UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

“DISEÑO E IMPLEMENTACION DE UN SISTEMA DE CONTROL PARA LA ILUMINACIÓN DE ESPECTACULOS BASADO EN EL PROTOCOLO DMX512”

TESIS

PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS EN ELECTRÓNICA Y COMPUTACIÓN

PRESENTA

Ing. Edgardo Yescas Mendoza

DIRECTOR DE TESIS

Dra. Irma Salinas Pérez

Huajuapan de León, Oaxaca. Septiembre de 2009

Tesis presentada el 11 de Septiembre de 2009

Ante los siguientes sinodales:

Dra. Virginia Berrón Lara

Dr. Mikhail Arkhipov

Dr. Jesús Linares Flores

Dr. Olexandr Mykolayouych Bondarchuk

Bajo la dirección de:

Dra. Irma Salinas Pérez

Dedicatoria

A mis padres, hermanos y a mi preciosa hija Tania Mónica.

A la memoria de mis abuelos, a quienes mucho les hubiera gustado ver llegar este momento.

A Dios, por darme la vida y la oportunidad de realizar este sueño.

Edgar.

Agradecimientos

A mis padres, quienes con tanto sacrificio no dudaron en apoyarme en mis estudios sin esperar otra recompensa más que la satisfacción de verme convertido en una persona de provecho. A Mayra por su amistad y ayuda. A mi hija, por iluminar mi camino y ser el porqué de las cosas. A todos mis tíos y primos, por quienes siempre recibí su apoyo y buenos consejos.

A mi asesora de tesis, Dra. Irma Salinas Pérez por haber depositado su confianza en mí para la elaboración de esta tesis y por su paciencia en los momentos de mayor dificultad que conllevaron a la realización de este trabajo. Al profesor M.C. Ricardo Ruiz Rodríguez y a la profesora M.C. María de Jesús Pérez Álvarez por sus valiosas asesorías.

A los sinodales: Dra. Virginia Berrón Lara, Dr. Mikhail Arkhipov, Dr. Jesús Linares Flores y al Dr. Olexandr Mykolayouych Bondarchuk por el tiempo dedicado a la revisión de esta tesis y por el interés mostrado para mejorarla.

A mis familiares y amigos que siempre me motivaron para continuar en este camino.

A todos los profesores de la División de Posgrado que participaron en mi formación académica, por haberme brindado sus conocimientos y amistad.

Al personal del Departamento de Mantenimiento de la Universidad por haberme apoyado todo el tiempo en la realización de este trabajo. En especial, al Ing. Marcelino Flores Alonso.

Edgar.

LISTA DE ACRÓNIMOS Y SIGLAS

Acrónimo y sigla	Significado
ASIC	Circuito de aplicación especifica
CA	Corriente alterna
CD	Corriente directa
CI	Circuito integrado
DAC	Convertidor analógico a digital
DMX	Multiplexación digital
ESTA	Asociación de Tecnología y Servicios de Entretenimiento
GUI	Interfaz de usuario gráfica
HW	Hardware
MAB	Mark After Break
Moc	Modelo de computación
MCU	Unidad Microcontrolador
PC	Computadora personal
PP	Puerto paralelo
SciDMX	Sistema de control de iluminación DMX
SW	Software
UML	Lenguaje unificado de modelado
USITT	Instituto de Teatro y Tecnología de Estados Unidos
UTP	Cable par trenzado sin apantallar
VB	Visual Basic

CONTENIDO

Lista de figuras

Lista de Tablas

Resumen

El presente trabajo, describe la aplicación de la metodología de diseño de sistemas empotrados con el diseño y construcción de un sistema de control para la iluminación de escenarios, basado en el protocolo DMX512. Este sistema es modelado formalmente como un sistema empotrado reactivo, el cual es jerarquizado en los siguientes subsistemas: una computadora, un controlador DMX y el sistema de iluminación formado por los receptores, dimmers y luces. El modelo de especificación formal utilizado en cada uno de los subsistemas se realiza con UML. Varios diagramas de UML como los casos de uso, máquinas de estado y diagramas de secuencia se utilizan para obtener un modelo funcional de todo el sistema. Para el desarrollo del programa de control de usuario, se utilizó Visual Basic; esta interfaz sirve como consola de control para la manipulación remota de la intensidad de las luminarias. El algoritmo del controlador desarrollado cumple con las especificaciones impuestas por el estándar DMX512.

En la implementación del controlador y los receptores se utilizaron los microcontroladores AT90S2313 de la firma Atmel, mientras que los dimmers se implementaron con triacs empleando la técnica de control por ángulo de fase. Éste diseño resultó ser simple y efectivo; los componentes utilizados en el diseño son económicos y de fácil adquisición. El sistema implementado se puede considerar como una herramienta práctica para el estudio de nuevas versiones del protocolo DMX o de protocolos emergentes de comunicación empleados en el control de iluminación.

Los resultados obtenidos en este trabajo, se pueden clasificar en dos categorías: los resultados del proceso de diseño y los experimentales. En los primeros, se considera que aplicando una metodología de diseño de sistemas empotrados, la comunicación entre el cliente y los diseñadores se facilita, a través del uso de los lenguajes de descripción como UML, ya que se evitan errores de comprensión que son difíciles de corregir en etapas posteriores de diseño, esto nos permite iniciar con la fase del diseño detallado del hardware y software casi inmediatamente para obtener un producto que cumple con los requerimientos del cliente. Los resultados de las pruebas experimentales muestran que la generación de la señal DMX cumple con el estándar de la USITT, y la funcionalidad de los dimmers es la esperada.

Se considera que las aportaciones de este trabajo son: la aplicación de la metodología de sistemas empotrados y el modelado funcional con diagramas de UML, en el diseño de un sistema de control de iluminación de escenarios.

Introducción

La iluminación en teatros, museos, galerías, conciertos, salas de baile y eventos corporativos (conferencias, reuniones, etc.) no es tan simple como encender luces sobre un escenario, sino por el contrario, es un proceso complejo que implica una correcta disposición de los dispositivos de iluminación, ángulos correctos de enfoque, iluminación posterior, frontal y lateral del escenario y un equilibrio de colores [Simpson, 2003]; todo esto con el propósito de lograr que el público pueda ver en todo momento a los actores y apreciar volúmenes en la escena. Para la iluminación de escenarios se emplean dos tipos básicos de dispositivos: focos, los cuales iluminan una amplia zona del escenario y proyectores, que iluminan intensamente áreas pequeñas. Los dispositivos de iluminación tienen cuatro propiedades controlables: la intensidad, el color, la distribución y el movimiento. Estas propiedades son llamadas *parámetros estáticos* ya que permanecen constantes sobre un periodo de tiempo y se utilizan para otorgar una determinada apariencia al contorno y al volumen de un intérprete u objeto determinado.

Los operadores de la iluminación controlan las luces de un escenario a través de dispositivos unidireccionales llamados *consolas de iluminación* que son interfaces entre el operador y luces proyectadas [Von, 2001]. En su versión más básica, una consola se reduce a controlar los dimmers para controlar la intensidad de las lámparas en los escenarios. Los *dimmers* son dispositivos capaces de cambiar gradualmente la intensidad de las lámparas, de tal manera que para ejecutar una determinada secuencia de luces y sombras, la consola transmite información basada en un protocolo de comunicación, el *dimmer* recibe esta información, la demultiplexa y ejecuta la función apropiada sobre el dispositivo de iluminación seleccionado. Otras consolas más sofisticadas permiten el control de más parámetros tales como el pandeo / tildeo, color, enfoque, forma de rayos, etc.

Uno de los últimos avances en luminotecnia es la consola automatizada basada en una computadora personal (PC), mediante la cual la intensidad de la luz en cada canal para cada entrada en escena (pie o *cue*) se archiva automáticamente en un banco de datos. De esta forma, el operador ya no necesita manipular manualmente cada uno de los dimmers; al pulsar un solo botón todos los focos cambiarán automáticamente según la intensidad programada y a la velocidad deseada [Simpson, 2003].

Sin embargo a pesar de todos estos adelantos tecnológicos la industria del entretenimiento sigue teniendo grandes dificultades para iluminar sus escenarios. Cuando un operador de una consola de iluminación crea la programación de su espectáculo, trata frecuentemente con detalles de muy bajo nivel que no tienen relación sobre la iluminación en sí misma como:

- Extender cables específicos para conectar una instalación de luz en la consola.
- Validar en la consola el tipo de instalación utilizada en el espectáculo, ya que cada fabricante tiene su propio mapeo de parámetros. Un mapeo de parámetros es la codificación de parámetros que realiza el usuario en cada evento.
- Especificar manualmente cuantas luces incluye la instalación conectada en la consola, y la forma en que ésta enviará un paquete de bytes para determinar uno de los parámetros a controlar de las luces.
- Asignar a cada dispositivo de iluminación una dirección para realizar el mapeo.

Cada dirección controla un parámetro del dispositivo de iluminación, algunos dispositivos usan una dirección, mientras que otros usan muchas para controlar sus múltiples parámetros [Randall, 2002]. Como por ejemplo, para controlar la intensidad de una instalación de iluminación sencilla, se utiliza una dirección para especificar el nivel de intensidad requerido; mientras que para un dispositivo de iluminación multi-funcional (como por ejemplo, un dispositivo con movimiento), se utilizan *20* direcciones para especificar los parámetros de: paneo, tildeo, intensidad, color, patrón y velocidad de rotación [Randall, 2002]. Esto significa que cuando un dispositivo de iluminación *incrementa su funcionalidad* también se *incrementa el problema de controlar* sus funciones.

Las instalaciones de iluminación espectacular son dispositivos extremadamente complejos, en consecuencia el diseño de un sistema de control de iluminación es un proceso también complejo, ya que en él convergen diferentes áreas de la electrónica y computación como: *programación orientada a objetos*, *redes de computadoras*, *microcontroladores (μC´s)* y *electrónica de potencia*. Todos estos elementos se combinan para obtener un sistema de control funcional y flexible que permita al operador combinar luces, colores y efectos especiales casi de la misma forma como él se los imagina [Von, 2001]. Esto realmente constituye un gran reto porque el software debe representar para el usuario una herramienta (interfaz) de control de todas o la gran mayoría de las funciones de los instrumentos de iluminación, es decir; el software debe incluir una serie de controles y arreglos que permitan al diseñador acceder de manera fácil algún parámetro del dispositivo de iluminación.

Para diseñar un sistema de control de iluminación es necesario desarrollar un hardware (el sistema eléctrico que soporta las operaciones deseadas por el diseñador) y un software formado por un conjunto de instrucciones dadas al hardware para ejecutar el diseño; el sistema eléctrico y electrónico debe responder de manera efectiva a éstas instrucciones; las cuales se transmiten a los dispositivos de iluminación a través de protocolos de comunicación. Las características de estos protocolos determinan el diseño estructural del hardware y la forma en la cual el software dirige a éste hardware.

Un buen protocolo de comunicación debe soportar todas o al menos la gran mayoría de las funciones de los dispositivos de iluminación [Von, 2001]. La industria de la iluminación ha creado una gran variedad de protocolos analógicos y digitales. Los protocolos analógicos son simples y se aplican al control de un solo parámetro: la intensidad de la luz; con una pequeña corriente se controla un voltaje de salida para disparar a los dimmers. Estos protocolos se pueden conectar a una instalación de iluminación de dos formas distintas:

Por medio de arreglos de mapeo completo: utilizando por separado una línea para cada circuito, es decir para colocar una instalación de iluminación con 48 dispositivos de luz, es necesario colocar 48 líneas conectadas desde la consola de iluminación hasta la instalación eléctrica.

Por medio de arreglos multiplexados: Periódicamente se transmiten diversas señales analógicas en una sucesión rápida, para formar paquetes de valores de intensidad. Esto requiere únicamente una sola línea. La instalación eléctrica debe demultiplexar la señal en los dimmers.

Un arreglo multiplexado es muy recomendable cuando el numero de dispositivos de iluminación excede de cierto número. Sin embargo, se incrementan las demandas en la calidad de la línea utilizada. Por otra parte, la electrónica involucrada es considerablemente más compleja y necesita de una memoria pequeña en la instalación eléctrica para conservar la intensidad de los dimmers entre los paquetes [Von, 2001].

Los protocolos digitales funcionan de manera similar a los protocolos analógicos, con la diferencia que los paquetes transmitidos representan valores numéricos digitales en lugar de valores que indican niveles de voltajes. Los paquetes transmitidos no tienen una estructura intrínseca (interna) y solamente representan secuencias de números. Su funcionamiento es el siguiente: Un transmisor envía paquetes con valores numéricos a múltiples receptores conectados a una línea, cada receptor demultiplexa el paquete y toma solamente el byte correspondiente a la dirección asignada. El sistema de control retransmite el paquete periódicamente para actualizar cualquier cambio del operador.

Estos protocolos se limitan a controlar la intensidad de los dispositivos de iluminación. Debido a su naturaleza digital, resulta razonable pensar que sería relativamente fácil controlar también otros parámetros de los dispositivos; pero esto no es así, ya que los protocolos en sí mismos no contienen la estructura necesaria para satisfacer estos requerimientos [Von, 2001]. Algunos de los protocolos digitales existentes en la industria de la iluminación son: CMX, K96 y el protocolo AVAB, sin embargo estos son incompatibles entre ellos, ya que la señal que manejan y los conectores que utilizan son diferentes, requiriendo de cajas convertidoras e inversores de voltaje para hacerlos compatibles, lo que constituye un gran problema, por este motivo éstos protocolos han sido reemplazados por el protocolo de comunicación *DMX512.* Tomando en consideración lo anteriormente mencionado, en el presente trabajo se decidió que el protocolo de comunicación implementado es DMX512, ya que no es nuestro objetivo realizar un análisis comparativo de los protocolos de comunicación.

El protocolo DMX512 se desarrolló en 1986, por el Instituto de Teatro y Tecnología de Estados Unidos (USITT) [URL 2], pero no fue sino hasta 2004 cuando se aprobó por el Instituto Nacional Estadounidense de Estándares (ANSI) con el nombre "E1.11, USITT DMX512–A" [Huntington, 2000]. USITT [URL 3] transfirió el derecho de mantenimiento del protocolo a la Asociación de Tecnología y servicios de Entretenimiento (ESTA) [URL 4]. El protocolo DMX512 usado inicialmente para controlar dimmers a través de una consola, reveló ser útil en la operación de equipo de iluminación “inteligente” como: intercambiadores de color, estrobos, máquinas del humo, láser, fuentes de agua, operaciones de control de palco (control de cortinas, movimiento del palco), entre otros.

Los últimos desarrollos en esta área se dirigen hacia el diseño de dispositivos *empotrados en red* [Rubinstein et al, 2003]. Un dispositivo empotrado en red es una colección de dispositivos (llamados *esclavos* ya que ellos son parte de una arquitectura maestro-esclavo), conectados a un bus común. Cada dispositivo tiene su dirección física propia en el espacio de direcciones. La arquitectura maestro-esclavo implica que el bus del maestro controla e inicializa todas las comunicaciones entre los dispositivos esclavos usando un conjunto de reglas (protocolo de comunicación).

Investigaciones recientes demuestran el interés por resolver los problemas anteriormente expuestos en la industria del espectáculo. Como por ejemplo, en el trabajo presentado en [Von, 2001] se desarrolla un software utilizando los métodos de *modelación conceptual y estructural* (ingeniería de software), para controlar dispositivos de iluminación. La consola de iluminación se implementa con una computadora (PC) y puede operar en un amplio rango de interfaces conectadas a los equipos de iluminación de escenarios modernos. Soporta tanto protocolos digitales (DMX512) como analógicos. Sin embargo, como no diseñan la consola de control, resulta problemático aplicar este software en consolas comerciales que incluyen controles continuos especializados. Para resolver este problema, recomiendan al usuario del paquete trabajar con consolas que usan pantallas de *touch-screen*. No es difícil de comprender la limitación de este trabajo ya que la iluminación de los escenarios tiene muchas fases dentro de las cuales el software es únicamente una parte pequeña.

En [Dasiewicz, 2001], se diseña e implementa un dimmer multicanal (cuatro canales) basado en el protocolo de comunicación DMX512, el controlador (dispositivo de transmisión que genera la señal DMX512) se realiza con una computadora personal (PC). Los dimmers están formados por un microcontrolador AT90S8515 y triacs, éstos últimos se activan utilizando el disparo por ángulo de fase controlando así la cantidad de potencia suministrada a las lámparas. Sin embargo, en [Dasiewicz, 2001] no se logra obtener la tasa de transmisión requerida por el protocolo DMX512 (250 kbps); logrando solamente transmitir datos confiables a una tasa de 19200 bps. Además no utiliza fusibles e inductores para la supresión de ruido en las lámparas.

En [Kopel et al, 2000], presentan el desarrollo de una consola DMX512 conectada a la computadora por el puerto USB. La consola permite controlar la iluminación en tiempo real, así como de manera pre-programada, por medio de un bloque de entradas, éste último tiene potenciómetros (faders) y botones para el control directo de la iluminación. La PC despliega el estado de la consola.

En [Ferreira, 2000], se presenta solamente una propuesta de diseño de un dimmer sin lograr su implementación. Un microcontrolador demultiplexa la señal DMX512 generada por un controlador externo; un DAC (Convertidor digital a analógico) convierte este dato en una señal analógica, y por medio de un CI (circuito integrado) para el control de fase (TCA785) se controla el disparo de los tiristores[1]. Este diseño parece ser muy práctico debido a que todo el trabajo de sincronía y disparo del triac lo realiza el CI TCA785 en función de un voltaje de control, proporcionado por el DAC. Aunque éste resulta ser complejo por la gran cantidad de elementos que se requieren para su implementación.

Justificación

En la industria del entretenimiento existe una gran demanda en el desarrollo de sistemas de control para iluminación de escenarios, ya que las herramientas existentes son limitadas e inapropiadas para interactuar con el usuario [Costa et al, 2001]. En México esta problemática se acentúa aún más; la industria del espectáculo no tiene desarrollos tecnológicos propios y depende completamente del exterior para satisfacer sus demandas; en consecuencia los equipos de iluminación son extremadamente caros. Lo que representa una gran desventaja para las empresas del entretenimiento nacional sobre todo para las pequeñas, quienes organizan sus eventos de manera manual y exponiéndose a situaciones de riesgo ya que:

- Los operarios realizan muchas conexiones del equipo de iluminación con cables grandes, pesados y peligrosos (se manejan corrientes de aproximadamente 60 Amperios).
- Las plantas de energía eléctrica están cerca del escenario generando ruido acústico no grato para los espectadores.
- Las conexiones que hacen los operadores toman demasiado tiempo.

La problemática a la cual nos enfrentamos es la siguiente: por una parte la industria del entretenimiento nacional (empresas pequeñas), necesitan sistemas de control que les permitan definir, predecir, probar y validar sus diseños de iluminación [Costa et al, 2001] y por otra, nos encontramos con una pobre contribución de nuestras Universidades en la solución de problemas de este tipo. Los programas

[1] Un tiristor es un dispositivo semiconductor de estado sólido, que actúa como interruptor de señales eléctricas.

educativos no estudian los protocolos de comunicación relacionados con esta industria ya que no cuentan con un equipo especializado para realizar las prácticas, y en otros se ha optado por utilizar sistemas industriales comerciales, con el inconveniente de que los alumnos no pueden visualizar y comprender lo que ocurre en las diferentes partes del sistema.

Atendiendo a estas demandas se consideró la posibilidad de diseñar e implementar un prototipo de un sistema de control de luces utilizando el protocolo de comunicación DMX512, con dos propósitos:

- Automatizar la iluminación de escenarios pequeños[2].
- Lograr una aplicación de uso práctico, así como brindar a los estudiantes una plataforma que les permita adquirir un conocimiento teórico y práctico del protocolo, con el fin de proponer mejoras, generar nuevas propuestas de protocolos y diseños, o trabajar integrando tecnologías como en [Jackman, 2005].

Descripción y formulación del problema

Para hacer posible nuestro proyecto se consideró desarrollar un prototipo económico con doce canales independientes; para doce lámparas de halógeno con capacidad máxima de 1.2 KW a 127 V, con un consumo promedio de 9.44 Amperios. El proyecto consistirá en diseñar un hardware y un software los cuales se deben diseñar en paralelo utilizando la metodología de *sistemas empotrados en red (maestro esclavo)* [Rubinstein et al, 2003]. El maestro será el controlador o transmisor DMX512 y los esclavos los 12 receptores de la señal conectados en una cadena tipo Daisy[3]. En la Figura 1, se muestra un diagrama a bloques de la estructura general del sistema de control para la iluminación de escenarios. El sistema estará formado por los siguientes subsistemas:

- Una consola de control, implementada en un programa de computadora (PC).
- Controlador DMX.
- Receptores DMX512.
- Dimmers capaces de manejar lámparas de hasta 1.2 KW / 9.44 A.

Estos requisitos se establecieron por el cliente (grupo musical: Sucesores del Norte), y por algunas características de dimmers comerciales. Los requisitos se presentan con más detalle en el sección 2.1.1 del capítulo 2.

- [2] Un escenario pequeño tiene las siguientes dimensiones: de frente menos de 10 metros, de fondo entre 4 y 6 metros, y de altura: 7 metros.

[3] Una cadena tipo Daisy es una conexión de varias unidades, en la cual los cables van desde la unidad 1 hasta la unidad 2, y de ésta a la unidad 3, y así sucesivamente hasta llegar a la última.

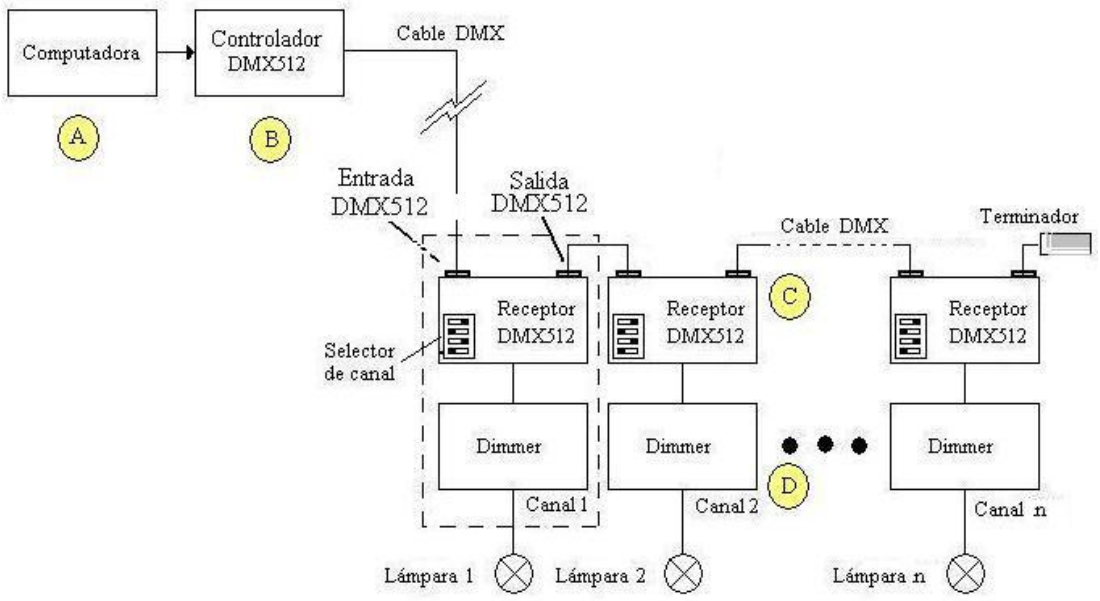

Figura 1. Estructura básica del sistema de control para la iluminación de escenarios.

El diseño y construcción del subsistema controlador y los receptores se realizará con los μC´s AT90S2313[4], mientras que para los dimmers, se propone el uso de tiristores utilizando la técnica de control por ángulo de fase. El cable DMX es un cable de par trenzado UTP, con una longitud mayor a 50 metros.

El diseño del software se desea que sea flexible y confiable para permitir al usuario:

- Controlar la intensidad de las lámparas por medio de un botón deslizante.
- Ejecutar la función *fade in (out)*, es decir una vez que es presionado el control correspondiente y elegir un intervalo de tiempo, automáticamente incrementar (decrementar) progresivamente la intensidad de las lámparas en la duración dada.
- Programar y reproducir secuencias de prendido y apagado de luces.

El programa a desarrollar será implementado en Visual Basic[5] (VB) v6.

Usuarios Potenciales

[4] Se seleccionaron estos μC´s porque cuentan con una memoria flash integrada de 2 KB necesaria para nuestra aplicación, el set de instrucciones es amplio y éste ejecuta la mayoría de instrucciones en un ciclo de reloj.

[5] Se utilizará VB por ser una herramienta de desarrollo de aplicaciones diseñada específicamente para la familia de sistemas operativos Windows. VB proporciona un ambiente flexible, permitiendo el desarrollo rápido de aplicaciones.

- Grupos musicales regionales, cines, discotecas, museos y auditorios de escuelas (SUNEO).
- Estudiantes y profesores en las áreas de Electrónica y Computación que deseen proponer cambios en el programa de control y mejoras al protocolo DMX512.

Objetivo general

Diseñar e implementar un prototipo de un sistema de control de iluminación programable con doce canales conectados en una cadena tipo Daisy, para regular la intensidad de las lámparas y otras funciones aplicando la metodología de sistemas empotrados maestro-esclavo y el protocolo de comunicación DMX512.

Objetivos específicos

- Diseñar e implementar los subsistemas: controlador y receptores DMX512 basado en μC´s.
- Realizar un programa de control en la PC con interfaz gráfica, que permita al usuario programar secuencias de luces y otras funciones como en una consola de control clásica[6].
- Diseñar y construir un dimmer basado en μC y una etapa de potencia con tiristores disparados por la técnica de control por ángulo de fase.

Hipótesis

Aplicando la metodología de maestro-esclavo y el protocolo de comunicación DMX512, es posible diseñar un sistema de control de iluminación para escenarios, que sea económico y flexible.

Limitaciones y delimitaciones

Limitaciones

- No existe una linealidad completa entre la alimentación de las lámparas y la brillantez que ellas entregan.
- EL ruido de radiofrecuencias (RF) generado por la conexión repentina de las lámparas y los armónicos generados por el dimmer no serán considerados en este trabajo.

Delimitaciones

- Debido a los pocos recursos con los que contamos, nuestro sistema de control de iluminación se enfocará principalmente al control de los dimmers y al encendido-apagado de lámparas, omitiendo el control de posicionamiento de luces[7]. Sin embargo, nuestro sistema de control es capaz de soportar estas funciones solamente adicionándolas en el programa principal, la arquitectura del controlador y del receptor no requerirán grandes cambios.

[6] Dispositivo con controles lineales (potenciómetros) para regular la intensidad de las lámparas.

[7] Para implementar ésta función se necesitan dispositivos de luces especializados tipo *Studio 250 High End* o similares, los cuales son caros y no contamos con los recursos suficientes para adquirirlos.

- El sistema de control de iluminación a desarrollar deberá utilizar componentes de fácil adquisición. También se buscará que tenga las funcionalidades básicas. Con esto esperamos que su costo esté al alcance de grupos musicales que no pueden darse el lujo de invertir en la adquisición de equipos sofisticados.

Metodología

El trabajo cubrirá las etapas de diseño, implementación y pruebas. Para la parte del diseño se necesita hacer un estudio sistemático de los siguientes puntos:

- Protocolo DMX512 con base a la arquitectura establecida en [E1.11-2004, 1990].
- Consolas de control DMX512 para PC.
- Dimmers digitales.
- La información que se ha considerado consultar es:
- Páginas de USITT y la ESTA, para la adquisición del estándar DMX512.
- Libros especializados en las áreas de:
- Protocolo DMX512 [E1.11-2004, 1990], [Mobsby, 2005].
- Industria de la iluminación [Huntington, 2000], [Simpson, 2003], [Scott, 1996].
- Electrónica de potencia [Couedic, 2000].
- Sistemas empotrados [Berger, 2002], [Marwedel, 2006].
- Redes de computadoras [Stalling, 1997].
- Programación orientada a objetos [Ramírez, 2001], [Russo et al, 1999].
- Microcontroladores [Gadre, 2001].
- Arquitectura del puerto paralelo [Axelson, 1996], [O´Sullivan et al, 2004].
- Recopilación de manuales de equipo DMX512 de casas comerciales y artículos científicos relacionados.

En el diseño del sistema de control de iluminación se aplicará la metodología de sistemas empotrados, la cual consta de siete fases: Especificación, Particionamiento, Iteración e Implementación, Diseño detallado del hardware y software, Integración, Pruebas y Mantenimiento. Cada una de las cuales se desarrollará por separado para controlador, el receptor y el dimer digital.

La modelación del funcionamiento de cada uno de los subsistemas se realizará con el Lenguaje Unificado de Modelado (UML). Los diagramas de UML serán utilizados para el análisis y diseño del hardware como se realiza en un sistema software; ya que nos permiten mostrar los diferentes protocolos y aspectos de modelo de servicios, tales como aspectos de comportamiento, estructural y de arquitectura de nuestro sistema.

Para la simulación del sistema, la parte de programación del MCU se realizará con el entorno de desarrollo integrado Avr Studio de la firma Atmel. Mientras que para el programa de control se empleará Visual Basic.

Estructura de la tesis

El presente trabajo está estructurado en seis capítulos, su distribución es la siguiente:

El capítulo 1 presenta el Estado del Arte en base a las tendencias actuales de los sistemas de control de iluminación, incluyendo los tipos de consolas, dimmers y protocolos de comunicación para la

iluminación. Además presenta las bases de la metodología de desarrollo de sistemas empotrados empleada y finaliza con los fundamentos del Lenguaje de Modelado Unificado.

El capítulo 2 inicia con la primera fase del desarrollo del Sistema de Control de Iluminación DMX (SciDMX), posteriormente se describe como se divide el diseño del SciDMX en subsistemas. El capítulo finaliza con el desarrollo del controlador DMX, presentando el diagrama a bloques, esquemático, diagramas de estado y diagramas de actividad del programa de controlador, las características más importantes del hardware utilizado, los circuitos empleados, y las consideraciones que fueron tomadas en el diseño de éste subsistema.

El capítulo 3 describe el diseño y desarrollo del programa de control de la computadora, se presentan las consideraciones tomadas en el programa orientado a eventos realizado en Visual Basic, así como su modelado en UML.

El capítulo 4 describe las fases de desarrollo del receptor DMX y del dimmer. Presentando los diagramas y las consideraciones que fueron tomadas en el diseño de ambos módulos integrados en un mismo microcontrolador.

El capítulo 5 se describe la manera en que como se realizaron las pruebas al SciDMX y los resultados obtenidos mediante gráficas de voltaje y corriente en las luces. Finalmente, se presenta el acabado que tienen las tarjetas electrónicas del SciDMX.

Por último, en el capítulo 6 se presenta las conclusiones y las líneas futuras de este trabajo.

Publicación generada

A continuación se muestran datos de la publicación que se generó durante el desarrollo del presente trabajo:

Autores: I. Salinas, E. Yescas

Titulo: Diseño e implementación de un sistema de control de iluminación DMX512

Congreso: ICED 2006

Publicación: Proceedings of the 2006 International Conference on Electronic Design

ISBN 968-9085-01-8

Lugar: Veracruz, México.

Año: 21-23 de Noviembre, 2006

Capítulo 1. Estado del Arte

La iluminación en los escenarios de teatros, auditorios, museos, discotecas y televisión; es el elemento primordial que determina en gran medida el éxito o fracaso de una producción; ya que usando los efectos de la luz se afectan los sentidos de la audiencia y de esta manera se evocan su emociones. La luz es la materia prima para crear efectos visuales que condicionan psicológicamente al espectador, de esta manera se puede pensar en la luz no como un elemento extra que se incluye sólo en algunas situaciones, sino como una elemento fundamental de todo tipo de producción visual. La iluminación en un escenario junto con otros elementos de la producción permiten al actor comunicarse con la audiencia, de esta manera se podría asegurar que la iluminación esta estrechamente relacionada con el sonido, y en [Shin et al, 1998] se afirma que: Los actores que son difíciles de observar, también son difíciles de escuchar. Por lo tanto, la principal función de la luz en un escenario es iluminar a los actores, de tal manera que ellos sean completamente visibles para la audiencia en todas las partes del auditorio. Si un actor no es completamente visible en un momento particular, esto se debe a una decisión deliberada del operador para dramatizar la escena.

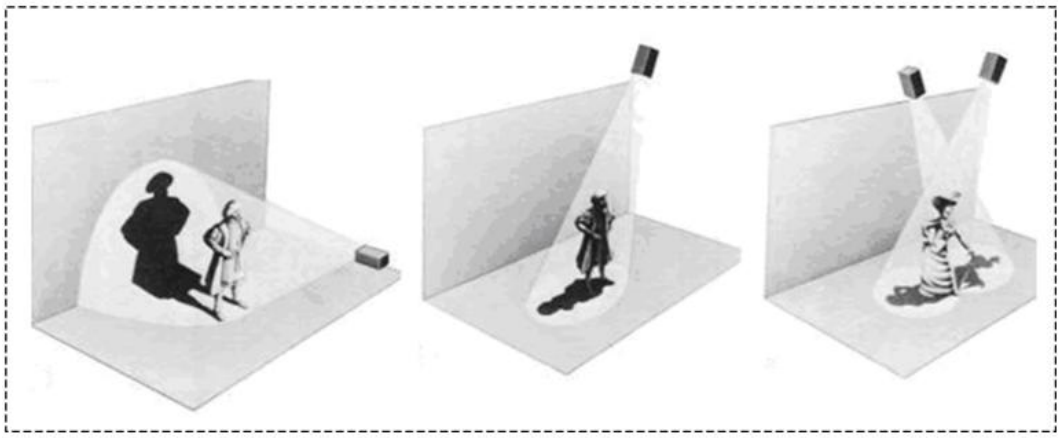

Figura 1.1. Iluminación de un escenario.

La cantidad de luz requerida en un escenario varía con respecto a la intensidad que refleja lo que se desea mostrar, Figura 1.1. Por ejemplo, en una escena hablada, los ojos así como el resto de las expresiones faciales del actor son muy importantes al momento de proyectar la de luz [Reid, 1998]. Mientras que en una representación de baile o danza, la iluminación de todo el cuerpo es de principal importancia. En la Figura 1.2, se presentan dos proyecciones distintas de iluminación en el escenario.

Por otra parte, es inevitable considerar que la luz impone algún grado de selectividad en la visión de la audiencia. Así, su influencia puede ser mínima o puede jugar un papel importante, en la medida que la imaginación visual de la audiencia sea usada para asistir a los actores para comunicar el trabajo de los escritores y compositores [Reid, 1998]. Con el prendido y apagado de luces o con movimientos continuos de rayos de luz, la concentración de la audiencia es dirigida hacia diferentes áreas del escenario y de un actor a otro. Estos movimientos deben ser tan sutiles que no lastimen la visibilidad de la audiencia, y al

mismo tiempo, deben ser tan fuertes que dirijan los ojos de los espectadores hacia las áreas deseadas. Es decir, las luces actúan como controles de movimiento.

Figura 1.2. Diferentes proyecciones de la luz en un escenario: a) la proyección de la luz se centra en los rasgos faciales de los actores. b) la proyección de la luz se centra en todo el cuerpo.

En la actualidad existe una gran variedad de instrumentos de iluminación utilizados en la industria del espectáculo llamados *luminarias*, las cuales tienen cuatro *parámetros estáticos* controlables que son: *el color, la intensidad, la distribución y el movimiento,* estos parámetros permiten al diseñador controlar las emisiones de las fuentes de luz para crear el ambiente visual, emocional y temático de la escena.

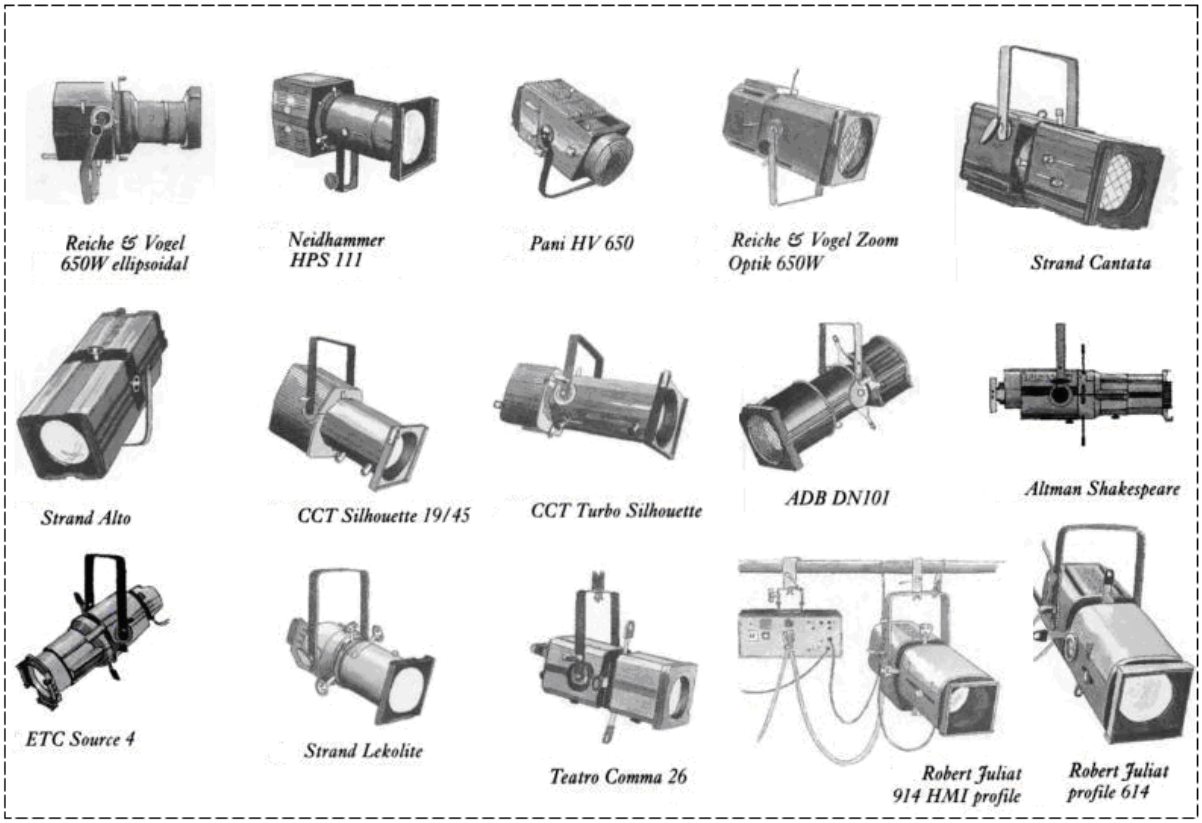

Figura 1.3. Instrumentos de iluminación

También existen instrumentos de iluminación sofisticados llamados *luces inteligentes,* que incluyen además de los parámetros estáticos algunos otros *atributos* tales como: *la selección* y *rotación de gobos*[8], *paneo (movimiento vertical)* y *tíldeo (movimiento horizontal)*, *focos*, *cambios de color*, *rayos, y otros efectos especiales*. Estas funciones dependen del modelo de luces y del fabricante; en la Figura 1.3, se muestran solo algunos de los diferentes modelos de instrumentos de iluminación que existen en el mercado, la gran mayoría de ellos se construyen sobre la base de las lámparas de tungsteno, por tener buenas características para reflejar la luz y para regular su intensidad [Reid, 1998].

La cantidad de luminarias necesarias en un escenario varía en función directa con el tamaño del mismo y con el tipo de atmósfera y textura que se desea recrear. Cada una de estas luces tiene una función especial en la escena y por lo tanto, debe prenderse con la intensidad adecuada y en el momento preciso, de no ser así, se deteriora la participación de los actores, e inclusive se podría generar confusión entre los mismos quienes estarían expuestos a juegos aleatorios de luces y sombras, sin ningún significado temático.

Para controlar la visibilidad en todas las partes del escenario, se colocan luces individuales en puntos estratégicos del mismo, como se muestra en la Figura 1.4. Anteriormente había un operador para cada posición, pero se necesitaban muchos operarios y constantemente se producían errores humanos. Cada lámpara o grupo de lámparas se controlaba por separado ya que se necesitaba balancear la iluminación para dar la vista global, requerida en la escena [Kopel et al, 2000].

En la actualidad, las luminarias se regulan electrónicamente por medio de sistemas modernos de control constituidos por dos partes principales: Una "consola" y dimmers. Los dimmers son válvulas de estado sólido que permiten el control la intensidad de las luces. La consola dispone de uno o más renglones de botones deslizantes que se programan en dos puntos diferentes: "establecer" y "desvanecer". El operador mueve manualmente el deslizador hacia estas dos posiciones dependiendo del efecto de luz que desee realizar. En la Figura 1.5 se muestra el diagrama a bloques de un sistema de control moderno de luces [Huntington, 2000].

[8] Los gobos son piezas metálicas de diferentes formas usadas para crear una imagen que será proyectada por la luz.

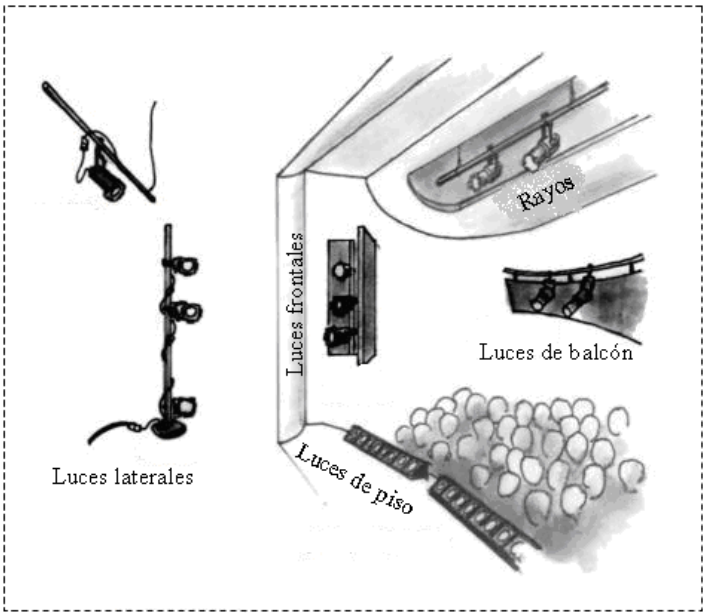

Figura 1.4. Posiciones de las luces en los escenarios.

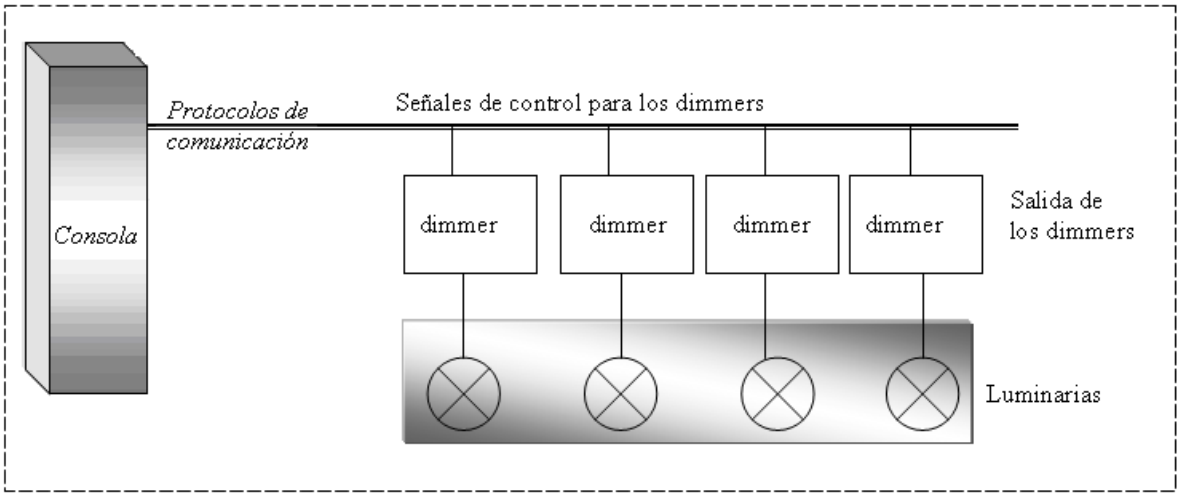

Figura 1.5. Diagrama a bloques de un sistema de control de iluminación.

A cada dispositivo de iluminación se le asigna una dirección para realizar el mapeo desde la consola. Cada dirección controla un parámetro de la luminaria, pero los dispositivos con funciones múltiples como las luces inteligentes requieren hasta 20 o más direcciones para realizar este mapeo de parámetros y atributos. Todas las consolas se comunican con los dimmers y con otros dispositivos de iluminación a través de protocolo de comunicación. Estos protocolos o códigos de información se envían a todos los dimmers, de los cuales sólo uno se activa y ejecuta la acción correspondiente. Un buen protocolo de comunicación debe ser capaz de soportar la gran mayoría de las funciones de los dispositivos de iluminación. Los dimmers, se colocan en una localidad apartada de las luces por consideraciones ambientales tales como el ruido o la temperatura.

Los modernos dispositivos de iluminación requieren de sistemas de control computarizado para regular a los parámetros estáticos y a los atributos. Muchos proveedores de consolas tienen software o editores “offline” para sus consolas que sirven para programar, almacenar y simular secuencias de luces para eventos. Sin embargo, los costos, de estas consolas se elevan y requieren de un operador o técnico profesional para su operación lo cual no es redituable para muchos grupos musicales sobre todo de nuestro país. La gran mayoría de estos grupos manipulan las luces casi de forma manual, poniendo en riesgo la vida de los operarios, por las altas corrientes que se manejan, además, para cada presentación los operadores deben conectar una gran cantidad de carretes de cables gruesos, que en muchas ocasiones sufren torceduras o rotura de pines, y distorsionan la señal produciendo prendidos aleatorios de las lámparas.

Sin embargo, a pesar de los grandes progresos que se han alcanzado, en materia de iluminación para espectáculos, la modelación conceptual de los recursos y habilidades de las consolas modernas aún no están bien desarrollados. En su lugar, estas consolas han sido consideradas como una colección variable de dispositivos de iluminación cuyos parámetros deben ser controlados. Cuando un operador crea su programación para una presentación, frecuentemente necesita tratar con detalles engorrosos, como por ejemplo: extender grandes cantidades de cables desde la consola hasta las luminarias, asignar canales a las lámparas multi-funcionales, y determinar como todas y cada una de ellas se mapeará desde la consola. Por lo tanto, se podría afirmar que las modernas consolas computarizadas operan al mismo nivel que las computadoras de los años 70’s [Von, 2001].

Un sistema de control de iluminación requiere de métodos holísticos que incluyan los conceptos de diseño de hardware, software y teoría de control de sistemas, comunicaciones y electrónica de una manera consistente, como se muestra en la Figura 1.6; y no de forma parcial como ocurre en los trabajos presentados en [Von, 2001], [Dasiewicz, 2001], [Kopel et al, 2000], [Ferreira, 2000]. En la Figura 1.6 se muestran las diferentes áreas involucradas en el diseño de un sistema de control de iluminación. En [Sperbe, 2001] se afirma que un sistema de control de iluminación es un sistema empotrado cuya función principal es controlar la brillantez de las lámparas eléctricas utilizadas en un escenario. Un sistema empotrado es un sistema que ejecuta tareas predefinidas, con requerimientos y restricciones específicos [Henzinger et al, 2000].

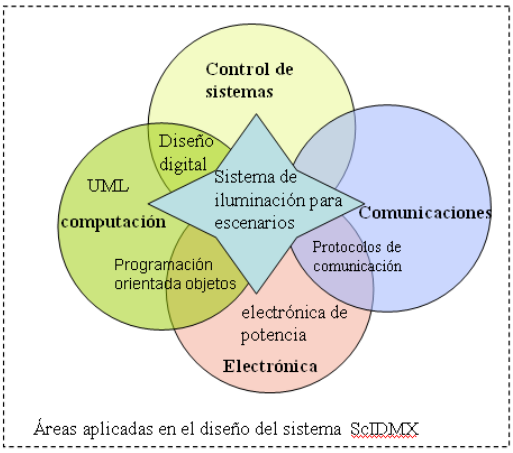

Figura 1.6. Áreas aplicadas en el diseño de un sistema de control de iluminación.

Cuando se diseña un sistema empotrado para el control de iluminación en escenarios, es necesario solucionar problemas de software y hardware; ya que por una parte el software debe incluir una serie de controles y arreglos que permitan al diseñador acceder de manera fácil a los parámetros y a los atributos del dispositivo de iluminación, y por otra parte, el hardware del sistema debe incluir una *arquitectura heterogénea* compuesta por procesadores y circuitos integrados específicos que implementen de manera efectiva los algoritmos que soportan la interfase con el usuario, para la ejecución de controles de luces sencillos y sofisticados. Además el hardware también debe ser capaz de controlar de manera individual o en grupo algunos dispositivos de iluminación para iluminar pequeños sectores del escenario de manera independiente.

Con el objeto de tratar con esta complejidad, el *proceso de diseño* de un sistema empotrado es analizado como una secuencia de pasos que transforman un conjunto de especificaciones descritas informalmente en una especificación detallada, que podrá ser usada para manufacturar un producto. Este proceso de diseño se estudia en el Capítulo 2.

Atendiendo a la problemática anteriormente expuesta, se considero la posibilidad de diseñar e implementar un prototipo de un sistema de control de iluminación llamado *Sistema de Control de Iluminación DMX (SciDMX)* con las siguientes características:

- Utiliza una computadora personal, esto hace que el sistema sea implementado fácilmente y utilizado como una arquitectura base para nuevas versiones del programa de control que incluyan nuevas funciones, realizando pequeños cambios o ninguno al hardware.
- Soporta el protocolo digital de comunicación DMX512, para el control de las luminarias.
- Utiliza una cadena tipo Daisy para conectarse con todos los dimmers.
- Cuenta con 12 botones deslizantes programables que se utilizan para regular la intensidad y el prendido y apagado de las luces.
- Dispone de funciones de *fade-in*[9] y *fade-out*[10] y funciones de sincronía entre dos o más lámparas.
- Además permite la programación anticipada de secuencias de encendido y pagado.

La industria del entretenimiento ha desarrollado una gran variedad de consolas, dimmers y protocolos de comunicación, algunos de ellos, los más representativos se estudian a continuación en los siguientes apartados, debido a su desarrollo histórico, consideramos conveniente incluir una breve reseña histórica para proporcionar un panorama mas detallado. Al mismo tiempo que se van introduciendo los conceptos y la terminología necesaria para presentar nuestro trabajo. En la sección 1.12 se presenta una introducción a los sistemas empotrados, y las bases de la metodología de desarrollo de los mismos, así como también los fundamentos del Lenguaje Unificado de Modelado.

1.1. *Consolas*

Una consola de iluminación es un dispositivo electrónico usado para controlar varias luces a la vez, en teatros, clubes, discotecas, auditorios, museos, etc. Muchas de las consolas modernas incluyen comandos específicos para controlar luces automáticas (es decir, luces que se mueven y cambian de

[9] Fade-in: Encendido progresivo de las luces.

[10] Fade-out: Apagado progresivo de las luces.

color), máquinas de niebla y otros dispositivos de efectos especiales. Las consolas varían en tamaño, precio y complejidad, que van desde un pequeño tablero programado hasta consolas sofisticadas que mueven luces de diferentes colores. Sin embargo, el propósito de todas ellas es el mismo: *Consolidar un sistema de control de luces organizado y de fácil uso, de tal manera que el diseñador de la iluminación se pueda concentrar en producir un buen espectáculo*. El protocolo de comunicación más ampliamente utilizado en la industria del entretenimiento en la actualidad es DMX-512. En la sección 1.3 se describen con mayor detalle los protocolos de comunicación utilizados en el control de la iluminación.

En la actualidad existen diferentes tipos de consolas y métodos de trabajo que seria imposible describirlas a todas ellas, Sin embargo las consolas se pueden clasificar en cuatro grandes grupos [Huntington, 2000]: *consolas de control programado de dos escenas (preset board), consolas de memoria*, *consolas computarizadas* también conocidas como controladores basados en PC y *controladores de movimiento de luz*. A continuación se describe de manera breve cada uno de estos tipos de consolas.

1.1.1. *Consolas de control programado de dos escenas.*

Es el tipo más básico de controlas de iluminación e introducen el concepto de pre-programación de las luces del escenario. Cada canal tiene dos conjuntos de controles (slider). El conjunto A se usa para poner la primera escena, y el conjunto B es usado para poner la segunda escena. Moviendo el control maestro A hasta el 100% se establece la escena 1. Cuando la escena 2 es requerida el control maestro B se mueve hasta el 100% y el control A es atenuado hasta cero; es decir, una escena se esta programando mientras que otra se está controlando las luces de la escena en ese momento y así sucesivamente. En este tipo de consolas el operador desconoce el efecto de las operaciones en los controles A o B, por lo cual, la iluminación de las escenas debe ensayarse previamente. Un ejemplo de este tipo de consola se muestra en la Figura 1.7; en ella se observa la consola Preset 12 de la firma *Artistic Licence*, la cual se puede usar también como una consola de 24 canales. El Preset A es la salida de los canales DMX 13 a 24 y el Preset B es la salida a los canales DMX 25 a 36; el master fader determina el nivel de salida de los canales DMX 13 a 36. La consola Preset 12 se alimenta con voltaje de 9V de corriente continua (9VDC).

Figura 1.7. Consola *Preset 12* de la firma *Artistic License*

1.1.2. *Consolas de memoria.*

Las consolas de memoria se utilizan ampliamente en instalaciones grandes y en teatros, preferentemente en producciones donde las escenas no cambian de un espectáculo a otro, tales como obras de teatro, ya que las escenas son diseñadas digitalmente, así que es menor la probabilidad de error humano y se requiere menos tiempo para producir el mismo resultado entre las secuencias de luces (o cues). Estas consolas han reemplazado a las consolas de control programado. Muchas de las consolas de memoria tienen un banco de *faders*[11] frecuentemente llamados submaster, los cuales se pueden programar para controlar un canal o múltiples canales. Un *canal* es un número o nombre que el diseñador utiliza para referirse a los dimmers o grupos de dimmers. Algunas de las consolas con memoria más sofisticadas, pueden usar a los submaster para controlar los efectos especiales y movimiento de las luces. Las consolas con memoria también pueden operar en forma analógica, para ello se dispone de un escritorio manual con una gran número de controles en "vivo" (es decir, en tiempo real), Figura 1.8. En la actualidad una consola de memoria esta diseñada con un microcontrolador o microprocesador, obteniendo de esta manara los siguientes beneficios:

- La posibilidad de tener un software de operación en lenguaje de alto nivel; que en muchas ocasiones permite la introducción de nuevas característica del producto de una manera muy versátil.
- La habilidad para respaldar datos del show en disquetes, y desplegar el estado de la iluminación sobre pantallas estándar.

Las consolas de memoria son computadoras personales con un teclado alfanumérico estándar; en donde todos los datos y la información operativa se introducen por medio del teclado. Estas consolas están disponibles en un amplio rango de tamaños con diferentes sistemas operativos. Si las instalaciones de iluminación inteligente se programan en una consola de memoria, el operador únicamente tiene que preocuparse por controlar la intensidad de la luz en los dimmers. Sin embargo, no debemos olvidar que cada luz tiene un número diferente de atributos que necesitan ser coordinados por la consola con la habilidad del operador [Simpson, 2003].

Figura 1.8. Consola de memoria.

[11] Botón deslizante.

1.1.3. *Consolas computarizadas.*

Las consolas computarizadas son relativamente nuevas y se basan en una combinación de hardware (computadora personal, PC) y software para controlar la secuencia de las luces; son excelentes para controlar eventos bien definidos de naturaleza secuencial, como las producciones teatrales o eventos corporativos. Cada secuencia se preprograma y se almacena con el tiempo límite de transmisión para la siguiente secuencia y así sucesivamente todas las secuencias; de esta forma, la iluminación del evento se ejecuta exactamente igual cada noche, permitiendo al operador y al director del escenario ejecutar la secuencia de luces en los momentos y puntos apropiados una y otra vez. Estas consolas generalmente cuentan con algunos controles manuales para funciones de fading (desvanecimiento de las luces) de tal manera que el operador puede manipular la intensidad de las luces en la escena de acción [Simpson, 2003].

Los dimmers, las instalaciones automatizadas de luces y otros dispositivos de iluminación generalmente no tienen interfaces estándar para computadoras, en su lugar se utilizan conectores seriales del tipo *USB*, puertos serie e inclusive puertos paralelos para conectarse a dispositivos llamados *controladores DMX-512* y paneles de sub-maestros a la computadora; de esta manera, el sistema permite construir sistemas de iluminación adecuados para el presupuesto y necesidades del usuario final con posibilidades de incrementar el número de salidas DMX o adicionar paneles de control de iluminación. El tipo de consola implementada en el sistema es la consola computarizada, la cuál utiliza el puerto paralelo de la PC.

1.1.4. *Controladores de movimiento de luces.*

Los *controladores de movimiento* de luz representan una modalidad sofisticada de las consolas de memoria. Ellos son capaces de controlar instalaciones ordinarias de luz hasta instalaciones de *luces inteligente*; las cuales tienen más parámetros a controlar que las luces convencionales estos *parámetros* incluyen: tildeo, paneo, intensidad, color, obturadores, focos y gobos. Los controles incluyen un arreglo de botones que permiten al operador seleccionar la instalación de luz que se desean controlar y un joystick, o encoders rotacionales para controlar las atributos de las instalaciones tales como la orientación (paneo y tíldeo), focos, color, gobos, etc. encontrados en este tipo de luz. Escritorios más avanzados tienen típicamente uno o más pantallas tipo touch screen y presentan un GUI (Interfaz Grafica de Usuario) que integra todos los aspectos de la luz.

1.2. *Introducción a los dimmers*

Los *dimmers* son dispositivos electrónicos especializados capaces de regular la intensidad de fuentes eléctricas de luz utilizadas en teatros, conciertos, discotecas, eventos corporativos y de entretenimiento aplicando la técnica de conmutación. Los dimmers fueron inventados en 1890, por Glanville Woods para evitar los incendios en los teatros ya que los métodos utilizados para controlar la intensidad de las lámparas eran peligrosos y frecuentemente causaban incendios [Smith, 2005]. Woods buscó un método económico y efectivo para regular la intensidad de las luces y así se creo la primera versión del dimmer moderno; el cual fue de *resistencia variable*.

El principio de funcionamiento de los primeros dimmers resistivos era sencillo: una resistencia variable se conectaba en serie con la carga y se variaba la corriente, de esta manera se regulaba la brillantez de las lámparas en los escenarios; pero existían varios inconvenientes en este tipo de dimmers:

- Ocupaban amplios espacios en los teatros.
- Disipaban grandes cantidades de calor en la carga la cual según [Simpson, 2003] era alrededor del 30%.
- Eran fijos para una carga específica, de tal manera que un dimmer de 1000W solo era adecuado para una lámpara de 1000W.

Para evitar las grandes pérdidas de energía, se desarrollaron los *dimmers de reactancia variable*, sin embargo su construcción mecánica era cara, lo que los hacia inaccesibles para muchas compañía de teatros. Posteriormente se diseñaron los *dimmers con autotransformadores* los cuales tienen una salida sinusoidal, por lo que no introducían armónicos; sin embargo eran grandes, pesados y caros. En la Figura 1.9 se muestran ejemplos de los dimmers resistivos y de autotransformadores.

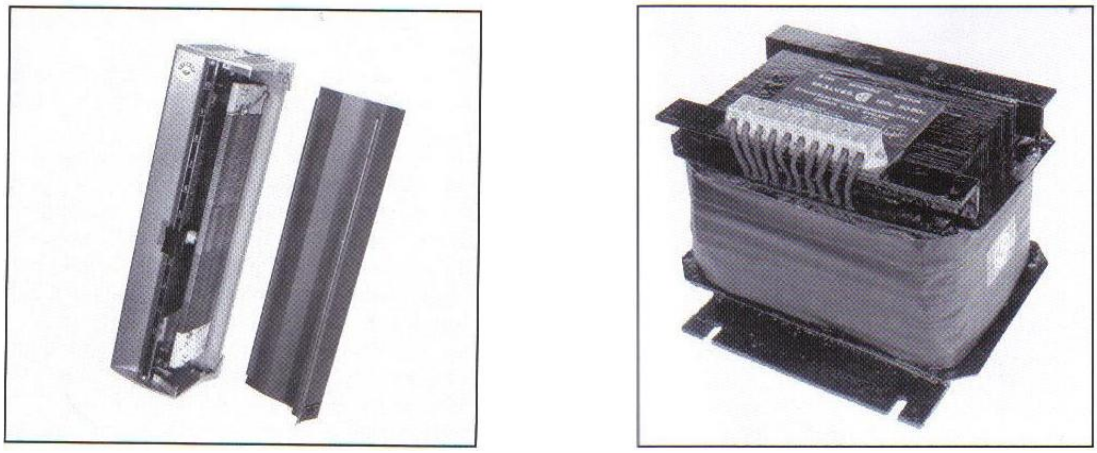

Figura 1.9. Ejemplos de dimmers: a) resistivos, y b) de autotransformador.

Con la invención del tiristor, los dimmers se transformaron en pequeños, económicos y eficientes. Los tiristores se usaron en el control de iluminación en la primera parte de la década de los 60's y durante 40 años han formado la base de control de iluminación profesional, ya que son más robustos y tienen la capacidad de soportar altas corrientes repentinas causadas por los fallos en el filamento de las lámparas de tungsteno. Los dimmers profesionales usualmente están construidos sobre un principio *modular*. Cada módulo representa uno o dos canales de dimmer. Los módulos son independientes, y están diseñados para ser remplazados fácilmente, ya sea por conexión de enchufe o por terminales de fácil conexión. Los dimmers profesionales pueden resistir ciertas perturbaciones producidas por variaciones en la frecuencia de la fuente [Simpson, 2003].

Todos los sistemas dimmers operan sobre la base de dos técnicas para limitar el flujo de corriente en las lámparas, los cuales son:

- Variación de voltaje.
- Variación en el intervalo de tiempo, en el cual la corriente fluye durante cada ciclo de la corriente alterna.

En la siguiente sección se presenta la segunda técnica por ser la más difundida y es la que se aplica en el presente trabajo.

1.2.1. *Dimmers diseñados con tiristores*

Todos los circuitos dimmer sobre la base de tiristores requieren que el dispositivo se dispare en algún punto predeterminado después que la señal sinusoidal cruza por cero. La técnica es conocida como control por ángulo de fase, la cuál consiste en controlar el tiempo de disparo o de conducción del tiristor, para regular la corriente que se entrega a una carga (o lámpara) y de esta manera, controlar la potencia que consume.

Un triac es una forma de tiristor que permite que ambos semiciclos de la corriente alterna (CA) fluyan a través de la carga. El triac es disparado cuando una señal de baja energía se aplica en su terminal G (Gate). El semiciclo positivo de la señal de CA pasará por el triac siempre que G sea activo, de esta manera, la corriente circulará de arriba hacia abajo (terminal MT2) como se muestra en el circuito de la Figura 1.10. Mientras que en el semiciclo negativo pasará por el triac siempre y cuando exista una señal de disparo en la entrada G, de esta manera la corriente circulará de abajo hacia arriba (terminal MT1).

El dispositivo que proporciona la señal G en ambas direcciones de la corriente es conocido como diac. El diac es un diodo bidireccional que únicamente permite disparar el flujo de corriente o voltaje cuando este ha encontrado un cierto nivel preestablecido. El diac controla el voltaje en la entrada G del triac y permite la transición de prendido a apagado de manera suave.

El intervalo de tiempo (retraso) a partir del cruce por cero de la corriente alterna hasta el tiempo en el cual el triac se dispara se conoce como ángulo de disparo y se representa por α. Los rangos de α varían de 0º (máxima potencia) hasta 180º (mínima o nula potencia). Controlando el ángulo de disparo, el voltaje rms suministrado a la carga cambia y por lo tanto, la intensidad de las luces también cambia; como se muestra en la Figura 1.10.

Cuando la corriente alterna cambia su dirección, el triac se apaga; esto hace que la corriente en la carga sea cero en cada semiciclo de CA. Por lo tanto, para que la lámpara se active continuamente, el triac necesita dispararse durante ambos semiciclos de la onda sinusoidal CA; para asegurar que la carga promedio de la corriente no sea cero.

1.2.2. *Dimmer controlado remotamente por tiristor*

La Figura 1.11 muestra el diagrama a bloques de un dimmer controlado remotamente por tiristores. El diagrama es el mismo sin tener en cuenta si el circuito final es digital, analógico o híbrido. Los circuitos de control consisten de tres partes principales:

Un detector de cruce por cero, que detecta el momento en el cual el la corriente alterna de la línea cruza por cero. Para controles de fase esto es la información de temporización esencial, ya que el disparo de los tiristores es medido desde este punto.

Un circuito de disparo, que compara el momento en que se detecta el cruce por cero con la señal de entrada, y dispara los tiristores después del retardo de fase apropiado.

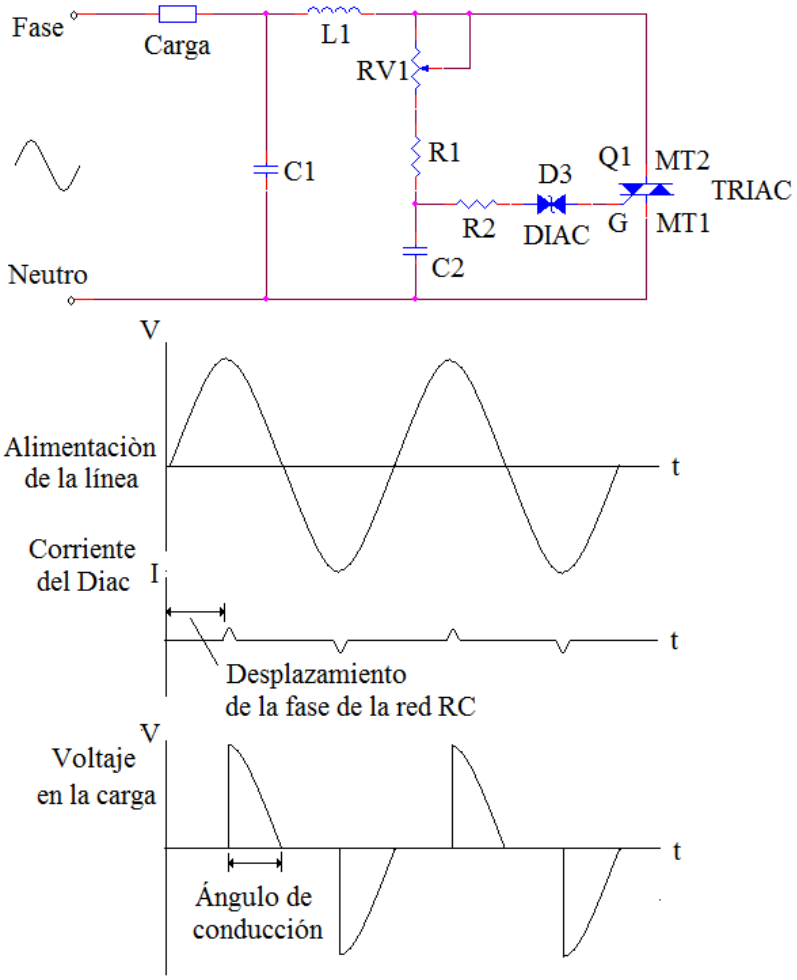

Figura 1.10. Esquemático de un dimmer analógico.

Un circuito de control. Este puede ser únicamente un circuito de voltaje de referencia, que permita al dimmer analógico ser controlado por una señal analógica de 0-10 V. Mientras que en un *dimmer analógico automático*, el circuito incluye circuitos de tiempo tipo rampa para producir desvanecimiento automáticos a niveles programados.

Opcionalmente en la salida de los dimmers se pueden incluir sensores que detectan variaciones en la carga presentadas por fallas de las lámparas, corrientes o temperaturas elevadas; estas señales son enviadas al sistema de control, para desactivar al dimmer correspondiente.

El aislamiento entre el circuito de disparo y el tiristor es logrado correctamente en el momento de disparo. Antes de la llegada de opto-aisladores confiables de voltaje alto. En la actualidad el disparo se realiza usando tiristores opto aislados (para disparo de tiristores) o triacs opto aislados (para disparo de triacs).

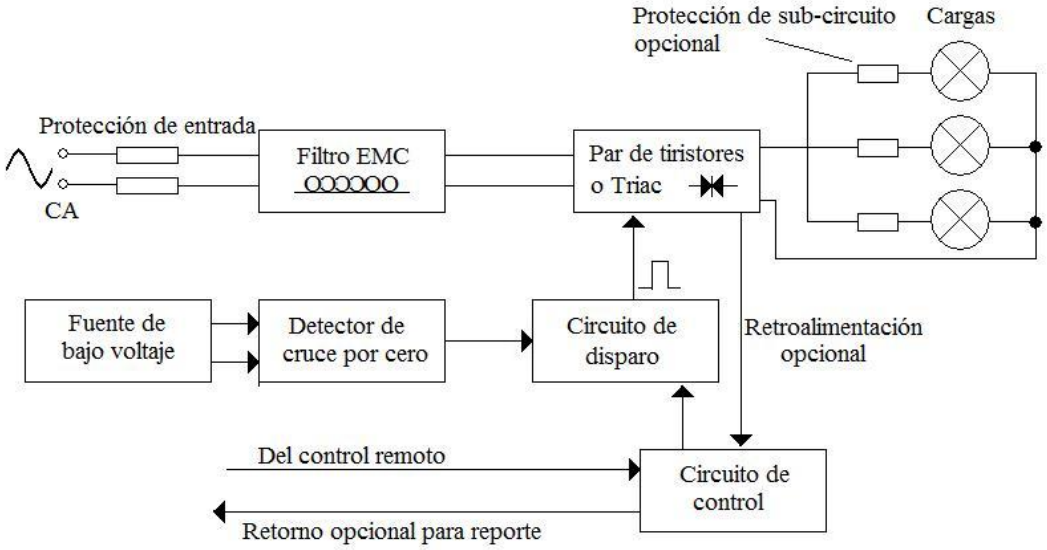

Figura 1.11. Diagrama a bloques de un dimmer controlado remotamente por tiristores.

Como con todos los circuitos de potencia, los problemas se incrementan cuando los voltajes o corrientes se conmutan a velocidades muy altas. Las altas velocidades de conmutación de la corriente y la anti-simetría de los disparos del Triac genera ruido, armónicos e interferencia electromagnética. El ruido del circuito dimmer podrá ser transmitido en circuitos de potencia y causar problemas en los equipos de sonidos y audio. Los armónicos pueden estar en la gama de los MHz dependiendo de los niveles de potencia del sistema y del número de fases en el mismo, y puede causar calentamiento en los transformadores y conductores. Además si se usan microprocesadores para el control del circuito dimmer, las interferencias pueden afectar severamente el desempeño del sistema digital. Además el flujo del campo magnético también causa vibraciones en a estructura del foco el cual se podría dañar. El circuito dimmers debe contener filtros de alta frecuencia para remover algunos de los armónicos y ruido en el circuito [Simpson, 2003].

1.2.3. *Circuitos de dimmer.*

Un microcontrolador (MCU) proporciona un método sencillo y económico de implementar la funcionalidad de un dimmer [Gadre, 2001]. Este puede procesar casi simultáneamente las entradas de cualquiera de las señales de control requeridas en las luminarias, y controla la potencia para cada uno de los canales individuales.

La tarea principal del MCU es controlar en tiempo real el ángulo de la fase. Para ello su firmware[12] debe ejecutar dos tareas: calcular el retraso entre el cruce por cero y el encendido del triac, conectándolo en el punto apropiado en el ciclo de CA [Couedic, 2000], [Microchip, 1997].

El funcionamiento digital del dimmer profesional introduce varios beneficios entre los que se encuentran:

[12] Software ejecutable que esta almacenado en la memoria ROM del MCU.

- Asegura que el rendimiento en un dimmer muti-canal, todos los dimmers tendrán un rendimiento equivalente.
- Simplifica la introducción de nuevas funciones especiales, como por ejemplo el cambio de temperatura en las luces.
- Permite la supervisión constante la intensidad de la corriente y de otras señales de control de las luces
- Simplifica el uso de un control tipo multíplex.

En la Figura 1.12 se muestra el diagrama a bloques de un dimmer profesional. El diagrama es representativo y muestra una gama de funciones. No todos los dimmers incluyen todas las mostradas y algunos dimmers pueden incluir funciones que no se incluyen aquí.

Este diseño esta basado en un microprocesador o microcontrolador. El dimmer en este diseño todavía requiere de tiristores y circuitos de disparo aislados y un detector de cruce por cero.

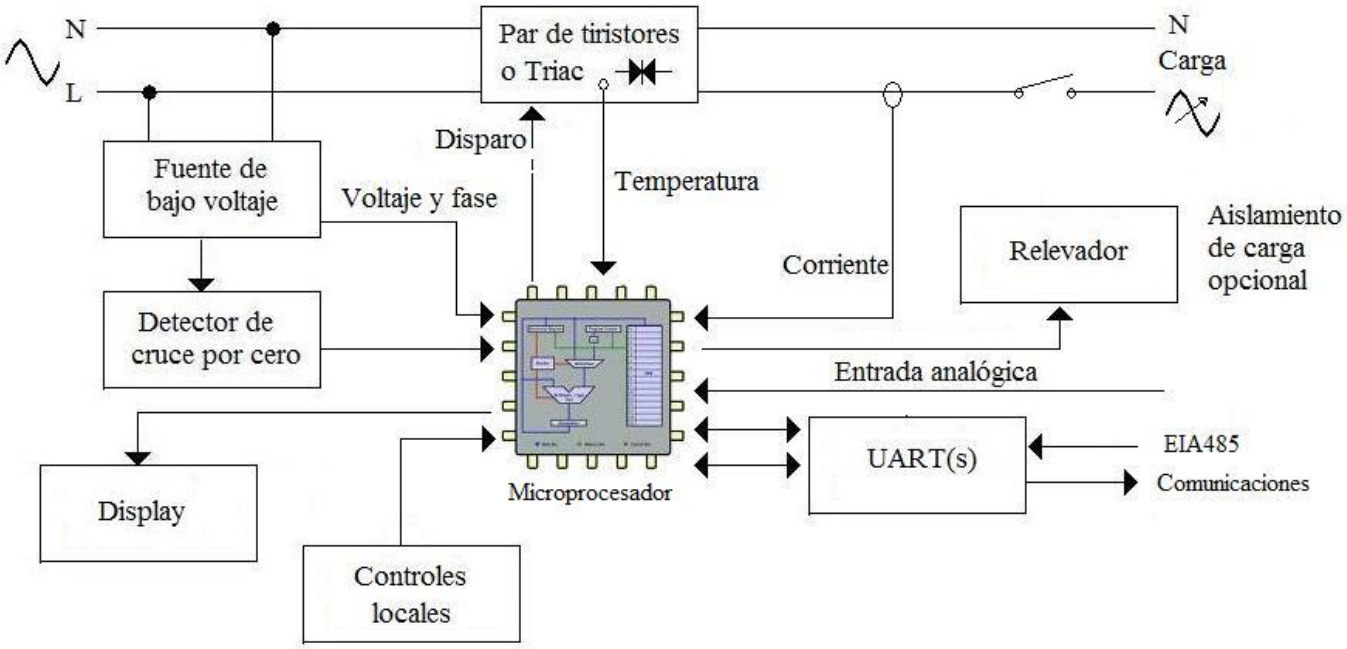

Figura 1.12. Diagrama a bloques de un dimmer profesional.

1.3. *Protocolos de comunicación utilizados en el control de la iluminación*

La parte central de toda instalación de iluminación en los escenarios es el sistema de control o consola de iluminación. Por medio de éste, se logra la regulación de los dimmers que a su vez regulan la intensidad de las luminarias para obtener un cierto nivel de brillantez y de esta manera producir el efecto visual deseado en el escenario. Todas las consolas estudiadas con anterioridad tienen como función principal la de regular o balancear la intensidad de la luz, en diferentes niveles que son determinados por el operador de luces. Por lo tanto, se podría decir que la consola es un sistema de control que proporciona una *interfaz* entre el operador y los dimmers, como se muestra en la siguiente Figura 1.13. El estado de cada dimmer se establece por el operador desde la consola, se envían las señales de control para cada uno

de los dimmers, que a su vez regulan la intensidad de las luces. La consola se comunica con los dimmers para que las luces adquieran el efecto deseado por el operador [Scott, 1996].

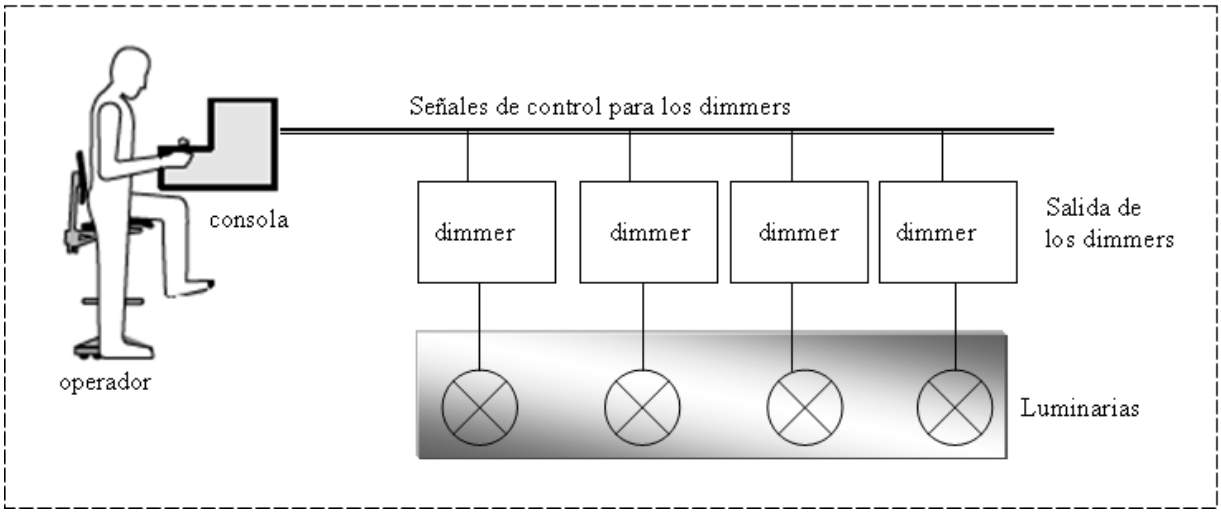

Figura 1.13. Interfaz entre el operador y el sistema de iluminación.

Existen una gran cantidad de protocolos tanto *analógicos* como *digitales* utilizados para la comunicación entre la consola y los dimmers. Los protocolos analógicos casi ya no utilizados, son llamados *controles analógicos lineales*, y estaban formados por una consola que enviaba voltaje o corriente directa (CD) a cada uno de los dimmers a través de cables, los cuales a su vez proporcionaban un voltaje variable a las luminarias. Estos sistemas eran voluminosos, obstaculizantes y costosos con múltiples cables desde la consola de control hacia los dimmers; ya que cada dimmer necesitaba de un canal. Por ejemplo, si se deseaba controlar 24 dimmers se necesitaba una consola con 24 canales. Frecuentemente los controles analógicos requerían de adaptadores de cables, debido a los diferentes tipos de contactos y enchufes utilizados por los fabricantes; además era muy común que las consolas y dimmers analógicos trabajaran con voltajes de control y polaridad diferentes, por lo que era necesario incluir inversores de voltajes y amplificadores para hacerlos compatibles [Simpson, 2003], [Mobsby, 2005].

Debido a todos estos problemas, las compañías de renta de estos equipos apoyaron la investigación de métodos alternativos para controlar las luminarias, que pudieran reducir el tiempo de conexionado, la cantidad de equipo en la gira, y la rentabilidad del sistema. Por lo que a principios de 1980, los fabricantes empezaron a desarrollar sus propias soluciones para reducir el número de cables, de esta manera surgió la idea de utilizar *protocolos multiplexados digitales*, los cuales pueden transmitir señales eléctricas que representan valores numéricos digitales en lugar de niveles de voltaje. En estos protocolos un solo transmisor envía paquetes de valores numéricos para múltiples receptores conectados en una sola línea. Por ejemplo, la marca inglesa *Strand* desarrolló un sistema llamado *"Estándar Multiplexado Analógico"* que permite enviar hasta 384 canales con un cable para micrófono. Por su parte *Colortran* (de Estados Unidos) lanzó su protocolo digital denominado *CMX* y en Europa, *ADB* de *Belgium* desarrolló los protocolos *S20* y *AVAB,* de los cuales existe su versión para Scandinavia y Francia.

De esta manera se desarrollaron una gran variedad de protocolos de comunicación tales como*: AMX 192 (USITT), D54, Pmx, SMX , S20 Avab* y *Control de Shows MIDI (MSC).* Estos protocolos no tienen una estructura intrínseca, sino que únicamente representan secuencias de números. Todos los

receptores reciben todos los paquetes pero solo se selecciona uno de ellos por medio del hardware [Von, 2001]. Sin embargo estos protocolos no son compatibles entre sí, por lo que el usuario final debe convertir el protocolo de la consola de control al protocolo analógico. Esto ocasiona que las compañías de renta, utilicen cajas convertidoras y sistemas de demultiplexado analógico lo cual es un verdadero problema. Además algunas fábricas iniciaron el empleo de sistemas de transmisión más eficientes, como el múltiplex analógico o digital. Esto creó más confusión porque se crearon nuevas incompatibilidades además de las ya existentes. Ya que se trataba de protocolos particulares, incompatibles entre ellos, y ninguna de las casas productoras de la competencia podía utilizar productos de la otra para no favorecerla.

Para evitar esto, el Instituto de Tecnología Teatral de los Estados Unidos (USITT) investigó las formas de producir estándares que todos los fabricantes pudieran adoptar. Inicialmente se basaron en el estándar CMX de Colortran Inc., Lighting Methods. En 1986 se inició la estandarización de los protocolos multiplexados, uno analógico y otro digital, llamados AMX 192 y DMX 512, respectivamente. Estos estándares empezaron a ser aceptados universalmente, siendo el más difundido el protocolo DMX 512; porque es sencillo de implementar y además permite la compatibilidad entre consolas y dimmers de diferentes fabricantes. El protocolo ha sufrido cambios y en 2004 una nueva versión llamada DMX512-A es adoptada, la cual define con mayor claridad las configuraciones usadas en el conector XLR de cinco pines, estos cambios son regulados por la Asociación de Tecnología y Servicios de Entretenimiento (ESTA, por sus siglas en inglés: Entertainment Services and Technology Association) [URL 2]. En 1990, el protocolo DMX512 se había convertido en el más popular en la industria del entretenimiento y se podría asegurar que este virtualmente ha reemplazado a todos los protocolos anteriormente mencionados.

Por ser éste el protocolo que se utiliza para diseñar nuestro sistema de control de luces, a continuación se presenta un breve análisis de las características más importantes de este protocolo.

1.3.1. *Introducción a DMX512*

El protocolo de transmisión de datos Digital MultipleX, abreviado DMX512 o simplemente DMX [E1], se utiliza para controlar la iluminación en los escenarios; de teatros, museos, auditorios y discotecas. Además esta diseñado para soportar datos de control repetitivo de un controlador simple a uno o más receptores. Se basa en el estándar internacional RS485 y fue desarrollado con el propósito de obtener un protocolo de comunicación estándar y eficiente que enlazará a las consolas de control con los dimmers de diferentes fabricantes [Rosenberg, 2003]. Este protocolo se ha convertido en el primer método de control no solo para controladores de enlace y dimmers, sino también para conectar dispositivos de efectos especiales como: generadores de humo [Von, 2001], scrollers, gobos rotatorios, strobos (luces de destello de alta intensidad), balastros, luminarias automatizadas y algún otro dispositivo que pueda ser controlado digitalmente. Sus principales características son:

- Es un protocolo de comunicación asíncrono simplex, con una tasa de trasmisión de 250 Kbit/s.
- Es inmune al ruido, ya que transfiere la información a través de una conexión balanceada.
- Puede controlar la iluminación de espectáculos a gran escala (hasta 512 canales), cada canal genera un valor de 8 bits para controlar un dispositivo determinado.
- La señal DMX512 puede ser transmitida confiablemente a una distancia de hasta 1500 pies (457.2 m), lo cual es mucho más de lo que se necesita en un auditorio, teatro o museo.
- No corrige errores y por lo tanto, no debe ser utilizado para controlar efectos pirotécnicos, de sonido o para mecanización de escenarios [E1.11-2004, 1990].

El sistema que utiliza este protocolo como medio de comunicación debe estar formado por un transmisor y un receptor. El dispositivo transmisor (o consola) genera la señal de control DMX correspondiente a una dirección que activa remotamente a uno de los dimmers. Mientras que el dispositivo receptor recibe la información y la decodifica para seleccionar el canal correspondiente, quien a su vez prenderá la luz apropiada con la intensidad adecuada. La señal DMX512 se basa sobre el estándar diferencial EIA-485, lo cual significa que: un enlace puede soportar hasta 32 dispositivos receptores conectados a un simple canal de datos por medio de una *cadena tipo Daisy* como se muestra en la siguiente Figura 1.14. Este canal de datos requiere de cables de *par trenzado*, los cuales tienen una alta inmunidad a los disturbios eléctricos y electromagnéticos comunes con impedancia característica nominal de 100 a 120Ω.

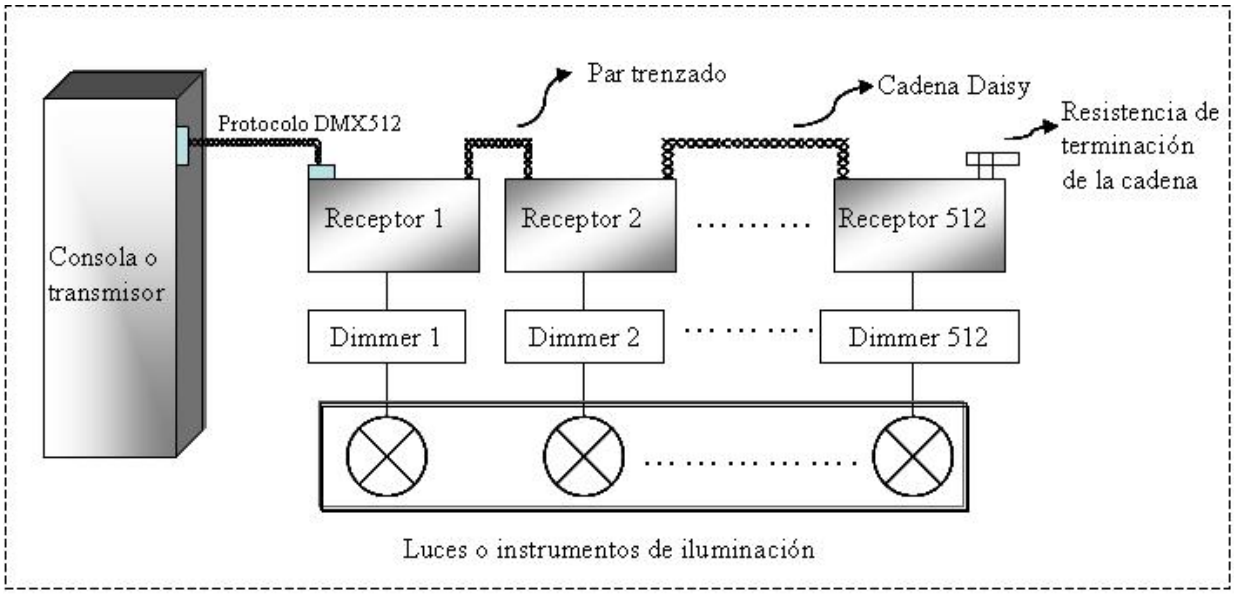

Figura 1.14. Red DMX simple con un transmisor.

De la Figura 1.14 se puede observar que el último eslabón en la cadena siempre finaliza con una resistencia cuyo valor esta en función de la impedancia característica del cable. La terminación equivocada o la falta de ella es la causa más común del mal funcionamiento de todo el sistema, estas fallas producen parpadeos de las lámparas o movimientos peculiares aleatorios de las luces móviles o scrollers [Mobsby, 2005]. El estándar DMX512 utiliza un protocolo simple de comunicación serial asíncrono de ocho bits manejados directamente por un Transmisor-Receptor Asíncrono Universal (UART) estándar. El medio es manejado usando técnicas de transmisión de datos balanceados en base al estándar industrial ANSI/TIA/EIA-485-A-1998. La conexión física entre los dispositivos es mediante un conector XLR de 5

pines, de los cuales únicamente tres son usados, los otros dos se reservan para un segundo enlace de datos opcional, el cuál rara vez es implementado. Los conectores hembra se usan en la salida de las consolas mientras que los conectores machos se utilizan en las entradas de los dispositivos receptores. En la Tabla 1.1, se muestra la asignación del conector XLR de 5 pines, y en la Figura 1.15 se muestra la asignación de los pines y el aspecto físico del conector XLR.

Tabla 1.1 Asignación de señales en los pines del conector XLR.

Uso	**Pines del conector XLR**	**Función DMX512**
Referencia común	1	Común del enlace de datos
Enlace de datos principal	2	Dato 1 -
	3	Dato 1 +
Enlace de datos secundario (opcional)	4	Dato 2 -
	5	Dato 2 +

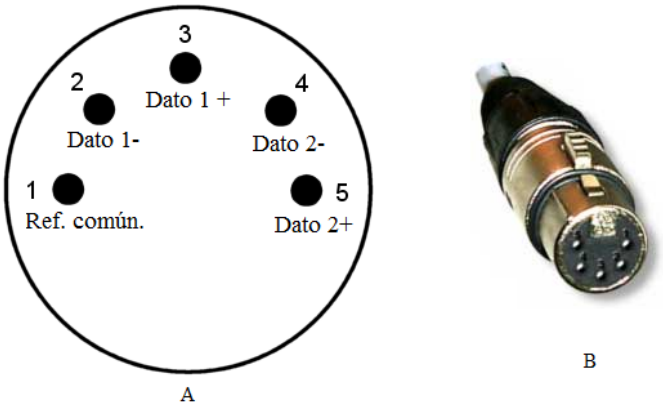

Figura 1.15. Conector XLR de 5 pines. A) Asignación de los pines, B) aspecto físico del conector XLR.

En el protocolo DMX512 se utiliza una señal analógica, la cual es muestreada en un formato de ocho bits, lo cual permite obtener 256 diferentes combinaciones de códigos digitales correspondientes a los diferentes niveles de intensidad de las lámparas. Esta señal se transmite a los receptores en forma de pulsos asíncronos en un formato serial, sin paridad y de manera repetitiva. La velocidad de muestreo es aproximadamente de 44 veces por segundo, la cual es una tasa de *refrescamiento* lo suficientemente rápida para muchas aplicaciones. El tiempo de bit es de 4 μs.

Una transmisión simple de todos los datos DMX512 es llamado un paquete [Rosenberg, 2003]. Un paquete típico esta formado por un *Start Code*, seguido por los valores de cada canal. Un bit de inicio y dos bits de stop separan cada canal. Para analizar cada una de las partes que forman la señal DMX512, se considera la Figura 1.16, en ella se puede observar que cuando inicia la transmisión de un paquete DMX, se envía una señal *Break*, que permite a los receptores reconocer el inicio de un nuevo paquete de datos, esta señal se mantiene en un nivel bajo por lo menos 88 μs (tiempo de dos tramas). Inmediatamente después se envía un pulso alto por un periodo corto de tiempo entre 8 μs y 1 segundo conocido como

MAB (*Mark After Break*), el cual advierte a los receptores que la siguiente transición de alto a bajo será el inicio del dato, por esta razón esta señal es considerada como el pulso de sincronización. Después se envía la señal *Start Code* de ocho 0´s que indican a los receptores que la siguiente información son niveles validos de intensidad. Esta señal además incluye un bit de inicio representado como *Bit Start*, dos bits de paro o *Stop*, y un pulso alto *Mark* lo que significa que se necesitan 11 bits para generar esta señal. Además aunque se permiten otros *Start Codes*, los receptores no deben actualizar sus valores sino se recibe un *Start Code* igual a 00h [Mobsby, 2005], [E1.11-2004, 1990].

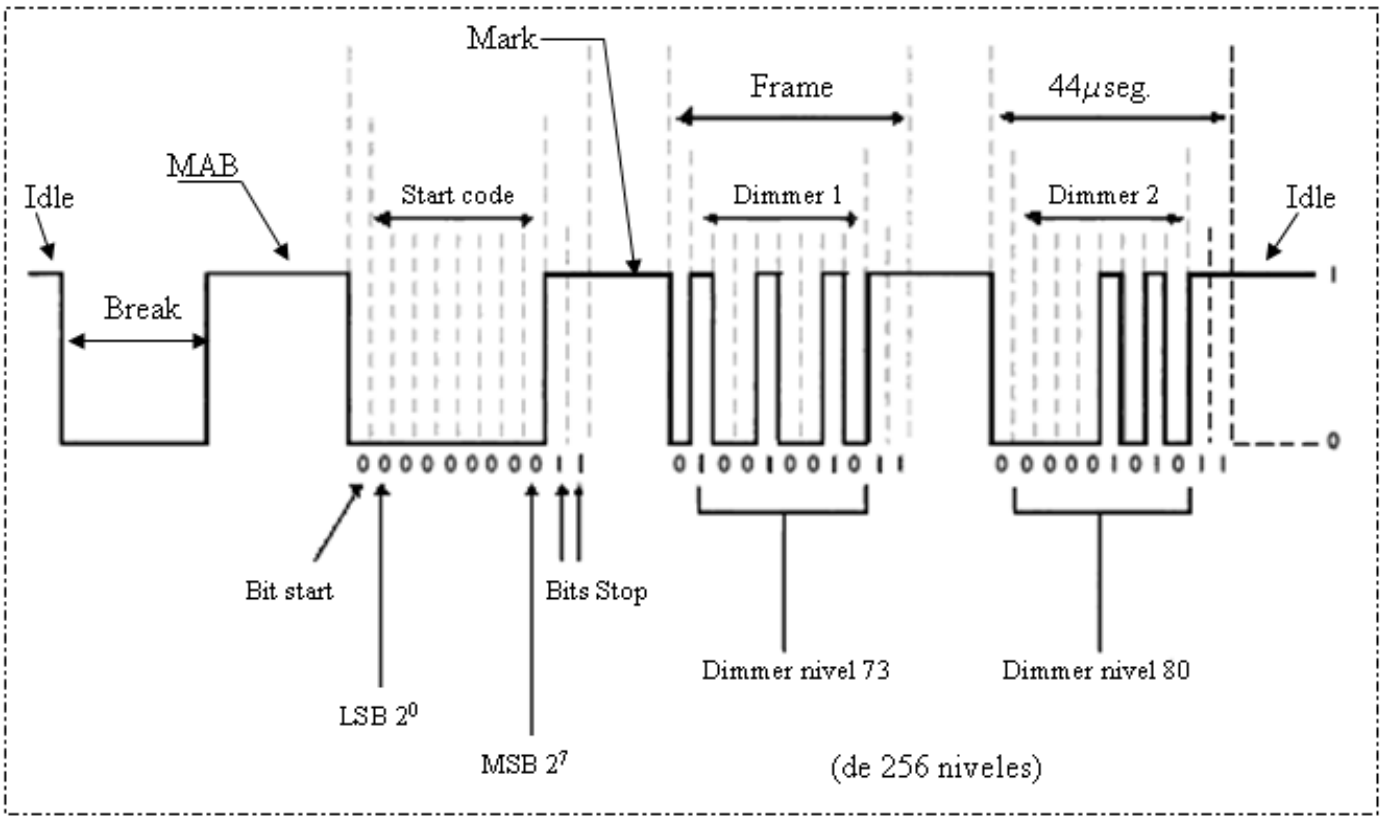

Figura 1.16. Diagrama de tiempos de una señal DMX512

Después de la señal Mark, se envían secuencialmente los niveles válidos de intensidad (*Frames*), los cuales deben estar entre 0 a 255 decimal (00h a FFh). El valor 0 representa una salida de dimmer apagada o mínima, y un valor de 255 representa una salida completa o del 100% de la intensidad. Un dimmer debe responder a los incrementos del valor de la señal DMX512 de 0 a 255, aumentando gradualmente del nivel mínimo (off) al nivel máximo (on). La relación exacta entre los valores de señal DMX512 y la salida del dimmer esta fuera del contexto del estándar DMX512. Cuando se ha terminado de enviar un paquete, la señal DMX envía un pulso alto llamada *Idle* que representa el estado inactivo de la señal.

1.4. *Metodología de diseño de sistemas empotrados*

1.4.1. *Introducción a los sistemas empotrados*

Los sistemas digitales se diseñan para dos tipos de aplicaciones: *sistemas de propósito general* y sistemas de *aplicaciones específicas*. Los primeros se pueden programar para correr diferentes aplicaciones; su uso más común es el *cálculo*; ejemplos de estos sistemas son las computadoras y las estaciones de trabajo. Mientras que los segundos se diseñan para una aplicación en *particular*, por lo que son comúnmente referidos como *sistemas empotrados*. Un *sistema empotrado* emplea una combinación de recursos de hardware (microprocesador) y software (un programa computacional) para realizar una única función o un conjunto limitado de funciones [Kumar, 1993]. En contraste con una computadora de propósito general que pueden realizar diferentes tareas dependiendo de la programación, un sistema empotrado está dedicado a tareas específicas, por lo cual los diseñadores podrán optimizar el sistema, reducir el tamaño y el costo del producto, o incrementar su seguridad y desempeño, además se producen masivamente, beneficiando la economía a escala.

Los sistemas empotrados tienen un amplio rango de aplicaciones que van desde automóviles, elevadores, instrumentos médicos, hornos de microondas, televisores, calentadores hasta teléfonos celulares, en la Figura 1.17 se muestran algunos ejemplos de las aplicaciones de los sistemas empotrados.

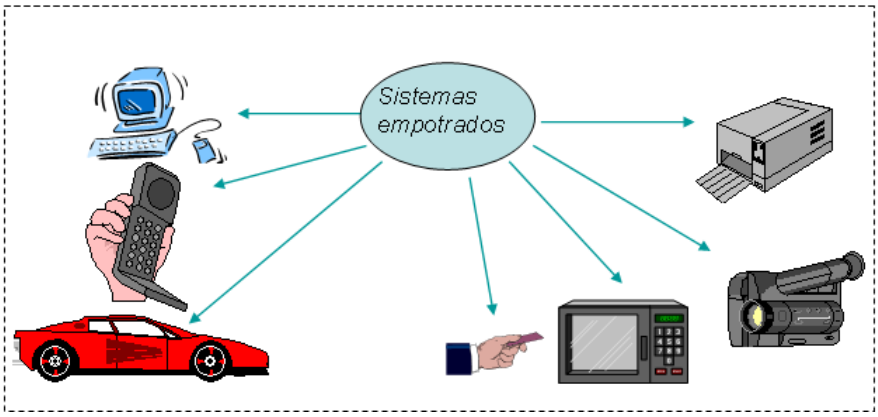

Figura 1.17. Ejemplos de los sistemas empotrados.

Mientras que los sistemas empotrados han tenido un notable incremento en usos y aplicaciones, las mejoras en los procesosH de diseño para tales sistemas no han sido exitosas [Gajski et al, 1994]; lo cual ocasiona una separación entre la evolución de la tecnología de componentes y sus aplicaciones en sistemas empotrados de computo. Mientras, que cada seis meses se anuncian nuevos procesadores y/o circuitos integrados (CI) programables / reprogramables con un aumento anual del 50% en desempeño, estos CI se podrán aplicar en los sistemas empotrados varios años después.

Las principales características de un sistema empotrado son [Vahid, 2000]:

Concurrencia: Los componentes del sistema funcionan simultáneamente, por lo que el sistema deberá operar a la vez.

Fiabilidad y seguridad: El sistema debe ser confiable y seguro frente a errores, ya que puede requerir un comportamiento autónomo. El manejo de estos errores puede ser a través del hardware o software.

Interacción con dispositivos físicos: Los sistemas empotrados interaccionan con el entorno a través de dispositivos de E/S no usuales, por lo que suele ser necesario un acondicionamiento de las diferentes señales.

Robustez: El sistema empotrado impondrá la necesidad de la máxima robustez ya que las condiciones de uso no tienen que ser buenas, sino que pueden estar en condiciones extremas de funcionamiento.

Bajo consumo: El hecho de poder usar el sistema en ambientes hostiles puede implicar la necesidad de operar sin cables. Por lo tanto, un menor consumo implica una mayor autonomía de operación.

Precio reducido: Esta característica es muy útil cuando hablamos de características de mercado.

Pequeñas dimensiones: Las dimensiones de un sistema empotrado dependen también del espacio disponible en el cual el dispositivo se va a ubicar.

Los sistemas empotrados se clasifican en tres grupos dependiendo de la interacción de estos sistemas con su entorno:

- *Sistemas empotrados reactivos*. Son aquellos sistemas que siempre interactúan con el exterior, de tal forma que la velocidad de operación del sistema deberá ser la velocidad del entorno exterior. Como por ejemplo, el sistema de control aéreo de un aeropuerto, ya que la velocidad del sistema dependerá de la velocidad con la que lleguen los datos de los diferentes aviones que se acerquen o salgan del mismo.
- *Sistemas empotrados interactivos*: Son aquellos sistemas que siempre interactúan con el exterior de tal forma que la velocidad de operación del sistema deberá ser la velocidad del propio sistema empotrado como por ejemplo las máquinas de videojuegos; ya que la velocidad del sistema depende de él mismo y el exterior es decir, el usuario del videojuego, se debe de adecuar a su velocidad.
- *Sistemas empotrados transformables*: Son aquellos sistemas que no interactúan con el exterior, únicamente toma un bloque de datos de entrada y lo transforma en un bloque de datos de salida, que no es necesario en el entorno. Ejemplo son los postes de publicidad electrónicos, en los que no existe ningún tipo de interactividad excepto la entrada de datos iniciales y la salida de datos finales [Vahid, 2000].

Cada vez son más las funciones que se integran en los sistemas empotrados, con los objetivos de mejorar su desempeño, ofrecer costos competitivos y aumentar su adaptabilidad, esto provoca que el diseño de estos dispositivos sea un proceso complejo debido a la gran cantidad de requerimientos que se deben satisfacer. Los académicos y aficionados han observado recientemente que los sistemas tienen un conjunto de principios comunes que conciernen a varios campos de la ingeniería. Los nuevos productos y sistemas requieren del involucramiento de profesionales en diversas disciplinas, tal como ingeniería de software y electrónica, etc.

El problema del diseño de un sistema empotrado es tan complejo que la primera metodología consiste en la estructuración y jerarquización del sistema, de forma tal que su diseño pueda ser abordado por partes. La jerarquización consiste en la subdivisión del sistema en bloques de forma recursiva para

conseguir que el nivel de complejidad de cada parte sea abordable, ya que en la mayoría de los casos tratar todo el sistema de forma unitaria sería imposible.

Además debido a la gran cantidad de variables que se deben controlar o considerar, es necesario enfrenarnos al diseño en diferentes niveles de abstracción, lo que nos permite reducir la cantidad de información que es necesario manejar en cada momento [Solá et al, 2003]. Los distintos niveles de abstracción estructuran al sistema (o a sus partes) desde el nivel circuito (el más fundamental) hasta el nivel arquitectura (o sistema), en el que se puede describir el sistema completo. Por lo tanto, se podría decir que la abstracción es el proceso mediante el cual se define un conjunto reducido de propiedades y elementos del sistema, con los cuales es factible abordar el problema de su diseño, especificación e implementación.

1.5. *Metodología de desarrollo de un sistema empotrado*

Es una práctica muy común identificar varias fases en el diseño de un sistema, cada una de ellas involucra ciertos procesos y tareas que tienen que ser realizadas por los diseñadores. Las fases principales del clásico modelo en cascada [Boehm, 1976], [Royce, 1970] son: análisis de requerimientos y especificación, diseño, implementación, prueba y mantenimiento. Durante los últimos 20 años, muchas variaciones de este modelo han sido propuestas, además de bastantes y diferentes enfoques para el ciclo de desarrollo, llamado a veces simplemente ciclo de vida o proceso de diseño. Algunos se centran en el prototipado rápido, el desarrollo incremental y el modelo en espiral [Harel et al, 1998]. En general, dichos procesos de diseño inician con el análisis de requerimientos, durante el cuál la especificación del sistema es construida. Aunque las especificaciones y por tanto los modelos que lo describen son tratados de manera diferente en las propuestas, la fase subsecuente casi siempre es el diseño, el cuál es un prerrequisito esencial para la implementación.

El ciclo de vida del desarrollo de sistemas empotrados plantea el diseño en paralelo del software (SW) y del hardware (HW) correspondiente, en donde se incluye una cantidad considerable de iteración y optimización en siete fases [Berger, 2002]. En este ciclo de desarrollo se muestran los avances en la ingeniería para maximizar la posibilidad de producir un sistema efectivo confiable y al que se le puede dar mantenimiento. Las siete fases son las siguientes:

1. Especificación del producto.

2. Partición HW y SW.

3. Iteración e implementación.

4. Diseño detallado HW y SW.

5. Integración de componentes HW y SW.

6. Prueba y liberación del producto.

7. Mantenimiento y actualización.

Estas fases se muestran en la Figura 1.18. Cada una de las fases se describe a continuación.

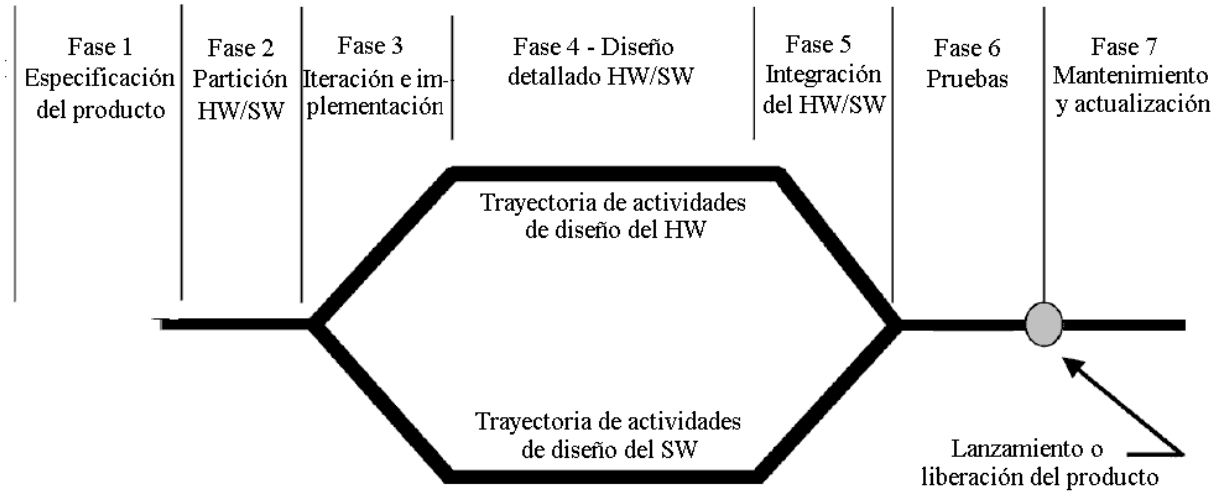

Figura 1.18. Esquema del ciclo de vida del desarrollo de sistemas empotrados.

1.5.1. *Fase 1: Especificación del producto*

En [IEEE STD-830, 1998] se define la *especificación de un sistema* como un *documento* que describe de manera completa, precisa y verificable el *funcionamiento* y los requerimientos de un sistema. Un *requerimiento* es una *descripción* de una condición o propiedad que el sistema debe poseer para resolver un problema del cliente (o usuario) [Marwedel, 2006]. El término especificación de *requerimientos* se refiere a un documento que describe: la *funcionalidad total* del sistema, prestaciones, limitaciones, tipos de interfaz y comunicación, temporización, prioridades, peso, tamaño, instalación, potencia y demás características propias del sistema.

Para definir la funcionalidad del sistema es necesario conocer los distintos conceptos (ideas o vistas) que los usuarios tienen del producto, esto se logra a través del *modelo conceptual* del sistema [Brooks, 2006], que se establece a partir de un *dialogo* (cuestionarios) entre los diseñadores y los clientes [Wood et al, 1989]. Este modelo conceptual es un *mapa* de conceptos que ayuda al diseñador a obtener una descripción de todos (o casi todos) los elementos que serán incluidos en el sistemas y de ser posible las reglas que determinaran su comportamiento [Brooks, 2006].

La obtención del modelo conceptual es una labor bastante complicada, porque aunque parece fácil preguntar al cliente qué es lo que quiere, en realidad, esto no es tan sencillo. Ya que por una parte, el dialogo entre los diseñadores y los clientes se realiza en *lenguajes naturales* como el Español e Ingles, los cuales son ambiguos e incompletos y les falta capacidad para detallar los requisitos de una tarea en particular [Kumar, 1993]. Y por otra parte, frecuentemente ocurre que los clientes no conocen con suficiente precisión el funcionamiento del dispositivo que están solicitando [Rodríguez, 2007] y por lo tanto, proporcionan explicaciones innecesarias sobre detalles que producen confusión entre los diseñadores; porque las descripciones proporcionadas son informales y frecuentemente no están lógicamente conectadas por un conjunto de relaciones precisas. En ambos casos se generan errores de compresión entre los diferentes agentes que intervienen en el proceso de diseño; provocando a su vez, que las técnicas de diseño aplicadas no sean adecuadas [Gajski et al, 1985], y como consecuencia de esto, se producen errores en la funcionalidad del sistema. En la Figura 1.19 se muestra un diagrama de esta situación.

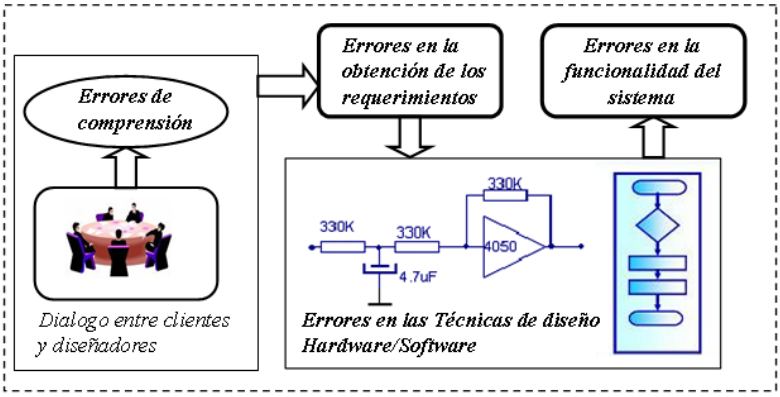

Figura 1.19. Errores en la especificación de un sistema.

Algunos de estos errores o fallas se detectan en etapas intermedias del proceso de diseño, mientras que otros serán detectados hasta que se termine la implementación del dispositivo [Gajski et al, 1994], y se verifique la funcionalidad total del sistema. En cualquier caso, es necesario reconsiderar cambios en las especificaciones [Arichika et al, 2004], lo cual implica cambiar las técnicas de diseño aplicadas. Esto hace que el proceso de diseño del producto sea difícil, engorroso, lento y costoso ya que requiere de diversas iteraciones y modificaciones para obtener el resultado esperado del producto [Brooks, 2006], [Lavagno, 1987]. Desafortunadamente, los errores de funcionalidad son más difíciles de corregir en fases posteriores del proceso de diseño que durante la primera fase [Gajski et al, 1985], por esta razón, los investigadores proponen el uso de especificaciones formales de requerimientos; que sean capaces de describir de forma precisa y detallada la funcionalidad total del sistema y que además, sirvan como un medio de comunicación entre los diferentes agentes que intervienen en el proceso de diseño [Gajski et al, 1985].

Como se puede observar, los requerimientos son componentes decisivos sobre el éxito del proyecto y rectificarlos suele ser una tarea complicada. Quizás por esta razón esta es la fase más complicada de todo el proceso de diseño de un sistema empotrado. Se podría considerar que esta fase es un puente que conduce al diseño y construcción del sistema ya que describe su funcionamiento y sus restricciones. La eficiencia del diseño depende de que tan bien se satisfagan estas restricciones.

Según [Gajski et al, 1994], [Rodríguez, 2007] y [Arichika et al, 2004], la obtención de una especificación formal en el diseño de sistemas empotrados se transforma en un problema de modelación. Un modelo proporciona una vista abstracta del funcionamiento del sistema; de tal manera que éste sea predecible con absoluta precisión ante cualquier secuencia de valores de entrada [Sgroi et al, 2000], [Kocik et al, 2002]; y también, hace posible la simulación del funcionamiento del sistema con el objetivo de detectar y corregir errores o fallas en etapas más tempranas del proceso de diseño y de esta manera optimizar costos ya que reduce el tiempo de diseño [Damm et al, 2008]. Los modelos utilizados en el diseño de sistemas empotrados se conocen como modelos de computación (MoC).

En un MoC el diseño se representa como un conjunto de componentes, los cuales pueden ser considerados como módulos (subsistemas o tareas) aisladas que interactúan entre sí y con su entorno. Un MoC define el comportamiento y el orden de ejecución de estos subsistemas, el tipo de comunicación entre los puertos y el avance del tiempo [Villoria et al, 2002]. Existen varios tipos de MoC, la selección apropiada de uno de ellos depende del dominio de aplicación del sistema; en nuestro caso seleccionamos

el modelo orientado a estado, por tratarse de un sistema empotrado reactivo cuyo comportamiento se puede analizar como una máquina de estado finito.

Un aspecto importante a considerar en el momento de seleccionar el MoC apropiado para una determinada aplicación, es el nivel de abstracción utilizado para especificar, detallar y analizar las diferentes partes que forman al sistema. El nivel de abstracción define la granulidad, es decir, el tamaño de los componentes que se utilizan en el diseño; en nuestro caso, para el diseño del sistema de control de iluminación el nivel de abstracción seleccionado es el *nivel sistema*, por lo tanto, los componentes utilizados son: microcontroladores, transceptores y circuitos integrados lineales, etc.

Todo modelo requiere de una notación especial formada por un conjunto de símbolos que se puedan combinar y que permita al diseñador crear modelos precisos, completos, fáciles de modificar y comprensibles, es decir, un modelo debe ser una ayuda para todos los agentes que intervienen en el proceso de diseño, en lugar de un impedimento para comprender el significado de las partes que forman el sistema; esto implica, la necesidad de utilizar lenguajes formales de especificación. Un lenguaje esta formado por un conjunto de símbolos que se pueden combinar entre sí (sintaxis) y un conjunto de reglas que sirven para interpretar las diferentes combinaciones de estos símbolos (semántica). Cabe mencionar, que existe una amplia variedad de herramientas para la formalización del diseño, pero en muchas de ellas los diseñadores describen el comportamiento del sistema como una relación de entradas y salidas. Esta relación puede ser informal siempre que este expresada en un lenguaje natural [Lavagno, 1987]. Por lo tanto, una especificación es llamada formal si es formulada por medio de un lenguaje de especificación, en caso contrario, la especificación es considerada informal [Sgroi et al, 2000]. En la Figura 1.20 se presenta un diagrama que resume todo lo que se ha discutido hasta este momento.

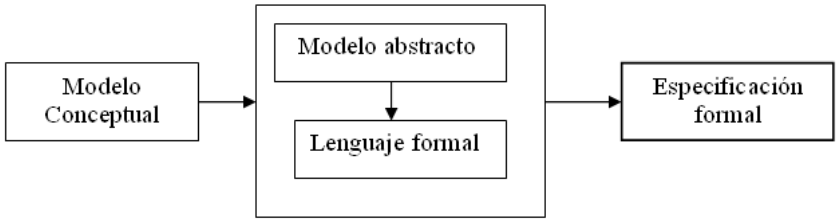

Figura 1.20. Fases del modelo conceptual a la especificación formal.

Existe una amplia variedad de lenguajes formales de especificación, tales como: Verilog-AMS, System Verilog, VHDL-AMS, C++, SysML, SystemC, SpecC, OpenVera, Esterel, Statechat, SpecChart, EDA, Java, Ptolemy II, Metropolis y UML. Algunos de ellos se utilizan para el diseño del software, mientras que otros se utilizan para el diseño del hardware. Sin embargo, en el diseño de sistemas empotrados se requiere de la integración de ambos componentes, lo cual en muchas ocasiones es difícil de lograr, como consecuencia para alcanzar esto los diseñadores utilizan diferentes lenguajes de especificación, cuando lo ideal sería el uso de uno sólo. Analizar las características de todos y cada uno de estos lenguajes esta fuera de los alcances del presente trabajo, sin embargo en [Gajski et al, 1994], [Marwedel, 2006], [Edwards, 2003], [Damm et al, 2008], [Gajski, 1996], [Shin, 1998], [Bunker, 2004] se presentan estudios serios de las características más comunes de estos lenguajes, así como también sus ventajas y desventajas.

La elección del modelo es el aspecto más importante para describir la funcionalidad del sistema sin embargo, en [Damm et al, 2008] se afirma que en la practica la elección del MoC se realiza

implícitamente cuando se selecciona el lenguaje formal de especificación. Entonces se puede asegurar que el éxito del proceso de diseño de un sistema empotrado en gran parte, se basa en la elección del lenguaje de especificación. Ya que este establece una plataforma común y precisa de términos sintácticos y semánticos para la especificación y simulación de las tareas que debe ejecutar el sistema. Por esta razón, omitimos el estudio de los diferentes tipos de MoC, pero en [Gajski et al, 1994], se puede encontrar más información sobre ellos.

De la gran variedad de los lenguajes de especificación que existen actualmente en el mercado, seleccionamos UML (Unified Modeling Language), para la modelación del sistema de control de iluminación. UML es una notación grafica, diseñada para especificar, visualizar, construir y documentar dispositivos de un sistema empotrado [Latella et al, 2000]. Un modelo en UML está compuesto por diferentes tipos de diagramas cada uno representando diferentes aspectos del sistema, por esta razón, este lenguaje ha llegado a ser una base para la especificación a nivel sistema; además permite el refinamiento global de las especificaciones de cada una de las partes que constituyen al sistema, este refinamiento[13] deberá ser comprobado globalmente, es decir, en la estructura total del sistema. Si bien es cierto que UML es un lenguaje de modelación del software, también es cierto que UML ha dado muy buenos resultados en la modelación de sistemas empotrados como lo demuestran los diseños presentados en [Martín, 2002], [Putten, 2001], [Bohn et al, 2002], [Reichmann et al, 2001]. En la sección 1.21 se analizan algunas de las características más importantes de UML, así como algunos de sus diagramas.

Otro aspecto que agudiza la complejidad del proceso de diseño de los sistemas empotrados es que debido a las exigencias de los nuevos diseños, se han transformado de sistemas monofuncionales a sistemas multi-funcionales. Para resolver este problema se utiliza el proceso de jerarquización que consiste en la división del sistema en subsistemas o bloques de forma recursiva para conseguir que el nivel de complejidad de cada parte sea abordable, ya que en la mayoría de los casos, tratar con todo el sistema sería imposible [Solá et al, 2003]. Cada subsistema debe implementar una función o parte de la especificación. En la Figura 1.21 se muestra esta situación.

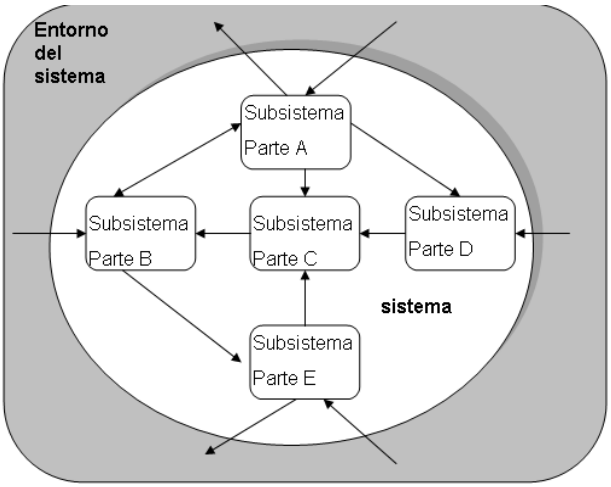

Figura 1.21. Proceso de jerarquización de un sistema.

[13] Procedimiento mediante el cuál se obtiene una descripción más detallada y completa de un sistema a partir de su descripción en un nivel superior.

Para el desarrollo del presente trabajo, siempre se cuido la obtención de las especificaciones del sistema de control de iluminación por métodos formales. Por este motivo, se crearon escenarios del usuario (situaciones en las que el usuario interactúa con el sistema). El análisis de estos escenarios nos permite obtener las entidades, atributos y métodos de diseño en cada una de las partes que forman el sistema. De la jerarquización del sistema se obtienen cuatro subsistemas: Computadora Personal, controlador, receptor y dimmer. En cada uno de estos subsistemas se utilizan varios diagramas en UML para especificar el funcionamiento de cada uno de estos desde diferentes aspectos. La información capturada en los diagramas sirve para desarrollar una especificación completa. Finalmente la información capturada en diagramas de secuencia y casos de uso, definen formalmente un modelo funcional de la totalidad del sistema. En la Figura 1.22 se muestra, los subsistemas que se modelan en UML para el sistema de control de iluminación.

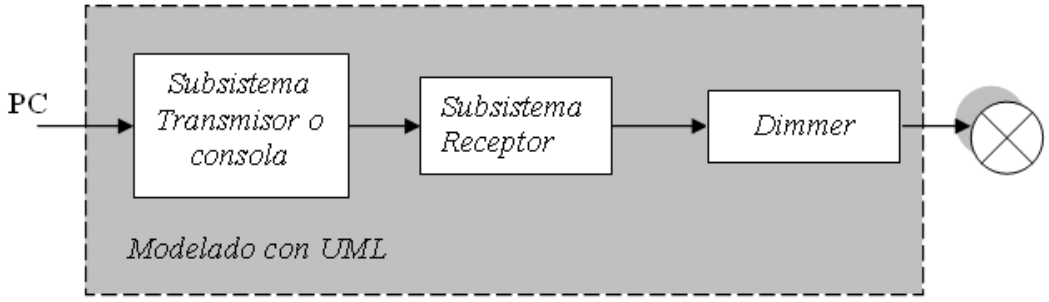

Figura 1.22. Subsistemas modelados en UML para el SciDMX.

1.5.2. *Fase 2: Particionamiento HW y SW*

En [Streichet, 2008], se define el particionamiento de un sistema empotrado como un proceso de *asignación* que consiste en encontrar los componentes más adecuados del HW y SW que permitan satisfacer los requerimientos impuestos en la fase anterior, es decir, en esta fase se debe *asignar* a cada componente una parte de la especificación. Ya que un sistema empotrado no se puede diseñar como un proceso de síntesis, tomando en cuenta únicamente su especificación, sino que también deben considerarse los componentes comerciales disponibles [Marwedel, 2006]. Generalmente los diseñadores tienen para seleccionar innumerables tipos de Circuitos Integrados de Aplicación Especifica (ASIC), FPGA´s, DSP´s, microcontroladores; los cuales varían en costo, desempeño, potencia, tamaño, confiabilidad y esfuerzo de diseño.

Existen dos métodos para particionar un sistema empotrado: *particionamiento estructural y el particionamiento funcional*. El primer método implementa objetos estructurales como finos granos, tales como compuertas, estos objetos son entonces particionados entre diversos componentes comerciales. Este método es fácil de implementar automáticamente, sin embargo, no considera implementaciones de software, ni tampoco retrasos entre los inter-componentes durante la implementación. Mientras que el segundo método, *particionamiento funcional* divide varias funciones del sistema en grupos y asigna a cada grupo un componente del sistema. Cada grupo es implementado como software (para un procesador) o como hardware (para un ASIC). En este caso, los criterios de particionamiento varían para diferentes tipos de asignación, y por lo tanto, los diseñadores evalúan diferentes tipos de asignación para encontrar el diseño más efectivo.

Sin embargo, estas evaluaciones consumen tiempo y esfuerzo, por eso los diseñadores examinan únicamente algunos *diseños potenciales*; frecuentemente aquellos que se pueden evaluar rápidamente debido a su experiencia previa. A través de usar una especificación formal, se puede evaluar rápidamente numerosos diseños potencialmente. Típicamente los componentes asignados incluyen memorias RAM, procesadores estándares, FPGA´s, buses físicos o circuitos ASIC. Es necesario asignar suficientes recursos para implementar todas las especificaciones de desempeño. Al mismo tiempo, la cantidad de recursos asignados no debe exceder el costo, la potencia, el tamaño y otras restricciones. Dependiendo de las cantidades métricas estimadas, el diseñador puede probar diferentes asignaciones, o hacer una buena asignación de manera automática.

Para el diseño de un sistema empotrado, los componentes HW y SW se representan como elementos que interactúan entre sí, formando una *estructura*. Una estructura es una posible representación de una *arquitectura* que contiene su propio conjunto de elementos representando propiedades e información de sus interrelaciones. Una estructura es por tanto una "vista" del HW y SW del sistema considerando el intervalo de tiempo de diseño, un ambiente particular y un conjunto de elementos. Debido a que es muy difícil para una "vista" capturar todas las complejidades de un sistema, una arquitectura típicamente se representa con más de una estructura. Todas las estructuras dentro de una arquitectura están inherentemente relacionadas una con otra, y la suma de todas esas estructuras forma la arquitectura empotrada de un sistema [Noergaard, 2005].

Para poder implementar un sistema, se debe conocer la arquitectura en la que va a ser alojado, por esta razón es necesario determinar las primitivas (componentes) con los que podemos trabajar, y así, por ejemplo, la partición HW/SW será diferente. De tal forma que la meta de la arquitectura es indicar:

- El número de componentes.
- El tipo de componentes.
- El tipo de conexión entre los diferentes componentes.

La arquitectura del software depende directamente del tipo de componentes utilizados en la arquitectura del hardware que soportan la funcionalidad del sistema.

Después de la asignación de componentes en el sistema, los objetos en la especificación serán particionados y *mapeados* en estos componentes. El proceso de asociar un comportamiento funcional a un elemento de la arquitectura que pueda implementar ese comportamiento (CPU, DSP o un HW dedicado) es llamado *mapeo* [Zurawsky, 2005]. Durante el proceso de particionamiento, las variables son mapeadas en las memorias, las funciones (comportamientos) son mapeados en los procesadores estándar comerciales y los canales son mapeados en los buses [Gajski, 1996], Tabla 1.2.

Tabla 1.2. Mapeo de los componentes.

Componentes	**Particionamiento**
Memorias	Variables a memorias
Procesadores	Comportamientos
Buses	Canales a buses.

En el presente trabajo se seleccionó el método de particionamiento funcional. La arquitectura de software engloba todos aquellos módulos cuya función puede ser definida a través de un programa software como pueden ser DSP, microprocesadores y microcontroladores, en nuestro caso seleccionamos al microcontrolador AT90S2313. Todas las operaciones de cálculo son realizadas por estos dispositivos, mientras que las principales *variables* asignadas a los elementos de memoria de los microcontroladores son: *dato DMX*, *dirección DMX y DMX_status;* los canales están formados por las siguientes señales: *D* que representa un bus de 8 bits, señal *DMX* y las señales habilitadoras $Triac_1$ y $Triac_2$. Cada una de las cuales se analizan en los siguientes capítulos. La figura 1.23 muestra el mapeo de la funcionalidad del SciDMX que corresponden a los diagramas de estado en UML.

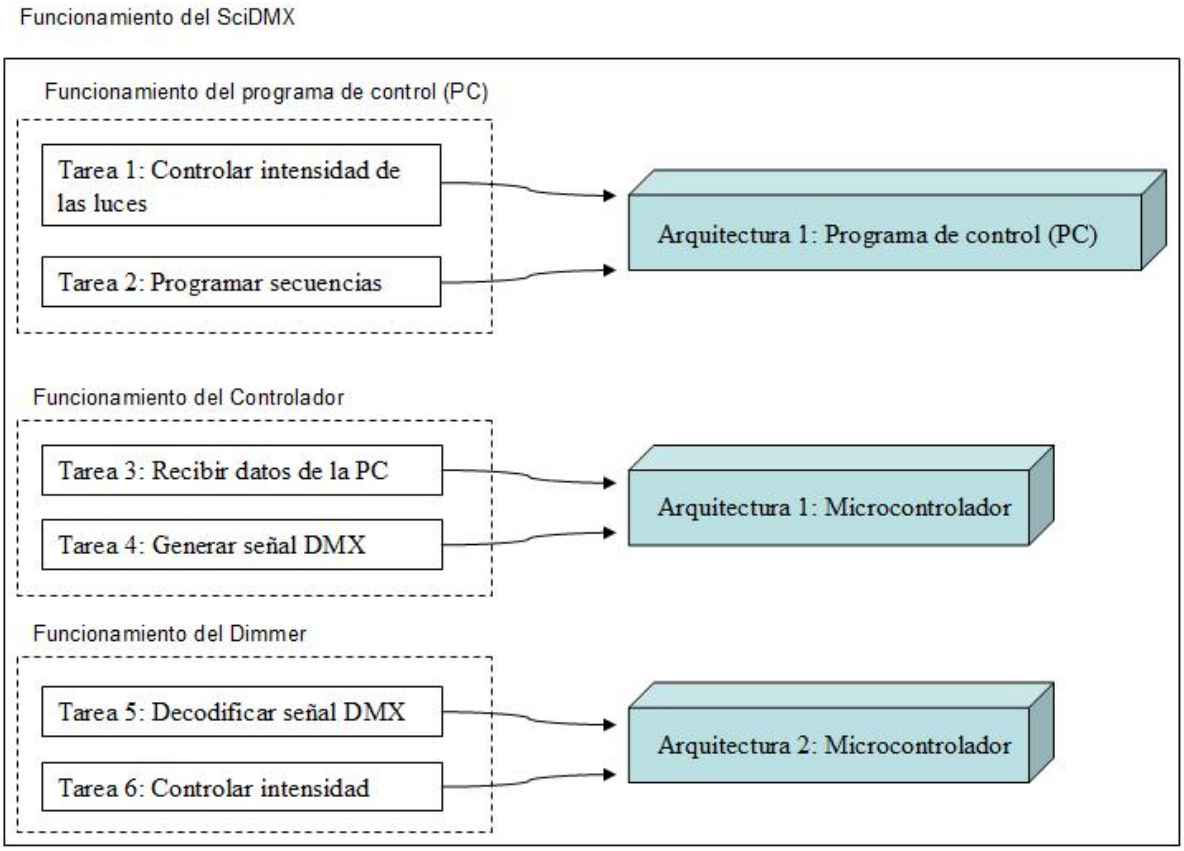

Figura 1.23. Mapeo de las tareas del SciDMX.

1.5.3. *Fase 3: Iteración e implementación*

Debido a la naturaleza dual de los sistemas empotrados, el diseño de las partes del HW y del SW no pueden ser realizadas de manera aislada, sino que estas deben ser realizadas de manera conjunta. Por lo tanto, los diseñadores de HW pueden utilizar herramientas como son los simuladores de arquitecturas llamados comúnmente IDE (Entorno de Desarrollo Integrado) para modelar el desempeño del MCU y de su memoria; mientras que los diseñadores de SW desarrollan código ejecutable que se compila, depura y simula para cumplir los objetivos de las especificaciones. Actualmente los equipos de desarrollo HW y SW trabajan conjuntamente para mantener activo el proceso de iteración.

1.5.4. *Fase 4: Diseño detallado HW y SW*

El objetivo principal de esta fase es obtener un diseño detallado del sistema con base en los requerimientos iniciales. Se debe considerar la interfaz de usuario y la funcionalidad del sistema.

Para un diseño adecuado se consideran los siguientes aspectos:

- Entornos de desarrollo y técnicas especiales de SW.
- Técnicas especiales de programación.
- Diseño digital y arquitectura de MCUs.

1.5.4.1. *Diseño HW*

En la fase de diseño HW se realizan las tareas específicas para el desarrollo del HW. La interfaz HW se define en la especificación del sistema, la cual debe soportar cualquier funcionalidad que el sistema requiera.

1.5.4.2. *Diseño SW*

En la fase de diseño SW se elabora un documento de requerimientos, el cual incluye:

Una declaración de requerimientos: consta de requerimientos, especificaciones de ingeniería, definiciones de HW, etc.

El protocolo de comunicación con otro SW: describe los mecanismos de acceso a otros dispositivos como memorias temporales, órdenes o respuestas de otros dispositivos, etc.

Una descripción de la implementación del sistema: realizada mediante diagramas de flujo, pseudo código, u otros métodos.

1.5.5. *Fase 5: Integración de componentes HW y SW*

En esta fase se debe contar con herramientas y métodos especiales para el manejo de la complejidad.

La clave de éxito en el diseño de sistemas empotrados es combinar el primer prototipo HW, el SW de aplicación, el código del controlador y el SW del sistema operativo.

Los métodos generales de depuración que se aplican en las computadoras personales (PC, *Personal Computer*) o estaciones de trabajo (*workstation*), son muy similares a las que se utilizan en sistemas empotrados, muchos de ellos son imposibles de depurar hasta que se encuentren operando a su máxima velocidad. En general, existen tres requisitos para depurar un sistema empotrado en tiempo real:

- Control de ejecución.
- Sustitución de memoria.
- Análisis en tiempo real.

1.5.6. *Fase 6: Prueba y liberación del producto*

Las pruebas y requisitos de seguridad de un sistema empotrado son más estrictas que la mayoría de las aplicaciones de escritorio. Esta fase tiene un significado especial ya que incluye aspectos de seguridad del sistema. Las pruebas consisten en determinar que el sistema final funcione correctamente.

1.5.7. *Fase 7: Mantenimiento y actualización de productos existentes*

La mayoría de diseñadores de sistemas empotrados más que diseñar nuevos productos, mantienen y actualizan productos existentes. Gran parte de estos ingenieros no son miembros del equipo original de diseño, por lo que tienen que confiar en su experiencia, habilidades, documentación existente y el producto en cuestión, para entender el diseño original y con ello proporcionarle mantenimiento y, si es necesario, actualizarlo.

Esta fase requiere de herramientas adecuadas para la reingeniería. El equipo de soporte tiene acceso a herramientas sofisticadas que le permiten observar la ejecución del código en tiempo real.

1.6. *Fundamentos del Lenguaje Unificado de Modelado*

El Lenguaje Unificado de Modelado es un lenguaje estándar para la especificación, visualización, construcción y documentación de sistemas complejos, que involucran una gran cantidad de software. UML es apropiado para modelar desde sistemas de información en empresas hasta aplicaciones distribuidas basadas en la Web, e incluso para sistemas empotrados de tiempo real muy exigentes. UML es sólo un lenguaje y por tanto es tan sólo una parte de un método de desarrollo de software o en nuestro caso de un sistema empotrado, por tal razón se considera que UML es independiente del proceso de diseño.

UML ha recibido una gran aceptación en ingeniería de software durante los últimos años, y debido a que el diseño de sistemas electrónicos se ha orientado hacia la ingeniería en software, hay un interés emergente por UML dentro de la comunidad del hardware, además de que los diferentes diagramas de UML y sus variaciones han encontrado su aplicación en: especificación de requerimientos, pruebas, descripciones de arquitectura y modelado de comportamiento.

UML contiene un número grande de tipos de diagramas, haciéndole un lenguaje gráfico complejo. De acuerdo con la especificación de UML 2.0 del Object Development Group, los diagramas se clasifican en dos grandes grupos: diagramas estructurales y diagramas de comportamiento, como está reflejado en la Figura 1.24.

Los diagramas estructurales representan elementos y así componen un sistema o una función. Estos diagramas reflejan las relaciones estáticas de una estructura, como lo hacen los diagramas de clases o de paquetes, o arquitecturas en tiempo de ejecución, tales como diagramas de objetos o de estructura de composición.

Los diagramas de comportamiento representan las características de comportamiento de un sistema o proceso de negocios y, a su vez, incluyen a los diagramas de: actividades, casos de uso, de estados, de tiempos, de secuencias, de repaso de interacciones y de comunicaciones [URL 5].

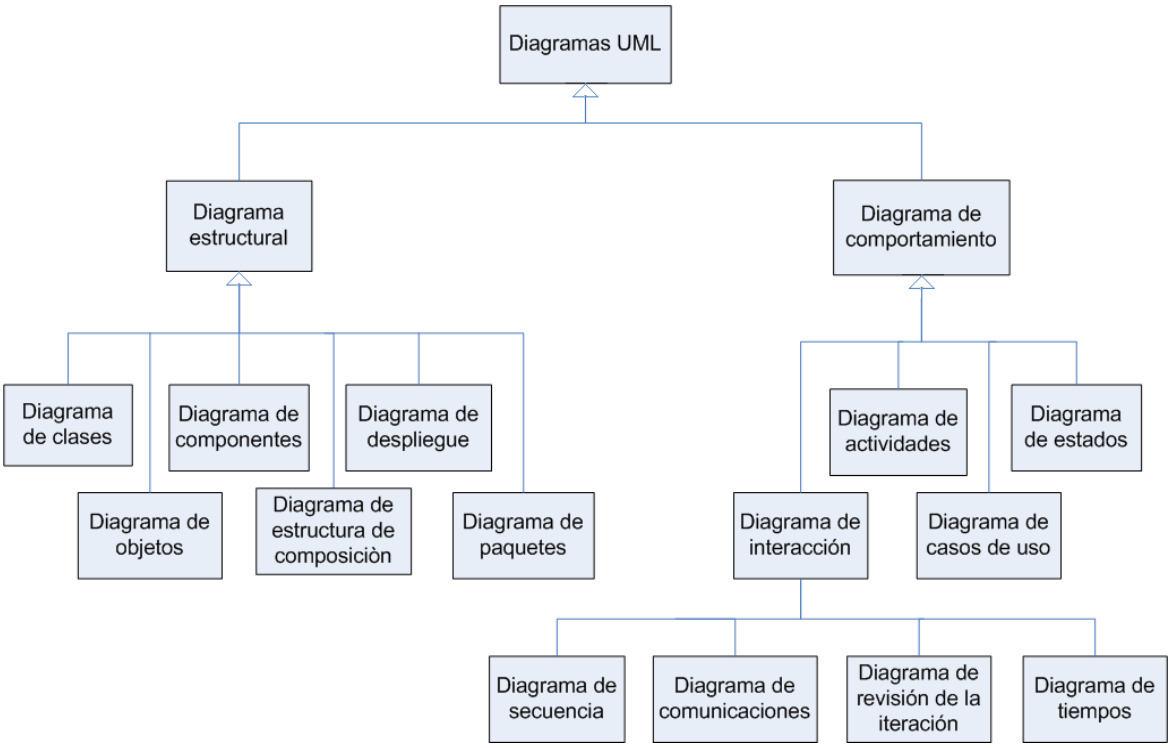

Figura 1.24. Estructura taxonómica de UML 2.0

Debido a la cantidad incremental de software en sistemas empotrados, UML también esta ganando importancia en el desarrollo de los sistemas empotrados. Por lo tanto, han existido varias propuestas para que las extensiones de UML soporten aplicaciones de tiempo real [McLaughlin & Moore, 1998], [Douglas, 2000]. Esas extensiones han sido consideradas durante el diseño de UML 2.0, que incluye 13 tipos de diagramas de 9 en UML 1.4 [Ambler, 2005].

De los diagramas estructurales y de comportamiento, los utilizados en este trabajo son: los diagramas de despliegue, diagramas de casos de uso, diagramas de máquinas de estado y los diagramas de actividad. Estos son descritos a continuación.

1.6.1. *Diagramas de despliegue*

Cuando se construye un sistema con gran cantidad de software, la atención principal del desarrollador se centra en diseñar y desplegar el software. Sin embargo para un ingeniero de sistemas, la atención principal está en el hardware, así como el software del sistema y en el manejo de los compromisos entre ambos. Mientras que los desarrolladores de software trabajan con artefactos en cierto modo intangibles, tales como modelos y código, los desarrolladores de sistemas trabajan a su vez con hardware bastante tangible.

UML se centra principalmente en ofrecer facilidades para visualizar, especificar, construir y documentar artefactos de software, pero también ha sido diseñado para cubrir al hardware. Esto no

equivale a decir que UML sea un lenguaje de descripción de hardware de propósito general, como VHDL[14], más bien UML ha sido diseñado para modelar muchos de los aspectos hardware de un sistema a un nivel suficiente para que un ingeniero de software pueda especificar la plataforma sobre la que se ejecutará el software del sistema, y para que un ingeniero en sistemas o en electrónica pueda manejar la frontera entre el hardware y el software del sistema.

Uno de los usos comunes de los diagramas de despliegue, es el modelar sistemas empotrados, los cuáles contienen frecuentemente una colección de hardware, con una gran cantidad de software que interactúa con el mundo físico, dicho software controla a los dispositivos como son: motores, actuadores, y pantallas; que a su vez están controlados por estímulos externos tales como entradas de sensores, movimientos y cambios de temperatura. Los diagramas de despliegue se pueden utilizar para modelar los dispositivos y los procesadores que comprenden un sistema empotrado. De manera general se puede decir que los diagramas de despliegue describen la "ejecución de la arquitectura" del sistema (nodos hardware o nodos software); es decir modelan la vista de despliegue estática de una arquitectura [Booch et al, 1999], mostrando la configuración de los nodos de procesamiento en tiempo de ejecución y los componentes que residen en ellos.

Gráficamente, un diagrama de despliegue es una colección de nodos y arcos que se pueden organizar agrupándolos en paquetes para especificar relaciones de dependencia y asociación entre ellos. Los elementos diagramáticos que pueden ser usados en un diagrama de despliegue son mostrados en la Figura 1.25. El ícono de principal importancia es el nodo, con el que se pueden representar procesadores, sensores, actuadores, routers, displays, dispositivos de entrada, memorias, PLAs, o algún objeto físico de importancia para el software.

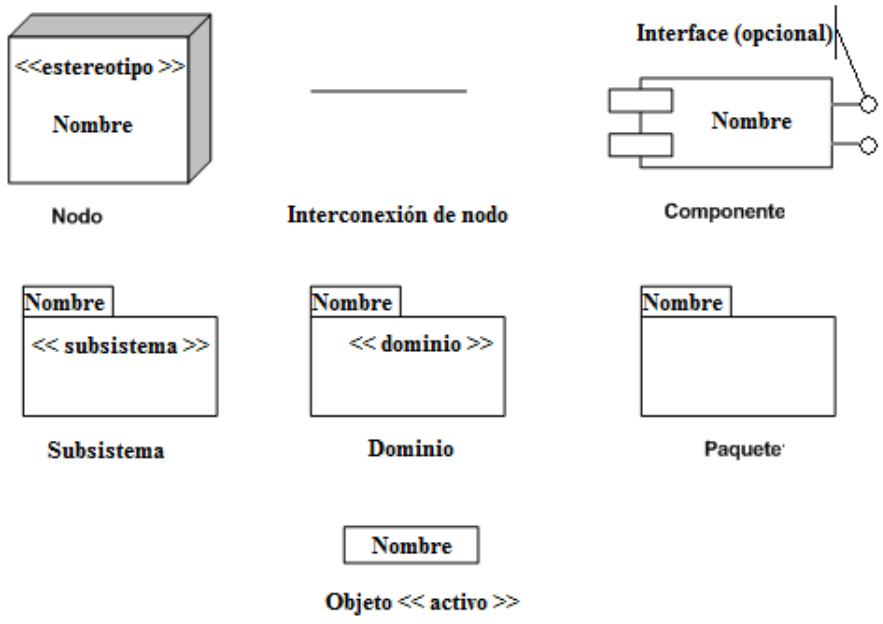

Figura 1.25. Notación de los diagramas de despliegue.

[14] Lenguaje de Descripción de Hardware

El tipo más común de relación entre nodos es la asociación. En este contexto, una asociación representa una conexión física (electrónica, óptica o telemétrica) entre nodos como puede ser una conexión Ethernet, una línea en serie o un bus compartido, como se muestra en la Figura 1.26.

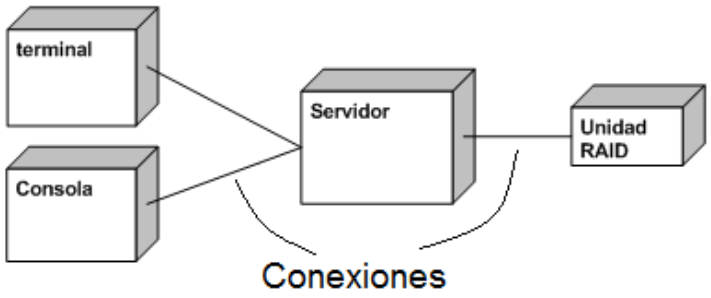

Figura 1.26. Conexiones.

Los *componentes* son parte de la arquitectura física de un sistema. Estos representan los conceptos lógicos del sistema y cómo se ligan entre ellos intrínsicamente. Los *componentes* son parte de la arquitectura física y son un artefacto de desarrollo que existe en tiempo de ejecución. Típicamente estos son ejecutables, librerías, archivos, tablas de configuración, etcétera. Una diferencia clara entre un nodo y un componente, es que los nodos son los elementos donde se ejecutan los componentes y estos últimos representan el empaquetamiento físico de los elementos lógicos [Booch et al, 1999].

Los diagramas de despliegue también pueden contener paquetes o subsistemas, los cuales se utilizan para agrupar elementos del modelo en bloques más grandes. Un *dominio* representa un conjunto de clases organizadas alrededor de un tema o materia en común y vocabulario para el modelo lógico.

Al igual que los demás diagramas, los diagramas de despliegue pueden contener notas y restricciones. Un ejemplo de un diagrama de despliegue se muestra en la Figura 1.27.

En el presente trabajo de tesis, el diagrama de despliegue es utilizado para modelar los dispositivos HW como nodos (transceptores, puerto paralelo, microcontroladores, PC). Los programas de los receptores, el controlador y el programa de control principal que reside en la PC son modelados como componentes.

1.6.2. *Diagrama de casos de uso*

Un diagrama de casos de uso describe las interacciones típicas entre los usuarios (actores) del sistema y el mismo sistema, tal diagrama produce un resultado a uno o más usuarios. Es decir, desde el punto de vista de un observador externo, un diagrama de casos de uso permite modelar *lo que hace un sistema*, la razón de ser de este diagrama se concentra en un *Que hace el sistema*, a diferencia de otros diagramas UML que intentan dar respuesta a *Cómo logra el sistema su comportamiento*. Los casos de uso definen una capacidad a nivel sistema, sin revelar o implicar alguna implementación particular de la capacidad [Douglass, 2000]. Los casos de uso descomponen la funcionalidad principal del sistema y los protocolos necesarios para cumplir esos requerimientos funcionales.

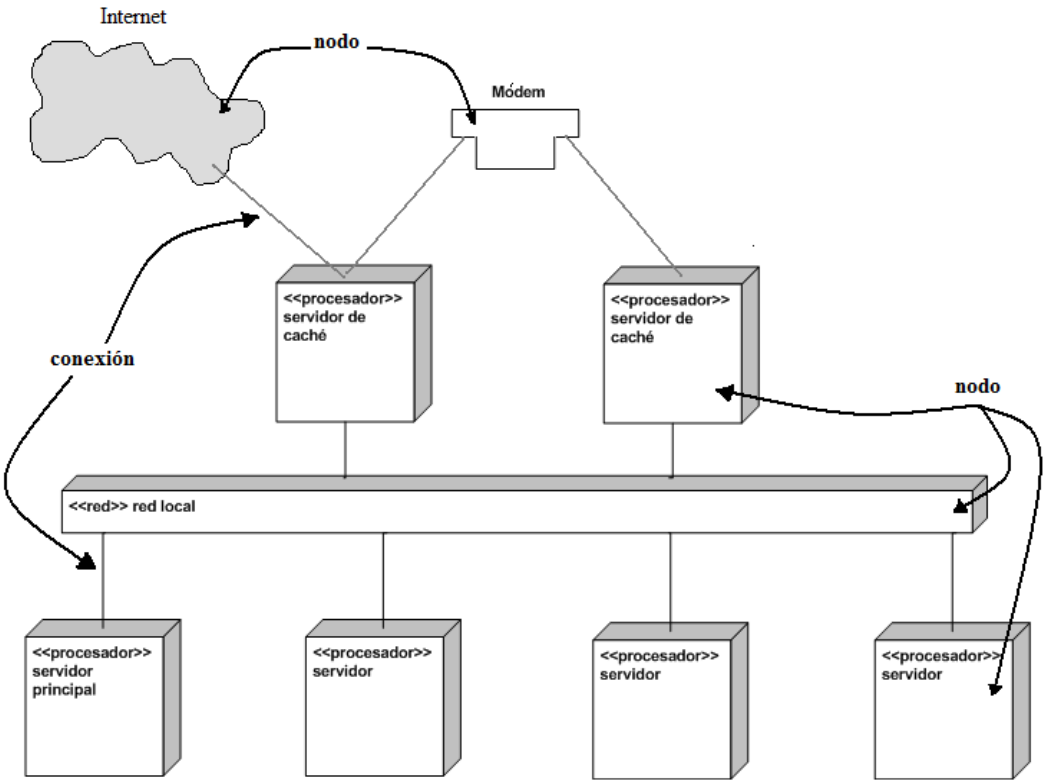

Figura 1.27. Un diagrama de despliegue.

Un elemento diagramático importante en los casos de uso es el *actor*, este último representa un rol que es jugado por una persona, un dispositivo HW o incluso otro sistema al interactuar con el sistema a modelar. Una instancia de un *actor*, por lo tanto representa una interacción individual con el sistema de una forma específica. Aunque se utilizan actores en los modelos, éstos no forman parte de él, son externos a éste [Booch et al, 1999]. Muchas veces, esto se mal interpreta, se cree que los actores deben ser usuarios humanos del sistema. Sin embargo, un actor es algún objeto que interactúa directamente con él.

Los casos de uso se emplean para capturar el comportamiento deseado del sistema en desarrollo, sin tener que especificar cómo se implementa ese comportamiento; está muy relacionado con lo que pudiera ser considerado un escenario en el sistema, esto es, lo que ocurre cuando alguien interactúa con el sistema. Los casos de uso también permiten modelar el contexto de un sistema, delimitándolo y especificando los actores que interactúan con él. Los casos de uso proporcionan un medio para que los desarrolladores, los usuarios finales y los expertos del dominio lleguen a una comprensión común del sistema. Además, los casos de uso ayudan a validar la arquitectura y a verificar el sistema mientras evoluciona a lo largo del desarrollo, es decir los casos de uso y sus escenarios asociados forman el conjunto clave de verificaciones que se aplicarán al sistema [Booch et al, 1999].

Los casos de uso son usados principalmente en el análisis de requerimientos, sin embargo también juega un rol en las demás fases. Una vez que el sistema es analizado en sus subsistemas principales, los casos de uso pueden ser aplicados a cada uno de los subsistemas sucesivamente para definir sus requerimientos con respecto a otros elementos del sistema.

Los requerimientos capturados por los casos de usos y diagramas asociados caen en dos categorías. Requerimientos funcionales que son representados directamente por los casos de uso. El otro tipo de requerimiento es llamado calidad de servicio (QoS, Quality of Service) el cual se encarga de capturar que tan bien los casos de uso deben ser realizados. Los requerimientos de calidad de servicio comunes para sistemas incluyen:

- Velocidad
- Puntualidad
- Rendimiento de procesamiento
- Capacidad
- Previsibilidad
- Confiabilidad
- Protección
- Seguridad

Los requerimientos de calidad de servicio usualmente son capturados como restricciones de algún tipo. Una restricción es una regla aplicada a un conjunto de elementos. Las restricciones representadas en los diagramas de casos, usualmente son expresiones textuales o formales contenidas dentro de llaves como:

{El sistema debe regresar un resultado de revisión de balance cada 30 segundos}.

Para la captura del comportamiento deseado de un sistema, para que el equipo de desarrollo conozca el problema puede entrevistar al cliente y hacerle preguntas como las siguientes [Douglas, 2000]:

¿Cuáles son las funciones principales del sistema?

¿Cuáles son las funciones secundarias del sistema?

¿Por qué debe ser construido el sistema?

¿Qué es lo que remplaza y porqué?

Con las respuestas a estas y otras preguntas que surjan, el equipo de desarrollo debe entonces identificar para cada caso de uso:

El rol que los actores y el sistema juegan en cada escenario, es decir; el comportamiento que cada actor espera o requiere que el sistema le proporcione.

El nombre que identifique a esos comportamientos comunes como casos de uso.

Las interacciones (flujos) necesarias para completar el escenario.

Las secuencias de eventos y datos necesarios para realizar el escenario.

Las posibles variaciones sobre el escenario (otros escenarios relacionados), es decir hay que factorizar el comportamiento común en nuevos casos de uso que puedan ser utilizados por otros; hay que factorizar el comportamiento variante en nuevos casos de uso que extiendan los flujos principales.

Las notas que enuncien los requisitos no funcionales.

Una vez que los casos de uso han capturado los requerimientos, tanto los funcionales y los de calidad de servicio, es necesario proveer de enfoques con mayor nivel de detalle para entender como trabaja cada caso de uso, ya que los nombres de éstos no son suficientes para entender lo que significa cada uno. Para eso, existen dos enfoques que proveen un mayor nivel de detalle: El primero es proveer un conjunto de interacciones ejemplificando el caso de uso; es decir, un conjunto de escenarios. El segundo es definir todas las posibles interacciones en una máquina de estados finita. Estos dos enfoques en UML son de la siguiente manera.

1.6.2.1. *Escenarios*

Un escenario es una interacción sistema-actor particular correspondiente a un caso de uso. Los escenarios modelan la secuencia de mensajes (dependientes del orden) de los objetos que colaboran para producir el comportamiento del sistema. Los diferentes escenarios dentro de un caso de uso muestran permutaciones de interacción de objetos. Inclusive al inicio del diseño, la ventaja de los escenarios es que los expertos en el dominio y los usuarios pueden encaminar al diseñador a través de docenas de escenarios típicos del uso del sistema. Los expertos en el dominio pueden explicar porque es realizado cada paso, porque es iniciado, que respuestas son apropiadas, y que tipo de cosas pueden ir mal. Al ir pasando por este proceso, el diseñador descubre muchas facetas importantes de comportamiento del sistema, que no son mencionadas dentro de la exposición del problema.

Es importante enfatizar que construir y analizar escenarios, es un proceso creativo de descubrir. No es cuestión de iniciar con postulados y aplicar deducción matemática para derivar todas las trayectorias de comportamiento.

En UML existen dos representaciones de escenario para modelar los aspectos dinámicos de un sistema, llamados diagramas de interacción, estos son: el diagrama de secuencia y el diagrama de colaboración. El primero enfatiza los mensajes y su secuencia, comúnmente es el más utilizado. El segundo tiende a enfatizar la estructura de objetos del sistema. Ambos diagramas muestran escenarios pero difieren en lo qué ellos enfatizan.

1.6.2.2. *Diagramas de secuencia*

Los diagramas de secuencia muestran la continuidad de mensajes entre objetos en un momento dado. Su sintaxis gráfica es mostrada mediante el ejemplo de la Figura 1.28. Las *líneas de instancias* representan objetos, con el nombre del objeto arriba o debajo de la línea. Las flechas horizontales son mensajes. Cada línea de mensaje inicia en el *objeto originador,* terminando en el *objeto objetivo*, el nombre del mensaje aparece sobre la misma línea. El eje vertical representa al tiempo, y fluye incrementándose de la parte superior de la página hacia abajo. Los ejes del tiempo muestran solo la secuencia, la escala no es lineal.

Es importante observar las anotaciones textuales del lado izquierdo a lo largo del diagrama. Este texto descriptivo identifica condiciones iniciales y acciones. Los requerimientos de calidad de servicio son indicados por las restricciones dentro de las notas de texto y por las marcas de tiempo entre paréntesis. También pueden añadirse estados al diagrama. Durante el uso, la recepción de algún mensaje puede causar un cambio en un estado. En este momento de la secuencia, el nuevo estado es indicado colocando un ícono sobre la *línea de instancia*. Se asume que el objeto esta en ese estado, hasta que un ícono de estado subsecuente es indicado más tarde sobre la *línea de instancia*.

1.6.2.3. *Diagramas de colaboración*

Un diagrama de colaboración destaca la organización de los objetos que participan en una interacción. Éste se construye colocando en primer lugar los objetos que participan en la colaboración como nodos del grafo, como se muestra en la Figura 1.29. A continuación se representan los enlaces que conectan esos objetos como arcos del grafo. Por último, estos enlaces se etiquetan con los mensajes que envían y reciben los objetos. Esto da al lector del diagrama una señal visual clara del flujo de control en el contexto de la organización estructural de los objetos que colaboran.

Los diagramas de colaboración tienen dos características que los distinguen de los diagramas de secuencia. En primer lugar, el camino, para indicar cómo se enlaza un objeto a otro. En segundo lugar, está el número de secuencia, para indicar la ordenación temporal de un mensaje, se precede de un número, que se incrementa secuencialmente por cada nuevo mensaje en el flujo de control.

En el presente trabajo, son utilizados varios diagramas de casos de uso, primero para modelar los requisitos del sistema a desarrollar, posteriormente son empleados para identificar la funcionalidad de cada uno de los módulos que forman al sistema completo. Finalmente, un diagrama de casos de uso modela la interacción entre el operador del sistema con el programa de control principal ejecutado en la PC.

1.6.3. *Diagrama de estados*

Un diagrama de estados también llamado comúnmente máquina de estados, es un comportamiento que especifica las secuencias de estados por las que pasa un objeto durante su vida en respuesta a eventos, junto con sus respuestas a éstos.

Las máquinas de estados se utilizan para modelar los aspectos dinámicos de un sistema. La mayoría de veces, esto implica modelar un caso de uso o un sistema completo. Estos últimos pueden responder a eventos tales como señales, operaciones o al paso del tiempo. Al ocurrir un evento, tendrá lugar cierta actividad, según el estado actual del objeto. Una actividad es una ejecución no atómica en curso, dentro de una máquina de estados. Las actividades acaban por producir alguna acción, la cual se compone de computaciones ejecutables atómicas que producen un cambio en el estado del modelo o devuelven un valor. El estado de un objeto es una condición o situación en la vida de un objeto durante la cual satisface alguna condición, realiza alguna actividad o espera algún evento.

Las máquinas de estados pueden visualizarse de dos formas utilizando:

- Diagramas de actividades, destacando el flujo de control entre actividades.
- Diagramas de estados, destacando los estados potenciales de los objetos y las transiciones entre esos estados, lo que es especialmente útil cuando se modelan sistemas reactivos.
- Las máquinas de estados bien estructuradas son como los algoritmos bien estructurados: eficientes, sencillas, adaptables y fáciles de comprender.

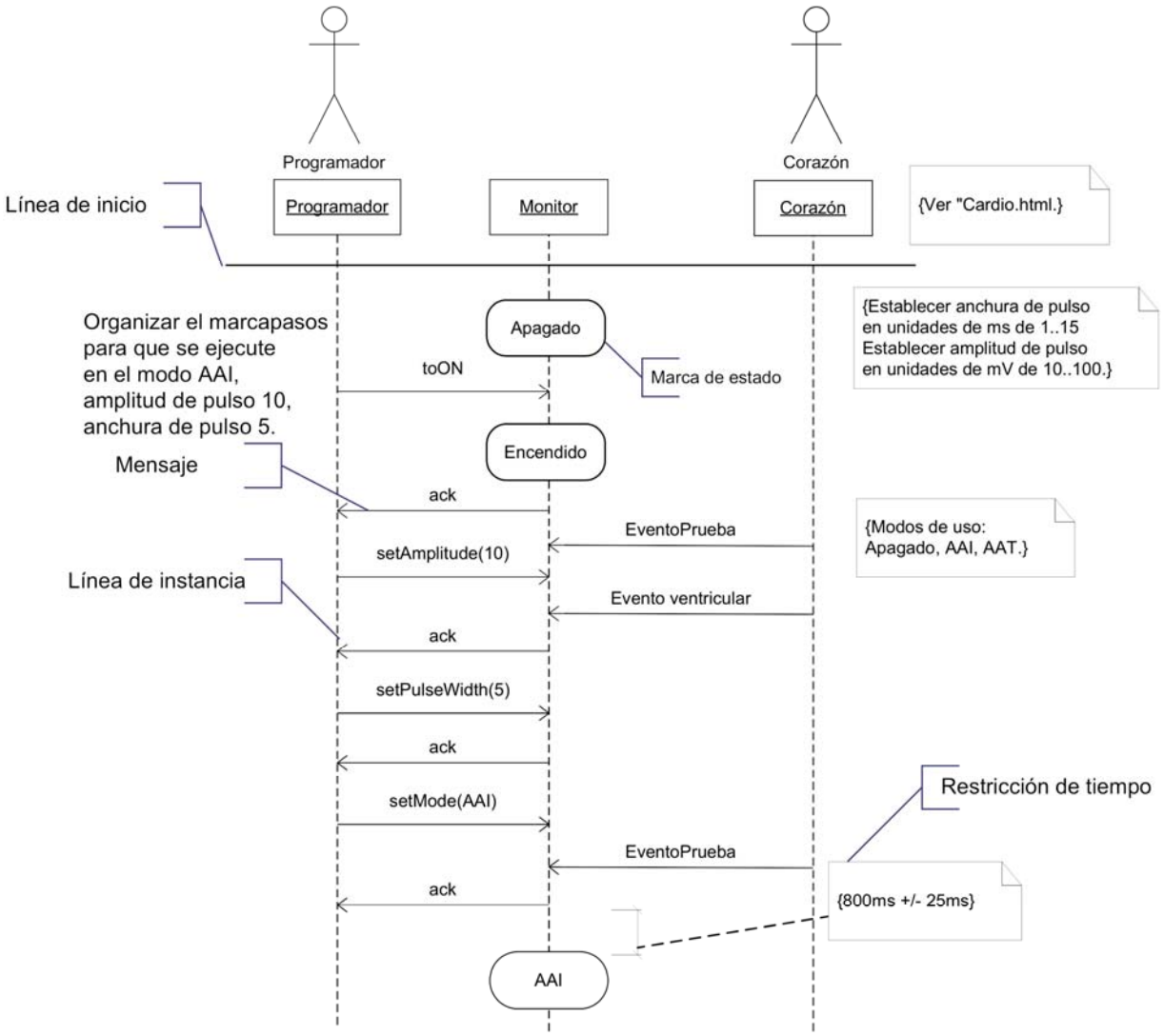

Figura 1.28. Sintaxis del diagrama de secuencia.

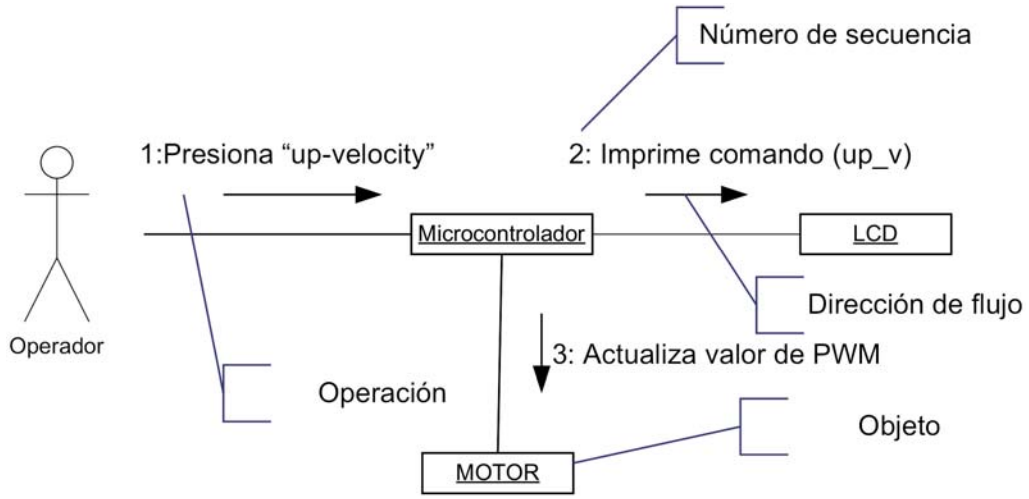

Figura 1.29. Diagrama de colaboración.

Los estados son mostrados como rectángulos redondeados. Las transiciones son líneas dirigidas empezando del estado inicial y terminando en el estado objetivo. Las transiciones usualmente tienen disparadores de eventos, opcionalmente seguidos por acciones (sentencias ejecutables o acciones) que son ejecutadas cuando la transición es tomada. En la Figura 1.30 se muestra un ejemplo de un diagrama de estados.

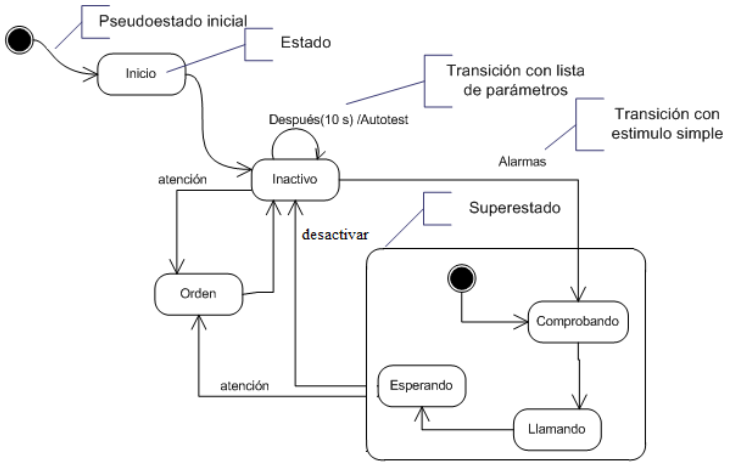

Figura 1.30. Diagrama de estados.

En el presente trabajo, las máquinas de estados son empleadas para describir el comportamiento de los casos de uso de cada uno de los módulos que forman al sistema.

1.6.4. *Diagrama de actividades*

El diagrama de actividad es un diagrama de flujo de procesos multi-propósito que se usan para modelar el comportamiento del sistema. Los diagramas de actividad se pueden usar para modelar un caso de uso, o una clase, o un método complicado, mostrando el flujo de actividades. Una actividad es una ejecución no atómica en curso, dentro de una máquina de estados. Las actividades producen finalmente una acción, que está compuesta de computaciones atómicas ejecutables que producen un cambio en el estado del sistema o la devolución de un valor. Las acciones incluyen llamadas a otras operaciones, envío de señales, cálculos como la evaluación de una expresión. Gráficamente, un diagrama de actividades es una colección de nodos y arcos. Un diagrama de actividad es parecido a un diagrama de flujo; la diferencia clave es que los diagramas de actividad pueden mostrar procesado paralelo (parallel processing) incluso para modelar programación concurrente.

Los diagrama de actividades es un caso especial de un diagrama de estados en el cual casi todos los estados son estados de acción (identifican que acción se ejecuta al estar en él) y casi todas las transiciones son enviadas al terminar la acción ejecutada en el estado anterior. Puede dar detalle a un caso de uso, un objeto o un mensaje en un objeto. Sirven para representar transiciones internas, sin hacer mucho énfasis en transiciones o eventos externos. Generalmente modelan los pasos de un algoritmo.

En el presente trabajo, los diagramas de actividad son empleados para describir el procedimiento secuencial de cada uno de los estados de las máquinas de estados empleadas para describir el comportamiento de SciDMX.

Capítulo 2. Desarrollo del SciDMX y el controlador DMX

El presente capítulo inicia con la primera fase del desarrollo del Sistema de Control de Iluminación DMX (SciDMX), posteriormente se describe como se divide el desarrollo del SciDMX en módulos. Por último, el capítulo finaliza con el desarrollo del controlador DMX.

Con base en la metodología de desarrollo de sistemas empotrados presentada en el capitulo anterior, se presenta el diseño del Sistema de Control de Iluminación DMX, SciDMX.

2.1. *Fase 1: Especificación general del SciDMX*

2.1.1. *Requerimientos funcionales del sistema de control de iluminación*

Antes de presentar las especificaciones del sistema de control de iluminación, a continuación se presentan los requerimientos (necesidades) del usuario. Las cuales permitieron extraer las habilidades que el sistema debe contener y formular las especificaciones. Algunas de las preguntas más sobresalientes que se le hicieron al usuario son las siguientes:

1. ¿Cuáles son las funciones principales que el sistema de control de iluminación debe realizar?

Las funciones principales que el sistema debe realizar son dos. La primera es permitir que un operador en tiempo de ejecución, pueda controlar la iluminación de un escenario, estudio, teatro, etc. de manera remota por medio de una consola de control, con la finalidad de proporcionar iluminación agradable a la escenografía. La segunda función llamada Fade, consiste en que el operador mediante la consola ordene el encendido o apagado progresivo de todas las luces en un intervalo de tiempo seleccionado por él.

2. ¿Cuáles son las funciones secundarias del sistema?

Permitir que un operador pueda "programar" secuencias de encendido/apagado de las luces, y posteriormente "reproducir" dicha secuencia.

3. ¿Por qué debe ser construido el sistema?

En realidad, en el mercado ya existen sistemas de control de iluminación, sin embargo su costo e importación aún son altos, muchos grupos musicales que inician en este género se ven obligados a prescindir de este equipo, por lo que se conforman con controlar exclusivamente el encendido/apagado de las luces directamente. Creando de esta forma un alto riesgo para el operador, ya que las corrientes de consumo de las luces que se manejan son relativamente altas y peligrosas. Por esta razón, se propone la construcción de un sistema de control de iluminación económico que sea compatible con equipos profesionales DMX512. Con dicho sistema se podrá encender/apagar las luces, controlar la intensidad de cada una de ellas, incluso encenderlas progresivamente (Fade in) o apagarlas gradualmente (Fade out).

Otra razón por la que debe ser construido el sistema, es que con este tipo de sistemas de control de iluminación, las conexiones son fáciles, prácticas y rápidas de realizar. Además de que el equipo es fácil de transportar y de operar.

4. ¿Las luces que se utilizan para la iluminación de que tipo son?

Las luces utilizadas son del tipo Par 64. Las cuales constan de un reflector y una bombilla de halógeno de 500 Watts, máximo 1000 Watts.

5. ¿Las luces son móviles o fijas?

Las luces del tipo Par 64 son fijas, sin embargo en el mercado existen luces inteligentes DMX512 que pueden ser empleadas en un futuro por los grupos musicales pequeños.

6. ¿Entonces, qué función debe realizarse sobre este tipo de luces?

El sistema debe darle al operador el control para que pueda regular la intensidad luminosa de las luces y que también pueda encenderlas y apagarlas.

7. ¿Las funciones de control sobre las luces es siempre la misma, o es diferente cada vez?

Cada vez es diferente, por lo que el usuario puede en el momento del evento controlar las luces a su conveniencia. Por lo regular el operador va controlando la intensidad de las luces, en momentos las enciende al máximo, apaga algunas, enciende otras o incluso realiza el encendido progresivo (Fade in) de las luces, u otras veces el apagado progresivo (Fader out).

9. ¿Cuántas luces utilizan por lo regular?

Varía, pero para pequeños eventos con al menos 12 luces de este tipo (Par 64), el escenario se ilumina lo suficiente para realizar el evento.

10. ¿Cómo describes un sistema DMX?

Un sistema DMX comercial consiste básicamente en una consola de control, cable DMX, y de 1 hasta 512 accesorios (receptores) DMX.

Las consolas de control, básicamente son de dos tipos: hardware o software; el primer tipo contiene físicamente controles como: botones, perillas, pantallas de cristal líquido y se conecta directamente con los receptores. Las de tipo software son un programa de computadora que despliega en pantalla los controles anteriores de manera virtual, además de presentar mayores prestaciones y funciones como el simular y programar con anticipación secuencias sin tener todo el equipo DMX conectado a la computadora. La funcionalidad de este tipo de consolas puede mejorar en cada nuevo lanzamiento del programa de control. Si la consola es del tipo software, muchas veces utilizan una tarjeta electrónica que es insertada en la PC o un dispositivo externo llamado controlador que se conecta entre la PC y los receptores. La consola de control regularmente se encuentra enfrente del escenario y atrás de los espectadores, a una distancia aproximada de 60 a 100 metros.

Los receptores también llamados accesorios DMX, pueden ser dimmers, luces robóticas, luces tipo Par 64, etc. Cada receptor tiene una entrada y una salida DMX, para poder formar una cadena tipo Daisy, es decir; la salida de la consola de control debe estar conectado con un cable al primer dimmer, la salida de éste debe estar conectado a la entrada del segundo y así hasta un máximo de 512 dimmers, la salida del último receptor debe estar "terminado" con un conector-terminador DMX. Cada receptor cuenta con un dip-switch de 8 interruptores como selector de canal, con el cuál se establece la dirección de cada receptor. Comercialmente cada dimmer es de 4 canales y 4 salidas para 4 luces. Las luces son de 600 a 1000 Watts. La ubicación de los dimmers es la misma que el de las luces, es decir arriba del escenario.

2.1.2. *Especificaciones del SciDMX*

Con base en la definición de los requerimientos, mediante la respuesta a preguntas realizadas al usuario y la consulta al estándar DMX512 [E1], ahora se tiene una idea general del sistema a desarrollar:

El sistema de control de iluminación SciDMX, tiene como principal objetivo: permitir el control de iluminación de escenarios a distancia vía una consola de control, de manera que se pueda regular la intensidad de las lámparas, así como la programación y reproducción de secuencias de encendido-apagado de luces. Para definir la funcionalidad del sistema de control de iluminación, se utiliza un diagrama de casos de uso mostrado en la Figura 2.1. Como casos de uso principales se han identificado cuatro, estos son: Controlar intensidad de luces, Emplear funciones de fade (atenuar/aumentar intensidad en un lapso de tiempo asignado), Programar secuencias de encendido-apagado de luces y Reproducir secuencias de encendido-apagado de luces. Como único actor se ha identificado al Operador. Las restricciones a considerar aparecen del lado derecho de la Figura 2.1.

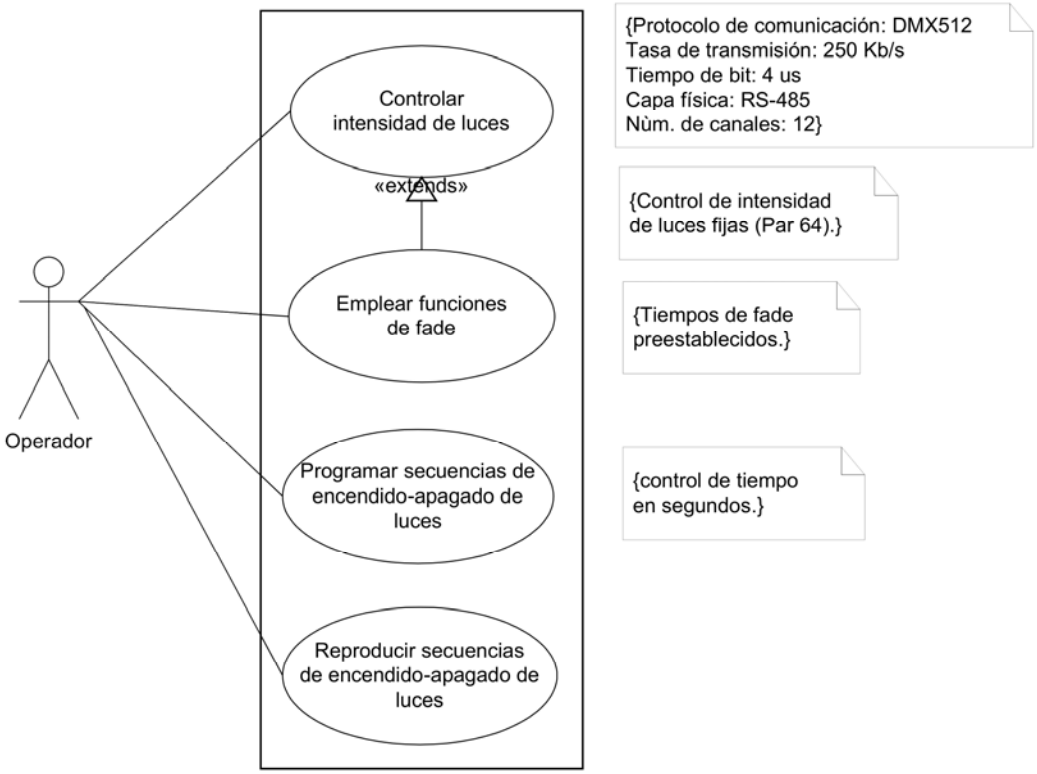

Figura 2.1. Casos de uso del sistema SciDMX.

A continuación se explican los dos casos de uso principales con ayuda de dos diagramas de secuencia.

Caso de uso: Controlar intensidad de luces

Este caso de uso permite al operador controlar de forma manual la iluminación de un escenario por medio de una consola de control.

En la Figura 2.2 se presenta el diagrama de secuencia para este caso de uso. Los objetos involucrados son: el usuario (operador), la consola, dos dimmers (para efectos de demostración tienen asignados los canales 2 y 12) y dos luces conectadas, una a cada dimmer. El estado de la consola, inicialmente es Apagado, y el estado de los dos dimmers es Inactivo.

La interpretación del diagrama de secuencia es la siguiente: El operador enciende la consola, provocando que el estado de la consola cambie a *Encendido*, lo que logra que empiece a generar repetidamente la señal DMX (con valores de intensidad a cero, luces apagadas) hacia los dimmers conectados, y que éstos cambien al estado Activo. Posteriormente el operador cambia en la consola la intensidad del canal 2 a la máxima intensidad (valor enviado 255), de esta manera la consola genera nuevamente la señal DMX con la que establece que la luz conectada al dimmer del canal 2 encienda al máximo mientras que la luz del dimmer del canal 12 sigue apagada, estos cambios son observados en tiempo de ejecución por el operador y/o espectadores. Un tiempo después, el operador cambia en la consola la intensidad del canal 12, a un nivel medio (valor enviado 128). La consola genera nuevamente la señal DMX, con la que establece que la luz conectada al dimmer del canal 12 cambie a una intensidad intermedia. Mientras que la luz conectada al dimmer del canal 12 mantiene una intensidad máxima. De esta manera, los cambios que el operador realice consecuentemente se realizarán de forma parecida.

Caso de uso: Emplear funciones de fade

Este caso de uso permite al operador encender (Fade In) o apagar (Fade Out) progresivamente las luces en un intervalo de tiempo preseleccionado. Es decir, para el caso de Fade In, el operador tiene sobre la consola un control (perilla o teclado) con el que establece el intervalo de tiempo que tardarán las luces en ir encendiendo progresivamente, hasta llegar a la máxima intensidad. Para el caso Fade Out, la funcionalidad es similar, con la diferencia de que las luces van disminuyendo su intensidad luminosa hasta apagarse en el intervalo seleccionado.

En la Figura 2.3 se presenta el diagrama de secuencia para este caso de uso. Nuevamente los objetos que interactúan son: el usuario (operador), la consola, dos dimmers (canales 2 y 12) y dos luces, una conectada a cada dimmer. Inicialmente, la consola está encendida y el estado de los dos dimmers es Activo.

La interpretación del diagrama de secuencia es la siguiente: El operador inicia el procedimiento estableciendo el tiempo de Fade (perilla o teclado) a un valor de *t* segundos, posteriormente presiona el botón Fade In, lo que ocasiona que la consola empiece a generar la señal DMX (inicialmente con valores de intensidad a cero, luces apagadas) hacia los dimmers conectados. Éstos en respuesta establecen el nivel de intensidad (x1) a cero en la luz correspondiente. De manera iterativa, la consola incrementa el valor de intensidad x1 para todos los canales conectados, y genera la señal DMX (contiene el nuevo valor de x1) hacia los dimmers. Nuevamente éstos en respuesta establecen el nivel de intensidad actual x1 en la luz correspondiente. Durante los *t* segundos, la iteración se lleva a cabo de la misma manera hasta que el nivel de intensidad x1 llega al máximo valor (255). Finalizando con esto la función Fade.

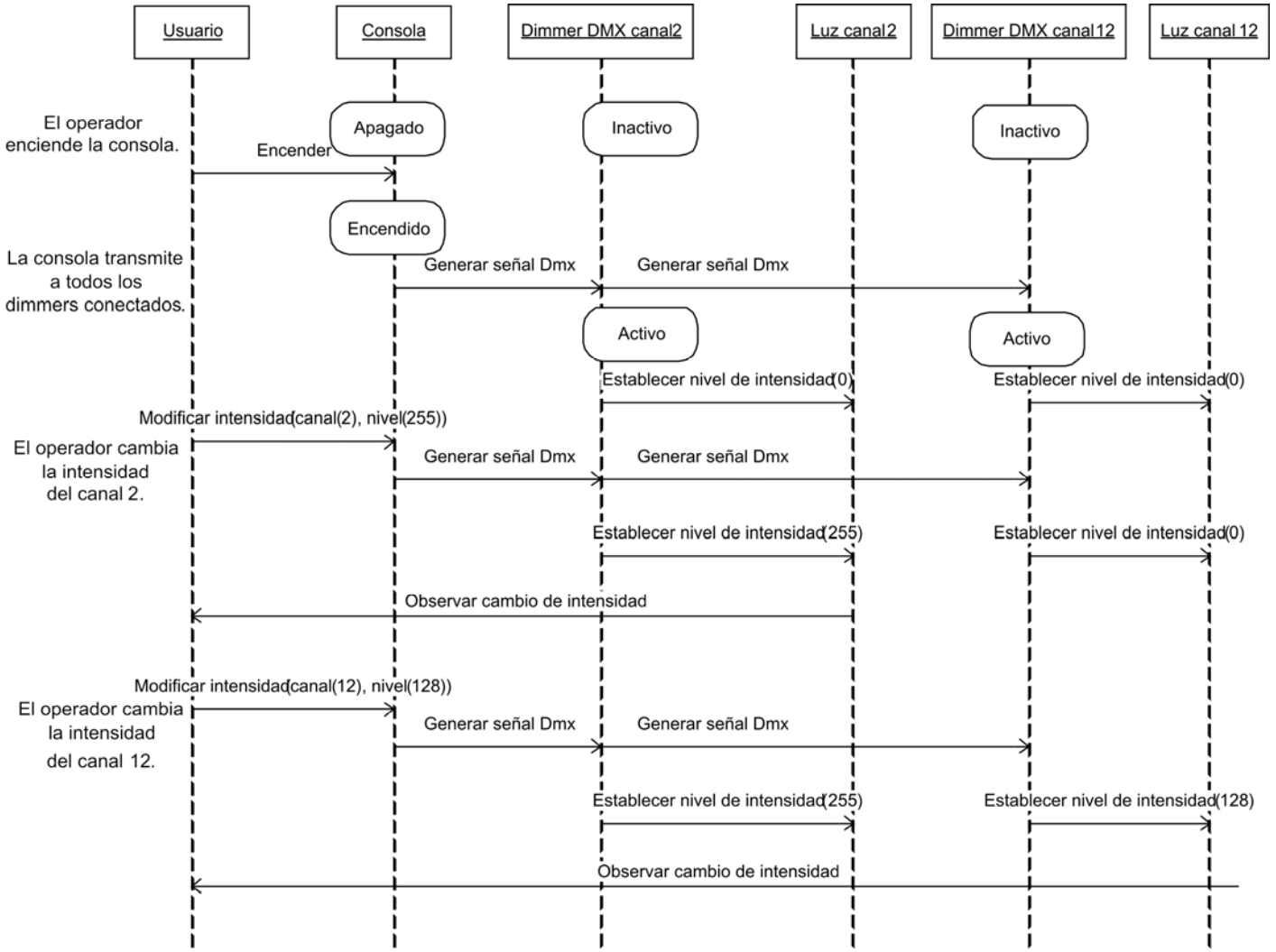

Figura 2.2 Diagrama de secuencia: Controlar intensidad de luces.

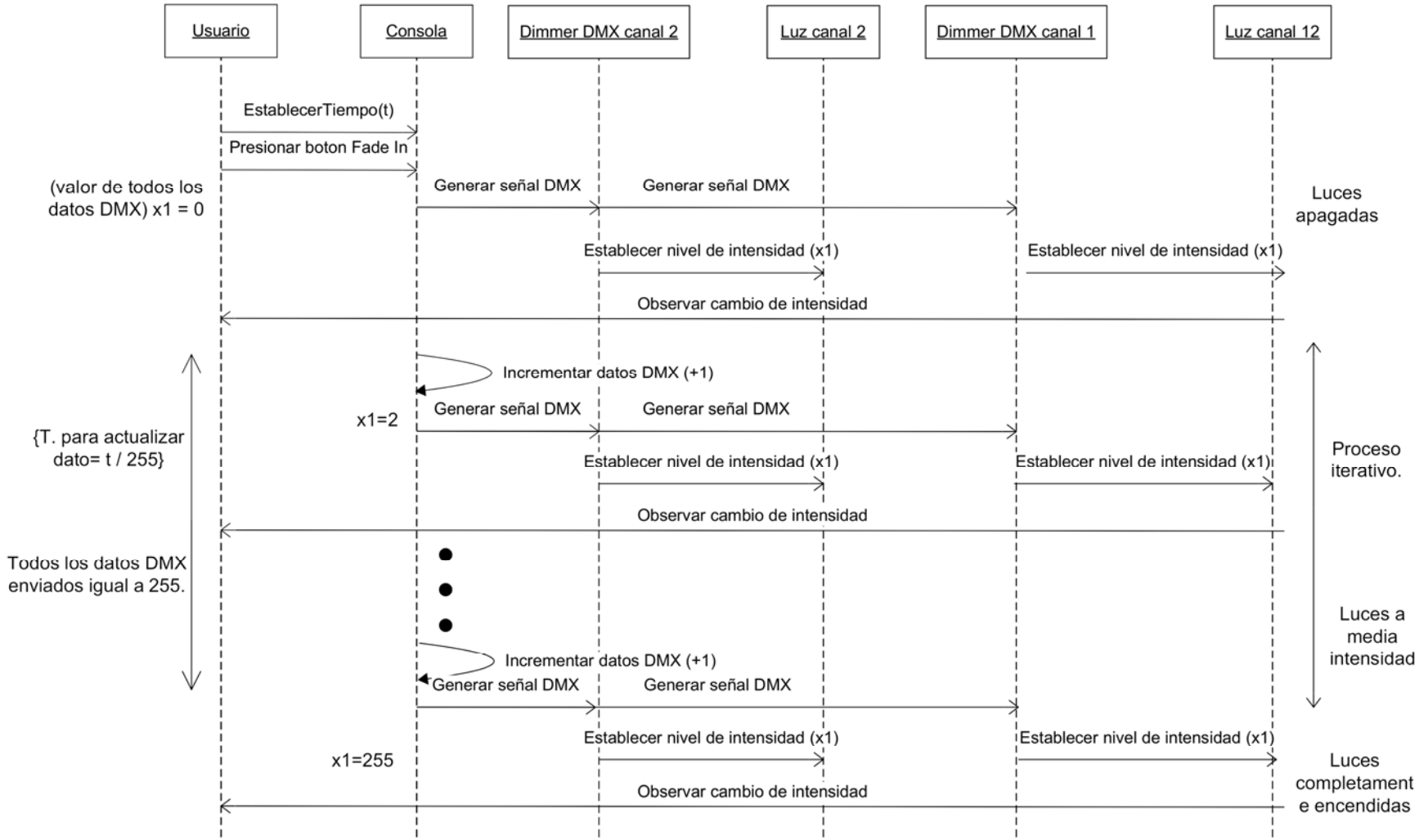

Figura 2.3 Diagrama de secuencia Emplear funciones de Fade In.

2.2. *División del SciDMX en subsistemas*

Una vez que se han presentado las especificaciones del SciDMX a desarrollar, se inicia el particionamiento HW y SW.

Después del análisis de las especificaciones y de una revisión exhaustiva del protocolo DMX512, se observó que la propia naturaleza de un sistema DMX hace que el diseño y construcción esté formado por módulos o subsistemas como lo muestra el diagrama a bloques de la Figura 2.4.

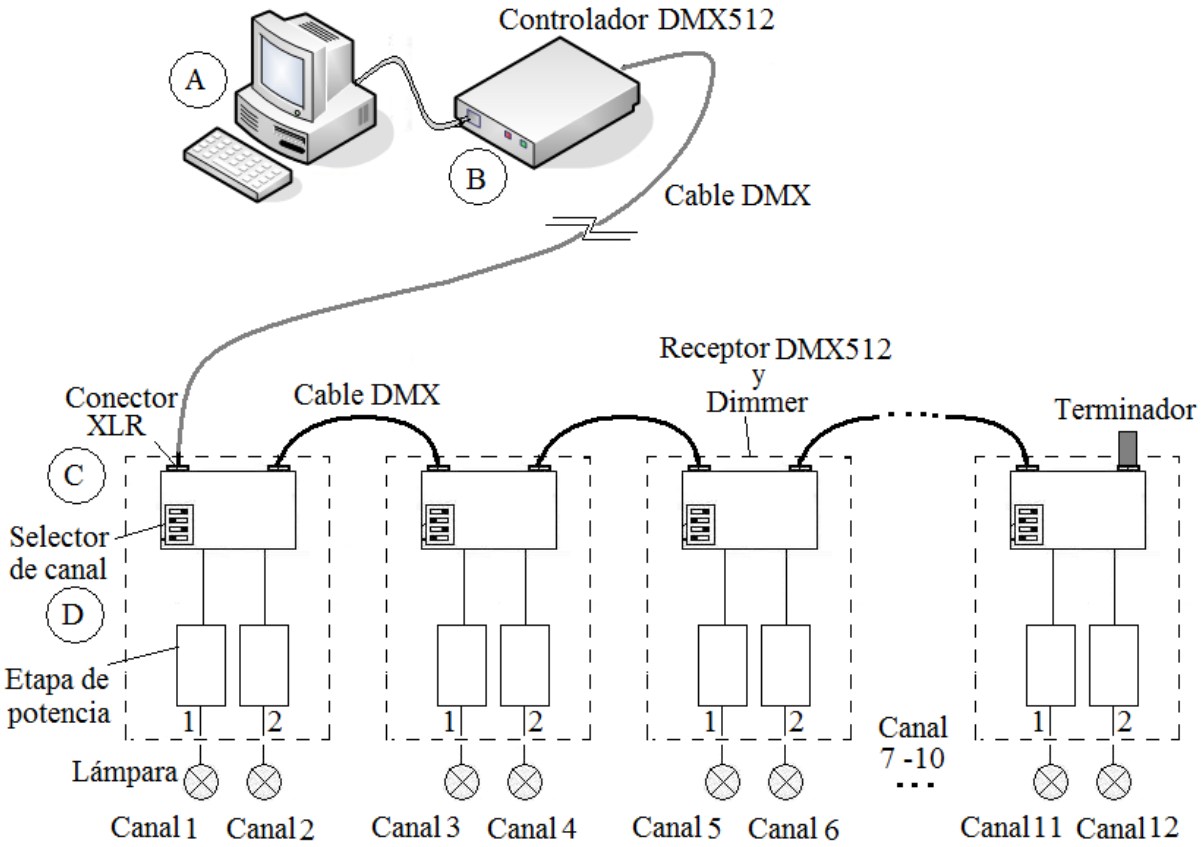

Figura 2.4. Diagrama a bloques de un sistema DMX512.

Los principales módulos de un sistema DMX son los siguientes:

- Una consola de control, representada por la PC y el controlador DMX, partes A y B, respectivamente.
- Cable DMX
- Un o hasta 512 dimmers, partes C y D.
- Una o hasta 512 luces (lámparas) fijas.

Su descripción general es la siguiente:

Una consola de control implementada en un programa de computadora, con una interfaz gráfica de usuario como se muestra en la Figura 2.5, la cual contiene principalmente barras de desplazamiento

(llamados *faders*) y botones como medios de control. El primero de ellos permiten al operador regular individualmente la intensidad de las luces, y el segundo sirve para realizar funciones de encendido (fade-in) y apagado (fade out) progresivo de las mismas. El programa de computadora y el controlador DMX se comunican a través del puerto paralelo (PP).

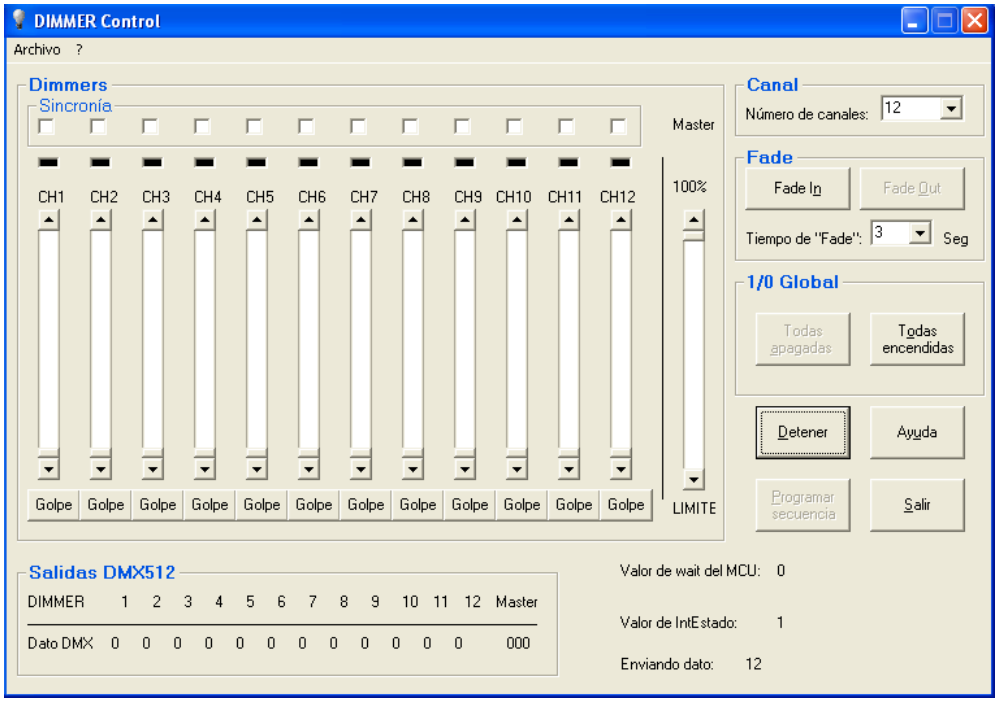

Figura 2.5. Interfaz del programa de control.

Un controlador (maestro) recibe de la PC los datos de intensidad deseados, posteriormente multiplexa dichos datos, en base al estándar DMX512, generando así la señal DMX512 que es transmitida serialmente [Yescas et al, 2006] a los dimmers.

El siguiente módulo a diseñar, es el dimmer (esclavo) de dos canales. Las dos funciones del dimmer son demultiplexar la señal DMX512 enviada por el controlador, obteniendo los dos 'datos DMX' que le corresponden, y la segunda función es que para cada canal, debe controlar la intensidad en la lámpara, en función de los valores de los 'datos DMX' recibidos. Para efectos de demostración, seis dimmers son conectados (doce canales en total) en una cadena tipo Daisy, con sus dos respectivas lámparas. El controlador y los dimmers se comunican a través del cable DMX. Los conectores utilizados son de tipo XLR de 3 pines y la impedancia del terminador es de 120 Ω.

La Figura 2.6 muestra el diagrama de despliegue del sistema de control de iluminación a desarrollar, en este se pueden observar los siguientes nodos: PC, Controlador, Bus DMX, y varios nodos dimmer. Todos a excepción del nodo bus DMX tienen componentes que se ejecutan dentro de ellos, etiquetados con el estereotipo[15] *executable*.

[15] Estereotipos son mecanismos de extensibilidad en UML. Permiten a los diseñadores en vocabulario de UML para crear nuevos elementos de modelo.

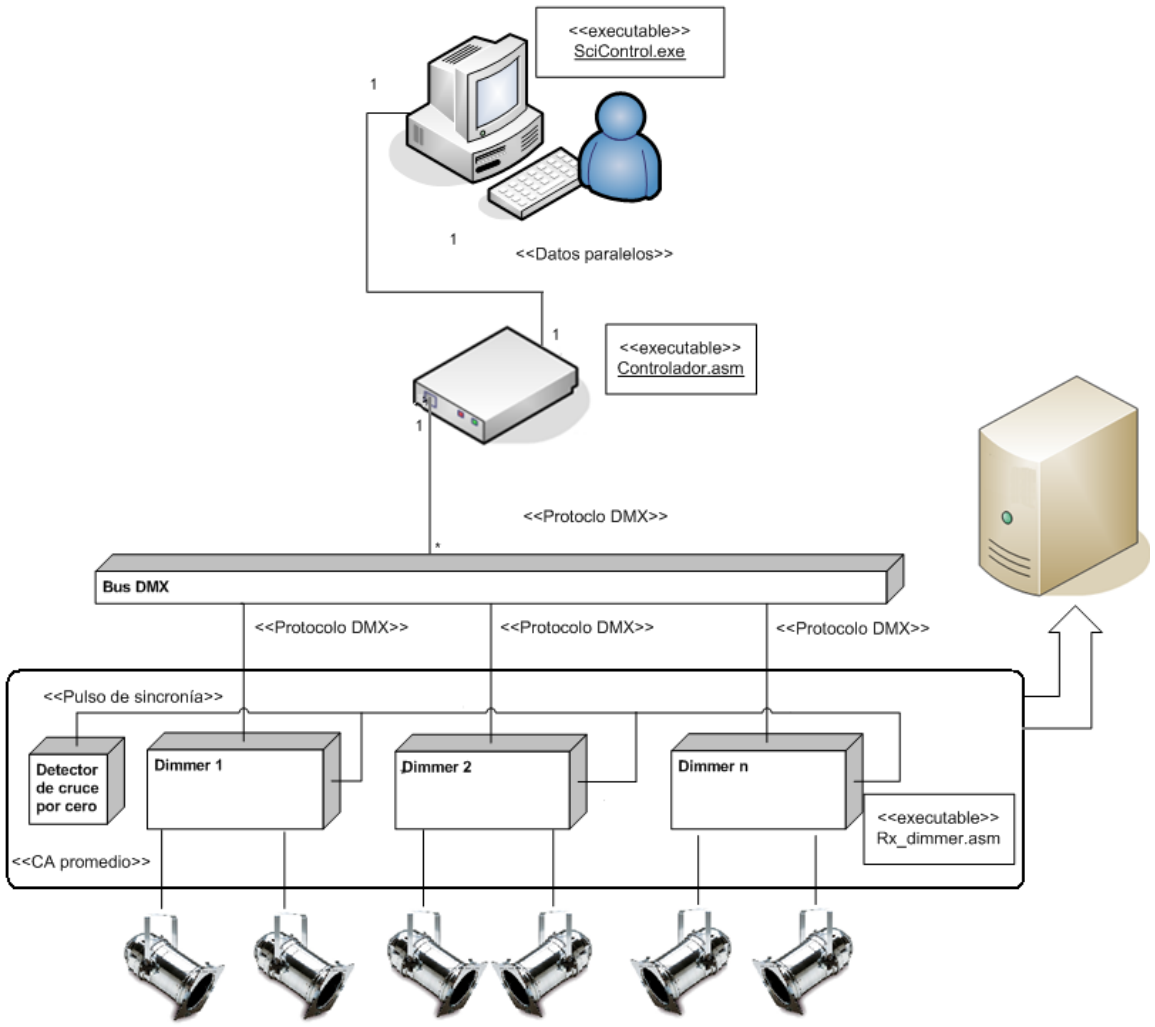

Figura 2.6 Diagrama de despliegue del sistema de control de iluminación DMX, SciDMX.

El diagrama de despliegue de la Figura 2.6 nuevamente enfatiza que el diseño y construcción del SciDMX debe ser modular, de esta manera siguiendo las siete fases de la metodología descrita, su desarrollo ha sido dividido de la siguiente manera:

Las fases 2, 3, 4 y 5 se han realizado tanto para el controlador DMX como para el dimmer, como se muestra en la Figura 2.7. Las fases 6 y 7 han sido desarrolladas de manera general para el SciDMX. Hay que notar que al inicio de éste capítulo, se ha presentado la primera fase de desarrollo del SciDMX.

La siguiente sección presenta las fases (2-5) correspondientes al controlador DMX. El capítulo 4, continúa con las fases 2, 3, 4 y 5 correspondientes al dimmer.

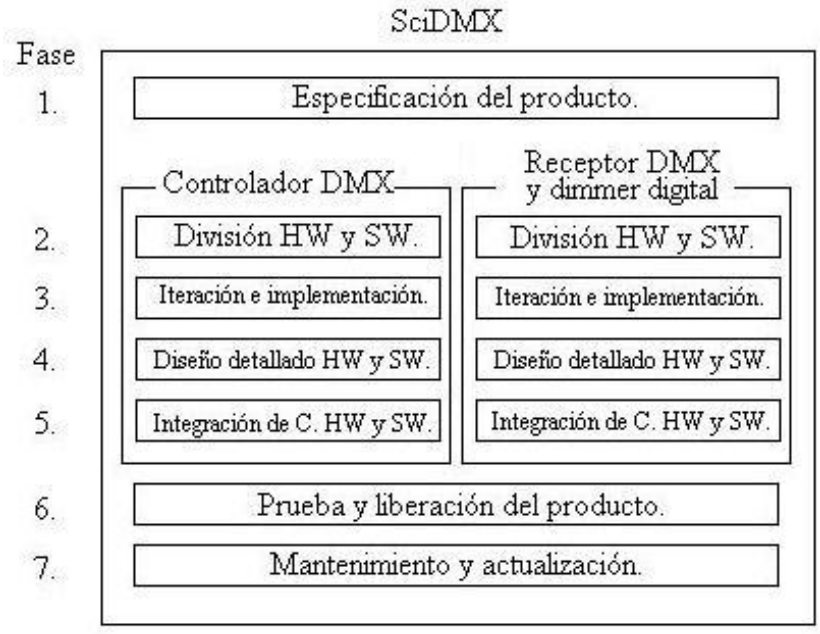

Figura 2.7. División de las fases de desarrollo del SciDMX.

2.3. *Fase 2: Particionamiento HW y SW del controlador DMX*

Para la fase 2 correspondiente al controlador DMX se emplea un diagrama de casos de uso para definir sus requerimientos e identificar claramente los actores que interactúan con él. El diagrama es mostrado en la Figura 2.8.

Los casos de uso identificados para el controlador son dos: "Recibir datos DMX" (de la PC) y "Transmitir trama DMX512". Como actores se han identificado a la "Computadora" y a los "Receptores DMX" en nuestro caso estos receptores son los dimmers.

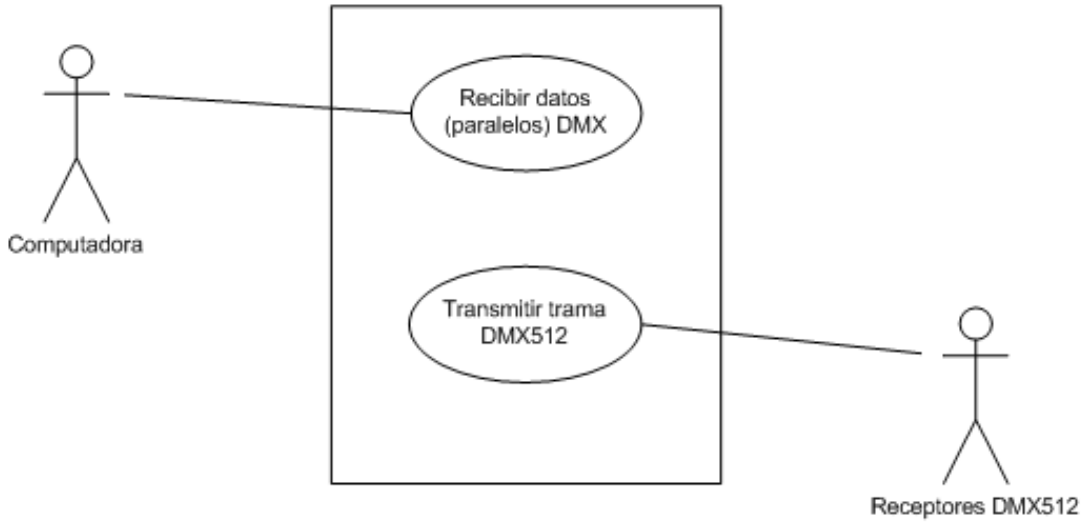

Figura 2.8. Casos de uso del controlador DMX.

Con base en las especificaciones del SciDMX y el diagrama de casos de uso anterior, la división del diseño del controlador en sus componentes HW y SW se realizó de acuerdo a la Tabla 2.1.

Tabla 2.1 División del diseño del controlador del SciDMX en sus componentes HW y SW.

HW	**Dispositivo**	**SW**	**Dispositivo**
Unión al medio físico	Transceptor	Control de interrupciones	Microcontrolador
Conexión a la PC	Buffers	Generación de la señal DMX512, que se envía a los receptores	Microcontrolador

El HW del controlador del SciDMX se compone de los siguientes dispositivos, Figura 2.9:

- MCU AT90S2313 de la firma Atmel.
- Transceptor de bus diferencial SN75176 compatible con las características del protocolo RS485.
- Buffer SN74HC245 y SN74HC244.
- Cable UTP (Unshielded Twisted Pair).
- Conector XLR hembra de 3 pines.
- Conector DB25 macho.
- Fuente de alimentación lineal de +5V.

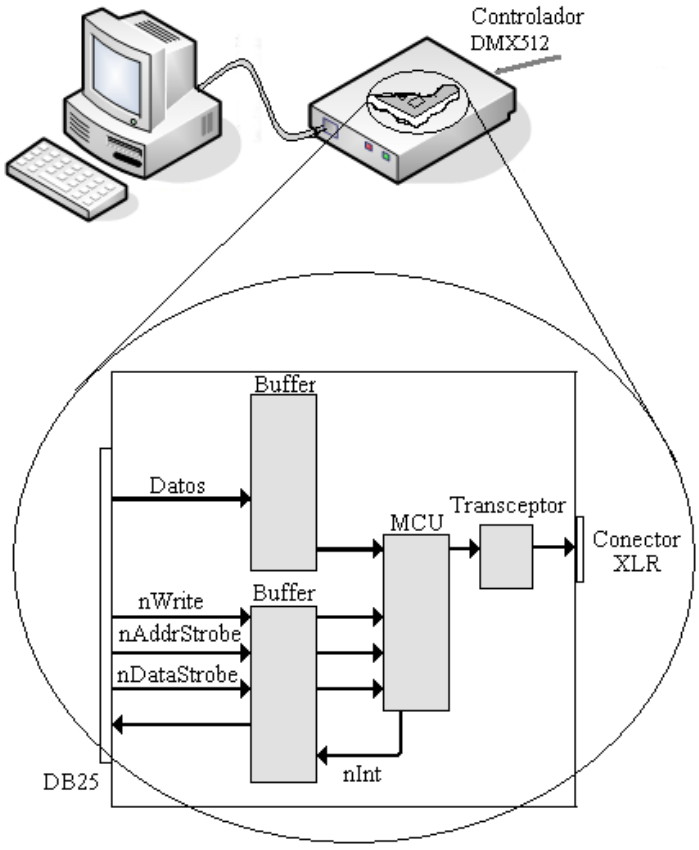

Figura 2.9. Diagrama a bloques del controlador DMX.

- Las tareas del controlador DMX son:

Recibir doce datos de entrada de 8 bits del PP de la PC, con los cuales generará la señal DMX512 cumpliendo con las especificaciones del estándar DMX512.

2.3.1. *Selección HW y SW para el controlador DMX*

Microcontrolador

Como controlador se seleccionó al MCU AT90S2313 por las siguientes razones:

- Su arquitectura es AVR RISC.
- Conjunto de instrucciones amplio.
- El número de E/S es suficiente para la aplicación.
- Velocidad de operación de hasta 20 MHz, permitiendo la realización de sistemas óptimos de bajo consumo de energía frente a velocidad de procesamiento.
- Disponible en el país.
- Programador del MCU AT90S2313 disponible.

Interfaz PC-controlador

Los métodos convencionales de conectar HW externo a la PC incluye el uso de tarjetas de interfaz de conexión. Este enfoque tiene varias desventajas, tal como:

Si el dispositivo es para uso de laboratorio o en el salón de clases, colocar HW dentro de la PC puede ser arriesgado para ésta última o para los usuarios (quienes pueden ser principiantes). Una pieza de HW cuando está fuera de la PC es fácilmente accesible para su revisión y medición de las señales internas. Al insertar una tarjeta de interfaz incrementa la complejidad de la operación. En algunos casos, agregar una tarjeta de interfaz puede ser una receta para un desastre, por ejemplo, cuando se maneja un multímetro, analizador lógico o un osciloscopio, puede originarse un inoportuno corto eléctrico sino se tiene cuidado.

Además, no todas las PC´s tienen una ranura de expansión disponible. Las PC´s portátiles no tienen ranuras de expansión convencional (aparte de las ranuras PCMCIA). Otras computadoras pueden tener ranuras, pero esas pueden estar destinadas para otros propósitos, tal como tarjetas de red y fax/módems.

Muchas aplicaciones que requieren transmitir los datos y control desde la PC, realmente no requieren lo sofisticado de una ranura de expansión.

Por tal razón, una forma de conectar HW externo deber ser simple, fácil y económica. Una alternativa para usar una tarjeta de interfaz es diseñar el HW para que esta pueda conectarse a la PC a través de alguna de sus interfaces comunes.

La Tabla 2.2 presenta las características de las interfaces comunes en una PC.

Tabla 2.2. Comparación de interfaces comunes en una PC.

Interfaz	**Formato**	**Número máx. dispositivos**	**Longitud máxima**	**Velocidad máxima**	**Uso típico**
RS-232	Serial asíncrono	2	50-100 ft 15-30 m	20 kbps (115 con HW dedicado)	Módem, mouse, instrumentación
USB	Serial asíncrono	127	16 ft (hasta 96 ft con 5 hubs) 4.8 m	1.5Mbps, 12Mbps, 480Mbps.	Mouse, teclado, drivers de discos, módem, audio.
Puerto Paralelo	Paralelo	2 (8 con soporte de cadena Daisy)	10-30 ft 3–9 m	8Mbps	Impresoras, discos, drivers de discos.

El RS-232 tiene varias ventajas, puede llegar a pensarse que la generación de la señal DMX puede llevarse a cabo desde el software de la PC, y solo utilizar un convertidor RS232 a RS485 como interfaz físico al bus DMX. Sin embargo, la velocidad del puerto serie hace imposible generar una señal de 250 Kbaudios, además de que en la actualidad muchas PC´s carecen de puerto serie, sobre todo las portátiles nuevas. Una buena opción para combinar la generación de la señal DMX con una PC, es utilizar una interfaz separada.

Con respecto a la interfaz USB, antes de decidir si el USB es adecuado para el controlador, es necesario conocer un poco más de cómo trabaja y que es lo que éste puede hacer. Una de las formas para saber si el USB es o no adecuado para la interfaz PC-controlador, es responder las siguientes dos preguntas:

1. ¿Cuales son los requerimientos mínimos que la PC debe tener para utilizar periféricos USB? Para utilizar el USB, una PC necesita HW y SW de soporte, El HW consiste de controlador de host USB, y un hub raíz con uno o más puertos USB. El SW de soporte es un sistema operativo que soporta USB.

2. ¿Como conectar los dispositivos a la PC? Los componentes físicos del USB consisten de circuitos, conectores y cables entre un host y uno o más dispositivos. Algunos controladores son compatibles con familias de microcontroladores existentes. Esto tiene dos ventajas: una es que muchos desarrolladores están familiarizados con la arquitectura y el set de instrucciones, y la familiaridad proporciona una gran delantera para el inicio de un proyecto. Algunos de los microcontroladores son: el AT76C711 de la firma Atmel o el PIC16C7x5 de la firma Microchip Technology [Axelson, 2001], sin embargo, éstos aún no están disponibles en el país.

Por el lado del puerto paralelo, éstos están disponibles universalmente sobre las PC´s. Es un puerto externo a la PC, por lo que no es necesario abrirla. Otro beneficio del puerto paralelo es que la IEEE continúa la mejora de las especificaciones del puerto paralelo. Durante los últimos años, los programadores cada vez más han favorecido al puerto paralelo como medio para conectar sistemas de seguridad de software (candados o dongles), así como varios tipos de impresoras de alto rendimiento

[Axelson, 1996], [Gadre, 2001] y quizá la más importante para nuestro caso, el puerto paralelo permite mantener conexión a alta velocidad y bajo costo [O´Sullivan et al, 2004], [Stalling, 1997].

El puerto paralelo fue originalmente diseñado como una interfaz limitada en uso y de una vía para controlar impresoras. Sin embargo, los chispsets de los host actuales para el puerto paralelo soportan múltiples modos de operación para reflejar este rol del puerto expandido como una interfaz mejorada.

Uno de esos modos avanzados de operación es conocido como EPP (Enhanced Parallel Port), puede proveer transferencias de datos bi-direccional de 8 bits a velocidades que exceden 1 MByte/s [Atmel, 2002].

El puerto paralelo es así una solución elegante y económica para sistemas de control de iluminación con una PC [Feldmeier, 2003], por tal razón ha sido elegido para la interfaz entre el controlador y la PC.

Interfaz Controlador-Receptores DMX512

Como interfaz física entre el controlador DMX y los dimmers, se seleccionó al transceptor de bus diferencial SN75176B debido a que cumple con la norma ISO 8482 (ANSI TIA/EIA-422-B y TIA/EIA-485-A), además de las siguientes características:

- La derivación amplificada puede tener hasta un máximo de 32 dispositivos EIA-485 conectados [Axelson, 1999].
- Aumenta la distancia del circuito EIA-485 a 1200 metros.
- Permite conectar amplificadores EIA-485 adicionales
- Cada amplificador aumenta la señal EIA-485 en 32.
- Transceptor bidireccional (para cambios futuros).
- Alimentación: +5V.
- Diseñado para transmisiones multipunto con líneas de bus largo en ambientes ruidosos.

Herramientas de desarrollo para el diseño HW y SW

Como herramientas de desarrollo para el diseño HW y SW del controlador se empleó:

Para la escritura, depuración y simulación de aplicaciones AVR se utilizó el SW de aplicación AVR Studio versión 3.53 de la firma Atmel, para el sistema operativo Windows 2000/XP. El AVR Studio incluye un entorno de desarrollo integrado (IDE, Integrated Development Enviroment), el cual contiene: editor, ensamblador, depurador y simulador, además de brindar soporte a emuladores AVR y tarjetas de desarrollo como la STK500. El AVR Studio es una aplicación de uso público y de distribución gratuita (freeware).

Programador ISP para MCUs AVR: Permite la descarga del código en forma rápida debido a la interfaz paralela con la PC. Éste programador no es de alguna firma, sino fue construido con anterioridad en base a los algoritmos de programación que proporciona libremente la firma Atmel.

2.4. *Fase 3: Iteración y desarrollo del controlador*

Las tareas que se realizaron para el desarrollo del controlador en el entorno *AVR Studio* son las siguientes:

- Escribir y depurar programas de prueba.
- Simular la lectura y escritura a direcciones de memoria externa, así como la configuración de los registros del MCU.
- Simular la escritura de bits en los registros de I/O.
- Verificar la ejecución de las subrutinas, la activación de interrupciones y el tiempo de ejecución de las mismas.
- Descargar el programa al MCU mediante el programador.
- Programar los bits internos del MCU para seleccionar las opciones de funcionamiento.
- Las tareas iterativas HW y SW, de corrección o de ajuste son las siguientes:
- Asignar direcciones de memoria interna en el MCU.
- Asignar funciones a los puertos del MCU.
- Configurar los registros del MCU.
- Agregar y/o modificar las conexiones HW necesarias.

2.5. *Fase 4: Diseño paralelo HW y SW del controlador DMX*

2.5.1. *Diseño HW del controlador*

Una vez seleccionado los componentes HW del controlador DMX del SciDMX, se definen los siguientes aspectos:

- Asignación de funciones a los puertos del MCU del controlador del DMX512.
- Asignación de los registros del MCU a emplear.

2.5.1.1. *Asignación de funciones de los puertos del MCU del controlador DMX*

La Tabla 2.3 muestra la asignación de los puertos y pines del MCU del controlador DMX.

Tabla 2.3. Función para cada pin de los puertos del MCU.

Puerto	Pin	E/S	Nombre	Función
B	PB0-7	E	Datos del PP.	Bus de datos del PP al MCU
D	PD0	S	DMX512 (TTL)	Señal DMX generada.
	PD1	E	nDataStrobe	Indica al MCU la escritura de un dato.
	PD2	E	nAddressStrobe	Indica al MCU la escritura de una dirección.
	PD3	E	nWrite	Indica al MCU la escritura de un dato o de una dirección.
	PD4	S	nIntr	Indica a la PC, que el MCU esta listo para la recepción de datos.

	PD5	S	Led DMX	Indicador de la generación de señal.
	PD6	S	nWait	Indica a la PC, la recepción exitosa de un dato.

2.5.1.2. *Asignación de los registros del MCU a emplear para el controlador DMX*

La Tabla 2.4 muestra la relación de los registros del MCU del controlador DMX.

Tabla 2.4. Mapa de memoria de los registros del MCU del controlador DMX.

Registro	**Nombre**	**Dirección**
SPL	Apuntador de pila (bajo)	3Dh
MCUCR	Control general del MCU	35h
PORTB	Datos del puerto B	18h
DDRB	Dirección de datos del puerto B	17h
PORTD	Datos del puerto D	12h
DDRD	Dirección de datos del puerto D	11h
GIMSK	Mascara de interrupciones generales	3Bh

2.5.2. *Diseño del SW del controlador DMX*

El diseño del SW del controlador se basa en las especificaciones iniciales del sistema, considerando los siguientes factores:

- Interfaz del MCU con el puerto paralelo en el modo EPP (Enhanced Port Parallel).
- Asignación de puertos del MCU.
- Interfaz de la capa física hacia los receptotes DMX.

En el SW del controlador, el código de configuración del MCU se basó en una programación estructurada considerando la fase de mantenimiento y actualización. Éste está basado en el diagrama de estados de la Figura 2.10. En las Tablas 2.5 y 2.6 se presentan la descripción de los estados y los estímulos del programa, respectivamente. El programa esta compuesto de una subrutina, una rutina de servicio a interrupción (ISR, Interrupt Service Routine) externa y una ISR por comparación del TMR1.

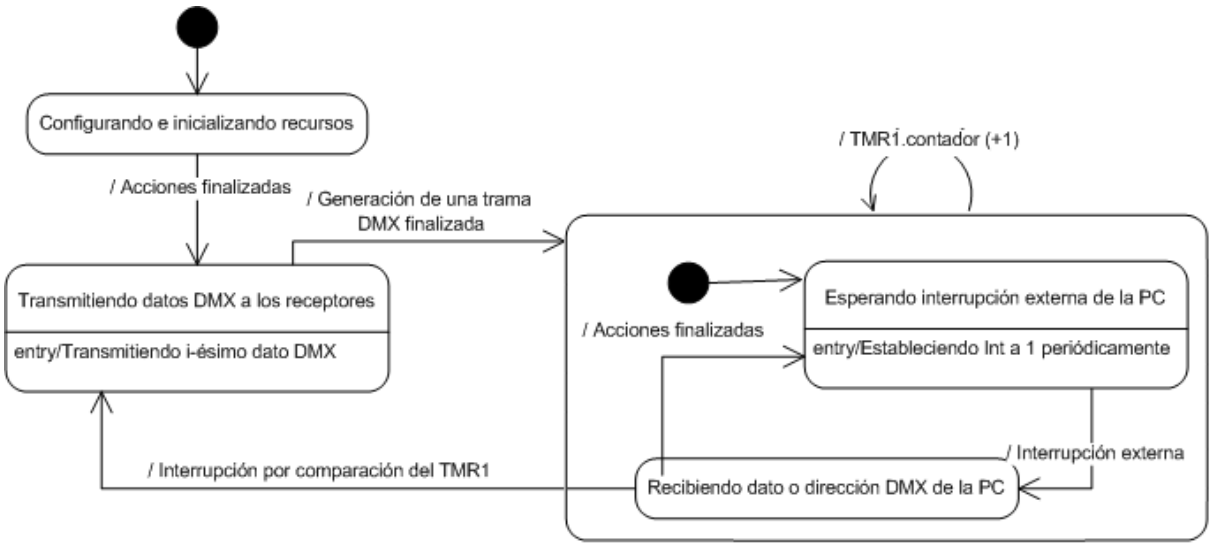

Figura 2.10. Diagrama de estados del programa del controlador del SciDMX.

Tabla 2.5. Descripción de los estados del controlador.

Estado	**Descripción**
Configurando e inicializando recursos	El programa del MCU configura la pila, los puertos y los registros a utilizar. Además de inicializar la memoria SRAM.
Transmitiendo datos DMX a los receptores	El programa del MCU genera la señal DMX.
Actualizando datos de la PC	El programa del MCU esta listo para la actualización de los datos de la PC.
Esperando interrupción externa de la PC	El programa del MCU le indica a la PC, por medio de la señal *nInt* que esta listo para recibir actualizaciones de datos.
Recibiendo dato o dirección DMX de la PC	El programa del MCU, detecta por medio de las señales *nAddressStrobe* y *nDataStrobe*, si es una escritura de dirección o de dato DMX, respectivamente. El dato recibido se almacena en la dirección (memoria RAM) recibida previamente

Tabla 2.6 Estímulos del programa del controlador.

Estímulo	Descripción
Generación de una trama DMX finalizada	El programa del MCU ha terminado de generar una trama DMX.
Interrupción externa	La PC le indica al MCU, por medio de la señal *nWrite* la escritura de una dirección o de un dato.
TMR1 contador (+1)	El *timer* incrementa su cuenta, mientras permanece en el estado *Recibiendo dato o dirección DMX de la PC*.
Interrupción por comparación del TMR1	La cuenta del TMR1 ha llegado a igualar el valor de comparación. El tiempo para actualizar los datos de la PC ha finalizado.
Acciones finalizadas	El conjunto de instrucciones del estado han terminado.

Los detalles de cada uno de los estados se describen a continuación.

2.5.2.1. *Estado: Configurando e inicializando recursos del MCU*

Para el estado *configurando e inicializando recursos del MCU*, las subrutinas son: configurar E/S, e inicializar memoria RAM interna.

Configurar E/S del MCU

La subrutina de configuración de entradas y salidas se ejecuta después que se alimenta con 5V o después de un reinicio manual del MCU del controlador. La Figura 2.11 muestra el diagrama de flujo de la subrutina configurar E/S del MCU. La subrutina realiza lo siguiente:

Inicializar la pila del MCU: se configura el registro SPL para asignar la dirección de memoria que debe tener el apuntador de la pila del MCU.

Declarar el vector de interrupciones del MCU: a la interrupción externa se le asigna su correspondiente número de vector, así como su etiqueta de identificación.

Configurar los puertos B y D por medio de los bits de los registros DDRB y DDRD para habilitarlos como entrada y salida, respectivamente, de acuerdo a la Tabla 2.3.

Configurar la interrupción externa INT1: se configuran los bits del registro MCUCR para seleccionar la activación por flanco de caída de la interrupción INT1.

Configurar la interrupción por comparación del TMR1, para generar interrupciones cada 37.6 μs.

Habilitar la interrupción por comparación del TMR1, configurando el registro TIMSK.

Habilitar la interrupción externa INT1: configurando el registro GIMSK.

Habilitar la interrupción global para habilitar las interrupciones que el PP puede generar.

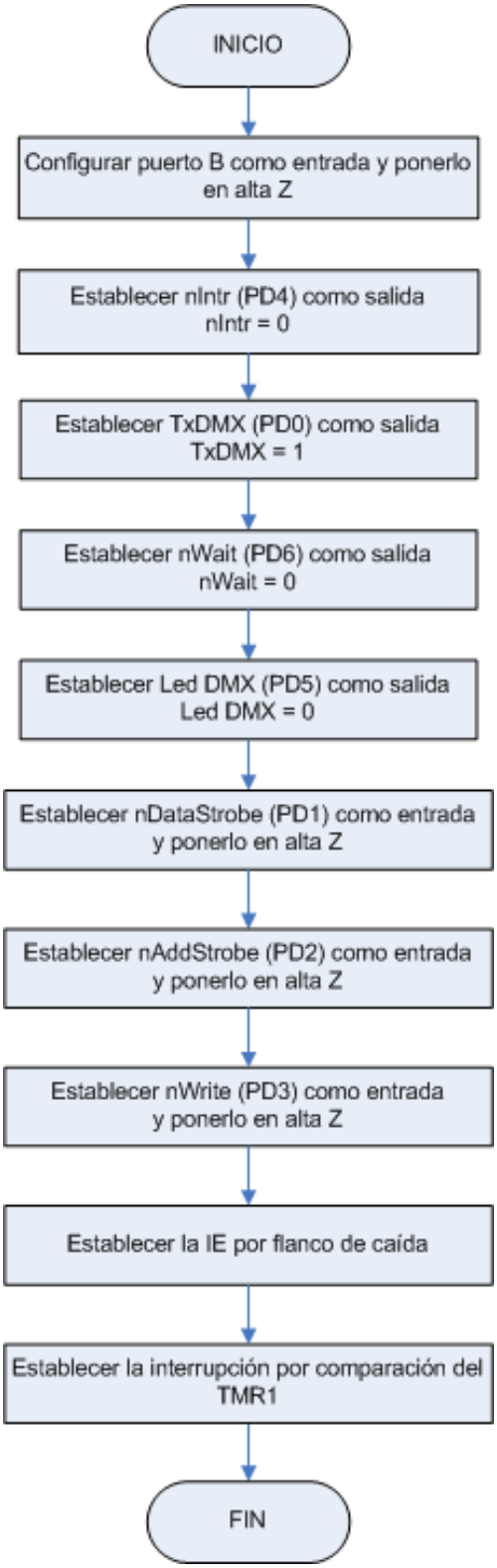

Figura 2.11. Diagrama de flujo de la subrutina configurar E/S del MCU.

Inicializar memoria SRAM interna

El mapa de memoria del MCU se muestra en la Figura 2.12. La memoria SRAM utilizada inicia en la localidad $60 y termina en la $68, pudiéndose almacenar hasta 120 datos desde $60 a $D8, dejando 7 localidades extra para la pila del MCU, la asignación de las localidades se muestra en la Tabla 2.7.

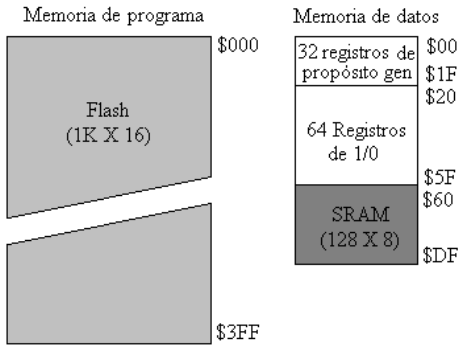

Figura 2.12. Mapa de memoria del MCU AT90S2313.

La subrutina de inicialización de memoria SRAM interna se ejecuta después de que las E/S han sido configuradas.

Esta subrutina se encarga de inicializar a cero las doce localidades que almacenarán posteriormente los datos que se reciben del PP, con la intención de que sí el controlador aún no ha sido conectado a la PC, pero este se encuentre alimentado, los datos que transmita a los receptores sean valores de intensidad nula. La Figura 2.13 muestra el diagrama de flujo para la inicialización de la memoria SRAM.

Tabla 2.7. Asignación de las localidades de memoria del MCU para almacenar los datos recibidos (PP).

Receptor	Localidad	Descripción
1	$60	Dato 1
	$61	Dato 2
2	$62	Dato 3
	$63	Dato 4
3	$64	Dato 5
	$65	Dato 6
4	$66	Dato 7
	$67	Dato 8
5	$68	Dato 9
	$69	Dato 10
6	$6A	Dato 11
	$6B	Dato 12

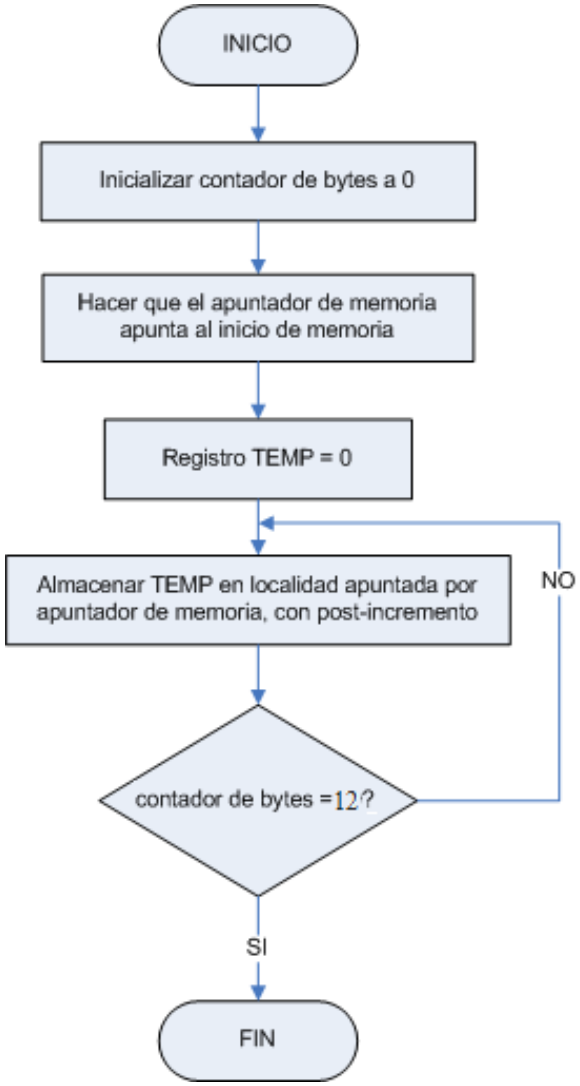

Figura 2.13. Diagrama de flujo de la subrutina de inicialización de memoria SRAM.

Al finalizar las acciones del estado: configurando e inicializando recursos, el programa pasa inmediatamente al siguiente estado, el cual se encarga de generar y transmitir la señal DMX a los receptores.

2.5.2.2. *Estado: Transmitiendo datos DMX a los receptores*

La Figura 2.15 presenta el diagrama de flujo de la subrutina encargada de generar la señal DMX. Tal generación se lleva a cabo con los datos que se tienen almacenados en memoria SRAM. La señal a transmitirse se genera en el pin PD0 del MCU, denominado como línea de ahora en adelante. El esquemático del controlador DMX se presenta en la Figura 2.14.

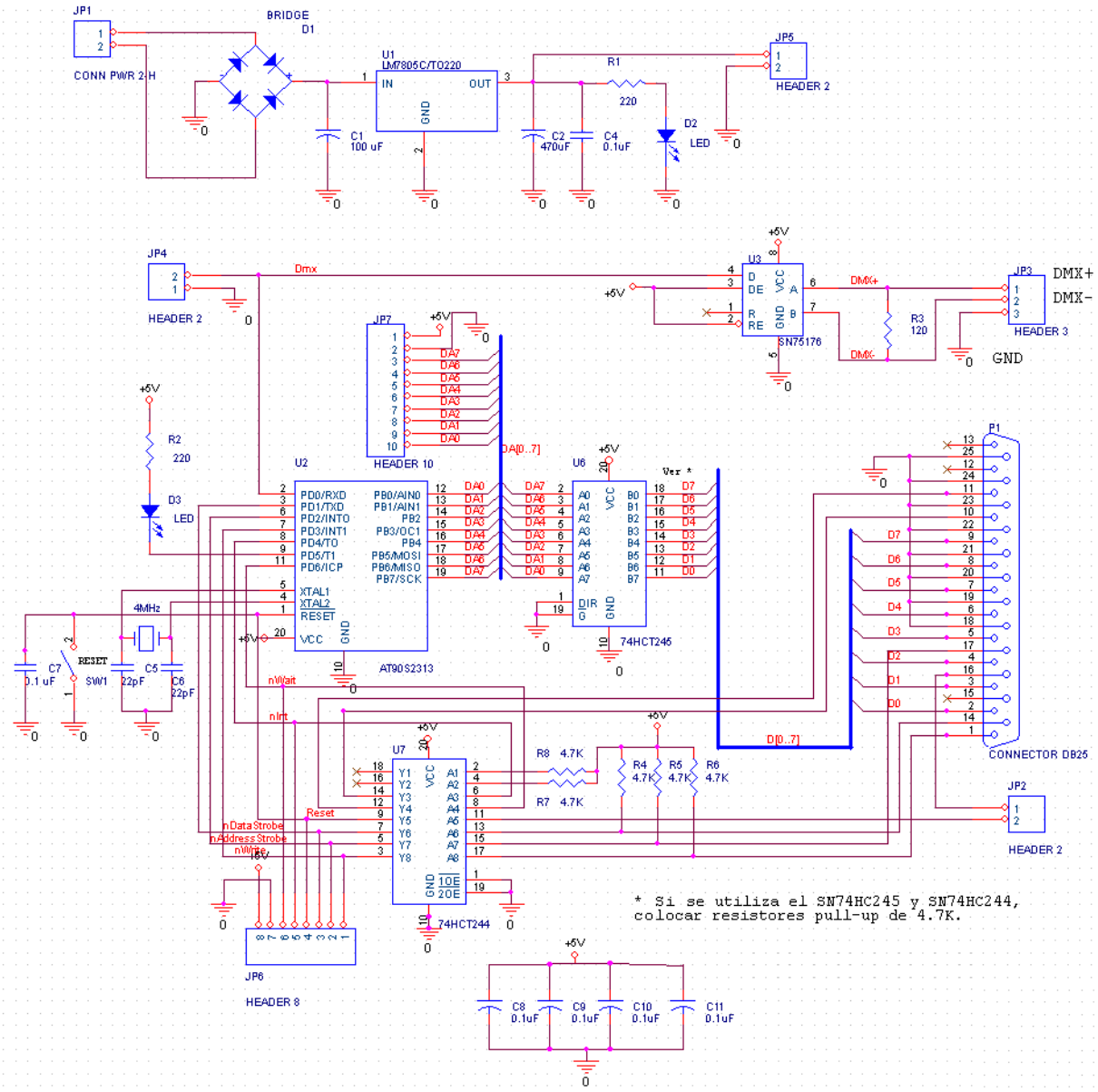

Figura 2.14. Esquemático del controlador DMX.

El funcionamiento del programa en este estado es el siguiente:

Inicialmente la línea permanece en alto, al entrar a este estado, se genera el Space for Break, el cual consiste en mantener en bajo a la línea durante 88 µs, para posteriormente producir la MAB, la cual mantiene la línea en alto durante 8 µs, para después generar la Slot Time, que no es más que enviar el código de inicio (START CODE) igual a 00h, con el bit de inicio en bajo y dos bits de stop (altos). Una vez terminado, la línea debe mantenerse en alto un intervalo entre 0 y 1 segundo, particularmente se ha seleccionado 40 µs, que es el tiempo promedio que tarda un receptor en procesar la señal DMX.

Hasta esta parte se ha generado la parte inicial de una trama DMX, queda enviar cada uno de los bits de los doce bytes de los datos almacenados en memoria SRAM. Para esto, el programa se apoya en

dos ciclos anidados, un ciclo cuenta los bytes y el segundo cuenta y procesa cada uno de los bits de cada uno de los byte de los doce datos.

El procesamiento de los bits a transmitir por la línea consiste en: enviar el bit de inicio (pulso bajo de 4 μs), posteriormente en un registro temporal (TEMP) se respalda el byte a transmitir (BYTE). Después se obtiene solo el bit LSB de TEMP, se envía por la línea (PD0) y se mantiene el nivel del bit hasta completar 4 μs (tiempo de bit). Posteriormente, el contenido de BYTE se rota a la derecha, el contador de bits enviados es decrementado, y se evalúa si ya se han enviado ocho bits. Si no, se realiza nuevamente el respaldo de BYTE en TEMP, y así sucesivamente.

Cuando ya se han enviado los ocho bits del enésimo byte, se transmiten los dos bits de stop y después se evalúa si han sido enviados ya los doce bytes, sino se carga en BYTE el dato correspondiente que esta almacenado en SRAM, y se procesa de la misma forma.

Es importante mencionar que la generación de la señal DMX, no se realiza directamente con la UART del MCU, debido a que se necesita tener un mayor control sobre la señal y los retardos. La señal se genera en el pin PD0 del MCU, mediante la generación de retardos y transiciones de bajo a alto o viceversa, cumpliendo con el estándar DMX. La conexión del pin PD0 hacia el transceptor SN75176 se muestra en el esquemático de la Figura 2.14.

2.5.2.3. *Estado: Actualizando datos de la PC*

El estímulo que permite entrar al súper estado *Actualizando datos de la PC* es cuando la generación de una trama DMX finaliza, de esta manera la recepción de la dirección y del dato de cada uno de las doce datos DMX es posible hacerla solo en los intervalos de una MBB.

Tomando en cuenta que el MCU debe considerar intervalos de tiempo para permitir que por medio de la interrupción se actualicen los datos provenientes del programa de control, y habiendo analizado el diagrama de tiempos del estándar DMX, se observó que debe existir una parte del programa donde se habilite/deshabilite la IE sin que afecte el cumplimiento de los tiempos que especifica el estándar DMX. Esta parte es la MBB (Mark Before Break) la cual puede tener intervalos de 0 a 1 segundo como máximo. De esta manera, el diagrama de flujo inicia habilitando la IE, para posteriormente generar la MBB, que no es más que mantener en alto la línea, un intervalo de 900 ms, dando la posibilidad que en este intervalo, se genere una IE para la actualización de los datos.

La subrutina para este estado se realiza todo el tiempo y sólo es suspendida por la interrupción de sobreflujo del TMR1, al generarse una interrupción externa para cambiar al estado: *Recibiendo datos DMX de la PC*.

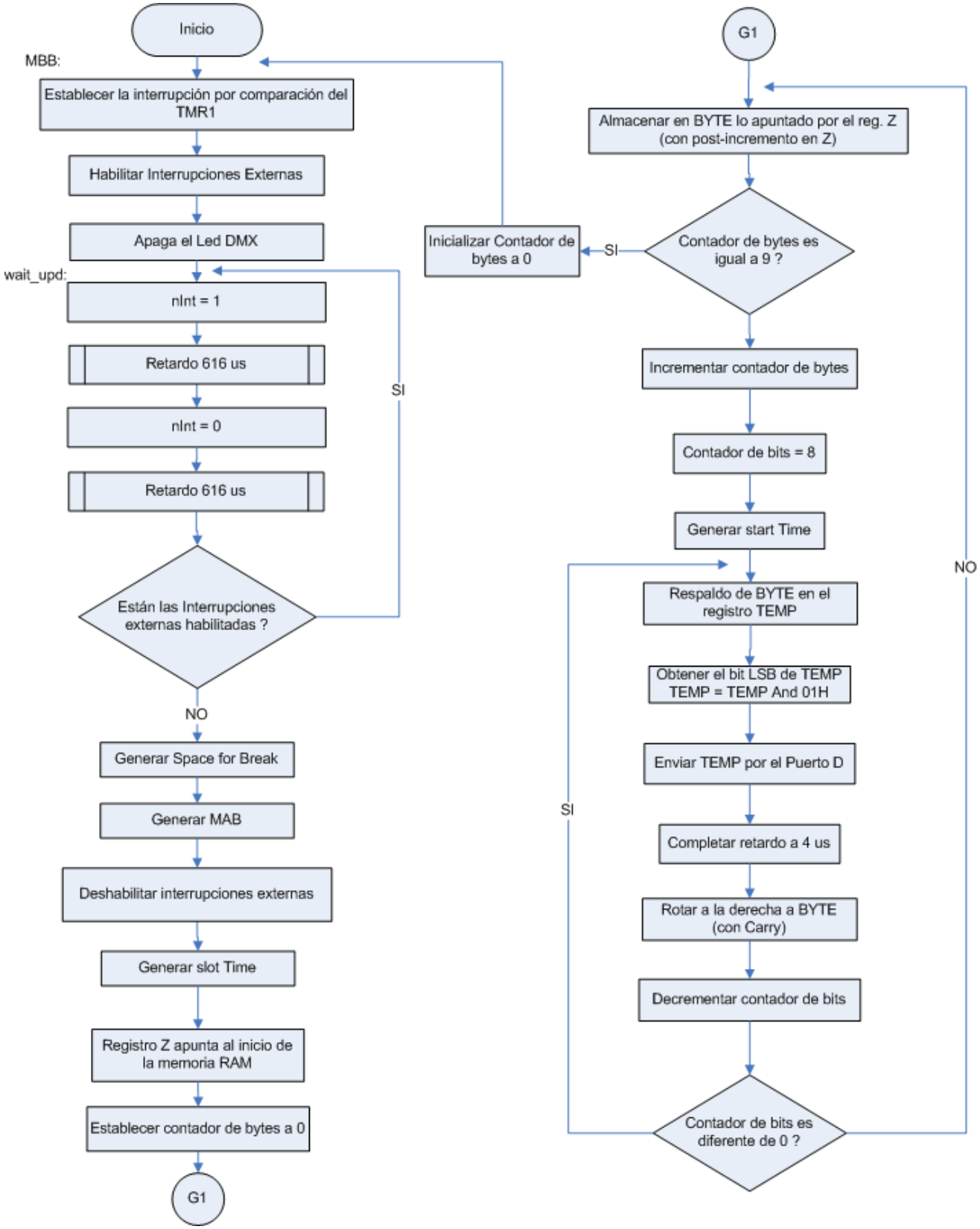

Figura 2.15 Diagrama de flujo para la generación de la señal DMX.

El súper estado esta constituido de dos sub-estados: *Esperando interrupción externa de la PC* y *Recibiendo dato o dirección DMX de la PC*. Al entrar al sub-estado *Actualizando datos de la PC*, el TMR1 es configurado para generar una interrupción por comparación cada 37.6 μs. La cuenta del *TMR1* es incrementada en cada ciclo de reloj del MCU. Cuando sucede la interrupción por comparación del TMR1, se lleva a cabo el diagrama de flujo de la Figura 2.16. En el servicio a esta interrupción, la interrupción externa es deshabilitada, el TMR1 es detenido, el Led indicador DMX es encendido y finalmente la señal *nInt* es puesta baja. Lo anterior, hace que las IE´s de la PC sean ignoradas, dando oportunidad a que la señal DMX vuelva a transmitirse con los valores que fueron almacenados y actualizados en memoria SRAM. Cuando la generación de una trama DMX finaliza, el TMR1 se habilita dando oportunidad de actualizar los datos.

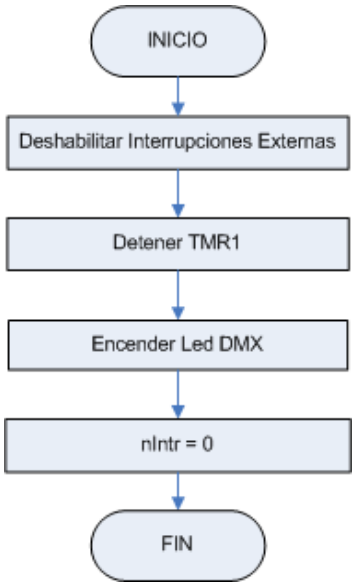

Figura 2.16. Diagrama de flujo de la subrutina de interrupción por comparación del TMR1.

2.5.2.4. *Estado: Esperando interrupción externa*

Este sub-estado realiza la secuencia del etiquetado ciclo wait_upd, Figura 2.15, básicamente se evalúa si la interrupción externa esta habilitada, si es así, se generan periódicamente pulsos de nInt hacia la PC, para indicarle que el MCU (controlador) esta listo para actualizar algún dato que haya cambiado, en caso de que la interrupción externa este deshabilitada, se saldrá del súper estado *Actualizando datos de la PC*.

2.5.2.5. *Estado: Recibiendo dato o dirección DMX de la PC*

El programa cambia a este sub-estado cuando se cumplen dos condiciones: el MCU pone la señal nInt hacia la PC, y ésta inicia un ciclo de escritura de dirección o de dato.

La versión del presente controlador a diferencia del presentado en [Yescas et al, 2006] radica en que este maneja el handshaking con la PC, no requiere señal de inicio ni de reloj por separado, lo que permite una comunicación más confiable. El controlador debe soportar el handshaking en modo EPP para hacerlo compatible con las señales que genera automáticamente el hardware del PP, cuando el programa

de control hace una escritura al registro de control. Las conexiones utilizadas se muestran en el esquemático del controlador.

La recepción que puede ser de dirección o de dato, se realiza mediante el handshaking entre el MCU del controlador y el PP de la PC, en el modo EPP. La Figura 2.17a y 2.17b muestran el diagrama de flujo que permite realizar esto.

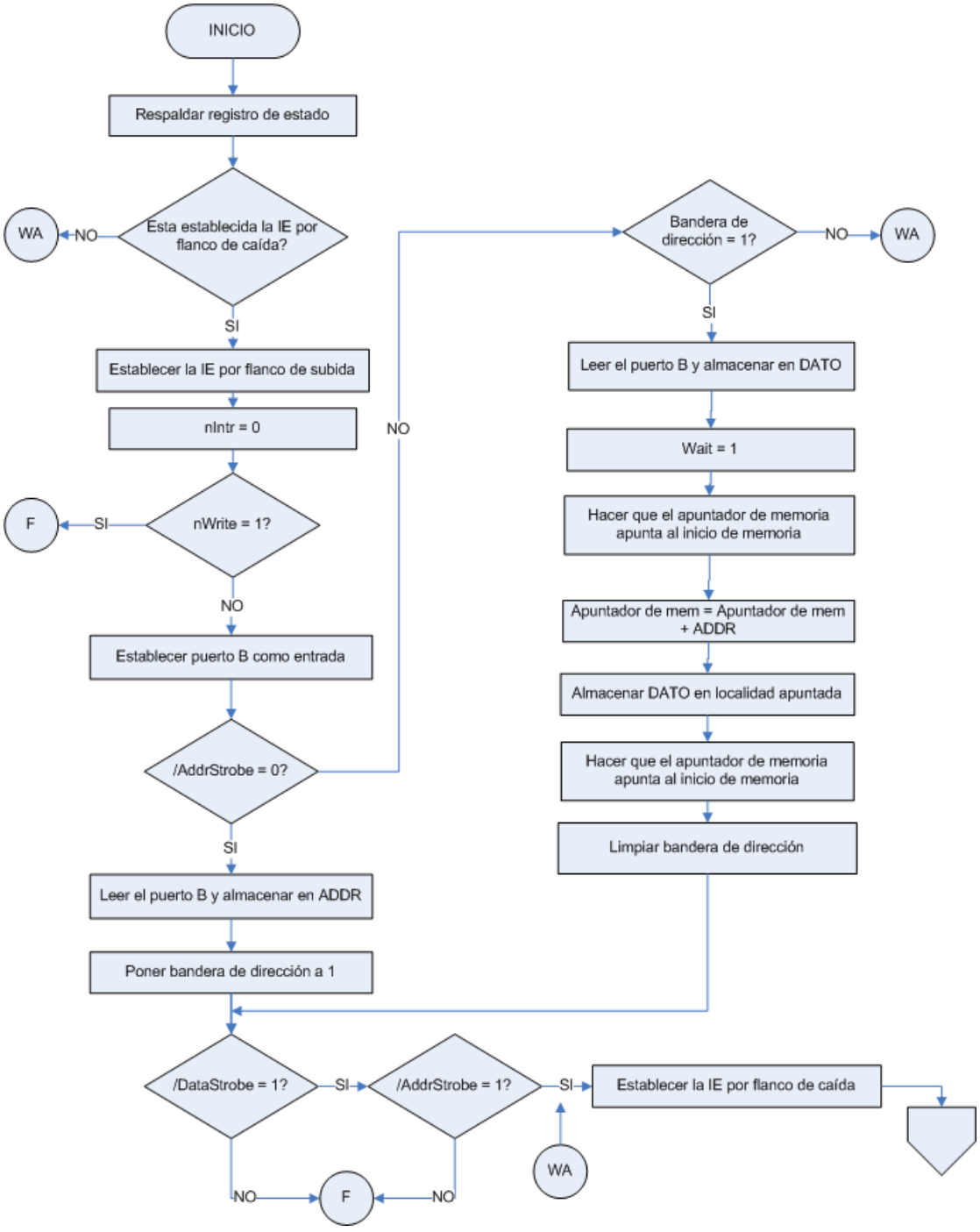

Figura 2.17a. Diagrama de flujo para el estado *Recibiendo dato o dirección DMX de la PC*.

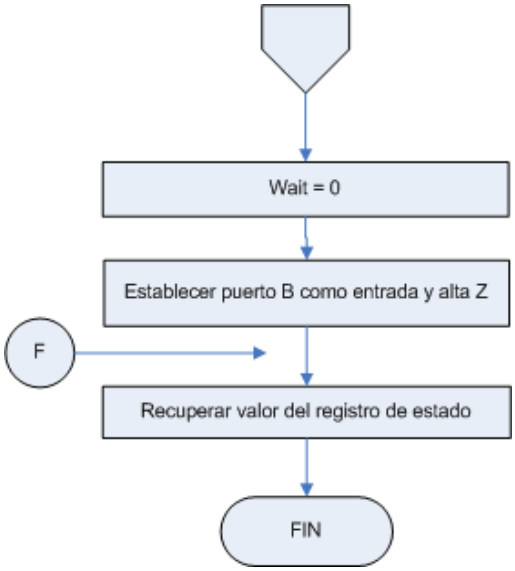

Figura 2.17b. Diagrama de flujo para el estado *Recibiendo dato o dirección DMX de la PC*.

El funcionamiento del programa en este sub-estado es el siguiente:

Este sub-estado es básicamente un servicio a la interrupción externa INT1 (PD3). Inicialmente la IE está configurada por flanco de caída, para detectar el cambio en la señal nWrite provocando que entre a este sub-estado.

Un ciclo de escritura de dirección o de dato, esta dividido en dos etapas: etapa inicial (EI) y etapa final (EF). Por otro lado, la IE tiene dos cambios de configuración: en la EI ésta es configurada por flanco de caída (FC) y en la EF es establecida por flanco de subida (FS). Estos cambios son efectuados, en las transiciones FC Y FS del diagrama de tiempos de la Figura 2.18.

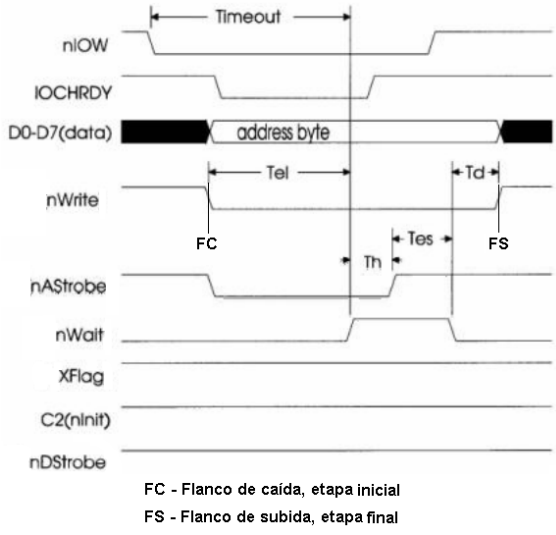

Figura 2.18. Diagrama de tiempos para el ciclo de escritura de dirección.

Al entrar al sub-estado lo primero que se hace es verificar en que etapa se está, si es una EI la IE se configura por FS, se quita la señal nInt (nInt = 0) y lo que sigue es verificar si es una escritura de dirección o de dato evaluando el nivel de la señal nAddrStrobe (activa baja). En el caso de escritura de dirección, se lee el puerto B y el valor leído es almacenado en ADDR, posteriormente se verifica si la señal nAddrStrobe ya ha regresado al estado alto (EF), sino el servicio a la IE finaliza, en otro caso la IE se configura por FC, se quita la señal nWait (nWait = 0) y la IE finaliza.

Cuando sucede una nueva IE y anteriormente no se dio el caso de la EF, se vuelve a verificar en que etapa se está, lo cual es la EF, por lo que la IE se configura por FC, se quita la señal nWait (nWait = 0) y la IE finaliza.

Como es posible observar, para almacenar un dato en memoria SRAM se debe recibir con anterioridad una dirección, de lo contrario se desconoce la localidad en la que debe ser almacenado el dato. Así, para que el programa del MCU pueda garantizar lo anterior, se utiliza una bandera de dirección, ésta es puesta a 1 una vez que se ha recibido una dirección, cuando el PP realiza un ciclo de escritura de dato, ésta tomará un efecto exitoso de la siguiente manera:

Una escritura de dato, es muy similar a la escritura de dirección. La diferencia radica en que de las dos señales nAddrStrobe y nDataStrobe, la última debe ser baja. De manera analoga, al entrar a este sub-estado, el programa verifica en que etapa se encuentra, cuando es una EI el MCU tiene que verificar si la bandera de dirección ha sido puesta (conocer si ya se ha recibido una dirección), en caso afirmativo se lee el puerto B y el valor leído es almacenado en el registro DATO, la señal nWait es puesta (nWait = 1), posteriormente el valor del registro DATO es almacenado en la localidad de dirección $60 + ADDR. Posteriormente, se evalúa si la señal nDataStrobe ya ha regresado al estado alto (EF), sino el servicio a la IE finaliza, en otro caso la IE se configura por FS, se quita la señal nWait (nWait = 0) y el servicio a la IE finaliza. En la EF, la IE se configura por FC, se quita la señal nWait (nWait = 0) y la IE finaliza.

2.6. *Fase 5: Integración HW y SW del controlador DMX*

La integración HW y SW del controlador DMX consistió en realizar las siguientes tareas:

- Verificar las conexiones de los componentes HW.
- Ejecutar programas de prueba a los periféricos (PP, transceptor).
- Descargar el programa principal al MCU mediante el programador.
- Ejecutar el programa del MCU y validar su funcionamiento.

2.7. *Fase 6. Verificación del controlador DMX*

Para verificar el correcto funcionamiento del controlador DMX, se realizaron pruebas de la siguiente forma:

- Transmisión de tramas DMX con datos almacenados en memoria SRAM programados previamente en el programa de control, esto sin la conexión del PP.
- Medición de los tiempos de BREAK, MAB, MBB, valor de dato (dato DMX) y los bits de stop, por medio del osciloscopio.
- Transmisión de datos DMX de la PC al controlador DMX, por medio de una aplicación simple de escritura de dirección y de datos, realizada en VB.

2.8. *Fase 7: Mantenimiento y actualización del controlador DMX*

La Tabla 2.8 muestra las actualizaciones realizadas al controlador del SciDMX durante su desarrollo.

Tabla 2.8. Versiones de actualización del controlador durante su desarrollo.

Versión	Actualizaciones
0.1	Configuración de registros del MCU para habilitar las funciones requeridas en las especificaciones iniciales. El HW del controlador DMX contiene solo el MCU. Establecimiento de los retardos y transiciones para generar la señal DMX. Transmisión de doce datos DMX almacenados en memoria RAM del MCU. Medición (en el controlador) de los tiempos de BREAK, MAB, MBB, valor de dato (dato DMX) y los bits de stop, por medio del osciloscopio. Ajuste de los tiempos por medio de retardos en el programa del MCU.
0.2	El HW del controlador DMX contiene: MCU y un dip-swicth, los cuales permiten cambiar en forma manual el valor de los datos DMX hacia el MCU. Configuración de la INT1 del MCU e implementación del handshake EPP, para dar soporte a la recepción de los datos DMX. Programación de las subrutinas e ISR´s para modificar los datos DMX. Transmisión de datos establecidos mediante un dip-switch, emulando los datos provenientes del PP, a una velocidad de transferencia de 250 Kbps.s
0.3	Se agrega dos transceptores SN75176 al HW existente, uno como transmisor y otro como receptor, conectados vía un cable UTP de 60 metros. Comunicación DMX a 250 Kbps. Recepción exitosa de tramas en el transceptor que esta conectado como receptor. Medición (en el transceptor receptor) de los tiempos de BREAK, MAB, MBB, valor de dato (dato DMX) y bits de stop, por medio del osciloscopio Tektronics TDS210.

Capítulo 3. Desarrollo del programa de control

El presente capítulo describe el diseño y desarrollo del programa de control de la PC, se presentan las consideraciones tomadas en el programa orientado a eventos realizado en VB.

La recolección de información del programa que controla la iluminación se realizó por medio de revisiones a programas comerciales que realizan las mismas funciones. Partiendo de estas revisiones se obtuvieron los requerimientos generales del programa de control, que se detallan en los casos de uso.

3.1. *Descripción del proceso de control de iluminación espectacular*

El proceso de control de iluminación se realiza con la finalidad de darle a un operador la facilidad y operabilidad de controlar la iluminación de las luces colocadas de manera estratégica dentro de un escenario para la iluminación de obras de teatro, espectáculos musicales, etc. El proceso inicia una vez que se han instalado y hecho las conexiones adecuadas de las luces a los receptores-dimmers, éstos al controlador DMX, y éste último a la PC que contiene el programa para el control de la iluminación. Mediante el programa, el operador podrá controlar directamente las luces durante todo el espectáculo, otra posibilidad que tiene el operador es programar secuencias de encendido-apagado que podrá reproducir (ejecutar) durante el espectáculo.

3.2. *Requerimientos funcionales del programa de control de iluminación*

Las revisiones realizadas a programas comerciales que realizan las mismas funciones dieron el panorama para conocer el proceso del control de iluminación espectacular y así determinar los requerimientos del programa. A continuación se describen los requerimientos funcionales:

El operador “indica” el momento en que desea empezar a controlar las luces.

El operador controla (incrementa/decrementa) la intensidad de luminosidad (función Dimmer) de las luces.

El operador realiza funciones de Fade In (incremento progresivo de la intensidad de las luces) en un intervalo de tiempo seleccionado.

El operador realiza funciones de Fade Out (desvanecimiento progresivo de la intensidad de las luces) en un intervalo de tiempo seleccionado.

El operador establece el número de luces (canales) a controlar.

El operador establece que luces deben sincronizarse ante un cambio en uno de los controles de las mismas.

El operador “programa” secuencias de encendido/apagado de las luces, por intervalos de tiempo.

El operador “reproduce” las secuencias de encendido/apagado de las luces.

El operador “controla” la iluminación mediante el Mouse y/o el teclado.

El programa despliega información tal como los niveles de intensidad por canal, valor del control maestro.

El programa despliega ayuda para el operador.

Requerimientos sobre eventos extraordinarios en el proceso de control de iluminación espectacular

- El programa advierte al cerrarse que las luces serán apagadas.

Requerimientos de mantenimiento del programa de control

- Aumento en el número de funciones de algún accesorio de iluminación que maneje más de un byte de control.
- Creación de una base de datos de accesorios existentes en el mercado.

3.3. *Requerimientos no funcionales del programa de control de iluminación*

- El sistema debe comunicarse con el controlador DMX por medio del PP mediante el protocolo handshake.
- El sistema debe ejecutarse en sistemas operativos de plataforma Microsoft Windows 95, 98, 2000 y NT.

3.4. *Descripción general del sistema*

El sistema controla las diferentes funciones que se requieren en la realización de un espectáculo, realizándolo de la siguiente manera:

- Control de la intensidad de las luces en tiempo real.
- El operador da órdenes de incrementar/decrementar el nivel de intensidad luminosa de cada una de las luces mediante un control asignado a éstas. El sistema se “comunica” con el controlador DMX para indicarle el cambio en el nivel deseado por el operador, además de proporcionarle información en pantalla relativa a los datos DMX.
- Programación de secuencias de encendido/apagado de las luces.

El operador con ayuda de los controles establece una secuencia, en la cual las luces deben encenderse/apagarse cada vez que transcurre un tiempo establecido por medio de otro control. La secuencia establecida por el operador podrá guardarse en archivos para su uso posterior. Cuando el operador abre un archivo de secuencia existente o crea una nueva secuencia, podrá ejecutarla/detenerla/modificarla mediante los controles correspondientes. Al ejecutarse una secuencia, el sistema se “comunicará” con el controlador para indicarle los niveles que deben tomar cada una de las luces. De la misma manera, el sistema proporcionará información en pantalla de los datos DMX que se están transmitiendo en ese momento al controlador.

3.5. *Propósito del sistema*

Controlar la intensidad de las luces conectadas al sistema SciDMX y programar secuencias de encendido/apagado de las mismas.

3.6. *Objetivos del sistema*

El objetivo es contar con una herramienta para el control de la iluminación y una automatización en el control de escenas para espectáculos, obras teatrales, etc. Lo cual incluye:

Control directo de las luces individual y completamente.

Observación rápida de las intensidades de luz proporcionadas en las lámparas.

Programación de escenas.

3.7. *Alcances del sistema*

- Control de la intensidad de las luces.
- Programación y reproducción de secuencias de encendido/apagado de las luces.

3.8. *Implementación*

La arquitectura general del sistema es una arquitectura maestro-esclavo, en donde el operador interactúa con el maestro (programa), el cual envía ordenes al esclavo (controlador DMX), éste las recibe y genera la señal DMX hacia los receptores DMX512.

3.9. *Casos de uso del programa de control*

Para el programa que permite controlar las luces se identificaron 5 casos de uso, englobando en ellos los requisitos funcionales. En la Figura 3.1 se muestra el diagrama de casos de uso para el programa, éste es muy similar al presentado como caso de uso del sistema SciDMX, hay que notar que aquí se identifica un nuevo actor: el controlador DMX, el cual también además del operador interactúa con el programa de control.

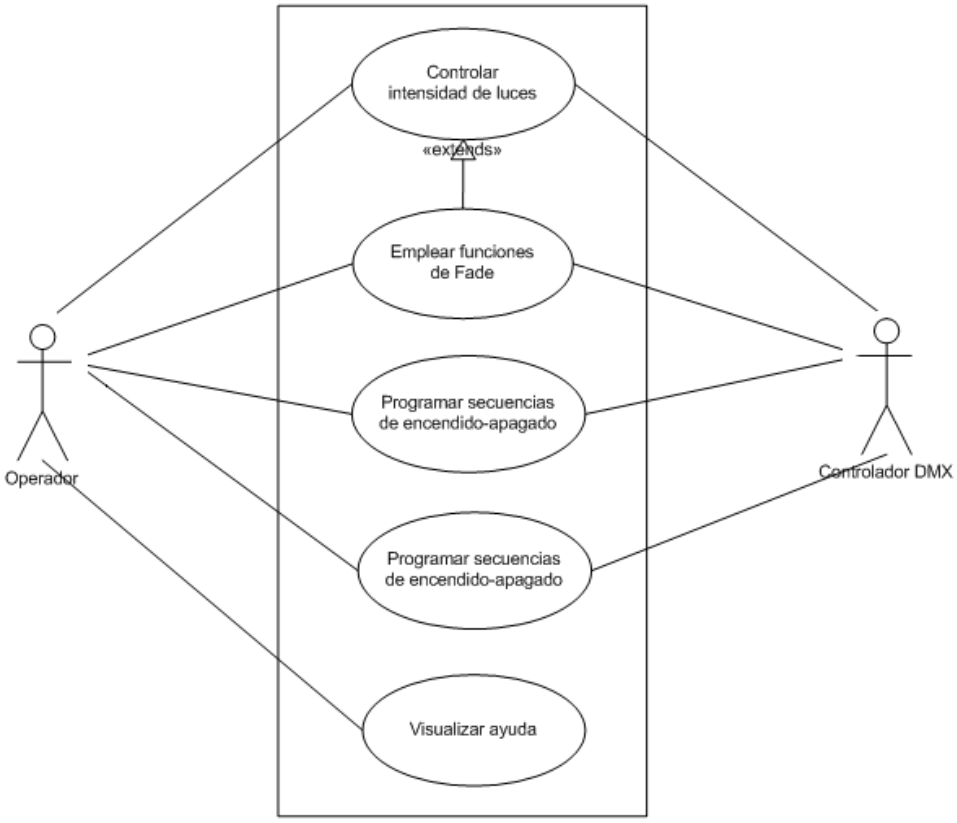

Figura 3.1. Casos de uso del programa de control DMX512.

Para hacer más clara la interacción del sistema con el operador y el controlador DMX, a continuación se presentan el flujo de eventos para cada uno de los 5 casos de uso: Controlar la intensidad de las luces, Emplear funciones de Fade, Programar secuencias de encendido-apagado, Reproducir secuencias de encendido-apagado y Visualizar ayuda.

Flujo de eventos para el caso de uso: Controlar la intensidad de las luces

Precondiciones

Antes de iniciar este caso de uso, el operador debe estar visualizando la pantalla principal, donde puede acceder a los controles mostrados en la Tabla 3.1:

Tabla 3.1. Controles para regular la intensidad de las luces.

Control	**Función**
N barras de desplazamiento (dependiendo del número de canales seleccionado).	Cada barra de desplazamiento regula la intensidad de la luz correspondiente, en función de la posición del indicador gráfico de valor (botón deslizante). Así, sí el operador realiza cambios en cualquiera de las barras de desplazamiento de los dimmers, la actividad seleccionada es DIMMER.
Control maestro (barra de desplazamiento)	Establece la intensidad (0-100%) máxima que las luces pueden alcanzar. Si el operador establece el nivel del control maestro a 45%, y posteriormente establece la intensidad de alguna de las luces con la barra de desplazamiento hasta su valor máximo, ésta ahora será proporcional al valor del control maestro; es decir ya no será el 100 % de su intensidad, sino el 45%. Ésta función es útil cuando se desea limitar la intensidad máxima de las luces. Si el operador realiza cambios en este control, la actividad seleccionada es CM.
N casillas de verificación de sincronía (Sinc), (dependiendo del número de canales seleccionado).	Al habilitar las casillas de verificación en al menos dos luces, las actividades que se realicen subsecuentemente (dimmer, encendido-apagado, golpe) serán sincronizadas. Por ejemplo, si la luces L1 y L2 son sincronizadas, cuando se realice un incremento de intensidad, las luces sincronizadas aumentarán en la misma razón, a partir de su valor actual. Este botón no tiene efecto directo sobre la o las luces, pero si en las actividades subsecuentes. Así al presionar el botón Sinc, la actividad seleccionada es SINC.
N botones *Golpe*, (dependiendo del número de canales seleccionado).	El sistema enciende la luz correspondiente o las luces que estén sincronizadas, también puede mantenerla encendidas mientras el operador mantiene presionado el botón *Golpe*. En este caso la actividad seleccionada es GOLPE.
Botón Salir	Antes de cerrar la aplicación, las luces son apagadas. En este caso la actividad seleccionada es SALIR.

Flujo principal

Este caso de uso inicia cuando el operador realiza algún cambio en uno de los controles anteriores (actividad deseada):

Operador	Sistema
	1. Espera a que el operador realice algún cambio por medio del mouse o el teclado, en alguno de los controles anteriores.
2. Realiza un cambio en los controles.	
	3. Si la actividad seleccionada es DIMMER (el operador realizó cambios en cualquiera de las barras de desplazamiento de los dimmers) se lleva a cabo el subflujo C1: *Controlar intensidad de las luces*. Si la actividad seleccionada es CM (el operador modificó el control maestro) se lleva a cabo el subflujo C2: *Cambiar nivel de control maestro*. Si la actividad seleccionada es SINC (el operador desea sincronizar los cambios subsecuentes en los controles de las luces habilitadas con *Sinc*) se lleva a cabo el subflujo C3: *Sinc*. Si la actividad seleccionada es GOLPE (el operador desea encender de golpe una o más luces (botón Sinc) o mantenerlas encendidas mientras mantiene presionado el botón) se lleva a cabo el subflujo C4: *Golpe*. Si la actividad seleccionada es SALIR (el operador desea abandonar la aplicación) se lleva a cabo el subflujo C5: *Salir de aplicación*.

Subflujos

C1 – Controlar la intensidad de las luces

Operador	Sistema
	1. Determina la nueva posición de la barra de desplazamiento que el operador movió, para determinar el valor del dato DMX en base al nivel del control maestro. 2. Para el dimmer correspondiente despliega en pantalla: El nuevo tono del led indicador. El tono del led está en función del dato DMX. El nuevo valor del dato DMX. 3. Envía los datos DMX al controlador DMX, en base al subflujo C6.
4. Visualiza en pantalla los cambios del dimmer correspondiente: el tono del led, el nuevo valor del dato DMX que se envió y el cambio de la luz correspondiente.	
	5. El caso de uso inicia otra vez en el paso 1 del flujo principal.

C2 – Cambiar nivel de control maestro

Operador	Sistema
	1. Determina la nueva posición del control maestro que el operador movió, y calcula los valores de los datos DMX para todas las luces. 2. Para cada dimmer despliega en pantalla: Los nuevos tonos de los led´s indicadores. El tono de cada led está en función del dato DMX correspondiente. Los nuevos valores de los datos DMX. 3. Envía los datos DMX al controlador DMX, en base al subflujo C6.
4. Visualiza en pantalla los cambios de todos	

los dimmers, el tono de los led´s, los nuevos valores de los datos DMX que se enviaron y el cambio en todas las luces.	5. El caso de uso inicia otra vez en el paso 1 del flujo principal.

C3 – Sinc

Operador	Sistema
	1. Establece una bandera de sincronía para la luz correspondiente. 2. El caso de uso inicia otra vez en el paso 1 del flujo principal.

C4 – Golpe

Operador	Sistema
3. Visualiza el encendido de la luz o las luces sincronizadas.	1. Establece el valor del dato DMX de la luz correspondiente a 255. Si al menos dos luces están sincronizadas, los datos DMX correspondientes son establecidos a 255. 2. Envía los datos DMX al controlador DMX, en base al subflujo C6. 4. El caso de uso inicia otra vez en el paso 1 del flujo principal.

C5 – Salir de la aplicación

Operador	Sistema
3. Visualiza el apagado de todas las luces.	1. Establece los valores de los datos DMX a 0. 2. Envía los datos DMX al controlador DMX, en base al subflujo C6. 4. El caso de uso termina.

C6 – Envío de los datos DMX al controlador DMX

Operador	Sistema
	El envío de los datos al controador DMX512 se lleva a cabo en dos pasos, primero se escribe la dirección del dato a enviar, y después se escribe el dato, para esto se realiza lo siguiente:

	Es importante mencionar que al escribir al registro de dirección o de datos del puerto paralelo, el propio HW genera automáticamente las señales del protocolo handshake. Escritura de la dirección: 1. El programa escribe al registro de dirección (base+3). 2. El host (PC) espera una señal de reconocimiento por un flanco de subida de Wait, en caso de que no recibirla se vuelve a intentar nuevamente con el paso 1, en otro caso continúa. Escritura del dato: 3. El programa escribe al registro de datos (base+4). 4. El host (PC) espera una señal de reconocimiento por un flanco de subida de Wait, en caso de no recibirla se vuelve a intentar nuevamente con el paso 3, en otro caso continúa. 5. Regresa el control al subflujo que lo llamo.

Flujo de eventos para el caso de uso: Emplear funciones de fade

Precondiciones

Antes de iniciar este caso de uso, el operador debe estar visualizando la pantalla principal, donde puede acceder a los controles mostrados en la Tabla 3.2:

Tabla 3.2. Botones para las funciones fade in y fade out.

Botón	**Función**
Fade in	Permite realizar el encendido progresivo de todas las luces en un tiempo de elección. Si el operador presiona el botón *fade in*, la actividad seleccionada es IN.
Fade out	Permite realizar el apagado progresivo de todas las luces en un tiempo de elección. Si el operador presiona el botón *fade out*, la actividad seleccionada es OUT.
Lista desplegable	Permite establecer el lapso de tiempo en el que se realiza la función Fade (in, out).
Todas apagadas	Permite realizar el apagado inmediato de todas las luces. Si el operador presiona el botón *todas apagadas*, la actividad seleccionada es APAGADAS.
Todas encendidas	Permite realizar el encendido inmediato de todas las luces. Si el operador presiona el botón *todas encendidas*, la actividad seleccionada es ENCENDIDAS.

El tiempo que tarde en completarse la función Fade (in, out) está determinada por el valor seccionado de la lista desplegable, la cual el operador puede cambiar antes de presionar uno de los botones de Fade. El valor de la lista desplegable por omisión es de 3 segundos.

La ejecución de la función *fade in* se realiza de la siguiente manera: Suponiendo que existen 3 luces: L1, L2 y L3, si de éstas tres al menos una de ellas, por ejemplo: L1 tiene una intensidad media y las luces L2 y L3 están apagadas, cuando se ejecute la función *fade in* las tres luces encenderán progresivamente hasta tener su máxima intensidad, L1 desde la media intensidad, y las luces L2 y L3 desde estar apagadas hasta encender a su máxima intensidad. La Figura 2.2 (línea con círculos), muestra el cambio que sufren las luces L2 y L3 en un tiempo de *fade in* de 10 segundos.

De manera similar, la ejecución de la función *fade out* es como sigue: Teniendo ahora las 3 luces de la siguiente forma: L1 sigue teniendo una intensidad media, en cambio las luces L2 y L3 ahora están encendidas, cuando se ejecute la función *fade out* éstas se apagarán progresivamente hasta apagarse por completo, L1 desde la media intensidad que tiene, y las luces L2 y L3 desde su máxima intensidad hasta apagarse. La Figura 3.2 (línea continua), muestra el cambio que sufren las luces L2 y L3 en un tiempo de *fade out* de 3 segundos.

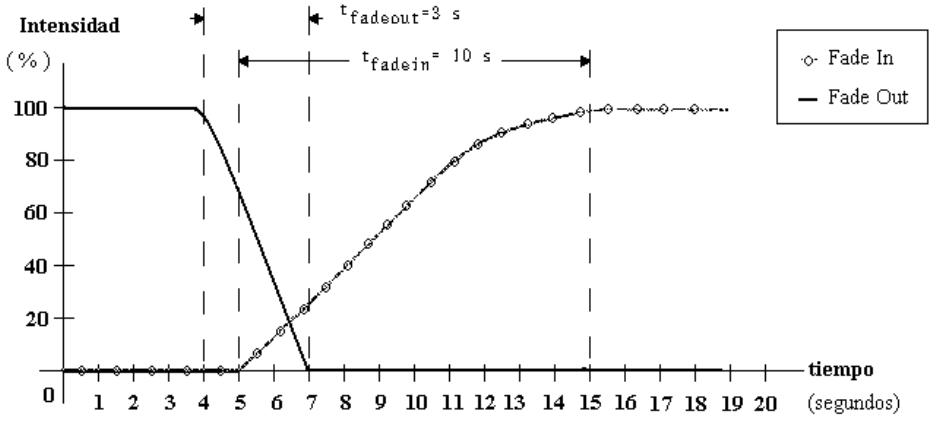

Figura 3.2. Esquema de las funciones de *fade in* y *fade out*.

Si el operador desea establecer el tiempo de fade, debe seleccionar un valor de tiempo en segundos de la lista desplegable, así la actividad seleccionada es ESTIEMPO, por lo que se lleva a cabo el subflujo F1: *Estiempo*.

Flujo principal

Este caso de uso inicia cuando el operador presiona en uno de los botones anteriores.

Operador	Sistema
	1. Espera a que el operador seleccione una de las actividades, haciendo clic con el mouse en alguno de los botones de la Tabla 4.2.
2. Selecciona una opción por medio de uno de los controles.	3. Evalúa que actividad se ha seleccionado: Si la actividad seleccionada es IN se lleva a cabo el subflujo F1: Fade in. Si la actividad seleccionada es OUT se lleva a cabo el subflujo F2: Fade out. Si la actividad seleccionada es APAGADAS se lleva a cabo el subflujo F3: Todas apagadas. Si la actividad seleccionada es ENCENDIDAS se lleva a cabo el subflujo F4: Todas encendidas.

Subflujos

F1 – Fade In

Operador	Sistema
1. Cambia el tiempo de fade, por medio de la lista de valores posibles o deja el valor establecido por default (3 segundos)	
	2. Abre una nueva ventana titulada Fade en proceso, con un botón Cancelar, que permite detener en algún momento la función fade. Si el operador cancela la función fade, el nivel de intensidad de las luces se mantiene en el que se tenía antes de cancelar. 3. Determina el tiempo de incremento ti, en base al tiempo de fade tf (ti = tf /255) elegido o toma por default los 3 segundos, inicializa el tiempo del reloj de interrupciones continuas y lo habilita para que se generen interrupciones cada ti segundos, así en cada interrupción se realiza lo siguiente: 4. Inicializa el dato DMX de los 3 dimmers a 0. 5. Para cada dimmer despliega en pantalla: Los tonos colores de los led´s indicadores (el tono de cada led está en función de cada dato DMX). Los nuevos valores de los datos DMX. 6. Envía los datos DMX al CONTROLADOR DMX, en base al subflujo F5.
7. Visualiza en pantalla los cambios de los tonos de los led's, los nuevos valores de los datos DMX que se envían y el encendido progresivo de las luces.	
	8. Incrementa en uno el dato DMX de cada uno de los tres dimmers. 9. (Condición: if anidado) Si el dato DMX no ha llegado a 255 y el usuario no ha presionado el botón Cancelar, entonces: [Si el dato DMX no ha llegado a 255 volver al punto 5. Si el usuario presionó el botón Cancelar, cierra la

	ventana “Fade Activo”, mantiene la última intensidad de las luces y continua en el paso 11], en otro caso continua con el punto 10. 10. Cierra la ventana Fade en proceso y emite un sonido de finalización. 11. El caso de uso inicia otra vez en el paso 1 del flujo principal.

F2 – Fade Out

Operador	Sistema
1. Cambia el tiempo de fade, por medio de la lista de valores posibles o deja el valor establecido por default (3 segundos). 7. Visualiza en pantalla los cambios de: los tonos de los led’s, los valores de los datos DMX que se envían y el apagado progresivo de las luces.	2. Abre una nueva ventana titulada Fade en proceso, con un botón Cancelar, que permite detener en algún momento la función fade. Si el operador cancela la función fade, el nivel de intensidad de las luces se mantiene en el que se tenía antes de cancelar. 3. Determina el tiempo de incremento ti, en base al tiempo de fade tf (ti = tf /255) elegido o toma por default los 3 segundos, inicializa el tiempo del reloj de interrupciones continuas y lo habilita para que se generen interrupciones cada ti segundos, así en cada interrupción se realiza lo siguiente 4. Inicializa el dato DMX de los 3 dimmers a 255. 5. Para cada dimmer despliega en pantalla: Los nuevos tonos de los led´s indicadores. El color de cada led está en función del dato DMX. Los nuevos valores de los datos DMX. 6. Envía los datos DMX al CONTROLADOR DMX, en base al subflujo F5.

	8. Decrementa en uno el dato DMX de cada uno de los tres dimmers. 9. (Condición: if anidado) Si el dato DMX no ha llegado a 0 y el usuario no ha presionado el botón Cancelar, entonces: [Si el dato DMX no ha llegado a 0 volver al punto 5. Si el usuario presionó el botón Cancelar, cierra la ventana "Fade Activo", mantiene la última intensidad de las luces y continua en el paso 11], en otro caso continua con el punto 10. 10. Cierra la ventana Fade en proceso y emite un sonido de finalización. 11. El caso de uso inicia otra vez en el paso 1 del flujo principal.

F3 – Todas apagadas

Operador	Sistema
3. Visualiza en pantalla los cambios de los dimmers: el tono de los led´s, el valor del dato DMX actual de cada dimmer y el apagado de las luces.	1. Establece el valor del dato DMX de los 3 dimmers a 0. 2. Envía los datos DMX al CONTROLADOR DMX, en base al subflujo F5. 4. El caso de uso inicia otra vez en el paso 1 del flujo principal.

F4 – Todas encendidas

Operador	Sistema
3. Visualiza en pantalla los cambios de los dimmers: el tono de los led´s, el valor del dato DMX actual de cada dimmer y el encendido de las luces.	1. Establece el valor del dato DMX de los 3 dimmers a 255 (2n-1, n = 8 bits). 2. Envía los datos DMX al controlador DMX, en base al subflujo F5. 4. El caso de uso inicia otra vez en el paso 1 del flujo principal.

F5 – Envío de los datos DMX al controlador DMX

Operador	Sistema
	El envío de los datos se lleva a cabo en dos pasos, primero se escribe la dirección del dato a enviar, y después se escribe el dato en sí, para esto se realiza lo siguiente: Es importante mencionar que en el puerto paralelo en el modo EPP, al escribir al registro de dirección o de datos, el HW del PP genera automáticamente las señales del handshake. Escritura de la dirección: 1. El programa escribe al registro de dirección (base+3). 2. El host (PC) espera una señal de reconocimiento por un flanco de subida de Wait, en caso de que no recibirla se vuelve a intentar nuevamente con el paso 1, en otro caso continúa. Escritura del dato: 3. El programa escribe al registro de datos (base+4). 4. El host (PC) espera una señal de reconocimiento por un flanco de subida de Wait, en caso de no recibirla se vuelve a intentar nuevamente con el paso 3, en otro caso continúa. 5. Regresa el control al subflujo que lo llamo.

Flujo de eventos para el caso de uso: Programar secuencias de encendido-apagado

Precondiciones

Antes de iniciar este caso de uso, el operador debió haber hecho click sobre el botón *Programar secuencias de encendido-apagado*, lo cual hace que se visualice una nueva ventana para la programación, posteriormente el operador debe estar visualizando la pantalla (ventana *programar secuencias*) donde puede acceder a los lista de controles mostrados en la Tabla 3.3:

Tabla 3.3. Botones disponibles para la programación de secuencias de encendido-apagado

Botón	**Función**
5 filas de 12 casillas (una por canal) de verificación cada una.	Cada casilla permite establecer si la luz correspondiente será encendida (casilla activada) o apagada. Si el operador activa una de las casillas, la actividad seleccionada es EC.
5 perillas de control de tiempo.	Cada perilla permite establecer el tiempo en que la luces correspondientes estarán encendidas/apagadas. Si el operador presiona y gira una de las perillas, la actividad seleccionada es EP.
Botón Abrir secuencia.	Abre un archivo (secuencia), para su modificación, y/o reproducción.
Botón Guardar secuencia.	Guarda en un archivo, la creación o modificación de una secuencia.
Botón Limpiar secuencia.	Desactiva todas las casillas de verificación y establecer a cero las perillas de control de tiempo. Lo anterior permite establecer una nueva secuencia desde el principio. Si el operador presiona el botón, la actividad seleccionada es LS.
Botón Reproducir secuencia.	Reproduce la secuencia abierta, interpretando las casillas de verificación y los tiempos establecidos con las perillas de tiempo. Una vez que el botón *Reproducir secuencia* es presionado, el nombre del botón cambia a *Detener secuencia,* el cual como su nombre indica detiene la reproducción de la secuencia, dejando a las luces en el último estado que se presento al presionar el botón. Si el operador presiona el botón, la actividad seleccionada es RS.
Botones de opción: secuencia cíclica y n veces.	Ambos botones permiten seleccionar entre reproducción de secuencia cíclica o reproducción por un número de veces. La opción por omisión es secuencia cíclica, sí el operador cambia de opción, la actividad seleccionada es TS.
Cuadro de texto *Número de veces*.	Si la opción de reproducción por un número de veces fue elegida, el cuadro de texto es habilitado, éste último permite establecer el número de veces que será reproducida la secuencia. Si el operador escribe en el cuadro de texto un número, la actividad seleccionada es RS.

Flujo principal

Este caso de uso inicia cuando el operador realiza algún cambio en uno de los controles anteriores (actividad deseada):

Operador	Sistema
2. Realiza un cambio en los controles.	1. Espera a que el operador realice algún cambio por medio del Mouse o el teclado, en alguno de los controles anteriores. 3. Si la actividad seleccionada es EC (el operador activó algunas de las casillas de verificación) se lleva a cabo el subflujo S1: Establecer encendido/apagado de la luz. Si la actividad seleccionada es EP (el operador modificó una de las perillas de tiempo) se lleva a cabo el subflujo S2: Establecer tiempo de paso. Si la actividad seleccionada es LS (el operador desea limpiar la secuencia) se lleva a cabo el subflujo S3: Limpiar secuencia. Si la actividad seleccionada es RS (el operador desea reproducir la secuencia abierta) se lleva a cabo el subflujo S4: reproducir secuencia. Si la actividad seleccionada es TS (el operador cambió la opción del modo de secuencia) se lleva a cabo el subflujo S5: cambiar modo de secuencia.

Subflujos

S1 – Establecer encendido/apagado de la luz

Operador	Sistema
	1. Revisa en que i-esimo renglón (paso) y en que j-esima columna (canal) está la casilla que fue activada para establecer el valor de cij a 255. 2. El caso de uso inicia otra vez en el paso 1 del flujo principal.

S2 –Establecer tiempo de paso

Operador	Sistema
.	1. Revisa en que renglón (enésimo paso) está la perilla de control de tiempo que se modificó y establece el valor del enésimo ti igual al valor que tiene la perilla. 2. El caso de uso inicia otra vez en el paso 1 del flujo principal.

S3 – Limpiar secuencia

Operador	Sistema
	1. Establece todos los elementos de la matriz de configuración a 0 y las perillas de control de tiempo a 0. 2. El caso de uso inicia otra vez en el paso 1 del flujo principal.

S4 – Reproducir secuencia.

Este caso de uso contempla que el operador dejó el modo de reproducción en *reproducción cíclica*.

Operador	Sistema
4. Visualiza el encendido y el apagado de las luces por el tiempo establecido. *. (Después de varias repeticiones de la secuencia) Presiona el botón Detener secuencia.	1. Establece el valor del contador de pasos i a 1. 2. Lee el i-ésimo renglón de la matriz de configuración, obteniendo los 12 valores para los 12 canales DMX. 2. Envía los datos DMX al controlador DMX, en base al subflujo F5. 3. Espera el tiempo establecido por la i-ésima perilla de control e incrementa el valor de i en uno. 5. (Condición: if anidado) Si el operador no ha presionado el botón Detener secuencia entonces [Si i es menor o igual a 10 volver al punto 2, en caso contrario volver al paso 1], en otro caso continua con el punto 6. 6. El caso de uso inicia otra vez en el paso 1 del flujo principal.

S5 – Cambiar modo de secuencia

Operador	Sistema
	1. Establece el modo de reproducción al seleccionado por el usuario. Si el modo seleccionado es reproducción por un número de veces, el cuadro de texto Número de veces es habilitado y el caso de uso continua con el paso 2, en caso contrario se sigue con el paso 4.
2. Establece el número de veces que desea reproducir la secuencia de los pasos 1 al 10.	
	3. Valida el dato introducido por el usuario, si este no es un número mayor a cero, muestra el mensaje "Introducir solo números". Si el dato es un número, se establece el número de veces a reproducirse la secuencia.
	4. El caso de uso inicia otra vez en el paso 1 del flujo principal.

3.10. *Implementación del programa de control*

En la etapa de implementación, se llevó a cabo la codificación del sistema que controla el proceso de control de iluminación espectacular. El sistema se probó en el sistema operativo Windows XP y Windows 98.

El sistema se desarrolló en Visual Basic (VB) versión 6, por las siguientes razones:

- VB es una herramienta de desarrollo de aplicaciones diseñada específicamente para la familia de sistemas operativos de Microsoft Windows. VB proporciona un ambiente flexible y poderoso, permitiendo el desarrollo rápido de aplicaciones [Ramírez, 2001].
- VB emplea el uso de objetos como un componente de software, el cual encapsula sus propiedades y métodos relacionados en una simple unidad reutilizable. Similar al módulo de código en aprovechamiento tradicional para programar, solo múltiples instancias de objetos pueden ser creadas y manipuladas en nuevas formas.

Es importante notar que VB, no sigue el modelo tradicional de iniciar la ejecución de la primera línea del programa y continúa secuencialmente hasta la última. En vez de esto, el código del programa esta dividido en procedimientos, los cuales son ejecutados en respuesta a una conjunto de eventos que son recibidos por el programa. Los eventos son generados como un resultado de varios tipos de interacción con el usuario y otras acciones del sistema. El orden en el cual los procedimientos son ejecutados corresponde al orden en el cual la aplicación recibe esos eventos. Si no se reciben eventos, no se ejecuta nada. Los eventos que ocurren son cada uno asociados y manejados por un objeto particular. Cuando un objeto de VB recibe un evento, este ejecuta una pieza asociada de código llamada procedimiento de evento [Russo et al, 1999].

Cabe señalar que los requerimientos del programa de control son mínimos, por mencionar algunos:

- Programación orientada a eventos.
- Eventos controlados por tiempo, temporizador.
- Interfaz visual.
- Generación de un programa ejecutable, basado en la compilación y no en la interpretación del código como se realiza en Java.

El programa pudo haberse desarrollado en un lenguaje de programación como Builder C++, Visual C++, Java; cada uno de éstos tan adecuado como Visual Basic versión 6. Sin embargo para nuestros propósitos; lo escogimos por la familiaridad y experiencia que se tiene con dicho lenguaje.

3.10.1. *Descripción funcional del sistema*

El sistema que realiza el proceso de controlar la iluminación espectacular, se desarrolló a través de una aplicación que se ejecuta sobre la plataforma de Microsoft Windows.

Para explicar a mayor detalle el sistema, en las siguientes secciones se presentan las dos funciones que lo componen: Controlar la intensidad de las luces y Programar secuencias de encendido-apagado.

3.10.1.1. *Ventana principal – Controlar la intensidad de las luces*

La ventana principal presentada en la Figura 3.3, como su nombre lo indica, es la primera página que el operador ve al ejecutar el programa, y ésta es la que esencialmente permite controlar la intensidad de las luces. Del lado derecho se tiene una lista desplegable que permite seleccionar el número de canales a controlar, así como dos botones de control para las funciones: Fade In y Fade Out, junto a estos botones se encuentra una lista desplegable que contiene intervalos de tiempo predefinidos a escoger para las funciones de Fade. Del lado izquierdo se aprecian trece barras de desplazamiento, las doce primeras son para cada una de las luces (canales) y la última de la más a la derecha es la barra de desplazamiento del control maestro. Cada una de las doce barras de desplazamiento tiene asociado un número de canal del 1 al 12, así como un indicador en forma de LED (Diodo Emisor de Luz) rectangular y una caja de selección para la función de sincronía. En la parte media derecha de la ventana, se muestran los botones para *Iniciar/Detener* el proceso de control, para *Programar secuencias* y para *Salir*.

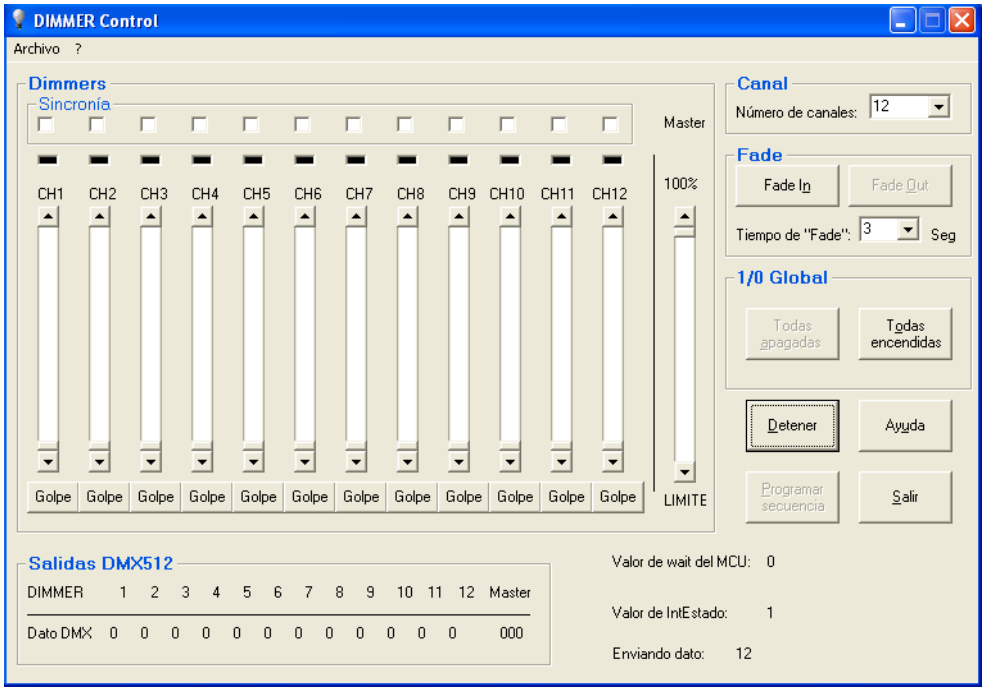

Figura 3.3 Ventana principal del programa de control de iluminación.

Con los controles anteriores, una vez que el operador presiona el botón Iniciar él tiene el control de todas las luces conectadas al sistema. Solo basta mover la barra de desplazamiento adecuada con el Mouse, para controlar la intensidad de una de las lámparas.

3.10.1.2. *Ventana secundaria: Programar secuencias de encendido-apagado*

La segunda ventana mostrada en la Figura 3.4, es desplegada cuando el operador presiona el botón *Programar secuencias de encendido-apagado*. Los controles en esta ventana son los siguientes:

- Casillas de verificación para activar/desactivar el encendido/apagado de cada una de las luces.
- Perillas de control de tiempo, para establecer el tiempo que durará un paso de la secuencia.
- Botón para guardar la secuencia en archivo.
- Botón para abrir una secuencia existente.
- Botón para Limpiar (desactivar) las casillas de verificación y establecer a cero las perillas de control de tiempo.
- Botón para Reproducir la secuencia.
- Dos botones de opción para establecer entre secuencia cíclica (opción por Default) o ejecución de un número determinado de veces. Si el operador escoge ésta última, se habilita una caja de texto donde podrá escribir el número de veces a reproducir.

Los controles anteriores permiten al operador programar secuencias de encendido-apagado de las luces en intervalos de tiempo. Es decir, para cada paso (del 1 al 5, p1 a p5) existe una fila de casillas de verificación (CV), una para cada canal (Figura 3.4). Si para el paso 1 (p1), la CV correspondiente al canal 1 se habilita, entonces se ha establecido que la luz conectada al canal 1 debe encender el intervalo de tiempo establecido por la perilla de control de tiempo (CT1). Quizás en el paso 2 (p2), el operador establezca que el canal 1 debe apagarse, y ahora deban encenderse las luces conectadas a los canales 2 y 3

durante 20 segundos establecidos por CT2. De manera similar el operador puede establecer los pasos restantes (p3-p5).

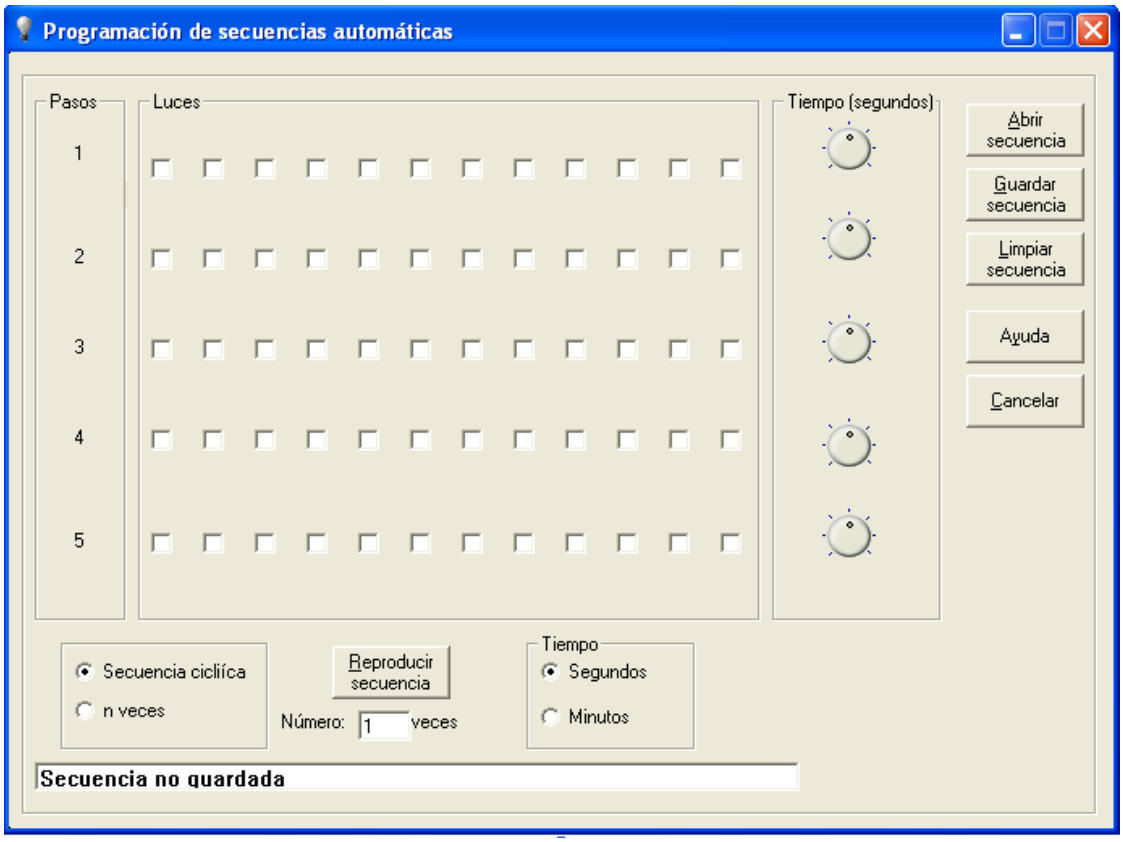

Figura 3.4. Ventana para la programación de secuencias.

Al presionar el botón Reproducir secuencia, el programa interpreta que luces conectadas al canal correspondiente deben encenderse y por cuanto tiempo, una vez terminado este tiempo ejecuta p2, y así sucesivamente, hasta llegar a p5 volviendo a ejecutar p1. Así, ésta secuencia se ejecuta el número de veces establecido en el cuadro de texto (Número) o hasta que el botón *Detener* es presionado. Con esto, el operador tiene la posibilidad de reproducir el encendido-apagado de una secuencia de 5 combinaciones, con la opción de establecer el tiempo entre un paso y otro.

Cada secuencia que el operador cree podrá guardarla en un archivo, la cual posteriormente podrá abrir para modificarla y/o reproducirla con ayuda de los botones *Guardar secuencia*, *Abrir secuencia* y *reproducir secuencias* respectivamente.

3.11. *Pruebas de funcionalidad*

Las pruebas que se realizaron al programa fueron de funcionalidad. Éstas se describen en la siguiente sección.

Se realizaron una serie de pruebas que determinaron el buen funcionamiento del mismo. A continuación se describen los casos de prueba más importantes.

Caso de Prueba 1: Controlar la intensidad de las luces.
Propósito: Verificar que las luces respondan al cambio que solicita el operador.
Datos de prueba: Establecimiento de la intensidad deseada en las luces.

No.	Paso	Dato generado
1	El operador hace click sobre el botón Iniciar.	El sistema cambia el nombre del botón Iniciar a Detener, habilita las barras de desplazamiento del número de canales a controlar, la del control maestro y los botones para la función Fade.
2	El operador desliza hacia arriba la barra de desplazamiento vertical (luz del canal 1).	El sistema despliega el nuevo valor del dato DMX, presenta un aumento en el tono del LED de color verde, proporcional al cambio realizado. Finalmente la luz incrementa su intensidad luminosa.
3	El operador hace click sobre las casillas de verificación (activándolas) de sincronía de las luces de los canales 2 y 3.	
4	El operador mantiene presionado el botón Golpe de la luz del canal 2.	El sistema despliega la posición del cursor de las barras de desplazamiento de las luces (canales 2 y 3) en su máximo valor, el valor del dato DMX en 255 (máximo valor), el tono del LED verde en su mayor intensidad y las luces de los canales 2 y 3 encienden a su máxima intensidad.
5	El operador suelta el botón Golpe de la luz del canal 2.	El sistema despliega el cursor de las barras de desplazamiento a la posición que tenia antes de que fuera presionado el botón Golpe. Tanto el valor del dato DMX, el tono del LED y las luces de los canales 2 y 3 vuelven a tener el mismo valor que tenían antes del paso 4.
6	El operador hace click sobre el botón Detener.	El sistema cambia el nombre del botón Detener a Iniciar, deshabilita las barras de desplazamiento de los doce canales, la del control maestro y los botones para la función Fade. Finalmente las luces son apagadas.

Caso de Prueba 2: Realizar funciones de Fade.
Propósito: Verificar que las luces respondan a la función fade que solicita el operador.
Datos de prueba: Establecimiento de las funciones Fade In y Fade Out.

No.	Paso	Dato generado
1	El operador hace click sobre el botón Iniciar.	El sistema cambia el nombre del botón Iniciar a Detener, habilita las barras de desplazamiento del número de canales a controlar, la del control maestro y los botones para la función Fade.
2	El operador selecciona de la lista desplegable "Tiempo de Fade" el valor de 5 segundos.	
3	El operador hace click sobre el botón Fade In.	El sistema despliega una nueva ventana con la leyenda Fade Activo y un botón Cancelar. Mientras tanto las posiciones de los doce cursores de las barras de desplazamiento van subiendo proporcionalmente al tiempo de fade seleccionado, al mismo tiempo los valores de los datos DMX desplegados van incrementando su valor de 0 a 255, el tono de los LEDs verdes va aumentando y las luces de todos los canales van encendiendo progresivamente a su máxima intensidad. Al llegar el dato DMX a su máximo valor, el sistema emite un sonido de finalización.
4	El operador visualiza en pantalla: el aumento del tono de los led's, los valores de los datos DMX que se envían y el encendido progresivo de las luces.	
5	El operador hace click sobre el botón Fade Out. (Recordar que el tiempo de Fade está en 5 segundos).	El sistema despliega una nueva ventana con la leyenda Fade Activo y un botón Cancelar. Mientras tanto las posiciones de los doce cursores de las barras de desplazamiento van bajando proporcionalmente al tiempo de fade seleccionado, al mismo tiempo los valores de los datos DMX desplegados van decrementando su valor de 255 a 0, el tono de los LEDs verdes va disminuyendo y las luces de todos los canales van apagandose progresivamente. Al llegar el dato DMX a cero, el sistema emite un sonido de finalización.
6	El operador hace click sobre el botón Detener.	El sistema cambia el nombre del botón Detener a Iniciar, deshabilita las barras de desplazamiento de los canales, la del control maestro y los botones para la función Fade. Finalmente las luces son apagadas.

Caso de Prueba 3	Programar y reproducir secuencias.
Propósito	Verificar que la secuencia se almacene en archivo y se ejecute correctamente.
Datos de prueba	Archivo de secuencia y visualización de la escena reproducida.

No.	Paso	Dato generado
1	El operador hace click sobre el botón Programar secuencia.	El sistema despliega una nueva ventana llamada Programar secuencias de encendido-apagado.
2	El operador hace click habilitando las casillas de verificación (paso 1) de las luces de los canales 1, 3, 5 y 7.	
3	El operador mantiene presionada y gira la perilla de control de tiempo del paso 1, para establecer el tiempo a 10 segundos.	
4	El operador hace click habilitando las casillas de verificación (paso 2) de las luces de los canales 2, 4, 6 y 8.	
5	El operador mantiene presionada y gira la perilla de control de tiempo del paso 2, para establecer el tiempo a 5 segundos.	
6	El operador hace click habilitando las casillas de verificación (paso 3 y 5) de las luces de los canales 1, 3, 5 y 7.	
7	El operador mantiene presionada y gira la perilla de control de tiempo del paso 3 y 5, para establecer el tiempo a 5 segundos.	
8	El operador hace click habilitando las casillas de verificación (paso 4) de las luces de los canales 2, 4, 6 y 8.	
9	El operador mantiene presionada y gira la perilla de control de tiempo del paso 4, para establecer el tiempo a 5 segundos.	
10	El operador hace click sobre el botón Guardar.	El sistema despliega la ventana de guardar archivo.
11	El operador escribe un Nombre de Archivo y hace click en Guardar.	El sistema guarda el archivo con el nombre proporcionado y cierra la ventana de guardar archivo.
12	El operador hace click sobre el botón Reproducir secuencia.	El sistema despliega el mensaje: “Los niveles de intensidad actuales se perderán”.
13		El sistema enciende las luces 1, 3, 5, 7 y espera a que transcurran 10 segundos (paso 1).
14	El operador observa que las luces 1, 3, 5 y 7 encienden durante 10 segundos.	
15		El sistema enciende las luces 2, 4, 6, 8, apaga las luces 1, 3, 5, 7 y espera a que transcurran 5 segundos (paso 2).
16	El operador observa que las luces 2, 4,	

	6 y 8 encienden, y que las luces 1, 3, 5 y 7 se apagan durante 5 segundos.	
17		El sistema enciende las luces 1, 3, 5, 7 y espera a que transcurran 5 segundos (paso 3).
18	El operador observa que las luces 1, 3, 5 y 7 encienden, y que las luces 2, 4, 6 y 8 se apagan durante 5 segundos.	
19		El sistema enciende las luces 2, 4, 6, 8, apaga las luces 1, 3, 5 y 7 y espera a que transcurran 5 segundos (paso 4).
20	El operador observa que las luces 2, 4, 6 y 8 encienden y que las luces 1, 3, 5 y 7 se apagan durante 5 segundos.	
21		El sistema enciende las luces 1, 3, 5, 7, apaga las luces 2, 4, 6, 8 y espera a que transcurran 5 segundos (paso 5).
22	El operador observa que las luces 1, 3, 5 y 7 encienden y que las luces 2, 4, 6 y 8 se apagan durante 5 segundos.	
23		El sistema enciende las luces 1, 3, 5, 7 y espera a que transcurran 10 segundos (paso 1, 2ª. vez).
24	El operador observa que las luces 1, 3, 5 y 7 encienden durante 10 segundos.	
25	El operador hace click sobre el botón Cerrar programación de secuencias.	El sistema despliega la ventana principal.

Capítulo 4. Desarrollo del receptor y dimmer

El presente capítulo continúa con el desarrollo de SciDMX siguiendo con la metodología de desarrollo de sistemas empotrados. El capítulo presenta las fases 2 a la 5 correspondientes al receptor y dimmer.

4.1. *Fase 2: Particionado HW y SW del receptor DMX*

Para la fase 2 correspondiente al receptor DMX, al igual que en el controlador DMX se utiliza un diagrama de casos de uso, para definir sus requerimientos e identificar los actores que interactúan con el dimmer. El diagrama es mostrado en la Figura 4.1.

Los casos de uso identificados para el receptor DMX son cuatro: “Decodificar trama DMX512”, “Controlar intensidad”, “Detectar cruce por cero de la señal alterna” y “Leer dirección DMX (canal) asignada”. Como actores se han identificado tres, estos son: “Controlador DMX”, circuito “Detector de cruce por cero”, “Dip-switch” (asigna la dirección DMX del receptor) y “Luces”.

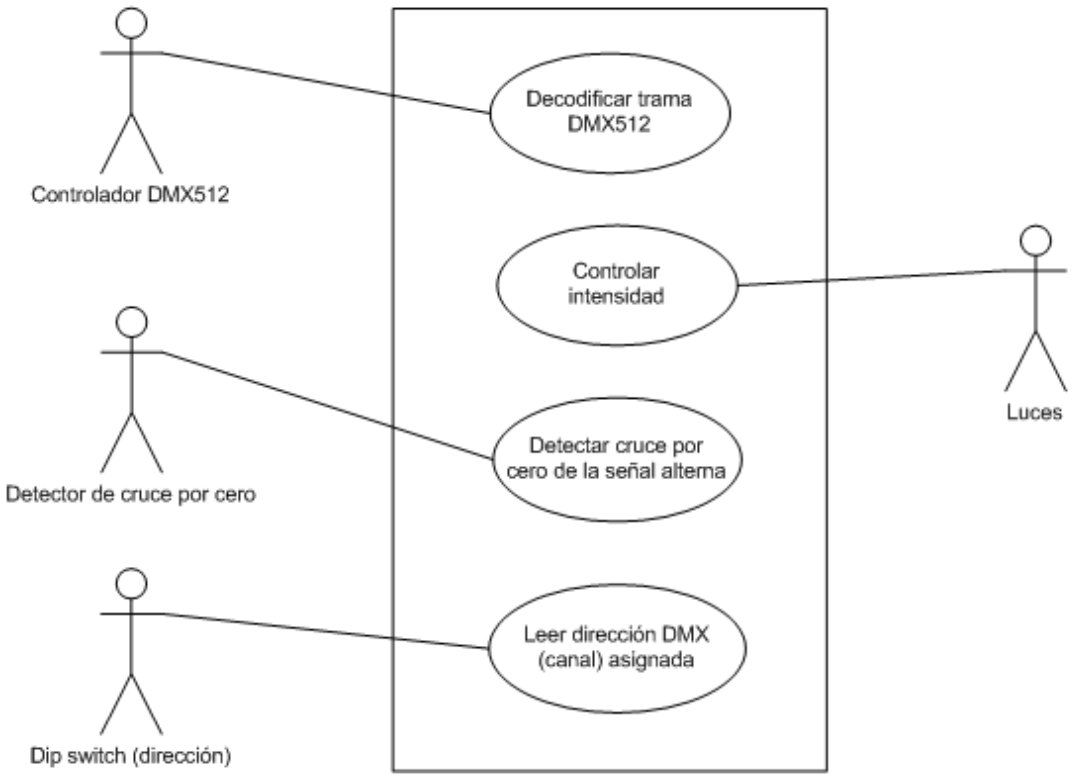

Figura 4.1. Casos de uso del receptor DMX.

Con base en las especificaciones del SciDMX y el diagrama de casos de uso anterior, la división del diseño del receptor DMX se dividió en sus componentes HW y SW de acuerdo a la Tabla 4.1.

Tabla 4.1 División del diseño del receptor DMX del SciDMX en sus componentes HW y SW.

HW	**Dispositivo**	**SW**	**Dispositivo**
Unión al medio físico	Transceptor	Lectura de la dirección DMX512	Microcontrolador
Asignación de la dirección DMX512 del receptor	Dip-switch	Decodificación de la señal DMX512 proveniente del controlador	Microcontrolador
Detección del cruce por cero de la señal de CA.	Circuito detector de cruce por cero.	Detección de la interrupción generada por el detector de cruce por cero.	Microcontrolador
		Control del disparo de los tiristores de la etapa de potencia del dimmer.	Microcontrolador

El HW del receptor DMX se compone de los siguientes dispositivos, Figura 5.2:

- MCU AT90S2313 de la firma Atmel.
- Transceptor de bus diferencial SN75176 compatible con las características del protocolo RS485.
- Dip-Switch para establecer la dirección o canal DMX del receptor.
- Circuito detector de cruce por cero.
- Cable UTP (Unshielded Twisted Pair).
- Conector XLR macho de 3 pines.
- Fuente de alimentación lineal de +5V.

Las dos tareas principales del receptor DMX son:

1, Demultiplexar la señal DMX512 proveniente del controlador DMX para obtener los datos DMX512 que le corresponden.

2. Funcionar como dimmer, controlando el disparo de los tiristores en sincronía con la CA.

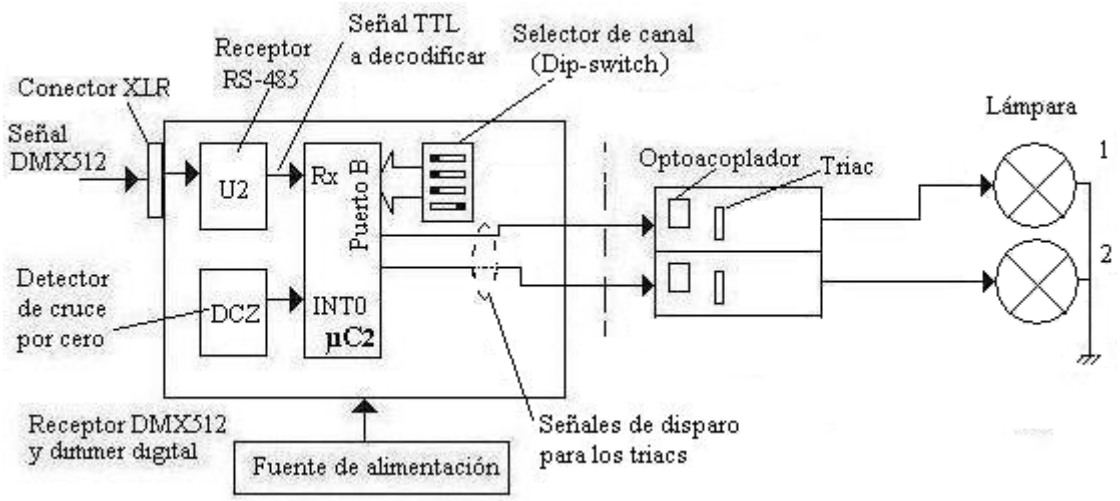

Figura 4.2. Diagrama a bloques del receptor DMX.

4.1.1. *Selección HW y SW para el receptor DMX*

Para el receptor se seleccionó el mismo MCU (AT90S2313) que se utiliza en el controlador, por las razones mencionadas en el Capítulo 2. Al igual que la interfaz física entre el controlador y el receptor del SciDMX, se seleccionó al transceptor de bus diferencial SN75176B.

Herramientas de desarrollo para el diseño HW y SW

Las herramientas de desarrollo para el diseño HW y SW del receptor DMX se utilizó:

- SW de aplicación AVR Studio versión 3.53 de la firma Atmel.
- Programador ISP para MCUs AVR.

4.2. *Fase 3: Iteración y desarrollo del receptor DMX*

Las tareas que se realizaron para el desarrollo del receptor en el entorno *AVR Studio* son las siguientes:

1. Escribir y depurar programas de prueba.

2. Simular la lectura y escritura a direcciones de memoria, así como la configuración de los registros del MCU.

3. Simular la escritura de bits en los registros de I/O.

4. Verificar la ejecución de las subrutinas, la activación de interrupciones y el tiempo de ejecución de las mismas.

5. Descargar el programa al MCU mediante el programador.

6. Programar los bits internos del MCU para seleccionar las opciones de funcionamiento.

Las tareas iterativas HW y SW, de corrección o de ajuste, son las siguientes:

1. Asignar direcciones de memoria interna en el MCU.
2. Asignar funciones a los puertos del MCU.
3. Configurar los registros del MCU.
4. Agregar y/o modificar las conexiones HW necesarias.

4.3. *Fase 4: Diseño paralelo HW y SW del receptor DMX*

4.3.1. *Diseño HW del receptor DMX*

Una vez seleccionado los componentes HW del receptor DMX, se definen los siguientes pasos:

- Asignación de funciones a los puertos del MCU.
- Asignación de los registros del MCU utilizado.

4.3.1.1. *Asignación de funciones de los puertos del MCU del receptor DMX*

La Tabla 4.2 muestra la asignación de los puertos y pines del MCU del receptor DMX.

Tabla 4.2. Función para cada pin de los puertos del MCU del receptor DMX.

Puerto	**Pin**	**Uso**	**Función**
B	PB0-PB7	Entrada	Dirección del canal DMX asignado al receptor.
D	PD0	Entrada	Entrada de la señal DMX512 (niveles TTL)
	PD1	Salida	Pulso de disparo para la etapa de potencia del dimmer con dirección DMX n.
	PD2	Salida	Pulso de disparo para la etapa de potencia del dimmer con dirección DMX n+1.

En el diagrama eléctrico del receptor DMX, mostrado en la Figura 4.3 puede apreciarse las conexiones de los pines del MCU con los demás dispositivos.

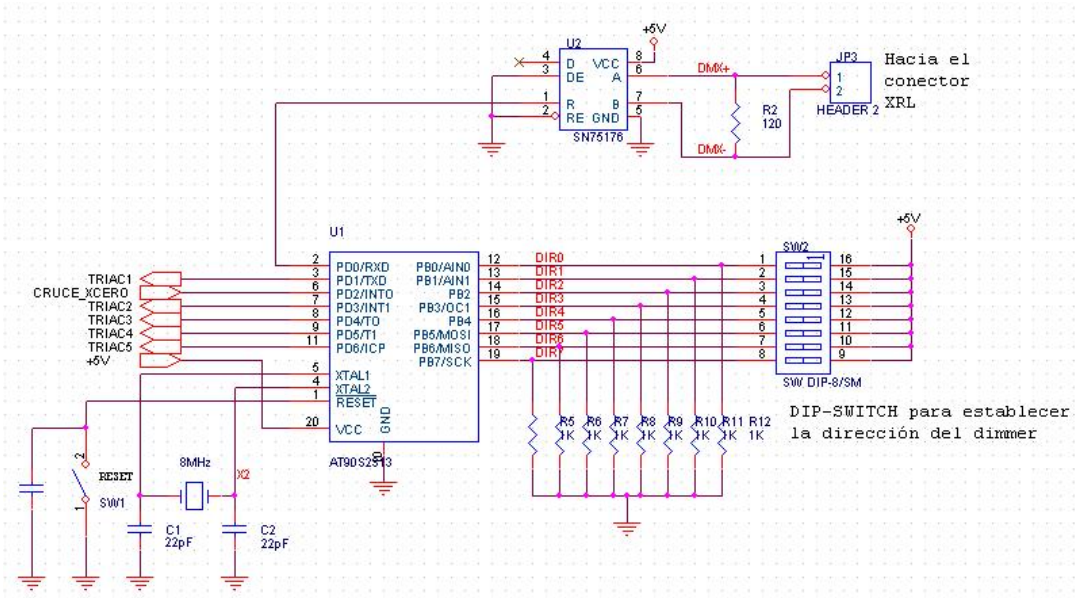

Figura 4.3. Diagrama eléctrico del receptor DMX.

4.3.1.2. *Asignación de los registros del MCU a emplear para el receptor DMX.*

La Tabla 4.3 muestra la relación de los registros del MCU del receptor DMX.

Tabla 4.3 Mapa de memoria de los registros del MCU utilizado.

Registro	Nombre	Dirección
SPL	Apuntador de pila	3Dh
MCUCR	Control general del MCU	35h
PORTB	Datos del puerto B	18h
DDRB	Dirección de datos del puerto B	17h
PORTD	Datos del puerto D	12h
DDRD	Dirección de datos del puerto D	11h
GIMSK	Máscara de interrupciones generales	3Bh

4.3.2. *Diseño del SW del receptor DMX*

El diseño del SW del receptor se basa en las especificaciones iniciales del sistema, considerando los siguientes factores:

- Interfaz del MCU con la señal DMX proveniente del controlador.
- Dip-switch selector del canal del receptor.
- Circuito detector de cruce por cero (DCZ).
- Señales (pulsos) de control hacia las dos etapas de potencia.
- Asignación de puertos del MCU.

El código de configuración del MCU del receptor está basado en programación estructurada considerando las fases de mantenimiento y actualización.

Para describir el comportamiento, tanto del receptor DMX como del dimmer, se ha utilizado un diagrama de máquinas de estado, el cual es mostrado en la Figura 4.4. En las Tablas 4.4 y 4.5 se presentan la descripción de los estados y los estímulos del programa, respectivamente.

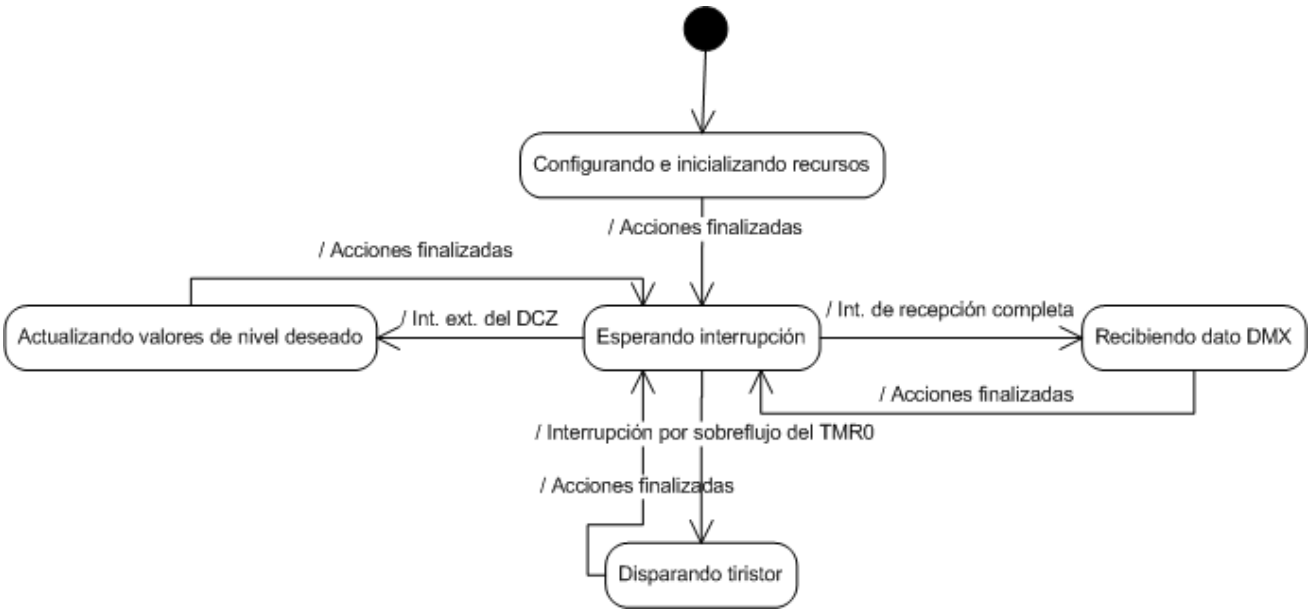

Figura 4.4 Diagrama de máquinas de estados del programa del receptor DMX.

Tabla 4.4. Descripción de los estados del programa del receptor DMX.

Estado	**Descripción**
Configurando e inicializando recursos	El programa del MCU configura: la pila, los puertos y registros a utilizar.
Esperando interrupción	El programa del MCU está esperando alguna de las tres interrupciones.
Actualizando valores de nivel deseado	En éste estado, en cada cruce por cero los valores de nivel deseados para los canales n y n+1, son leídos de la memoria SRAM.
Disparando tiristor	El programa del MCU, en este servicio a interrupción evalúa si debe disparar a alguno de los dos tiristores

	(canal n y canal n+1) o inclusive a ambos.
Recibiendo dato DMX	En cada trama DMX, se transmiten hasta 512 bytes como datos DMX, de éstos el receptor sólo almacena los dos bytes que corresponden en su valor de posición al valor de las direcciones DMX n y n+1. Los bytes recibidos son datos DMX validos y son almacenados para su procesamiento en el estado Actualizando valores de nivel deseado.

Tabla 4.5 Estímulos del programa del receptor DMX.

Estímulo	**Descripción**
Interrupción de recepción completa	La interrupción ha sido generada debido a que la recepción de un byte ha finalizado.
Interrupción del DCZ	Una interrupción externa ha sido generada por el circuito DCZ, debido a que éste último generó un pulso al detectar el cruce por cero de la señal de CA que alimenta a las lámparas.
Interrupción por sobreflujo del TMR0	Una interrupción ha sido generada debido a que ha transcurrido un lapso de 32.5 μs.
Acciones finalizadas	El conjunto de instrucciones del estado han terminado

Los detalles de cada uno de los estados se describen a continuación.

4.3.2.1. *Estado: Configurando e inicializando recursos del MCU*

Las acciones realizadas en el estado *configurando e inicializando recursos* son:

- Inicializar la pila del MCU: configurar el registro SPL para asignar la dirección de memoria que debe tener el apuntador de la pila del MCU.
- Declarar el vector de interrupciones del MCU: a la interrupción externa se le asigna su número de vector correspondiente, así como su etiqueta de identificación.
- Configurar puertos B y D: se configuran los bits de los registros DDRB y DDRD para habilitarlos como entrada y salida, respectivamente, de acuerdo a la Tabla 4.2.
- Habilitar la interrupción INT1: se configura el registro GIMSK para habilitar la interrupción INT1.
- Configurar la interrupción INT1 por medio de los bits del registro MCUCR, para seleccionar la activación por flanco de subida de la interrupción INT1.
- Habilitar la interrupción global: se ejecuta la orden para habilitar las interrupciones que el circuito DCZ puede generar.
- Inicializar la memoria RAM interna, que es empleada para almacenar los valores de los datos DMX de los canales n y n+1.

Los procedimientos anteriores se muestran en el diagrama de flujo de la Figura 4.5.

4.3.2.2. *Estado: Esperando interrupción*

El programa cambia a este estado cuando las acciones del estado *Configurando e inicializando recursos del MCU* han finalizado. Este estado básicamente es un ciclo en el que se realizan solo dos cosas:

- Leer la dirección DMX512 asignada al receptor. Así, si el usuario cambia la dirección DMX con el dip_switch (SW2) en el momento de uso, no requerirá que el sistema se reinicie (volver a encender).
- Esperar la generación de alguna de las interrupciones para cambiar a uno de los otros tres estados del receptor. Las tres interrupciones posibles que el programa contempla son las mostradas en el diagrama de estados de la Figura 4.4.

4.3.2.3. *Estado: Recibiendo dato DMX*

El programa cambia al estado *Recibiendo dato DMX* cuando ocurre una interrupción por recepción completa de la UART del MCU. Esto significa que el control de la UART se hace por medio de interrupción y no por sondeo (Polling), con la intención de tener una comunicación serial en segundo plano para que el MCU este dedicado a trabajar como dimmer.

Antes de explicar con mayor detalle el funcionamiento de este estado, es importante presentar el principio de operación general de una comunicación serial y en particular el funcionamiento de la UART del MCU.

Principio de operación general de una comunicación serial

En una comunicación serial asíncrona, los datos son transmitidos secuencialmente por la línea (línea para transmitir), enviando bit por bit. Para informar al receptor que un nuevo byte está llegando, cada uno de estos es colocado entre un bit de inicio y un bit de stop, esta construcción es llamada un marco (frame). El formato del marco es mostrado en la Figura 4.6. El marco tiene un bit de inicio, ocho bits de datos y al menos un bit de stop.

La línea en el estado inactivo (Idle) es señalizada manteniendo la línea en “1” lógico. El bit de inicio es siempre un “0”, y el receptor de la UART detectará el inicio de un marco con el primer flanco de caída, seguido del bit de inicio (nivel bajo) y de los bits de datos, finalizando con un bit de stop el cual siempre es “1” lógico. El valor del bit de stop es mantenido en “1” lógico, hasta que es enviado el siguiente bit de inicio.

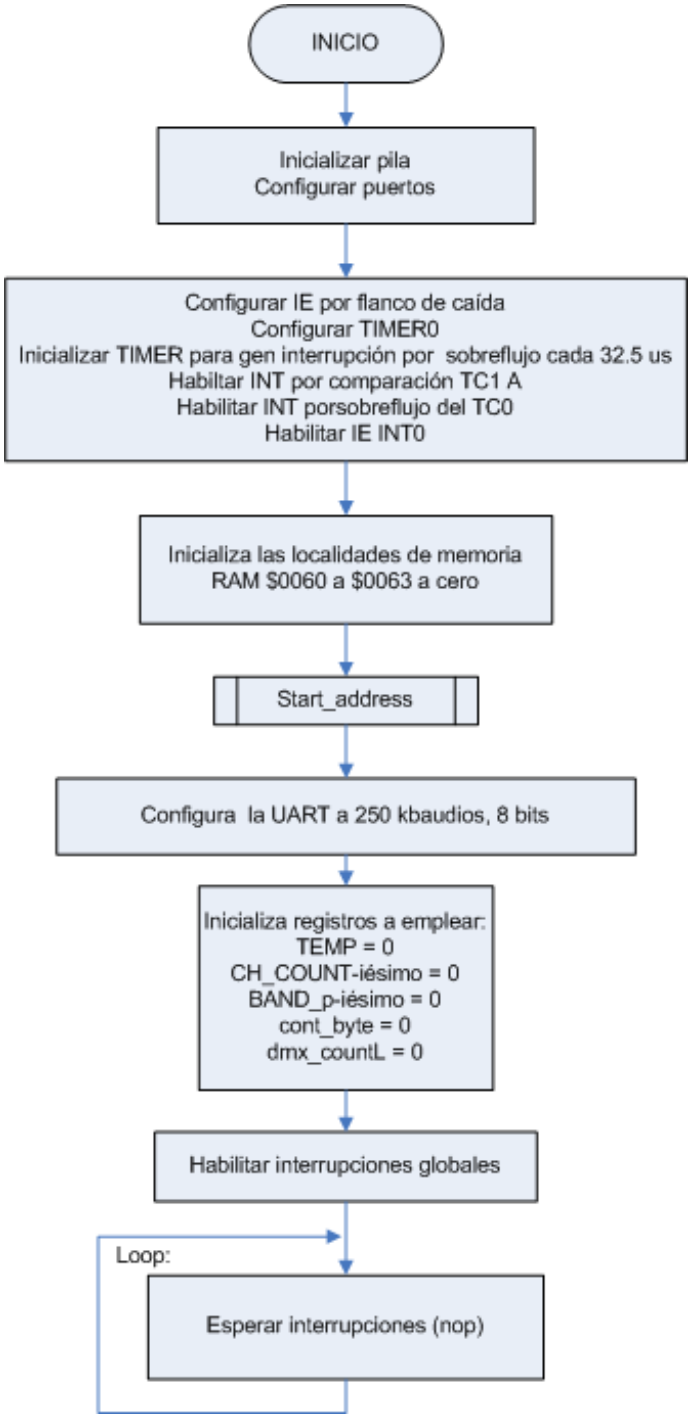

Figura 4.5. Diagrama de flujo del programa principal del receptor del SciDMX.

La velocidad a la que se envían los datos en forma serial a través de una línea de comunicación, se denomina velocidad en baudios. La velocidad en baudios es expresada en unidades de bits por segundo. Así, sí en una transmisión se pueden enviar 1200 bits en un segundo como máximo, el inverso de 1200 dará como resultado el *tiempo de bit* (período de un bit).

Otro punto importante, es que en una transmisión asíncrona, no es proporcionada ninguna información de reloj por separado al receptor. La correcta recepción de los datos es garantizada manteniendo la misma velocidad de baudios entre el transmisor y receptor [AVR304: Half Duplex Interrupt Driven Software UART, Atmel].

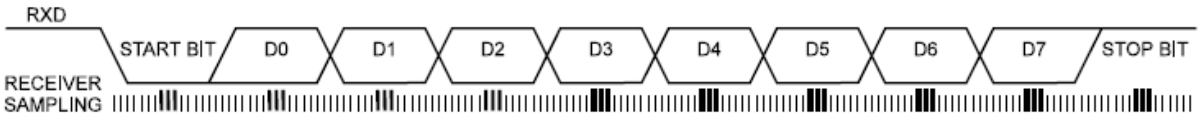

Figura 4.6. Muestreo de un carácter recibido en el pin RXD de la UART.

Funcionamiento de la UART del MCU – Recepción de los datos.

El receptor de la UART del MCU muestrea la señal sobre el pin RXD a una frecuencia mayor (16 veces más) que la tasa de baudios establecida en la UART, Figura 5.6 (muestreo del receptor). Cuando la línea está en el estado inactivo (puesta a 1), una muestra de un "0" lógico será interpretada como el flanco de caída de un bit de inicio, originando con esto, que la secuencia de detección del bit de inicio sea inicializada. Dada la muestra número 1, denotada como la primer muestra del "0", seguida de la transición de 1 a 0 (alto a bajo), el receptor evalúa el pin RXD en las muestras 8, 9 y 10 (de un total de 16). Si se encuentra que dos o más de las tres muestras son "1´s" lógicos, el bit de inicio es rechazado, considerando que la transición de 1 a 0 se debe a disparos falsos ocasionados por una señal de ruido, lo que implica que el receptor deberá iniciar de nuevo la detección de la siguiente transición de 1 a 0.

Si un bit de inicio es detectado exitosamente, lo que sigue es el muestreo de los bits de datos (D0 a D7). Esos bits son evaluados en las mismas muestras 8, 9 y 10. El valor lógico encontrado en al menos dos de las tres muestras son tomadas como el valor del bit. El muestreo de un carácter (en nuestro caso, un dato DMX) es mostrado en la Figura 5.6, como se puede observar el orden de muestreo de los ocho bits de un dato, es del bit menos significativo al bit más significativo, y es así como éstos se van recorriendo en el registro de desplazamiento del receptor de la UART del MCU.

Una vez que los ocho bits de datos son muestreados y recorridos en el registro de desplazamiento del transmisor, solo queda muestrear el bit de stop que llega al receptor para completar la recepción de un carácter. El valor del bit de stop debe ser 1, para aceptar el bit de stop la mayor parte de las tres muestras deben ser 1. Si dos o más muestras son 0´s lógicos, el bit de stop es incorrecto y la manera en que la UART del MCU lo indica automáticamente es poniendo a 1 la bandera FE de *error de marca* (Framing Error) del registro de estado, USR. Es importante señalar que esta condición se dará en la recepción de una trama DMX cada vez que se reciba el *espacio para el break*, recordar que ésta última permanece en bajo durante 88 μs, por lo que el receptor de la UART al detectar la transición de 1 a 0, muestrea el bit de inicio, los 8 bits como ceros y finalmente los bits de stop los sensa como ceros, por lo que la bandera FE será puesta a 1, de esta manera es posible que el MCU determine el *espacio para el break*, sincronizándose así con el controlador DMX, lo que implica que los siguientes datos que reciba son datos DMX validos.

Si después de haber recibido exitosamente un carácter, el registro UDR no ha sido leído desde la última recepción, la bandera OverRun (OR) del registro USR es puesta a 1 [Manual AT90S2313].

Una vez que se ha presentado lo referente a la comunicación serial, es posible describir el funcionamiento del estado *Recibiendo dato DMX*. La Tabla 4.6 describe el nombre y la descripción de los registros de propósito general utilizados en el presente estado.

Tabla 4.6. Nombre de los registros empleados en el estado *Recibiendo dato DMX*.

Registro	**Descripción**
dmx_adrL	Registro que contiene la dirección del dimmer, ésta última es establecida leyendo el valor del dip-switch que está conectado al puerto B del MCU.
dmx_byte	Registro que contiene el enésimo byte (dato dmx) recibido de la UART.
dmx_countL	Registro usado como contador de bytes recibidos por la UART, su valor es inicializado a cero cada vez que se detecta el *espacio para el break.*
dmx_count_in	Registro usado como contador de bytes válidos que han sido almacenados en memoria SRAM. Un byte válido es aquel que su posición en la trama DMX corresponde a los canales DMX *n* y *n+1*, es decir; como cada receptor es de 2 canales, si éste tiene la dirección DMX igual a 1, de los 512 datos DMX que se transmiten, los que debe procesar y almacenar en memoria son los datos DMX 1 y 2. Si tiene la dirección 6, procesará y almacenará en memoria los datos DMX 6 y 7.
Channel	Registro que contiene el número de canales que el receptor maneja, en este caso es igual a 2.

Es importante recordar que el estímulo que permite cambiar a este estado, es una interrupción de recepción completa de la UART, realmente éste estado es un servicio a dicha interrupción. La Figura 4.7 muestra el diagrama de flujo correspondiente.

Para describir el funcionamiento del estado, supongamos que el receptor DMX tiene la dirección DMX igual a 1. Inicialmente al entrar al estado, se lee el valor del registro UDR, el cual contiene el dato recibido por la UART, su valor es almacenado en el registro dmx_Byte. Posteriormente se verifica la bandera OR para determinar si el último dato recibido ha sido leído con anterioridad: Si OR es igual a 1 entonces se indica con la bandera dmx_status a 1 y las acciones del estado finalizan, volviendo al estado: *Esperando interrupción*, esto último implica que las siguientes veces que se entre a este estado, se saldrá inmediatamente, esto será por el resto de la trama DMX (será omitida) hasta recibir el *espacio para el break* de una nueva trama DMX.

Si OR es igual a 0, se verifica la bandera de error de marca (FE), la cual permite detectar el *espacio para el break*, tal como se explicó anteriormente. Así si FE es igual a 1, el MCU conoce que una nueva trama DMX ha iniciado, por lo que inicializa el apuntador de memoria (registro Z) a la localidad $0060 para almacenar los datos DMX válidos, y los registros empleados son inicializados a cero.

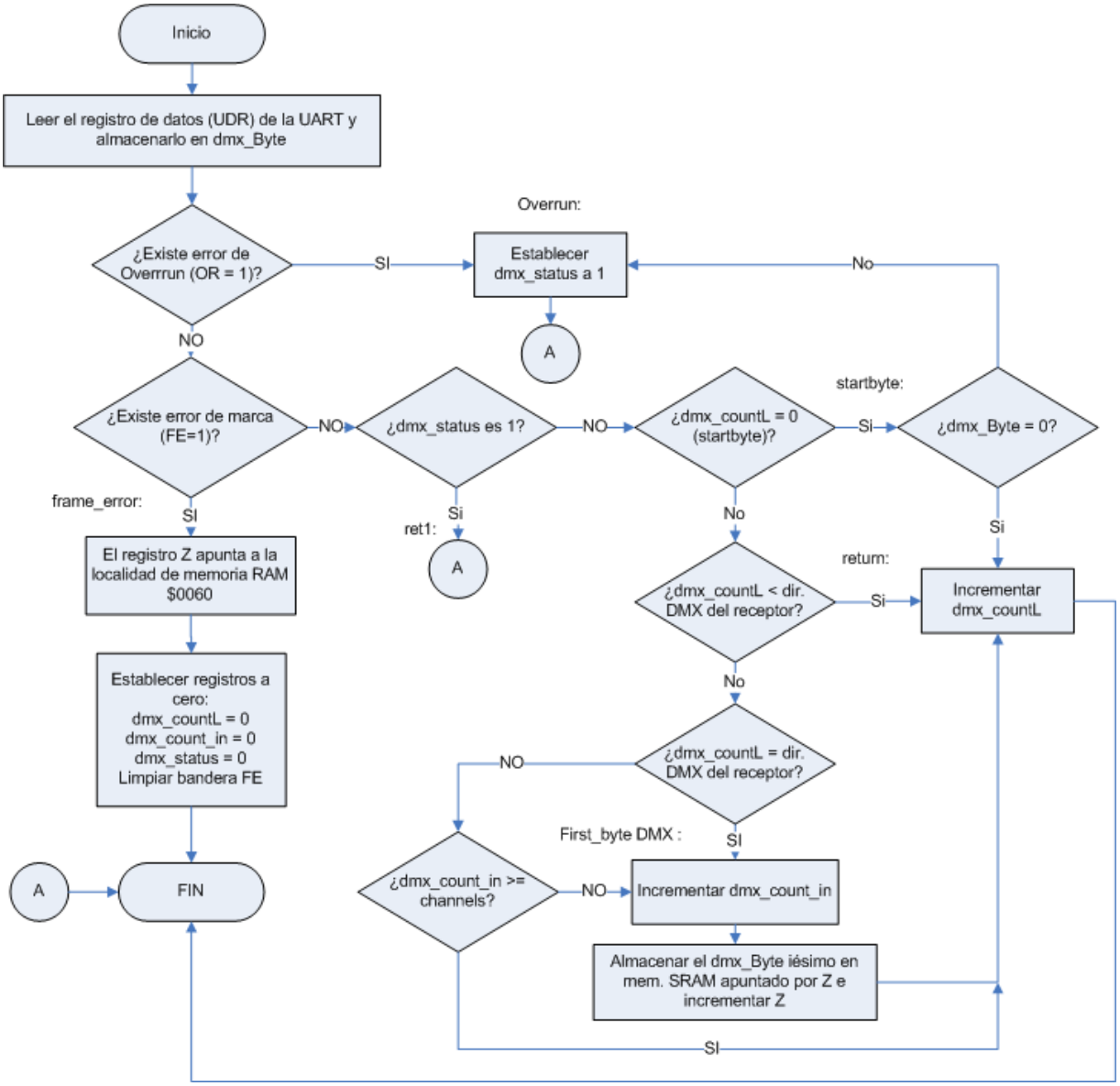

Figura 4.7. Diagrama de flujo para el estado *Recibiendo dato DMX*.

La segunda vez que se entre a este estado, es decir se reciba un nuevo dato es el START CODE con su valor a 0 (dmx_Byte = 0), OR y FE así como el valor de dmx_countL serán 0, entrando casi directamente a la sección *startByte* indicada en el diagrama de flujo, en tal caso el valor de dmx_countL es incrementado en 1, finalizando con esto las acciones del estado.

Con la siguiente vez (tercera) que se entre a este estado, es decir se reciba un nuevo dato, éste será el primer dato DMX o el valor para el canal 1. Como el valor de dmx_countL y la dirección DMX (dmx_adrL) del receptor son iguales, es decir 1, esto causa que se entre a la sección *First byte DMX* indicada en el diagrama de flujo de la Figura 4.7. Con esto el valor de dmx_count_in es incrementado en uno y el valor de dmx_Byte leído inicialmente al entrar al estado es almacenado en la localidad ($60) apuntado por Z (el valor del apuntador Z es incrementado automáticamente). Análogamente, el dato DMX

del canale 2 es almacenado en memoria con dmx_count_in igual a 2. Los canales 3 a 512 serán omitidos por el receptor con dirección DMX igual a 1, un receptor con dirección DMX igual a 5, aceptará los datos de los canales 5 y 6. De esta manera los datos DMX de los canales 1 y 2 han sido almacenados en memoria SRAM, en las localidades $0060 a $0063 respectivamente, para su procesamiento en el siguiente estado.

4.3.2.4. *Estado: Actualizando valores de nivel deseado*

El estímulo que permite cambiarse a este estado es cuando se recibe el flanco de subida de un pulso del circuito DCZ, el flanco de este pulso permite sincronizar al MCU con la CA. El diagrama eléctrico del DCZ se muestra en la Figura 4.8.

El circuito DCZ está formado por un comparador de nivel que utiliza un amplificador operacional, sus dos señales de entrada y su salida se describen a continuación.

La señal de CA, llamada Sen1 es tomada del divisor de voltaje formado por los resistores R13 y R14, la cual esta conectada a la entrada inversora del amplificador operacional U5A, el resistor R13 esta conectado al puente rectificador de diodos de onda completa.

La señal de CD, llamada Vref1 es aplicada a la entrada no inversora del amplificador operacional U5A, ésta es tomada del potenciómetro R1 y es la que determina la anchura del pulso generado por el DCZ. La salida CRUCE_XCERO es un pulso de anchura de 5 μs, con un periodo de 8.33 ms.

En la Figura 4.8 también se muestra la fuente de alimentación lineal, en base al regulador LM7805. El diodo D4 aísla la señal de CA de onda completa del capacitor C1.

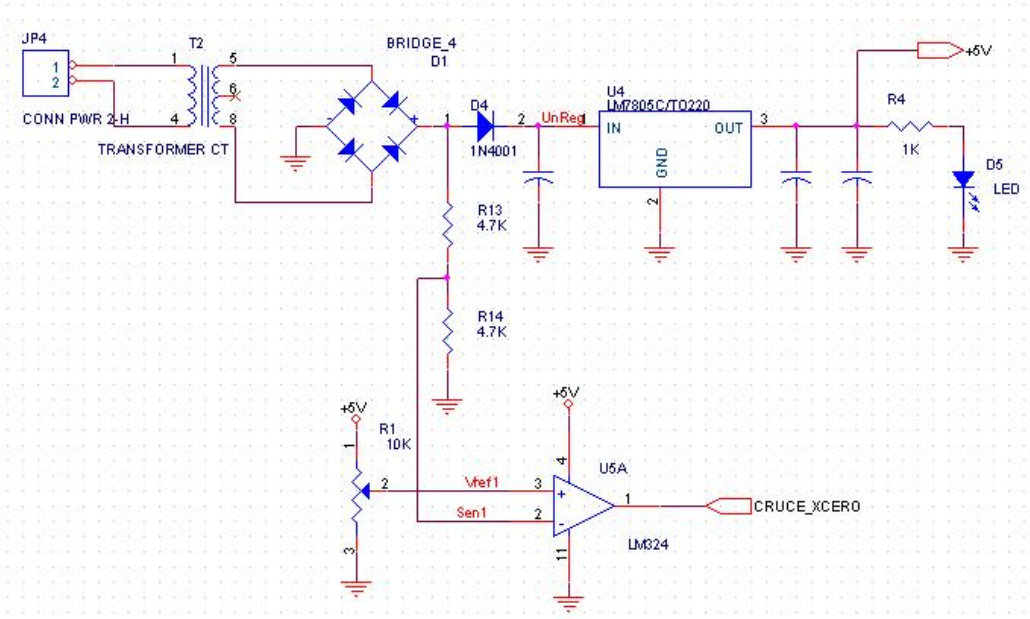

Figura 4.8. Diagrama eléctrico del detector de cruce por cero, y fuente de alimentación.

La Tabla 4.7 muestra el nombre y la descripción de los registros de propósito general empleados en el estado *Actualizando valores de nivel deseado*.

Tabla 4.7. Nombre de los registros empleados en el estado *Actualizando valores de nivel deseado*.

Registros	Descripción
CH_COUNT1 y CH_COUNT2	Registros que contienen el valor del dato DMX del canal 1 y 2, respectivamente, que fueron almacenados en el estado *Recibiendo dato DMX*.
BAND_p1 y BAND_p2	Banderas BAND_p1 y BAND_p2 asociadas a los registros CH_COUNT1 y CH_COUNT2, respectivamente. Permiten establecer y determinar si el triac correspondiente al canal ya fue disparado, durante el semiciclo actual de la CA.

Como se ha mencionado, el estímulo que permite cambiar a este estado, es una interrupción externa generada por el pulso del DCZ, por tanto, este estado es realmente un servicio a la interrupción de INT0 del MCU. La Figura 4.9 muestra el diagrama de flujo correspondiente a éste estado.

Los dos datos DMX que fueron almacenados en memoria SRAM durante la ejecución del estado *Recibiendo dato DMX* son tomados y almacenados en los registros CH_COUNT1 y CH_COUNT2, respectivamente, a su vez cada uno de estos registros tiene asociado una bandera: BAND_p1 y BAND_p2, respectivamente, las cuales son establecidas a uno indicando que existe un nuevo cruce por cero, por lo que es necesario disparar a los triacs correspondientes cuando sea adecuado, esto último será determinado por el siguiente estado.

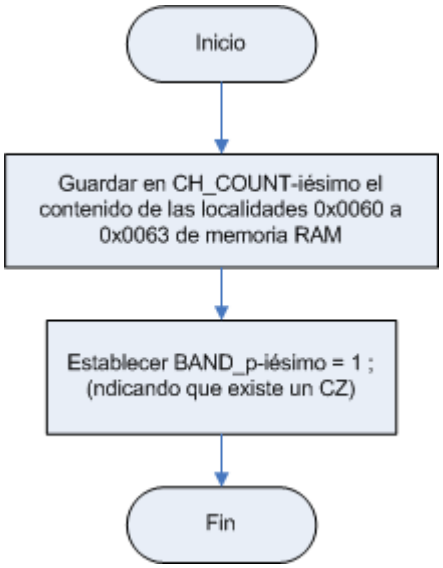

Figura 4.9. Diagrama de flujo para el estado *Actualizando valores de nivel deseado*.

4.3.2.5. *Estado: Disparando tiristor*

El programa cambia al estado *Disparando tiristor* cada que ocurre un sobre flujo en el TMR0 del MCU, el intervalo o periodo en el cual ocurre el sobre flujo es calculado tomando en cuenta algunas consideraciones que se explicarán más adelante.

El estado *Disparando tiristor*, es el estado donde se controla el disparo del tiristor de la etapa de potencia. Cada módulo receptor-dimmer es de dos canales DMX512, a su vez cada canal contiene una etapa de potencia, es decir cada receptor-dimmer tiene dos etapas de potencia como la que se muestra en el esquemático de la Figura 4.10.

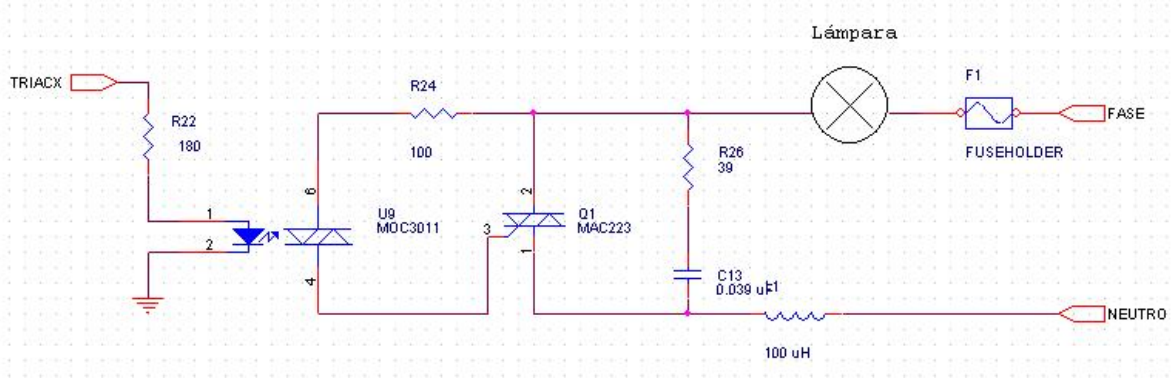

Figura 4.10. Diagrama eléctrico de la etapa de potencia del dimmer.

Cada etapa de potencia, está formada por una red de optoacoplamiento, un tiristor, una red de protección (snubber) del tiristor, un inductor choke para suavizar los cambios repentinos de corriente en la red eléctrica, los contactos para conectar la lámpara a controlar y un fusible de protección. La entrada de control para el disparo del triac es la señal llamada TRIACX. Las terminales de entrada de CA, llamadas FASE y NEUTRO son conectadas a la red eléctrica. Los valores de la red snubber fueron calculados por medio de la ecuaciones 4.1 y 4.2 [Phillips, 1994] y tomando en cuenta que el $dV_{(com)}/dt$ e I_T para el Triac MAC223 son 10 V/s y 25 A, respectivamente. La frecuencia f es de 60 Hz, y la inductancia de la carga es de 19.773 μH.

$$C \geq 25L\left[\frac{fI_{T(RMS)}}{dV_{(com)}/dt}\right]^2 \qquad \text{(Ecuación 4.1)}$$

$$R = \sqrt{\frac{3L}{C}} \qquad \text{(Ecuación 4.2)}$$

Para poder describir la funcionalidad del estado, es necesario presentar las siguientes consideraciones:

Un dato DMX tiene una longitud de 8 bits, es decir $2^8 = 256$ (0 a 255) posibles valores, lo que se traduce a tener 255 niveles de intensidad en la luz correspondiente. La frecuencia del suministro de energía de CA en México es de 60 Hz, con un periodo de 16.66 ms.

Uno de los requerimientos del sistema es que éste debe controlar la intensidad de las luces, para esto se utiliza la técnica de control por ángulo de fase. El principio consiste en que el flujo de potencia hacia la carga queda en función del retardo del ángulo de disparo α del tiristor [Muhammad, 2001], como se muestra en la Figura 4.11.

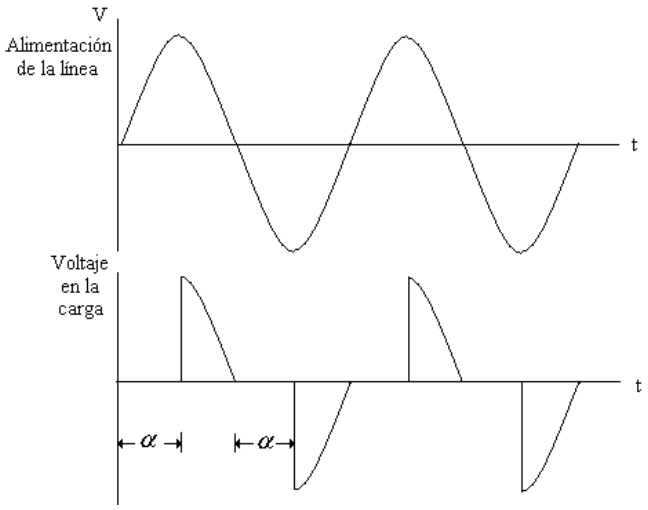

Figura 4.11. Principio de control por ángulo de fase.

Con lo anterior, es posible determinar que el diseño del dimmer requiere que los 8.33 ms de un semiciclo se dividan en 255 posiciones, y posteriormente comparar con el nivel de intensidad deseado, cuando ambos valores sean iguales debe dispararse al triac, con lo anterior se logrará atrasar el pulso de disparo (respecto del cruce por cero) dependiendo del valor DMX recibido por el receptor.

Así, el lapso de tiempo en que debe ocurrir una interrupción del TMR0 del MCU, para entrar a este estado esta dado por:

$$\text{lapso de tiempo} = \frac{T_{semiciclo}}{2^8 - 1} = \frac{8.33\,ms}{255\,niveles} = 32.66\,\frac{\mu s}{nivel}$$

Lo anterior permite dividir un semiciclo de 8.33 ms en 255 niveles de intensidad, de esta manera tenemos 255 posibles incrementos de tiempo para α.

En la Figura 5.12, se presenta el diagrama de tiempos del dimmer, la Figura 4.12-A muestra los pulsos del cruce por cero (DCZ) de la señal de CA, la Figura 4.12-B muestra la división de dos semiciclos de CA en 255 segmentos de 32.66 us, la Figura 4.12-C presenta los dos pulsos de disparo para un ciclo completo de la señal de CA, finalmente, la Figura 4.12-D muestra el voltaje eficaz (zona oscura) que alimenta a la lámpara.

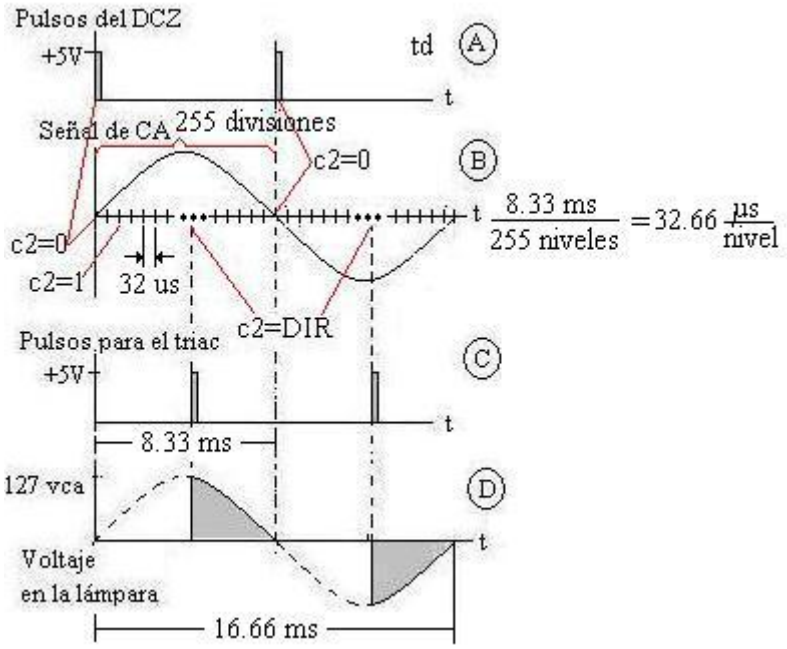

Figura 4.12. Diagrama de tiempos del dimmer.

Con las consideraciones anteriores, se está en posibilidades de describir el funcionamiento del presente estado mediante un ejemplo siguiendo el diagrama de flujo de la Figura 4.13.

Como se ha comentado, en el momento en que el receptor detecta un cruce por cero actualiza el valor de los registros CH_COUNT1 y CH_COUNT2, leyendo el contenido que les corresponde de la memoria RAM del MCU; hasta ahí mantienen su valor intacto, pero una vez que entran a este estado incrementan su valor conforme a lo siguiente:

Suponiendo que el valor del registro CH_COUNT1 tiene un valor de 128, esto significa que se desea que la lámpara mantenga una intensidad media. Al entrar al estado lo primero que se hace es reiniciar el Timer 1 para que genere otra interrupción en los siguientes 32.66 µs, posteriormente el valor de CH_COUNT1 es comparado con el valor de 255, si es igual se revisa la bandera BAND_p1 que indica sí en una interrupción anterior ya se ha disparado al triac, si la bandera permanece en 0, el triac es disparado y dicha bandera es puesta a 1, indicando que para ese semiciclo de la CA el triac ya ha sido disparado y se debe esperar al siguiente semiciclo para el disparo correspondiente. En caso de que CH_COUNT1 no es igual a 255, su valor es incrementado en uno, dando oportunidad que se revise lo mismo para CH_COUNT2. Posteriormente el servicio a la interrupción es finalizado y el control del programa regresa al estado *Esperando interrupción* (Figura 4.4). Con lo anterior es posible hacer que el disparo del triac correspondiente se retrase un tiempo proporcional al dato DMX recibido.

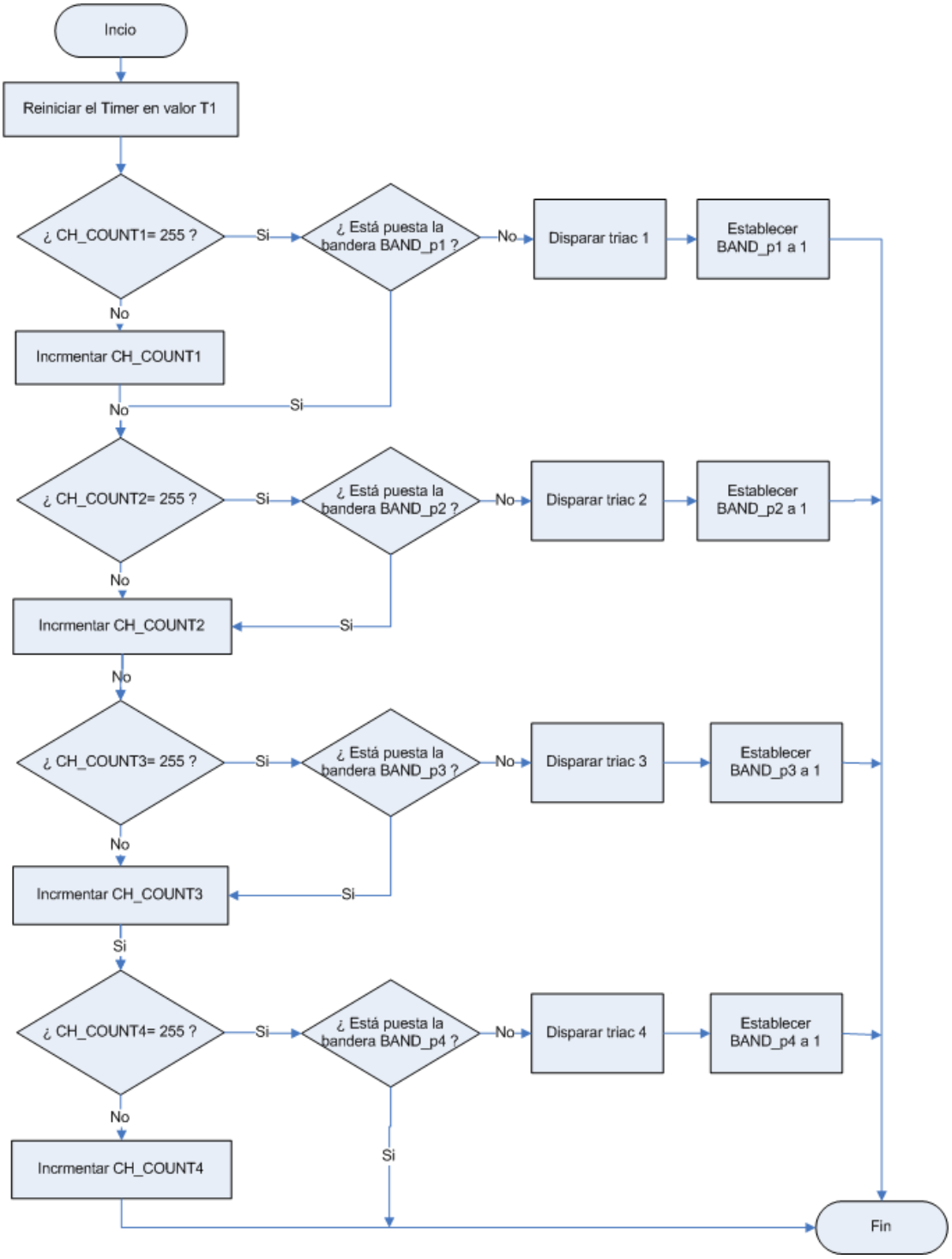

Figura 4.13. Diagrama de flujo para el estado *Disparando tiristor*.

4.4. *Fase 5: Integración HW y SW del receptor DMX*

La integración HW y SW del receptor DMX consistió en realizar las siguientes tareas:

- Verificar las conexiones de los componentes HW.
- Ejecutar programas de prueba a los periféricos (UART del MCU).
- Descargar el programa principal al MCU del receptor DMX mediante el programador.
- Ejecutar el programa principal y validar su funcionamiento.

4.5. *Fase 6. Verificación del receptor DMX*

Para verificar el correcto funcionamiento del sistema final, se realizaron pruebas de la siguiente forma:

- Recepción de tramas DMX, e indicación de un dato DMX en particular por medio de leds.
- Medición del voltaje en la lámpara con diferentes niveles de intensidad.

4.6. *Fase 7: Mantenimiento y actualización del receptor DMX*

La Tabla 4.8 muestra las actualizaciones realizadas al receptor durante su ciclo de vida.

Tabla 4.8. Versiones de actualización del receptor durante su desarrollo.

Versión	Actualizaciones
0.1	Configuración de registros del MCU para habilitar las funciones requeridas en las especificaciones iniciales. El HW del receptor DMX contiene solo el MCU y el dip-swicth.
0.2	Configuración de los registros de la UART del MCU. Se agrega al HW del receptor DMX el transceptor SN75176. Recepción de datos en el MCU desde el controlador a una velocidad de transferencia de 250 Kbps.
0.3	Configuración de la INT1 del MCU para dar soporte a la recepción del cruce por cero del DCZ. Programación de las subrutinas e ISR´s para recibir el pulso del DCZ. El HW del dimmer contiene todo lo necesario para la etapa de potencia. Regulación de la intensidad de una lámpara con el dimmer.
0.4	Fusión del código del receptor DMX y el del dimmer. Pruebas finales con los tres módulos: Programa de computadora, controlador DMX y receptores DMX512 con los dimmers.

Capítulo 5. Pruebas y resultados experimentales

Como resultado de la investigación realizada, el modelado con diagramas UML, y de la aplicación de la metodología de sistemas empotrados se obtuvo el prototipo SciDMX, el cual integra las especificaciones iniciales del sistema.

Debido a que el sistema está formado por varios módulos, a continuación se presentan los resultados de manera individual. Para las pruebas experimentales del SciDMX se realizaron las siguientes mediciones:

- Las pruebas de funcionalidad del programa de control fueron las siguientes:
- Prueba 1. Controlar la intensidad de las luces.
- Prueba 2. Realizar funciones de Fade.
- Prueba 3. Programar y reproducir secuencias.
- Los resultados de estas pruebas, están documentadas en la sección 3.11.
- Medición del tiempo en los segmentos de la señal DMX: Break, MAB (Mark After Break), Start Code, Mark y el tercer bit de uno de los datos enviados, con esta medición se comprobó que la señal DMX cumple con lo establecido en el estándar DMX-512A. La señal se midió en uno de los receptores DMX, con un osciloscopio Tektronix TDS210.
- Medición de voltaje y corriente en una de las lámparas, con una intensidad fija.
- Medición de voltaje y corriente en una de las lámparas, con intensidad incremental.
- Medición del factor de potencia (PF), factor de potencia de desplazamiento (DFP) y distorsión armónica total (THD) del dimmer.

5.1. *Mediciones de tiempo de los segmentos de la señal DMX generada por el controlador DMX*

En la Figura 5.1 se muestra el diagrama de tiempos del segmento Break de la señal DMX generada por el controlador cuando está transmitiendo. La gráfica muestra la medición de la señal *Break*, que se mantiene en bajo durante 88μs. En el diagrama de tiempos también se pueden observar las tramas de la señal DMX.

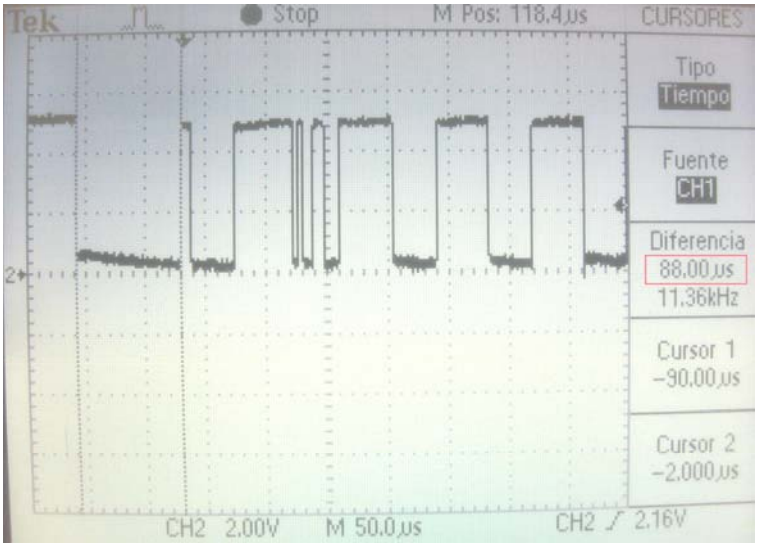

Figura 5.1. Diagrama de tiempos del segmento *Break* de la señal DMX512.

La Figura 5.2 muestra el diagrama de tiempos del segmento MAB de la señal DMX generada por el controlador. La gráfica muestra la medición de la señal *Break*, que se mantiene en alto durante 8µs, que según el estándar esta puede estar entre 8 µs y 1 segundo en nivel alto.

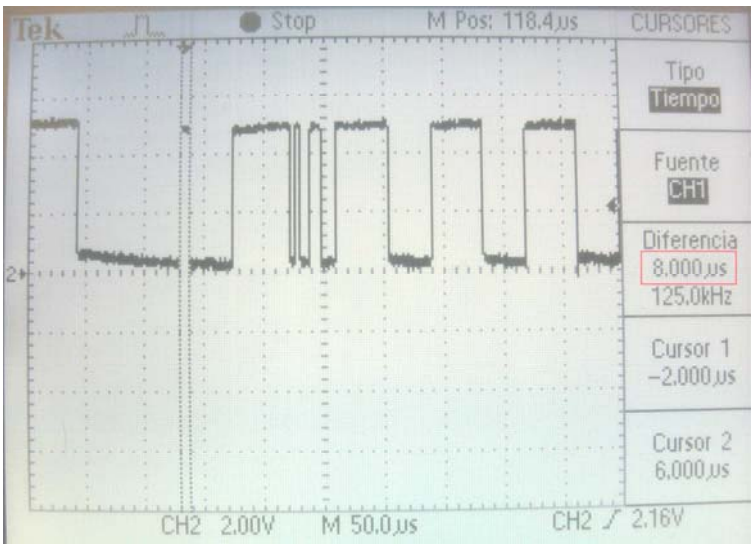

Figura 5.2. Diagrama de tiempos del segmento *MAB* de la señal DMX512.

La Figura 5.3 muestra el diagrama de tiempos del segmento *Start Code* de la señal DMX generada por el controlador. La gráfica muestra que ésta se mantiene en bajo durante 36µs, es decir 8 bits del Start Code y un bit mas del bit de inicio, por cada bit se necesitan 4µs.

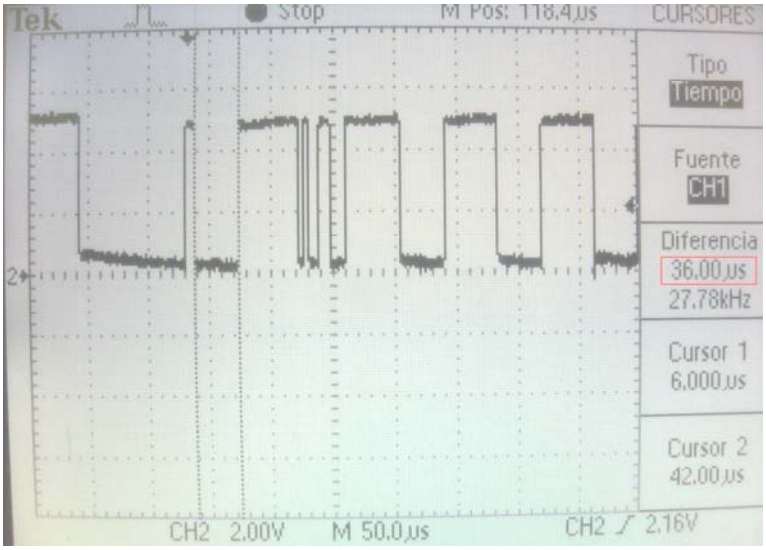

Figura 5.3. Diagrama de tiempos del segmento Start Code de la señal DMX512.

La Figura 5.4 muestra el diagrama de tiempos del segmento *Mark* de la señal DMX generada por el controlador. La gráfica muestra que ésta se mantiene en alto durante 48µs, que según el estándar esta puede estar entre 0 y 1 segundo en nivel alto.

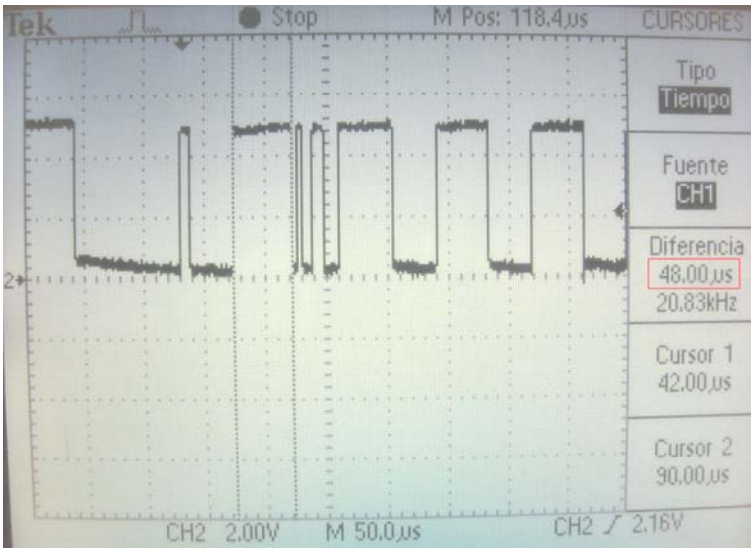

Figura 5.4. Diagrama de tiempos del segmento *Mark* de la señal DMX512.

La Figura 5.5 muestra el diagrama de tiempos de una trama de un byte de la señal DMX. La gráfica muestra el tiempo que dura el tercer bit de uno de los datos enviados en alto, de 4 µs. El dato enviado corresponde a un 85_{10}

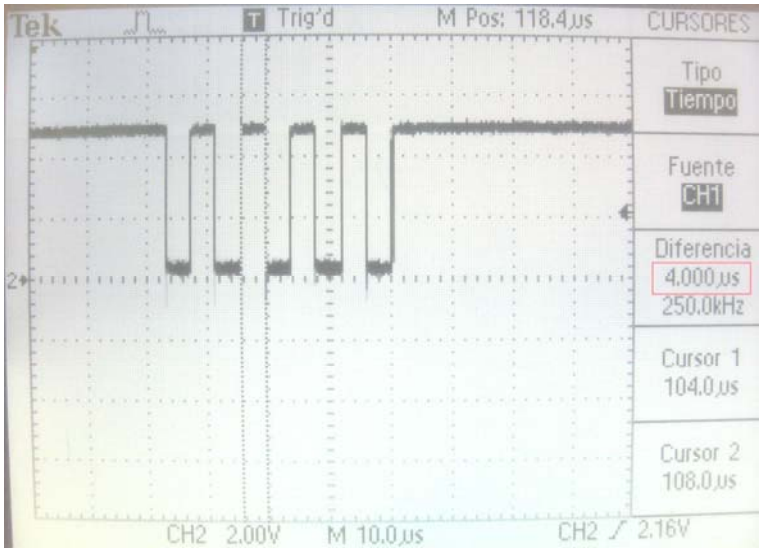

Figura 5.5. Diagrama de tiempos de una trama DMX512.

5.2. *Medición de voltaje y corriente en una de las lámparas, con intensidad fija*

La transmisión de datos DMX entre el controlador y el receptor se realizó a una distancia de 200 m, superando la distancia comúnmente usada en los espectáculos reales [Scott, 1996]. Las pruebas fueron realizadas con 12 dimmers conectados a 12 reflectores de 1KW/127Vrms.

Para la medición de voltaje y corriente rms en una de las luces versus dato DMX fijo, se realizó la conexión de la Figura 5.6. El dato DMX correspondiente al canal 3 enviado por el programa de control de la PC fue fijado a 38_{10}. La recepción de este dato ocasionó que el dimmer proporcionará un voltaje fijo en la luz de 12.43 Vrms con un consumo de corriente de 2.27 A. La Figura 5.7 muestra las formas de onda de voltaje y corriente en la luz. Estas mediciones se realizaron con el analizador de calidad de energía, modelo 43B de la marca Fluke.

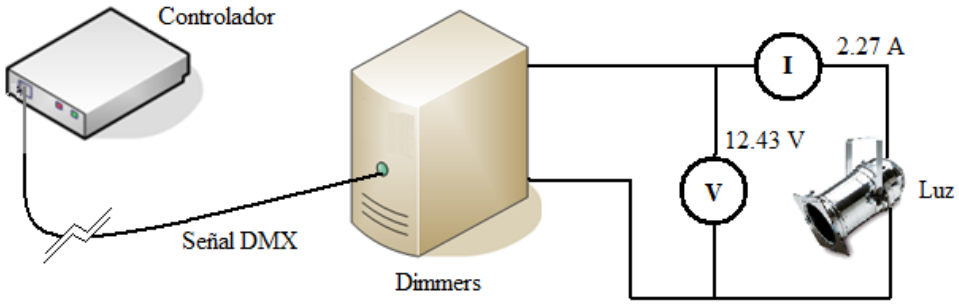

Figura 5.6. Conexión para la medición de voltaje y corriente en una luz con un dato DMX igual a 38_{10}.

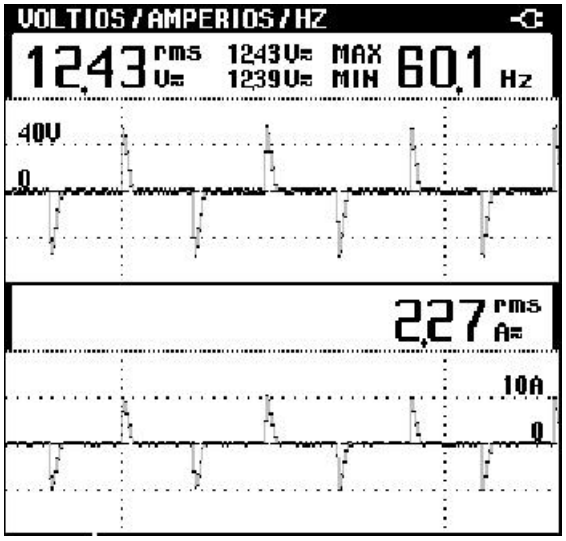

Figura 5.7. Voltaje y corriente en la lámpara (1KW), con un dato DMX igual a 38_{10}.

5.3. *Medición de voltaje y corriente en una de las lámparas, con intensidad incremental*

Para la obtención de las dos gráficas: dato DMX versus Vrms y dato DMX versus Irms en una de las luces, se realizó la conexión de la Figura 5.8. Posteriormente la intensidad de la lámpara conectada al canal 3, se controló desde el programa de la PC empleando la barra de desplazamiento correspondiente, incrementando (cada 10 unidades) pausadamente el valor del dato DMX, desde 0 hasta 255. Por cada incremento realizado se midió el voltaje y corriente rms en dicha luz. La Figura 5.9 muestra las gráficas del dato DMX vs. Vrms y del dato DMX vs. Irms obtenidas. En la primera de ellas es posible observar que la respuesta del filamento de la lámpara no es lineal con el voltaje aplicado a él.

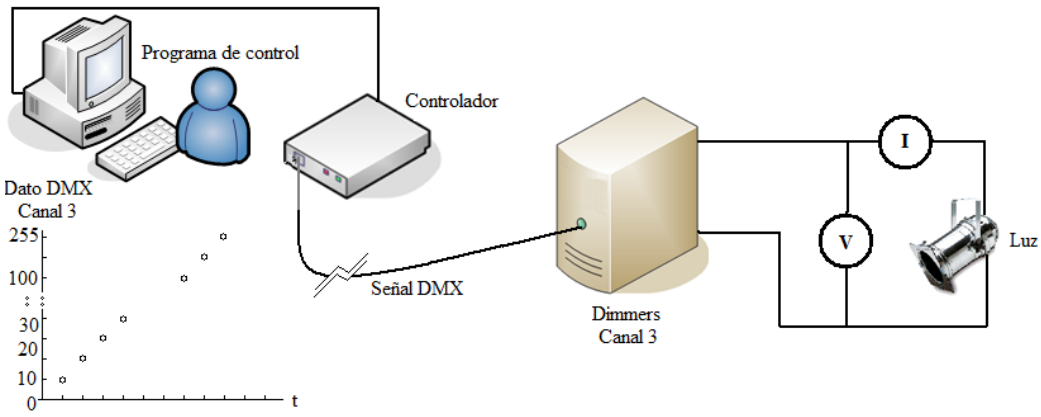

Figura 5.8. Conexión para la medición de voltaje y corriente en una luz con datos DMX de 0 a 255.

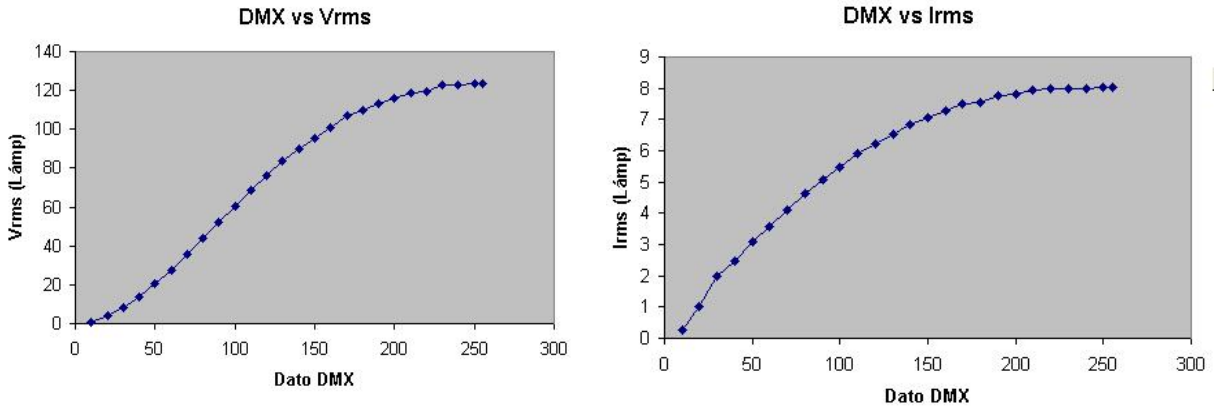

Figura 5.9. a) Gráfica de datos DMX vs. Vrms, b) gráfica de datos DMX vs. Irms.

5.4. *Mediciones del factor de potencia, factor de desplazamiento y distorsión armónica total en el dimmer*

Para las mediciones del factor de potencia (PF), el factor de desplazamiento (DPF) y la distorsión armónica total (THD), se utilizó el analizador de calidad de energía de la firma FLUKE, modelo 434.

Las mediciones se realizaron en función del dato DMX enviado por el controlador, y con incrementos cada 50 unidades. La Tabla 5.1 muestra el dato DMX enviado por el controlador y el ángulo de disparo del Triac del dimmer correspondiente.

Tabla 5.1. Relación del ángulo de disparo con respecto al dato DMX enviado por en controlador.

Dato DMX	**Ángulo de disparo (º)**
50	35.29
100	70.58
150	105.8
200	141.17
250	176.47
255	180

La figura 5.10 muestra las formas de onda de voltaje y corriente, para un ángulo de disparo de 35.29º.

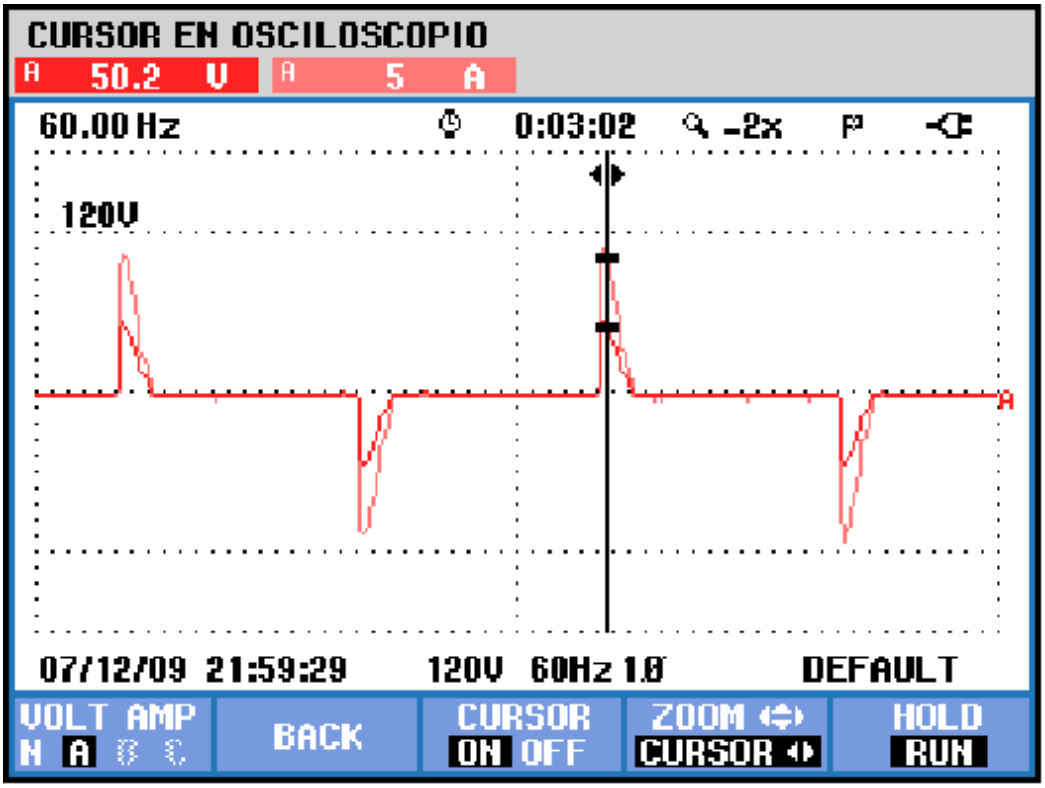

Figura 5.10. Formas de onda de voltaje y corriente para un ángulo de disparo de 35.29º.

En la figura 5.11 se muestra los valores obtenidos del PF, DPF y THD, para un ángulo de disparo igual a 35.29º.

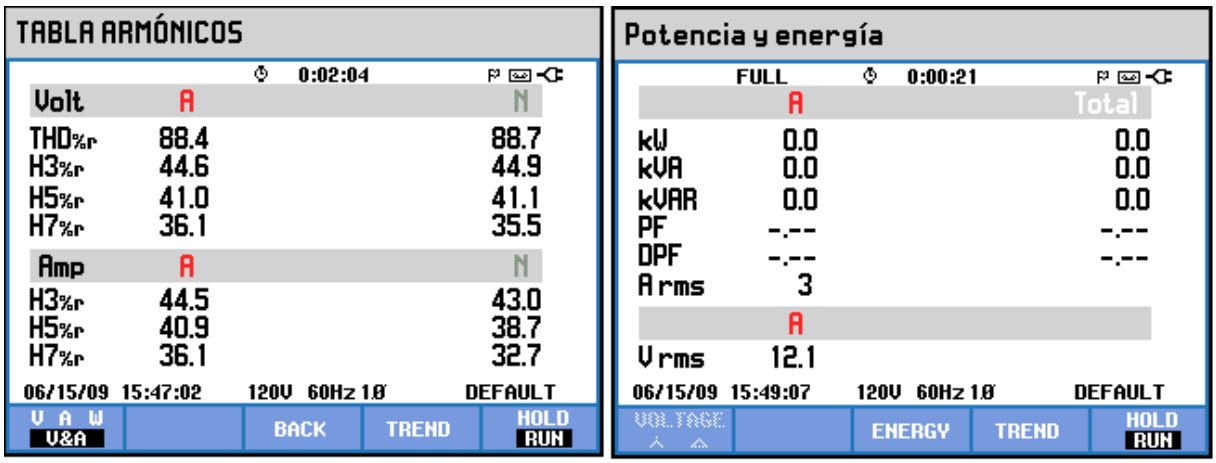

Figura 5.11. Valores obtenidos del PF, DPF y THD, para un ángulo de disparo igual a 35.29º.

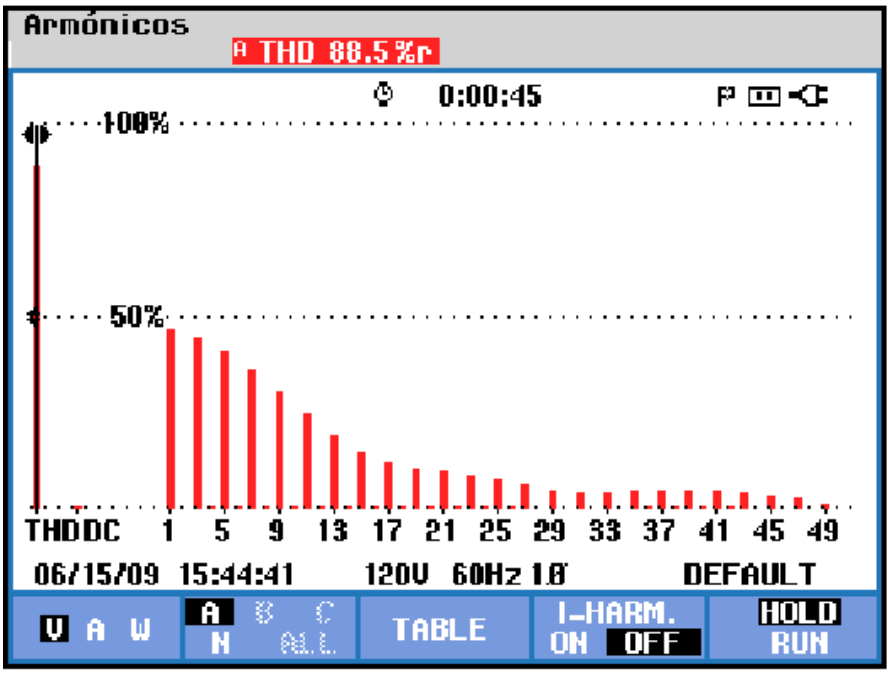

Figura 5.12. Armónicos en voltaje para un ángulo de disparo igual a 35.29º.

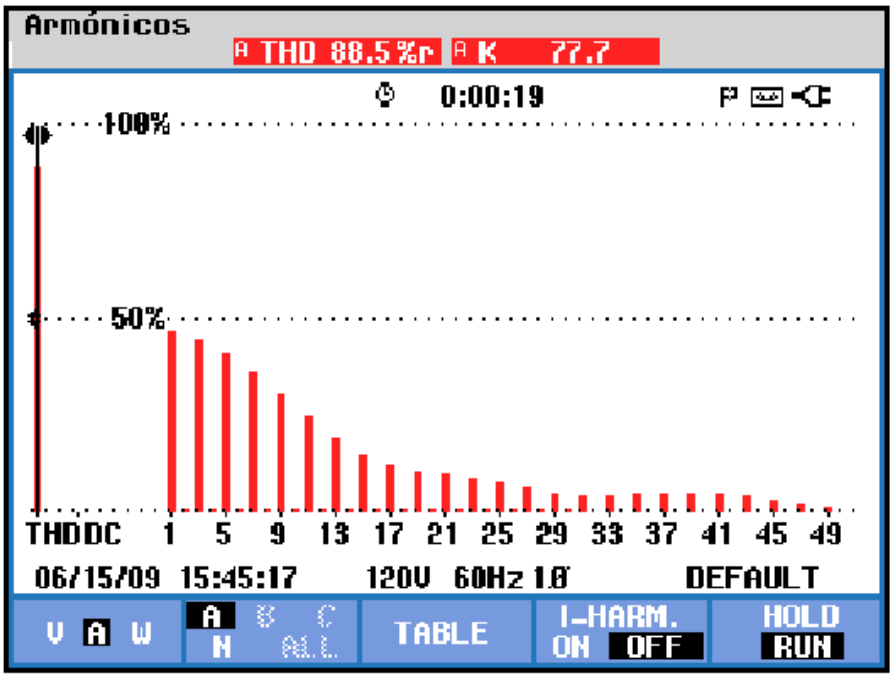

Figura 5.13. Armónicos en corriente para un ángulo de disparo igual a 35.29º.

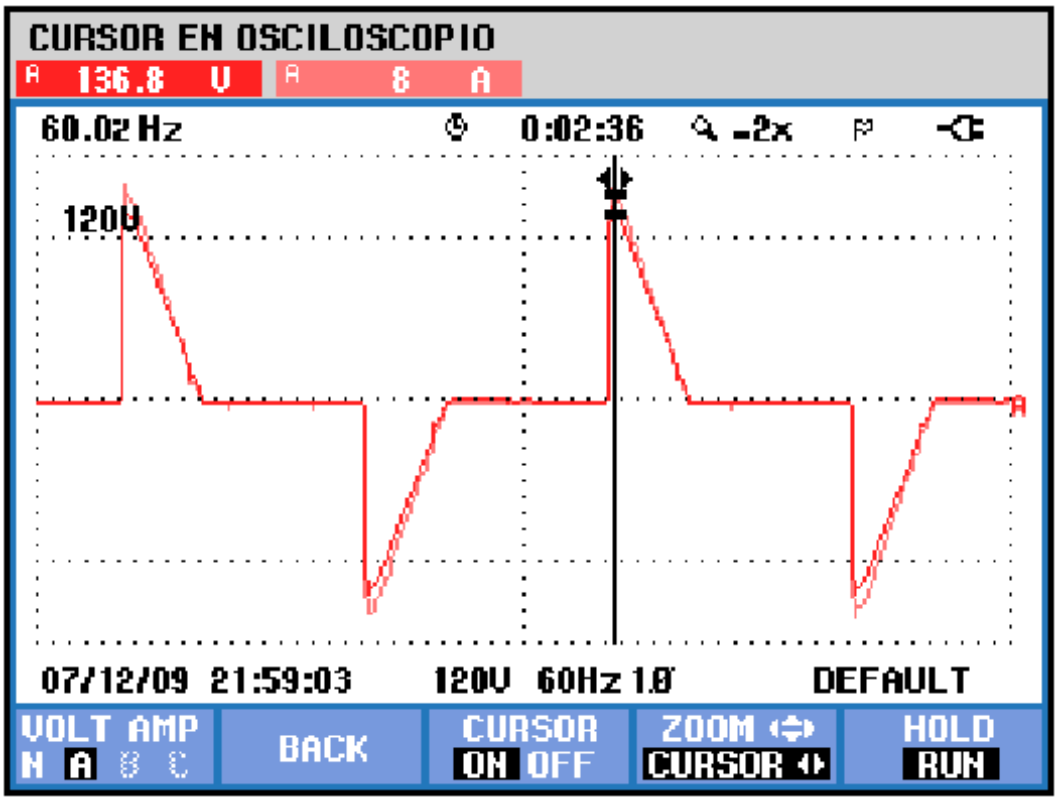

Figura 5.14. Formas de onda de voltaje y corriente para un ángulo de disparo de 70.58º.

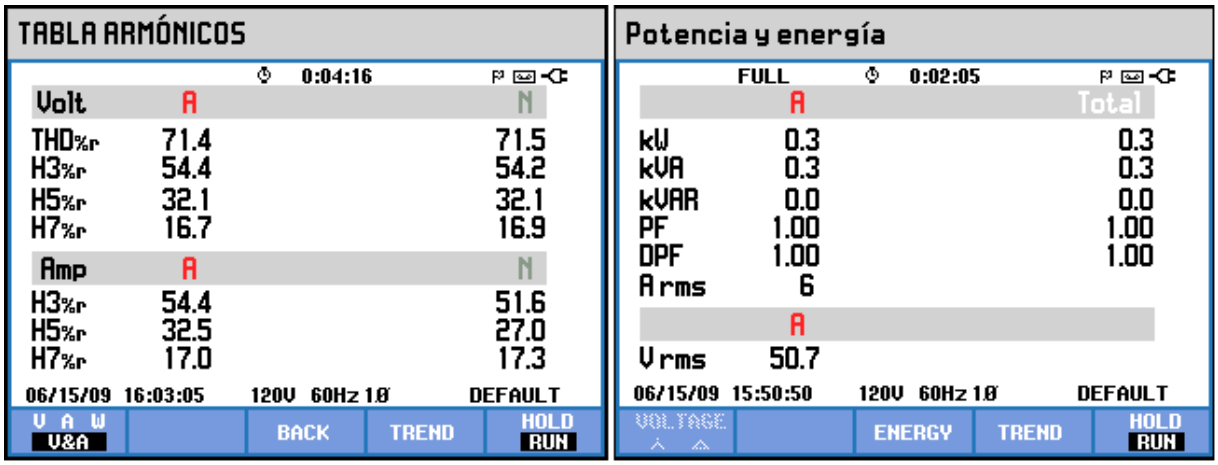

Figura 5.15. Valores obtenidos del PF, DPF y THD, para un ángulo de disparo igual a 70.58º.

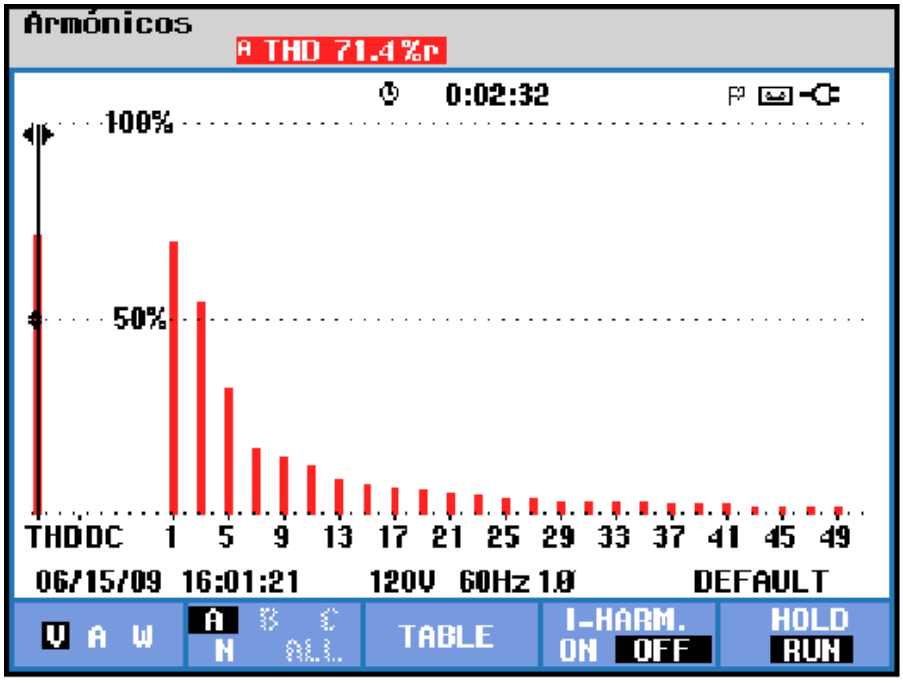

Figura 5.16. Armónicos en voltaje para un ángulo de disparo igual a 70.58º.

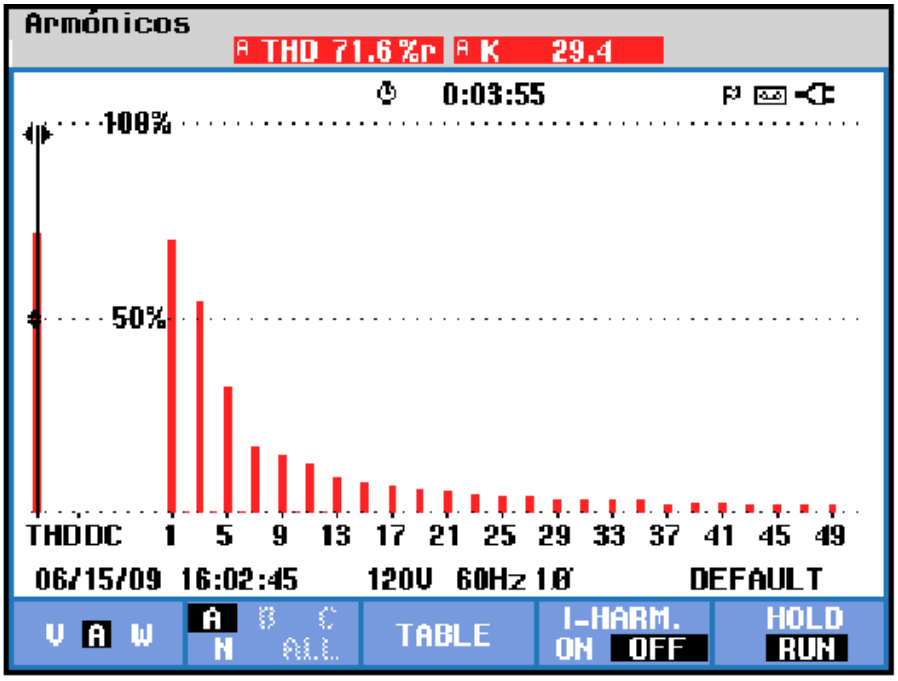

Figura 5.17. Armónicos en corriente para un ángulo de disparo igual a 70.58º.

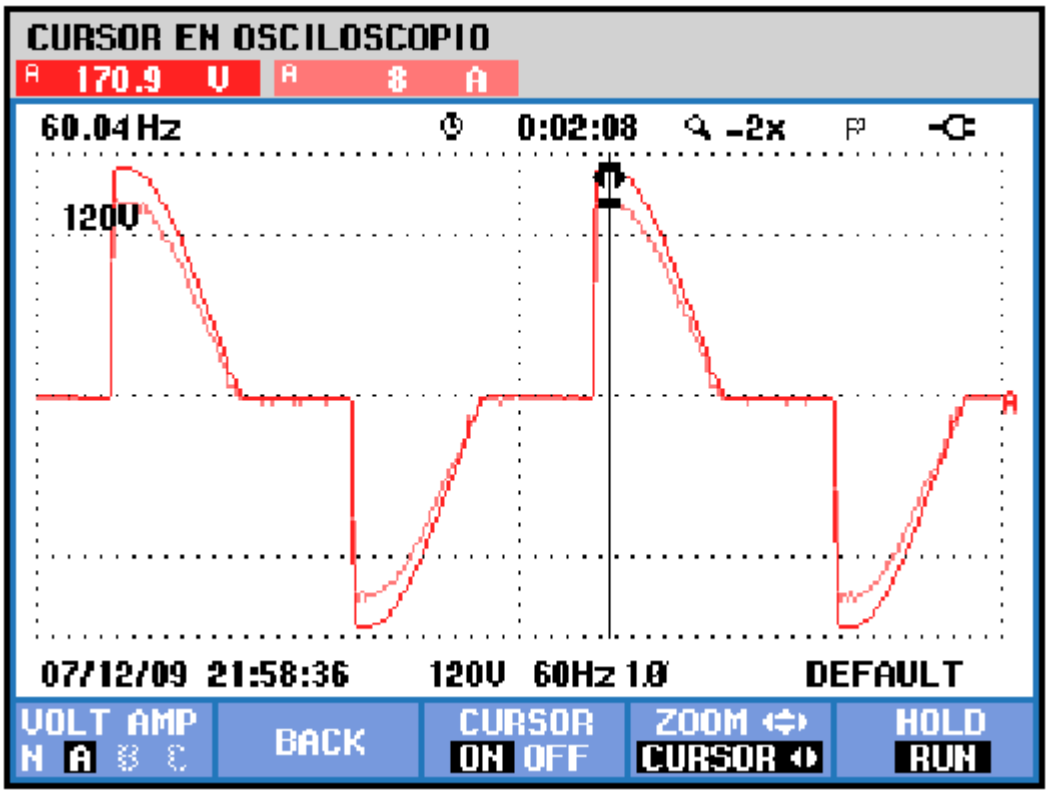

Figura 5.18. Formas de onda de voltaje y corriente para un ángulo de disparo de 105.8º.

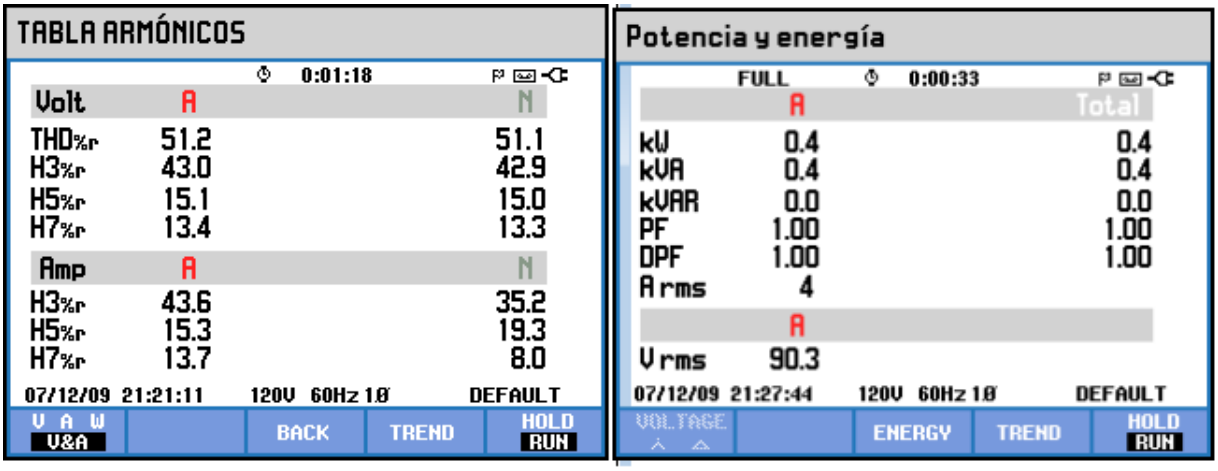

Figura 5.19. Valores obtenidos del PF, DPF y THD, para un ángulo de disparo igual a 105.8º.

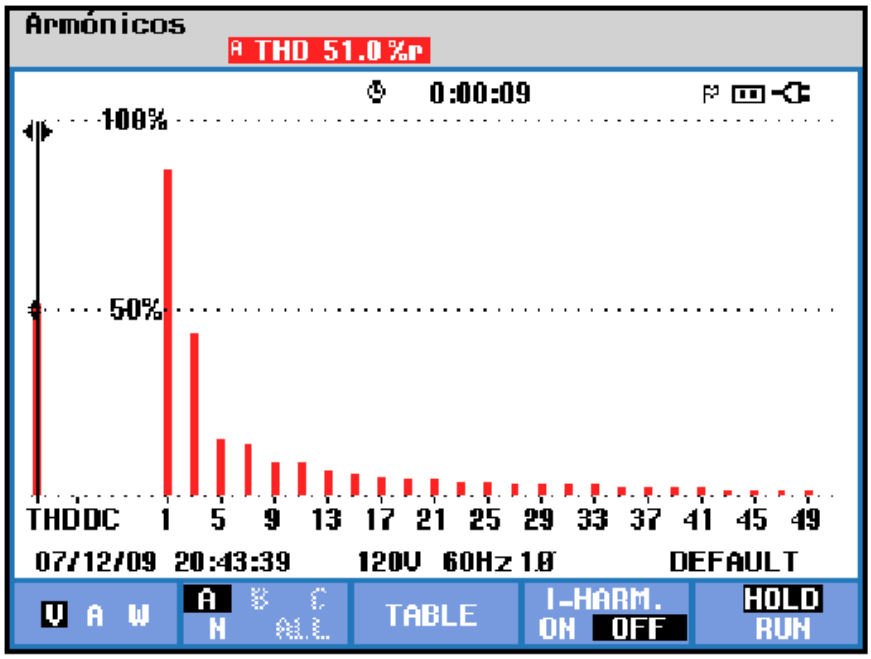

Figura 5.20. Armónicos en voltaje para un ángulo de disparo igual a 105.8º.

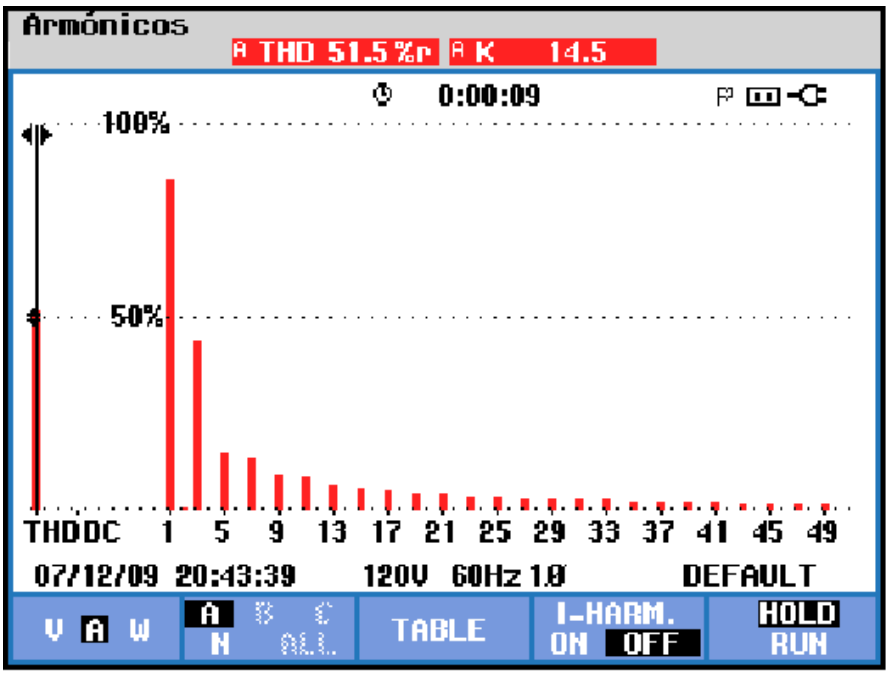

Figura 5.21. Armónicos en corriente para un ángulo de disparo igual a 105.8º.

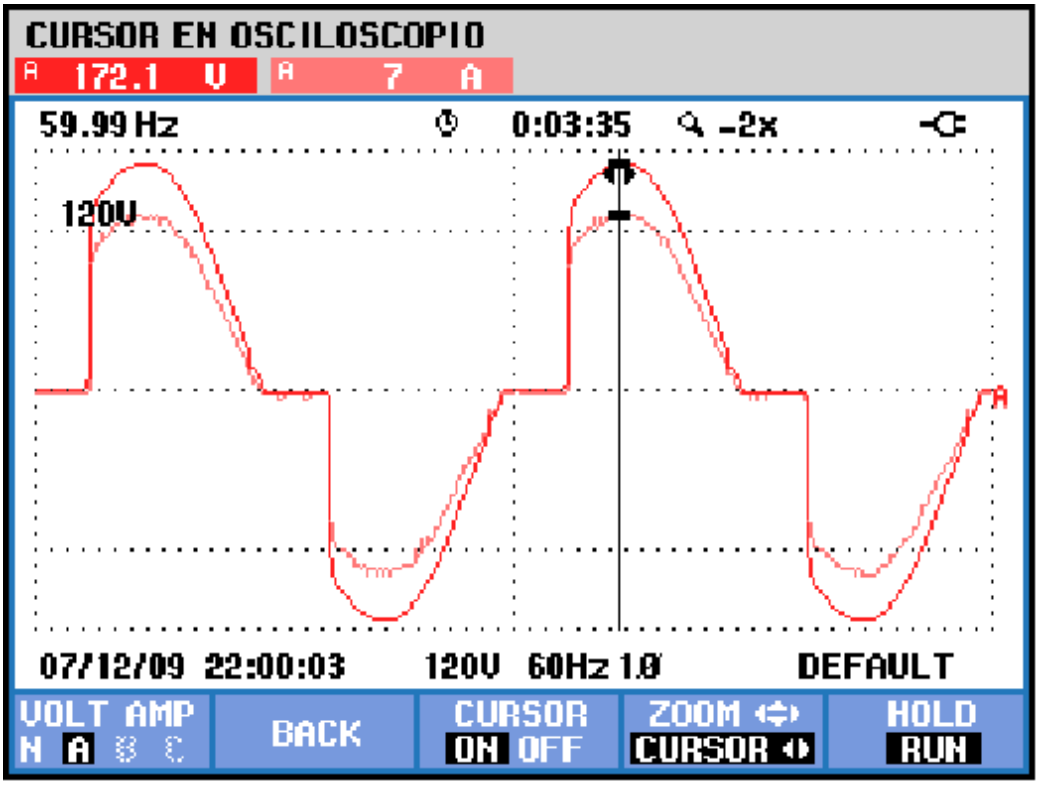

Figura 5.22. Formas de onda de voltaje y corriente para un ángulo de disparo de 141.17º.

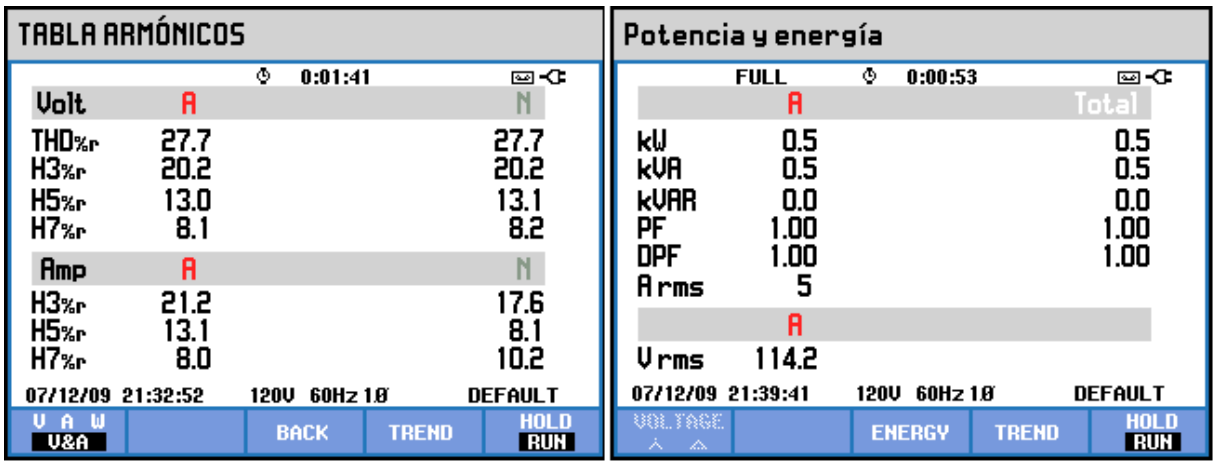

Figura 5.23. Valores obtenidos del PF, DPF y THD, para un ángulo de disparo igual a 141.17º.

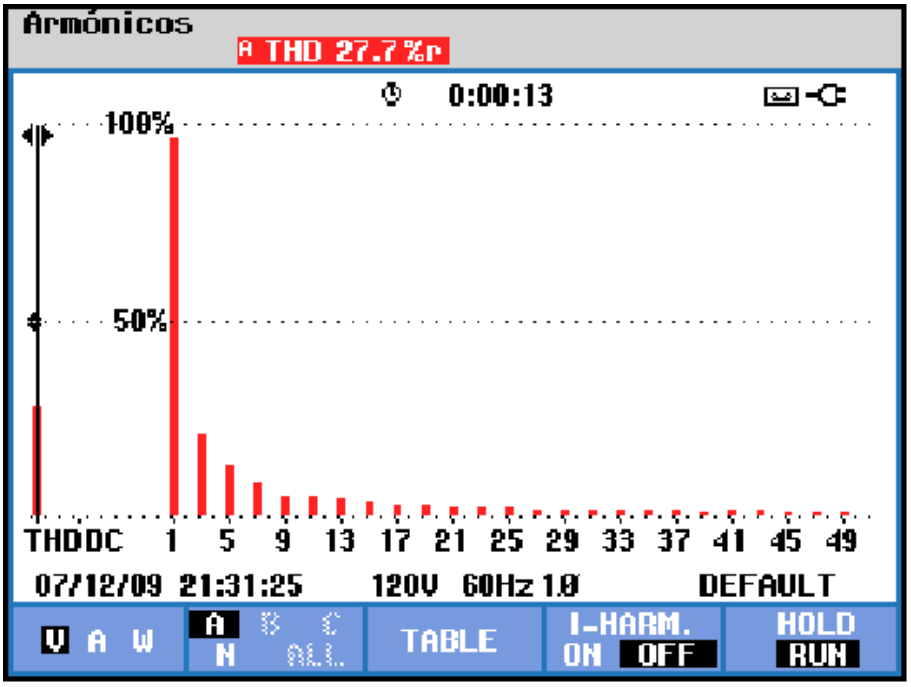

Figura 5.24. Armónicos en voltaje para un ángulo de disparo igual a 141.17º.

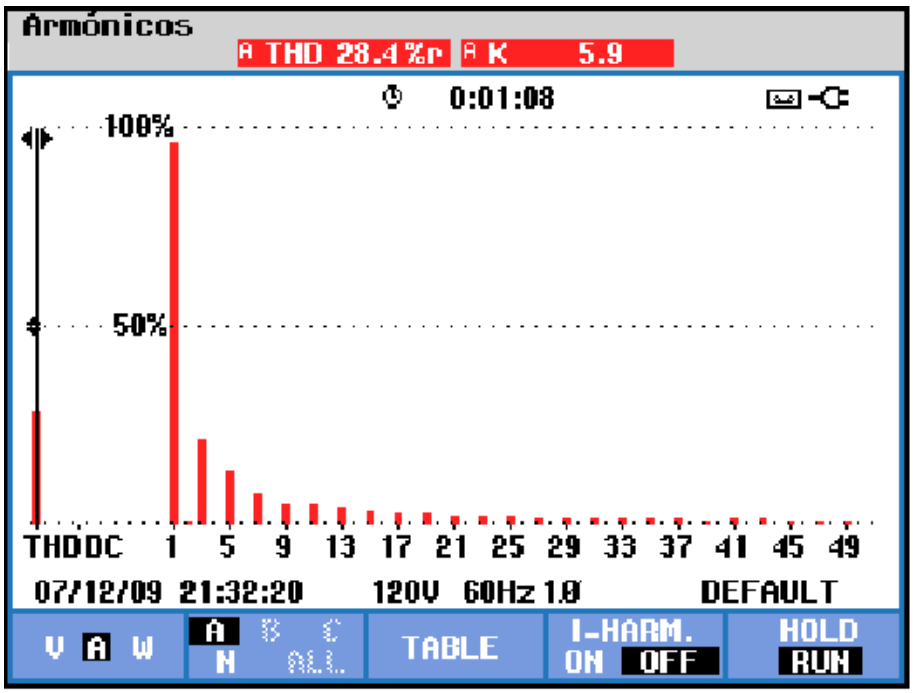

Figura 5.25. Armónicos en corriente para un ángulo de disparo igual a 141.17º.

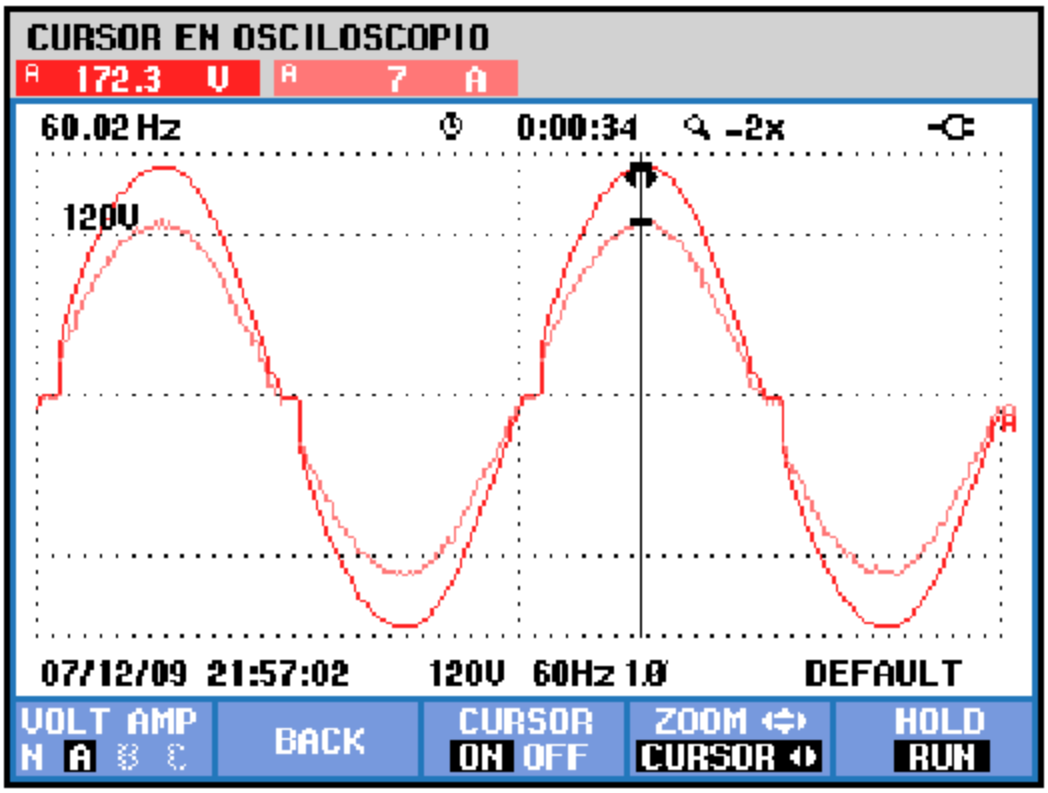

Figura 5.26. Formas de onda de voltaje y corriente para un ángulo de disparo de 176.47º.

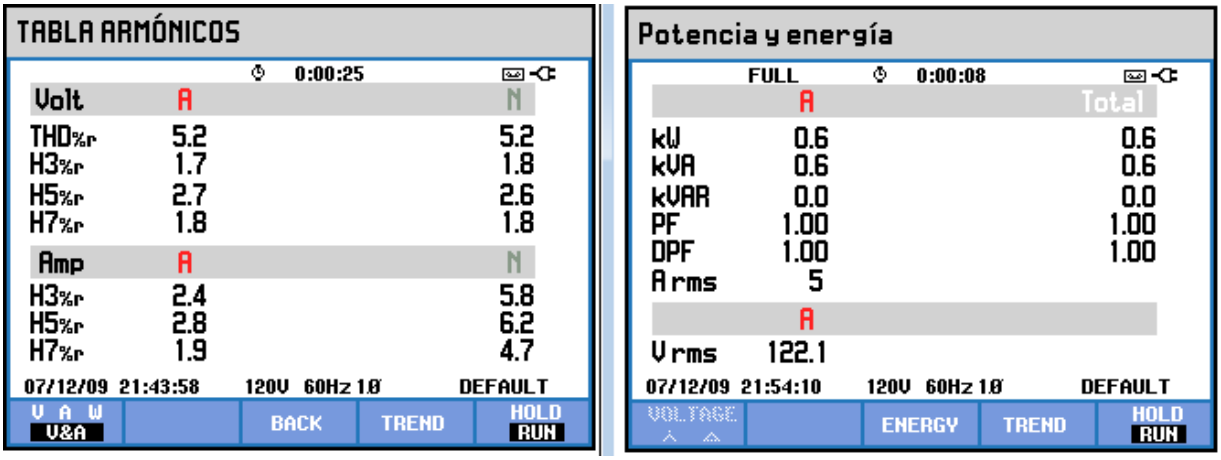

Figura 5.27. Valores obtenidos del PF, DPF y THD, para un ángulo de disparo igual a 176.47º.

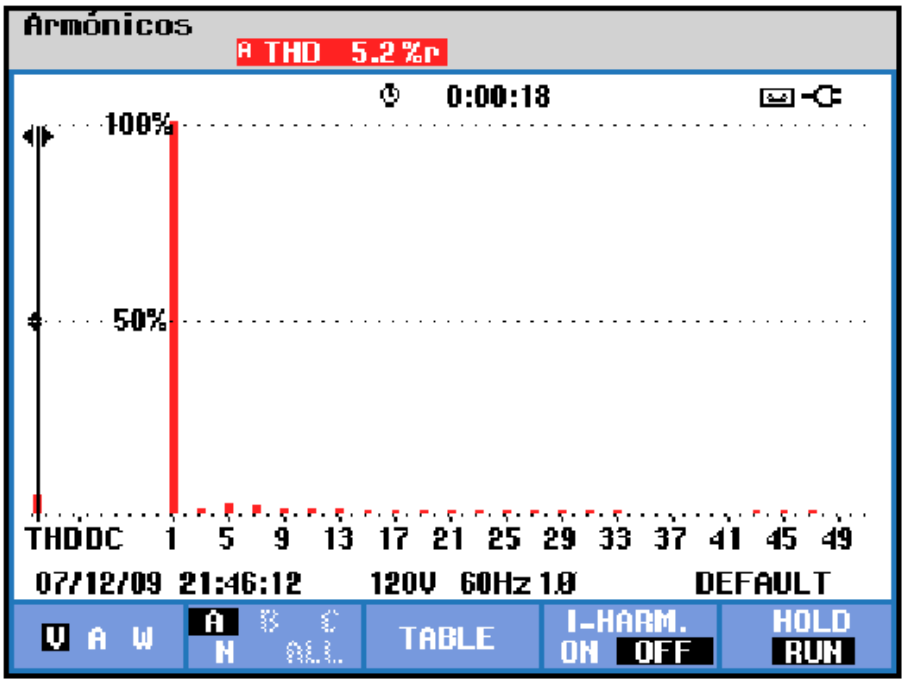

Figura 5.28. Armónicos en voltaje para un ángulo de disparo igual a 176.47º.

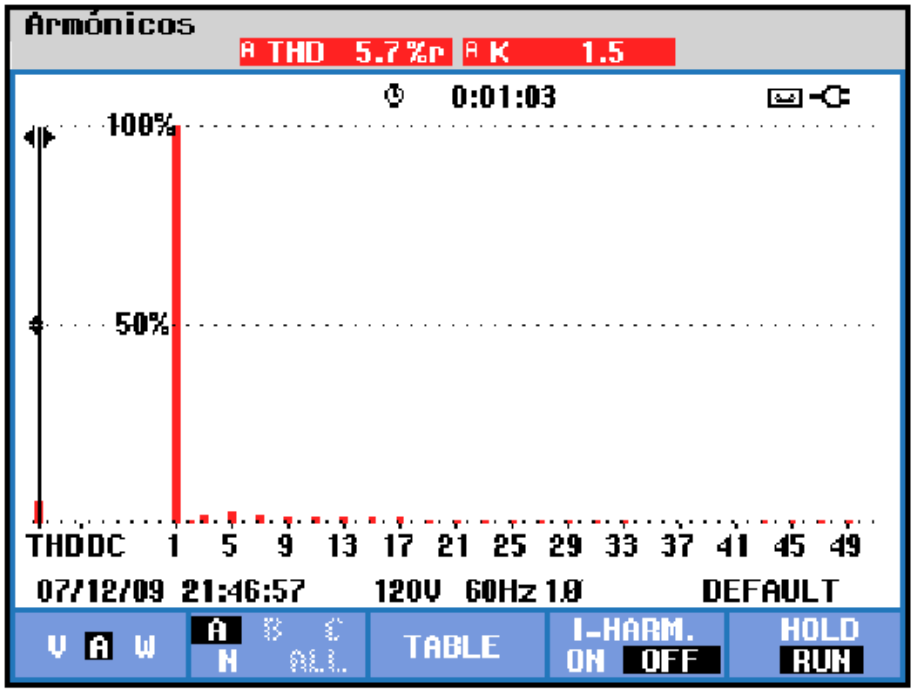

Figura 5.29. Armónicos en corriente para un ángulo de disparo igual a 176.47º.

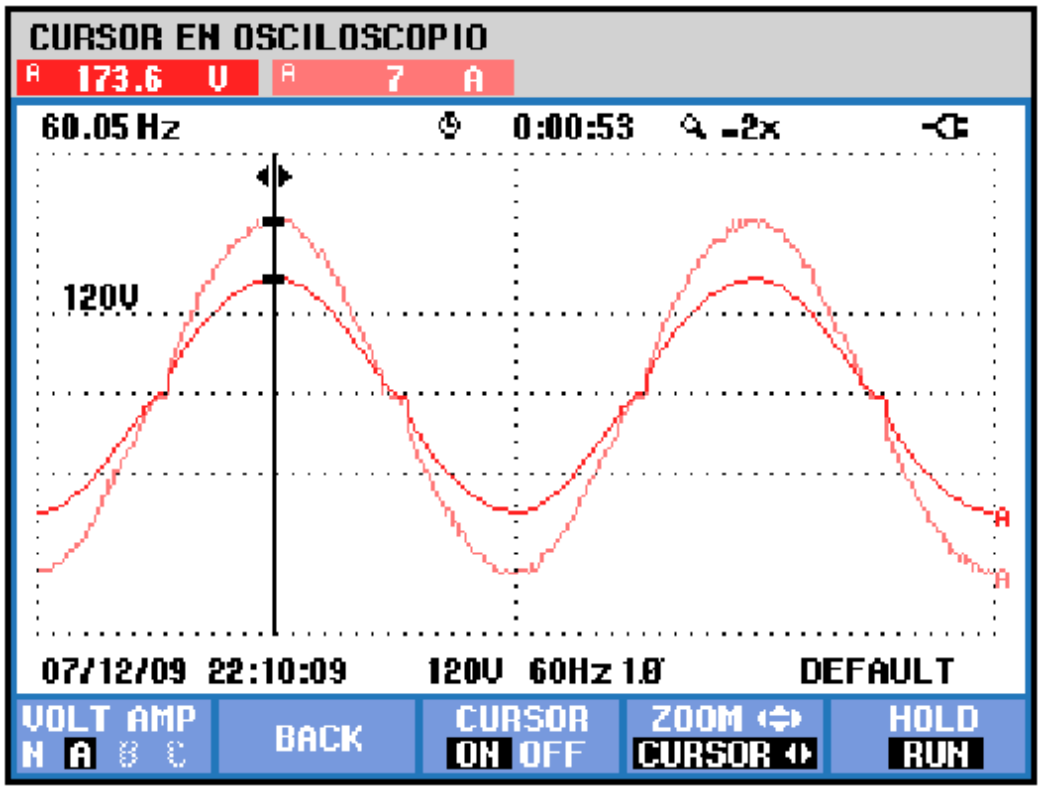

Figura 5.30. Formas de onda de voltaje y corriente para un ángulo de disparo de 180º.

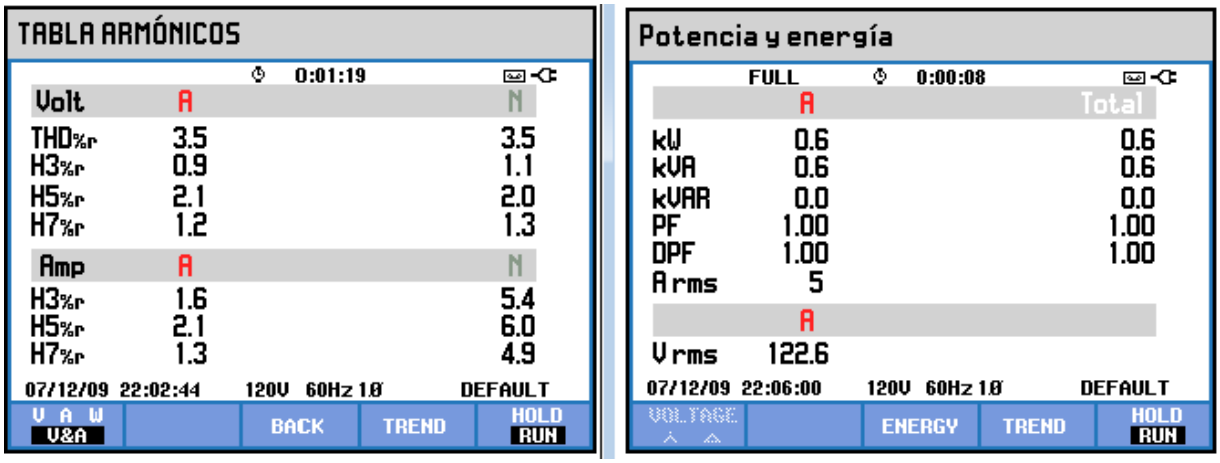

Figura 5.31. Valores obtenidos del PF, DPF y THD, para un ángulo de disparo igual a 180º.

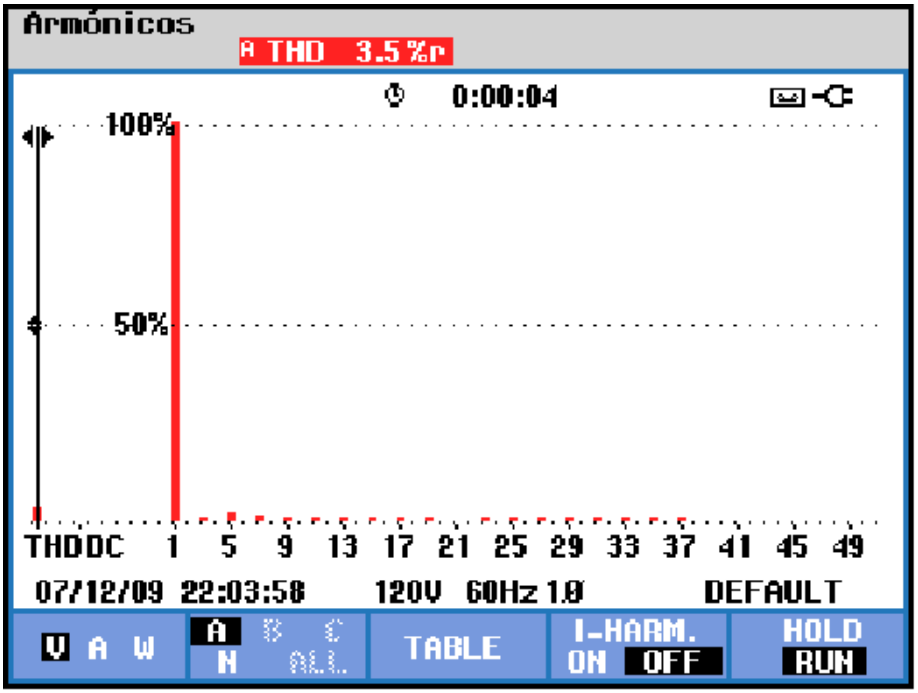

Figura 5.32. Armónicos en voltaje para un ángulo de disparo igual a 180º.

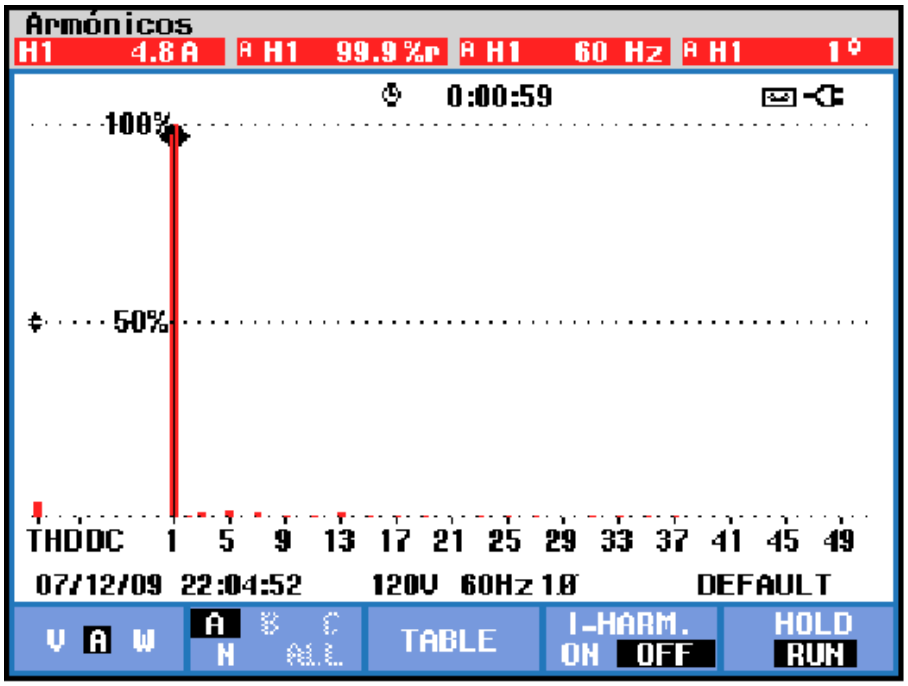

Figura 5.33. Armónicos en corriente para un ángulo de disparo igual a 180º.

Las gráficas anteriores se resumen en la Tabla 5.2.

Tabla 5.2. Valores de V, I, PF, DPF, THD y % de armónicos para diferentes ángulos de disparo.

Ángulo de disparo (º)	**Dato DMX**	**V (V)**	**I (A)**	**PF**	**DPF**	**THD (%)**	**V**			**I**		
							H3%	H5%	H7%	H3%	H5%	H7%
35.29	50	12.1	3	-	-	88.4	44.6	41.0	36.1	44.5	40.9	36.1
70.58	100	50.7	6	1	1	71.4	54.4	32.1	16.7	54.4	32.5	17.0
105.8	150	90.3	4	1	1	51.2	43.0	15.1	13.4	43.6	15.3	13.7
141.17	200	114.2	5	1	1	27.7	20.2	13.0	8.1	21.2	13.1	8.0
176.47	250	122.1	5	1	1	5.2	1.7	2.7	1.8	2.4	2.8	1.9
180	255	122.6	5	1	1	3.5	0.9	2.1	1.2	1.6	2.1	1.3

Finalmente, a continuación se presentan algunas fotografías del aspecto físico del SciDMX. En la Figura 5.34 se presenta el estuche donde está contenido el prototipo de 12 canales.

Figura 5.34. Prototipo del SciDMX de 12 canales.

La Figura 5.35a muestra la entrada y salida DMX del SciDMX, y la Figura 5.35b identifica las salidas de los dimmers, para el canal n y el canal n+1.

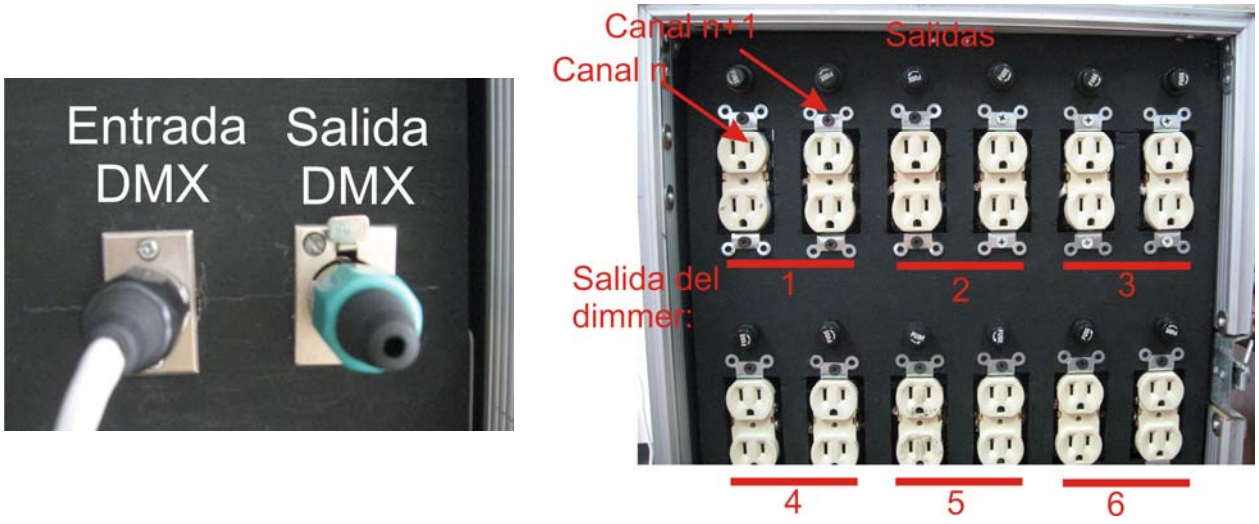

Figura 5.35. a) Entrada y salida DMX, b) Salidas de los dimmers.

La Figura 5.36 se muestra la tarjeta del controlador DMX.

Figura 5.36. a) Tarjeta del controlador DMX, b) Controlador y su fuente de alimentación.

La Figura 5.37 muestra la tarjeta del receptor DMX, y la tarjeta de la etapa de potencia para el dimmer.

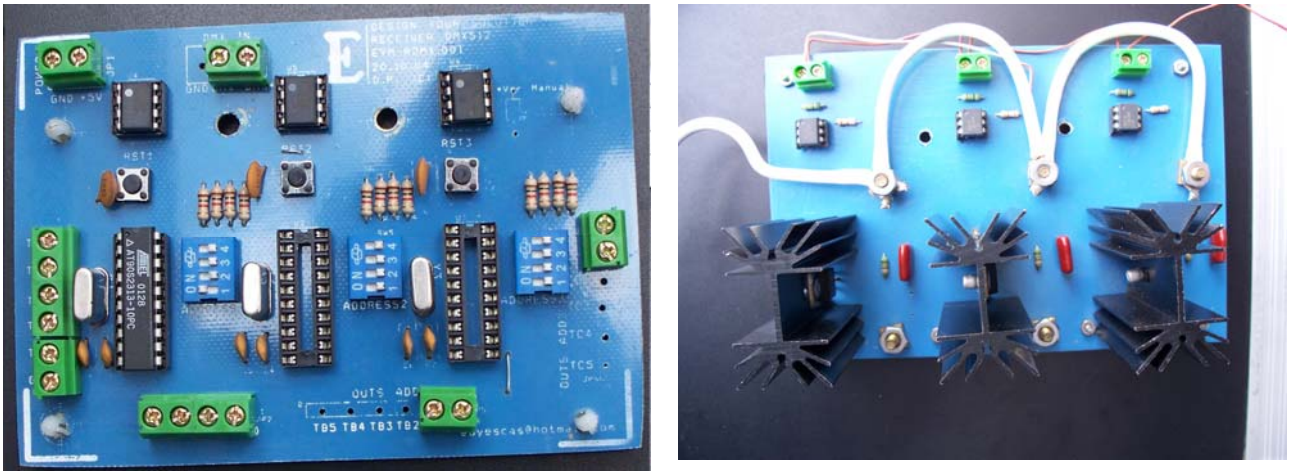

Figura 5.37. a) Tarjeta del receptor DMX, b) Tarjeta de la etapa de potencia del dimmer.

La Figura 5.38 muestra la tarjeta del circuito detector de cruce por cero y la fuente de alimentación del receptor DMX.

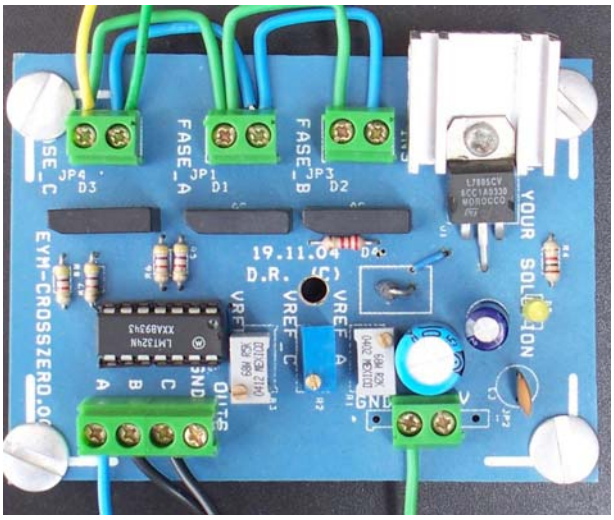

Figura 5.38. Tarjeta del circuito detector de cruce por cero, y fuente de alimentación del receptor DMX.

Por último, en la Figura 5.39 se presentan las tarjetas del receptor DMX.

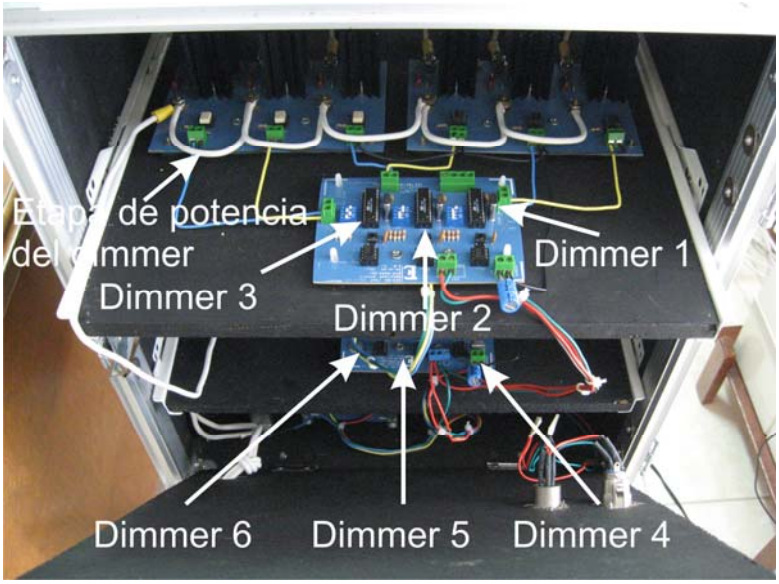

Figura 5.39. Tarjetas del receptor DMX y etapas de potencia.

Capítulo 6. Conclusiones y trabajos futuros

Conclusiones

El empleo de la metodología de desarrollo de los sistemas empotrados, ha sido de gran ayuda en el diseño y desarrollo del SciDMX, sobre todo en el aspecto de las especificaciones iniciales, éstas ayudan a los diseñadores a comunicar las ideas de diseño y a plantear los procedimientos para evaluar el correcto funcionamiento del sistema durante su ciclo de vida.

El trabajo presentado, proporciona las bases teóricas suficientes para hacer frente a las nuevas tendencias e investigaciones, realizadas en el campo de control de iluminación de eventos y espectáculos; ya que se diseñó y construyó un sistema de control de iluminación (SciDMX) basado en el protocolo DMX512, que permite mediante la PC, controlar la intensidad de las lámparas en forma programada y en tiempo real. La arquitectura del sistema esta formada por una combinación de software y de hardware de bajo costo tales como: microcontroladores y dispositivos de electrónica de potencia. Debido a la facilidad con la cual el SciDMX se puede instalar, programar y reproducir; este sistema se puede considerar como una *arquitectura base* que permitirá realizar trabajos de investigación como los presentados en [Von, 2001], [Ferreira, 2004], [Feldmeier, 2003] y [Jackman, 2005], para ayudar a estudiar y comprender los protocolos emergentes.

Como la funcionalidad del SciDMX se jerarquizó en 3 subsistemas, los cuales fueron modelados con diagramas UML, utilizando diagramas de casos de uso, diagramas de despliegue y diagramas de actividad, al momento de unir estos subsistemas se observó una buena integración, de tal manera que no se encontraron errores críticos en el diseño, y por lo tanto, no fue necesario cambiar la especificación de ninguno de ellos, lo cual nos permitió disminuir el tiempo de diseño. Además, la aplicación del lenguaje UML tanto para el diseño del programa de control residente en la PC como para el diseño del sistema empotrado SciDMX favoreció la comunicación con el cliente.

Se considera que los objetivos planteados inicialmente se han satisfecho, ya que se desarrollo un sistema de control para la iluminación de espectáculos basado en el protocolo DMX512 con capacidad de manejar 12 canales; relativamente económico según se muestra en el Anexo B.

Trabajos futuros

Algunas sugerencias para futuros desarrollos que tomen como base al presente trabajo son:

Las nuevas versiones del programa de control, pueden contemplar la creación de nuevas clases de objetos; e incluir una base de datos de accesorios (dimmers, luces inteligentes, etc.) comerciales para que el mapeo y uso en el programa de control sea más cómodo y rápido de emplear.

Actualmente, el puerto paralelo aún es empleado, sobre todo en las computadoras portátiles, pero se desconoce si éste seguirá permaneciendo en las nuevas generaciones de las computadoras de escritorio,

tal como ha sucedido con el puerto serie, por tal razón se sugiere que los trabajos futuros que estén basados en esta arquitectura contemplen que la comunicación entre la PC y el controlador DMX sea con interfaz USB, sin dejar la opción del puerto paralelo.

Los resultados obtenidos permiten observar que el diseño del dimmer resulto ser simple, debido a que es posible regular la intensidad de la lámpara desde 0 hasta el 100% cambiando el ángulo de retardo del disparo del Triac en función del dato DMX recibido. Sin embargo, el porcentaje de distorsión armónica es alto para los ángulos de disparo menores a 176.47º. Lo que sugiere un cambio de técnica en el control de disparo del Triac, utilizando como dispositivo de manejo de potencia al IGBT, y aplicar como técnica la de control por ángulo de fase invertida o PWM sinusoidal.

Anexo A. Acta de publicación

Diseño e implementación de un sistema de control de iluminación DMX512

I. Salinas, E. Yescas
División de Estudios de Postgrado en Electrónica y Computación
Universidad Tecnológica de la Mixteca
Carretera a Acatlima Km. 2.5 Huajuapan de León, Oaxaca
isalinas@mixteco.utm.mx, yescas@mixteco.utm.mx

Resumen

El presente trabajo describe el diseño y construcción de un sistema didáctico de control para la iluminación de escenarios, basado en el protocolo DMX512 mediante la aplicación de los microcontroladores AT90S2313 de Atmel. El sistema permite a través de una computadora personal (PC) controlar en tiempo real la intensidad de las lámparas conectadas. El sistema está formado por los siguientes módulos: Una PC, un controlador DMX512, nueve receptores y nueve dimmers digitales. El algoritmo del controlador desarrollado cumple con las especificaciones del protocolo DMX512. El diseño del dimmer digital con triac resultó ser simple y efectivo. Este sistema se diseño con componentes económicos y de fácil adquisición proporcionando una herramienta práctica para el estudio de los protocolos emergentes y el soporte de nuevas aplicaciones en la industria del espectáculo.

Abstract

The present work describes to the design and construction of a didactic system of control for the illumination of scenes, based on protocol DMX512 by means of the application of microcontrollers AT90S2313 of Atmel. The system allows through a personal computer (PC) to control in real time the intensity of the connected lamps. The system is formed by the following modules: A PC, a controller DMX512, nine receivers and nine dimmers digitals. The algorithm of developed controller fulfills the specifications of the protocol DMX512. The design of digital dimmer with triac resulted to be simple and effective. This system designed with economic components and of easy acquisition providing itself a practical tool for the study of the emergent protocols and the support of new applications in the industry of the spectacle.

1. Introducción

Hace algunas décadas, los centros de entretenimiento cultural y de diversión como teatros, museos, discotecas, etcétera, no contaban con sistemas automatizados para iluminar sus escenarios, no existían reguladores de intensidad de las luces, éstas solo se encendían o se apagaban cuando se requería. Esto significaba un gasto de energía y era imposible hacer que la iluminación del ambiente fuera agradable en áreas que así lo requerían. Además, los técnicos operaban manualmente las luces, exponiéndose a situaciones de riesgo. El control de intensidad de las luces, sirve para crear diferentes ambientes dentro de los escenarios y constituye un elemento fundamental en la determinación de la calidad del espectáculo. Recientemente han surgido diferentes controles para luces inteligentes robotizadas controladas por computadoras con diferentes programas de movimientos colores y figuras [URL 1].

El diseño de estos controladores involucra diferentes áreas de la electrónica como: microcontroladores (μC´s), dimmers, electrónica de potencia, programación y telecomunicaciones. Sin embargo, muchos de nuestros centros docentes, no cuentan con un equipo especializado para realizar prácticas de este tipo y en otros se ha optado por utilizar sistemas industriales reales, con el inconveniente de que los alumnos no pueden visualizar y comprender lo que ocurre en las diferentes partes del sistema. Por tal motivo, en el presente trabajo mostramos un diseño completo, de un sistema de control de luces inteligentes cuyo comportamiento es similar al de cualquier sistema DMX512, con la ventaja de poderse reproducir en el laboratorio con dispositivos comerciales de bajo costo, lo cual brinda una gran ayuda a los estudiantes.

El diseño desarrollado, tiene por objetivo contribuir al aprendizaje de los alumnos de ingeniería electrónica y áreas afines; en los principios de funcionamiento del protocolo de comunicación DMX512, y el diseño de dimmers digitales.

El protocolo de transmisión de datos Digital MultipleX, abreviado como DMX512 o simplemente DMX [E1], se utiliza para controlar la iluminación de escenarios, se basa en el estándar internacional RS485. Fue desarrollado con el propósito de obtener un protocolo de comunicación estándar y eficiente que enlazará

Proceedings of the 2006 International Conference on Electronic Design ISBN 968-9085-01-8 196

Anexo B. Integración del costo del SciDMX

Integración del costo del SciDMX

La siguiente sección describe la integración de costos del SciDMX y se realiza una comparación de precios con un sistema comercial Auto Lighting System.

La contabilidad de costos, se realizó contemplando la materia prima, la mano de obra y su costo social, los gastos indirectos y la depreciación del equipo y herramientas, para la producción de 1000 unidades. Estos cálculos se muestran en la siguiente Tabla B-1:

Tabla B-1. Contabilidad de costos del SciDMX.

Concepto	Cantidad	Observación
Materia prima	$223.43	Controlador, Tabla B-3
	$1693.00	Receptor/dimmer Tabla B-4
	$1377.792	Etapa de potencia Tabla B-5
Mano de obra	$128.00	Considerando un salario de un técnico de $3000 + el 28% de costo social.
Gastos indirectos	$138.12	Luz, agua y otros.
Depreciación	$2.6	
Total	$3562	
Gastos de venta y operación	$157	
Total de costos y gastos	$3517	
Utilidad	$1033.32	Incluye el porcentaje de utilidad y los impuestos correspondientes.
Precio de venta	$4553.27	

El sistema de control de iluminación del auditorio de la Universidad Tecnológica de la Mixteca tiene un costo de $11,000, el cual está formado por tres dimmers y una consola clásica, que son usados para 72 focos dicroicos de 50 W/12V.

Los dimmers son de 4 canales marca Auto Lighting, tienen un precio de $3,0000 c/u, para tener 12 canales como el SciDMX se requieren de 3 unidades, resultando un total de $9,000 pesos. La consola clásica utilizada en el auditorio es de la marca Prolight tiene un valor de $2000. Mientras que el sistema SciDMX diseñado en este trabajo es de 12 canales y tiene un precio de producción en serie aproximado de $**4553.27**, con lo cual se demuestra que en realidad es un sistema económico. Las características y precios de los dimmers y la consola, se presentan en la Tabla B-2 y Tabla B-3, respectivamente.

Tabla B-2. Precios y características de dimmers.

Características	**Dimmer**	**Dimmer y consola**
Modelo	Dimpack MPX-4800	SciDMX
Marca	Auto Lighting System	---
Precio dimmer	$ 3000	
N. de canales por dimmer	4	12
Potencia por canal	1200 Watts	1200 Watts
Consumo por canal	10 A	10 A
Alimentación	120 V	120 V
U. requeridas	3	1
Costo total	$11000	$4553.27

Tabla B-3. Precios y características de consola clásica y de computadora.

Características	**Consola basada en PC**	**Consola clásica**
Modelo		DIM-P8
Marca	Prolight	Auto Lighting System
N. de canales	Hasta 512	12
Precio	$12000	$ 2000

A continuación se presentan la lista de precios de la materia prima utilizada en cada uno de los subsistemas, la cual es tabulada para producir 1 unidad, 10 y 100 unidades (u).

Tabla B-3. Lista de material para el controlador DMX.

Cant.	Dispositivo	Precio					
		x 1 dispositivo	Para producir 1 sistema	x 10 dispositivos	Para producir 10 sistemas	x 100 dispositivos	Para producir 100 sistemas
1 u.	Microcontrolador AT90S2313	29.7	29.7	26.78	26.78	13.45	13.45
1 u.	Cristal 4 MHz	27.8	27.8	25	25	12.75	12.75
1 u.	Transceptor SN75176	6	6	6	6	5.47	5.47
1 u.	C. I. Buffer 74245	7.66	7.66	7.66	7.66	6.89	6.89
1 u.	Regulador 7805	5.09	5.09	5.09	5.09	4.59	4.59
1 u.	Circuito impreso	80	80	60	60	50	50
1 u.	Conector DB25	6.957	6.957	6.957	6.957	6.261	6.261
1 u.	Conectores dobles	8	8	8	8	3.22	3.22
1 u.	Conector XLR hembra, 3 pines	21	21	13.8	13.8	10.93	10.93
8 u.	Resistores	0.696	5.568	0.696	5.568	0.626	5
2 u.	Capacitores electrolíticos	2.609	5.218	2.609	5.218	2.348	4.696
6 u.	Capacitores cerámicos	1.739	10.43	1.739	10.43	1.565	9.39
3 u.	Bases para C.I.	2.6	7.8	2.6	7.8	2.34	7.02
2 u.	Switch	1.739	3.478	1.739	3.478	1.565	3.13
1 u.	Caja de plástico	39	39	26.45	26.45	20.7	20.7
1 u.	Adaptador de corriente 9V	87	87	60.95	60.95	51.18	51.18
4 m	Cable calibre 22	4	16	4	16	2.19	8.76
	TOTAL		$366.7		$295.2		$223.43

Tabla B-4. Lista de material para el dimmer DMX de 12 canales.

Cant.	Dispositivo	Precio					
		x 1 dispositivo	Para producir 1 sistema	x 10 dispositivos	Para producir 10 sistemas	x 100 dispositivos	Para producir 100 sistemas
6 u.	Microcontrolador AT90S2313	29.7	178.2	26.78	160.68	13.45	80.7
6 u.	Cristal 4 MHz	27.8	166.8	25	150	12.75	76.5
6 u.	Transceptor SN75176	6	36	6	36	5.47	32.82
1 u.	Regulador 7805	5.09	5.09	5.09	5.09	4.59	4.59
1 u.	Circuito impreso	90	90	70	70	60	60
6 u.	Dip-switch	11	66	6.21	37.26	4.6	27.6
19 u.	Conectores dobles	8	152	8	152	3.22	61.18
1 u.	Conector XLR macho, 3 pines	17	17	10.7	10.7	8.63	8.63
24 u.	Resistores	0.696	16.7	0.696	16.7	0.626	15.02
6 u.	Capacitores cerámicos	1.739	10.43	1.739	10.43	1.565	9.39
12 u.	Bases para C.I.	2.6	31.2	2.6	31.2	2.34	28.08
6 u.	Switch	1.739	10.43	1.739	10.43	1.565	9.39
1 u.	Rack para uso pesado	1000	1000	900	900	850	850
50 m	Cable UTP	8	400	8	400	8	400
2 u.	Conector XLR 3 pines	20	40	13.8	27.6	10.24	20.48
8 m	Cable calibre 10	22	176	21	168	20	
4 m	Cable calibre 22	4	16	4	16	2.19	8.76
	TOTAL		$2401.85		$2202		$1693

Tabla B-5 Lista de material para la etapa de potencia del dimmer.

Cant.	Dispositivo	Precio					
		x 1 dispositivo	Para producir 1 sistema	x 10 dispositivos	Para producir 10 sistemas	x 100 dispositivos	Para producir 10 sistemas
12 u.	Optoacoplador MOC3010	9.565	114.78	9.565	114.78	8.609	103.308
12 u.	Triac	24.34	292.08	21.91	262.92	10.11	121.32
12 u.	Choke	80	960	69	828	65	780
4 u.	Circuito impreso	90	360	70	280	60	240
12 u.	Conectores dobles	8	96	8	96	3.22	38.64
24 u.	Resistores	0.696	16.704	0.696	16.704	0.626	15.024
12 u.	Capacitores cerámicos	1.739	20.868	1.739	20.868	1.565	18.78
12 u.	Bases para C.I.	2.6	31.2	2.6	31.2	2.34	28.08
24 u.	Terminales	1.5	36	1.2	28.8	0.8	19.2
24 u	Tornillos	1	24	0.8	19.2	0.56	13.44
	TOTAL		$1951.632		$1698.472		$1377.792

u. – Unidades, m.- Metros

Anexo C. Comparación funcional del SciDMX con sistemas comerciales.

Comparación funcional del SciDMX con sistemas comerciales.

La tabla C-1 presenta la comparación de funciones que realiza el sistema de control de iluminación SciDMX con respecto a la consola computarizada y a la consola clásica de las marcas LightControl, Auto Lighting Systems, respectivamente.

Tabla C-1. Comparación de funciones de sistemas de control de iluminación

Características	**SciDMX**	**Consola computarizada**	**Consola clásica Auto Lighting System**
Protocolo DMX512	Si	Si	Si
Regulación de intensidad	Si	Si	Si
Encendido-Apagado directo	Si	Si	No
Sincronización de Luces	Si	No	No
Función Golpe	Si	No	No
Función fade-in	Si	Si	No
Función fade-out	Si	Si	No
Programación y reproducción de secuencias	Si	Si	Si
Secuencia de encendido-apagado de luces	Si	Si	No
Secuencia de intensidad	No	Si	Si
Control de luces robóticas	No	Si	No
Guardar secuencias en archivo	Si	Si	No

La versión actual del programa de control de SciDMX proporciona las funciones básicas; como se puede apreciar en la Tabla C-1, la consola computarizada tiene más funciones que el SciDMX, sin embargo, estas funciones se pueden agregar en nuestro sistema como trabajos futuros, derivados del presente desarrollo.

Referencias

[Arichika et al, 2004] Y. Arichika, K. Araki, “Reusable formal specification for embedded Systems”, Proceeding of the 11th Asia-Pacific Software Engineering Conference (APSEC´04) 1530-1362/04.

[Atmel, 2002] Atmel, Application Note AVR325: High speed interface to host EPP parallel port, Atmel, Rev. 2506A, 2002.

[Axelson, 1996] J. Axelson, Parallel Port *Complete, Programming, Interfacing & Using the PC´s Parallel Printer Port*, Lakeview Research Edition,1996.

[Axelson, 1999] J. Axelson, *Serial Port Complete, Programming and Circuits for RS-232 and RS-485 links and Networks*, Lakeview Research Edition 1999.

[Axelson, 2001] J. Axelson, *USB Complete: Everything you need to develop custom USB peripherals*, Lakeview Research Edition, 2001.

[Berger, 2002] A. Berger, *Embedded System Design: An Introduction to Process, Tools and Techniques*, CMP Books, 2002.

[Boehm, 1976] B. Boehm, “Software Engineering”, IEEE Transactions on Computer, 1976.

[Bohn et al, 2002] J Bohn, W. Damm, H. Wittke, J. Klose, A. Moik, “Modeling and validating train systems applications using statemate and live sequence charts”, Integrated Design and process technology IDFT-2002.

[Booch et al, 1999] G. Booch, J. Rumbaugh, I. Jacobson, *El lenguaje unificado de modelado*, 1a Edición. Addison Wesley, 2004.

[Brooks, 2006] R. Brooks, “Some thoughts conceptual modeling: performance, complexity and simplification”, Proceedings of the 2006 OR Society Simulation Workshop.

[Bunker, 2004] A. Bunker, G. Gopalakrishnan, “Formal hardware specification languages for protocol compliance verification”, ACM Transactions on Design Automation of Electronic Systems, Vol. 9, No. 1, January 2004.

[Costa et al, 2001] C. A. Costa, A. Sousa, F. Ferreira, “Lighting Design: A goal Based Approach Using Optimisation”, 2001.

[Couedic, 2000] M. Couedic, *Circuitos integrados para tiristores y triacs*, Editorial Alfaomega Marcombo, Barcelona, España, 2000.

[Damm et al, 2008] M. Damm, J. Haase, C. Grimm, F. Herrera, E. Villar, “Bridging MoCs in SystemC Specifications of Heterogeneous Systems”, Hindawi Publihing Corporation, EURASIP Journal of Embedded Systems, Volume 2008, Article ID 738136, doi: 10.1155/2008/738136.

[Dasiewicz, 2001] P. Dasiewicz, “Microcontroller Based Multichannel Light Dimmer”, University of Waterloo - Department of Electrical and Computes Engineering, 2001.

[Douglass, 2000] B. Douglass, *Real-Time UML,* 2a Edición. Addison Wesley, 2000.

[Edwards, 2003] S. Edwards, “Design languages for embedded Systems”, Columbia University, NY, May 2003.

[Feldmeier, 2003] M. Feldmeier, J. A. Paradiso, M. Malinowski, “Large group musical interaction using disposable wireless motion sensors”, Massachusetts Institute of Technology, Master Thesis, 2003.

[Ferreira, 2000] L. Ferreira Menezes, “Descripción y Aplicación del Protocolo Multiplexado Serial de 8 bits – DMX512”, Universidad Estadual de Campinas – Unicamp, 2000.

[Ferreira, 2004] L. Menezes Ferreira, “Controle Automatizado para Scanners de Luz”, Tesis, Universidade Estadual de Campinas – UNICAMP, Facultade de Engenharia Mecânica, Departamento de Projeto Mecânico, 2004.

[Gadre, 2001] D. V. Gadre, *Programming and customizing the AVR microcontroller*, Editorial Mc Graw-Hill, 2001.

[Gajski et al, 1985] D. Gajski, F. Vahid, “Specification and design of embedded hardware-software systems”, University of California, IEEE Design and Test of computers, ISSN 0740-7475, Spring, 1995.

[Gajski et al, 1994] D. Gajski, Frank Vahid, Sanjiv Narayan, Jie Gong, *Specification and Design of Embedded Systems*, Prentice Hall, 1994.

[Gajski, 1996] D. Gajski, “Methodology is the future”, Proceedings of the IEEE Asia Pacific Conference on Circuits and Systems ´96, November 18-21, 1996, Seoul, Korea.

[Harel et al, 1998] D. Harel, M. Politi, *Modeling Reactive Systems with Statecharts*, Mc Graw-Hill, 1998.

[Henzinger et al, 2000] T. A. Henzinge, J. Sifakis, “The Embedded Systems Desing Challenge”, Activity Report, Verimag Research center in embedded systems, Gieres, France, 2000.

[Huntington, 2000] J. Huntington, *Control systems for live entertainment*, 2a edición, Focal Press, 2000.

[IEEE STD-830, 1998] IEEE-STD-830-1998: IEEE recommended practice for software requirements specifications, Software Engineering Standards Committee of the IEEE Computer Society, USA, E-ISBN: 0-7381-0448-5.

[Jackman, 2005] S. Jackman, “Ethernet Communication in Lighting Control”, Tesis Ingeniería – Universidad Simon Fraser, 2005.

[Kocik et al, 2002] R. Kocik, “A methodology to reduce the design lifecycle of real-time embedded control systems”, Yves SOREL, 2002.

[Kopel et al, 2000] K. Kopel, J. Sand, “DMX512 Programmable Theater Lighting Controller”, Universidad Bradley, 2000.

[Kumar, 1993] R. Kumar, “Co-synthesis of hardware and software for digital embedded systems”, 1993, Stanford University.

[Latella et al, 2000] D. Latella, I. Majzik, M. Massink, “Towards a formal operational semantics of UML statechart diagrams”, ESPRIT Project n. 27439m- HIDE, 2000.

[Lavagno, 1987] L. Lavagno, S. Edwards, E. A. Lee, A. Sangiovanni, “Design of embedded systems: Formal, Models, Validation and Synthesis”, Proceeding of the IEEE, Vol. 85, No. 3, March1997.

[Martín, 2002] G. Martín, “UML for embedded systems specification and design: motivation and overview”, Proceedings of the 2002 Design, Automation and Test in Europe Conference and Exhibition (DATE´02).

[Marwedel, 2006] P. Marwedel, *Embedded System Design*, University of Dortmund, Germany, Springer, 2006.

[Mobsby, 2005] N. Mobsby, *Practical DMX*, Entertainment Technology Press, 2005.

[Noergaard, 2005] T. Noergaard, “Embedded system architecture, A comprehensive guide for engineers and programmers”, Elsevier, 2005.

[O´Sullivan et al, 2004] D. O´Sullivan, T. Igoe, *Physical computing, Sensing and Controlling the Physical World with Computers*, Thomson Course Technology PTR, 2004.

[Phillips, 1994] Phillips Semiconductor, Power semiconductor Applications Laboratory, Thyristor and Triacs, 1994.

[Putten, 2001] V. D. Putten, J.P.M. Voeten, M.C.W. Geilen, M.P.J. Stevens, “Systems Level Design Methodology”, Section of information and communications systems, Eindhoven University of Technology, The Netherlands.

[Microchip, 1997] Microchip, Application Note: PICDIM Lamp Dimmer for the PIC12C508, Microchip Technology Inc, 1997.

[Muhammad, 2001] Rashid H. Muhammad, Power Electronics Handbook, Academic Press, 2001.

[Ramírez, 2001] J. F. Ramírez, Aprenda Visual Basic practicando, Editorial Prentice Hall, Primera Edición, 2001.

[Randall, 2002] J. Randall, “Real-time Lighting System for Large Group Interaction”, Massachusetts Institute of Technology, 2002.

[Reichmann et al, 2001] C. Reichmann, D. Gebauer, K.D. Müller-Glaser, “Model Level coupling of heterogeneous embedded systems”, ITIV, University of Karlsruhe, Germany, 2001.

[Reid, 1998] F. Reid, Discovering stage lighting, Focal Press, 2ª Edición, 1998, ISBN 0240515455.

[Rodríguez, 2007] D. Rodríguez H., “Análisis, diseño y mantenimiento del software, fase de especificación”, Nov. 2007.

[Rosenberg, 2003] A. Rosenberg, “Real-Time DMX512 DMX FACTORY, Intelligent controller”, Stetson University.

[Royce, 1970] W. W. Royce, “Managing the development of large software systems: Concepts and techniques”, Proceedings IEEE WESCON, August 1970.

[Rubinstein et al, 2003] F.Rubinstein, S. Treado, P. Pettler, “Standardizing Communication between Lighting Control Devices – A Role for IEEE P1451”, IEEE.

[Russo et al, 1999] M. F. Russo, M. M. Echols, “Automating Science and Engineering Laboratories with Visual Basic”, Wiley Interscience.

[Scott, 1996] S. Scott, Technical Theatre Handbook, Worcester Polytechnic Institute, 1996.

[Shin et al, 1998] J. Shin Young, J. MacDonald, “Design and Specification of Embedded Systems in Java Using Successive, Formal Refinement”, DAC 98, June 15-19, 1998 San Francisco, CA USA. ISBN 1-58113-049-x/98/06.

[Sgroi et al, 2000] M. Sgroi, L. Lavagno, A. Sangiovanni-Vincetelli, “Formal models for embedded system design”, IEEE Design and test of Computers, April-June 2000.

[Simpson, 2003] R. Simpson, Lighting Control: Technology and Applications, Focal Press, ISBN-10: 0240515668, ISBN-13: 978-0240515663 May 2003.

[Solá et al, 2003] R. Solá, J. A. Aragonés Cervera, Xavier, Diseño de circuitos y sistemas integrados, Ediciones UPC, 2003.

[Sperbe, 2001] M. Sperbe, “Developing a stage lighting system from scratch”, International Conference on Functional Programming, Florence Italy, 2001.

[Smith, 2005] J. Smith, J. Speakes, M. H. Rashid, “An overview of the moderns light dimmer: Design, Operation and Application” IEEE 2005.

[Stalling, 1997] William Stalling, Data and Computer Communications, Editorial Prentice Hall, Quinta Edición.

[Vahid, 2000] F. Vahid, T. Givaigis, “Embedded system design a unified hardware/software approach”, Department of Computer Science and Engineering University of California, 2000.

[Villoria et al, 2002] A. Villoria, P. Chamorro, “Desarrollo de una herramienta de simulación de sistemas de comunicaciones ópticas”. Departamento de TSCIT, Universidad de Valladolid, 2002.

[Von, 2001] V. Von, “Computer-Assisted Lighting Design and Control”, Universidad Tübingen, Disertación, 2001.

[Wood et al, 1989] D. P. Wood, W. G. Wood, “Comparative evaluations of four specification methods for real-time systems”, December 1989.

[Yescas et al, 2006] Yescas M. E., Salinas P. I., “Diseño e implementación de un sistema de control de iluminación DMX512”, Proceedings of the 2006 International Conference on Electronic Design, pp 196-201, México (Veracruz), 21-23 de Noviembre, 2006, ISBN 968-9085-01-8.

[Zurawsky, 2005] Richard Zurawsky, Embedded system handbook, 1ª Edition, Publisher CRC, 2005.

Páginas Web

[URL 1] //es.encarta.msn.com/encyclopedia_761553217_2/Producción_teatral.html

[URL 2] http://lighteducation.com/article.php?sid=45, Página Web que proporciona un resumen de los protocolos de comunicación. Fecha de última consulta: 27 de febrero de 2006

[URL 3] UUwww.usitt.orgUU, Página Web de la asociación de profesionales de diseño, producción y tecnología de la industria del entretenimiento. Fecha de última consulta: 27 de febrero de 2006.

[URL 4] UUwww.esta.orgUU, Página Web de la asociación no lucrativa que representa a la industria de tecnología del entretenimiento. Fecha de última consulta: 27 de febrero de 2006.

[URL 5] www.epidataconsulting.com/tikiwiki/tiki-read_article.php?articleId=15

[E1.11-2004, 1990] American National Standard E1.11-2004 Entertainment Technology USITT DMX512-A, Asynchronous Serial Digital Data Transmission Standard for Controlling Lighting Equipment and Accessories. Última rev.: 1990. Cubre todo sobre la estructura del protocolo, tipos de conectores e información de conexionado. Publicado por la ESTA.