CIENCIA E INGENIERÍA NEOGRANADINA, VOL 22-1, PP 151 - 164, BOGOTÁ JUNIO DE 2012, ISSN 0124-8170

COMUNICACIÓN ENTRE UN ESCÁNER DEVICENET AUTÓNOMO Y UNA APLICACIÓN SOFTWARE MEDIANTE MODBUS/TCP

A LINK BETWEEN AN AUTONOMOUS DEVICENET SCANNER AND SOFTWARE VIA MODBUS/TCP

Asfur Barandica López
Magister en Electrónica., Profesor asistente,
Escuela de Ingeniería Eléctrica y electrónica, Universidad del Valle,
asfur.barandica@correounivalle.edu.co

Edwin Andrés León Castro
Ingeniero Electrónico. Escuela de Ingeniería Eléctrica y electrónica, Universidad del Valle,
edaleon@hotmail.es

Erik Javier Bravo Ruano
Ingeniero Electrónico. Escuela de Ingeniería Eléctrica y electrónica, Universidad del Valle,
erikjavierbravo@hotmail.com

Fecha de recepción: 15 de diciembre de 2011
Fecha de aprobación: 28 de mayo de 2012

RESUMEN

Las redes industriales para automatización han experimentado un gran auge en los últimos años y DeviceNet es uno de los protocolos con mayor popularidad. En una red DeviceNet se identifican dispositivos esclavos, por lo general sensores y actuadores, y al menos, un dispositivo escáner encargado de configurar y administrar las transacciones en la red. El escáner debe comunicarse directamente con los dispositivos de la red DeviceNet y entregar la información recolectada a un dispositivo de mayor nivel jerárquico como un PLC o un PC. Aunque existen especificaciones y recomendaciones para elaborar un dispositivo esclavo DeviceNet, no hay una pauta clara de cómo implementar el maestro o escáner DeviceNet, ni tampoco para diseñar el enlace de comunicación entre el escáner y el equipo de control. En este artículo, se presenta un esquema de comunicación diseñado especialmente para soportar el intercambio de datos entre un escáner DeviceNet y una aplicación software residente en un PC.

Palabras clave: DeviceNet, escáner, Modbus/tcp, protocolos de comunicación, buses a nivel de dispositivos.

ABSTRACT

Industrial automation networks recently have boomed, including DeviceNet as one of the most popular protocols. In a DeviceNet network, slave devices usually sensors and actuators are identified, and at least a scanner in charge of configuring and managing network transactions. Such machine must communicate directly to other devices on DeviceNet network and deliver the collected information to send it to a higher hierarchical level equipment, such as a PLC or a PC. Although there are specifications and recommendations to build a DeviceNet slave, there is not a clear pattern to implement the DeviceNet master or scanner nor a design of the communication link between the scanner and control equipment. In this paper we show a communication scheme specifically designed to support data exchange between a DeviceNet scanner and a software resident on a PC.

Keywords: DeviceNet, scanner, MODBUS/TCP, communication protocols, device-level buses.

INTRODUCCIÓN

DeviceNet es una de las redes de comunicación más comunes en el ambiente industrial automatizado [1, 2]. En términos generales, es un protocolo abierto de comunicación de dispositivos de nivel bajo, que proporciona comunicación entre equipos industriales simples (actuadores, sensores), y equipos de nivel alto (controladores), desarrollado por Allen Bradley en 1993 y administrado actualmente por la ODVA (Open DeviceNet Vendor Association). Se caracteriza por el bajo costo en su implementación, por utilizar sólo cuatro hilos para conectar todos los dispositivos. Por ser un protocolo abierto, es fácil lograr que dispositivos de diversos fabricantes funcionen sin problemas en una misma red DeviceNet.

Un escáner es el dispositivo de la red que hace las veces de maestro, facilitando las tareas de detección y configuración de los demás dispositivos denominados esclavos. Además, debe enviar y recibir los datos de proceso (entradas y salidas), de cada dispositivo DeviceNet e intercambiar esta información con un equipo de mayor nivel como un PC o un PLC.

Numerosas publicaciones presentan arquitecturas para desarrollar esclavos DeviceNet [3, 4, 5], mientras los reportes relativos al desarrollo de escáneres son escasos. La mayoría de los escáneres comercialmente disponibles provienen de los grandes fabricantes de automatización y han sido desarrollados para acoplarse como módulos de PLCs, lo cual los hace costosos, pues debe adquirirse las herramientas software del mismo fabricante. Un escáner autónomo (stand-alone), evita la necesidad del hardware de soporte de los sistemas típicos. Sin embargo, debe contar con mecanismos estandarizados de comunicación con los equipos de control, para permitir la utilización del aplicativo software que exista en las empresas. El estándar OPC proporciona el elemento de enlace que independiza las aplicaciones software del hardware, con la inclusión de una aplicación denominada Servidor OPC.

Mei Liu Fu Song [6] desarrolló un escáner con una plataforma basada en ARM9 y WindowsCE. Net. Este artículo describe la arquitectura del escáner sin indicar los mecanismos para intercambiar información con los dispositivos de nivel superior. Baohua Tani et al [7] creó un sistema completo que permite el acceso por Internet a la red de dispositivos DeviceNet con aplicaciones convencionales para navegación. En este sistema, el hardware del escáner se implementó por medio de una tarjeta PCI al interior de un PC; los autores no utilizaron la tecnología OPC, y se limitó la posibilidad de intercambiar datos con aplicaciones tipo SCADA. Li Dongjiang Sun Ruiqi [8] trae un ejemplo de la comunicación entre una aplicación tipo SCADA y un servidor OPC con el protocolo Modbus/TCP, con el cual se accede a dispositivos industriales que se comunican usando dicho protocolo. Aun así, no considera las especificaciones del protocolo DeviceNet.

En el presente artículo, se propone un esquema de comunicación basado en el protocolo Modbus/TCP, mediante el cual es posible un intercambio eficiente de información entre un escáner DeviceNet y un servidor OPC instalado en un PC. El esquema adaptado a las particularidades de DeviceNet, permite configurar el escáner y la red, e intercambiar datos con los dispositivos esclavos desde las aplicaciones cliente OPC residentes en el PC.

1. DESCRIPCIÓN DEL SISTEMA

Una red DeviceNet se compone de hasta 64 nodos, uno de los cuales es el maestro. Cada nodo debe tener una dirección física única entre 0 y 63, denominada dirección MAC (Medium Access Control). La labor del maestro o escáner es detectar los dispositivos presentes en la red, configurarlos y establecer el intercambio de datos de entrada y salida. Para detectar y configurar los esclavos, se emplean tramas especiales conocidas como mensajería explícita. Para intercambiar los datos necesarios para monitorear y controlar el proceso, se usa la mensajería I/O, de la cual hacen parte las denominadas conexiones predefinidas maestro-esclavo, un conjunto de tramas de función específica que facilitan el intercambio de datos de proceso bajo diferentes modos de operación: por solicitud del maestro (modos poll o bit strobe), ante el cambio de estado de la variable (modo COS -Change of state-), o con una periodicidad fija (modo cyclic). Este escenario se presenta en la red DeviceNet, mediante el bus CAN (Controller Area Network).

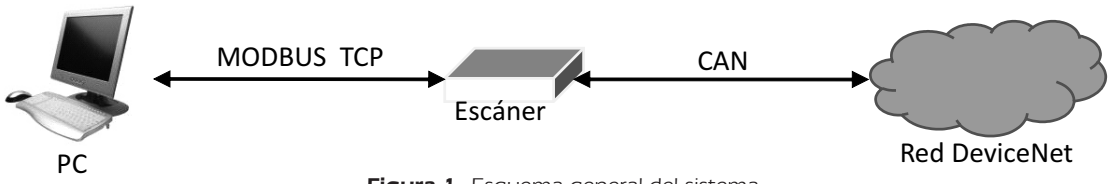


Figura 1. Esquema general del sistema

Por otra parte, toda la información de configuración y de proceso de cada dispositivo de la red DeviceNet, debe ser puesta a disposición de un equipo de mayor nivel jerárquico, para lo cual se

define un protocolo de comunicación entre el escáner y el dispositivo. Para el caso que se presenta en este artículo (Figura 1), el equipo de jerarquía superior es un PC y el protocolo de comunicación definido es MODBUS/TCP con Ethernet en las capas inferiores, debido a ser un protocolo abierto de baja complejidad que ofrece alta velocidad y confiabilidad [9].

MODBUS efectúa el intercambio de información mediante la lectura y escritura de bits individuales o de registros de 16 bits. Utiliza el modelo cliente-servidor para realizar las transacciones; en el sistema bajo estudio, el cliente MODBUS es el PC y el escáner hace las veces de servidor MODBUS. El escáner soporta las funciones MODBUS de lectura y escritura de registros (comandos 03H, 06H y 10H) [10]. La información se intercambia por medio de un espacio de memoria compartida de 130 Kbytes organizada en registros de 16 bits (2 bytes). Así, el escáner DeviceNet se comunica directamente con los dispositivos esclavos usando el conjunto de conexiones predefinidas maestro/esclavo establecido en la especificación DeviceNet y se comunica con el PC a través de Ethernet, según la especificación MODBUS/TCP.

Debido a que sólo se implementan funciones de lectura o escritura de registros, se definió un bloque de memoria en donde se ubican registros de funciones específicas que le permiten al usuario, ejecutar órdenes y recibir información acerca del estado de operación del escáner y la red DeviceNet. La memoria se dividió en tres bloques principales como se muestra en la Figura 2.

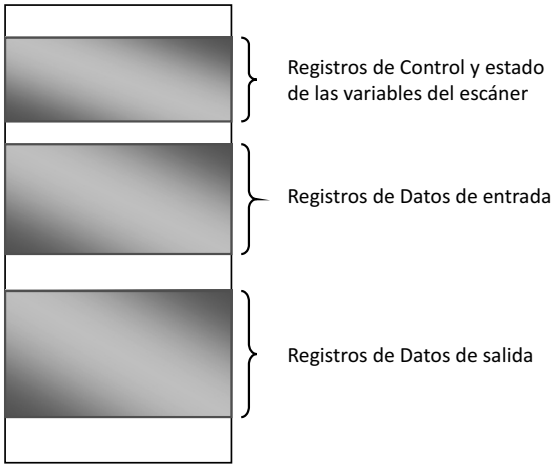


Figura 2. Organización de la memoria del escáner

En el primer bloque, *Registros de control, estado y variables del escáner,* se encuentra la información específica del estado funcional del escáner y la red DeviceNet, como también los registros dedicados mediante los cuales se solicita al escáner la ejecución de acciones; es decir, si la aplicación software desea ejecutar alguna acción sobre la red DeviceNet, deberá escribir un valor determinado sobre un registro específico de este bloque. La aplicación del escáner lee

el dato escrito en el registro, ejecuta la acción y modifica el valor de otro registro para informar sobre el resultado de la acción. Los bloques *Registros de datos de entrada y Registros de datos de salida* son destinados a compartir la información de los datos de entrada y salida de cada esclavo presente en la red DeviceNet.

1.1. PARTICIÓN DE LA MEMORIA

En la Figura 3, se muestra el mapa de memoria del escáner DeviceNet. Desde el PC, se le pueda ordenar al escáner que ejecute las siguientes acciones:

- Buscar todos los nodos presentes en la red DeviceNet y poner a disposición, la información sobre la cantidad y el tipo de dispositivos encontrados.
- Establecer conexiones I/O, enviar y recibir los datos de entrada y salida de cada esclavo.
- Enviar solicitudes y recibir respuestas de mensajería explícita.

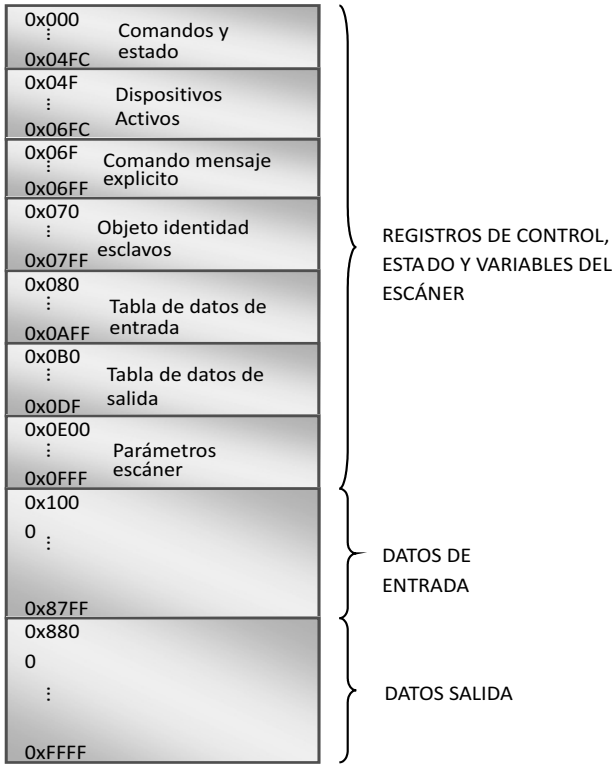


Figura 3. Mapa de memoria de escáner.

En el mapa de memoria, se encuentran los registros y tablas de datos necesarios para realizar cualquier acción. Utilizando correctamente cada registro, se logra comunicar el PC con la red DeviceNet a través de la interfaz Ethernet del escáner.

1.1.1. Registros de comando y estado

Este bloque tiene un tamaño de 1.276 registros (desde 0x0000 hasta 0x04FC), los cuales indican la solicitud de alguna acción del computador e informan el estado de operación y errores en el escáner. En la Tabla 1, se especifica cada registro dedicado a ejercer control sobre el escáner.

Tabla 1. Registros de comando y estado

NOMBRE DEL REGISTRO (posición)	DESCRIPCIÓN FUNCIONAL DEL REGISTRO	SIGNIFICADO DEL VALOR DEL REGISTRO
Comando De escaneo (0x0001) *Registro modificado por MODBUS*	Ordena al escáner iniciar la búsqueda de los dispositivos presentes en la red.	00= no hace algo 01= escanear la red completa. Posiciones 00-63. 02= escanear posiciones 00-07 03= escanear posiciones 08-15 04= escanear posiciones 16-23 05= escanear posiciones 24-31 06= escanear posiciones 32-39 07= escanear posiciones 40-47 08= escanear posiciones 48-56 09= escanear posiciones 57-63 10= escanear posiciones 00-31 11= escanear posiciones 32-63

Estado de escaneo (0x0002)
 	
Registro modificado por escáner
	Informa al computador el estado en donde se encuentra la petición de escaneo.	00= comando en ejecución
		1-63= # esclavos encontrados (repetido en 2 bytes)
		0x94= comando de escaneo erróneo
		0x64= no hay esclavos
		#diferente = error

Estado Mensajería Explícita (0x0003) *Registro modificado por MODBUS y el escáner*	Ordena al escáner, procesar un mensaje explícito. Informa al computador el estado en donde se encuentra la petición	0x00 = en espera de una petición 0x0F = valor escrito por el usuario para indicar que hay lista una petición de mensajería explícita. 0xFF = valor escrito por el scanner para indicar que fue leída la petición, pero se está procesando. 0x55 = el escáner indica que la respuesta está lista
MAC no existente (0x0004) *Registro modificado por escáner*	Informa al computador que hay un error en el valor de la MAC a donde se quiere enviar un mensaje.	0xAA= escrito en el byte 1 Valor de la MAC errónea=escrito en el byte 2 0x00= No hay error

Revisar Tabla Dispositivos Activos (0x0005) *Registro modificado por MODBUS y el escáner*	Ordena al escáner revisar la tabla de dispositivos activos. Informa al computador cuando la tabla ha sido revisada.	0x00= no hace algo. 0xF0= valor escrito por el usuario, indicando que se modificó la tabla de dispositivos activos. 0x0F= valor escrito por el escáner, indicando que la tabla fue revisada y actualizada.

1.1.2. Dispositivos activos

En este segmento de memoria de 512 registros (desde 0x04FD hasta 0x06FC) se encuentran los dispositivos esclavos que además de estar presentes en la red se encuentran activos, es decir, están interactuando con el escáner a través de la mensajería I/O.

Tabla 2. Formato dispositivos activos

MAC	TIPO DE CONEXIÓN	TAMAÑO DATOS DE ENTRADA	TAMAÑO DATOS DE SALIDA
1 byte	1 byte	1 byte	byte

El formato de este rango de memoria contiene cuatro campos de un byte, como se observa en la Tabla 2. El primer campo corresponde a la MAC del dispositivo esclavo. El segundo corresponde al modo de conexión; en él se indica si la conexión actual es en modo *Bit Strobe* (0x03), *Poll* (0x3C), *Cylcic* (0xC4) o *COS* (0xB7). Un nodo podría aparecer varias veces en este listado, en el caso de tener más de una conexión activa. Los campos tercero y cuarto corresponden al tamaño de los datos de entrada y de salida con los cuales fue configurada esta conexión.

1.1.3 Comando mensaje explícito

Este bloque está constituido por tres registros (direcciones 0x06FD, 0x06FE y 0x06FF). Se utiliza para proporcionarle la información necesaria al escáner y enviar el mensaje explícito. El escáner usa estos mismos registros para reportar la respuesta del mensaje explícito.

Tabla 3. Formato comando de mensajería explícita

MAC	CÓDIGO DE SERVICIO	CLASE ID	INSTANCIA ID	ATRIBUTO ID	VALOR
1 byte	1 byte	1 byte	1 byte	1 byte	1 byte

DeviceNet ha sido definido mediante el modelo de objetos, por lo cual cada valor que se desee leer o modificar, debe ser direccionado apropiadamente. Acogiendo las definiciones de la especificación, el formato establecido para los registros de mensajería explícita se muestra en la

Tabla 3. En primer lugar, se encuentra el valor de la MAC del dispositivo seguido por el código del servicio, ya sea para obtener o modificar el valor de un atributo; estos códigos de servicio son determinados por la especificación DeviceNet. La *clase ID* hace referencia al objeto en donde se encuentra el atributo; por ejemplo: el atributo *tipo de dispositivo* está dentro del objeto *identidad*. De igual forma, se hace para la instancia y el atributo.

El comando de mensajería explícita trabaja en conjunto con el registro llamado estado mensajería explícita descrita en la Tabla 1. Mediante este registro, el cliente conoce el estado en el cual se encuentra el escáner con respecto del mensaje explícito en proceso.

1.1.4. Objeto identidad esclavos

En este espacio de memoria, se almacenan los atributos necesarios de todos los esclavos encontrados durante el escaneo, para que la aplicación del PC busque los archivos EDS (hojas de datos electrónicas), correspondientes a cada esclavo. Esta acción se hace una vez se ha dado la orden de escanear la red. El formato del objeto identidad de cada esclavo se muestra en la Tabla 4.

Tabla 4. Formato objeto identidad esclavos

MAC ID	VENDOR ID	TIPO DE DISPOSITIVO	CÓDIGO DE PRODUCTO	REVISIÓN MAYOR	REVISIÓN MENOR
1 byte	2 bytes	2 bytes	1 byte	1 byte	byte

1.1.5 Tablas de datos de entrada y salida

Para facilitar la comunicación con el cliente MODBUS y optimizar el uso de la memoria de modo que se ajuste dinámicamente a las características de la red, se han definido unas tablas de datos que informan sobre la ubicación de los datos de entrada y salida en la memoria; de esta manera, para acceder a los datos ya sea de entrada o salida de un esclavo, primero se debe consultar la tabla de datos para saber la dirección de la memoria del escáner en donde se localiza la información y la cantidad de datos que se deben leer. Debe resaltarse que estas tablas de datos deben ser de sólo lectura para el cliente MODBUS, puesto que el escáner es el responsable de actualizarlas.

Tabla 5. Formato tabla de datos de entrada o salida

MAC ID	TIPO DE CONEXIÓN	CANTIDAD DE DATOS	DIRECCIÓN PARTE ALTA	DIRECCIÓN PARTE BAJA
1 byte	1 byte	1 byte	1 byte	byte

El formato de cada campo en este rango de memoria, se muestra en la Tabla 5. La tabla de datos de entrada se encuentra desde la dirección 0x0800 hasta 0x0AFF y la tabla de datos de salida se encuentra desde 0x0B00 hasta 0x0DFF.

1.1.6 Parámetros del escáner

En este rango de memoria (desde 0x0E00 hasta 0x0FFF), están los datos de configuración del escáner tales como la MAC, la velocidad, fabricante, serial, etc.

1.2. ESCANEAR LA RED DEVICENET

Para ejecutar la acción de escanear la red, se utilizan dos registros denominados *comando de escaneo y estado de escaneo.* Por lo tanto, cuando el PC desee realizar un escaneo en la red, debe escribir en el registro *comando de escaneo,* el valor correspondiente al rango de direcciones que desee escanear (Tabla 1). Después de haber escrito un valor válido en este registro, el escáner inicia el barrido de la red e informa al PC por medio del registro *estado de escaneo,* si ya terminó de escanear la red o aún se encuentra ocupado.

Una vez el PC haya sido informado del número de esclavos que hay en la red, debe usar los comandos de lectura MODBUS para revisar el bloque de memoria correspondiente a *Objeto Identidad esclavos*, ya que en este bloque de memoria, el escáner guarda la información correspondiente al objeto identidad de cada esclavo detectado. La información es necesaria para que la aplicación busque en su base de datos, la hoja de datos del dispositivo que permite identificarlo y suministrar al usuario información como el tipo de dispositivo, fabricante, cantidad de datos de entrada o salida, modos de operación, interpretación de los datos I/O, etc.

1.3. ESTABLECIMIENTO DE CONEXIONES I/O

Se recomienda que el escáner DeviceNet implemente los cuatro modos de comunicación I/O (Poll, Cyclic, COS y Bit strobe), para soportar la mayoría de esclavos, ya que los desarrolladores de dispositivos DeviceNet son libres de escoger el tipo de conexión I/O que deseen. Para establecer conexiones I/O, se creó el bloque *Dispositivos Activos*, en donde se le indica al escáner, los esclavos con los cuales se va a establecer intercambio de datos I/O. Este bloque es una tabla en donde se especifica el esclavo, el tipo de conexión I/O por establecer, y la cantidad de datos I/O. Cuando la aplicación software termina la escritura de datos en este bloque de memoria, debe ordenar al escáner que revise la tabla de *dispositivos activos* y establezca las conexiones I/O requeridas.

Para ordenar al escáner la ejecución de esta acción, existe el registro *Revisar tabla de Dispositivos Activos* (Tabla 1), donde el usuario debe escribir el valor indicado y esperar que el escáner revise la tabla y reporte la finalización de la tarea. Cuando el escáner está revisando la tabla de dispositivos y encuentra una conexión válida, intenta establecerla con el dispositivo. Si la conexión es exitosa, va generando automáticamente la tabla de datos de entrada y salida. En estas tablas, se encuentra la ubicación en donde se van a almacenar los datos de entrada o salida.

1.4. MENSAJERÍA EXPLÍCITA

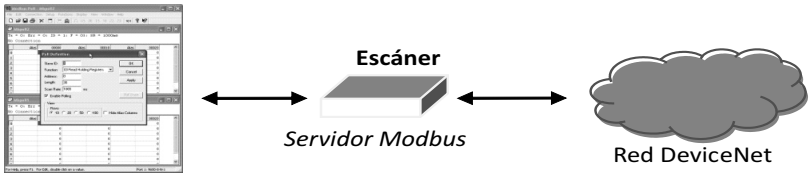
Para enviar un mensaje explícito, primero hay que especificar la MAC, la clase, la instancia, el atributo y el código de servicio en el bloque Comando Mensaje Explícito. Es entonces cuando se ordena al escáner, enviar el mensaje explícito mediante el registro Estado Mensajería Explícita y se espera hasta que la operación termine. La respuesta del mensaje explícito se debe encontrar en el bloque Comando Mensaje Explícito.

2. RESULTADOS Y DISCUSIÓN

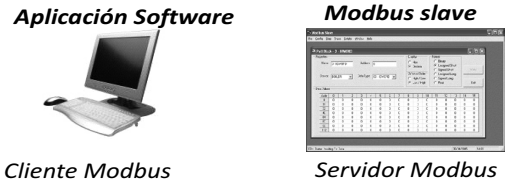
Se desarrolló un escáner DeviceNet basado en la plataforma AT91SAM7X-EK, sobre el cual se implementó MODBUS/TCP [11]. De forma paralela, se desarrolló la aplicación software que además de permitir la configuración del escáner, hace las veces de servidor OPC (OLE for Process Control), que describe detalladamente Reina Pérez y Ruíz Olaya [12].

En cuanto al servidor OPC, la aplicación software cumple con los requerimientos para intercambiar información con clientes OPC, de acuerdo con la especificación OPC Data Access 3.0, y consta básicamente de una interfaz de comunicación con el escáner, el espacio de direccionamiento y la interfaz gráfica de usuario. Por su parte, la herramienta de configuración dispone de una amigable interfaz gráfica que utiliza el contenido de un archivo EDS (Electronic Data Sheet), para llevar a cabo los cambios en los parámetros accesibles al usuario de cualquier dispositivo en la red DeviceNet.

En principio, tanto la aplicación software como el escáner, se probaron de forma independiente, utilizando software de soporte para establecer las conexiones por MODBUS/TCP y emular el comportamiento de cada uno, como se observa en la Figura 4.



a) Pruebas escáner



b) Pruebas aplicación software

Figura 4. Pruebas Iniciales del sistema desarrollado

Para aplicación software, se utilizó el programa Modbus slave 4.6.1, que es un servidor MODBUS que permitió emular y manipular el mapa de memoria del escáner.

Para el escáner, se usó el programa Modbus Poll 4, 3, 4, que es un cliente Modbus que, permite observar todas las tramas de la comunicación y el bloque completo de memoria del escáner.

Para medir la calidad de los enlaces, se varió el tiempo de espera de cada mensaje (time out), para Modbus y se hicieron peticiones repetidas, de acuerdo con el time out ajustado. Para el escáner, se registraron los datos que se exponen en la Tabla 6.

Tabla 6. Pruebas comunicación MODBUS del escáner.

TIEMPO ENVÍO ENTRE CADA PETICIÓN (ms)	PAQUETES ENVIADOS	NÚMERO DE ERRORES	PORCENTAJE DE ERROR
1000	800	0	0%
500	958	0	0%
400	896	0	0%
300	789	16	2.03%
250	811	68	8.38%
100	865	158	18.26%

En el caso de la aplicación software, no se presentaron errores en la misma prueba, aunque a diferencia del escáner, tanto la aplicación software como el servidor Modbus se encontraban instalados en el mismo equipo.

Todos los errores presentados durante las pruebas con el escáner, fueron por time-out, es decir, que el escáner no respondió los mensajes dentro del tiempo establecido. Esto se debe principalmente a la arquitectura de programación implementada en la gestión directa de la red DeviceNet con el escáner y a que la plataforma AT91SAM7XEK no cuenta con la librería TCP completa. Teniendo en cuenta que no se presentaron errores de otro tipo con el protocolo propuesto, se puede afirmar que con una plataforma hardware de mejor desempeño y con un stack TCP/IP más robusto, se puede mejorar las prestaciones del enlace Modbus/TCP.

Tabla 7. Esclavos DeviceNet presentes en la red

No. ESCLAVO	DESCRIPCIÓN	FABRICANTE
1	Sensor de proximidad	Allen Bradley
2	Sensor de contacto	Allen Bradley
3	Módulo DeviceNet PLC koyo	Koyo
4	Gateway Hart-DeviceNet	TICI

Posteriormente, la aplicación y el escáner fueron acoplados con un time out de 400 ms para mostrar el comportamiento previsto por los diseñadores. Para realizar las pruebas con el sistema completo, se contó con los esclavos descritos en la Tabla 7, y se comparó el desempeño del sistema desarrollado con uno comercial.

Se probaron con éxito, las tareas de escaneo en todos sus rangos, mensajería explícita y mensajería I/O. Los tiempos de escaneo fueron mucho menores que los requeridos por un escáner Allen Bradley 1756-DNB en las mismas condiciones. En este caso, los tiempos son del orden de decenas de segundos, por lo cual se utilizó un reloj convencional. Mientras el escáner de Allen Bradley tardó entre 2 ó 3 minutos en detectar los dispositivos presentes en la red, el sistema basado en AT91SAM7X-EK tardó cerca de un minuto.

Los parámetros y pruebas desarrolladas para la mensajería I/O y la mensajería explícita se hicieron acordes con la especificación DeviceNet, cumpliendo los tiempos definidos para este tipo de conexiones. Para el envío de mensajes explícitos, se percibió el mismo tiempo de respuesta, comparando la interfaz de configuración de la aplicación software desarrollada con la interfaz de configuración para el envío de mensajes explícitos del escáner Allen Bradley, RSNetWorx for DeviceNet; tiempo tomado desde cuando se ejecutó la opción de enviar el mensaje explícito hasta que se visualizó la confirmación del envío y los datos recibidos.

Por último, los datos de la mensajería I/O fueron integrados al servidor OPC, colocando la aplicación LabView como cliente, para observar las variaciones en la medida del sensor de proximidad, el sensor de contacto y las entradas del PLC Koyo. Así mismo, se logró la activación de las salidas del PLC.

El mecanismo de comunicación propuesto también fue sometido a condiciones típicas de falla, como la pérdida repentina de alimentación del escáner, ruptura del enlace de comunicación DeviceNet o Ethernet y suspensión inesperada de la aplicación software. En las situaciones que implicaban ausencia de comunicación con el PC, el escáner sostuvo las conexiones con los dispositivos de la red DeviceNet que estaban activas y, una vez recuperado el enlace, las comunicaciones con el PC se restablecieron sin perturbar el funcionamiento de la red DeviceNet. De igual forma, el escáner se recuperó satisfactoriamente y sin intervención humana, de la pérdida de alimentación o de los problemas que había en la red DeviceNet, informando sobre su estado a la aplicación software.

3. CONCLUSIONES

El esquema de comunicación descrito brinda autonomía y eficiencia para el desarrollo de un escáner DeviceNet autónomo, ya que la organización de la memoria es una guía para establecer las tareas y la estructura general del escáner.

Al utilizar un protocolo abierto como MODBUS/TCP para comunicar el PC con el escáner, es posible separar los desarrollos del hardware y del aplicativo software, facilitando la asignación de tareas y la actualización independiente de ambos subsistemas.

El protocolo implementado permitió que los datos de entrada y salida fueran asignados de forma dinámica, y se lograra mayor eficiencia en el uso de la memoria dedicada al intercambio de datos de entrada y salida, superando ampliamente propuestas previas [12].

El protocolo propuesto permitió acceder a una red DeviceNet desde el PC de forma transparente y satisfactoria, brindando al usuario un entorno amigable para configurar y administrar una red DeviceNet.

AGRADECIMIENTOS

La ODVA facilitó a la Universidad del Valle, la especificación DeviceNet para fines académicos, gracias a lo cual este tipo de desarrollos fue posible.

REFERENCIAS BIBLIOGRÁFICAS

[1] Spiegel R., (2008). PTO to Step Up Profinet Promotion. En: Automation World, septiembre, 25 p.

[2] Caro R., (2007). Fieldbus: ¿Where do we stand?. En: Intech, abril, 38 p.

[3] Suárez O., y Rosero A., (2008). Implementación de un módulo DeviceNet con sus 4 modos de comunicación. Trabajo de grado. Facultad de Ingeniería. Universidad del Valle. Santiago de Cali.

[4] Guohong Li, Cheng Xiao, and Zhuang Wu, (2011). Development and Aplication Control Network Based on DeviceNet. International Conference on Information Science and Technology, March 26-28. Nanjing, Jiangsu, China.

[5] Xiaoke Fang, Min Huang, Jianhui Wang and Shusheng Gu, (2004). Development of DeviceNet Fieldbus Intelligent Node. Proceedings of the 5" World Congress on Intelligent Control and Automation, June 15-19. Hangzhou, P.R. China.

[6] Mei Liu and Fu Song, (2010). DeviceNet Master Research Based on Embedded System. 2nd International Conference on Information Science and Engineering (ICISE). Dec. pp. 4-6

[7] Baohua Tani, and Juntao Wang, (2010). A DeviceNet Fieldbus Data Acquisition System Base don Flex Technology and RIA Model. IEEE International Conference on Progress in Informatics and Computing (PIC). Dec. pp. 10-12.

[8] Li Dongjiang and Sun Ruiqi, (2011). Implement of Communication between Configuration Software and OPC Server Based on Modbus/TCP. The Tenth International Conference on Electronic Measurement & Instruments (ICEMI).

[9] Ruíz Olaya A.F., Barandica López A., y Guerrero Moreno F.G., (2004). Implementación de una red MODBUS/TCP. En: Revista Ingeniería y Competitividad, Vol. 6 (2), 35 p.

[10] Modbus-IDA, (2006). MODBUS Application Protocol Specification V1.1b. En: http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf (Agosto de 2012).

[11] Bravo Ruano E. J., y León Castro E. A., (2010). Implementación de un escáner DeviceNet basado en la plataforma AT91SAM7X-EK. Trabajo de grado. Facultad de Ingeniería. Universidad del Valle. Santiago de Cali.

[12] Reina Pérez J.L., y Ruiz Olaya D., (2010). Desarrollo de una herramienta software para la configuración y gestión de información de un scanner DeviceNet. Trabajo de grado. Facultad de Ingeniería. Universidad del Valle. Santiago de Cali.