Propuesta de un protocolo de interconexión de dispositivos IP inteligente de red

Roberto Ferro Escobar*, Hernán Paz Penagos** y Roberto Cárdenas***

Se realizó el estudio del protocolo estandarizado para dispositivos inteligentes de redes Ethernet mediante la norma EIA/IS-709 [4] una vez entendido el funcionamiento mundial de esta norma, se propone la creación del protocolo IDTP mediante el uso de herramientas OO para su modelamiento y la ejecución del mismo mediante el uso de Java JDK.2.1.2 con el fin de correr en plataformas de uso gratuito como Linux. Se propone un ejemplo específico para la simulación de un sistema control de acceso donde el Usuario debe estar autorizado dentro del sistema para poder acceder al edificio. El tratamiento de estos datos puede ser ampliado a cualquier tipo de dispositivos, como sensores actuadores de temperatura, de presión, de iluminación.

Palabras clave: actuador, dispositivo, Ethernet, Linux, OO, sensor

* Ingeniero electrónico, Universidad Distrital Francisco José de Caldas, magíster en teleinformática, Profesor de la facultad de Ingeniería Electrónica, Universidad Distrital Francisco José de Caldas. rferro@segobdis.gov.co
** Magíster en teleinformática de la Universidad Distrital Francisco José de Caldas. Ingeniero electricista de la Universidad Nacional de Colombia, ingeniero electrónico de la Universidad Distrital Francisco José de Caldas y filósofo de la Universidad Santo Tomás de Aquino. Docente del área de comunicaciones, facultad de ingeniería electrónica de la Escuela Colombiana de Ingeniería. hpaz@escuelaing.edu.co.
*** Ingeniero electrónico, Universidad Distrital. Magíster en teleinformática, profesor de la facultad de Ingeniería en la maestría y especialización en teleinformática. bitek@multi.net.co

Introducción

La presente investigación tiene como objetivo principal formular y proponer las bases iniciales que sirvan de sustento en el campo de las redes Ethernet para la conexión de cualquier dispositivo electrónico a través de un conector RJ45, que pueda comunicarse a un servidor de red que pueda ser monitoreado y administrado por medio de un software gráfico GUI, que pueda ser administrado desde cualquier computador conectado a la red o incluso desde Internet. De esta manera se puede controlar cualquier tipo de proceso domótico, industrial o de una factoría.

> En los países de avanzada tecnológica ya se están implementando muchos dispositivo inteligentes autónomos que poseen circuitos electrónicos encargados de comunicarse con la red utilizando la capa física de datos.

En los países de avanzada tecnológica ya se están implementando muchos dispositivo inteligentes autónomos que poseen circuitos electrónicos encargados de comunicarse con la red utilizando la capa física de datos.

En Colombia estos dispositivos Ethernet aún no están disponibles y sus costos son elevados en comparación con dispositivos de uso tradicional RS232 que funcionan en forma aislada; por tanto, es importante tener el horizonte claro y ver que la avanzada tecnológica tiende a usar dispositivos autónomos con cierto grado de inteligencia por medio de CI microcontrolados con opciones de conexión a redes Ethernet. En EU y en Europa existen empresas como Echelon [4], que se encuentran desarrollando estos dispositivos desde el hardware hasta el software, que para este caso se convierte en el protocolo de transmisión sobre los cuales viajan los datos y sirve como interfaz de comunicación entre el cliente y el servidor. En este artículo se citan los precedentes utilizados por la empresa Echelon [4] para el desarrollo de su protocolo LonTalk, reconocido por la EIA como el estándar EIA/IS-709X para automatización de casas, edificios e industrias.

Modelamiento en UML

Para el modelamiento de este artículo una de las clases, la más importante, es la llamada Servidor principal que en la práctica es un computador principal con sistema operativo Linux o Win-

dows, dependiendo de la aplicación y los recursos impuestos en la respectiva industria. A esta clase se le denomina Servidor_computador, ya que es el encargado de dirigir todas las funciones principales de la red de control distribuido. Las demás clases son "n" dispositivos IP conectados a la red mediante un conector RJ45 que tiene en su interior una interfaz de red o un chip adecuado para el mismo (Realtek 8019), un microcontrolador y los sensores a utilizar dependiendo de la aplicación que se necesite implantar. Para el prototipo se utilizan los elementos mas sobresalientes que corresponden a las clases: Sensor_actuador_acceso, sensor_actuador_camaras, Sensor_actuador_fuego Sensor_actuador_presion, Sensor_actuador_temperatura y Sensor_actuador_valvulas. Se eligieron sensores y actuadores en el mismo objeto debido a que deben realizar funciones de control y funciones de medición y monitoreo continuo de su entorno. El diseño asociado para el Servidor_computador se muestra en la figura 1. Se puede apreciar cada uno de los atributos y las operaciones realizadas en el interior de esta clase, que en realidad será la que pertenece al servidor; cada operación realizada corresponde a cada comando enviado por la red a los clientes. Se debe tener en cuenta que se creará un canal de control por donde se enviará cada uno de los comandos, y por este mismo se obtendrán las respectivas respuestas emitidas por cada dispositivo inteligente de red para este caso. En la figura 2 se muestra la clase "Cliente", que para este caso será la principal representada por Nv_dispositivo_ip, que en realidad será cualquier dispositivo IP conectado a la red Ethernet por medio de un RJ45 y hará las veces del respectivo Cliente. Para cualquier dispositivo inteligente Sensores_actuadores que deben controlar variables lineales como presión, temperatura, válvulas y control de acceso, se puede tomar el siguiente modelo que corresponde en el protocolo IDTP de un "cliente".

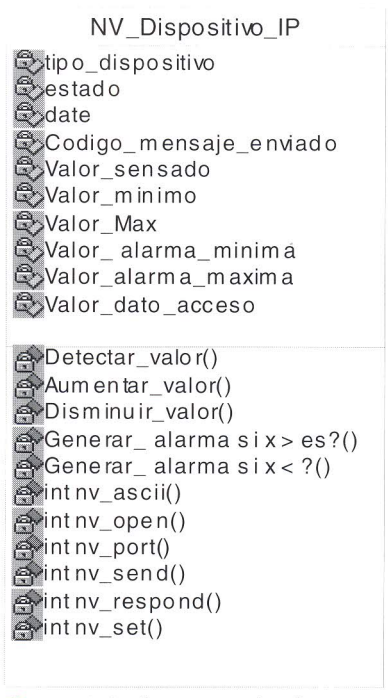

Figura 1 Atributos y métodos para el Objeto Servidor.

Para este artículo se definen en su orden los siguientes atributos para la clase Servidor_ computador: TTL es el tiempo de duración de un paquete enviado en la red máximo hasta 255 saltos de duración. Tipo_dispositivo_ ip es el encargado de identificar qué tipo de dispositivo IP se conectó a la red. Direccion_ip se dedica a tomar el dato referente a la dirección IP del dispositivo que se conecta y, por último, el atributo "estado" se encarga de crear el socket que monitorea el dispositivo IP para saber si se encuentra en estado encendido o apagado.

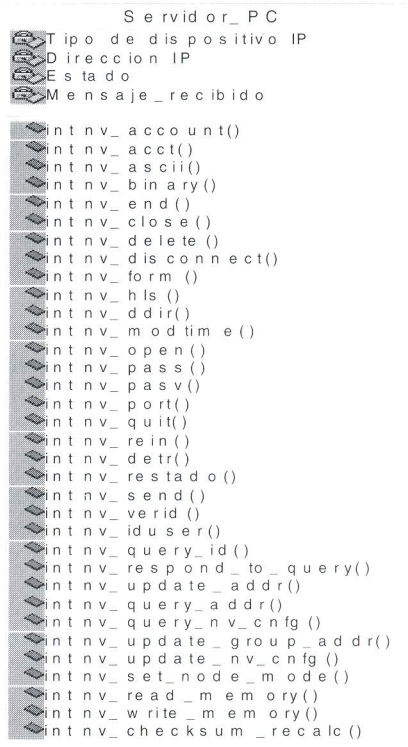

Figura 2 Modelo para sensores actuadores genéricos.

Para el dispositivo genérico sensor_actuador se definen los atributos de la figura 2.

El primero de los atributo es estado, valor_temperatura, o valor_presion, encargado de capturar el dato correspondiente a la temperatura o presión, medida en ese momento por el dispositivo. Dependiendo de la aplicación que se quiera dar al dispositivo, se pueden implementar los atributos temperatura_min, temperatura_maxima, temperatura_alarma, etc.

Diagrama de casos de uso cliente servidor para dispositivos IP

El diagrama de casos de uso (figura 3) representa la forma en que un Cliente (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en que los elementos interactúan (operaciones o casos de uso).

Elementos

- El **Actor** es el administrador de la red de dispositivos IP.

Mensaje al mismo objeto

- El usuario/cliente presiona el botón de comienzo para activar el servidor

• Existe un operador que desea hacer lo siguiente:

a. Entrar al sitio de trabajo mediante una tarjeta o un sistema sofisticado de control de acceso.

b. Al final de cada día se solicita un resumen diario o semanal de las entradas y salidas de usuarios en el sistema.

• El Administrador, además de la visualización, debe estar en capacidad de:

a. Cambiar la información asociada a temperatura, los usuarios restringidos, revisar válvulas, cambiar valores, encender o apagar luces, etc.

b. Dar una alarma en el caso que haya un sensor de fuego activado, se dañe una válvula, o detecte la presencia de un intruso.

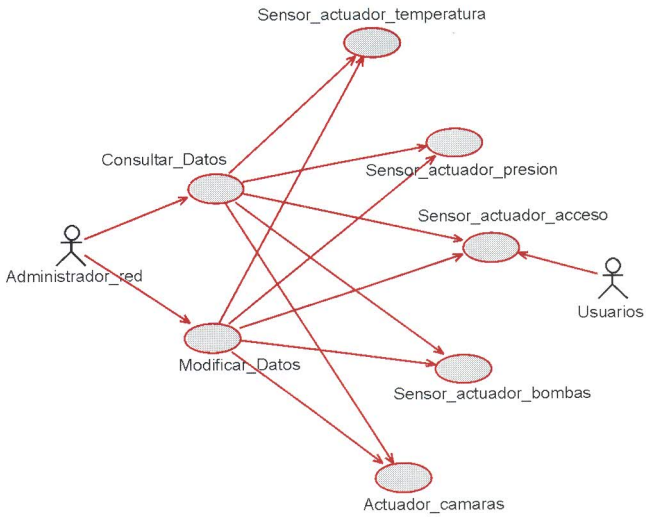

Figura 3 Diagrama de casos de uso en la red de dispositivos IP inteligentes.

Diagrama de interacción de cliente servidor para dispositivos IP

Para el protocolo IDTP se realizó todo el diagrama de Interacción desde la creación del socket hasta el envío de la respectiva dirección IP y el cierre de la Comunicación entre el cliente y el servidor; para ello se puede visualizar la figura 4 donde se observan claramente cada uno de los elementos asociados al diagrama de casos de uso.

En la figura 5 se puede apreciar los elementos asociados al diagrama de interacción para la medición de temperatura de un Sensor_actuador_temperatura; para los demás dispositivos, se emplea el mismo modelo. El modelo UML de todo el protocolo aplicado a la red de dispositivos IP inteligentes se puede ver en la figura 6, donde se observan todas las clases principales y el respectivo modelo OO del protocolo IDTP propuesto en esta investigación. En la figura 7 se puede apreciar la red global de dispositivos IP para todas las clase modeladas en este artículo. Es de entenderse que la red puede crecer hasta cualquier cantidad de dispositivos IP, ya que depende de parámetros como la dirección IP y la segmentación propia de la red.

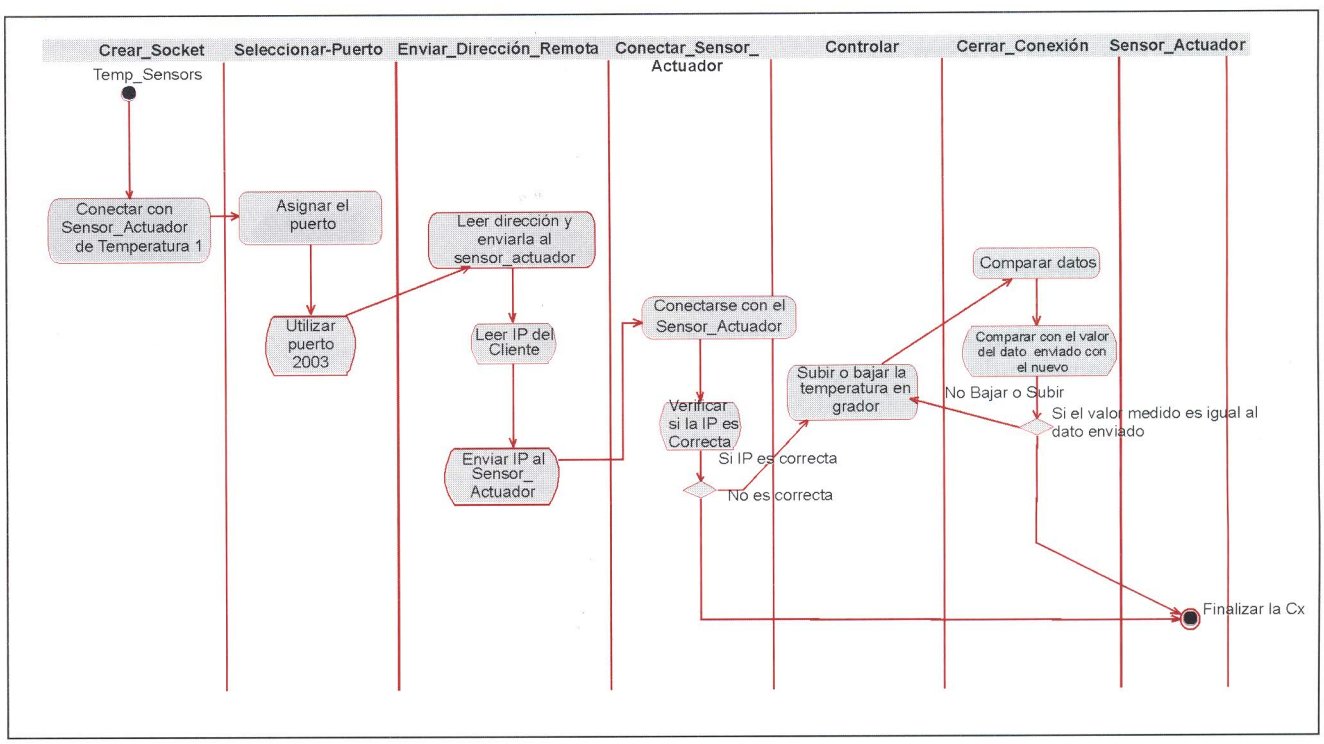

Figura 4 Diagrama de interacción en la red de dispositivos IP.

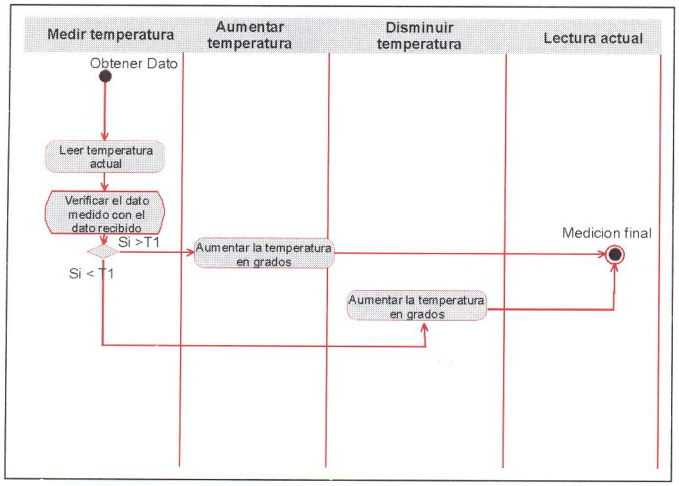

Figura 5 Diagrama de Interacción para medición de temperatura.

Protocolo propuesto

El protocolo propuesto para transferir datos se llama ***IDTP*** *(Intelligent, Device Transfer Protocol)* y está basado en la familia de protocolos TCP/IP. Mediante comandos de control puede manipular dispositivos inteligentes conectados a la red, independiente del sistema operativo del servidor.

IDTP trabaja bajo el paradigma **cliente/servidor**. El cliente es la aplicación creada en un dispositivo inteligente que brinda un servicio; el servidor almacena o controla los datos enviados por éste.

Los objetivos principales de IDTP son:

• Enviar datos entre dispositivos remotos que estén conectados bajo el protocolo TCP/IP.

• Transferir datos de forma segura, óptima y fiable a través de la red e Internet

Para iniciar una sesión IDTP, este protocolo emplea dos conexiones TCP simultáneas. Una es la llamada **conexión de control** y la otra es la **conexión de datos.** La primera se usa para el intercambio de comandos con mensajes y respuestas, una configuración donde el cliente se conecta con el servidor. IDTP utiliza el protocolo TELNET en la conexión de control para manejar estas operaciones, ya que está establecido como estándar. La conexión de datos se utiliza para el intercambio de datos. Ésta se crea y se cierra según los comandos que ejecute el cliente; en este caso, el dispositivo inteligente de red.

Cuando el cliente contacta con el servidor, éste hace una apertura en el puerto 2004 y espera la conexión del cliente; cuando éste le responde, empiezan a negociar una conexión TCP. La conexión de control se crea durante toda la sesión y, según los datos que se transmitan, IDTP crea conexiones por cada transferencia mediante Sockets Stream. Se pueden enviar y recibir datos simultáneamente; a esto se le denomina **full dúplex**.

La transferencia de datos IDTP se realiza en:

• **ASCII**. Es el mas común. Usado para la transferencia de archivos de texto, el equipo que envía debe convertir los archivos, independientemente de su formato o su estructura, al formato genérico de 8 bits. El que recibe deberá volver a convertirlos en el formato original.

• **Modo de transmisión**. Es el paso para transferir el dato. Se tendrán que especificar los puertos de transferencia para cada dispositivo. Una vez hecho lo anterior, el que va a transferir debe estar en la escucha de su puerto hasta que el otro dispositivo reciba la petición de abrir el suyo. Una vez recibido el comando, se inicia la comunicación de transferencia para verificar la conexión, enviando un bloque con una cabecera de formato específico. Una vez confirmado, se empieza a enviar bloques de 8 bits hasta cuando se completa la transferencia. Se envía siempre un paquete con una cabecera para especificar que la transferencia está completa; al concluir, los puertos se cierran adecuadamente.

• **Block Mode**. El dato es transferido en bloque con una cabecera. Este modo es el recomendado para que los usuarios estén disponibles para recuperar datos no enviados y completar la transferencia.

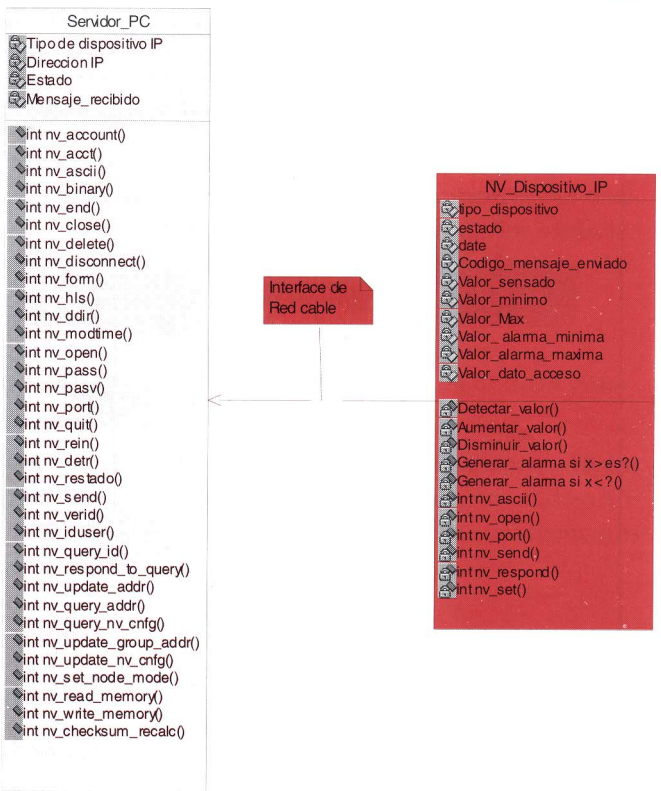

Figura 6. Diagrama de Clases IDTP "Cliente-Servidor".

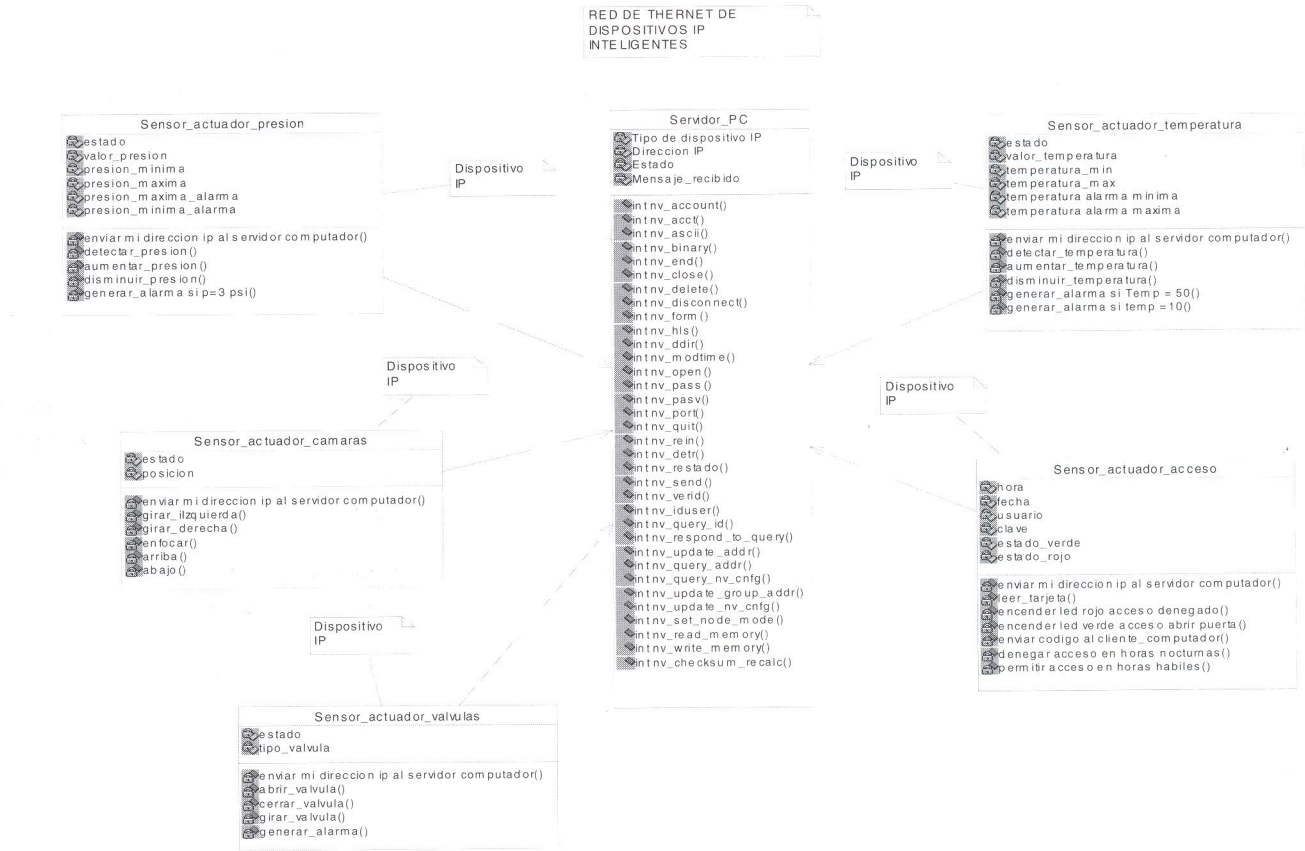

Figura 7. Diagrama de Clases para la red de dispositivos IP "Inteligentes".

A continuación se presentan los comandos utilizados en la conexión de datos en el protocolo IDTP.

Para realizar la conexión a un servidor IDTP, el dispositivo inteligente cliente deberá crear la conexión con el servidor, utilizando un *socket*, con el puerto estándar de IDTP. Se tomó el 2004, que luego conectará al puerto del servidor. Una vez que se inicia la sesión, el cliente y el servidor intercambiaran comandos de petición y respuesta.

Los comandos se clasifican en:

• **Identificadores para el control de acceso**. Sirven para identificar el dispositivo inteligente con el servidor. Entonces el servidor lo dotará de privilegios de manipulación de datos que tiene cualquier dispositivo o en qué ubicación se le permite acceder.

• **Parámetros para la transferencia de datos**. Con estos parámetros, el dispositivo inteligente cliente establece las opciones de transmisión, estructuras de datos y modo de transmisión.

• **Comandos de servicio**. Son el resto de comandos suplementarios del IDTP, como los de abortar **transferencia, eliminar datos o crear directorios.**

Comandos propuestos

• **nv_account** [contraseña]: suministra una contraseña suplementaria para tener acceso a los recursos una vez se haya ejecutado login con éxito. Depende del dispositivo.

• **nv_acct**: mensaje del servidor que requiere un argumento del cliente para identificar la cuenta de usuario.

• **nv_ascii**: transferir los archivos en modo ASCII. Valor por defecto.

• **nv_binary**: transferir los archivos en modo binario.

• **nv_end**: cerrar la conexión con el servidor y apagar el dispositivo.

• **nv_close**: terminar la conexión con el servidor, pero no apagar el dispositivo

• **nv_delete** [datos]: eliminar datos almacenados en el dispositivo.

• **nv_disconnect**: igual que close.

• **nv_form** [formato]: establecer la forma de transferencia de un archivo.

• **nv_hls**: lista la hora del dispositivo.

• **nv_ddir**: escribe un listado de los datos remotos en un archivo local.

• **nv_modtime**: muestra la fecha de última modificación de un dato.

• **nv_open** [dirección] *[puerto]*: conecta a un servidor. El número de puerto es opcional; el número predeterminado es 2004.

• **nv_pass**: mensaje del servidor que requiere que el dispositivo inteligente cliente envíe la contraseña del nombre de usuario.

• **nv_pasv** [puerto]: pedir proceso de conexión de datos en un determinado número de puerto del dispositivo inteligente cliente.

• **nv_port** [socket]: especificar al servidor un socket (combinación de dirección IP de 32 bits y número de puerto de 16 bits, dividido en campos de 8 bits) del dispositivo inteligente cliente para establecer una conexión de datos.

• **nv_quit**: terminar sesión y desconectar la conexión.

• **nv_rein**: regresar al estado que sigue inmediatamente al establecimiento de la conexión de control (reiniciar) de la comunicación.

• **nv_detr** [dato]: comando utilizado para enviar datos del servidor al cliente.

• **nv_restado**: muestra el estado del dispositivo inteligente remoto.

• **nv_send**: envía un dato al dispositivo inteligente.

• **nv_verid**: muestra la versión del dispositivo inteligente.

• **nv_IDUSER**: Cuando el servidor lo solicite, con este mensaje se requiere un parámetro del dispositivo inteligente cliente para identificar su nombre dispositivo inteligente en la red.

• **nv_query_id**: pregunta por el tipo de dispositivo y su dirección IP

• **nv_respond_to_query**: responde a la solicitud enviada con el tipo y la dirección IP

• **nv_update_addr**: actualiza la dirección en caso de ser necesario.

• **nv_query_addr**: pregunta la dirección IP

• **nv_query_nv_cnfg**: configura el dispositivo inteligente.

• **nv_update_group_addr**: actualiza las direcciones IP de un grupo.

• **nv_update_nv_cnfg**: actualiza la configuración del dispositivo.

• **nv_set_node_mode**: configura el tipo de dispositivo en la red.

• **nv_read_memory**: lee la memoria del dispositivo.

• **nv_write_memory**: escribe la memoria del dispositivo con el valor deseado.

• **nv_checksum_recalc**: solicita el recálculo de la suma de verificación de bits del dato.

El resto de comandos carecen de importancia, ya que no son son comandos transferidos, sino de funcionamiento interno de los clientes de IDTP.

Una vez que el usuario ejecuta un comando desde el servidor o se ha realizado alguna otra operación, el protocolo IDTP debe reaccionar para confirmar o responder peticiones del cliente enviando algún mensaje de error con su posible motivo. Para ello emplea un lenguaje de números o dígitos de códigos llamados **códigos de respuesta.**

Código	Descripción
1 ??	Respuesta preliminar positiva. El dispositivo inteligente cliente inició la acción solicitada.
2 ??	Respuesta de terminación positiva. El dispositivo inteligente cliente terminó con éxito la acción solicitada.
3 ??	Respuesta intermedia positiva. El cliente dispositivo inteligente aceptó el comando, pero la acción solicitada necesita más información.
4 ??	Respuesta de terminación negativa transitoria. El cliente dispositivo inteligente no aceptó el comando, y no se realizó la acción solicitada.
5 ??	Respuesta de terminación negativa permanente. El cliente dispositivo inteligente no aceptó el comando y no se realizó la acción solicitada.
? 0 ?	Se refiere a errores de sintaxis.
? 1 ?	Se refieren a solicitudes de información, como estado del dispositivo.
? 2 ?	Se refieren a las conexiones de control o de datos.
? 3 ?	Para el proceso de inicio de sesión y procedimientos contables.
? 4 ?	Aún no se ha especificado.
? 5 ?	Indican el estado del sistema de datos del cliente dispositivo inteligente frente a la solicitud realizada u otra acción.

Programa realizado en Visual J++ para la aplicación del protocolo a dispositivos IP inteligentes

Para poner en práctica lo propuesto en el presente artículo se implantó, mediante la ayuda de Racional Rose para la realización del código aplicado al modelo UML, la siguiente interfaz gráfica para probar la transmisión y recepción de datos mediante el uso del protocolo. Se propuso el siguiente ejemplo práctico que pretende simular el caso de un control de acceso para usuarios que pretenden ingresar utilizando un nombre de usuario y la respectiva clave.

En primera instancia, la realización del protocolo OO y, por tanto, el uso del mismo será mediante eventos como hacer clic sobre algunos botones que tienen como función reemplazar las tareas realizadas antiguamente por medio de interfaz de comandos.

La ventaja en la realización de esta interfaz gráfica en Java es su sencilla transportabilidad a otros sistemas operativos como Linux mediante la instalación de JDK JDK.2.1.2 para Linux sin mas preámbulos. En las figuras siguientes se puede apreciar cada una de las ventanas que puede ser exhibida por el programa realizado.

El código creado se puede solicitar escribiendo a la dirección de correo electrónico de los autores.

El programa utiliza la comunicación a través de Sockets de flujo, debido a que la comunicación debe ser fiable y

A continuación se listan los códigos de respuesta IDTP:

Código	Descripción
110	Reiniciar respuesta.
120	Dispositivo listo.
125	Conexión de datos abierta. Inicio de la transferencia de datos.
150	Buen estado del dispositivo. Todos su componentes funcionan correctamente.
200	Comando correcto.
202	Comando no implementado.
211	Estado del sistema.
212	Estado del cliente dispositivo inteligente.
213	Estado del sensor.
220	Servicio listo para nuevo dispositivo.
221	Servicio cierra la conexión de control.
225	Conexión de datos abierta. No hay transferencia de datos en curso.
226	Cierra la conexión de datos. Acción de datos solicitada terminada con éxito.
227	Entra al modo pasivo a la espera de transferencia.
230	Usuario inició acceso.
250	La acción de datos solicitada es correcta.
331	Nombre de usuario correcto. Se pide contraseña, tarjeta o huella dactilar.
332	Se necesita permiso para acceder.
350	Acción de datos solicitada pendiente en espera de más comandos.
421	Servicio no disponible, se cierra conexión de control.
425	No se puede abrir la conexión de datos.
426	Conexión cerrada, transferencia abortada.
450	Acción de archivo solicitada no se tomó; datos no disponibles.
451	Abortada acción solicitada: error local en el procesamiento.
452	Acción solicitada no se tomó. Espacio de almacenamiento en sistema insuficiente.
500	Error de sintaxis, comando no reconocido.
501	Error de sintaxis en parámetros o argumentos.
502	Comando no implementado.
503	Secuencia de comandos errónea.
504	Comando no implementado para ese parámetro.
530	No se inició acceso.
532	Necesita cuenta para trasmitir datos.
550	Acción solicitada no se tomó.
551	Abortada acción solicitada. Se desconoce tipo de dispositivo.
552	Abortada acción de dato solicitado.

debe llevarse a cabo para que en todo momento se estén transmitiendo los datos al servidor de la red.

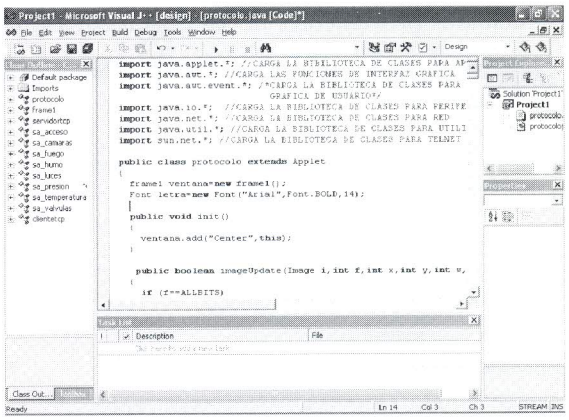

Figura 8. Clases de dispositivos IP "Inteligentes".

En la parte izquierda de esta ventana se puede visualizar cada una de las clases que conforman toda la red de dispositivos IP inteligentes. Cada uno de éstos es creado para conformar cada una de las clases.

En la parte superior se observan las bibliotecas cargadas al inicio del programa.

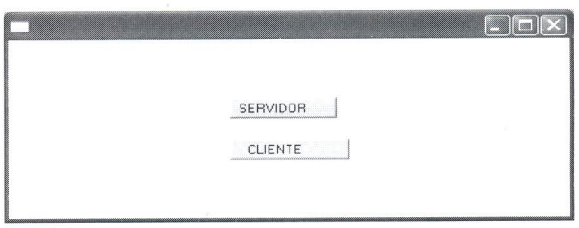

Figura 9. Ventana de selección del cliente Servidor.

Dependiendo de quién va a ejecutar la interfaz gráfica, se debe seleccionar en esta ventana qué computador realizará la aplicación como servidor o cliente en la red. El primer computador será el servidor; el siguiente será el cliente.

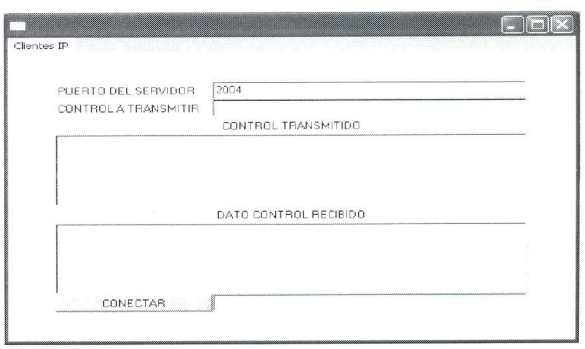

Figura 10. Ventana principal del menú clientes IP.

En esta ventana se puede apreciar el puerto del servidor escogido: 2004. También se pueden observar los campos de texto para transmitir datos de control y visualizar los datos recibidos provenientes de los dispositivos IP.

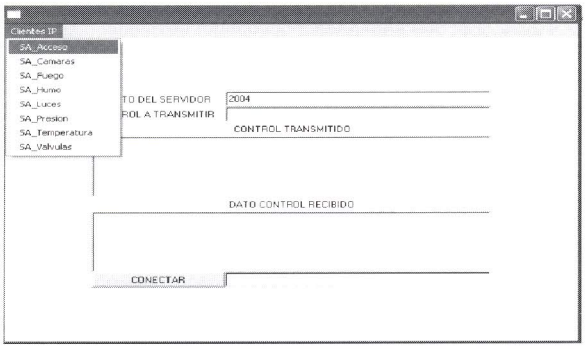

Figura 11. Ventana principal del menú clientes IP para selección de SA_Acceso.

El programa posee en su interior el registro de los posibles usuarios autorizados para ingresar o no a las instalaciones del sistema creado. En caso de que el usuario sea un funcionario o esté habilitado puede ingresar y será generado un mensaje de "ACEPTADO" con el cual se comprueba que el usuario ha realizado el respectivo acceso por la puerta controlada con la respectiva dirección IP asociada.

Finalmente, se aprecia la ventana servidor donde se conoce el estado del usuario, por qué puerta realizó el acceso, y los datos como la dirección IP y el puerto por el cual se conecta.

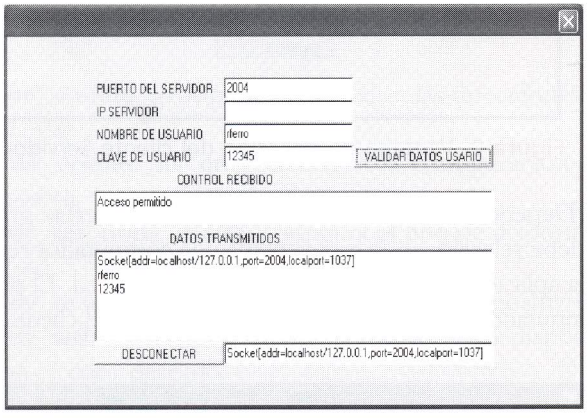

Figura 12. Ventana principal de SA_Acceso y validación de datos del usuario.

Conclusiones

Una vez realizado el presente artículo de investigación, se presentaron las bases fundamentales para la creación del protocolo IDTP teniendo en cuenta los fundamentos establecidos en el estándar EIA/IS-709.X. Con este protocolo se pretende crear el medio de comunicación para el gobierno de los sistemas de control de redes ques que utilizan

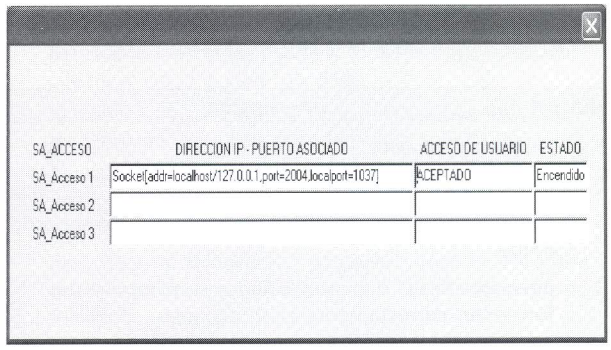

Figura 13. Datos observados en el servidor de red.

dispositivos IP Inteligentes, como sensores de humo, sensores de temperatura o cualquier tipo de sensor cuyo objetivo sea realizar la función de controlar todo tipo de variable física o electrónica, muy importante hoy día en domótica, la industria y la construcción de edificios inteligentes.

La importancia del protocolo propuesto radica en el uso de Sockets Stream, ya que su funcionalidad permite la transmisión confiable de datos, la comunicación a través de Internet por cualquier puerto de comunicaciones en el servidor y que el flujo de paquetes sea pequeño, lo que aumenta la velocidad de la red.

Puesto en funcionamiento el protocolo con la ayuda de herramientas como UML y Racional Rose, se puede crear el código inverso en cualquier aplicativo de desarrollo de software y desde allí se generan las ventanas de aplicación.

Los sistemas de distribución se refieren a un conjunto de procesos autónomos, como mecanismos de almacenamiento de datos. La distribución de tareas se genera en procesos con acciones ejecutadas al mismo tiempo con intercambio de mensajes (datos y comandos) para transferir información a través de la red de comunicaciones para la distribución de procesos, controlando el almacenamiento de datos.

La integración de hardware inteligente con programas adecuados permite la realización de la integración de cualquier dispositivo Ip para ser conectado en la red.

Referencias

[1] Bandel, D. *Linux*. Editorial Prentice Hall, 2000, capítulos 1-10, 12, pp. 3-299.

[2] Becerra, C. *Los 600 Métodos de Java*, Editorial Kimpres, 2001, capítulos 1-10, 12, pp. 1-168, 307-324, 509-702.

[3] Cheesman J. y Daniels, J. *UML Components: A simple Process for Specifying Component Based Software*, Addison-Wesley, 2000, pp 13-109.

[4] Loy, D. *Open control Networks*, Editorial Kluwer Academia Publishers, 2001, capítulos 2, 3, 4, 6, 7, pp. 43-189.

[5] Mark, M. *Lan Protocol Handbook*, Mand Publishing, 1992, pp. 35-97.

[6] Meyers, N. *Programacion Java en Linux*, Editorial Prentice Hall, 2000, capítulos 1-5, 6, 10, 12, pp. 3-185, 351-397.