ENTORNO DE DESARROLLO SOBRE FPGA PARA APLICACIONES DOMÓTICAS BASADAS EN TECNOLOGÍA X10

MANUEL D. CRUZ DÍAZ[1], JUAN A. ORTEGA RAMÍREZ[1], ÁNGEL BARRIGA BARROS[2] Y ALEJANDRO FERNÁNDEZ-MONTES GONZÁLEZ[1]

[1]*Departamento de Lenguajes y Sistemas Informáticos. Escuela Técnica Superior de Ingeniería Informática. Universidad de Sevilla. España.*

[2]*Departamento de Electrónica y Electromagnetismo. Escuela Técnica Superior de Ingeniería Informática. IMSE-CNM, CSIC - Universidad de Sevilla. España.*

mcrudia@us.es

Esta comunicación propone una arquitectura y un software básicos para la elaboración de un entorno de desarrollo que nos permita crear un controlador para la interfaz CM11A de la tecnología, orientada al mundo de la domótica, X10, sobre la tarjeta de desarrollo Spartan-3 de Xilinx. Con este controlador un usuario final será capaz de controlar y gestionar un conjunto de dispositivos repartidos por un espacio domotizado y que estén intercomunicados a través de una red de dicha tecnología. Se trata de un entorno de desarrollo, pues se han dispuesto los elementos necesarios (tanto hardware como software) que nos servirán como base para poder implementar futuras funcionalidades adicionales.

1. Introducción

Con objeto de encontrar soluciones tecnológicas económicas, poco intrusivas y de fácil manejo para el usuario final en el mundo de la domótica e inmótica, el uso de sistemas empotrados sobre FPGA *(Field Programmable Gate Array)* llegará a ser, al cumplir con todas estas premisas, una de las vías de desarrollo más importantes y a tener en cuenta en los próximos años en ese ámbito [1, 2, 3]. Los términos domótica e inmótica se definen como la aplicación de tecnologías de automatización y control en casas o edificios de viviendas, o en edificios de uso terciarios o industriales (oficinas, naves industriales, etc.), respectivamente. Estas tecnologías de automatización para "edificios inteligentes" se basan en dispositivos de control, sensores y actuadores comunicados entre sí empleando para ello diferentes esquemas (buses específicos, comunicación inalámbrica, red de alimentación eléctrica). Desde finales de la década de los noventa han visto la luz diferentes estándares de comunicación, algunos tan importantes como CEBus, LonWorks, KNX [4, 5].

El uso de la red de alimentación eléctrica facilita la instalación de estas tecnologías en los edificios ya que no se requiere de ninguna infraestructura de comunicación adicional. En este sentido se han desarrollado dispositivos y protocolos de comunicaciones adecuados a este medio [6, 7]. Uno de los más extendidos es el protocolo X10 [7, 8]. Éste fue desarrollado en 1978 por ingenieros de la empresa Pico Electronics Ltd. en Glenrothes, Escocia. Desde entonces las tecnologías basadas en la utilización de la red eléctrica conocida por *Power Line Carrier* (PLC) se han ido extendiendo, apareciendo una gran variedad de productos comerciales y estándares [9], aunque no de uso muy extendido en España.

En el ámbito de las aplicaciones domóticas enfocadas a la enseñanza, los sistemas empotrados sobre FPGA abren un ancho campo de posibilidades [10]. La presente comunicación sintetiza el proceso de construcción de un controlador capaz de comunicarse con todos los dispositivos conectados a una red X10 y distribuidos por el espacio domotizado. En particular el sistema desarrollado se centra en una aplicación

para el control de la iluminación dando pie a un incremento en el confort y el ahorro energético de los usuarios de la vivienda.

2. El protocolo X10

El protocolo de comunicaciones X10 permite controlar dispositivos de manera remota haciendo uso del tendido eléctrico y de los módulos receptores a los que están conectados. Las señales de control se basan en la transmisión de ráfagas de pulsos de 120 kHz. Las transmisiones se sincronizan con el paso por el cero de la corriente alterna. Un 1 binario se representa por un pulso de 120 KHz. durante 1 milisegundo mientras que un 0 binario se representa por la ausencia de ese pulso de 120 KHz.

El protocolo [11] consta de bits de direcciones y de comandos. En una red de comunicaciones X10 pueden coexistir hasta 256 módulos distintos interconectados. La forma de identificarlos es gracias a un código (*housecode*) compuesto por una letra (de la A a la P) y por un número (del 1 al 16). Estos módulos podrán llegar a aceptar (dependiendo de las características del mismo) hasta 16 códigos de operación distintos (*operationcode*). El protocolo X10 emplea estos dos tipos de códigos combinándolos para llevar a cabo las transmisiones entre la interfaz y el controlador.

Una transmisión de un código de operación desde la placa de desarrollo hasta el interfaz X10 consiste en dos pasos. En primer lugar la placa direcciona el dispositivo con el que quiere comunicarse y en segundo lugar envía el código de operación interpretable por dicho dispositivo. La secuencia de la transmisión puede ser observada con más detalle en la figura 1. En ella podemos ver qué significa cada uno de los bytes enviados. Este formato es siempre el mismo con independencia de si lo que se envía es la

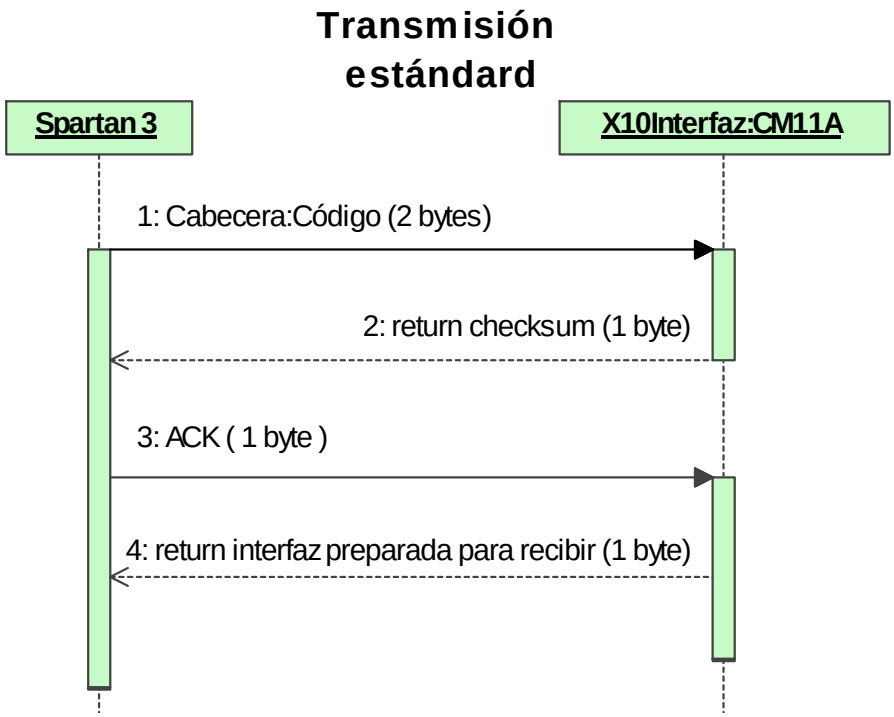

Figura 1. Transmisión estándar X10

dirección o el comando. En primer lugar la placa de desarrollo transmite la cabecera seguida de la dirección o el comando a la interfaz X10. A continuación la interfaz X10 recibe las dos tramas y crea una suma de comprobación, transmitiéndola a la Spartan 3. La placa comprueba la suma recibida y si es correcta responde con una trama de acuerdo (ACK). Finalmente la interfaz recibe la trama ACK y comunica su disposición a continuar con la comunicación. Podemos ver un ejemplo de código de esta transmisión en la figura 7, sección 4.2.

X10 posee una gama de productos bastante extensa, dividiéndose en las siguientes categorías: controladores, módulos y complementos. En el sistema que se presenta en este artículo se ha empleado un programador PC modelo *CM11A* (perteneciente a la categoría de los controladores) y un módulo lámpara (que recibe comandos desde la red eléctrica y actúa sobre el encendido de una lámpara).

3. Plataforma de desarrollo hardware

El sistema de control se ha implementado sobre una placa de desarrollo de FPGA, en concreto, una de la compañía Xilinx que contiene un *Spartan-3*. Trabaja con una frecuencia de reloj de 50 MHz impuesta por un oscilador incluido en la propia tarjeta. Dispone, además, de dos módulos de memoria SRAM de 256 KB. También incorpora un conjunto de elementos que facilitan el desarrollo de sistemas y permiten diversas aplicaciones, entre ellos se han empleado los siguientes: un puerto RS232 que permite la comunicación con la interfaz X10; cuatro botones pulsadores que configuran la dirección (*housecode*) y el comando (*operationcode*); un *display* 7-segmentos que permite verificar la operación e informa del estado del controlador o los dispositivos; un puerto de comunicación serie JTAG para la descarga y chequeo de la aplicación en la FPGA. En la figura 2 se pueden observar dichos componentes.

El entorno de desarrollo *Xilinx Platform Studio* (XPS) dispone de las herramientas que permiten cubrir el flujo de diseño de sistemas empotrados. Este flujo consta de las etapas de descripción de la arquitectura hardware, la síntesis e implementación del circuito controlador, el desarrollo de las aplicaciones software, la compilación de dichas aplicaciones y la programación del FPGA y de la memoria RAM.

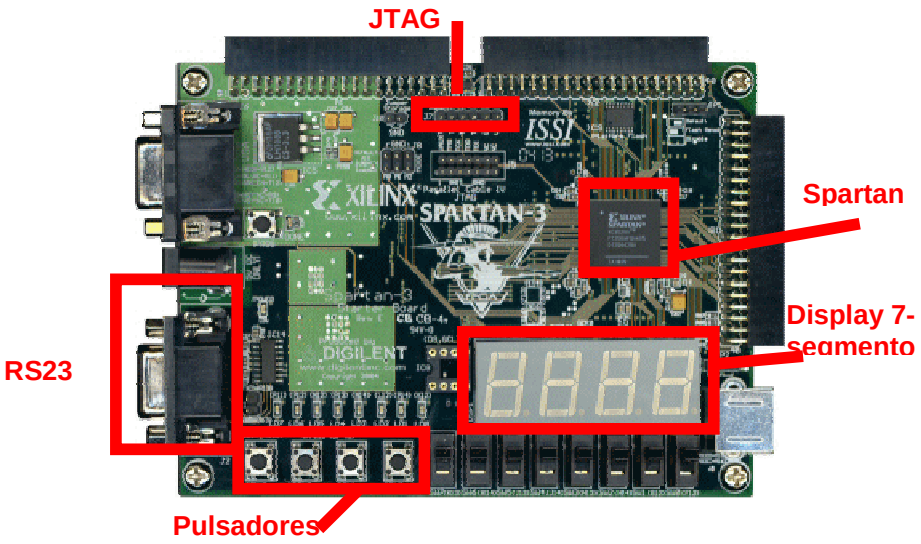

Figura 2. Tarjeta de desarrollo *Spartan-3* de Xilinx

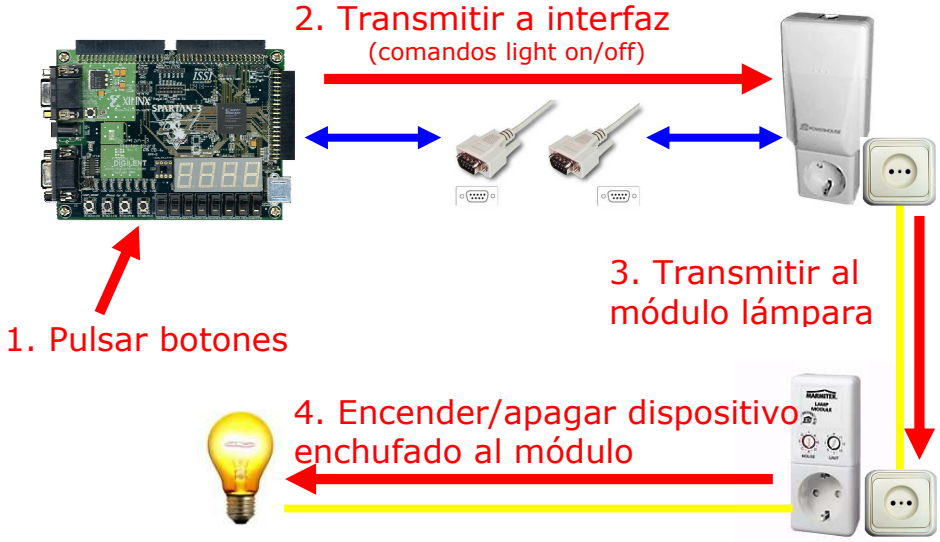

Figura 3. Diagrama de comunicaciones

La tarjeta FPGA controla cualquier módulo receptor de X10 instalado en el espacio domotizado a través de la interfaz CM11A de dicha tecnología (figura 3). Dicha interfaz se enchufa a la red eléctrica y se conecta con la placa por medio de un cable serie conectado a su puerto RS232. El modulo lámpara es usado para controlar la lámpara que esté enchufada a él. Es gestionado por el módulo CM11A, el cual irá indicando los cambios de estado por los que deberá ir pasando la lámpara en cuestión (encendido, apagado, intensidad).

4. Diseño del sistema

El diseño del sistema se ha basado en una metodología de codiseño hardware&software siguiendo el flujo de diseño que se establece en el entorno de desarrollo EDK de Xilinx. En concreto la plataforma hardware contiene los dispositivos de interfaz y de procesado que han sido sintetizados e implementados sobre el FPGA. La aplicación software se ha desarrollado en C++ e implementa el algoritmo de control y el protocolo X10. Dicha aplicación ha sido compilada dentro del entorno de EDK mediante un compilador cruzado que genera código ejecutable del procesador implementado en la plataforma hardware. El flujo de diseño finaliza cuando se han integrado el ejecutable, que contiene el software, con el hardware en un único fichero binario que se ha programado en el dispositivo Spartan-3.

4.1. Arquitectura hardware

En la figura 3 se ilustra el esquema de comunicaciones del sistema. Como puede comprobarse, el controlador se ha implementado sobre la placa de desarrollo de FPGA descrita en la sección anterior. Dicha placa se comunica con el módulo de interfaz X10 mediante el puerto RS232 que a su vez transmite la dirección y comandos a través de la red eléctrica. El módulo lámpara X10, una vez direccionado, recibe los comandos y actúa sobre el dispositivo que tiene conectado.

El diseño del controlador se ha realizado aplicando una metodología de codiseño *hardware&software* que ha permitido definir una arquitectura abierta, en el sentido que permite incorporar nuevos elementos, y flexible ya que permite ser reconfigurada. Para ello el circuito ha sido desarrollado con módulos IP (*Intellectual Property*), los cuales permiten la reutilización y la fácil inserción de nuevas

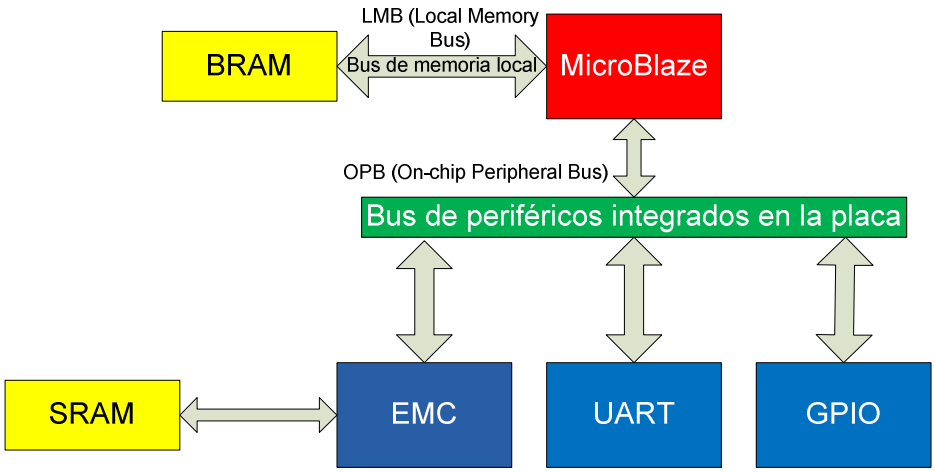

Figura 4. Diagrama del sistema

funcionalidades. La arquitectura, que se muestra en la figura 4, se basa en la utilización del procesador MicroBlaze de Xilinx. Dicho procesador es un *soft-core* con arquitectura RISC de 32 bits que puede ser integrado sobre los FPGA de Xilinx, como un módulo IP. MicroBlaze tiene arquitectura Harvard lo que significa que tiene buses de datos e instrucciones separados. De todas las estructuras de buses de las que dispone se han empleado el LMB (*Local Memory BUS*) y el OPB (*Peripheral On-chip Bus*) estándar de IBM. El bus LMB permite acceder a la memoria interna del FPGA en tan solo un ciclo de reloj. La función del bus OPB la veremos un poco más adelante.

El sistema lo constituyen dos tipos de elementos: los componentes hardware que conforman el circuito programado en la FPGA, y la aplicación software que es ejecutada en el hardware y es almacenada en memoria. Esta división de la funcionalidad del sistema en componentes hardware y software ha sido realizada teniendo en cuenta la ya mencionada arquitectura abierta y flexible.

El procesador se comunica con los dispositivos periféricos a través del bus OPB. Los periféricos son mapeados en el espacio de direcciones de la memoria. Esto significa que la inclusión de cualquier tipo de dispositivo consiste en conectarlo a dicho bus e incluirlo dentro del espacio de direcciones. En la aplicación que se presenta en esta comunicación solo se han necesitado los tres dispositivos periféricos mostrados en la figura 4. Estos se corresponden con un puerto paralelo (*gpio*), un puerto serie (*uart*) y un controlador de memoria (*emc*). El puerto paralelo recibe datos de los pulsadores de la placa y genera las señales a los *displays* 7-segmentos. El puerto serie permite realizar la transmisión de información hacia el módulo X10. El controlador de memoria externa facilita la transferencia de datos con la memoria SRAM de la placa que contiene el software de la aplicación.

En la figura 4 también puede observarse como en el sistema se dispone de dos bloques de memorias RAM: BRAM y SRAM. La BRAM es la memoria RAM interna de la FPGA mientras que la SRAM es el módulo de memoria externa a la FPGA. El empleo de las dos memorias se justifica debido a que el tamaño de la aplicación software requiere el uso de los dos recursos de memorias RAM. El módulo BRAM tiene mejores tiempos de acceso (un ciclo de reloj), sin embargo su tamaño está muy limitado. El bloque de SRAM puede ser ampliado basado en los requerimientos del sistema. Hay tres conectores de expansión en la placa de desarrollo para adaptar cualquier dispositivo adicional, como pueda ser por ejemplo un módulo de memoria.

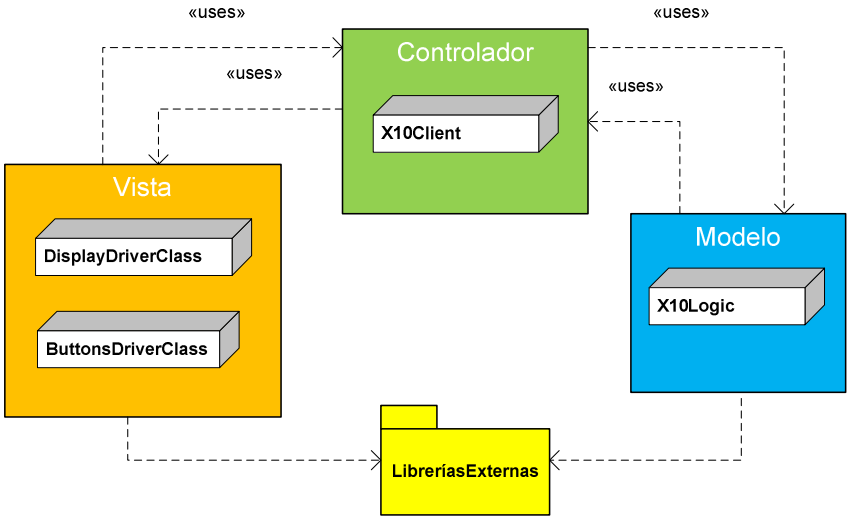

Figura 5. Modelo vista-controlador

4.2. Desarrollo de la aplicación

El lenguaje empleado para la implementación de la aplicación software ha sido C++. El entorno de desarrollo XPS proporciona una serie de paquetes de software desarrollados en ANSI C que facilitan la codificación de las aplicaciones de usuario (*drivers* de periféricos, librerías y sistemas operativos).

La figura 5 muestra el modelo vista-controlador que describe el diagrama de clases. El patrón ha sido usado para conseguir una completa separación entre la lógica y la interfaz de usuario. De esta forma si se quisiera mejorar y ampliar la aplicación debido a un cambio de la arquitectura se conseguiría de una manera más fácil y cómoda. Este modelo divide la aplicación en tres capas:

- Vista: es la capa que se encarga de transmitir y recibir información del usuario final. Si nosotros queremos ofrecer más servicios y para eso necesitamos ampliar nuestra arquitectura con otros dispositivos periféricos que se han añadido al bus OPB, simplemente se sugiere la inclusión de las clases que sean necesarias (normalmente una por dispositivo añadido) para implementar la interconexión entre el usuario final y los susodichos dispositivos. En caso a esta capa pertenecen dos clases:
    - *DisplayDriverClass*: es la clase que muestra los resultados e información de interés por el *display* 7-segmentos de la tarjeta de desarrollo.
    - *ButtonsDriverClass*: es la clase que recibe las pulsaciones de los botones de la placa y transmite la información al controlador para su interpretación y procesado. En esta clase se debe implementar algún tipo de espera, ya sea por interrupción o por espera activa, que nos posibilite detectar cuándo un botón es pulsado.
- Controlador: en esta capa se lleva a cabo el procesado de la información recibida y la transmisión de la información lista para ser usada por alguna de las otras dos capas. Por ejemplo, se recibe la información de que ha sido pulsado un botón, se procesa y se transmiten las órdenes deseadas a la

```
. . .
//next transmission, we send the address A1
outputBuffer = _HEADER_;//header
while (!XUartLite_Send  (&uart, &outputBuffer, 1));

outputBuffer = _DEVICE_ADDRESS;//device address
while (!XUartLite_Send  (&uart, &outputBuffer, 1));

while (!XUartLite_Recv  (&uart, &inputBuffer, 1));
//check the checksum
if (inputBuffer == ((_HEADER_ + _DEVICE_ADDRESS)&0xff))
{
        //checksum ok
        //transmit ACK to the board
        outputBuffer = _OK_;
        while (!XUartLite_Send (&uart, &outputBuffer, 1));

        while (!XUartLite_Recv  (&uart, &inputBuffer, 1));
        for (j=0; j<20000000; j++);//wait for synchronize

        if (inputBuffer == 0x55)
        {
                //interface ready for the next transmission
                . . .
```

Figura 6. Código ejemplo de una transmisión X10

capa del modelo, la cual se encargará de transmitir el comando al dispositivo indicado a través del puerto serie.

  - *X10Client*: esta clase funciona a modo de adaptador, restando responsabilidad a las clases de las otras dos capas y mejorando el modelado del sistema. Así se consigue una mayor independencia entre todas las clases facilitando modificaciones posteriores.

- Modelo: capa en la que se lleva a cabo toda la lógica de la aplicación. Es la que se encarga de establecer la comunicación con los dispositivos a través del puerto serie.

  - *X10Logic*: es la clase que aloja todo el conjunto de métodos necesarios para establecer la comunicación con la interfaz X10 a través del puerto serie, es decir, es, entre otras cosas, dónde podemos encontrar implementado el protocolo de comunicaciones de dicha tecnología. La figura 6 muestra el código de una interacción común entre la placa de desarrollo y la interfaz X10. En el caso en que quisiéramos ampliar el controlador para que fuese capaz de comunicarse con otras interfaces de la misma tecnología sólo tendríamos que modificar esta clase.

5. Conclusiones

Se ha descrito un sistema capaz de controlar los dispositivos de tecnología X10 repartidos por un espacio domotizado empleando para ello una tarjeta de desarrollo *Spartan-3* de Xilinx. Uno de los

objetivos alcanzados ha sido conseguir que el sistema sea ampliable y modificable, de manera que podamos en un futuro ir añadiendo nuevas funcionalidades y servicios al usuario final. La posibilidad de incorporar nuevos dispositivos externos (como puedan ser teclados numéricos, pantallas lcd's, pantallas táctiles, etc.) permite incrementar la funcionalidad y mejorar la interfaz de usuario. El sistema permite disponer de un entorno de desarrollo de aplicaciones enfocadas a buscar soluciones económicas y de fácil manejo en el mundo de la domótica y el hogar digital.

Referencias

[1] Renato Nunes, *Implementing Tiny Embedded Systems with Networking Capabilities*, IADIS International Conference on Applied Computing 2005, Algarve, Portugal, February 2005.

[2] German Dario Baron Chacon, Diego Alexander Tibaduiza Burgos. *Diseño y construcción de un sistema de seguridad para un recinto cerrado implementando FPGA.* Tesis de grado. 2006

[3] Darrell Wilburn, Dan Hafeman, Al Rogers, Helen Yu. *Using Spartan-3 FPGA's as low-cost controllers for remote digital cameras.* Xilinx, wp200 (v1.1). 2003.

[4] LonWorks (ANSI/EIA 709.1-A), *http://www.echelon.com*.

[5] KNX, *http://www.knx.org/*.

[6] HomePlug, *http://www.homeplug.org/home*.

[7] R. N. Bucceri, *The Latest Technology in Automated Home Control - Book System Design Manual Using X-10 & Hardwired Protocols*, Silent Servant, Inc, 2003.

[8] Home Systems, *http://www.homesystems.es/*

[9] Universal Powerline Bus, *http://www.pcslighting.com/upb/overview.html*.

[10] Felipe Mateos, Víctor M González , Reyes Poo, Marta García, Rosana Olaiz, *Design and Development of an Automatic Small-Scale House for Teaching Domotics*, 31st ASEE/IEEE Frontiers in Education Conference, Reno, NV-USA, (2001)

[11] X10, CM11A Interface Communication Protocol, *ftp://ftp.x10.com/pub/manuals/cm11a_protocol.txt*