos,
String[] nombresColumnas)
    {
        super(nombre, true, true, true, true);
        JTable tabla=new JTable(datos, nombresColumnas);
        this.setSize(500,500);
        this.setVisible(true);
        JScrollPane p=new JScrollPane(tabla);
        this.add(p);
    }
}
```

A.9- Código fuente archivo VentanaInternaMostrar.java

```
package gráficos;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.util.Vector;

import javax.swing.JDesktopPane;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;

import files.Archivo;
import operaciones.Operaciones;

public class VentanaPrincipal extends JFrame
{
    public static final String NOMBRE_BOTÓN_ABRIR_OPCUA="Abrir OPCUA";
    public static final String NOMBRE_BOTÓN_ABRIR_IEC="Abrir IEC";
    public static final String NOMBRE_BOTÓN_MOSTRAR_OPCUA="Mostrar OPCUA";
    public static final String NOMBRE_BOTÓN_MOSTRAR_IEC="Mostrar IEC";
    public static final String NOMBRE_BOTÓN_MOSTRAR_OPCUA_ACK="Mostrar OPCUA
con ACK";
    public static final String NOMBRE_BOTÓN_MOSTRAR_IEC_ACK="Mostrar IEC con
ACK";
    public static final String NOMBRE_BOTÓN_CALCULAR_OPCUA="Calcular
diferencias de tiempo OPCUA";
    public static final String NOMBRE_BOTÓN_CALCULAR_IEC="Calcular diferencias
de tiempo IEC";

    public static final String TABLA_OPCUA="tabla_opcua";
    public static final String TABLA_IEC="tabla_iec";

    public JDesktopPane desktop;

    public VentanaPrincipal()
    {
        super("Procesador Paquetes");

        JMenuBar barraMenú=new JMenuBar();
        JMenu menuArchivo=new JMenu("Archivo");
        JMenuItem archivoAbrirOPCUA=new
JMenuItem(VentanaPrincipal.NOMBRE_BOTÓN_ABRIR_OPCUA);
        archivoAbrirOPCUA.addActionListener(new AbrirListener());
        menuArchivo.add(archivoAbrirOPCUA);
        JMenuItem archivoAbrirOtro=new
JMenuItem(VentanaPrincipal.NOMBRE_BOTÓN_ABRIR_IEC);
        archivoAbrirOtro.addActionListener(new AbrirListener());
        menuArchivo.add(archivoAbrirOtro);
        barraMenú.add(menuArchivo);
        JMenu menuMostrar=new JMenu("Mostrar");
        JMenuItem itemMostrarTablaOPCUA=new
JMenuItem(VentanaPrincipal.NOMBRE_BOTÓN_MOSTRAR_OPCUA);
        itemMostrarTablaOPCUA.addActionListener(new MostrarTablaListener());
        menuMostrar.add(itemMostrarTablaOPCUA);
        JMenuItem itemMostrarTablaIEC=new
JMenuItem(VentanaPrincipal.NOMBRE_BOTÓN_MOSTRAR_IEC);
        itemMostrarTablaIEC.addActionListener(new MostrarTablaListener());
        menuMostrar.add(itemMostrarTablaIEC);
        JMenuItem itemMostrarTablaOPCUAACK=new
JMenuItem(VentanaPrincipal.NOMBRE_BOTÓN_MOSTRAR_OPCUA_ACK);
```

```

        itemMostrarTablaOPCUAACK.addActionListener(new
MostrarTablaListenerACK());
        menuMostrar.add(itemMostrarTablaOPCUAACK);
        JMenuItem itemMostrarTablaIECACK=new
JMenuItem(VentanaPrincipal.NOMBRE_BOTÓN_MOSTRAR_IEC_ACK);
        itemMostrarTablaIECACK.addActionListener(new
MostrarTablaListenerACK());
        menuMostrar.add(itemMostrarTablaIECACK);

        barraMenú.add(menuMostrar);

        JMenu menuCalcular=new JMenu("Calcular");
        JMenuItem itemCalcularOPCUA=new
JMenuItem(VentanaPrincipal.NOMBRE_BOTÓN_CALCULAR_OPCUA);
        itemCalcularOPCUA.addActionListener(new CalcularListener());
        menuCalcular.add(itemCalcularOPCUA);
        JMenuItem itemCalcularIEC=new
JMenuItem(VentanaPrincipal.NOMBRE_BOTÓN_CALCULAR_IEC);
        itemCalcularIEC.addActionListener(new CalcularListener());
        menuCalcular.add(itemCalcularIEC);
        barraMenú.add(menuCalcular);

        this.setJMenuBar(barraMenú);

        desktop = new JDesktopPane();
        this.setContentPane(desktop);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setExtendedState(JFrame.MAXIMIZED_BOTH);
        setSize(500,500);
    }

    public static File seleccionarArchivoYDevolverArchivo(String nombre)
    {
        JFileChooser fc=new JFileChooser();
        fc.setDialogTitle(nombre);
        fc.showOpenDialog(new JFrame());
        File archivoSeleccionado=fc.getSelectedFile();
        return archivoSeleccionado;
    }

    public class AbrirListener implements ActionListener
    {
        Archivo a;

        public void actionPerformed(ActionEvent event1)
        {
            JMenuItem m=(JMenuItem)event1.getSource();
            String nombreBotón=m.getText();

            int tipo=Operaciones.getTipo(nombreBotón);

            File
file=VentanaPrincipal.seleccionarArchivoYDevolverArchivo(nombreBotón);
            if (file!=null)
            {
                a=new Archivo(file.getAbsolutePath(),tipo);
                if (a.esArchivoVálido)
                {
                    VentanaInterna v=new
VentanaInterna(file.getName(),true,true,true,true,a,tipo);
                    desktop.add(v);
                }
            }
            else
            {

```

```
        JOptionPane.showMessageDialog(desktop, "El
archivo no es válido", "Error", JOptionPane.ERROR_MESSAGE);
    }
    else
    {
        JOptionPane.showMessageDialog(desktop, "No se ha
elegido ningún archivo", "No archivo", JOptionPane.INFORMATION_MESSAGE);
    }
}

public class MostrarTablaListener implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
        //obtener qué botón se pulsó para saber el tipo
        JMenuItem m=(JMenuItem)e.getSource();
        String nombreBotón=m.getText();
        int tipo=Operaciones.getTipo(nombreBotón);

        //mostrar el diálogo para rellenar los datos de nombre de
        tabla, ip, etcétera
        Diálogo1 diálogo=new Diálogo1(tipo);
        diálogo.setTitle(nombreBotón);
        diálogo.setVisible(true);
        boolean opción=diálogo.continuar;

        if (opción==true)//si se presiona continuar en el diálogo
        {
            diálogo.continuar=false;//se vuelve al estado inicial
            para la próxima vez que se llame

            //se obtiene la información de la interfaz gráfica.
            estas cadenas se usan para la conexión a la base de datos
            String hostCadena=diálogo.getHost();
            String puertoCadena=diálogo.getPuerto();
            String tabla=diálogo.getTabla();
            String usuario=diálogo.getUsuario();
            String contraseña=diálogo.getContraseña();
            String baseDatosCadena=diálogo.getBaseDeDatos();

            //se realiza la conexión y se lee de la base de datos
            String
            cadenaConexión="jdbc:mysql://" + hostCadena + ":" + puertoCadena + "/" + baseDatosCadena;

            Vector<Object>v=Operaciones.leerTodoDeBaseDeDatos(cadenaConexión, tabla,
            usuario, contraseña);

            String error=Operaciones.getErrorLectura(v);
            if(error.equals(Operaciones.NO_HAY_ERROR))
            {

                String[] nombres=Operaciones.getNombresBaseDeDatos(v);
                Object[][]
                datos=Operaciones.getDatosVectorBaseDeDatos(v, nombres.length);
                VentanaInternaMostrar ventanaInternaMostrar=new
                VentanaInternaMostrar(tabla, datos, nombres);
                desktop.add(ventanaInternaMostrar);

            }
            else
            {
                JOptionPane.showMessageDialog(new JFrame(),
                v.get(1), "BBDD", JOptionPane.WARNING_MESSAGE);
            }
        }
    }
}
```



```

public class MostrarTablaListenerACK implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
        //obtener qué botón se pulsó para saber el tipo
        JMenuItem m=(JMenuItem)e.getSource();
        String nombreBotón=m.getText();
        int tipo=Operaciones.getTipo(nombreBotón);

        //mostrar el diálogo para rellenar los datos de nombre de
        tabla, ip, etcétera
        Diálogo1 diálogo=new Diálogo1(tipo);
        diálogo.setTitle(nombreBotón);
        diálogo.setVisible(true);
        boolean opción=diálogo.continuar;

        if (opción==true)//si se presiona continuar en el diálogo
        {
            diálogo.continuar=false;//se vuelve al estado inicial
            para la próxima vez que se llame (por si acaso)

            //se obtiene la información de la interfaz gráfica.
            estas cadenas se usan para la conexión a la base de datos
            String hostCadena=diálogo.getHost();
            String puertoCadena=diálogo.getPuerto();
            String tabla=diálogo.getTabla();
            String usuario=diálogo.getUser();
            String contraseña=diálogo.getContraseña();
            String baseDatosCadena=diálogo.getBaseDeDatos();

            //se realiza la conexión y se lee de la base de datos
            String
            cadenaConexión="jdbc:mysql://" + hostCadena + ":" + puertoCadena + "/" + baseDatosCadena;

            Vector<Object>v=Operaciones.leerTodoDeBaseDeDatosACK(cadenaConexión,
            tabla, usuario, contraseña);
            String error=Operaciones.getErrorLectura(v);
            if (error.equals(Operaciones.NO_HAY_ERROR))
            {

                String[]nombres=Operaciones.getNombresBaseDeDatos(v);
                Object[][]
                datos=Operaciones.getDatosVectorBaseDeDatos(v, nombres.length);
                VentanaInternaMostrar ventanaInternaMostrar=new
                VentanaInternaMostrar(tabla, datos, nombres);
                desktop.add(ventanaInternaMostrar);

            }
            else
            {
                JOptionPane.showMessageDialog(new JFrame(),
                v.get(1), "BBDD", JOptionPane.WARNING_MESSAGE);
            }
        }
    }
}

public class CalcularListener implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
        //obtener qué botón se pulsó para saber el tipo
        JMenuItem m=(JMenuItem)e.getSource();
        String nombreBotón=m.getText();
        int tipo=Operaciones.getTipo(nombreBotón);

```

```
//mostrar el diálogo para rellenar los datos de nombre de
tabla, ip, etcétera
DiálogoCalcular diálogo=new DiálogoCalcular(tipo);
diálogo.setTitle(nombreBotón);
diálogo.setVisible(true);
boolean opción=diálogo.continuar;

if (opción==true)//si se presiona continuar en el diálogo
{
    diálogo.continuar=false;//se vuelve al estado inicial
para la próxima vez que se llame

    //se obtiene la información de la interfaz gráfica.
estas cadenas se usan para la conexión a la base de datos
String hostCadena=diálogo.getHost();
String puertoCadena=diálogo.getPuerto();
String tabla=diálogo.getTabla();
String usuario=diálogo.getUsuario();
String contraseña=diálogo.getContraseña();
String baseDatosCadena=diálogo.getBaseDeDatos();
String ip1=diálogo.getIP1();
String ip2=diálogo.getIP2();

    //se realiza la conexión y se lee de la base de datos
String
cadenaConexión="jdbc:mysql://" + hostCadena + ":" + puertoCadena + "/" + baseDatosCadena;

    Vector<Object>v1=Operaciones.leerDeBaseDeDatos(cadenaConexión, tabla,
usuario, contraseña, ip1, ip2, Operaciones.LEER_TODOS);

    Vector<Object>v2=Operaciones.leerDeBaseDeDatos(cadenaConexión, tabla,
usuario, contraseña, ip2, ip1, Operaciones.LEER_TODOS);
    String error1=Operaciones.getErrorLectura(v1);
    String error2=Operaciones.getErrorLectura(v2);
    if
(error1.equals(Operaciones.NO_HAY_ERROR) && error2.equals(Operaciones.NO_HAY_ERROR)
)
    {
        double[]datos1=Operaciones.getDelta(v1);
        int[]paquetes1=Operaciones.getPaquetes(v1);
        double[]datos2=Operaciones.getDelta(v2);
        int[]paquetes2=Operaciones.getPaquetes(v2);
        VentanaInternaCalcular ventanaInterna=new
VentanaInternaCalcular(tabla,paquetes1, datos1,"eq1:"+ip1,paquetes2,
datos2,"eq2:"+ip2);

        desktop.add(ventanaInterna);

    }
    else
    {
        JOptionPane.showMessageDialog(new JFrame(),
v1.get(1),"BBDD",JOptionPane.WARNING_MESSAGE);
    }
}

}

public class Listener implements ActionListener
{
    //un listener para todos los botones. no sé si es buena idea
    public void actionPerformed(ActionEvent e)
    {
        JMenuItem m=(JMenuItem)e.getSource();
        String nombreBotón=m.getText();
        switch (nombreBotón) {
            case VentanaPrincipal.NOMBRE_BOTÓN_ABRIR_IEC:
```

```
                JOptionPane.showMessageDialog(new JFrame(),
nombreBotón,"1",JOptionPane.WARNING_MESSAGE);
                break;
                case VentanaPrincipal.NOMBRE_BOTÓN_MOSTRAR_IEC:
                JOptionPane.showMessageDialog(new JFrame(),
nombreBotón,"3",JOptionPane.WARNING_MESSAGE);
                break;
                case VentanaPrincipal.NOMBRE_BOTÓN_CALCULAR_IEC:
                JOptionPane.showMessageDialog(new JFrame(),
nombreBotón,"4",JOptionPane.WARNING_MESSAGE);
                break;
                case VentanaPrincipal.NOMBRE_BOTÓN_MOSTRAR_IEC_ACK:
                JOptionPane.showMessageDialog(new JFrame(),
nombreBotón,"5",JOptionPane.WARNING_MESSAGE);
                break;
                case VentanaPrincipal.NOMBRE_BOTÓN_ABRIR_OPCUA:
                JOptionPane.showMessageDialog(new JFrame(),
nombreBotón,"7",JOptionPane.WARNING_MESSAGE);
                break;
                case VentanaPrincipal.NOMBRE_BOTÓN_MOSTRAR_OPCUA:
                JOptionPane.showMessageDialog(new JFrame(),
nombreBotón,"8",JOptionPane.WARNING_MESSAGE);
                break;
                case VentanaPrincipal.NOMBRE_BOTÓN_CALCULAR_OPCUA:
                JOptionPane.showMessageDialog(new JFrame(),
nombreBotón,"9",JOptionPane.WARNING_MESSAGE);
                break;
                case VentanaPrincipal.NOMBRE_BOTÓN_MOSTRAR_OPCUA_ACK:
                JOptionPane.showMessageDialog(new JFrame(),
nombreBotón,"10",JOptionPane.WARNING_MESSAGE);
                break;
                default:
                break;
            }
        }
    }

    public static void main(String[]args)
    {
        VentanaPrincipal v=new VentanaPrincipal();
        v.setVisible(true);
    }
}
```

A.10- Código fuente archivo Operaciones.java

```
package operaciones;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.Statement;
import java.util.Vector;

import files.Archivo;
import files.Paquete;
import gráficos.VentanaPrincipal;
import java.sql.SQLException;

public class Operaciones
{
    public static final int LEER_DELTA=0;
    public static final int LEER_TODOS=1;
    public static final String NO_HAY_ERROR="Se hizo la lectura";

    public static double media(double[]v)
    {
        double sum = 0.0;
        for(double a : v)
            sum += a;
        return sum/v.length;
    }
    public static double varianza(double[]v)
    {
        double mean = Operaciones.media(v);
        double temp = 0;
        for(double a :v)
            temp += (mean-a)*(mean-a);
        return temp/v.length;
    }

    public static double desviaciónEstándar(double[]v)
    {
        return Math.sqrt(Operaciones.varianza(v));
    }
    public static int encontrarMínimo(double[]v)
    {
        if (v==null || v.length<=0)
            return -1;
        int posiciónMínimo=0;
        double valorMínimo=v[0];
        for(int i=1;i<v.length;i++)
        {
            if (v[i]<valorMínimo)
            {
                posiciónMínimo=i;
                valorMínimo=v[i];
            }
        }
        return posiciónMínimo;
    }

    public static int encontrarMáximo(double[]v)
    {
        if (v==null || v.length<=0)
            return -1;
        int posiciónMáximo=0;
        double valorMáximo=v[0];
        for(int i=1;i<v.length;i++)
        {
```

```

        if (v[i]>valorMáximo)
        {
            posiciónMáximo=i;
            valorMáximo=v[i];
        }
    }
    return posiciónMáximo;
}
public static int getTipo(String nombreBotón)
{
    int tipo=Paquete.ERROR;

    if (nombreBotón.equals (VentanaPrincipal.NOMBRE_BOTÓN_ABRIR_IEC)

||nombreBotón.equals (VentanaPrincipal.NOMBRE_BOTÓN_MOSTRAR_IEC)

||nombreBotón.equals (VentanaPrincipal.NOMBRE_BOTÓN_MOSTRAR_IEC_ACK)

||nombreBotón.equals (VentanaPrincipal.NOMBRE_BOTÓN_CALCULAR_IEC))
        tipo=Paquete.PROTOCOLO_IEC;
    else if
(nombreBotón.equals (VentanaPrincipal.NOMBRE_BOTÓN_ABRIR_OPCUA)

||nombreBotón.equals (VentanaPrincipal.NOMBRE_BOTÓN_MOSTRAR_OPCUA)

||nombreBotón.equals (VentanaPrincipal.NOMBRE_BOTÓN_MOSTRAR_OPCUA_ACK)

||nombreBotón.equals (VentanaPrincipal.NOMBRE_BOTÓN_CALCULAR_OPCUA))
        tipo=Paquete.PROTOCOLO_OPC;

    return tipo;
}

public static String[] getNombresBaseDeDatos (Vector<Object>v)
{
    return (String[]) v.get(0);
}

public static Vector<Object> leerTodoDeBaseDeDatos (String
cadenaConexión,String tabla, String usuario,String contraseña)
{
    //el vector que se devuelve tiene:
    //      - si hay error: nulo en la posición 0 y el mensaje en
la posición 1;
    //      - si no hay error: vector con nombres en la posición 0
y un vector con los datos de cada fila en cada una de las siguientes columnas

    String error="";
    Vector<Object> datos=new Vector<>();
    Connection conexión=null;
    try
    {
        Statement sentencia;
        Class.forName ("com.mysql.jdbc.Driver");

        conexión=DriverManager.getConnection (cadenaConexión,usuario,contraseña);

        sentencia=conexión.createStatement();//se crea una sentencia
mysql

        String sentenciaLeer="select * from "+tabla;//

        ResultSet rs = null;
        rs=sentencia.executeQuery (sentenciaLeer);
        ResultSetMetaData metaData=rs.getMetaData();
        int númeroDeColumnas=metaData.getColumnCount();

        String[]nombres=new String[númeroDeColumnas];

```

```
for (int i=0;i<númeroDeColumnas;i++)
{
    nombres[i]=metaData.getColumnName(i+1);
}
datos.add(nombres);

while (rs.next())
{
    Object[]v=new Object[númeroDeColumnas];
    for (int i=0;i<númeroDeColumnas;i++)
    {
        v[i]=rs.getObject(i+1);
    }
    datos.add(v);
}
rs.close();
}
catch (Exception e){error=e.getMessage();}
finally
{
    if (error.length()==0)
    {
        try {conexión.close();}
        catch (Exception e){error=e.getMessage();}
    }
    if (error.length(>0)//hay error
    {
        datos.clear();
        datos.add(0,null);
        datos.add(1, error);
    }
    datos.trimToSize();
    return datos;
}

public static Vector<Object> leerTodoDeBaseDeDatosACK(String
cadenaConexión,String tabla, String usuario,String contraseña)
{
    //el vector que se devuelve tiene:
    //      - si hay error: nulo en la posición 0 y el mensaje en
la posición 1;
    //      - si no hay error: vector con nombres en la posición 0
y un vector con los datos de cada fila en cada una de las siguientes columnas

    String error="";
    Vector<Object> datos=new Vector<>();
    Connection conexión=null;
    try
    {
        Statement sentencia;
        Class.forName("com.mysql.jdbc.Driver");

        conexión=DriverManager.getConnection(cadenaConexión,usuario,contraseña);

        sentencia=conexión.createStatement();//se crea una sentencia
mysql
        String sentenciaLeer="select * from "+tabla+" where
ack_de_frame!=\"null\"";//
        ResultSet rs = null;
        rs=sentencia.executeQuery(sentenciaLeer);
        ResultSetMetaData metaData=rs.getMetaData();
        int númeroDeColumnas=metaData.getColumnCount();

        String[]nombres=new String[númeroDeColumnas];
```

```

        for (int i=0;i<númeroDeColumnas;i++)
        {
            nombres[i]=metaData.getColumnName(i+1);
        }
        datos.add(nombres);

        while (rs.next())
        {
            Object[]v=new Object[númeroDeColumnas];
            for (int i=0;i<númeroDeColumnas;i++)
            {
                v[i]=rs.getObject(i+1);
            }
            datos.add(v);
        }
        rs.close();
    }
    catch (ClassNotFoundException | SQLException
e){error=e.getMessage();}
    finally
    {
        if (error.length()==0)
        {
            try {conexión.close();}
            catch (Exception e){error=e.getMessage();}
        }
        if (error.length(>0)//hay error
        {
            datos.clear();
            datos.add(0,null);
            datos.add(1, error);
        }
        datos.trimToSize();
        return datos;
    }

    public static Vector<Object> leerDeBaseDeDatos(String
cadenaConexión,String tabla, String usuario,String contraseña,String ipSrc,String
ipDst,int opción)
    {
        //el vector que se devuelve tiene:
        //          - si hay error: nulo en la posición 0 y el mensaje en
la posición 1;
        //          - si no hay error: vector con nombres en la posición 0
y un vector con los datos de cada fila en cada una de las siguientes columnas

        //SELECT field1, field2,...fieldN table_name1, table_name2,...,
tablaN [WHERE Clause] [OFFSET M ][LIMIT N]
        //select delta from tabla where ack!=null && ip(tabla)==ip(dato)

        String cadenaParámetro;
        if (opción==Operaciones.LEER_DELTA)
            cadenaParámetro="delta";
        else
            cadenaParámetro="*";

        String error="";
        Vector<Object> datos=new Vector<>();
        Connection conexión=null;
        try
        {
            Statement sentencia;

```

```
        Class.forName("com.mysql.jdbc.Driver");

        conexión=DriverManager.getConnection(cadenaConexión,usuario,contraseña);

        sentencia=conexión.createStatement();//se crea una sentencia
mysql
        String ipSrcComillas="\")+ipSrc+"\");
        String ipDstComillas="\")+ipDst+"\");
        String sentenciaLeer="select delta from "+tabla+" where
ack_de_frame!=\"null\" and ip_src="+ipSrcComillas+" and ip_dst="+ipDstComillas;

        sentenciaLeer="select * from "+tabla+" where
ack_de_frame!=\"null\" and ip_src="+ipSrcComillas+" and ip_dst="+ipDstComillas;

        sentenciaLeer="select "+cadenaParámetro+" from "+tabla+"
where ack_de_frame!=\"null\" and ip_src="+ipSrcComillas+" and
ip_dst="+ipDstComillas;
        ResultSet rs = null;
        rs=sentencia.executeQuery(sentenciaLeer);
        ResultSetMetaData metaData=rs.getMetaData();
        int númeroDeColumnas=metaData.getColumnCount();

        String[]nombres=new String[númeroDeColumnas];
        for (int i=0;i<númeroDeColumnas;i++)
        {
            nombres[i]=metaData.getColumnName(i+1);
        }
        datos.add(nombres);

        while (rs.next())
        {
            Object[]v=new Object[númeroDeColumnas];
            for (int i=0;i<númeroDeColumnas;i++)
            {
                v[i]=rs.getObject(i+1);
            }
            datos.add(v);
        }
        rs.close();
    }
    catch (ClassNotFoundException | SQLException
e){error=e.getMessage();}
    finally
    {
        if (error.length()==0)
        {
            try {conexión.close();}
            catch (Exception e){error=e.getMessage();}
        }
    }
    if (error.length(>0)//hay error
    {
        datos.clear();
        datos.add(0,null);
        datos.add(1, error);
    }
    datos.trimToSize();
    return datos;
}

public static String getErrorLectura(Vector<Object>v)
{
    if (v.get(0)==null)//hubo error. se devuelve el elemento 1
    {
        return (String) v.get(1);
    }
    else
        return Operaciones.NO_HAY_ERROR;
}
```



```

    }

    public static Object[][] getDatosVectorBaseDeDatos(Vector<Object>v,int
columnas)
    {
        int tamaño=v.size();//número de filas
        Object[][]datos;
        if (tamaño>1)//hay datos en el vector->la base de datos contenía
información
        {
            datos=new Object[tamaño-1][columnas];
            for(int i=1;i<tamaño;i++)
            {
                Object[] fila=(Object[]) v.get(i);
                System.arraycopy(fila, 0, datos[i-1], 0, columnas);
            }
        }
        else
            datos=new Object[0][0];
        return datos;
    }
    public static double[] getDelta(Vector<Object>v)
    {
        //el vector entero después de la lectura de la base de datos.
        //si hay error: v(0)==null, v(1)=error
        //si no hay error: v(0)=nombres, v(i)=fila i de base de datos
        double[]datos=new double[0];
        int[]paquetes=new int[0];
        if (v.get(0)!=null)
        {
            //no error. se han leído los datos de la base de datos
            String[]nombres=(String[]) v.get(0);
            int índice=0;
            boolean encontrado=false;
            while (encontrado==false && índice<nombres.length)
            {
                if(nombres[índice].equals("delta"))
                    encontrado=true;
                else
                    índice++;
            }
            if (encontrado==true)
            {
                int tamaño=v.size();
                if (tamaño>1)//hay datos en el vector->la base de datos
                contenía información
                {
                    datos=new double[tamaño-1];
                    paquetes=new int[tamaño-1];
                    for(int i=1;i<tamaño;i++)
                    {
                        Object[] fila=(Object[]) v.get(i);
                        Object o=fila[índice];
                        if (o!=null)
                        {
                            System.out.println(o.getClass());
                            double d=(double) o;
                            int p=(int)fila[0];
                            datos[i-1]=d;
                            paquetes[i-1]=p;
                        }
                        else
                        {
                            datos[i-1]=-1;
                            paquetes[i-1]=-1;
                        }
                    }
                }
            }
        }
    }
}

```

```
        }
        else
        {
            }
        }
    }
    else //hay error
    {
        }
    }
    return datos;
}
public static int[] getPaquetes(Vector<Object>v)
{
    //el vector entero después de la lectura de la base de datos.
    //si hay error: v(0)==null, v(1)=error
    //si no hay error: v(0)=nombres, v(i)=fila i de base de datos
    double[]datos=new double[0];
    int[]paquetes=new int[0];
    if (v.get(0)!=null)
    {
        //no error. se han leído los datos de la base de datos
        String[]nombres=(String[]) v.get(0);
        int índice=0;
        boolean encontrado=false;
        while (encontrado==false && índice<nombres.length)
        {
            if(nombres[índice].equals("delta"))
                encontrado=true;
            else
                índice++;
        }
        if (encontrado==true)
        {
            int tamaño=v.size();
            if (tamaño>1)//hay datos en el vector->la base de datos
                contenía información
                {
                    datos=new double[tamaño-1];
                    paquetes=new int[tamaño-1];
                    for(int i=1;i<tamaño;i++)
                    {
                        Object[] fila=(Object[]) v.get(i);
                        Object o=fila[índice];
                        if (o!=null)
                        {
                            System.out.println(o.getClass());
                            double d=(double) o;
                            int p=(int) fila[0];
                            datos[i-1]=d;
                            paquetes[i-1]=p;
                        }
                        else
                        {
                            datos[i-1]=-1;
                            paquetes[i-1]=-1;
                        }
                    }
                }
            }
        }
    }
}
```

```

        else //hay error
        {
            }
        return paquetes;
    }

    public static String guardarEnBaseDeDatos(String cadenaConexión,String
tabla, String usuario,String contraseña, Archivo a,String[]columnas,int tipo)
    {
        String[] atributosBaseDeDatos;
        if (tipo==Paquete.PROTOCOLO_IEC || tipo==Paquete.PROTOCOLO_OPC)

        atributosBaseDeDatos=Paquete.obtenerAtributosBaseDeDatos(tipo);
        else
            return "El tipo del archivo no es correcto";

        String error="";
        Connection conexión=null;
        try
        {
            Statement sentencia;
            Class.forName("com.mysql.jdbc.Driver");

            conexión=DriverManager.getConnection(cadenaConexión,usuario,contraseña);

            sentencia=conexión.createStatement();//se crea una sentencia
mysql
            for (int i=0;i<a.paquetes.size();i++)
            {
                //la sentencia dividida en dos partes
                String partel="Insert into "+ tabla +"(";
                String parte2="values (";

                Paquete p=a.paquetes.get(i);
                for (int j=0;j<atributosBaseDeDatos.length;j++)
                {
                    partel=partel+atributosBaseDeDatos[j]+",";
                    parte2=parte2+"'+p.atributos[j]+'','";
                }

                partel=partel+"delta";
                parte2=parte2+p.delta;

                String orden=partel+") "+parte2+")";

                sentencia.executeUpdate(orden);
            }
        }
        catch (ClassNotFoundException | SQLException
e){error=e.getMessage();}
        finally
        {
            if (error.length()==0)
            {
                try {conexión.close();}
                catch (Exception e){error=e.getMessage();}
            }
        }
        return error;
    }

    public static void main(String args[])
    {
        String
cadenaConexión="jdbc:mysql://"+ "localhost"+" :"+ "3306"+" /"+ "mydb";

```

```
        Vector<Object>
v=Operaciones.leerDeBaseDeDatos(cadenaConexión,"tabla_opcua",
"root","1234","127.0.0.1","127.0.0.1",Operaciones.LEER_DELTA);
        for (int i=0;i<v.size();i++)
            System.out.println(v.get(i));
    }
}
```

PLANOS Y PROGRAMAS

PP.1- INTRODUCCIÓN

En este apartado se presentan los algoritmos utilizados en el desarrollo de este trabajo. En primer lugar, se presenta la distribución en módulos de las funciones creadas para un mejor entendimiento del funcionamiento de los programas realizados.

PP.2- DISTRIBUCIÓN DE ALGORITMOS

- **Paquete Files**

- Archivo.java

```
public Archivo(String path,int tipo)
public void print(String separador)
public String[][] getMatrizDatos()
public void calcularDelta()
private static int buscarFrame(String frame,int inicio,int fin, Vector<Paquete>
elementos, int tipo)
```

- Paquete.java

```
public Paquete(Element e,int tipo)
public void llenarInformación(Element e,int tipo)
private static String[] obtenerCampos(int tipo)
private static String[] obtenerAtributosBuscar(int tipo)
static String[] obtenerAtributosMostrar(int tipo)
public static String[] obtenerAtributosBaseDeDatos(int tipo)
public static void print(Element elemento)
public static String toString(Element elemento)
public static String getInformaciónPaquete(Element elemento,String separador,int
tipo)
public static String getInformaciónPaquete(Element elemento,int tipo)
public static String getInformaciónPaquete(Element elemento,String[]
atributos,String[]campos,String separador)
public static String getInformación(Element elemento,String nombreCampo, String
nombreAtributo)
public static String getInformación(Element elemento,String nombreCampo, String[]
nombreAtributos)
public static String[][] getMatrizDatos(Element raíz,int tipo)
public static String buscarInfo(Element elemento, String campo)
public static Element encontrarElemento(Element elemento, String valor)
public static boolean esPaquete(Element elemento)
public static boolean esProtocolo(Element elemento)
public static boolean contieneOPCUA(Element elemento)
public String infoACadena(String separador)
public void print(String separador)
```

- Paquete charts

- PanelChart.java

```
public static ChartPanel crearGráfico(int[] paquetesF1, double[] deltasF1, String
nombreF1, int[] paquetesF2, double[] deltasF2, String
nombreF2)
public static JPanel crearPanelDatos (String mínimo, String máximo, String
media, String desviación, String ip)
public static JPanel crearPanelGráfico (ChartPanel g, JPanel p, JPanel p2)
```

- Paquete operaciones

- Operaciones.java

```
public static double media (double []v)
public static double varianza (double []v)
public static double desviaciónEstándar (double []v)
public static int encontrarMínimo (double []v)
public static int encontrarMáximo (double []v)
public static int getTipo (String nombreBotón)
public static String[] getNombresBaseDeDatos (Vector<Object>v)
public static Vector<Object> leerTodoDeBaseDeDatos (String cadenaConexión, String
tabla, String usuario, String contraseña)
public static Vector<Object> leerTodoDeBaseDeDatosACK (String
cadenaConexión, String tabla, String usuario, String contraseña)
public static Vector<Object> leerDeBaseDeDatos (String cadenaConexión, String
tabla, String usuario, String contraseña, String ipSrc, String ipDst, int opción)
public static String getErrorLectura (Vector<Object>v)
public static Object[] [] getDatosVectorBaseDeDatos (Vector<Object>v, int columnas)
public static double[] getDelta (Vector<Object>v)
public static int[] getPaquetes (Vector<Object>v)
public static String guardarEnBaseDeDatos (String cadenaConexión, String tabla,
String usuario, String contraseña, Archivo a, String[] columnas, int tipo)
```

- Paquete gráficos

- Diálogo1.java

```
public Diálogo1(int tipo)
public class PanelCentralDiálogo extends JPanel
public PanelCentralDiálogo(int tipo)
public class PanelCentralDiálogo extends JPanel
public PanelCentralDiálogo(int tipo)
public String getHost(){return this.host.getText();}
public String getPuerto(){return this.puerto.getText();}
public String getUsuario(){return this.usuario.getText();}
public String getTabla(){return this.tabla.getText();}
public String getBaseDeDatos(){return this.baseDeDatos.getText();}
public String getContraseña()
public class ContinuarListener implements ActionListener
public void actionPerformed(ActionEvent e)
private class CancelarListener implements ActionListener
public void actionPerformed(ActionEvent e)
```

- DiálogoCalcular.java

```
public DiálogoCalcular(int tipo)
public class PanelCentralDiálogo extends JPanel
public PanelCentralDiálogo(int tipo)
public String getHost() {return this.host.getText();}
public String getPuerto() {return this.puerto.getText();}
public String getUsuario() {return this.usuario.getText();}
public String getTabla() {return this.tabla.getText();}
public String getBaseDeDatos() {return this.baseDeDatos.getText();}
public String getContraseña()
public class ContinuarListener implements ActionListener
public void actionPerformed(ActionEvent e)
private class CancelarListener implements ActionListener
public void actionPerformed(ActionEvent e)
```

- VentanaInterna.java

```
public VentanaInterna(String nombre ,boolean b1,boolean b2,boolean b3,boolean
b4,Archivo archivo,int tipo)
private class GuardarListener implements ActionListener
public void actionPerformed(ActionEvent event1)
```

- VentanaInternaCalcular.java

```
public VentanaInternaCalcular(String nombre,int[] p1, double[]d1,String
s1,int[]p2,double[]d2,String s2)
```

- VentanaInternaMostrar.java

```
public VentanaInternaMostrar(String nombre,Object[][]datos,
String[]nombresColumnas)
```

- VentanaPrincipal.java

```
public static File seleccionarArchivoYDevolverArchivo(String nombre)
public class AbrirListener implements ActionListener
public void actionPerformed(ActionEvent event1)
public class MostrarTablaListener implements ActionListener
public void actionPerformed(ActionEvent e)
public class MostrarTablaListenerACK implements ActionListener
public class CalcularListener implements ActionListener
public void actionPerformed(ActionEvent e)
public class Listener implements ActionListener
public void actionPerformed(ActionEvent e)
```


PLIEGO DE CONDICIONES

PLIEGO DE CONDICIONES

Para el presente proyecto, se ha utilizado una serie de herramientas software y hardware, indicadas a continuación.

Como elementos hardware se empleó:

- Un **ordenador portátil** con microprocesador Intel® Core® 2 Duo a 1,8 GHz, 4 GB de memoria RAM, y 120 GB de disco duro, el cual se utilizó en la fase de desarrollo, primario con Netbeans.
- Un **ordenador personal** con microprocesador Intel® Core I 7™ a 4,00 GHz, 16 GB de memoria RAM y 240 GB de disco duro SSD, en el cual se llevó a cabo gran parte de los trabajos relativos a la virtualización y captura de datos.

También se utilizaron las siguientes herramientas software:

- **Microsoft Office® 2013:** Es el paquete de herramientas, en las que se incluye Microsoft Word y Microsoft PowerPoint, que se han utilizado para la elaboración de la memoria y presentación del proyecto.
- **Netbeans 8.1:** Software de la factoría Oracle para la programación en Java. Esta herramienta es de carácter libre y no ha habido que pagar por uso.
- **Windows® 10:** Es el sistema operativo bajo el que se trabajó en el ordenador personal.
- **Windows® 7:** Es el sistema operativo bajo el que se trabajó en el ordenador portátil.
- **Wireshark:** es el analizador de paquetes de red de carácter libre distribuido con licencia GPL/GNU. Se ha usado tanto para el Sistema Operativo Windows como para Linux.
- **MySQLWorkbench:** Es el software de Oracle de distribución libre para la gestión de base de datos.
- **Vmware Player:** Es la herramienta de virtualización de máquinas. Esta herramienta es de carácter libre y por lo tanto no tiene coste.
- **Linux Ubuntu 16.04 LTS:** Se ha usado para el funcionamiento del simulador OPCUA, tanto del servidor como del cliente.

Otros elementos software que se utilizaron fueron los simuladores.

- **Simulador de paquetes OPCUA:** Es el simulador de la compañía alemana Unifed Automation. Este software tiene licencia demo y solo se puede usar de forma continua durante 1 hora. También se puede descargar el código de dicha aplicación.
- **Simulador de paquetes IEC-60870:** Este simulador es de también de carácter demo, desarrollado por el ingeniero alemán Reinhard Fin que desde su página web promociona sus productos para empresas, pudiendo ser descargados en versión demo.

Software de paquetes diseñado para este proyecto.

Para la ejecución de este software, es necesario la utilización de un equipo con características similares al ordenador personal nombrado en este pliego de condiciones. Hay que tener en cuenta que cuando se analiza una red de entorno industrial, el wireshark puede capturar varias decenas de miles de paquetes. Como ya se ha explicado en el *capítulo 4* de la memoria, un archivo de datos del wireshark puede ocupar varios cientos de MB e incluso varios GB. Esto implica una capacidad de procesamiento muy elevada, tardando el programa diseñado, bastante tiempo en procesar la información y guardarla en la base de datos.

También es necesario para el uso del wireshark, que la capacidad de memoria RAM y disco duro sea grande.

PRESUPUESTO

Don Jesús Galván Ruiz, autor del presente Proyecto Fin de Carrera, declara que:

El Proyecto Fin de Carrera con título “*Desarrollo de un entorno de test de una red comunicaciones de Dispositivos Electrónicos Inteligentes*”, desarrollado en la Escuela de Ingeniería de Telecomunicación y Electrónica (EITE), de la Universidad de Las Palmas de Gran Canaria, en el período de un 8 meses, tiene un coste de desarrollo total de **36.495,90 €** correspondiente a la suma de las cantidades consignadas a los apartados considerados a continuación.

El autor de proyecto
Jesús Galván Ruiz

Las Palmas de Gran Canaria a 4 de Octubre de 2013.

P.1- DESGLOSE DEL PRESUPUESTO

Para la realización del presupuesto se han seguido las recomendaciones del Colegio oficial de Ingenieros de Telecomunicación (COIT), debido a que no existe el colegio de Ingenieros en Electrónica, sobre los baremos orientativos mínimos para trabajos profesionales en 2013. El presupuesto se ha desglosado en varias secciones en las que se han separado los distintos costes asociados al desarrollo del proyecto. Estos costes se dividen en:

1. Recursos materiales.
2. Trabajo tarifado por tiempo empleado.
3. Costes de redacción del proyecto.
4. Material fungible.
5. Derechos de visado del COIT.

6. Gastos de tramitación y envío.
7. Aplicación de impuestos.

P.2- RECURSOS MATERIALES

Los paquetes software usados para la redacción de la memoria, y los sistemas operativos bajo los que se ejecutó el trabajo. Asimismo, se incluyen los equipos hardware usados para dar soporte a estas herramientas.

Se estipula el coste de amortización para un período de 3 años. Para ello, se utilizará un sistema de amortización lineal o constante, en el que se supone que el inmovilizado material se deprecia de forma constante a lo largo de su vida útil. La cuota de amortización anual se calcula usando la siguiente fórmula:

$$Cuota\ Anual = \frac{Valor\ de\ Adquisición - Valor\ residual}{Número\ años\ de\ vida\ útil}$$

El “valor residual” es el valor teórico que se supone que tendrá el elemento después de su vida útil.

P.2.1- RECURSOS SOFTWARE

Las herramientas software utilizadas en presente proyecto fueron:

- Microsoft Office® 2013
- Windows® 7
- Windows 10

Teniendo en cuenta que la duración del proyecto es de aproximadamente de un año y el cálculo del coste de amortización se establece en un período de 3 años, los costes de amortización se calcularán para el primer año. Estos costes se pueden ver en la tabla de costes de las herramientas software.

Costes de las herramientas Software			
Descripción	Coste Total	Valor Residual (3 años)	Valor Amortización (8 meses)
Microsoft Office® 2013	375,16 €	0 €	83,37 €
Netbeans 8.1	0 €	0 €	0 €
Vmware Player	0 €	0 €	0 €
MySQL Workbench	0 €	0 €	0 €
Wireshark	0 €	0 €	0 €
Simulador Unifed Autom.	0 €	0 €	0 €
Simulador Winpp104	0 €	0 €	0 €
Linux Ubuntu	0 €	0 €	0 €
Windows® 10	126,10 €	0 €	28,02 €
Windows® 7	126,10 €	0 €	28,02 €
Total de Costes			139,41 €

Tabla P.1 Costes de las herramientas Software

Por lo tanto, el coste total del material software libre de impuestos asciende a *ciento treinta y nueve euros con cuarenta y un céntimos (139,41 €)*.

P.2.2- RECURSOS HARDWARE

Las herramientas hardware en la que se apoya el presente proyecto son:

- Un **ordenador portátil** con microprocesador Intel® Core® 2 DUO 1,8 GHz, 4 GB de memoria RAM, y 120 GB de disco duro SSD.
- Un **ordenador personal** con microprocesador Intel® Core™ I 7 a 4 GHz, 16 GB de memoria RAM y 240 GB de disco duro SSD.

Aplicando la regla de costes anterior se tiene:

Costes de las herramientas Hardware			
Descripción	Coste Total	Valor Residual (3 años)	Valor Amortización (8 meses)
Ordenador portátil	800 €	266,66 €	177,77 €
Ordenador Personal	990 €	150 €	280 €
Total de Costes			457,77 €

Tabla P.2.- Costes de las herramientas Hardware

Por lo tanto, el coste total del material hardware libre de impuestos asciende a *cuatrocientos cincuenta y siete euros con setenta y siete céntimos (457,77 €)*.

P.3- TRABAJO TARIFADO POR TIEMPO EMPLEADO

El proyectante ha invertido 8 meses en las tareas de formación, especificación, desarrollo y documentación necesarias para la elaboración del presente proyecto fin de carrera. El importe de las horas de trabajo empleadas para la realización del proyecto se calcula siguiendo las recomendaciones del COIT:

$$H = C_t \times 74,88 \times H_n + C_t \times 96,72 \times H_s \text{ €}$$

Donde:

H → son los honorarios totales por el tiempo dedicado.

H_n → son las horas normales trabajadas (dentro de la jornada laboral)

H_s → son las horas especiales.

C_t → es un factor de corrección función del número de horas trabajadas.

Teniendo en cuenta que el tiempo destinado para la formación y documentación no se incluye en el cálculo debido a que estas tareas son necesarias y benefician sólo al alumno, para la realización de este proyecto han sido necesarias 1280 horas (8 horas/día x 5 días/semana x 4 semanas/mes x 8 meses), todas ellas dentro del horario normal.

Según el COIT, el coeficiente C_t tiene un valor variable en función del número de horas empleadas de acuerdo con la siguiente tabla:

Horas empleadas	Factor de corrección C_t
Hasta 36 horas	1,00
De 36 a 72 horas	0,90
De 72 a 108 horas	0,80
De 108 a 144 horas	0,70
De 144 a 180 horas	0,65
De 180 a 360 horas	0,60
De 360 a 540 horas	0,55
De 540 a 720 horas	0,50
De 720 a 1080 horas	0,45
Más de 1080 horas	0,40

Tabla P.3.- Factor de corrección en función del número de horas invertidas

Como se puede observar el número de horas es superior a 960, según la tabla P.3, $C_t = 0,45$ por lo que según la ecuación del importe de horas de trabajo se obtiene una tarifa total por el tiempo empleado de *treinta y dos mil trescientos cuarenta y 8 euros con dieciséis céntimos (32.348,16 €)*.

$$H = 0,45 \times 74,88 \times 960 + 0,45 \times 96,72 \times 0 = 32.348,16$$

En la tabla siguiente se desglosa el tiempo de trabajo invertido.

Costes por Tiempo Empleado			
Descripción	Tiempo	Coste/mes	Importe
Formación	1 mes	0 €	0 €
Documentación	1 mes	0 €	0 €
Especificación	2 meses	5391,36 €	10.782,72 €
Desarrollo	4 meses	€	21.565,44 €
Total de Costes			32.348,16 €

Tabla P.4.- Costes por Tiempo Empleado

Los honorarios totales por tiempo dedicado libres de impuestos ascienden a *treinta y dos mil trescientos cuarenta y 8 euros con dieciséis céntimos (32.348,16 €)*.

P.4- COSTES DE REDACCIÓN DEL PROYECTO

El importe de la redacción del proyecto se calcula de acuerdo a la siguiente expresión:

$$R = 0,07 \times P \times C_h$$

Donde:

P es el presupuesto del proyecto.

C_h es el coeficiente de ponderación en función del presupuesto.

En la siguiente tabla se muestra el presupuesto calculado hasta el momento:

Descripción	Costes
Recursos Software	139,41 €
Recursos Hardware	457,77 €
Trabajo Tarifado por Tiempo Empleado	32.348,16 €
Total de Costes	32.945,34 €

Tabla P.5.- Presupuesto de ejecución material

El presupuesto **P** calculado hasta el momento asciende a **32.945,34 €**. El coeficiente de ponderación para presupuestos en torno a la cantidad anterior viene definido por el COIT con un valor de 0,40, el coste derivado de la redacción del proyecto es de:

$$R = 0,07 \times 32.945,34 \times 0,40 = 922,47 €$$

Por tanto el coste libre de impuestos derivado de la redacción del proyecto es de *novecientos veintidós euros con cuarenta y siete céntimos (922,47 €)*.

P.5- MATERIAL FUNGIBLE

Además de los recursos hardware y software, en este proyecto se han empleado otros materiales, como son los folios y el tóner de la impresora entre otros, que se especifican como material fungible. En la tabla P.6. se muestran los costes generados por estos recursos.

Descripción	Costes
Folios	10 €
Tóner de la impresora	75 €
Encuadernación	105 €
Total de Costes	190 €

Tabla P.6.- Coste del material fungible

P.6- DERECHOS DE VISADO DEL COIT

Los gastos de visado del COIT se tarifican mediante la siguiente expresión:

$$V = 0,006 \times P \times C_v$$

Donde:

P es el presupuesto del proyecto.

C_v es el coeficiente reductor en función del presupuesto del proyecto.

El presupuesto **P** calculado hasta el momento asciende a la suma de los costes de ejecución material, de redacción y de material fungible.

$$P = 32.945,34 + 922,47 + 190 = 34.057,81€$$

Como el coeficiente **C_v** para presupuestos de más de 30.050 € y menos de 60.101 €, viene definido por el COIT con un valor de 0,90, el coste de los derechos de visado del proyecto asciende a la cantidad de:

$$V = 0,006 \times 34.057,81 \times 0,90 = 183,91 \text{€}$$

Por tanto el coste de los derechos de visado del proyecto *ciento ochenta y tres euros con noventa y un céntimos (183,91 €)*.

P.7- GASTOS DE TRAMITACIÓN Y ENVÍO

Los gastos de tramitación y envío están fijados en **6,01 €**.

P.8- APLICACIÓN DE IMPUESTOS

El coste total del proyecto, antes de aplicarle los correspondientes impuestos, asciende **34.108,32 €**, a lo que hay que sumarle el 7% de IGIC, con lo que el coste definitivo del proyecto es:

Costes Totales del Proyecto		
Descripción	Coste Parcial	Total
Hardware	177,77 €	
Software	280 €	
Recursos Materiales		457,77 €
Coste de Ingeniería		32.348,16 €
Coste de Redacción		922,47 €
Material Fungible		190,00 €
Derechos de Visado		183,91 €
Tramitación y Envío		6,01 €
Subtotal:		34.108,32 €
Aplicación de Impuestos (7% I.G.I.C)		2.387,58 €
Total de Costes		36.495,90 €

Tabla P.7.- Costes Totales del Proyecto

El presupuesto total asciende a la cantidad de **treinta y seis mil cuatrocientos noventa y cinco euros con noventa céntimos (36.495,90 €)**.

El Autor del Proyecto.
Jesús Galván Ruiz.