ASADES

Avances en Energías Renovables y Medio Ambiente
Vol. 16, 2012. Impreso en la Argentina. ISSN 0329-5184

IMPLEMENTACION DE MODULOS MODBUS INALAMBRICOS PARA CONTROL E INSTRUMENTACION EN SISTEMAS SOLARES

T. Moya, D. Hoyos.
INENCO (Instituto Nacional de Energías No Convencionales)
CIUNSa Consejo de Investigación Universidad Nacional de Salta
Tel. 0387-4255578 – Fax 0387-4255579 e-mail: tmoya@unsa.edu.ar

Recibido: 13/08/12; Aceptado: 02/10/12

RESUMEN: En este trabajo se presenta la implementación de módulos inalámbricos Zigbee controlados mediante MODBUS para instrumentación y control de sistemas solares. Se utilizó un microcontrolador de 8 bits para implementar la traducción de funciones MODBUS de la PC en tramas Zigbee para los módulos. La conexión a la PC es por USB emulando un puerto serie con lo que se logra una máxima compatibilidad con el software de control preexistente.

Palabras clave: energía solar, instrumentación, control, microcontroladores.

INTRODUCCIÓN

Para el control de concentradores solares del tipo fresnel(Mills, 2000) resulta necesaria una red de sensores conectada a una computadora con un programa HMI (interfase hombre máquina). Como medio físico de transmisión es preferible un medio inalámbrico ya que debido a las dimensiones de los concentradores los costos y el mantenimiento del cableado resultan considerables. Dentro del abanico de protocolos de campo disponibles MODBUS resulta ser el de mayor difusión y aceptación, mientras que como protocolo inalámbrico Zigbee resulta especialmente interesante debido a sus bajos requerimientos de procesador y su gran capacidad de ampliación. Si bien es posible utilizar Zigbee de una manera transparente para transportar las funciones MODBUS las pruebas anteriormente realizadas mostraron que el control directo de los módulos Zigbee mediante tramas API aumentaría considerablemente el desempeño de la red (Moya, 2010).
Este modo de control mejorado implicó implementar la traducción de funciones MODBUS en tramas Zigbee y viceversa, lo que a su vez conllevó a un detallado estudio de ambos protocolos para lograr compatibilizarlos en un microcontrolador, relativamente limitado en capacidad de procesamiento y memoria.
En el modelo propuesto el microcontrolador se conecta a un puerto USB de la PC y se comporta como un puerto serie virtual por el que usaremos funciones MODBUS RTU para controlar todas las funcionalidades de los módulos Xbee: entradas y salidas digitales, ADCs y PWMs de los módulos Xbee Pro, ver Fig. 1.

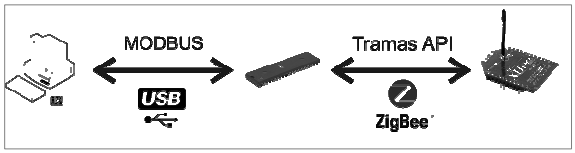

Fig. 1. PC, microcontrolador y módulo Xbee: Protocolos utilizados.

ZIGBEE

Descripción.
Zigbee es un conjunto de protocolos para redes WPAN (Wireless Personal Area Network) según IEEE 802.15.4-2003 (IEEE, 2003). Ofrece velocidades de hasta 250 Kbps y sus dispositivos prevén modos de muy bajo consumo. Entre sus aplicaciones más inmediatas se encuentran las redes de sensores. Permite tres topologías de red: estrella, árbol y malla, destacando ésta última que permite la existencia de más de un camino para los datos (Caprile, 2009).

Define tres tipos de dispositivos:

- Coordinador: Es el encargado de armar la red asignando direcciones a los demás dispositivos, también se comporta como router. Debe existir al menos uno por red.
- Router: Establece rutas de datos en la red posibilitando la interconexión de los demás dispositivos.
- Dispositivo final: Puede comunicarse con los 2 anteriores y normalmente lleva conectados los sensores y actuadores.

Según el direccionamiento las transmisiones Zigbee pueden ser unicast (punto a punto), multicast (punto multipunto) o broadcast (difusión).

Módulos Xbee.

Los módulos Xbee de Digi están implementados sobre un Zigbee SoC (System on Chip) EM250 de Ember el cual incluye un transceptor de 2.4GHz y un microprocesador XAP2b de 16 bits (Digi, 2009). Para facilitar el desarrollo el fabricante provee un stack Zigbee (EmberZNet PRO) sobre el que Digi implementó dos modos de control: comandos AT y tramas API.

El módulo podrá comportarse como Coordinador, Router o End Device según el firmware instalado. Digi provee un software para Windows llamado X-CTU que permite grabar el firmware y configurar los parámetros.

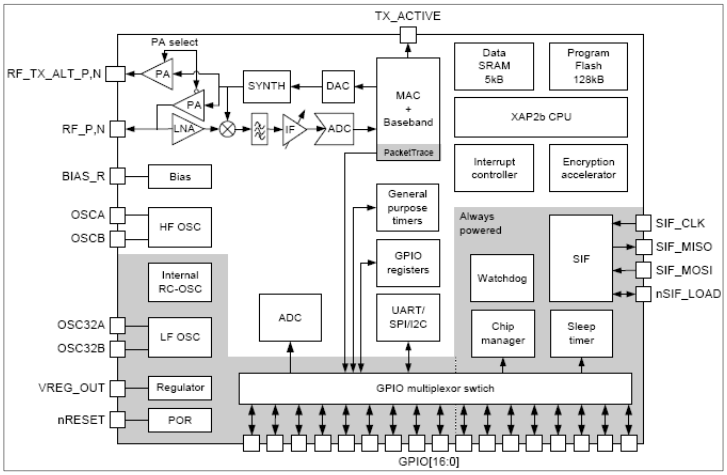

Fig. 2. Diagrama de bloques del EM250

Los módulos ofrecen 15 de los pines del XAP2b destinados a GPIO (Entrada Salida de Propósito General) que pueden configurarse como entradas/salidas digitales con o sin pull-up, ADCs o PWMs (sólo XBee Pro).
En la Fig. 2 se puede ver el diagrama de bloques del EM250 y en la Tabla 1 las principales características del procesador XAP2b (Ember Corporation, 2009).

Microprocesador XAP2b de 16 bits
Frecuencia de Operación: 12MHz
Memoria Flash: 128KB
Memoria RAM: 5KB
17 Pines de Entrada/Salida de propósito general (GPIO)
2 Controladores Serie con DMA
2 Timers de 16 bits
Conversor Analógico Digital Sigma Delta con 12 bits de resolución
Acelerador de encriptación AES integrado
Regulador de voltaje integrado (1.8 V)

Tabla 1: Características del microprocesador XAP2b

Direccionamiento Zigbee en Xbee.
El direccionamiento Zigbee utiliza dos direcciones para cada dispositivo:

- Una de 64 bits (identificador EUI-64) también llamada dirección extendida que es fija y única para cada dispositivo, asignada por el fabricante. Sus primeros 24 bits corresponden a un identificador de vendedor asignado por la IEEE (OUI: Organizationally Unique Identifier). En el caso de los módulos Xbee siempre es 00-13-A2 que corresponde a MaxStream Inc. (ahora Digi International Inc.). Los siguientes 40 bits son un espacio de direcciones definido por el vendedor que debe encargarse de asignar a cada dispositivo fabricado un valor diferente.
- Una de 16 bits que les es asignada por el coordinador o router al unirse a una red.

Para una transmisión unicast Zigbee necesita ambas direcciones, la de 64 bits para identificar inequívocamente al destinatario y la de 16 bits para rutear el mensaje (ZigBee Alliance, 2008). Cuando una transmisión sólo especifica la dirección de 64 bits entonces el stack Zigbee deberá encargarse de descubrir la dirección de 16 bits correspondiente.

Los módulos Xbee poseen una tabla de direcciones que mapea direcciones de 64 bits en direcciones de 16 bits (Digi, 2009).

Esta tabla almacena las últimas 10 entradas utilizadas y en caso de no contener la dirección necesaria el dispositivo deberá realizar un descubrimiento de dirección enviando paquetes broadcast lo cual le insume un cierto tiempo y por lo tanto reduce el rendimiento de la transmisión. Para mejorar la performance de la red el fabricante recomienda que las aplicaciones en las que un dispositivo realiza transmisiones unicast a más de 10 dispositivos implementen una tabla de direcciones propia y de tamaño suficiente a fin de que cada trama enviada ya incluya las direcciones de 16 y 64 bits.

MODBUS

Descripción.

MODBUS fue desarrollado por Modicon (ahora Schneider Electric) en 1979 como una manera de comunicarse con varios dispositivos sobre un único par trenzado. El esquema original corría sobre RS232 y también fue adaptado a RS485 para obtener mayores velocidades y distancias. MODBUS se convirtió rápidamente en un standard de facto en la automatización industrial y Modicon lo liberó al público como un protocolo sin royalties. Actualmente MODBUS-IDA es el mayor grupo organizado de usuarios y vendedores de MODBUS y continúa dando soporte al protocolo.

MODBUS es un sistema Master-Slave donde el Master se comunica con uno o varios Slaves. El Master típicamente es un PLC (Programmable Logic Controller) o una PC. Los Slaves MODBUS RTU normalmente son dispositivos de campo conectados a la red en una configuración multipunto. Cuando un Master MODBUS RTU necesita información de un Slave envía un mensaje que contiene la dirección del dispositivo, los datos que requiere y una suma de control para detección de errores. Cada uno de los dispositivos de la red ven el mensaje, pero sólo el dispositivo direccionado responde.
Los Slaves de una red MODBUS no pueden iniciar una comunicación, sólo responden. Algunos fabricantes desarrollaron dispositivos híbridos que actúan como Slaves MODBUS pero también poseen capacidad de escritura, por lo que se comportan como seudo-Masters.

Mensajes MODBUS.

Para comunicarse con un dispositivo Slave el Master envía un mensaje conteniendo:
- Device Address: Dirección del dispositivo
- Function Code: Código de Función
- Data: Datos
- Error Check: Chequeo de error

Device Address es un número de 0 a 65535 utilizado para direccionar los dispositivos, los mensajes enviados a la dirección 0 (mensajes broadcast) son recibidos por todos. Excepto para los mensajes broadcast, un esclavo siempre contesta a un mensaje MODBUS de tal manera que el Master sepa que el mensaje fue recibido.
Function Code define el comando que el Slave debe ejecutar, como leer datos, aceptar datos, reportar estado, etc. Los Function Codes van de 1 a 255.
Los Datos definen las direcciones en el mapa de memoria del dispositivo para operaciones de lectura, contienen los valores de los datos a escribir en la memoria del dispositivo o contienen cualquier otra información necesaria para llevar a cabo la función requerida.
El Error Check es un valor numérico de 16 bits que representa el Chequeo de Redundancia Cíclico (CRC). El CRC es generado por el Master y vuelto a calcular por el dispositivo receptor, si los valores de CRC no coinciden se solicita la retransmisión del mensaje.
Cuando el Slave realiza la función solicitada envía un mensaje de vuelta al Master. Este mensaje contiene la dirección del esclavo, el código de función solicitado (para que el Master sepa quien está contestando), los datos requeridos y un valor de Error Check.

Modelo de datos MODBUS.

El modelo de datos (Modbus-IDA, 2006) establece cuatro tablas primarias que permiten operaciones de lectura y escritura direccionando los datos por bit o por palabra de 16 bits, ver Tabla 2. Para cada tabla se permite seleccionar un elemento individual y también sus múltiples consecutivos.

Tabla primaria	Tipo de dato	Operación
Discrete Inputs	1 bit	Lectura
Coils	1 bit	Lectura Escritura
Input Registers	16 bits	Lectura
Holding Registers	16 bits	Lectura Escritura

Tabla 2: Tablas primarias Modbus

Funciones MODBUS.

Existen tres categorías de códigos de funciones MODBUS: públicas, definidas por el usuario y reservadas. De las públicas, que están bien definidas y validadas por la comunidad MODBUS-IDA.org, destacaremos sólo algunas:

Function 02: Read discrete inputs

Esta función puede usarse para solicitar de una vez el estado de varias entradas discretas (1 bit), esto se hace especificando la dirección y el rango de de las entradas a leer en el campo de datos del mensaje. Sólo es posible consultar a un dispositivo por vez, por lo tanto esta función no soporta direccionamiento broadcast.
La respuesta incluye un bit para cada entrada leída, el LSB del primer byte contiene la entrada direccionada en la consulta. Las consecutivas continúan hacia el MSB de ese byte y desde el LSB al MSB de los bytes siguientes. En caso de sobrar bits en el último byte se ponen a 0.

Function 03: Read holding registers
Los valores internos en un dispositivo MODBUS se guardan en registros de almacenamiento o "holding registers". Estos registros son de 2 bytes y pueden usarse para varios propósitos. Algunos registro contienen parámetros de configuración y otros se usan para valores medidos (temperaturas, etc.). La función 03 se utiliza para solicitar uno o varios registros de almacenamiento de un dispositivo. No admite direccionamiento broadcast. Los registros se direccionan desde 0: los registros 1 a 16 se direccionan como 0-15.
En la respuesta los datos de los registros se empaquetan en 2 bytes por registro con los datos binarios justificados a la derecha dentro de cada byte. Para cada registro el primer byte contiene los bits de mayor orden y el segundo byte los de menor orden.

Function 05: Force Single Coil
En el lenguaje Modbus un "coil" es un valor discreto (1 ó 0) de salida. Esta función fuerza a ON / OFF el valor de un coil determinado. Los colis se direccionan desde 0 que corresponde al coil 1. Soporta broadcast.
La respuesta es un eco de la solicitud que se envía una vez que se cambió el estado del coil.

Function 06: Preset single register
Esta función permite escribir en un determinado registro de almacenamiento (holding register) de un dispositivo el valor indicado en el campo de datos. Soporta brodcast.
La respuesta es un eco de la solicitud que se envía una vez que se cambió el estado del registro.

Function 17: Report Slave ID
Devuelve una descripción del tipo de dispositivo presente en la dirección de esclavo solicitada. No soporta broadcast.

IMPLEMENTACIÓN

Descripción.
El diseño contempla un único procesador central conectado al módulo Coordinador de la red el cual se comunica con los módulos remotos mediante tramas Zigbee.
El procesador se encarga de recibir los comandos MODBUS de la PC, interpretarlos, generar las tramas con los comandos para los módulos remotos y despacharlas a través del Coordinador Zigbee.
El gateway se conecta vía USB a una PC y se comporta como un puerto serie virtual utilizando un firmware que implementa la especificación USB Communication Device Class (CDC) (Microchip Technology, 2004). Esto le permite funcionar de manera totalmente transparente con todo el software existente que maneja MODBUS RTU.

Microcontrolador.
Se utilizó un microcontrolador de 8 bits PIC 18f4550 de Microchip el cual presenta las siguientes características destacables para esta implementación: soporte USB, 32 K de memoria de programa, 2 K de memoria RAM y voltaje de operación de 2 a 5,5 V.

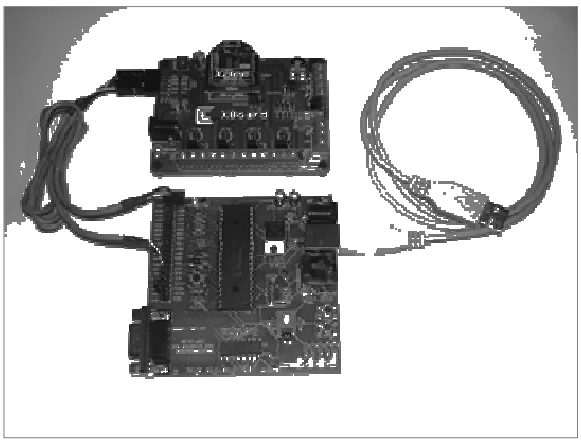

Fig. 3. Conexión de un kit de entrenamiento para PIC 18f4550 y un kit Xboard con módulo Xbee.

Firmware.
El firmware escrito en C 18 realiza las siguientes tareas:

- Gestionar comunicaciones USB

- Decodificar protocolo MODBUS
- Convertir funciones MODBUS en comandos Xbee
- Codificar tramas Zigbee
- Gestionar comunicaciones serie con módulo Xbee
- Decodificar tramas Zigbee
- Codificar mensajes de respuesta MODBUS
- Gestionar tabla de direcciones de dispositivos
- Gestionar errores MODBUS

Red Zigbee.
La red Zigbee implementada cuenta con un módulo como Coordinador (en el Gateway) al que se conectan los demás módulos, configurados como Routers o End Devices. Normalmente el ruteo será punto a punto, pero como existen Routers puede darse también una red mesh, según la ubicación y el alcance de los módulos, en cuyo caso el ruteo necesario lo gestionará el stack Zigbee de manera transparente para nosotros.

Control de los módulos Xbee.
Para el control de los módulos se utilizan tramas API debido a su mejor rendimiento y a que algunas funciones necesarias sólo están disponibles en este modo. El Coordinador envía tramas API que ejecutan comandos AT remotos (como cambiar el estado de un pin por ejemplo) y en el caso de operaciones de lectura los remotos devuelven la información también en tramas API.

Modo de ahorro de energía de los módulos.
Como en esta aplicación no es deseable que los dispositivos hibernen (ocasionando timeouts al protocolo MODBUS) se prefirió programar los módulos remotos con firmware Router que, a diferencia del End Device, permite deshabilitar las opciones de ahorro de energía.

Configuración de los pines Xbee.
La función de cada pin de los módulos es fija durante la marcha del dispositivo, pero previo a su uso se la debe establecer, ya sea DIO, ADC o PWM. En la Fig. 4 se puede observar, por ejemplo, la configuración con X-CTU de D3 como Entrada Digital. Con la opción "Remote Configuration" de X-CTU es posible configurar inalámbricamente cualquier dispositivo de la red Zigbee.

Modo "bridge".
En el firmware se dejó previsto el uso de un modo denominado "bridge" (puente), en el cual el procesador repite todo lo que recibe por el puerto USB (en modo CDC) en su UART y viceversa. Esto permite configurar con X-CTU en modo "Remote Configuration" los módulos sin tener que desconectar el coordinador del microcontrolador.
Para seleccionar el modo "bridge" deberá ponerse a masa el pin 4 del microcontrolador al momento del reset.

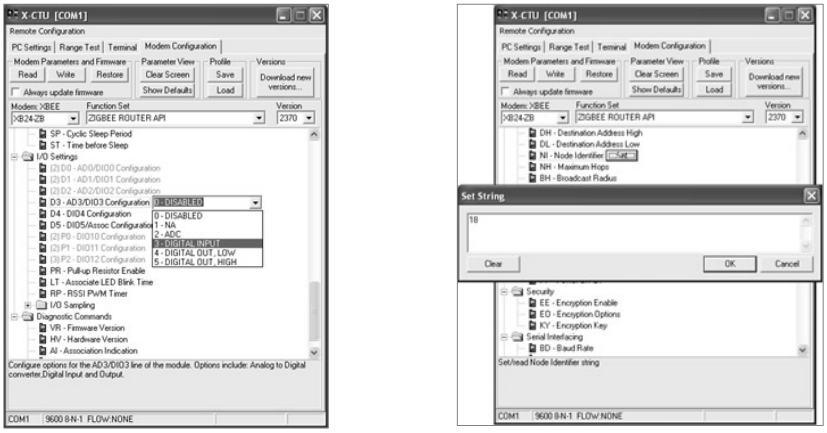

Fig. 4. Uso de X-CTU para configuración de los pines del módulo y la ID de Esclavo MODBUS.

MODBUS ID de los módulos.
Se utilizó la cadena NI (Node Identifier) de los módulos Xbee para almacenar su ID de esclavo MODBUS.
Por lo tanto cada módulo deberá programarse previamente con un valor diferente (de 01 a 99) en dicha cadena. En la Fig. 4 se puede ver la configuración con X-CTU de un módulo con ID = 18.

Funciones MODBUS implementadas.
A fin de liberar la memoria del microcontrolador para la tabla de direcciones se establecieron dos premisas para la implementación de las funciones MODBUS: que no impliquen consumo de memoria del microcontrolador y que su ejecución no requiriera el envío de más de un frame Zigbee. Es por esto que funciones como las de escritura múltiple no están soportadas (implicaban varios frames Zigbee) o la lectura de coils (operación no admitida por los módulos que implicaba almacenar su estado en memoria).

La Tabla 3 muestra el subset de funciones MODBUS que se definió. El mismo es suficiente para manejar todos los recursos disponibles en los módulos: DIO, ADC y PWM.

Función MODBUS	Recurso Xbee
02: Read input status	Entradas digitales
03: Read holding registers	ADCs
05: Force Single Coil	Salidas digitales
06: Preset single register	PWMs
17: Report Slave ID	Direcciones Zigbee y Node Identifier

Tabla 3: Funciones Modbus implementadas

Comandos MODBUS a Zigbee.
La Tabla 4 muestra los comandos AT enviados a los módulos para cada una de las funciones MODBUS implementadas.
Cuando el gateway recibe una función MODBUS soportada envía por Zigbee el comando necesario para dar cumplimiento al pedido. Por ejemplo, para la escritura de un coil se envía una trama que ejecuta el comando AT remoto para cambiar el estado del pin correspondiente. Ante la recepción de una función no implementada se genera una respuesta de error MODBUS.

Función MODBUS	Comando AT
02: Read input status	Remote IS (Force Sample)
03: Read holding registers	
05: Force Single Coil	Remote D0-D7 / P0-P2 = 04/05 (DIO configure)
06: Preset single register	Remote M0 / M1= XX (Set PWM)
17: Report Slave ID	Local ND (Node Discover)

Tabla 4: Funciones Modbus y comandos AT

Funciones MODBUS de lectura.
Para las funciones MODBUS de lectura (FC = 2 ó 3) se envía al módulo el comando remoto AT IS (Fore Sample) para solicitarle el estado de los pines configurados como entradas digitales y de los ADCs. La fig. XX muestra un ejemplo.

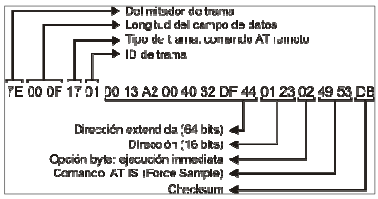

Fig XX.Trama dirigida al dispositivo con Extended Address (64 bits) = 00 13 A2 00 40 32 DF y Address (16 bits) = 01 23

Para Frame ID utilizaremos un índice (de 1 a 255) que luego deberá coincidir con el ID de la respuesta Zigbee. En caso de no ser así descartamos la respuesta y devolveremos un mensaje de error MODBUS.
Los datos de respuesta al comando remoto AT IS llegan en un frame con Frame Type = 0x92 (ZigBee IO Data Sample Rx Indicator) que contiene el estado de todos los pines (sólo son significativos los configurados como entrada) y las lecturas de todos los pines configurados como ADC (si los hay). Utilizando esta información el gateway elaborará la respuesta MODBUS según la función, dirección y rango.

Funciones MODBUS de escritura.
Para la función de escritura de coil (fc = 05) se envía un comando remoto AT que controla uno de los pines del módulo (por ejemplo AT D0 = 4 ó AT D7 = 5).
La escritura de registro (fc = 06) se utiliza para cambiar un PWM, lo que se consigue con los comandos AT remotos M0 y M1.

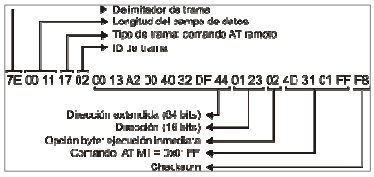

Fig. XY: Comando MODBUS que setea el PWM1 al 50% (AT M1 = 0x01FF) del dispositivo con Extended Address (64 bits) = 00 13 A2 00 40 32 DF y Address (16 bits) = 01 23

Mapeo de pines.
La Tabla 5 muestra la correspondencia entre coils MODBUS y pines del módulo Xbee. Como se ve el mapeo de coils a pines es fijo, independientemente de la cantidad de entradas o salidas definidas en el módulo, por ejemplo el coil direccionado como 4 en un mensaje MODBUS siempre corresponde al pin 11 (DIO4) aunque sea el único pin configurado como salida.

MODBUS COIL	Registro AT XBee	# Pin del Módulo
0	D0	20
1	D1	19
2	D2	18
3	D3	17
4	D4	11
5	D5	15
6	D6	16
7	D7	12
8	P0	6
9	P1	7
10	P2	4

Tabla 5: Mapeo de pines DIO

Por el contrario, para los ADCs este mapeo no es fijo. Cuando leemos su estado (con fc = 03: Read holding registers) el primer registro MODBUS hace referencia al primer pin del módulo configurado como ADC, que puede ser el 17, 18, 19 ó 20 según la programación.

Tabla de direcciones.
El firmware maneja el mapeo de las direcciones MODBUS (ID de Esclavo) en direcciones Zigbee. Como se mencionó anteriormente los mensajes Zigbee unicast necesitan de ambas direcciones (la de 64 bits y la de 16 bits) por lo que la tabla implementada contempla 10 bytes (64 + 16 bits) para las direcciones Zigbee de cada Esclavo.
La tabla se llena por primera vez cuando el Gateway se inicia y envía al Coordinador de la red un comando AT ND (Node Discover). Esto ocasiona que cada uno de los nodos presentes en la red conteste con un frame que contiene sus direcciones Zigbee y su Node Identifier (que es su MODBUS Slave ID).
Al recibir una función MODBUS se consulta la tabla con el Slave ID para conocer las direcciones Zigbee y poder armar la trama.
Adicionalmente se implementó la función MODBUS 17 que ejecuta un Node Discover durante el funcionamiento del Gateway, forzando la actualización de la tabla. El MODBUS Master deberá tener en cuenta que la ejecución de esta función puede demandar varios segundos por tratarse de mensajes broadcast.

Software MODBUS Master.
Al conectar el Gateway a un puerto USB de una PC el sistema operativo (tanto Windows como Linux) lo detecta como un nuevo puerto serie, por lo tanto una vez configurado lo único que debe hacerse para utilizar los módulos es seleccionar dicho puerto en el software MODBUS RTU Master. Se realizaron pruebas satisfactorias con LabVIEW, utilizando la biblioteca MODBUS provista por National, y en FreeSCADA con el plugin para MODBUS.

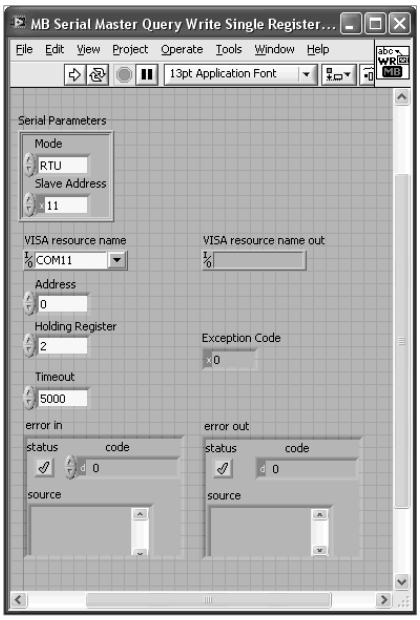

Fig. 6. Manejo de los PWMs de los módulos en LabVIEW 8.5 mediante la Función MODBUS 06: Preset single register.

CONCLUSIONES

El uso de las tramas API, nativas de los módulos Xbee, permitió obtener una red de elevadas prestaciones. La cuidadosa elección de las funciones MODBUS y su correcta implementación permitieron el uso de un microcontrolador ampliamente difundido y de bajo costo. La conectividad USB con la PC resulta ser sumamente práctica al adaptarse a prácticamente todas las plataformas actuales y futuras.

REFERENCIAS

Caprile S. (2009). Equisbí: desarrollo de aplicaciones con comunicación remota basadas en módulos ZigBee y 802.15.4, Gran Aldea Editores.

Digi. (2009). XBee®/XBee-PRO® ZB RF Modules, Digi International Inc.

Ember. (2009). EM250 - Single-Chip ZigBee / 802.15.4 Solution, Ember Corporation.

IEEE Computer Society. (2003). 802.15.4 IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements, Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), The Institute of Electrical and Electronics Engineers, Inc.

Modbus.org. (2006). MODBUS Application Protocol Specification V1.1b, Modbus-IDA.

Moya T., Goglino J., Hoyos D. (2010). Red de sensores y control inalámbrica para un sistema de generación de vapor solar térmico. Avances en Energías Renovables y Medio Ambiente 14, 8, 151-157

Rojvanit R. (2004). Migrating Applications to USB from RS-232 UART with Minimal Impact on PC Software: Application Note 956, Microchip Technology Inc.

ZigBee Standards Organization. (2008). ZigBee Specification, ZigBee Alliance Inc.

Mills D., Morrison G., (2000). Compact Linear Fresnel Reflector Solar Thermal Power Plants. Solar Energy Vol. 68, No.3, pp. 263 – 283.

ABSTRACT

This paper presents the implementation of Zigbee wireless modules controlled via MODBUS for instrumentation and control of solar systems. We used an 8-bit microcontroller to implement a gateway that decodes the received MODBUS functions from a PC and generates frames for Zigbee modules. The PC connection is done via USB, emulating a serial port to ensure maximum compatibility with existing control software.

Keywords: solar energy, instrumentation, control, microcontrollers.