DISEÑO E IMPLEMENTACIÓN DE UNA TARJETA DE ADQUISICIÓN DE DATOSPOR BUS USB

NATHALIE HERNÁNDEZ PINEDA
IVONNE ANDREA MANTILLA GONZALEZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE CIENCIAS FISICOMECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES
BUCARAMANGA
2004

DISEÑO E IMPLEMENTACIÓN DE UNA TARJETA DE ADQUISICIÓN DE DATOSPOR BUS USB

NATHALIE HERNÁNDEZ PINEDA
IVONNE ANDREA MANTILLA GONZALEZ

Tesis de grado

Director
Msc. Jaime Barrero Pérez

Codirector
Ing. Jorge Eduardo Higuera P.

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE CIENCIAS FISICOMECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES
BUCARAMANGA
2004

Nota de aceptación:

_____

_____

_____

_____

_____

_____

Firma del calificador

_____

Firma del calificador

Bucaramanga, 30 de Junio de 2004.

“Porque este trabajo no es del orgullo de los hombres, sino es gozo del reino de los cielos.”

A Jesús Misericordioso.

“Porque es una bendición encontrar una palabra de apoyo y fortaleza; por todo el esfuerzo y permitirme llegar hasta aquí…”

A mis padres y mi hermana Dayra.

“Porque es una bendición encontrar el paraíso en esta vida terrenal…”

A mi amor Oscar Mauricio R.

“Porque es una bendición que a pesar de las contrariedades contar con la confianza y el ánimo para continuar…”

A mi Abuela, familiares y buenos amigos.

A tí nuestro aliciente, nuestro sueño…

L.M.R.M

Ivonne Andrea Mantilla G.

“A Dios por darme las fuerzas para seguir adelante”

“A mi familia por todo su apoyo y compresión”

“A mis amigos por toda su ayuda, colaboración y compañía”

“A mis profesores, por darme los cimientos para construir una carrera profesional”

Nathalie Hernández Pineda.

CONTENIDO

LISTA DE FIGURAS

Pag.

LISTA DE TABLAS

TITULO: **DISEÑO E IMPLEMENTACIÓN DE UNA TARJETA DE ADQUISICIÓN DE DATOS POR BUS USB**[*]

AUTORES: **IVONNE ANDREA MANTILLA GONZÁLEZ**
NATHALIE HERNÁNDEZ PINEDA[**]

PALABRAS CLAVE:
Bus Universal Serial, USB
Tarjeta de adquisición de datos
Inserción en caliente
Conectar y listo
Asíncrono,
LabVIEW.

DESCRIPCIÓN:

En la actualidad, las interfaces de comunicación son cada vez más rápidas y flexibles y permiten un manejo fácil por parte del usuario. Un ejemplo es el Bus Universal Serial (USB), una interfaz asíncrona que en los últimos años ha permitido diseñar y construir una gran variedad dispositivos que emplean este puerto de comunicación.

Este trabajo presenta el sistema de adquisición de datos Sad800-L, que utiliza el bus USB versión 1.1 de baja velocidad, cuyo diseño cuenta con cuatro entradas analógicas de tensión entre ?10 V no diferenciales, ancho de banda de 50 Hz por canal y resolución de 16 bits. Además, se usó un circuito de temporización independiente para el conversor analógico/digital que permite al microcontrolador ser utilizado como un dispositivo dedicado a la transmisión de datos a través de su módulo USB embebido.

El Sad800-L se conecta a un computador por el bus USB y envía las muestras de los cuatro canales a una tasa efectiva de 800 bytes por segundo, que corresponde a la máxima posible con esta versión del bus. Estas señales se visualizan en el dominio del tiempo y la frecuencia mediante una interfaz desarrollada en LabVIEW 7.0 y Visual Basic 6.

De esta forma, se tiene un sistema que permite “conectar y listo”, inserción en caliente, reduce la intervención del usuario para su configuración, y explora y aprovecha al máximo las prestaciones de este bus asíncrono, en su versión 1.1, lo cual es un precedente para futuros proyectos que utilicen las versiones más recientes del bus para desarrollar aplicaciones más acordes a sus características.

---

[*] Trabajo de grado

[**] Facultad de Ingenierías Fisicomecánicas. Escuela de Ingeniería Eléctrica, Electrónica y Telecomunicaciones. Director Jaime Guillermo Barrero Pérez.

TITLE: DESIGN AND IMPLEMENTATION OF A USB DATA ACQUISITION CARD*

AUTHORS: **IVONNE ANDREA MANTILLA GONZÁLEZ**
NATHALIE HERNÁNDEZ PINEDA**

KEYWORDS: Universal Serial Bus, USB
Data acquisition card
Hot insertion
Plug&Play
Asynchronous
LabVIEW.

DESCRIPTION:

At the present time, the communication interfaces are more and more quick and flexible and they allow an easy handling on the part of the user. An example is the Universal Serial Bus (USB), an asynchronous interface that has allowed to design and to build a great variety devices to use this communication port in the last years.

This work presents the system of acquisition of data Sad800-L that uses the USB version 1.1 low speed whose design has four analog voltage inputs, non differential, ±10V input range, 50Hz bandwidth per channel and 16-bit sampling. Also, an independent timing circuit was used for the ADC that allows to the microcontroller to be used as a device dedicated to the transmission of data through its embedded module USB.

Sad800-L is connected to a computer by USB and it sends the samples from the four input channels with an effective rate of 800 bytes per second that corresponds to the maximum possible in this version of the bus. These signals are visualized in time and frequency domain using an interface developed in LabVIEW 7.0 and Visual Basic 6.

This way, this is a plug&play system that allows hot insertion, reduces the user's intervention for its configuration, and it explores and it takes advantage of to the maximum the benefits of this asynchronous bus, in its version 1.1, that which is a precedent for future projects that use the most recent versions of the bus to develop applications in agreement to its characteristics.

---

* Degree work.

** Physics-mecanical Faculty. Electrical, Electronic and Telecommunications Engineering School. Jaime Guillermo Barrero Pérez, director.

INTRODUCCIÓN

El desarrollo e implementación de nuevos puertos de comunicación para los sistemas de cómputo durante los últimos años, han logrado cambiar la visión en cuanto manejo y empleo de periféricos se refiere.

El bus serial universal, USB[1], es una de las interfaces más populares que desde 1996 después de muchos estudios previos por parte de las empresas interesadas en el desarrollo de esta interfaz (Intel Corp., Microsoft Corp., entre otras), comenzó con las primeras especificaciones para la versión 1.0. En el año de 1.998 se consolida USB 1.1, el cual solucionaba muchos de los inconvenientes identificados en la versión anterior; y es en abril del año 2.000, cuando surgen las especificaciones actualmente usadas para USB 2.0.

A diferencia de las dos versiones anteriores, USB 2.0 cuenta con un incremento considerable en la velocidad de transmisión de la información y diferentes tipos de transferencias de datos que responden a las necesidades de los periféricos que se manejan en la actualidad, consolidándose como un estándar que se caracteriza por la flexibilidad y rapidez con que maneja el envío de datos, además de contar con ventajas como "conectar y listo" (Plug and Play), conexiones sencillas y gran capacidad de expansión. Estas razones han motivado a grandes empresas involucradas en el desarrollo de hardware y software de computadores y periféricos, a diseñar dispositivos encaminados que mejoren las prestaciones en los sistemas de cómputo.

Todo este adelanto tecnológico no es ajeno al campo de la electrónica; por el contrario, con el fin de mantener en vigencia líneas de trabajo como la

[1] Por sus siglas en inglés *Universal Serial Bus.*

instrumentación electrónica, control y automatización de procesos, bioingeniería, entre otras; se comienza el estudio e investigación del diseño de periféricos que manejen comunicación USB, desarrollando en este proyecto un sistema de adquisición de señales de tensión provenientes de una tarjeta de acondicionamiento para enviarlas a través del bus *USB* V1.1 a un computador, y presentarlas en una interfaz gráfica. Los resultados de este trabajo permiten marcar una pauta para futuros proyectos en la Escuela de Ingeniería Eléctrica, Electrónica y Telecomunicaciones.

El proceso de diseño y construcción del sistema de adquisición de datos por bus USB V1.1, comienza con el estudio previo de todo lo relacionado con las especificaciones y protocolo USB, la determinación y construcción de todas las herramientas necesarias para el desarrollo del sistema de adquisición a nivel de hardware y software, es decir, la selección de un controlador USB adecuado, la elección y configuración del sistema de conversión analógica/digital al igual que el microcontrolador empleado para la transmisión de datos por bus USB y por último la programación de la aplicación en alto nivel que permite la visualización de los datos obtenidos.

De esta forma, este libro presenta en el primer capítulo parte de las especificaciones USB V1.1 enfatizando en los dispositivos de baja velocidad y sus principales características. El capítulo 2 muestra el sistema de desarrollo implementado para programar y ejecutar rutinas en el microcontrolador MC68HC908JB8JP de Motorola, propuesto en el manual de referencia de dicho microcontrolador.

En el capítulo tres se hace una breve descripción de los controladores (*drivers)* para dispositivos USB, el software para la programación del microcontrolador y las herramientas que permiten hacer un seguimiento de los dispositivos y de las tramas de las transferencias USB.

El cuarto capítulo presenta el diseño del hardware de la tarjeta de adquisición de datos *Sad800-L*, describiendo las etapas que la constituyen y sus componentes. El diagrama de flujo implementado en el microcontrolador, los resultados obtenidos en la etapa de conversión analógica/digital y la transmisión de datos por USB se describen en el capítulo cinco.

Por último, se describe el funcionamiento de la Interfaz SAD800-L empleada para la adquisición y visualización de los datos provenientes de la Tarjeta SAD800-L.

1. ESPECIFICACIONES USB 1.1

1.1 GENERALIDADES

El USB (*Universal Serial Bus*) es una interfaz rápida y flexible para conectar periféricos a un computador (PC) o *host*. Según las especificaciones USB 1.1, un dispositivo USB puede emplear cuatro tipos de transferencia de datos y dos velocidades diferentes (baja velocidad y velocidad media). Las especificaciones USB 2.0 definen alta velocidad. Cuando un dispositivo es conectado a un computador o *host*, el sistema operativo debe responder a una serie de peticiones que le permiten al *host* conocer las características necesarias acerca del dispositivo, con el fin de establecer una comunicación con él.

1.1.1 Velocidades

Existen tres velocidades diferentes definidas en las especificaciones USB 2.0, las cuales se muestran en la Tabla 1. Para las especificaciones USB 1.1 solo existen la Velocidad Baja (*Low Speed*) y la Velocidad Media (*Full Speed*). En la tabla, también se muestra un paralelo entre la velocidad de transferencia de bits en el bus USB, y la tasa máxima de transferencia de datos teórica para cada velocidad.

Tabla 1. Tabla de velocidades USB.

VELOCIDAD	Tasa de transferencia de bits (Mbps)	Tasa de transferencia máxima de datos teórica
Low Speed (Velocidad Baja)	1.5	800 bytes/s
Full Speed (Velocidad Media)	12	1.2 Mbytes/s
High Speed (Velocidad Alta – USB 2.0)	480	53 Mbytes/s

Fuente: los autores.

1.1.2 Componentes de la comunicación por bus USB

Los componentes físicos de la interfaz USB son: el hardware controlador, los conectores, y los cables entre el *host* y uno o más dispositivos.

"El *host* es el *PC* (computador personal) que contiene dos componentes: el controlador del *host* y un *hub* raíz. Estos dos trabajan en conjunto para permitir al sistema operativo comunicarse con los dispositivos que se encuentran en el bus USB. El controlador del *host* le da formato a los datos para transmitirlos por el bus, y traduce los datos recibidos a un formato que los componentes del sistema operativo puedan entender. También realiza otras funciones relacionadas con el manejo de la comunicación en el bus. El *hub* raíz tiene uno o más conectores donde acoplar los dispositivos. El *hub* raíz, en conjunto con el controlador del *host* detectan cuando un dispositivo es conectado o desconectado. También se encargan de enviar las peticiones desde el controlador del *host* y transmitir datos entre el dispositivo y el controlador del *host*" [1].

Los dispositivos son los periféricos y *hubs* adicionales que se conectan al bus. Un *hub* tiene uno o más puertos para conectar dispositivos USB. Un *hub* 1.x (se refiere a las especificaciones USB 1.0 y 1.1) reenvía el tráfico USB recibido en ambas direcciones, además de que administra la energía, envía y responde mensajes de status y control, y previene que datos de velocidad media sean transmitidos a dispositivos de baja velocidad. Un *hub* 2.0, además de realizar todas las funciones de un *hub* 1.x, puede transmitir datos a alta velocidad, y en lugar de sólo repetir información, puede convertir datos entre las diferentes velocidades, según sea necesario. También realiza otras funciones que aseguran un uso eficiente del tiempo en el bus.

1.1.3 Cable y conectores USB

Los cables USB tienen cuatro conductores: V_{BUS}, GND, D+ y D-.

V_{BUS} es la alimentación (+5 V)

GND es la referencia a tierra de V_{BUS}, D+ y D-.

D+ y D- son el par de conductores para la señal diferencial.

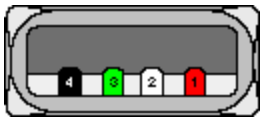
La Tabla 2 presenta las asignaciones estándar (según las especificaciones 1.1) para los terminales de los conectores USB y en la Figura 1, se muestra su distribución en el conector.

Tabla 2. Tabla de asignaciones para los terminales de un conector USB.

Número del contacto	Nombre de la señal	Color asignado
1	V_{BUS}	Rojo
2	D-	Blanco
3	D+	Verde
4	GND	Negro
Shell (Cubierta)	*Shield* - Protección	Cable de drenaje

Fuente: Especificaciones USB 1.1.

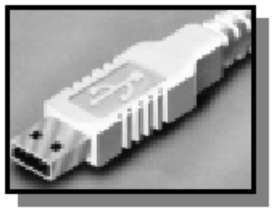
Figura 1. Diagrama de distribución de los terminales de un conector USB.



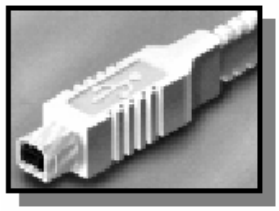
Fuente: los autores.

Las Especificaciones USB 1.1 definen dos tipos de conectores USB: los conectores tipo A que son destinados a los puertos de subida (*upstream*) ubicados en el host, y los conectores tipo B que son destinados a los puertos de bajada (*downstream*) ubicados en los dispositivos; esto facilita a los usuarios finales la conexión de los periféricos y les permite reemplazar los cables sin problemas de compatibilidad debido a que ya se encuentran estandarizados. En la Figura 2 se muestran estos tipos de conectores.

Figura 2. Tipos de conectores USB



Tipo A. Macho



Tipo B. Macho

Tipo A. Hembra

Tipo B. Hembra

Fuente: Especificaciones USB 1.1.

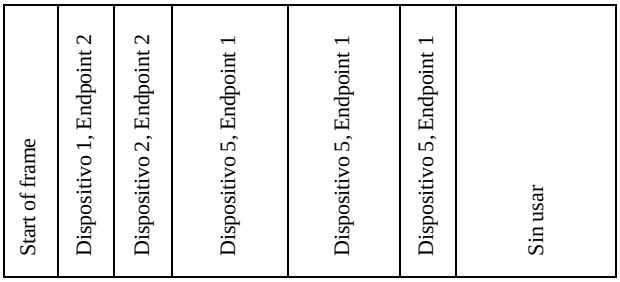
En el capítulo 6 de las Especificaciones USB V1.1 se encuentra una descripción detallada sobre las características eléctricas y mecánicas del cable y del conector USB.

1.1.4 Descripción de la trama USB

Las dos líneas de datos del cable USB (D+ y D-) forman una sola vía de transmisión, la cual es compartida por todos los dispositivos. Este par de líneas llevan una señal diferencial, cuya dirección (datos hacia el *host* o hacia el dispositivo) debe tomar turnos (*half-duplex*). El *host* se encarga de hacer que las transferencias ocurran lo más rápido posible. Para lograrlo, maneja el tráfico dividiendo el tiempo en tramas o *frames.*

Para baja velocidad (*low-speed*) y velocidad media (*full-speed*) las tramas son de 1 ms. Cada trama comienza con una referencia de tiempo llamada *Start-of-Frame* (SOF) o Inicio-de-Trama (para *low speed* esta referencia no existe). En la Figura 3 se muestra un ejemplo de trama.

Figura 3. Ejemplo de una trama para un dispositivo de velocidad media. Para baja velocidad no existe la referencia de tiempo SOF.



Fuente: Axelson, Jan. USB Complete.

Las transferencias[2] consisten en una o más transacciones. Dependiendo de la forma en que el *host* organice las transacciones y de la velocidad de respuesta del dispositivo, las transacciones de una transferencia pueden estar en una sola trama o en varias.

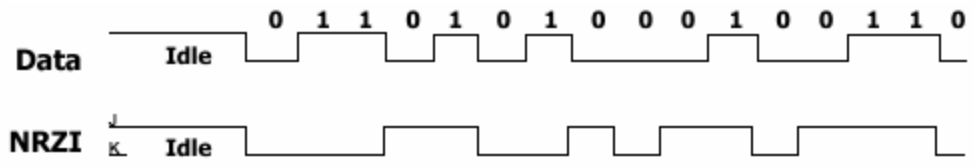
Todas las transferencias comparten una sola vía de datos. Por esta razón cada transacción debe incluir la dirección del dispositivo. Cada transacción comienza cuando el *host* envía un bloque de información que incluye la dirección del dispositivo receptor y una locación específica (o buffer) llamado *endpoint*, que se encuentra en el dispositivo.

1.1.5 Codificación de los datos

El formato de codificación de los datos que se transmiten por el bus USB es el No-retorno a cero invertido (*NRZI*) con *bit stuffing*. En lugar de definir los ceros y unos lógicos como voltajes, el *NRZI* define un cero lógico cuando hay un cambio de voltaje, y un uno lógico cuando se mantiene el voltaje; el bit menos significativo se transmite primero. En la Figura 4 se muestra un ejemplo ilustrativo de este tipo de codificación.

[2] Ver aparte 1.3 del libro.

Figura 4. Codificación de los datos empleando NRZI.



Fuente: Especificaciones USB V1.1.

El receptor emplea las transiciones que ocurren cuando se transmite un cero para sincronizarse con el transmisor.

Si los datos a transmitir son ceros, ocurren gran cantidad de estas transiciones y no hay problema. Pero si los datos son una cadena de unos, la falta de estas transiciones puede ocasionar que el receptor pierda sincronía. Por este motivo se emplea el *bit stuffing* o la inserción de bit, después de seis unos consecutivos, el transmisor inserta un cero, el cual está representado por una transición. Esto asegura por lo menos una transición por cada siete bits transmitidos. El receptor detecta y descarta cualquier bit que venga después de seis unos consecutivos.

Este formato asegura que el receptor permanezca sincronizado con el transmisor, dentro de ciertos límites de tolerancia estipulados en las Especificaciones USB V1.1, sin necesidad de enviar una señal de reloj.

El hardware USB se encarga de realizar toda la codificación y decodificación de los datos automáticamente. Debido al formato de los datos, es difícil interpretarlos empleando un osciloscopio, pero para este propósito existen los analizadores de protocolo USB, que son aplicaciones de *software* que traducen los niveles diferenciales a las tramas descritas anteriormente, para facilitar su análisis.

1.2 COMUNICACIÓN USB

Es posible dividir la comunicación USB en dos categorías, dependiendo de si es empleada para configurar el dispositivo USB o en la aplicación para la cual el dispositivo fue diseñado.

1.2.1 Comunicación de configuración

En la comunicación empleada para configurar el dispositivo USB, el *host* adquiere información acerca del dispositivo y lo prepara para intercambiar datos. La mayoría de esta comunicación ocurre en el momento de encendido o conexión del dispositivo.

Cuando ocurre este tipo de comunicación, el *firmware* del dispositivo responde a una serie de peticiones estándar[3] del *host*. El dispositivo debe identificar la petición, responder con la información adecuada, y si es el caso, realizar otras acciones especificadas por la petición.

1.2.2 Comunicación de aplicación

La comunicación de aplicación ocurre cuando el *host* intercambia datos con el dispositivo, que serán usados en la aplicación para la cual el dispositivo fue diseñado. Esto ocurre después de que se ha culminado el proceso de configuración del dispositivo USB, y se ha asignado y cargado el *driver*. "En el *host*, las aplicaciones pueden emplear las funciones API (Interfaz de programación de aplicaciones) estándar de Windows para leer y escribir al dispositivo. En el dispositivo, el proceso de transferir datos requiere colocar los datos a transmitir en el *buffer* de transmisión, leer los datos del *buffer* de recepción y asegurar que el dispositivo está listo para la siguiente transferencia" [1].

1.3 TRANSFERENCIAS USB

Cada transferencia USB se compone de transacciones, y a su vez, cada transacción está hecha de paquetes. Los paquetes son los que contienen la información de status, del tipo de transferencia, del dispositivo al que va dirigida la transacción o los datos a transmitir para la aplicación. En los apartes 1.3.1 y 1.3.2 del libro se explican brevemente los conceptos de *Endpoints* y *Pipes*.

[3] Ver aparte 9.4 de las Especificaciones USB 1.1.

1.3.1 *Endpoints*

Todas las transferencias USB van desde o hacia un *endpoint*. Un *endpoint* es un buffer que guarda múltiples bytes, que pueden ser datos recibidos o datos esperando a ser transmitidos, dependiendo del tipo de *endpoint*.

Todos los dispositivos USB deben tener un *Endpoint* 0 configurado como de control, el cual debe transmitir datos en ambas direcciones. Generalmente no se necesitan más *endpoints* de control, aunque algunos dispositivos los soportan. Además del *endpoint* 0 de control, los dispositivos *low speed* soportan dos adicionales: *Endpoint* 1, que sólo se puede configurar como de entrada, y *Enpoint* 2 que se puede configurar ya sea como de entrada o de salida.

Los *endpoints* sólo contienen datos que han sido o serán transmitidos en una dirección, siendo el *endpoint* de control (el encargado de la comunicación de configuración) la excepción. El *host* no tiene *endpoints*, pero sirve como punto de inicio de la comunicación con los *endpoints* del dispositivo.

Para tener acceso a un *endpoint*, se debe tener en cuenta su número y su dirección. El número puede ir de 0 a 15 (para baja velocidad son 3: 0, 1 y 2), y la dirección se toma desde la perspectiva del *host*: *IN* (de entrada) si es hacia el *host* y *OUT* (de salida) si es en dirección opuesta.

1.3.2 *Pipes*

"Un *pipe* es una asociación entre el *endpoint* de un dispositivo y el *driver* o controlador del dispositivo USB" [1] que se encuentra en el *host*, y se emplea como canal para la transmisión de datos, o para ejecutar alguna acción sobre el dispositivo, especificada en el *driver*. Los *pipes* deben ser establecidos antes de que ocurra cualquier transferencia.

Al encender o conectar un dispositivo USB, el *host* realiza peticiones de información de configuración del dispositivo, estableciendo y creando los *pipes* definidos en la configuración del dispositivo seleccionada. Si el dispositivo es desconectado, el *host* remueve los *pipes* que no necesita. Si se realiza un cambio de configuración del dispositivo, el *host* añade o remueve *pipes* dependiendo de las diferencias presentes en las distintas configuraciones. Todos los dispositivos deben tener un *Pipe* de control por defecto, el cual está asociado al *Endpoint* 0.

Las especificaciones USB 1.1 clasifican los *pipes* según la dirección del flujo de los datos en: *pipes* de mensaje y *pipes* de datos.

Pipes de mensaje

Los *pipes* de mensaje son bidireccionales, y por lo tanto son empleados por los *endpoints* de control. En este tipo de *pipe*, las transferencias se inician con una petición por parte *host*. Para completar la transferencia, el *host* y el dispositivo pueden intercambiar datos o información de estado, o simplemente el dispositivo envía información sobre su estado. Si el dispositivo soporta la petición, realiza la acción requerida. Si por el contrario, el dispositivo no soporta la petición, responde con un código que indica que no lo soporta.

***Pipes* de datos**

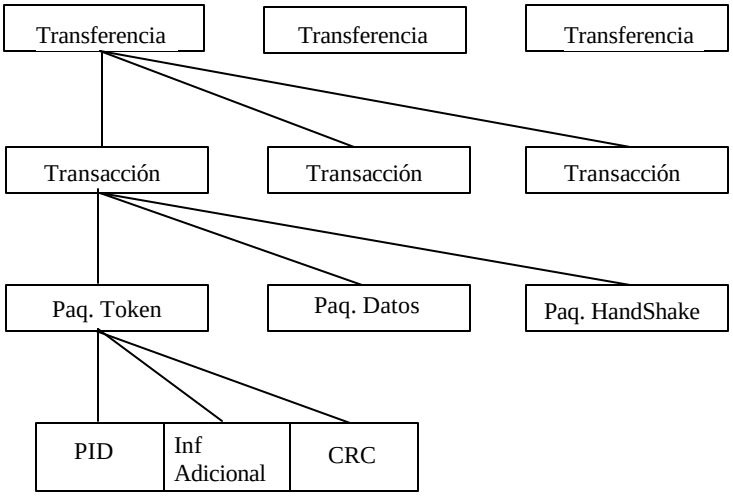
Los demás *endpoints* emplean *pipes* de datos. Las especificaciones USB 1.1 no definen el formato de los datos en este tipo de *pipe*. El receptor (puede ser el *host* o el dispositivo USB, dependiendo de la dirección) simplemente acepta todo lo que llega. El *firmware* del dispositivo y el *software* del *host* se encargan luego de procesar los datos de manera adecuada, de acuerdo con la aplicación. De cualquier forma, los dispositivos transmisor y receptor deben acordar un formato para los datos, para que puedan ser interpretados por la aplicación.

1.3.3 Componentes de una transferencia USB

En la Figura 5 se pueden observar el diagrama de bloques para los componentes típicos de una transferencia USB.

Los tipos de transferencias y sus componentes se detallan en el siguiente aparte de este capítulo.

Figura 5. Bloques componentes típicos de una transferencia USB.



Fuente: Axelson, Jan. USB Complete.

1.4 TIPOS DE TRANSFERENCIAS USB

La interfaz USB está diseñada para manejar diferentes tipos de periféricos, cada uno con requerimientos específicos, ya sea de tasa de transferencia de datos, de respuesta en el tiempo o de detección y corrección de errores.

Las especificaciones USB 1.1 definen cuatro tipos de transferencias: de control, por interrupción, isócrona y de volumen (*bulk*); cada una de ellas maneja una necesidad específica. Un dispositivo soporta la(s) transferencia(s) que mejor se ajuste(n) a su propósito.

Los dispositivos USB de baja velocidad solo manejan dos tipos de transferencias: de control y por interrupción, los cuales son explicados a continuación.

1.4.1 Transferencias de control

Las transferencias de control tienen básicamente dos usos: llevar las peticiones definidas por las especificaciones USB, las cuales serán empleadas por el *host* para configurar y obtener información sobre el dispositivo, y llevar las peticiones definidas, ya sea por la clase del dispositivo o por el fabricante, para cualquier otro propósito.

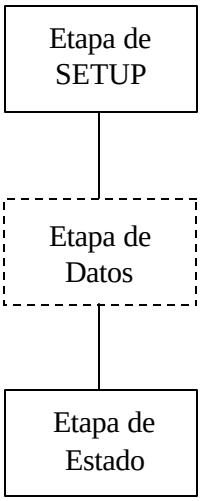
El *host* debe asegurar que las transferencias de control ocurran lo más rápido posible. Por esta razón, reserva una porción del ancho de banda[4] para este tipo de transferencias (10% para baja velocidad). El número de transferencias de control pueden ser más o menos de las que caben en el tiempo reservado para este fin, dependiendo de los requerimientos de los dispositivos USB conectados al bus.

Cada transferencia de control debe tener una Etapa de *Setup* y una Etapa de Estado (*Status*). La Etapa de Datos es opcional y algunas peticiones particulares la requieren. Estas etapas se pueden observar en la Figura 6.

Cada etapa de la transferencia de control corresponde a una o más transacciones. Las transacciones correspondientes a cada etapa se describen a continuación.

[4] Ver apartes 4.7.3 y 5.7.4 de las especificaciones USB 1.1.

Figura 6. Etapas de una transferencia de Control

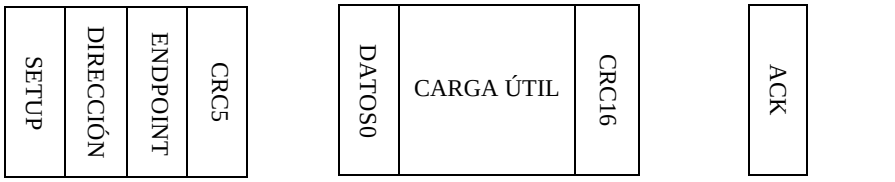


Fuente: Axelson, Jan. USB Complete.

Etapa de *SETUP*

La Etapa de *SETUP* corresponde a una transacción que recibe el nombre de *Setup*. Una transacción *Setup* está compuesta por un paquete de Señalización o *Token*, un paquete de datos que contiene la petición y un paquete *Handshake* con información acerca del estado de la transacción, tal como se muestran en la Figura 7. Los tipos de paquetes se describen en el aparte 1.5 del libro.

Figura 7. Paquetes de una transacción *SETUP*.



Paquete *Token* Paquete de datos Paquete *Handshake*

Fuente: los autores.

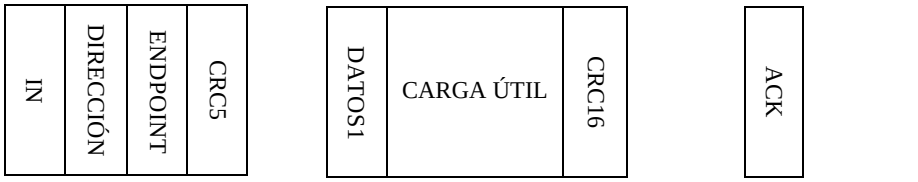
Etapa de datos

Si existe una Etapa de datos, ésta contiene una o más Transacciones de datos. Dependiendo de la petición, el *host* o el periférico pueden ser la fuente de estas

transacciones; pero durante una Etapa de datos en particular, la fuente de los datos debe ser la misma.

Los paquetes que intervienen en una Transacción de datos se muestran en la Figura 8 y corresponden a: un paquete de señalización (*Token*) de entrada indicando que los datos van hacia el *host* o de salida *OUT* indicando que los datos van hacia el *endpoint*, un paquete de datos (se inicia con Datos1 y se alterna en adelante entre Datos0 y Datos1[5], por cada Transacción de datos), que es enviado desde o hacia el *host* dependiendo de la dirección indicada en el paquete *Token*, y un paquete *Handshake*.

Figura 8. Paquetes de una transacción de datos.



Paquete *Token* Paquete de datos Paquete *Handshake*

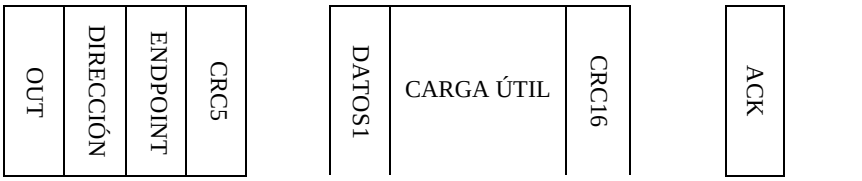
Fuente: los autores.

Etapa de estado

La Etapa de Estado consiste en una Transacción de Estado. Se emplea para reportar el éxito o el fracaso de la etapa anterior. Los paquetes que intervienen en una Transacción de Estado son los que se muestran en la Figura 9. En esta etapa se emplea un paquete *Token IN* si el dispositivo es el que suministra el estado, o un paquete *Token OUT* si es el *host* el que lo suministra. El paquete de Datos es de tamaño 0, y el paquete *Handshake* indica el éxito o una condición de error de las dos etapas anteriores.

[5] Ver aparte 1.5.2 del libro.

Figura 9. Paquetes de una transacción de Estado.



Fuente: los autores.

1.4.2 Transferencias por interrupción

Las transferencias por interrupción son útiles cuando los datos se deben transmitir a intervalos determinados de tiempo máximo, sin necesidad de asegurar una tasa de transferencia de datos constante. Las aplicaciones típicas son el teclado, *mouse*, *joystick*, entre otros. Los dispositivos de baja velocidad deben emplear las transferencias por interrupción para enviar datos genéricos. Este tipo de transferencia también es popular debido a que los sistemas operativos incluyen *drivers* que habilitan a las aplicaciones para realizar transferencias por interrupción con Dispositivos de Interfaz Humana (HID – *Human Interface Devices*)[6].

El nombre de este tipo de transferencia puede sugerir que un dispositivo causa una interrupción por *hardware* que provoca una respuesta inmediata del *host*, pero en realidad lo que ocurre es que *host* encuesta el dispositivo cada cierto intervalo de tiempo (varía entre 10 y 255 ms), para saber si éste requiere atención. La estructura de este tipo de transferencia es idéntica a la de las Transferencias de volumen, su diferencia radica en el tiempo empleado para la transmisión.

Una transferencia por interrupción consiste en una o más Transacciones de datos de entrada (*IN*)[7], o una o más Transacciones de datos de salida (*OUT*). Las transferencias por interrupción se realizan en una sola dirección: todas las

[6] Corresponde a una clase de dispositivos USB. Ver aparte 9.7 de las Especificaciones USB V1.1.
[7] Ver Figura 6 del libro.

transacciones deben ser de entrada *IN* o todas deben ser de salida *OUT*. Las transferencias en dos direcciones requieren transferencias separadas y un *pipe* para cada dirección.

El final de una transferencia por interrupción está marcado de dos formas: cuando se ha enviado toda la información requerida por la petición, o cuando se envía un paquete conteniendo menos datos que el máximo permitido (o un paquete sin datos).

1.5 TIPOS DE PAQUETES USB

Cada paquete es un bloque de información con un formato definido. Todos los paquetes comienzan con un identificador de paquete *PID* (*Packet Identifier*). El campo *PID* es de 8 bits, donde los 4 bits menos significativos contienen el identificador, y los 4 bits más significativos son su complemento. Dependiendo de la transacción, después del *PID* el paquete puede contener una dirección de *endpoint*, datos o información de estado, seguido de los bits de chequeo de error (CRC)[8].

Existen cuatro tipos de paquetes USB: los de señalización o *Token*, los de datos, los de estado y los especiales, y son explicados a continuación.

1.5.1 Paquetes de señalización (*Token*)

El *host* emplea los paquetes de señalización para enviar una petición de comunicación al dispositivo USB. El PID indica el tipo de transacción como *SETUP*, *IN* o *OUT*. El formato de los 3 paquetes de señalización o *Token* es el mismo, y se observa en las Figuras 5, 6 y 7 del libro.

[8] Ver aparte 1.6 del libro.

Los primeros 8 bits corresponden al identificador de paquete (*PID*), seguido de la dirección del dispositivo de 7 bits, la dirección del *endpoint* de 4 bits y 5 bits de chequeo de errores (*CRC5*).

Los paquetes *IN* indican una transferencia de un dispositivo al *host*. Los paquetes *OUT* indican una transferencia del *host* a un dispositivo. Estos dos paquetes pueden ir dirigidos a cualquier dispositivo. El paquete *SETUP* es un caso especial del *OUT*. Es de alta prioridad y siempre se dirige al *enpoint* 0, o *endpoint* de control.

1.5.2 Paquetes de datos

El *host* y el dispositivo pueden transmitir cualquier tipo de información en un paquete de datos, identificado como DATOS0 o DATOS1; estos paquetes se muestran en las Figuras 5 y 6 respectivamente.

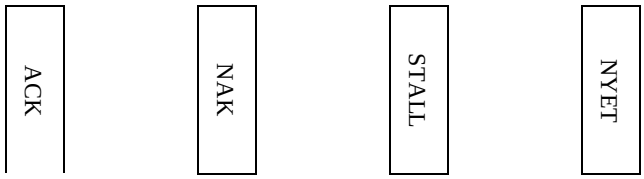
Cuando se necesita enviar múltiples paquetes de datos, el identificador alterna entre los dos identificadores de paquetes DATOS0 o DATOS1, para permitir el reensamble de los datos e identificar la pérdida de un paquete. Estos paquetes pueden tener de 0 a 8 bytes de carga útil para baja velocidad seguidos de 16 bits para chequeo de errores (CRC16).

1.5.3 Paquetes de estado (*handshake*)

Son empleados por el *host* o el dispositivo para enviar información acerca de su estado o para indicar el éxito de una transacción.

En la Figura 10 se muestran los diferentes paquetes de estado definidos por las Especificaciones USB 1.1.

Figura 10. Paquetes de estado



Fuente: los autores.

El *PID* contiene el código de estado (ACK, NAK, STALL, NYET)[9], el cual indica cómo fue la recepción. No requiere bits de datos, ni de chequeo de errores porque el *PID* es redundante.

1.5.4 Paquetes especiales

En las Especificaciones USB1.1 están definidos cuatros tipos de paquetes especiales, uno de ellos exclusivo para baja velocidad. Su identificador de paquete es PRE. El host se encarga de transmitir este paquete que contiene un código preliminar que indica que el siguiente paquete que va a transmitir por el bus USB es de baja velocidad. Su utilidad se hace evidente en redes que contienen dispositivos USB de velocidad baja y media, para indicar a los *hubs*[10] de la red cuándo debe habilitar a los dispositivos de baja velocidad para recibir un paquete.

1.6 CHEQUEO DE ERRORES

Son los bits incluidos en los paquetes de señalización y de datos, con el fin de detectar errores en el paquete recibido por el *host* o por el dispositivo USB. Estos bits son calculados empleando un algoritmo matemático llamado Chequeo de Redundancia Cíclica (CRC – *Cyclic Redundancy Check*). El número que acompaña a

[9] La definición de cada paquete de estado se encuentra en el aparte 8.4.4 de las especificaciones USB1.1.
[10] Concentrador de dispositivos. Contiene uno o más puertos USB para conectar dispositivos u otros *hubs.*

las letras CRC indica el número de bits empleados para el chequeo de errores (por ejemplo, CRC5 contiene 5 bits de chequeo de errores).

El dispositivo transmisor (*host* o dispositivo USB) realiza los cálculos y envía el resultado junto con los datos. El dispositivo receptor realiza los mismos cálculos con los datos recibidos. Si el resultado concuerda, los datos han llegado sin error y el dispositivo receptor envía un Paquete de estado (ACK) indicando que la recepción del paquete fue exitosa. Si el resultado no concuerda, el receptor simplemente no envía ningún paquete de estado (*handshake*), lo que le indicará al transmisor que debe intentarlo de nuevo.

Típicamente, el *host* reintenta tres veces, aunque las especificaciones dan al *host* un poco de flexibilidad para determinar el número de reintentos. Si de todas maneras el *host* no recibe ningún paquete de estado, deja de intentarlo y le informa al *driver* de la situación.

2. TARJETA DE DESARROLLO PARA EL MICROCONTROLADOR MC68HC908JB8JP

En este capítulo se describe el *hardware* de la Tarjeta de desarrollo para el microcontrolador MC68HC908JB8JP, la cual se emplea para programar el microcontrolador y para ejecutar la aplicación programada en el mismo.

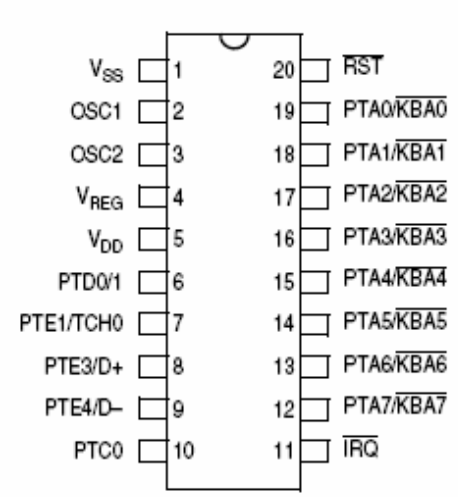
2.1 MICROCONTROLADOR

El microcontrolador empleado en este proyecto es el MC68HC908JB8JP, un microcontrolador con MCU (*Microcontroller Unit* - Unidad microcontroladora) de 8-bits, perteneciente a la familia HC08 de Motorola. Esta familia de microcontroladores de Motorola es la más comercial, y en Colombia son relativamente sencillos de conseguir. Se escogió este microcontrolador debido, principalmente, a sus características técnicas de memoria (cuenta con una memoria FLASH de 8 kbytes, en la cual se almacena el programa) y al módulo USB embebido en su empaquetado, aunque también se tuvo en cuenta su facilidad de consecución.

2.1.1 Asignación de pines del microcontrolador.

El diagrama de asignación y distribución de pines, para el empaquetado DIP de 20 pines empleado en la tarjeta de desarrollo y en la tarjeta SAD800-L (Sistema de Adquisición de Datos diseñado en este proyecto) se muestra en la Figura 11.

Figura 11. Diagrama de asignación de pines del microcontrolador MC68HC908JB8JP



PTD0/1 pin: PTD0 and PTD1 internal pads are bonded together to PTD0/1 pin.

Pins not available on 20-pin package:			
PTB0			PTE0/TCLK
PTB1	PTC1		
PTB2	PTC2	PTD2	PTE2/TCH1
PTB3	PTC3	PTD3	
PTB4	PTC4	PTD4	
PTB5	PTC5	PTD5	
PTB6	PTC6	PTD6	
PTB7	PTC7	PTD7	
Internal pads are unconnected.			

Fuente: Especificaciones Técnicas del microcontrolador MC68HC908JB8.

2.1.2 Módulo USB

El módulo USB fue diseñado para funcionar como un dispositivo de baja velocidad (*low speed)* bajo las Especificaciones USB 1.1; soporta transferencias de control y por interrupción. El módulo USB se encarga de manejar la comunicación entre el *host* y las funciones USB del microcontrolador.

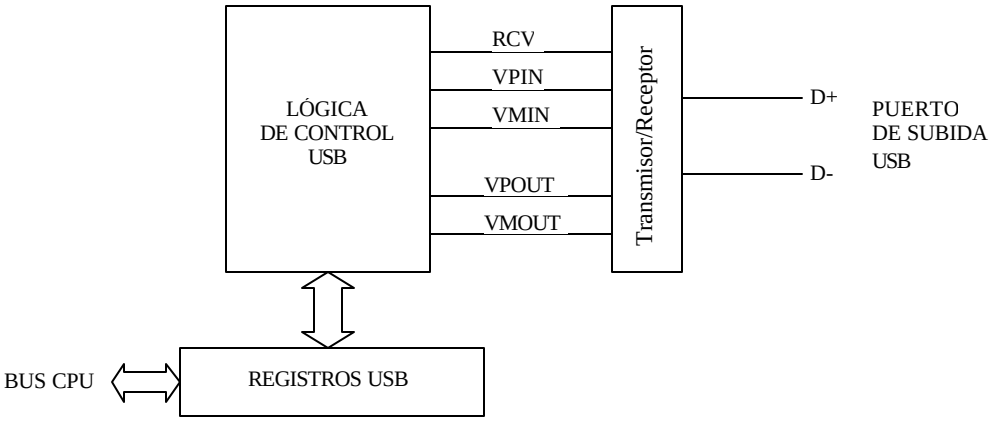
Este módulo realiza la lógica de control de los datos autónomamente: codifica y decodifica los paquetes, se encarga de la generación y chequeo de los errores (CRC - *Cyclic Redundancy Check* – Chequeo de redundancia cíclica), codificación y decodificación de los datos de acuerdo al protocolo NRZI (*Non Return to Zero invertid* - No retorno a cero invertido) y se encarga también del *bit stuffing*[11]. También genera interrupciones al identificar las señales pertinentes en el bus USB.

Diagrama de bloques del modulo USB

El módulo USB está particionado en 3 bloques funcionales: un trasmisor/receptor, el bloque encargado de la lógica de control y los registros de los *endpoints*; como se muestra en la Figura 12.

[11] Referirse al aparte 1.1.5 del libro.

Figura 12. Diagrama de bloques del módulo USB



Fuente: Especificaciones Técnicas del microcontrolador MC68HC908JB8.

El transmisor/receptor provee la interfaz física a las líneas de datos USB D+ y D-. El transmisor/receptor está compuesto de dos partes: un circuito que maneja las salidas (manejador) y un receptor. RCV corresponde al voltaje regulado de 3.3 V, VPIN y VMIN corresponden a las señales referenciadas a tierra de entrada a la lógica de control USB, y VPOUT y VMOUT corresponden a las señales referenciadas a tierra de salida de la lógica de control USB.

La lógica de control USB se encarga del movimiento de los datos entre la CPU y el transmisor/receptor. La lógica de control maneja las operaciones de transmisión y recepción, y contiene la lógica que manipula tanto el transmisor/receptor como los registros de los *endpoints*[12]. Durante la operación de transmisión, el *buffer* contador de bytes se carga con la cuenta actual de bytes del *endpoint* de transmisión. Este mismo *buffer* es usado para contar los bytes recibidos en las operaciones de recepción. Durante la transmisión, la lógica de control se encarga de la conversión paralelo a serial, la generación de los bits de control de errores (CRC[13]), la codificación NRZI y el *bit stuffing*. Durante la recepción, la lógica de control se

[12] Referirse al aparte 1.3.1 del libro.
[13] Referirse al aparte 1.6 del libro.

encarga de la detección de la señal de sincronismo *(SYNC[14])*, la identificación de paquetes, la detección de final de paquete, eliminar los bits insertados durante el *bit stuffing*, decodificación NRZI, validación de los bits de control de errores (CRC) y conversión de serial a paralelo.

Los registros USB son los registros de control, de estado y de datos del módulo USB y de los *endpoints.*

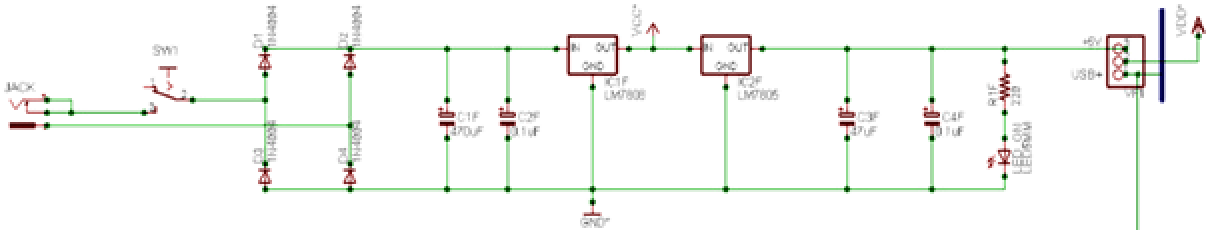
2.2 DISEÑO DE LA TARJETA DE DESARROLLO PARA EL MICROCONTROLADOR MC68HC908JB8JP

El hardware correspondiente a la tarjeta de adquisición de datos cuenta con una etapa de alimentación, una etapa de comunicación con el PC, una etapa de aplicación para el microcontrolador y los conectores de salida.

2.2.1 Etapa de alimentación

La alimentación se toma de un adaptador DC graduado a 12 V, 500mA. Este voltaje pasa a través de un puente de diodos y un filtro RC para reducir cualquier residuo de AC. Se cuenta con dos reguladores de voltaje, un LM7808 para alimentar con 8 V al pin $\overline{IRQ}$ durante la programación del microcontrolador (modo monitor), y un LM7805 para alimentarlo con 5 V durante su operación en modo aplicación. A través del *jumper* JP1_M se habilita ya sea una u otra alimentación. Esta etapa se muestra en la Figura 13.

Figura 13. Diagrama esquemático de la alimentación para la Tarjeta de Desarrollo.

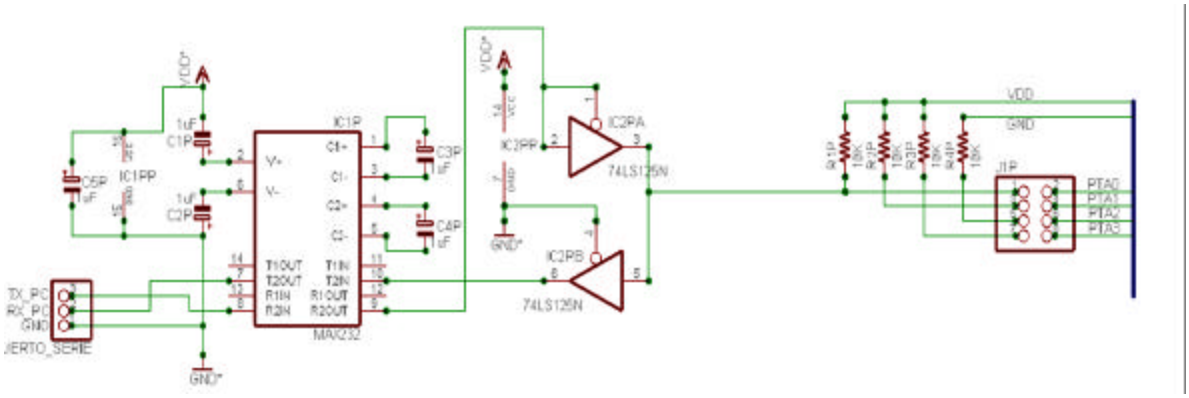


[14] Referirse al aparte 7.1.10 de las Especificaciones USB 1.1.

2.2.2 Etapa de comunicación con el PC

Esta etapa, cuyo esquemático se muestra en la Figura 14, se emplea para borrar y programar la memoria FLASH de la MCU a través de una interfaz hacia el computador de una sola línea, durante el modo monitor.

Figura 14. Diagrama esquemático de la Etapa de comunicación con el PC.



El modo monitor recibe y ejecuta comandos desde el computador, los cuales pueden acceder a cualquier dirección de memoria del microncontrolador. Toda la comunicación entre el computador y el microcontrolador ocurre a través del pin PTA0; interfaz que tiene la característica de ser bidireccional, *half-duplex* y cumple con el protocolo RS232, con una tasa de 9600 baudios.

La configuración seleccionada en la tarjeta de desarrollo para entrar a modo monitor es de “alto voltaje” (8 V) con un reloj externo de 6 MHz, niveles de tensión alto en los pines PTA0, PTA1 y PTA3, y nivel de tensión bajo en el pin PTA2. Se escogió esta opción debido principalmente a que no se requiere un valor particular en las posiciones de memoria $FFFE y $FFFF.

Para entrar a modo monitor, se debe realizar un RESET con los *jumpers* J1P colocados y el *jumper* JP1_M colocado en modo de programación asegurando que los niveles de voltaje de los pines PTA0-PTA3 sean los apropiados; estos niveles se logran con las resistencias R1P, R2P y R3P que elevan los niveles de tensión a V_{DD} y

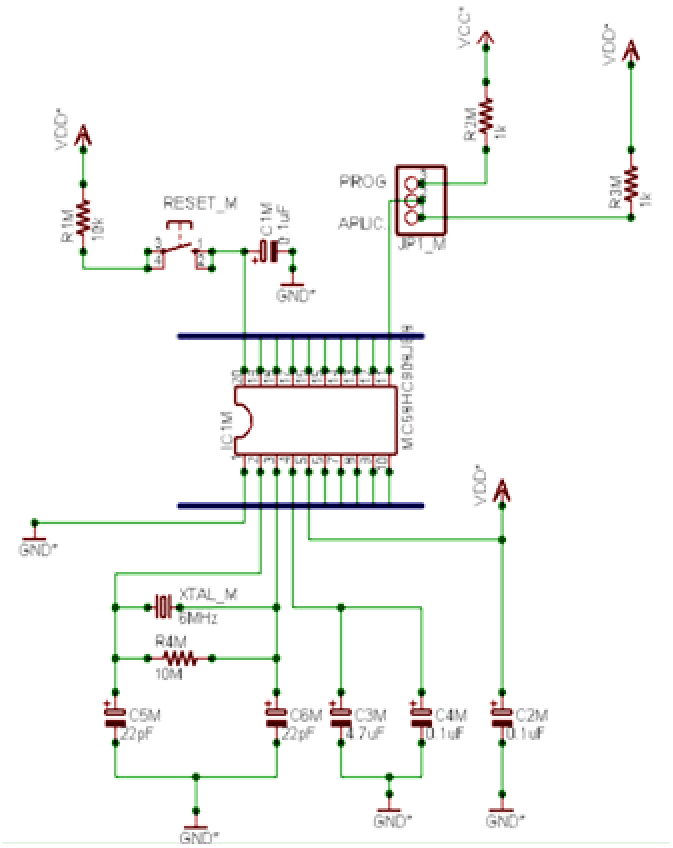
R4P que lo baja a GND. El flanco de subida del pin $\overline{RST}$ permite la entrada al modo monitor. Una vez que se ejecuta el RESET, la MCU entra a modo monitor y los niveles de tensión de los pines del puerto A (PTA0-PTA3) ya no son requeridos.

La interfaz también requiere un transmisor/receptor para RS232 (Circuito Integrado MAX232) y un buffer (CI 74LS125) para comunicarse con el *host* (*PC*) a través del puerto PTA0.

2.2.3 Etapa de aplicación para el microcontrolador

Esta etapa se muestra en la Figura 15, e incluye los elementos necesarios para habilitar el microcontrolador, tales como el oscilador, niveles de tensión adecuados, y condensadores para evitar el efecto de rizado.

Figura 15. Diagrama esquemático de la etapa de aplicación para el microcontrolador.



Oscilador

Se empleó un oscilador de cristal de 6 MHz, dos condensadores (C5M, C6M) y una resistencia (R4M), configurados como un oscilador Pierce, con los elementos activos integrados en la MCU. Este oscilador produce un reloj externo de 6MHz. El reloj interno del bus es de 3 MHz, y el reloj del bus USB de 1.5 MHz; dichas frecuencias son derivados del reloj principal externo.

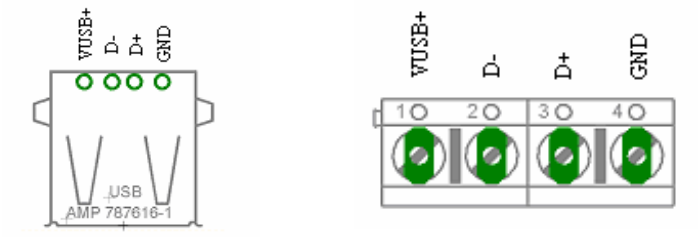
2.2.4 Conectores de salida

En la Tarjeta de desarrollo se emplearon dos conectores de salida para el módulo USB, y un conector adicional con las salidas de los pines del microcontrolador.

Salidas USB

Las salidas de datos (D+ y D-) del bus USB se pueden tomar en dos puntos diferentes, esto es: en un conector de bornera (tipo tornillo) y en un conector tipo A (female - hembra) (ver Figura 16). Para el conector tipo bornera fue necesario adaptar un cable USB.

Figura 16. Conectores de salida para el puerto USB.



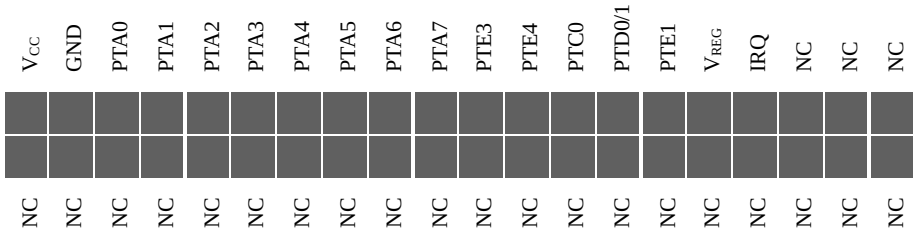
a. Tipo A (hembra)

b. Tipo Bornera

Salidas adicionales

El conector que se muestra en la Figura 17 contiene todas las salidas de los puertos del microcontrolador y otros valores útiles como alimentación, etc. Para ésta última se empleó una correa de 40 conductores, debido a que es una conexión robusta.

Figura 17. Conector para las salidas de la Tarjeta de Desarrollo



Fuente: los autores.

3. HERRAMIENTAS USB

Las herramientas USB presentadas en este capítulo forman parte importante de este proyecto, y ayudan a entender mejor el concepto de USB, el protocolo que emplea y características principales. Dentro de estas herramientas se encuentran los *drivers* o controladores USB, aplicaciones, analizadores de protocolo, entre otros.

3.1 CONTROLADOR (*DRIVER*) USBIO V1.51

La empresa de software Thesycon[15] ha desarrollado diferentes paquetes de *drivers* para controlar dispositivos USB dependiendo del sistema operativo que maneja el computador personal (PC) donde se desea conectar el dispositivo USB. Además, cada paquete de *drivers* tiene una versión *light* (sin costo), con restricciones de configuración para el dispositivo USB y una versión demo (sin costo), con todas las opciones de configuración, pero limitada a 20 minutos de trabajo continuo. En este proyecto se empleó la versión *light* del *driver* USBIO V1.51 para Windows XP contenida en el paquete `usbio_el.exe`. Para mayor información sobre el contenido del paquete USBIO *light* y el proceso de instalación del *driver* USBIO ver Anexo A.

3.1.1 Modelo de objetos del *driver* USBIO

"El *driver* USBIO provee un modelo de comunicación que consiste en objetos[16] de dispositivos y objetos de *pipes* (canales[17]) los cuales son creados, destruidos y manejados por el *driver* y se explican más adelante en este aparte. Una aplicación

[15] Thesycon SystemSoftware & Consulting Gmbh
[16] Un objeto es una estructura de programación que contiene funciones y datos.
[17] Ver aparte 1.3.2 del libro.

puede puede abrir o crear manejadores para los objetos de dispositivos y asociarlos a objetos de *pipes* como se muestra en la Figura 18.

Objetos de dispositivos

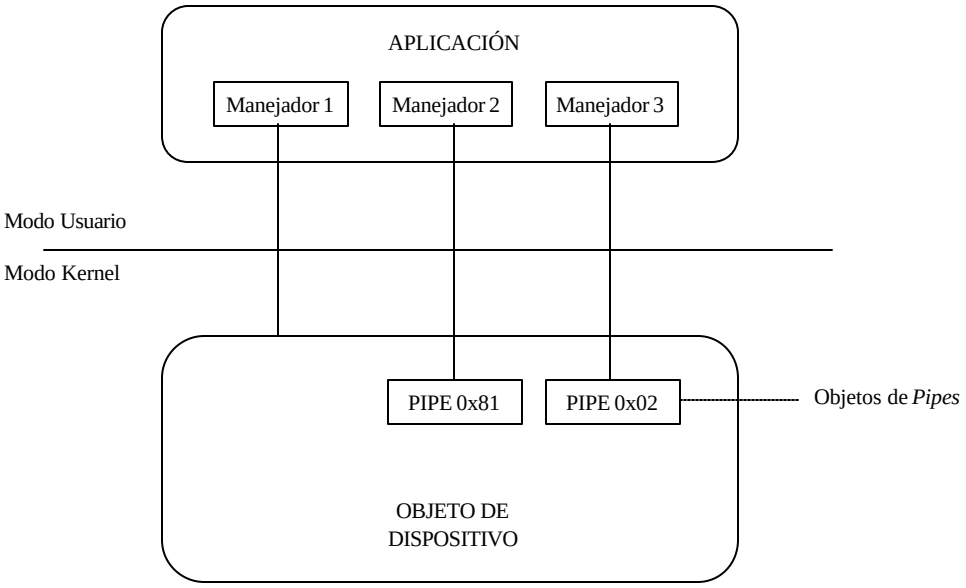
Cada objeto de dispositivo está asociado a un dispositivo USB físico, conectado al bus USB. Un objeto de dispositivo es creado por el *driver* USBIO en respuesta a la petición de añadir un dispositivo generada por el Manejador de *Plug&Play* (inserción en caliente) del sistema operativo. El *driver* USBIO puede manejar varios dispositivos USB conectados al mismo tiempo.

Objetos de *Pipes*

El *driver* USBIO emplea los objetos de *Pipes* para representar los *endpoints* de un dispositivo. Los objetos de *Pipes* son creados cuando se ajusta una configuración para el dispositivo USB. El número y tipo de objeto de *Pipe* creado depende de la configuración del dispositivo seleccionada. El *driver* USBIO no controla el endpoint por defecto (Endpoint 0 o endpoint de control), debido a que éste es controlado por el *driver* USBD.SYS el cual hace parte del sistema operativo de Windows.

Manejadores. Los manejadores son creados por la aplicación para tener acceso a los objetos de dispositivo y a los objetos de Pipes. Si la aplicación requiere establecer comunicación con un *endpoint*, primero es necesario abrir el objeto de dispositivo para crear el manejador. Este manejador se debe asociar al objeto de *Pipe* que representa al *endpoint* con el cual se desea establecer la comunicación. Cada *pipe* puede ser asociado sólo a un manejador y un manejador sólo puede ser asociado a un *pipe*. Por esta razón la aplicación debe crear un manejador por cada *Pipe* que requiera accesar.

Figura 18. Modelo de objetos del *driver* USBIO, donde se muestran un objeto de dispositivo y dos objetos de *Pipes*.



Fuente: Manual de referencia de la interfaz USBIO COM

Si un manejador no está asociado a ningún objeto de *pipe* (como el manejador 1 de la Figura 18) es llamado manejador de dispositivo y es usado sólo para realizar operaciones relacionadas con el dispositivo.

Un manejador asociado a un objeto de *Pipe* que representa a un *endpoint* de entrada (manejador 2 de la Figura 18) puede ser usado para trasmitir los datos desde el *endpoint* de entrada del dispositivo USB a la aplicación. Un manejador asociado a un objeto de *pipe* que representa un *endpoint* de salida (manejador 3 de la Figura 18) puede ser usado para trasmitir los datos desde la aplicación hacia el *endpoint* de salida del dispositivo USB. Estos manejadores son llamados manejadores de *pipes* y pueden realizar operaciones relacionadas tanto con el dispositivo USB, como con los *pipes*" [5].

3.1.2 Interfaz de programación USBIO COM

La interfaz USBIO COM es una interfaz de programación de alto nivel y se encuentra dentro del paquete del *driver* USBIO. Se encarga de manejar las operaciones de lectura y escritura, almacenamiento de datos en los *buffers* respectivos (de lectura o escritura) y generación de eventos para informar que los datos se encuentran disponibles. Los métodos, propiedades y eventos disponibles para esta interfaz se encuentran en el manual de referencia de la interfaz USBIO COM[18] incluido en el paquete del *driver* USBIO.

3.2 CONTROLADOR NI-VISA V3.1

NI-VISA es la implementación del *driver* genérico VISA (Arquitectura de *Software* para instrumentos virtuales) desarrollada por National Instruments. NI-VISA permite controlar gran cantidad de dispositivos de medición y adquisición de datos, entre los que se encuentran los dispositivos USB. Para mayor información sobre la instalación y herramientas del *driver* NI-VISA V3.1 ver Anexo A.

El objetivo principal del controlador NI-VISA es encapsular las capacidades de un dispositivo en general (leer datos, transmitir datos, responder a peticiones, entre otras) en un "recurso" (*resource*). El "recurso" USB RAW se emplea para controlar los dispositivos USB y permite recibir datos por interrupción.

Es posible implementar una aplicación en LabVIEW 7.1 (lanzado al mercado en mayo del presente año) empleando el *driver* VISA 3.1 para controlar un dispositivo USB.

[18] USBIO COM Interface Reference Manual.

3.3 MONITOR DE PROTOCOLO USB (*USB MONITOR*)

El Monitor de protocolo USB (USB MONITOR[19]) es una herramienta que permite visualizar los paquetes de peticiones y datos trasmitidos o recibidos por el puerto USB para verificar el protocolo de comunicación. La información se visualiza de una manera que facilita su análisis, además de que cuenta con cuadros que ayudan a interpretar la información transmitida y recibida por el bus USB.

El funcionamiento de la herramienta USB MONITOR se encuentra explicado en el Anexo A.

3.4 CODEWARRIOR - ESTUDIO DE DESARROLLO PARA MICROCONTROLADORES DE LA FAMILIA HC08 (V2.1)

CodeWarrior es un software desarrollado por Metrowerks. Se emplea para programar microcontroladores de la familia HC08 de Motorota y contiene una herramienta llamada Procesador Experto (*Processor Expert*). El Procesador Experto es una interfaz gráfica que permite crear un proyecto en el cual se escoge el microcontrolador y se configuran los puertos y módulos internos o externos del microcontrolador a utilizar, empleando menús desplegables. Esta parte del código es generada automáticamente en lenguaje de alto nivel (lenguaje ANSI C), y el resto del código se puede seguir escribiendo en este lenguaje o en ensamblador (empleando la instrucción asm).

Además de esto, la herramienta también genera automáticamente las funciones principales de cada puerto, como obtener un valor si el puerto está configurado como entrada, o colocar un valor si está configurado como salida, y las funciones principales de cada módulo, por ejemplo enviar y recibir datos para el módulo USB.

[19] Fabricante HHD Software.

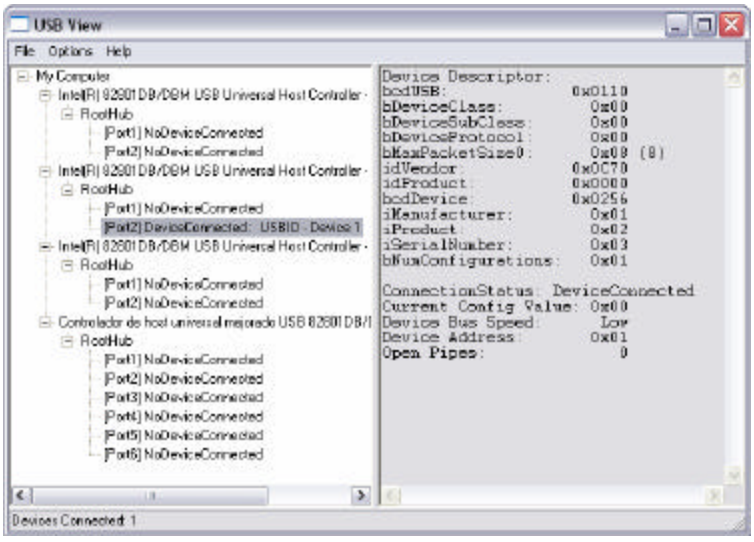
Para programar el microcontrolador MC68HC908JB8JP y el módulo (*bean*) USB fue necesario descargar un *patch* o parche para la versión 2.1 de CodeWarrior, de la página de Metrowerks[20]. Éste requiere una licencia especial que se puede pedir a Metrowerks[21].

3.5 VISOR DE DISPOSITIVOS USB (USB VIEW)

Esta herramienta fue diseñada por Microsoft[22]. USB View muestra en la parte izquierda de su ventana (Figura 19) un listado de los dispositivos conectados al computador personal (PC), y en su parte derecha muestra el estado y la información contenida en el descriptor del dispositivo USB seleccionado en la parte izquierda.

USB View es una herramienta muy útil para analizar dispositivos USB ya que obtiene la información de los descriptores y del estado del dispositivo automáticamente. Para observar cambios en la ventana, se debe actualizar empleando la tecla F5.

Figura 19. Ventana de la aplicación USB View.



Fuente: los autores.

[20] http://www.metrowerks.com/
[21] license@metrowerks.com
[22] Esta herramienta se puede descargar de la página www.usbman.com.

4. DISEÑO DEL HARDWARE DE LA TARJETA SAD800-L

4.1 GENERALIDADES

La metodología de diseño del sistema de adquisición de datos ***Sad800-L*** tiene en cuenta velocidad, tipo de transferencia y demás prestaciones que brinda el bus USB v1.1 y que determinan la máxima frecuencia de transmisión del sistema (100 muestras/s/canal), de las cuales se derivan las etapas necesarias para el funcionamiento del sistema, teniendo presente los requerimientos por parte de la Escuela de Ingeniería Mecánica, que son:

Un sistema para adquirir datos de cuatro señales analógicas entre ±10V no diferenciales y transmitirlos por bus USB (V 1.1) de baja velocidad (*low speed*),empleando un conversor analógico/digital con resolución de 16 bits por muestra, alimentado por una fuente alterna de 110V a 60Hz.

De esta forma el ***Sad800-L*** está distribuido en cuatro etapas principales:

- Etapa de potencia
- Etapa de aislamiento
- Etapa de conversión
- Etapa de transmisión por bus USB

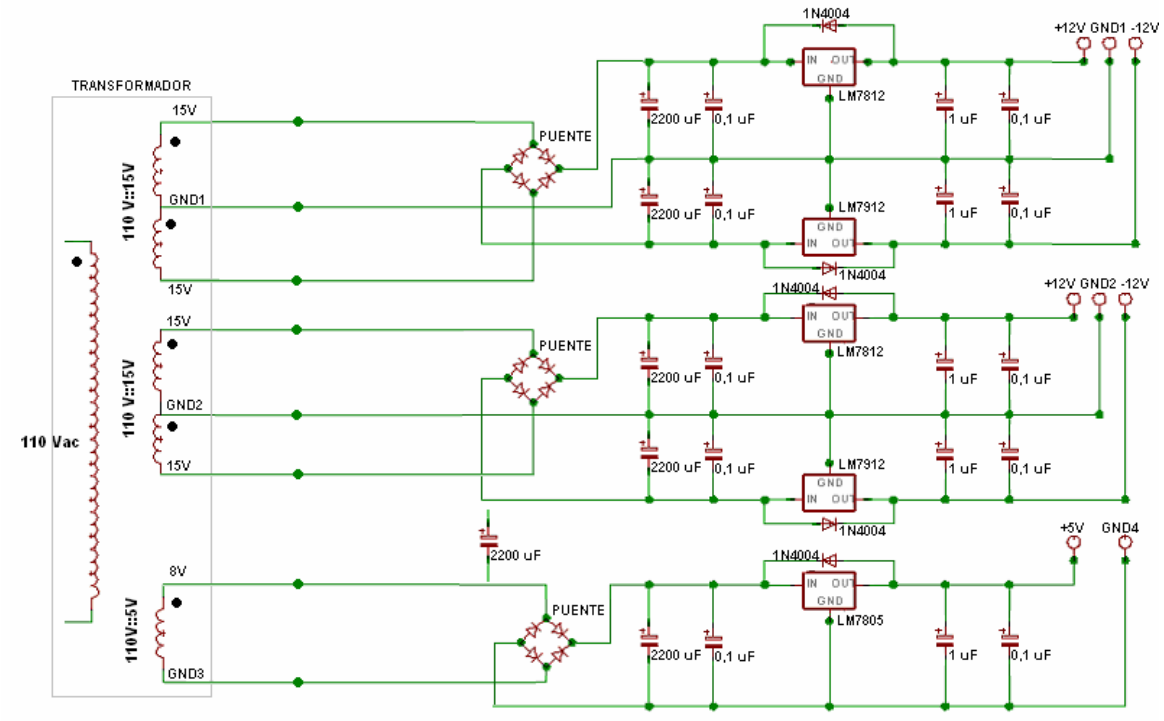
4.2 ETAPA DE POTENCIA

Esta fase tiene en cuenta bs requerimientos de las etapas siguientes, en cuanto a la potencia eléctrica que demanda cada elemento, y los requerimientos especiales en la etapa de aislamiento.

El diseño propuesto requiere la implementación de dos fuentes duales de ±12V para alimentar la etapa de aislamiento, y dos fuentes de +5V para alimentar los dispositivos digitales y analógicos del sistema, con 3 referencias distintas (tierra

analógica, tierra digital y tierra analógica de entrada.). Para este propósito se construyó una fuente de alimentación que opera a 110V, tensión que es reducida por un transformador de tres devanados con tap central, a niveles de tensión de ±15V, y ±8V (niveles requeridos por los reguladores de voltaje). Las señales rectificadas (rectificación de onda completa con rectificación con puente de diodos), se acoplan a través de un par de condensadores que actúan como filtro pasabajos (con el fin de minimizar el rizado) a los respectivos reguladores: LM7812, LM7912 y LM7805. Además se emplean condensadores para el desacople de alta frecuencia y permiten mantener los niveles a la salida de los reguladores, tal como lo muestra la Figura 20.

Figura 20. Esquemático fuente de alimentación tarjeta USB



Fuente: Los autores.

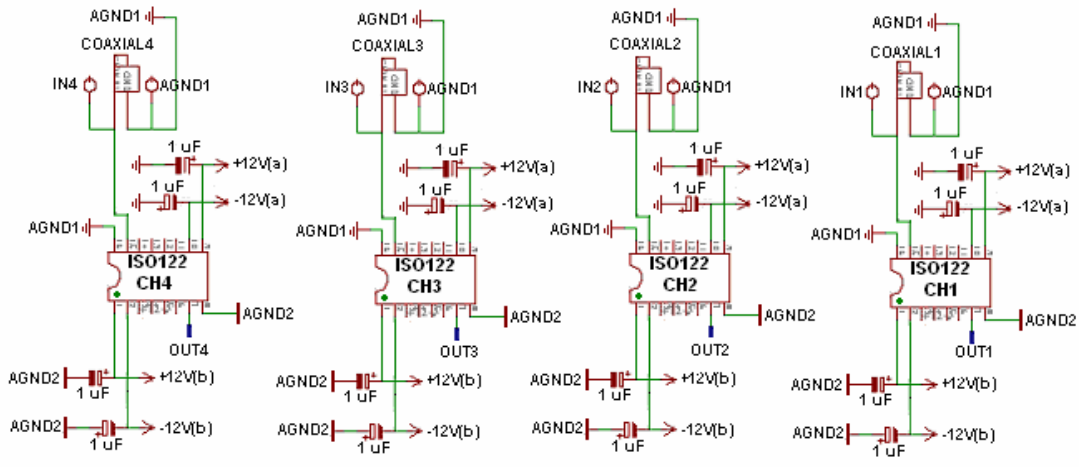
4.3 ETAPA DE AISLAMIENTO

En el diseño de la tarjeta de adquisición de datos se utiliza un amplificador de aislamiento que protege al resto del sistema de entradas con sobretensión hasta de 2400Vrms utilizando niveles de alimentación de ?12V.

En esta fase se emplearon amplificadores de aislamiento ISO122 por cada canal de entrada; cuenta con una sección de aislamiento galvánica en medio de la entrada y la salida; al igual que un sistema de modulación para la señal de entrada (pin 15), y demodulada a la salida (pin 7) para obtener una señal idéntica a la de entrada.

La configuración típica (sin ganancia de voltaje) con conexiones de alimentación (fuente dual de ±12V) requeridas lo muestra la Figura 21.

Figura 21. Esquemático Etapa de aislamiento tarjeta de adquisición USB



Fuente: Los autores.

4.4 ETAPA DE CONVERSIÓN A/D

Debido a las condiciones de diseño (transmitir datos de cuatro señales analógicas), y la limitación de no contar con un conversor A/D embebido en el encapsulado del microcontrolador, se implementó una etapa de conversión analógica/digital externa que cumple satisfactoriamente con los requerimientos de diseño.

4.4.1 Etapa de conversor analógico digital

En el mercado de los semiconductores, grandes fabricantes como *Texas Instruments*, *Analog Devices*, entre otros, ofrecen a sus consumidores conversores A/D de 8 bits,

12 bits, 16 bits, etc; que trabajan a tasas de reloj del orden de los Hz hasta los MHz. Actualmente con estas características de resolución y rapidez, la etapa de conversión permite mayor flexibilidad de diseño para la transmisión de datos a través de interfaces tan rápidas como **USB** (V 1.1 *Full Speed* , V 2.0 *High Speed*) y ***FireWire***.

La elección de un conversor de datos no depende sólo de la interfaz utilizada (serie/ paralela), sino también del *transceiver* (transmisor/receptor) USB empleado. Teniendo en cuenta la tasa de bits de USB 1.1 (1.5Mbps), y resolución de 16 bits por muestra, se estima teóricamente, una frecuencia máxima de muestreo para una señal analógica, tal como lo muestra la ecuación (1).

$$\frac{1.5 Mbps}{16 bits/muestra} \; ? \; 93.75 \frac{kmuestras}{segundo} \quad (1)$$

De igual forma, para cuatro canales teóricamente, la máxima frecuencia de muestreo se muestra en la ecuación (2).

$$\frac{93.75 \frac{kmuestras}{segundo}}{4 Canales} \; ? \; 23.437 \frac{kmuestras}{segundo} \quad (2)$$

Para este sistema en particular se eligó un conversor analógico/digital, que cumple con los siguientes requerimientos:

- Cuatro señales analógicas de entrada, bipolares en un rango entre ±10V.
- Resolución de 16 bits.
- Frecuencia de muestreo no superior a 23 $\frac{kmuestras}{segundo}$ por canal.
- Interfaz de datos serial (SPI ó I2C) o paralela.
- Conversión de datos en modo continuo para las señales de entrada.
- Señales de entrada referenciadas a tierra.

Teniendo en cuenta estos parámetros, se muestra en la **tabla 3** algunos de los conversores A/D disponibles en el mercado y que se asemejan a estas necesidades:

Tabla 3. Alternativas comerciales de conversores analógicos/digitales para el sistema SAD800-L.

REF	*FABRICANTE*	*RESOLUCIÓN*	*FRECUENCIA DE MUESTREO*	*INTERFAZ DE SALIDA DE DATOS*	*ALIMENTACIÓN*	*CANAL DE ENTRADA*		*POTENCIA*
AD 7701	***Analog Devices***	16 bits	$4\frac{kmuestras}{segundo}$	Serial (SPI)	Fuente dual ±5V	Una entrada unipolar 2.5V	Una entrada bipolar de 2.5V	37mW
ADS 7807	*Texas Instruments*	16 bits	$40\frac{kmuestras}{segundo}$	Serial (SPI)/ Paralela		Una entrada analógica unipolar	Una entrada analógica bipolar	28mW
ADS 7825P	***Texas Instruments***	16 bits	$40\frac{kmuestras}{segundo}$	Serial (SPI)/ Paralela	Fuente +5V	---	4 señales analógicas entre ±10V	50mW

Fuente: Los autores.

El **ADS 7825P** de *Texas Instruments* es un conversor de datos A/D, que cuenta con entradas para cuatro señales analógicas entre ±10V; muestreando a una tasa de 25µs, con una resolución de 16 bits por muestra y utilizando la técnica de aproximaciones sucesivas para su funcionamiento.

El sistema de adquisición utiliza el **ADS7825P,** ya que es un dispositivo que satisface las necesidades de diseño: un solo integrado posee las cuatro señales de entrada bipolares en los rangos de tensión requeridos, muestreo en modo continuo para las cuatro señales, resolución de 16 bits; dos interfaces (serial/paralelo) de datos, una sola fuente de alimentación, moderado consumo de potencia y una frecuencia de muestreo que le garantiza al microcontrolador tener los ocho bytes correspondientes a las muestras de los cuatro canales (la máxima demora es de cuatro períodos de muestreo) y dentro del tiempo estipulado por el protocolo USB para la transmisión de paquetes; asegurando así que dicha transmisión corresponda a muestras cuasiperíodicas.

Configuración del conversor ADS7825P

La etapa de conversión se bifurca en múltiples opciones de implementación al contar con: dos interfaces de salida de datos (serie/paralelo), modo de conversión manual y continuo, y lectura de datos durante y después de la conversión.

La descripción completa del funcionamiento de cada una de las configuraciones se encuentra en la hoja de datos del conversor ADS7825P (ver anexo C), y presentan ventajas y desventajas para la implementación en el sistema de adquisición de datos como lo muestra la **tabla 4.**

El sistema de adquisición, cuenta con un microcontrolador **MC68HC908JB8** que actúa como controlador del sistema de adquisición de datos; posee 20 pines de los cuales once (11) son entradas / salidas programables, además de un pin de interrupción ($\overline{IRQ}$), que están disponibles para acoplar la etapa de conversión con la etapa de transmisión; por lo tanto la decisión de acoplar la etapa de adquisición de los datos con la etapa de conversión analógico/digital a través de una interfaz de tipo serial o paralela debe tener en cuenta la cantidad de entradas/ salidas programables disponibles.

La alternativa serial tiene una ventaja en este aspecto, debido a la cantidad de pines requeridos para su ejecución; pero la velocidad a la cual transmiten los bits muestreados de las cuatro señales de entrada es más elevada respecto a la velocidad de procesamiento del microcontrolador **MC68HC908JB8** empleado (ver tabla 4).

La alternativa de implementar una interfaz de salida de datos de tipo serial requiere de una temporización de datos interna o externa, para la conversión y adquisición de los datos (ver anexo C).

La temporización interna utiliza el reloj interno del conversor de datos (f =900kHz. (anexo X.1)), acoplado a la etapa de transmisión por medio del microcontrolador

MC68HC908JB8 que trabaja a una frecuencia de 1.5MHz, no es suficiente para leer los datos a la salida del conversor analógico/digital y enviarlos al PC.

La temporización externa utiliza un reloj externo (DC ? f ? 10MHz), esto quiere decir, que se requiere de un sistema adicional que determine la frecuencia de adquisición de los datos (ya que éste no es un reloj de conversión (ver anexo C), y que además cumpla con los requerimientos de diseño mencionados al inicio de este numeral; además de presentar sensibilidad al ruido de conmutación, que provoca degradación en el desempeño del conversor de datos.

El modo de conversión de datos significa que la selección del canal a muestrear puede ser manual (externa) o continua (interna). El modo manual de conversión implica un control total del dispositivo y creación de un nuevo canal (*pipe out*) de comunicación por parte del módulo USB (V1.1) del microcontrolador **MC68HC908JB8**; además se debe tener en cuenta que el bus USB (V1.1) es asíncrono, con una tasa de transferencia de datos efectiva de 800 bytes por segundo, lo que disminuiría aún más el ancho de banda disponible para transmisión.

Por otra parte, el modo de conversión continua muestrea cíclicamente las cuatro señales de entrada analógicas sin necesidad de un control estricto por parte del microcontrolador, sin reducir la tasa efectiva de envío de datos.

Analizando la implementación, desempeño y optimización de recursos en cada una de las alternativas presentadas, se diseñó un sistema de adquisición de datos que en su etapa de conversión presenta una configuración con interfaz de datos de salida paralela, en modo continuo.

Tabla 4. Ventajas y desventajas de los diferentes modos de operación del Conversor analógico/digital ADS 7825P.

ALTERNATIVA		VENTAJAS	DESVENTAJAS
INTERFAZ DE SALIDA DE DATOS	SERIAL (SPI)	✍ Menor cantidad de pines necesarios para implementación en el sistema de adquisición.	✍ Elevada tasa de bits para la adquisición de los mismos por parte del microcontrolador. ✍ Necesita una señal de sincronismo para la toma de datos, que se presenta sólo una vez para cada muestra. ✍ Requiere un reloj de datos (ver temporización interfaz serial*), que demanda monitoreo por parte de un dispositivo externo (microcontrolador)
	PARALELO	✍ Velocidad de transmisión de los datos digitalizados (8 bits), apropiada al sistema de adquisición . ✍ Los datos están presentes durante las fases de conversión y adquisición. ✍ No necesita de una señal de sincronismo para el envío de datos.	✍ Mayor número de pines dedicados para la implementación en el sistema de adquisición.
MODO DE CONVERSIÓN	MANUAL	✍ Se puede elegir el canal que se desea muestrear en un momento determinado. ✍ Variación de la frecuencia de muestreo (aumenta), cuando se muestrea un número menor de canales por segundo.	✍ Se requiere de un dispositivo externo (microcontrolador) que controle los canales a muestrear.
	CONTINUO	✍ Frecuencia de muestreo fija para cada canal. ✍ No demanda un dispositivo externo (microcontrolador) que tenga el control de los canales a muestrear.	✍ No se puede elegir el orden, ni el número de canales a muestrear.
(*)TEMPORIZACION IINTERFAZ SERIAL	EXTERNA	✍ Permite el control externo de la velocidad de lectura de los bits correspondientes a cada muestra.	✍ Sensible al ruido de conmutación por parte de un reloj asíncrono, que provoca degradación en el desempeño del conversor de datos. ✍ Requiere la implementación de un reloj externo
	INTERNA	✍ La implementación del reloj interno se logra colocando en bajo el pin 12 del conversor analógico/digital.	✍ Demanda monitoreo de la señal por parte de un dispositivo externo (microcontrolador)

Fuente: Los autores.

Temporización de conversión de datos

El conversor de datos ADS7825P en cualquier configuración debe contar con una señal de entrada tipo pulso ($R/\overline{C}$) que determina la frecuencia de muestreo del

conversor analógico/digital (ver figura 2 del Anexo C). Para lograr este propósito se implementa un circuito temporizador externo que está acoplado al conversor de datos y al microcontrolador.

La figura 22 muestra el esquemático del sistema de temporización de conversión, donde dos multivibradores redisparables **74HC123** (ver anexo D) generan los pulsos requeridos por el conversor de datos, a la frecuencia en la cual se van a muestrear las señales de entrada.

En la etapa de adquisición, el monoestable 1 del multivibrador 74HC123 genera los pulsos de adquisición de datos a la entrada BYTE del conversor ADS 7825P (pin 21) y al microcontrolador MC68HC908JB8JP (pin 6); esta retroalimentación con el microcontrolador le asegura la toma de datos válidos a la del conversor ADS 7825P .

El segundo monoestable genera los pulsos de conversión a la entrada $R/\overline{C}$ del conversor ADS 7825P (pin 22). Parámetros tales como ancho de pulso de conversión y adquisición están definidos en la hoja de datos del conversor ADS 7825P (ver anexo C).

El ancho de pulso de las señales que intervienen en la etapa de conversión, está dado por la ecuación (3), donde el valor mínimo para R_x es 5k? y para C_x es 0 F (Anexo D):

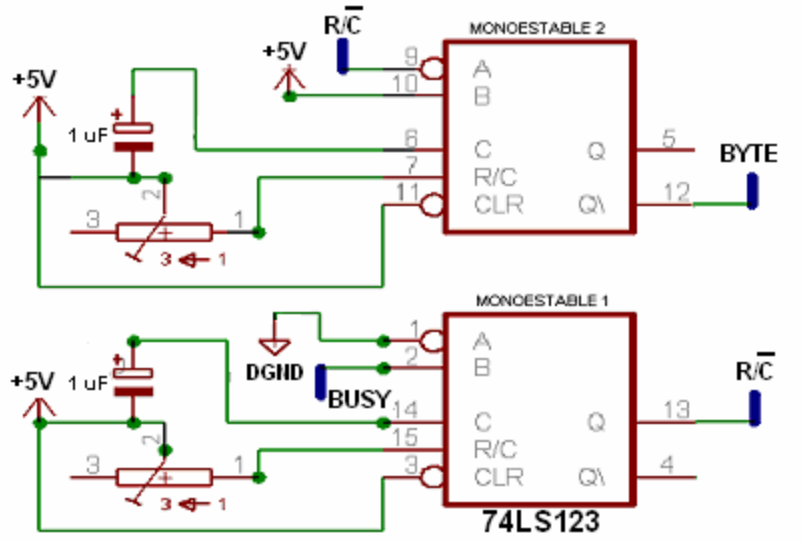
$$t_w \; ? \; 0.45 * R_x * C_x \qquad (3)$$

La tarjeta de adquisición de datos, está implementada con valores de condensadores fijos y resistencias variables (*trimmers*) con el propósito de brindar al usuario la posibilidad de ajustar diferentes frecuencias de muestreo en un solo dispositivo (monoestable 2); cabe anotar que esta variación es desarrollada por hardware; sin

embargo, esta es una señal de salida del dispositivo que permite al usuario monitorear fácilmente la frecuencia a la cual el sistema está muestreando las señales de entrada.

Por otra parte, esta variación debe ser conjunta con el primer monoestable, ya que la frecuencia de adquisición de datos válidos cambia de acuerdo a la variación del pulso de conversión $R/\overline{C}$.

Figura 22. Circuito de temporización del ADS 7825P en el sistema de adquisición de datos USB



Fuente: Los autores

4.5 ETAPA DE TRANSMISIÓN POR BUS USB

Esta fase es la que determina en gran medida la configuración y los elementos implícitos en las etapas anteriores, y permite disponer de las características básicas de un dispositivo USB (V1.1) entre las cuales se destacan: conexión y funcionamiento sin configuración previa (*Plug & Play)*, inserción en caliente, de uso portátil serial a una tasa de 1.5Mbps.

La versión más reciente de la interfaz USB es la 2.0, que admite una tasa de transferencia de 480Mbps y a su vez soporta los estándares anteriores (V1.0 y V1.1),

y otros tipos de transferencias como isócronas y de volumen (*Bulk*), lo que permite al desarrollador implementar aplicaciones interesantes (humano-máquina) con esta interfaz.

Es necesario resaltar que al comienzo de este proyecto, distintas empresas distribuían microntroladores que incorporan un módulo USB (V1.1) y sólo hasta ahora se ha comenzado la comercialización de dispositivos con módulo USB (V2.0); entre otras, una razón por la cual el sistema de adquisición trabaja con la versión 1.1 de la interfaz USB.
Por otra parte, en la E3T no se había trabajado en el diseño de dispositivos que utilicen este puerto, lo que implicaba un estudio preliminar fundamentado en el manejo del protocolo USB, razón por la cual el empleo de una versión del estándar menos compleja (V1.1) es lo más apropiado para desarrollar el sistema de adquisición maximizando los recursos que ofrece la versión USB de baja velocidad (V1.1).

Definida la versión con la cual la etapa de adquisición trabaja; en el mercado distintas empresas como: Philips, Motorola, Microchip, entre otras, han desarrollado microcontroladores con módulo USB embebido en el hardware que permite el desarrollo de aplicaciones para la descarga flexible de datos.

Motorola ofrece una gama de microcontroladores de la familia HC08 que responden a las características de manejo de USB (MC68HC908JB8, MC68HC08JB8, MC68HC08JT8); cada uno cuenta con un número de entradas/salidas y encapsulado diferentes de los cuales el **MC68HC908JB8JP** es el más comercial y asequible para este sistema de adquisición.

4.5.1 Configuración del microcontrolador MC68HC908JB8JP Motorola

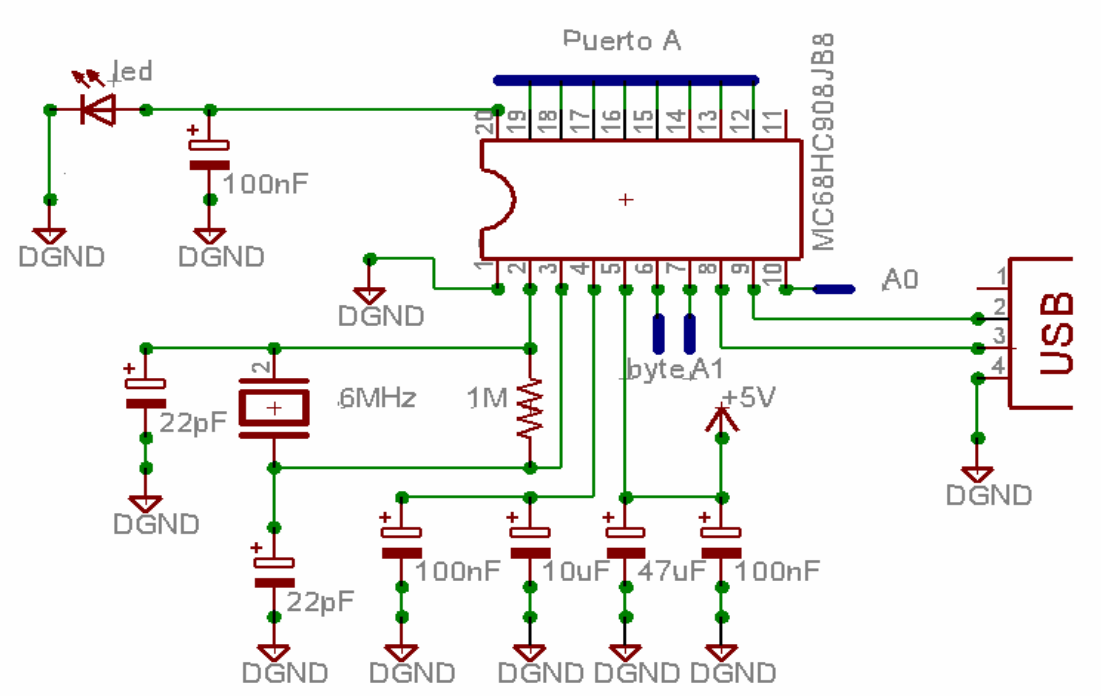
El microcontrolador de 8 bits **MC68HC908JB8JP** de Motorola, es un circuito integrado, con encapsulado DIP de 20 pines que tiene embebido en el hardware un

módulo USB (V1.1) que permite el manejo de este bus serial[23]; los detalles de distribución y funcionamiento se encuentran en el manual de referencia del microcontrolador **MC68HC908JB8** de Motorola[24].

La figura 23 muestra el microcontrolador **MC68HC908JB8** implementado en modo aplicación.

Los pines de tierra y alimentación, DGND (pin1), VCC (pin5) respectivamente, están referenciados a una fuente digital (+5V) diseñada en la etapa de potencia y acoplada en el sistema a través de condensadores de 100nF.

Figura 23. Esquemático de la etapa de transmisión de datos a través del bus USB (V.1.1)



Fuente: Los autores

[23] Ver numeral 2.1.
[24] MOTOROLA. M68HC08 Microcontrollers. Technical Data. Rev. 2, 2/2002.

El cristal externo de 6MHz en conjunto con dos condensadores de 22pF y una resistencia de 10M? conforman el circuito oscilador (pines 2, 3)[25]. Esta frecuencia es la que determina la tasa a la cual el dispositivo procesa instrucciones; además, ya que el microcontrolador cuenta con una arquitectura con repertorio complejo de instrucciones (CISC), implica que en el mejor de los casos (instrucciones cortas), la velocidad a la cual se ejecutan las instrucciones es la cuarta parte de la frecuencia del oscilador, es decir, aproximadamente cada 0,6?s se realiza una instrucción de programa.

Tomando en cuenta los requisitos de diseño mencionados en la etapa de conversión y la tasa efectiva de bits por segundo que puede transmitir USB (V 1.1), se estima teóricamente la frecuencia máxima a la que es posible muestrear una señal analógica, sin pérdida de datos en la etapa de transmisión como lo muestra la ecuación (4).

$$\frac{6.4kbps}{16bits} ? 400\frac{muestras}{segundo} \quad (4)$$

Así mismo, para cuatro canales se estima teóricamente, una frecuencia de muestreo máxima por canal de acuerdo a la ecuación (5).

$$\frac{400\frac{muestras}{segundo}}{4Canales} ? 100\frac{muestras}{segundo}/canal \quad (5)$$

El reinicio del programa (reset) (pin 20) está conectado a la tierra digital, a través de un condensador de 100nF, con un pulsador que permite comenzar nuevamente la aplicación. Los datos de entrada se toman del conversor analógico/digital a través del puerto A del microcontrolador MC68HC908JB8JP (pines del 12 al 19). La señal BYTE del conversor ADS 7825P, es la que permite encuestar y adquirir los datos

[25] Ver numeral 2.4.2

válidos, está conectada al pin 6 (de propósito general) del microcontrolador MC68HC908JB8JP.

De igual forma, la dirección de canal muestreado (A0 y A1) que es una salida del conversor ADS7825P, se lleva al microcontrolador a través de los pines 10, y 7 respectivamente. Por último, la transmisión a través del bus USB (V1.1) se realiza por medio de un conector tipo B[26] como salida del sistema, acoplado a los pines 8 (D+) y 9 (D-) del microcontrolador.

[26] Ver numeral 1.1.3.

5. FUNCIONAMIENTO DEL SISTEMA DE ADQUISICIÓN DE DATOS USB 1.1 Sad800-L

5.1 IMPLEMENTACIÓN

El sistema de adquisición de datos USB 1.1 Sad800-L se basa en el microcontrolador **MC68HC908JB8JP** de Motorola, tal como se mencionó en el capítulo anterior. Éste se encarga de tomar los datos del conversor analógico digital y enviarlos al PC a través del bus USB 1.1, tal y como se describe en el diagrama de flujo de la Figura 25.

5.1.1 DIAGRAMA DE FLUJO DEL PROCESO DE ADQUISICIÓN

Con el propósito de cumplir con los requerimientos de velocidad de transmisión y un buen desempeño es posible diseñar un dispositivo dedicado o no dedicado. Desde el punto de vista del software, es equivalente a implementar un programa en el que la atención de eventos se realiza mediante encuesta (dedicado) o interrupción (no dedicado). La primera opción reduce las repercusiones en la tasa de transmisión, debido a que el tiempo requerido para realizar la encuesta es menor que el tiempo de conversión de datos.

La segunda alternativa resulta menos ventajosa ya que el módulo de integración del sistema (SIM) en el control de excepciones[27] del microcontrolador MC68HC908JB8JP tiene dispuestos niveles de prioridades para cada excepción, que enumeradas en orden descendente son: atención a interrupciones de ruptura, $\overline{IRQ}$, USB, y otras; es decir, las interrupciones de mayor prioridad mantienen

[27] MOTOROLA. M68HC08 Microcontrollers. Technical Data. Rev. 2, 2/2002.

constantemente ocupada la CPU del microcontrolador impidiendo así la transmisión de datos por medio del bus USB (V 1.1).

Teniendo en cuenta este análisis, el sistema de adquisición está implementado como un dispositivo dedicado a la transmisión de datos USB (V1.1) de cuatro señales analógicas de entrada a una tasa efectiva de 800 bytes por segundo; que es la velocidad más alta posible con esta versión del bus[28]. Es importante resaltar que aunque el bus USB no está diseñado para transferencias síncronas, el Sad800-L garantiza transferencia de datos en tiempo real, que si bien se realiza a una tasa relativamente baja comparada con otros dispositivos en el mercado (ver Tabla 4), es posible incrementarla fácilmente utilizando versiones USB de mayor velocidad[29].

De esta manera, se dispone de un sistema de conversión que se inicializa al momento de ser encendido, y que cuenta con temporización externa, la cual permite controlar la frecuencia de conversión del ADS7825P de forma completamente independiente de la velocidad de adquisición de datos por parte de la CPU del microcontrolador[30].

Bloque de inicio

Las rutinas de este bloque incluyen el código de inicialización (*firmware*) del dispositivo, el cual habilita la comunicación con el manejador (*host*) y otros periféricos. Se encarga principalmente de configurar el hardware y permitir al dispositivo el intercambio de datos necesarios en ambas direcciones; además debe chequear periódicamente el tipo de transferencia que se envía al dispositivo, en particular si se refiere a un paquete *Setup*, ante el cual debe responder abandonando la transferencia en curso y atendiendo la demanda presente.

---

[28] AXELSON, Jan. USB Complete. LakeView Research. 2ª ed. 2001. Pág 53.

[29] Al iniciar el desarrollo de este proyecto no se disponía comercialmente de microcontroladores que contaran con módulo USB de mayor velocidad.

[30] Para la programación del microcontrolador se empleó *CodeWarrior Development Studio V2.1 SE*, de Metrowerks, el cual incluye las herramientas necesarias para la configuración y manejo del módulo USB.

De tal forma que: se inicializan los registros comunes de la CPU, se habilitan las interrupciones, el dispositivo y el módulo USB, permitiendo la comunicación dispositivo/manejador (*host*) para su configuración y reconocimiento.

Configuración y reconocimiento

La aplicación envía a través del manejador (*host*) comandos requeridos por el dispositivo USB (V1.1) tal y como lo estipula el protocolo, como son:

- Get_Port_Status: establece un canal de comunicación entre el dispositivo y el bus.
- Get_Descriptor: envía una petición del máximo tamaño de paquete del canal de control (*Endpoint 0*)
- Set_Address: asigna una dirección al dispositivo conectado.
- Get_Descriptor: envía una petición de la información del descriptor de dispositivo que ha direccionado.
- Set_Configuration: una vez el manejador (*host*) conoce las principales características del dispositivo, envía a éste el número de configuración correspondiente.

Este proceso de reconocimiento del dispositivo USB por parte del manejador (*host*) es totalmente oculto y transparente al usuario (*Plug & Play*).

El descriptor del dispositivo USB V1.1 debe ser identificado por el manejador (*host*) para lo cual este último debe contar con el controlador (*driver*) apropiado. Estos controladores se pueden ser:

- ***Drivers* genéricos**: para dispositivos estándar, que están instalados en el sistema operativo.

- ***Drivers* propietarios:** desarrollados por empresas para dispositivos de propósito específico, o en desarrollo.

Se emplearon controladores de dos empresas diferentes como alternativas para controlar el sistema de adquisición de datos.

- **Thesycon**: empresa alemana que está desarrollando dispositivos USB y ofrece a los desarrolladores paquetes de software en lenguaje de alto nivel para el control de dispositivos USB (V 1.1 , USBIO V1.51 , V 2.0 USBIO development kit 2.0)
- **National Instruments**: VISA (Virtual Instrument Software Aplication) y LabView 7.0 permiten también el desarrollo de aplicaciones para controlar dispositivos USB (V1.1, V2.0).

Cualquiera de las dos alternativas es viable y la aplicación en lenguaje de alto nivel es la que determina el controlador empleado. Una descripción detallada de las peticiones enviadas por el manejador (*host*) al dispositivo se pueden encontrar en la bibliografía relacionada con el tema[31].

Cuerpo del código

Corresponde a la etapa de adquisición de datos válidos en el conversor analógico/digital.

La señal de adquisición de datos efectivos (BYTE) a la entrada del conversor ADS7825P (pin 21) y salida del primer monoestable, es la que permite encuestar y leer los *bytes* de información mediante el puerto A del microcontrolador MC68HC908JB8JP (pines 12 al 19).

[31] Ver AXELSON, Jan. USB Complete. Lakeview Research. 2a edición, 2001.

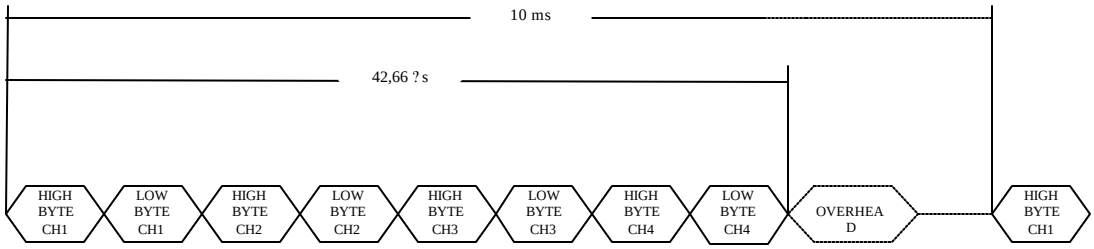
El *byte* alto del dato del canal muestreado por el conversor ADS7825P, se lee cuando al encuestar el estado de la línea BYTE (pin 21) del conversor ADS7825P es en bajo, y el *byte* bajo de la misma muestra se toma cuando el estado de la línea BYTE (pin21) del conversor ADS7825P está en alto (ver figura 2 del Anexo C).

Transmisión por bus USB (V1.1)

En este caso particular el módulo USB (V1.1) del microcontrolador, cuenta con un canal de comunicación de entrada (*pipe IN*) que permite el envío de datos en el momento en que la aplicación del manejador (*host*) habilite el canal y comience la lectura[32].

La transmisión se realiza en bloques de ocho (8) *bytes*/*frame* a una tasa promedio de 10 ms/frame (ver Figura 24). De esta forma, por cada ocho *bytes* de datos se envía adicionalmente información de control, estado y chequeo de errores (*overhead*). Esto determina la máxima velocidad de transferencia empleando el protocolo USB V1.1, la cual es de 800 bytes por segundo.

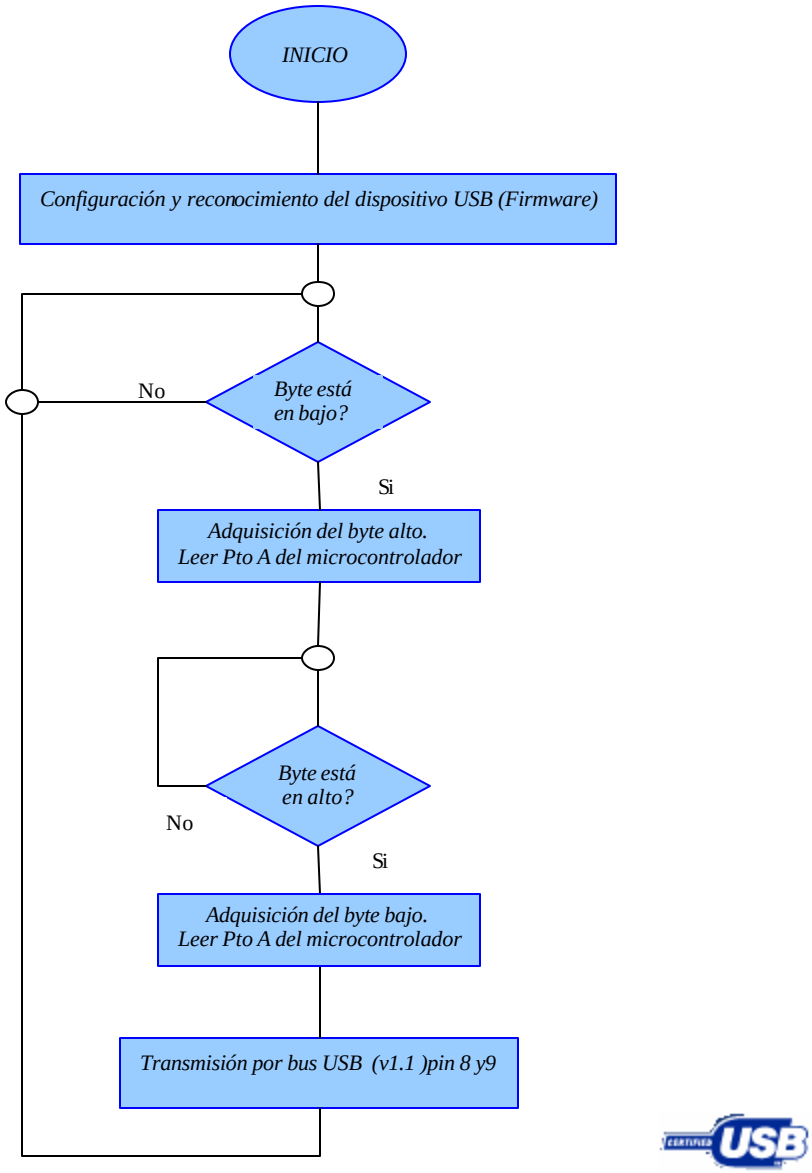
Figura 24. Velocidad de transmisión de datos USB V1.1



[32] Los detalles del manejo de paquetes y campos de paquetes entre dispositivo y manejador (*host*) se encuentran en el libro USB Complete (ver nota anterior)

Así, de manera continua y sin bifurcaciones para evitar saltos y demoras innecesarias, se ejecuta un único lazo en el programa principal que garantiza el envío de datos de las cuatro señales analógicas muestreadas a una tasa aproximada de 10kHz por canal y trasmitidas por bus USB (V1.1) a una tasa efectiva de 6400bps.

Figura 25. Diagrama de flujo implementado en el microcontrolador MC68HC908JB8JP de Motorola.



Fuente: Los autores

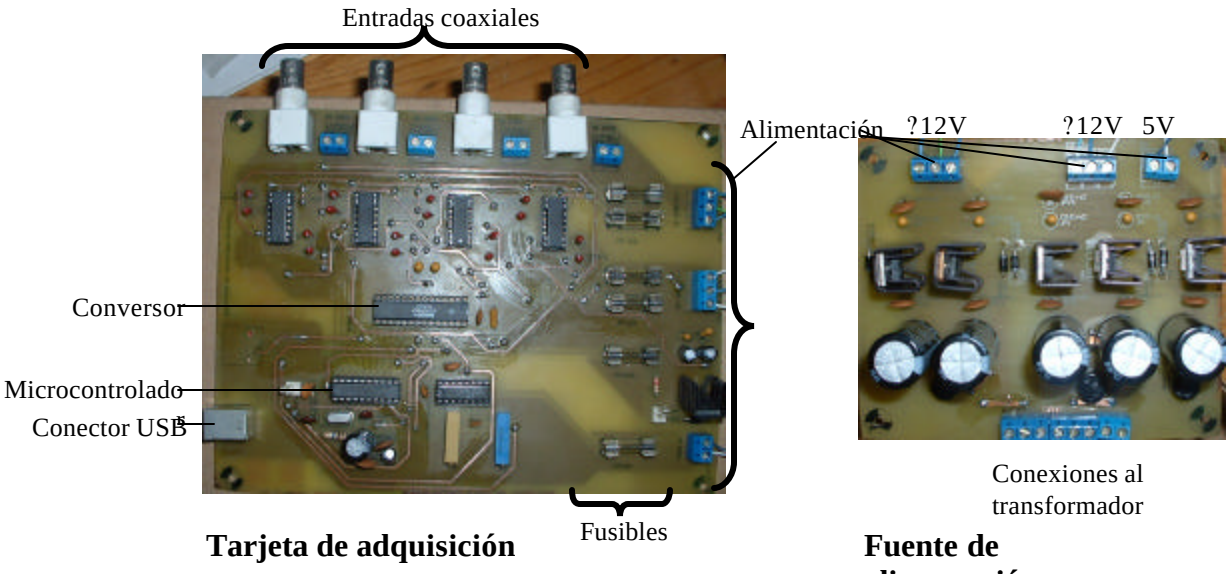
5.1.2 Tarjeta de adquisición de datos por bus USB V1.1: Sad800-L

El Sad800-L está diseñado pensando en aprovechar las ventajas del bus USB V1.1 en aplicaciones de baja velocidad que requieren el uso de un dispositivo portátil de fácil conexión y configuración. Además, es posible migrar fácilmente hacia versiones de USB más rápidas, que se han constituido en poco tiempo en uno de los estándares más populares para la conexión de periféricos y dispositivos a computadores personales.

En la Figura 26 se muestra el Sad800-L, el cual posee las siguientes características:

- Cuatro (4) canales analógicos referenciados a tierra (SE) con aislamiento galvánico.
- Un (1) puerto de comunicación USB 1.1 (Low Speed) que permite una tasa efectiva de transmisión 800 bytes/segundo.
- Una (1) salida de sincronismo que permite monitorear la frecuencia de muestreo del conversor ADS7825P.
- Un (1) pulsador para reiniciar la aplicación implementada en el microcontrolador MC68HC908JB8JP (*RESET*).
- Un (1) led indicador de encendido.
- Alimentación tripolar monofásica (120V).
- Frecuencia de muestreo variable entre 5,55 $\frac{kmuestras}{segundo}$ y 33,3 $\frac{kmuestras}{segundo}$ para los cuatro canales.
- Resolución efectiva de13 bits.
- Señales de entrada bipolares en el rango de ±10V.
- Transmisión de datos por bus.
- Controlador y aplicación con Labview 7.0
- Dispositivo *Plug&Play*
- Permite inserción en caliente.
- Dimensiones: 24cm x 18 cm x 12 cm.

Figura 26. Fuente de alimentación y tarjeta de adquisición de datos del Sad800-L



Fuente: Los autores

5.2 RESULTADOS OBTENIDOS

5.2.1 Conversor de datos ADS7825P

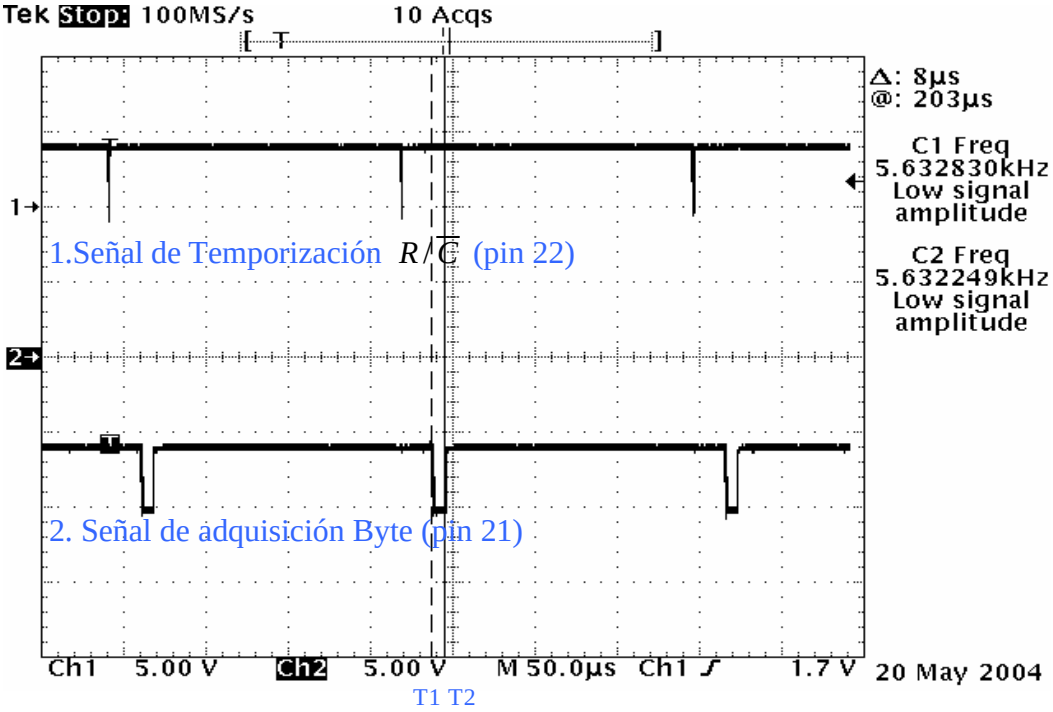
En la etapa de adquisición del sistema ***Sad800-L***, el conversor analógico/digital ADS7825P requiere señales de temporización y adquisición (Figura 27). Estas señales corresponden a:

Lectura / Conversión. $R/\overline{C}$ (*Read / conversión* (pin 22)) Señal periódica de temporización que determina la frecuencia de muestreo del sistema. Teóricamente, empleando la expresión que determina el ancho de pulso para los multivibradores del circuito integrado 74HC123, (encargados de la generación de esta señal) se tienen períodos de muestreo mínimo y máximo entre los 6kHz y los 40kHz respectivamente; en la práctica se obtienen frecuencias de muestreo mínima de 5.55kHz y máxima de 33.3kHz, variando el potenciómetro del monoestable 2.

Byte (pin 21) Señal periódica de adquisición que permite la lectura de los datos a la salida del conversor analógico/digital después de realizar la conversión, de tal manera que en la caída de Byte se toma los 8 bits más significativos, y en la subida de Byte se toman los 8 bits restantes que corresponden al byte menos significativo.

El ancho de pulso de adquisición varía entre 2?s y 7.6?s ajustando el potenciómetro del primer monoestable.

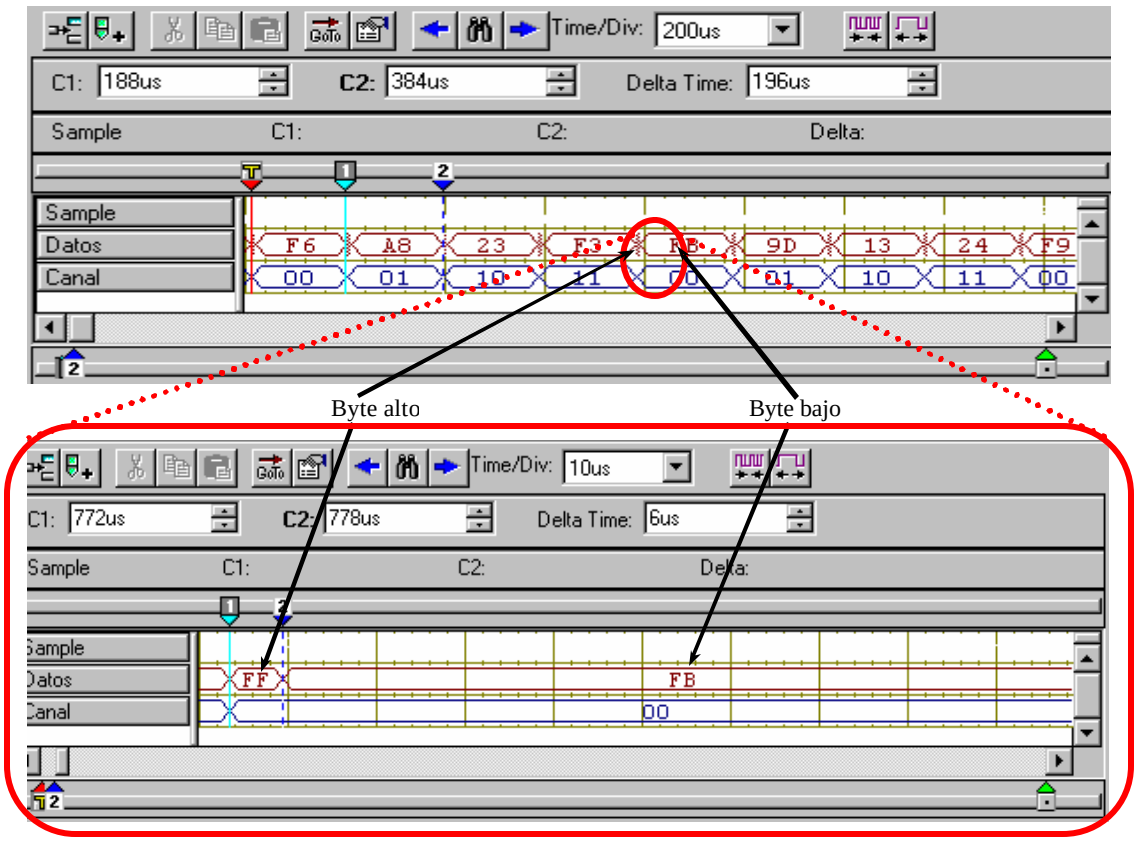
Figura 27. Señales de temporización y adquisición del conversor ADS7825P.



Fuente: Los autores.

Como resultado de las señales de temporización y adquisición el conversor analógico/digital ADS7825P genera los bits de datos (D7-D0) y la información del siguiente canal (A1-A0) tal como lo muestra la Figura 28.

Figura 28. Señales de datos y dirección de canal a la salida del conversor ADS7825P.

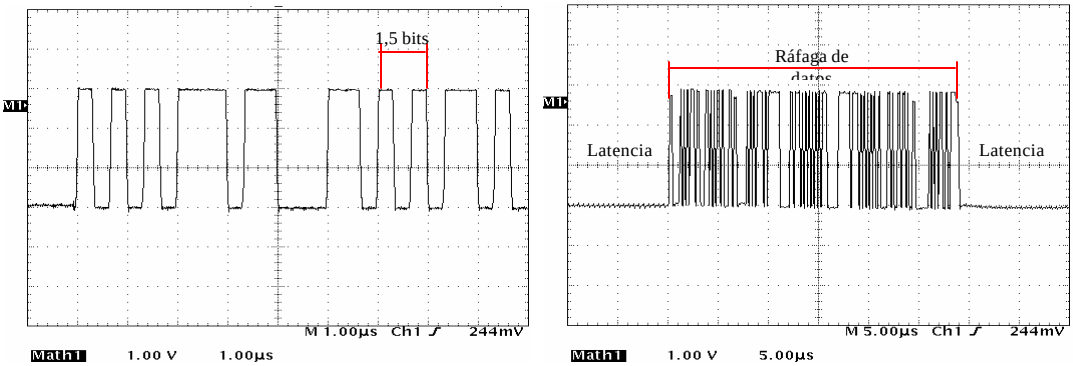


Fuente: Los autores

5.2.2 Módulo USB (V1.1) del microcontrolador MC68HC908JB8JP

Las líneas de transmisión de datos D^+ y D^- (pines 8 y 9) del módulo USB 1.1 embebido en el microcontrolador MC68HC908JB8JP, son las encargadas físicamente de enviar las señales de tensión de forma diferencial (codificación NRZI) (Figura 29) a través del cable USB al manejador (*host*) del PC, donde se observa un paquete de información por parte del dispositivo a las peticiones de datos por parte del *host*.

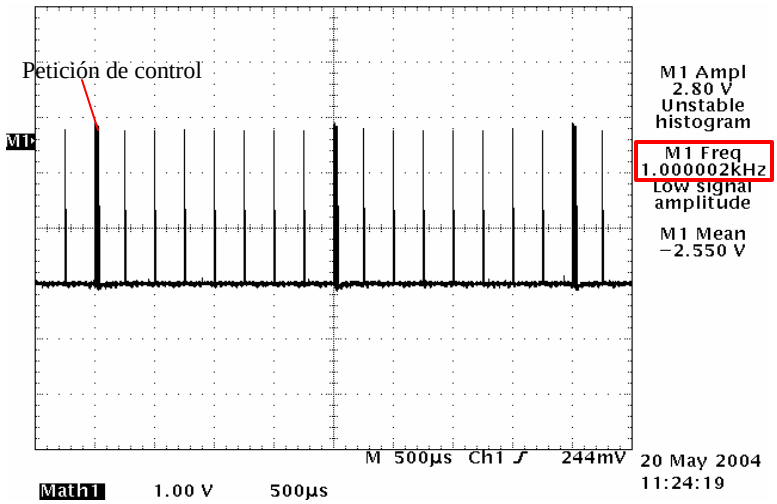
Figura 29. Señal de tensión diferencial USB 1.1 (D^+ y D^-)



Fuente: los autores.

La Figura 30 muestra las señales de tensión diferenciales USB V1.1 en un rango de tiempo de 5 ms; allí se pueden observar las peticiones de control (realizadas a través del *Pipe 0*), que se intercalan con las peticiones de interrupción cada 10ms a través del canal de comunicación de entrada (*Pipe IN*) por medio de las cuales se realiza el envío de datos hacia el manejador (*host*).

Figura 30. Señal de tensión diferencial USB 1.1



Fuente: los autores

Las tensiones que manejan las señales diferenciales que se envían a través del cable USB se muestran en la Tabla 5.

Tabla 5. Valores de tensión para señales USB 1.1 baja velocidad.

Parámetro	Baja Velocidad (Low Speed) (V)
V_{out} bajo - mínimo	0 V
V_{out} bajo - máximo	0.3 V
V_{out} alto - mínimo	2.8 V
V_{out} alto - máximo	3.6 V

Fuente: AXELSON, Jan. USB complete.

5.2.3 Señales adquiridas

Se realizaron siete pruebas empleando el sistema completo de adquisición Sad800-L, con el fin de verificar el monitoreo de señales de tensión analógicas entre ? 10V y frecuencias entre los 0 y 50 Hz en el dominio del tiempo y la frecuencia.

PRUEBA NÚMERO 1.

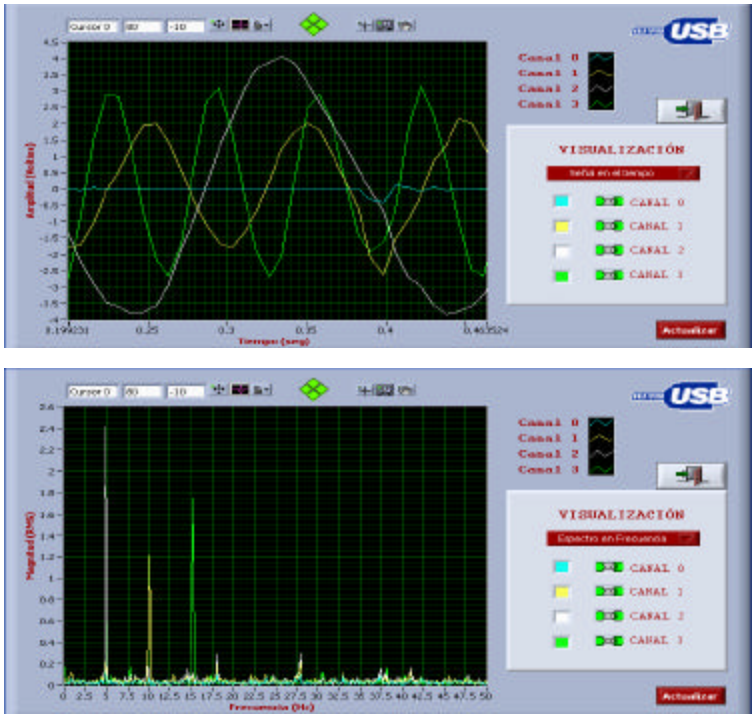
Descripción: adquirir cuatro señales de tensión analógicas con el *Sad 800-L*.

Procedimiento: Se generaron cuatro señales senoidales con niveles de tensión, y frecuencias distintas (Tabla 6).

Tabla 6. Valores de tensión y frecuencia para la prueba 1.

Canal	Tensión (Vpp)	Frecuencia (Hz)
Cero	0 V	0 Hz
Uno	4 V	10 Hz
Dos	8 V	5 Hz
Tres	6 V	15Hz

Figura 31. Resultado de la prueba número uno.



Fuente: Los autores

Comentario: en esta prueba se puede apreciar el comportamiento del sistema en frecuencias muy por debajo del límite de operación propuesto donde se observa una distorsión armónica que no excede el 13% y un nivel de ruido que oscila entre 2 ?V y 20 mV.

PRUEBA NÚMERO 2

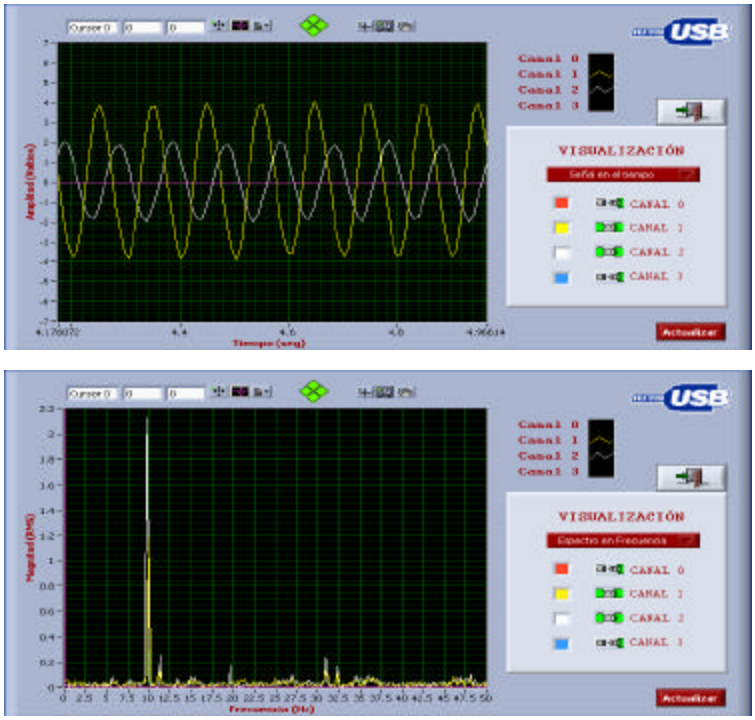
Descripción: adquirir dos señales de tensión analógicas con el *Sad800-L*.

Procedimiento: se generaron dos señales senoidales con niveles de tensión diferentes y la misma frecuencia (Tabla 7).

Tabla 7. Valores de tensión y frecuencia para la prueba 2.

Canal	Tensión (Vpp)	Frecuencia (Hz)
Uno	8 V	10 Hz
Dos	4 V	10 Hz

Figura 32. Resultados de la prueba número dos.



Fuente: Los autores

Comentario: se observa cómo la medida de frecuencia es fiable para diferentes canales y los niveles de ruido y distorsión se mantienen en un margen tolerable.

PRUEBA NÚMERO 3

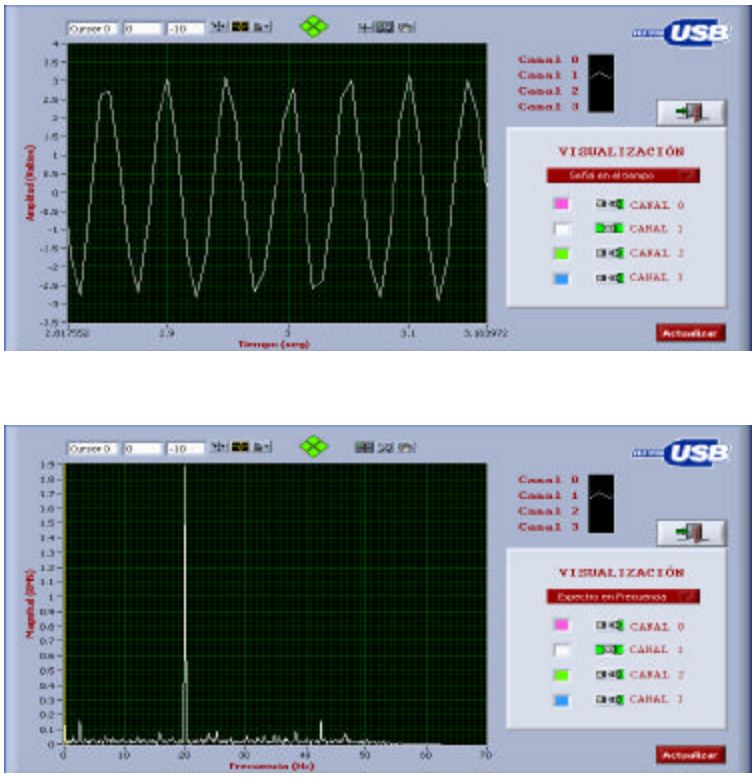
Descripción: adquirir una señal de tensión analógica con el *Sad 800-L* con una frecuencia de muestreo de 30?s.

Procedimiento: se generó una señal senoidal con nivel de tensión y frecuencia (Tabla 8).

Tabla 8. Valores de tensión y frecuencia para la prueba 3.

Canal	Tensión (Vpp)	Frecuencia (Hz)
Dos	6 V	20 Hz

Figura 33. Resultados de la prueba número tres.



Fuente: Los autores

PRUEBA NÚMERO 4

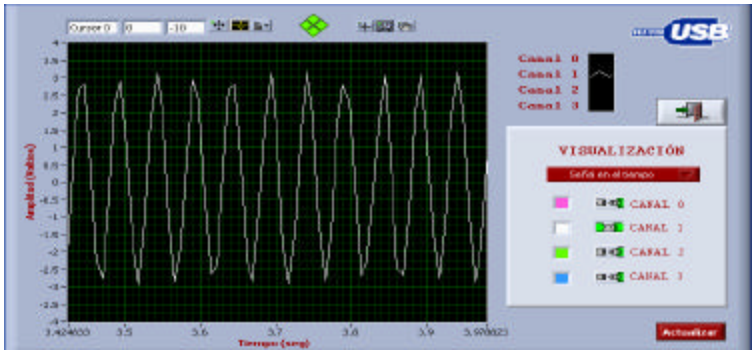
Descripción: adquirir una señal de tensión analógica con el *Sad 800-L* con una frecuencia de muestreo de 60?s.

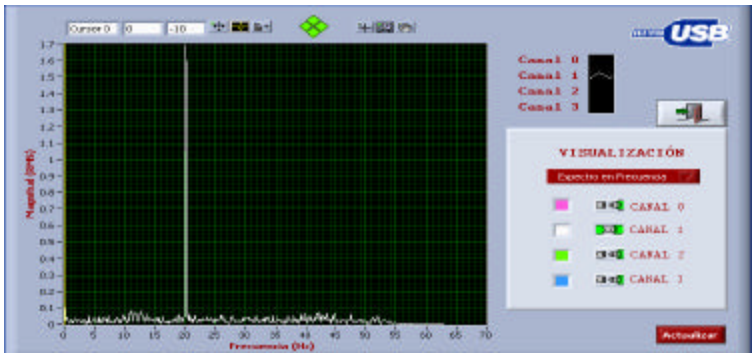
Procedimiento: Se generó una señal senoidal con nivel de tensión y frecuencia (Tabla 9).

Tabla 9. Valores de tensión y frecuencia para la prueba 4.

Canal	Tensión (Vpp)	Frecuencia (Hz)
Dos	6.5 V	20 Hz

Figura 34. Resultados de la prueba número cuatro.





Fuente: Los autores

PRUEBA NÚMERO 5

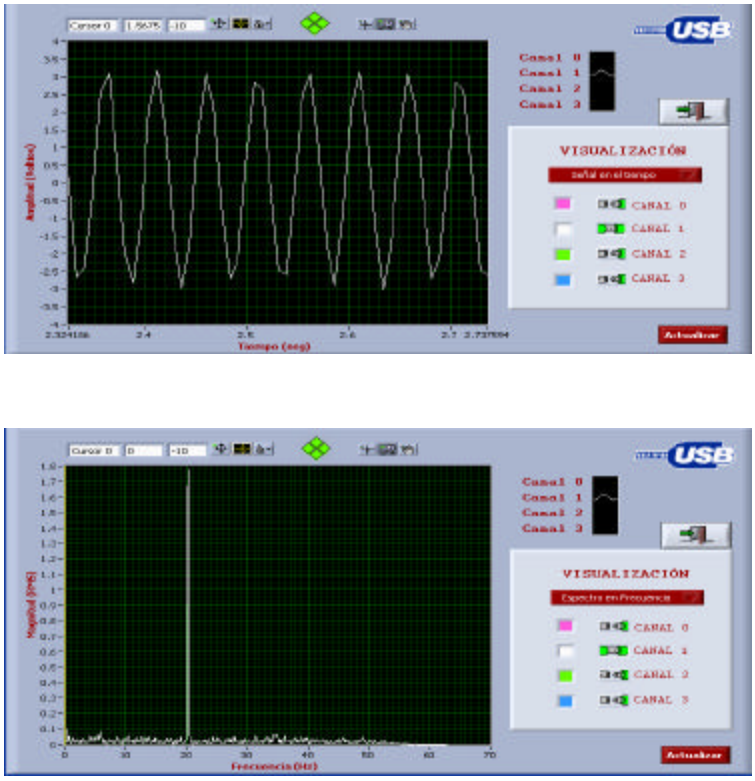
Descripción: adquirir una señal de tensión analógica con el *Sad 800-L* con una frecuencia de muestreo de 120?s.

Procedimiento: Se generó una señal senoidal con nivel de tensión y frecuencia (Tabla 10).

Tabla 10. Valores de tensión y frecuencia para la prueba 5.

Canal	Tensión (Vpp)	Frecuencia (Hz)
Dos	6 V	20 Hz

Figura 35. Resultados de la prueba número cinco.



Fuente: Los autores

PRUEBA NÚMERO 6

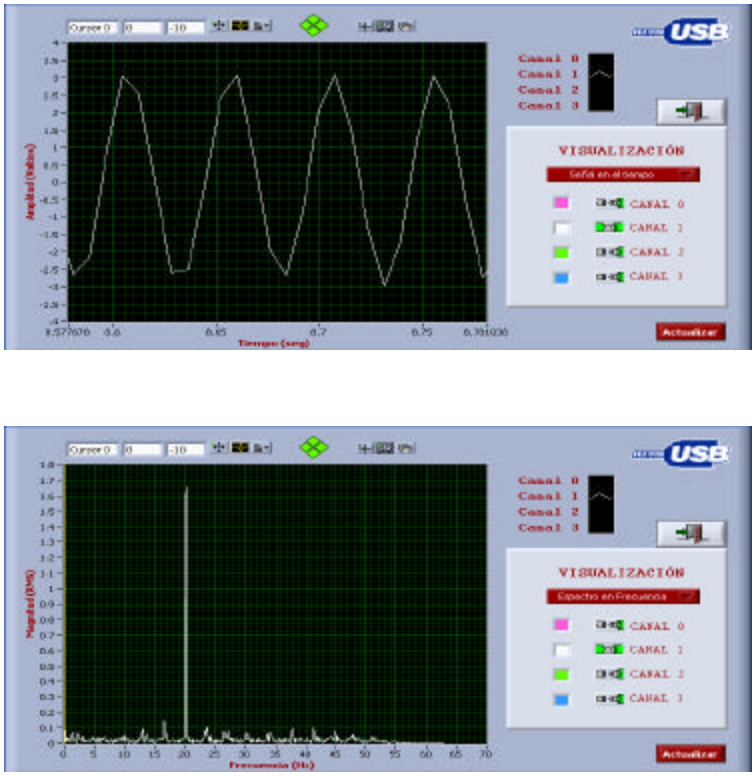
Descripción: adquirir una señal de tensión analógica con el *Sad 800-L* con una frecuencia de muestreo de 180?s.

Procedimiento: Se generó una señal senoidal con nivel de tensión y frecuencia (Tabla 11).

Tabla 11. Valores de tensión y frecuencia para la prueba 6.

Canal	Tensión (Vpp)	Frecuencia (Hz)
Dos	6 V	20 Hz

Figura 36. Resultados de la prueba número seis.



Fuente: Los autores

Comentario: se observa que a pesar de variar la frecuencia de muestreo desde los límites mínimo y máximo, las medidas de las señales en el dominio del tiempo y la frecuencia corresponden a los parámetros de la señal analógica de entrada, es decir el tener un sistema sobremuestreado garantiza que el error en la tasa de adquisición sea mínima.

PRUEBA NÚMERO 7.

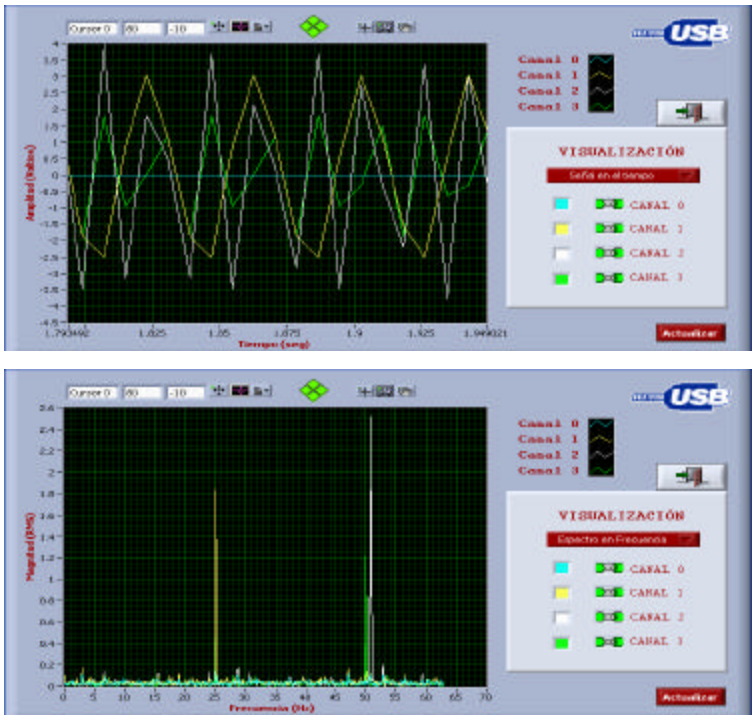
Descripción: adquirir cuatro señales de tensión analógicas con el *Sad 800-L*.

Procedimiento: Se generaron cuatro señales senoidales con niveles de tensión, y frecuencias distintas (Tabla 12).

Tabla 12. Valores de tensión y frecuencia para la prueba 7.

Canal	Tensión (Vpp)	Frecuencia (Hz)
Cero	0 V	0 Hz
Uno	6 V	25 Hz
Dos	8 V	50 Hz
Tres	4 V	75Hz

Figura 37. Resultados de la prueba número siete.



Fuente: Los autores

Comentario: esta prueba se verifica la frecuencia de muestreo límite del sistema de adquisición de datos *Sad 800-L* que corresponde a 50Hz; ya que al introducir una señal de tensión de frecuencia de 75Hz por efecto de solapamiento el sistema la muestra una señal de frecuencia de 51Hz.

5.3 COMPARACIÓN CON OTROS DISPOSITIVOS

Teniendo en cuenta el tipo de sistema que se va a monitorear o controlar y/o la aplicación que se desea implementar, se definen los requerimientos de tasa de transmisión, resolución y puerto empleado para el envío de datos. Aunque, la tendencia hacia interfaces de comunicación más flexibles y versátiles como USB (1.1, 2.0) se acentúan y superan a las existentes (serial, paralelo), lo que implica que es el momento de comenzar a desarrollar sistemas de adquisición que se estén en el camino de los avances en puertos de comunicación.

En este caso, se diseñó el Sad800-L, un sistema de adquisición que emplea el bus USB (V1.1) de baja velocidad (Low Speed) en una aplicación de diagnostico de emisión de gases de motores Diesel (Diagma D-100), que monitorea señales de baja velocidad (0Hz- 60Hz) y que se ajustan a la tasa de transferencia efectiva del estándar 1.1 (Low Speed). En la Tabla 13 se hace una comparación entre algunos sistemas de adquisición de datos y sus aplicaciones.

Tabla 13. Comparación con otros sistemas de adquisición

Tarjeta	Conversores estáticos de potencia	MRS	DAQpad6020E	Sad800-L
Interfaz de comunicación	PARALELO	SERIAL	USB 2.0	USB 1.1
Frecuencia máxima de muestreo	46.3kHz	1.2kHz	100kHz	400Hz
Número de canales analógicos.	0 SE / 4 DIF	3 SE / 0 DIF	16 SE / 8 DIF	4 SE / 0 DIF
Rango de las señales de entrada.	0V a 700V	0V a 10 V	Bipolar entre ±0.05V a ±10V	Bipolar entre ±10V
Alimentación	Dos fuentes duales de ±15V y una fuente de +5V	Batería	120 VAC	120 VAC
Resolución	8 bits	12 bits	12 bits	16 bits
Aplicaciones	Control de un sistema trifásico de potencia	Medidor de resistividad de suelos	Tarjeta de adquisición de datos de alta velocidad	Diagnóstico de motores Diesel
Particularidades	Adquisición por puerto paralelo empleando DMA.	Adquisición por puerto serie SCI.	Adquisición por bus USB 2.0 (*High Speed*)	Adquisición por bus USB 1.1 (*Low Speed*).

Fuente: Los autores.

6. INTERFAZ SAD800-L

La Interfaz SAD 800-L consta de dos aplicaciones que pueden ejecutarse simultáneamente, una principal que fue diseñada en LabView 7.0 Express y una secundaria diseñada en Visual Basic 6.0. Cada una de estas aplicaciones tiene una función particular y están descritas en los apartes 5.2 y 5.3 de este capítulo.

6.1 FUNCIONAMIENTO

Al iniciar la aplicación principal de LabView, cuya interfaz se muestra en la Figura 38, se carga automáticamente la aplicación de Visual Basic, cuya interfaz se muestra en la Figura 39; las dos se encuentran activas al mismo tiempo, permitiendo a LabView graficar los datos adquiridos con la aplicación de Visual Basic (a partir de un archivo generado por Visual Basic) o cargar una gráfica con datos obtenidos por el programa para la tarjeta de adquisición de datos en sesiones anteriores.

Figura 38. Interfaz de la aplicación de LabView.

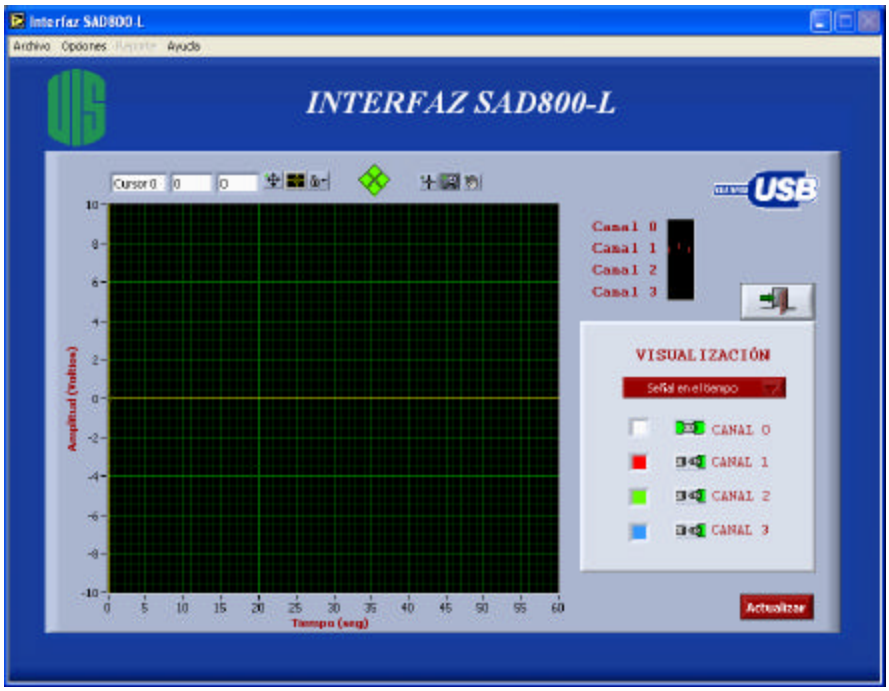
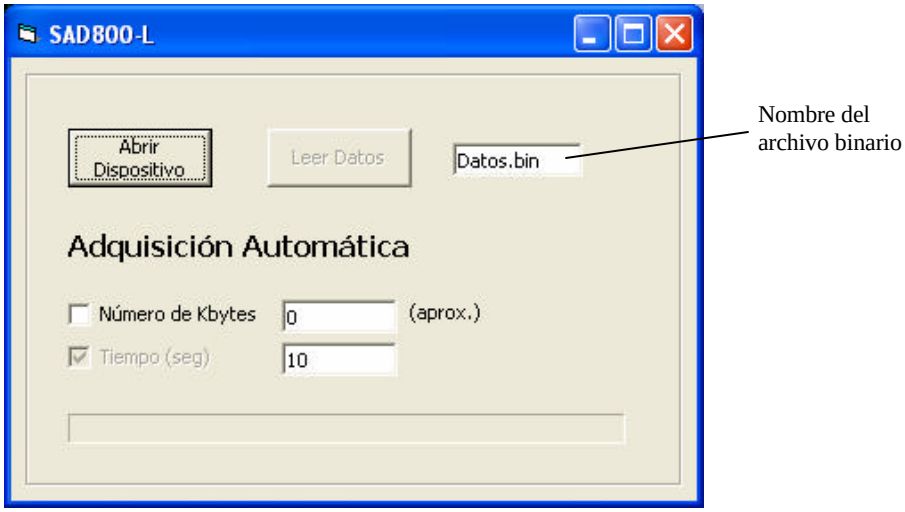


Figura 39. Interfaz de la aplicación de Visual Basic.



6.1.1 Adquirir y graficar los datos transmitidos por el SAD800-L.

Los datos son adquiridos por la aplicación de Visual Basic y guardados en un archivo binario que se crea dentro de la carpeta donde se encuentra la aplicación. Es posible cargar esta aplicación las veces que sea necesario haciendo clic en el ítem "`Adquirir datos`" del menú "`Archivo`" de la Interfaz SAD800-L, que abre un cuadro de diálogo indicando reiniciar la tarjeta SAD800-L y luego ejecuta la aplicación de Visual Basic. Para iniciar la adquisición de datos se debe hacer clic en el botón "`Abrir Dispositivo`", que realiza todos los procedimientos y operaciones necesarias con el *driver* USBIO para habilitar un canal[33] que permita iniciar la comunicación, y si la operación anterior fue exitosa, se habilita el botón "`Leer Datos`". Cuando se habilita el botón "`Leer Datos`", es posible iniciar y detener la lectura cuantas veces sea necesario haciendo clic en dicho botón; los datos son guardados en archivos binarios a los cuales se les puede asignar el nombre en la casilla de texto de la interfaz de la aplicación (ver Figura 39). Si el nombre en la casilla de texto corresponde a un archivo binario existente dentro de la carpeta donde

[33] Ver aparte 5.2 del libro.

se encuentra la aplicación, se sobrescribe en éste. Los detalles de su funcionamiento se encuentran en el aparte 5.2 de este capítulo.

La lectura de datos se puede detener de forma manual o automática, indicando el tiempo de adquisición en segundos o el número de Kbytes aproximado a recibir. Adjunto a los datos, se guarda el tiempo de adquisición en el archivo binario.

Para graficar los datos a partir del archivo binario generado por Visual Basic, en la aplicación principal de LabView (Interfaz SAD800-L) se hace clic en el ítem "Graficar datos" del menú "Archivo" de la interfaz. Los datos se muestran en la gráfica de la Interfaz SAD800-L. Es posible adquirir y graficar datos cuantas veces se requiera sin límite de tiempo.

Los datos desplegados en la gráfica de la aplicación principal de LabView se pueden guardar como gráfica de LabView (*.lvm), haciendo clic en "Guardar" del menú "Archivo" de la Interfaz SAD800-L. Una vez se hace clic en la opción "Guardar", se abre un cuadro de diálogo donde se asigna la ruta, el nombre y la extensión deseada a la gráfica y se hace clic en el botón "OK".

6.1.2 Cargar una gráfica de LabView (*.lvm) almacenada

Las gráficas que se almacenan en formato gráfica de LabView (*.lvm) (como se describe en el aparte anterior) se pueden visualizar haciendo clic en "Cargar gráfica" del menú "Archivo" de la aplicación principal de LabView. A continuación se abre un cuadro de diálogo que permite escoger la ruta y el nombre del archivo que contiene la gráfica que se desea visualizar. Cuando se escoge la ruta y el archivo, la gráfica es desplegada en la gráfica de la Interfaz SAD800-L.

6.1.3 Cerrar la Interfaz SAD800-L

La aplicación principal de LabView se puede cerrar de dos maneras diferentes: la primera es haciendo clic en "`Salir`" del menú "`Archivo`" de la Interfaz, que permite detener la aplicación y salir de LabView, y la segunda es haciendo clic en el botón "`Salir`" de la interfaz que detiene la aplicación, pero no cierra LabView.

La aplicación de Visual Basic es independiente, por lo que se cierra únicamente si se hace clic en el botón para cerrar su ventana (botón de la esquina superior derecha de la ventana).

6.1.4 Opciones adicionales de la Interfaz SAD800-L

La aplicación principal de LabView tiene tres opciones adicionales que facilitan el análisis de los datos obtenidos por la tarjeta de adquisición de datos. Estas opciones son: visualización de los datos, filtrado de las señales adquiridas y trigger (disparo), explicadas en el aparte 6.3.7 de este capítulo. También permite generar reportes; esta opción se explica en el aparte 6.3.9 de este capítulo.

6.2 APLICACIÓN DE VISUAL BASIC 6.0 PARA EL SAD800-L (Aplicación.exe)

En la Figura 39 se muestra la interfaz de la aplicación de Visual Basic 6.0. Cuando se ejecuta **Aplicación.exe**, se carga la librería de vínculos dinámicos (dll) usbiocom.dll incluida en el paquete del *driver* de Thesycon[34]. La librería de vínculos dinámicos **usbiocom.dll** debe colocarse en la subcarpeta **system32** de la carpeta WINDOWS (que se encuentra en el directorio raíz del disco duro donde está instalado Windows) y debe registrarse empleando el comando "`regsvr32 usbiocom.dll`" en la ventana "`Ejecutar`" que se abre haciendo clic en el programa "`Ejecutar…`" del

[34] Ver aparte 3.1 del libro.

menú de "`Inicio`" de Windows. Si la librería no está registrada, cuando se ejecuta la aplicación, aparece una ventana de error indicando esta situación. Para solucionar este inconveniente, se debe registrar la librería, y ejecutar nuevamente el programa.

Para iniciar la adquisición de datos es necesario, en primer lugar, iniciar la comunicación con el dispositivo USB (tarjeta de adquisición de datos SAD800-L) haciendo clic en el botón "`Abrir dispositivo`". Los métodos y propiedades de la librería de vínculos dinámicos **usbiocom.dll** empleados para abrir la comunicación con la tarjeta de adquisición de datos se muestran en la Tabla 14 y su secuencia es presentada en la Figura 40.a.

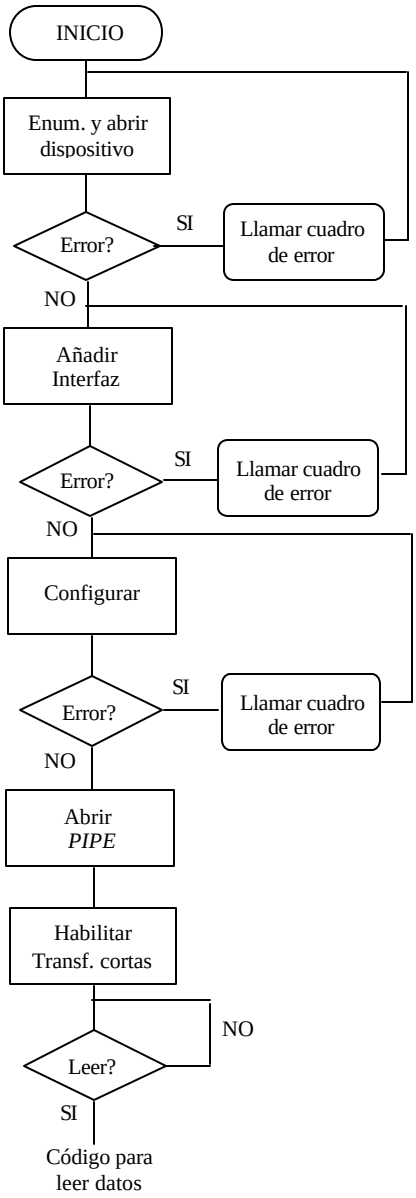
Tabla 14. Tabla de propiedades y métodos de usbiocom.dll para iniciar la comunicación con la tarjeta de adquisición de datos.

Acción	Método o propiedad de usbiocom.dll
Enumerar dispositivo	`EnumerateDevice` (método)
Abrir dispositivo	`OpenDevice` (método)
Añadir interfaz[35]	`AddInterface` (método)
Configurar	`SetConfiguration` (método)
Abrir *Pipe* (Canal)	`Bind` (método)
Habilitar transferencias cortas	`ShortTransferOK` (propiedad)

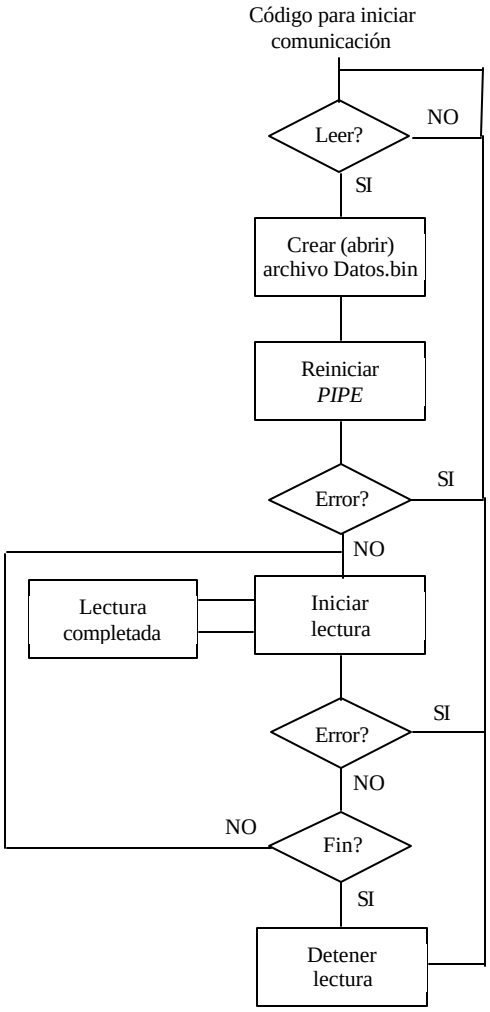
Una vez abierta la comunicación, se activa el botón "`Leer Datos`" que permite adquirir los datos enviados por el bus USB y guardarlos por defecto en el archivo binario **Datos.bin** creado dentro del directorio donde se encuentra la aplicación de Visual Basic. Si se desea guardar los datos que serán adquiridos con otro nombre, se debe digitar el nombre del archivo en el cuadro de texto de la aplicación con el siguiente formato **nombre del archivo.bin**, y luego hacer clic en el botón "`Leer Datos`". Si se realiza una nueva adquisición de datos con un nombre de archivo binario existente, se sobrescribe en el archivo y los datos de la adquisición anterior se pierden.

[35] Ver aparte 1.3.3 del libro.

Figura 40. a) Diagrama de flujo para iniciar la comunicación con la tarjeta de adquisición de datos. b) Diagrama de flujo para leer los datos de la tarjeta de adquisición de datos.



a)



b)

Los métodos y eventos empleados para adquirir los datos de la tarjeta de adquisición de datos se muestran en la Tabla 15 y su secuencia es presentada en la Figura 40.b.

Tabla 15. Métodos y eventos de usbiocom.dll empleados para adquirir datos.

Acción	Método o propiedad de usbiocom.dll
Reiniciar *PIPE*	ResetPipe (método)
Iniciar lectura	StartReading (método)
Lectura completada	ReadComplete (evento)
Detener lectura	StopReading (método)

Cuando se hace clic en el botón "`Leer Datos`", se activa el método *StartReading* del driver, con el cual se inicia una transmisión continua de datos desde el dispositivo hacia un *buffer* ubicado en el *host*, cuyo tamaño depende de los valores de entrada que se asignen al método *StartReading*.

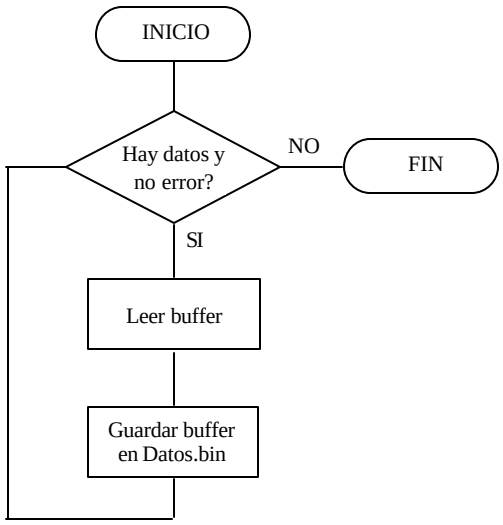
Para finalizar la adquisición de datos se debe hacer clic en el botón "`Detener`" (corresponde al botón "`Leer Datos`" después de iniciar la lectura) de la aplicación de Visual Basic (ver Figura 39), el cual marcará el fin de la transmisión con el método *StopReading* (detener lectura).

El evento "*ReadComplete*" (Lectura Completada) es llamado automáticamente por el *driver* cuando el *host* llena un *buffer* con los datos transmitidos por el dispositivo. Cuando este evento ocurre, se debe implementar una rutina de Lectura completada para que la aplicación lea los datos de los *buffers* y los coloque por defecto en el archivo binario **Datos.bin** (o en otro archivo binario dependiendo del nombre asignado en el cuadro de texto de la interfaz), dentro del directorio donde se encuentra la aplicación. Los métodos empleados durante el evento "`Lectura Completada`" se muestran en la Tabla 16 y su diagrama de flujo es presentado en la Figura 41.

Tabla 16. Métodos empleados durante el evento "Lectura completada"

Acción	Método o propiedad de usbiocom.dll
Leer datos de un buffer	ReadData (método)

Figura 41. Diagrama de flujo del evento Lectura Completada

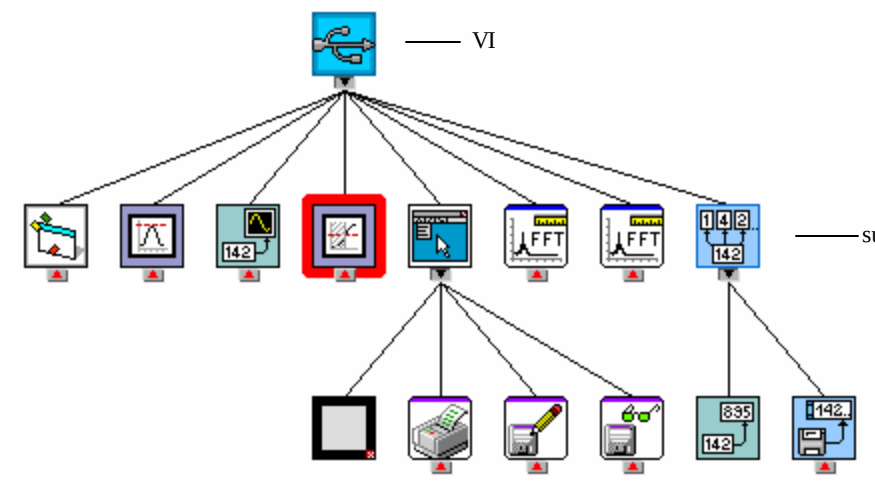


6.3 APLICACIÓN PRINCIPAL DE LABVIEW 7.0 (SAD800-L.vi)

La aplicación de LabView se encarga de cargar los datos guardados en el archivo binario generado en Visual Basic, y de realizar a éstos el tratamiento necesario para desplegarlos como las señales de voltaje obtenidas por la tarjeta de adquisición de datos. Su interfaz se muestra en la Figura 35.

La aplicación está constituida por los archivos de LabView (VI's – Instrumento Virtual y subVI's – VI's empleados dentro de otro VI) descritos en los siguientes apartes de este capítulo. En la Figura 42 se observa la gráfica de la jerarquía del VI y los subVI's principales empleados para la aplicación de LabView.

Figura 42. Jerarquía del VI y los SubVI's básicos de la aplicación principal de LabView para la tarjeta de adquisición de datos por bus USB.



6.3.1 Ejecutar la aplicación de Visual Basic (subVI – CallVisual.vi).

Este VI de LabView toma la ruta donde se encuentra la aplicación de Visual Basic Aplicación.exe empleando el subVI **Current VI's Path.vi** y la convierte en una cadena de caracteres (*string*). A esta cadena le agrega otros caracteres para formar el comando: `cmd /x /start "C:\(ruta de Aplicación.exe)" "C:\(ruta de la Aplicación.exe)"` que permite ejecutar la aplicación si se coloca como entrada al VI **System Exec.vi** (su función es ejecutar el comando de entrada en la ventana de símbolo del sistema.

6.3.2 Adquirir los datos desde un archivo (subVI – TomaDatos.vi)

Este subVI adquiere los datos del archivo binario indicado en la ruta de entrada, les cambia el formato a cadena de caracteres (*string*), elimina los caracteres que no contienen datos y recorta la cadena de caracteres (el tamaño de la cadena debe ser par y separa el tiempo de adquisición de los datos) para que el siguiente subVI pueda organizar los datos correctamente.

6.3.3 Ordenar y dar formato a los datos (subVI – OrdenarDatos.vi)

 Este subVI contiene al subVI TomaDatos.vi, que adquiere los datos del archivo binario seleccionado. La función de OrdenarDatos.vi es darle formato numérico a los datos y ordenarlos de tal forma que puedan ser graficados.

La primera parte de este subVI adquiere los datos en formato de cadena de caracteres (desde el subVI **TomaDatos.vi**) y cambia el formato a byte. Luego ordena los bits de cada dato y cambia nuevamente el formato de éstos a enteros sin signo de 16 bits (U16), donde el dato (que es un carácter ASCII de 8 bits) se encuentra en el byte bajo y el byte alto está vacío. De esta forma se crea un arreglo de enteros de 16 bits. A continuación toma el primer número de 16 bits del arreglo, pasa la información al byte alto y coloca el segundo número del arreglo en el byte bajo[36]. Este proceso se repite con todos los datos del arreglo original, creando un arreglo nuevo. A medida que se crea el arreglo, se convierte el dato de 16 bits en el valor de voltaje correspondiente de acuerdo con las características del conversor de la tarjeta de adquisición de datos SAD800-L (conversor de 16 bits, configurado de -10 a 10 voltios, datos con formato complemento a dos[37]) empleando el subVI **ConVolt.vi**[38].

La segunda parte toma el arreglo creado y organiza los datos en cuatro arreglos diferentes, cada uno correspondiente a un canal de entrada del conversor de la tarjeta de adquisición de datos. El primer dato del arreglo corresponde al primer canal, el segundo dato al segundo canal, el tercer dato al tercer canal, el cuarto dato al cuarto canal, el quinto dato al primer canal, y así sucesivamente.

[36] Referirse al aparte 5.5.1 del libro.
[37] Referirse al aparte 4.4.1 del libro.
[38] Referirse al aparte 5.3.4

6.3.4 Convertir los datos al valor de voltaje correspondiente (subVI – ConVolt.vi)

Este subVI toma el valor de entrada, que corresponde a uno de los valores del arreglo generado en el subVI descrito en el aparte anterior, compara su valor con el número 32768 (de acuerdo al formato complemento a dos para números de 16 bits), si es mayor o igual representa a un número negativo y si es menor representa un número positivo. Luego cambia su formato a doble precisión. En el caso de un número negativo, se toma el valor, se le resta 2^{16}, se multiplica por 20 (rango de voltaje de -10 a 10) y se divide entre 2^{16}. Para el caso de un número positivo se multiplica por 20 y se divide entre 2^{16}.

6.3.5 Graficar los datos (subVI – Graficar.vi)

Este subVI toma los 4 vectores que contienen los datos correspondientes a cada uno de los 4 canales y los convierte de formato de arreglo de números a formato de datos dinámicos (este formato es el que emplean los VI's express de LabView[39], y es compatible con cualquier indicador o entrada numérica, de forma de onda -waveform- o booleana). Luego les cambia el formato a forma de onda (*waveform*) para ajustar el delta de tiempo entre las muestras (de acuerdo con el número de muestras y el tiempo de adquisición) empleando el subVI **Build Waveform.vi** (construir forma de onda) y nuevamente se pasa a formato de datos dinámicos. Por último, los cuatro canales se unen en una sola línea o cable (*wire*) empleando el subVI **Merge signals.vi** (unir señales) creando un cable que contiene los datos de los cuatro canales.

[39] Los VI's express de LabView son aquellos que requieren pocas conexiones ya que se configuran empleando un cuadro de diálogo.

6.3.6 Espectro en frecuencia de la señal (subVI express- Spectral Measurements.vi)

Para encontrar el espectro en frecuencia y la fase de la señal se empleó el VI express **Spectral Measurements.vi** configurado para mostrar la magnitud RMS del espectro con escala lineal, y realizando la operación de ventaneo para disminuir el error (efecto de fuga) producido al realizar una transformada rápida de Fourier (FFT) a una señal que no es continua en el tiempo. La ventana empleada para esta operación es la de Hanning. La opción de promediar los datos se encuentra deshabilitada y la fase de la señal se muestra en grados. La entrada del subVI express **Spectral Measurements.vi** es el cable (*wire*) del diagrama de bloques de LabView que contiene los datos de los cuatro canales y la salida corresponde a un cable (*wire*) que contiene el espectro de los cuatro canales.

Para visualizar el Espectro en frecuencia de las señales en la Interfaz SAD800-L se hace clic en "`Visualización`" del menú "`Opciones`" de la Interfaz, y se selecciona la opción "`Espectro en frecuencia`" del menú desplegable del cuadro de visualización que aparece en la Interfaz. Para mostrar el espectro se debe hacer clic en el botón "`Actualizar`".

6.3.7 Opciones de la Interfaz SAD800-L

En el menú "Opciones" de la Interfaz SAD800-L se puede acceder a tres cuadros diferentes que se muestran, uno a la vez, en el panel de control de la Interfaz. Las tres opciones son: "Visualización", "Filtrado" y "Trigger", y están descritas a continuación.

Ítem "`Visualización`" del menú "`Opciones`"

El cuadro de "`Visualización`" está habilitado por defecto al iniciar la aplicación. En caso de que no se encuentre habilitado, se puede seleccionar haciendo clic en

"Visualización" del menú "Opciones" de la Interfaz. Empleando cuatro estructuras de casos (*Case Estructures*) de LabView, uno por canal, se pueden realizar dos procedimientos diferentes, dependiendo del valor de los controles booleanos para seleccionar los canales que se desean visualizar, que se encuentra en el cuadro "Visualización" de la Interfaz SAD800-L.

Cuando se selecciona un canal para visualizar, en la estructura de casos (correspondiente al canal seleccionado) del diagrama de bloques de la aplicación principal de LabView, se ingresa al caso con la etiqueta Verdadero (*True*) y se realiza el procedimiento para mostrar la gráfica que se describe más adelante en este mismo aparte. Si no se desea visualizar el canal, en la estructura de casos del diagrama de bloques de la aplicación principal de LabView se entra al caso Falso (*False*) donde se realiza el procedimiento para ocultar la señal, descrito también más adelante en este aparte.

Para mostrar una señal, se crea un Nodo de propiedad[40] de la gráfica de la Interfaz SAD800-L y se selecciona la señal a la que se le desea cambiar el color empleando la propiedad *ActPlot* y asignándole a la propiedad una constante numérica correspondiente al número de la señal (las señales se enumeran con enteros de cero en adelante dependiendo del orden en que fueron unidas en el cable -*wire*-[41]). Luego se le asigna a la señal un color, seleccionándolo con el control **Caja de color** (*Color box*) que se muestra en el cuadro de "Visualización" y conectando esta caja a la propiedad *Plot.Color* en el mismo nodo de propiedades. Si no se desea visualizar la gráfica, se realiza el mismo procedimiento seleccionando el color Transparente (T) en un constante de **Caja de color**.

[40] Permite leer o cambiar las propiedades de un objeto (control o indicador) del diagrama de bloques de LabView. Para crealo se hace clic con el botón derecho sobre el objeto, y se selecciona la opción *Property Node* del menú *Create*.
[41] Ver aparte 5.3.5 del libro.

Ítem "`Filtrado`" del menú "`Opciones`" (subVI – Filtros.vi)

 El cuadro "`Filtrado`" se muestra en la Interfaz SAD800-L cuando se selecciona esta opción del menú principal. Este ítem permite escoger entre tres tipos de filtros diferentes: pasabajas, pasabanda y pasaaltas, o sin filtro. La selección se realiza en el panel frontal de la aplicación principal de LabView empleando una lista desplegable que contiene las cuatro opciones internamente enumeradas 0, 1, 2 y 3 respectivamente. Una vez seleccionado el filtro en el panel frontal de la aplicación, se habilita la lista desplegable que permite escoger la topología del filtro entre: Butterworth, Chebyshev y Bessel, internamente enumerados 0, 1 y 2 respectivamente. El número del filtro (pasabajas, pasabanda, pasaaltas y no filtro) seleccionado ingresa a una estructura de casos de LabView (*Case Estructure*) donde se ejecuta un proceso diferente de acuerdo a la opción seleccionada. Cada caso (excepto el de no filtrado) contiene otra estructura de casos con la cual se selecciona la topología del filtro dependiendo de la opción seleccionada en la lista desplegable (Butterworth, Chebyshev y Bessel) del panel frontal de la aplicación. Dependiendo de las dos opciones seleccionadas (tipo y topología del filtro) el cable (*wire*) que contiene los datos de los cuatro canales entra (o no, si se selecciona no filtrar la señal) al VI express de LabView **Filter.vi** que se encuentra configurado dependiendo del caso de la estructura en el que se halla, o lo que es lo mismo, de la selección del filtro.

Ítem "`Trigger`" del menú "`Opciones`" (subVI – trigger.vi)

 El cuadro "`Trigger`" se muestra en la Interfaz SAD800-L cuando se selecciona esta opción del menú principal. En el mismo cuadro se puede habilitar o deshabilitar esta opción, haciendo clic en el botón de encendido o apagado (*ON-OFF*). Cuando se habilita, se enciende el led que se encuentra en el cuadro de "`Trigger`" en la Interfaz, y cuando se deshabilita, se apaga. Si se desea emplear el trigger, se debe configurar en el panel frontal de la aplicación principal de LabView

el "Nivel" de voltaje con el que se desea disparar el trigger, la "Histéresis" (variación de voltaje de la señal por encima o por debajo del nivel que soporta el trigger) y el "Flanco" (de subida – rising, o de bajada - falling) de la señal, de tal forma que se ajuste a la señal que se desea capturar. También se debe escoger cuál es la señal que debe disparar el trigger. Esta selección también se realiza en el cuadro de "Trigger", y se puede escoger una señal o varias, dependiendo de las necesidades. Cuando se seleccionan dos o más señales para disparar el trigger, éste se dispara con la primera señal que alcance las condiciones necesarias para su disparo.

El subVI **trigger.vi** se ejecuta de la siguiente manera: el cable (*wire*) que contiene las señales de los cuatro canales es separado en cuatro cables, cada uno correspondiente a un canal, los cuales son pasados a formato de forma de onda (*waveform*). Dependiendo de la selección de las señales que disparan el trigger que se realice en el panel frontal de la aplicación principal de LabView, el cable de cada canal seleccionado entra al subVI llamado **Trigger.vi** de LabView (si no está seleccionado, no es procesado y el valor de indexado que se empleará más adelante es cero), que de acuerdo al nivel, a la histéresis y al flanco, dispara o no el trigger.

Si el subVI **Trigger.vi** de LabView encuentra en la señal las condiciones configuradas en el subVI (nivel, histéresis y flanco), ajusta a verdadero una variable booleana de salida, y en otra variable de salida coloca el indexado del valor donde encontró las condiciones. Esto ocurre con todos los canales que se han seleccionado para disparar el trigger. Luego, empleando el subVI **comparar.vi** se selecciona el menor valor de indexado (diferente de cero) encontrado por el trigger como el valor inicial de las cuatro señales, y si en alguna de las señales seleccionadas para disparar el trigger se encontró un valor que corresponda con la configuración determinada para disparar el trigger, o en otras palabras, la variable booleana de salida es verdadera, se recortan las señales de los cuatro canales con el subVI Get Waverform **Subset.vi** empleando el menor indexado encontrado anteriormente como valor inicial

de indexado de las señales de los cuatro canales. En caso de que no se encuentren las condiciones de nivel, histéresis y flanco en las señales designadas para disparar el trigger, no se despliega ninguna señal.

Para encontrar el menor valor diferente de cero de un arreglo se empleó el subVI **comparar.vi** descrito a continuación.

 Se toma el arreglo con los 4 valores de indexado encontrados en el subVI **trigger.vi** y se coloca como entrada a una estructura *for* con N igual a 4 (realiza cuatro iteraciones). En cada una de las iteraciones del *for*, se compara si el número correspondiente a la iteración (si es la primera iteración corresponde al primer valor del arreglo, y así sucesivamente) es igual a cero. Si es diferente de cero, compara el valor de la iteración actual con el valor obtenido en la iteración anterior y coloca el menor valor como variable de entrada para comparar en la siguiente iteración. En la primera iteración, debido a que no existe un valor con el cual comparar el primer valor del arreglo, se compara con la constante infinito. Al final de la cuarta iteración se debe obtener el menor número diferente de cero contenido en el arreglo de cuatro valores.

6.3.8 Menú principal de la Interfaz SAD800-L

Para la aplicación principal de LabView se creó un menú empleando la herramienta "*Run-Time Menu*" (Menú en tiempo de ejecución) que se encuentra en la opción "Edit" del menú principal de LabView. El menú creado con la herramienta *Run-Time Menu* sólo se encuentra disponible cuando se ejecuta la aplicación. La distribución del menú principal de la Interfaz SAD800-L es la que se muestra en la Figura 43.

Figura 43. Menú principal de la Interfaz SAD800-L

>Archivo
  >Adquirir Datos
  >Graficar Datos
  >Cargar gráfica
  >Guardar gráfica
  >Salir
>Opciones
  >Visualización
  >Filtrado
  >Trigger
>Reporte
  >Imprimir
  >HTML
>Ayuda
  >Mostrar ayuda
  >Acerca de

Inicialmente, al correr la aplicación principal de LabView, los ítems "`Guardar`" y "`Reporte`" del menú se encuentran deshabilitadas. Una vez se haya desplegado una gráfica en la aplicación, ya sea graficando los datos desde un archivo binario, o cargando una gráfica guardada anteriormente, estos ítems se habilitan. Para habilitar y deshabilitar las opciones del menú principal de la aplicación se empleó el subVI **Set Menu Item Info.vi**, colocando en el diagrama de bloques de la aplicación principal de LabView la constante verdadero o falso en la entrada *Enable* del subVI respectivamente, y en la entrada *Menubar* se cablea el número de referencia del menú al que se le desean cambiar las características. El número de referencia del menú se obtiene empleando el subVI **Current VI's Menubar.vi**, el cual tiene como salida el número de referencia del menú principal de la aplicación de LabView que se está ejecutando actualmente.

Para identificar en cuál de los ítems que contiene el menú se está haciendo clic se emplea el subVI **Get Menu Selection.vi** el cual tiene como entrada el número de referencia del menú de la aplicación principal de LabView (obtenido con el subVI **Current VI's Menubar.vi**) y como salida el mismo número de referencia del menú y la etiqueta (en formato de cadena de caracteres) o *Tag* del ítem del menú seleccionado. Algunos de estos ítems son manejados por el subVI **ManejoMenú.vi.** Tanto la etiqueta, como el cable (*wire*) que contiene las cuatro señales en el dominio

del tiempo y el cable que contiene su correspondiente espectro en frecuencia, entran al subVI **ManejoMenú.vi** donde son procesadas, dependiendo de la selección del menú. Este proceso se explica en el siguiente aparte.

6.3.9 Manejo de ítems del menú principal de la Interfaz SAD800-L (subVI - ManejoMenú.vi)

El proceso que la aplicación debe ejecutar cuando se selecciona un ítem del menú principal se escoge con la etiqueta del ítem mediante una estructura de casos de LabView. Si no se selecciona ningún ítem, la etiqueta no concuerda con ninguno de los casos programados en la estructura de casos de LabView, y se ejecuta el caso por defecto en el cual no se realiza ningún proceso. Los demás casos son descritos a continuación:

Ítem "`Cargar gráfica`" del menú "`Archivo`"

Si se selecciona el ítem "`Cargar gráfica`" del menú "`Archivo`" de la aplicación para la tarjeta de adquisición de datos por bus USB, en el subVI **ManejoMenú.vi** se entra al caso con la etiqueta "`Cargar`". En este caso ejecuta el subVI llamado **Leer.vi** el cual corresponde al subVI **ex_subFileRead.vi** tomado del VI express **Read LVM.vi** con la siguiente modificación en el diagrama de bloques:

- El subVI llamado **FirstCall.vi** contenido en **ex_subFileRead.vi** fue reemplazado por la constante booleana verdadero, para permitir que el subVI **Leer.vi** pueda cargar cuantas gráficas sea necesario sin importar si ya ha sido llamado durante la ejecución de la aplicación principal.

Para que la aplicación principal de LabView reconozca si la operación de cargar una gráfica fue exitosa, en el indicador de error de salida (*error out*) de **Leer.vi,** se

adicionó la herramienta *unbundle* para separarlo en sus variables componentes, y se toma la variable booleana que indica existe error durante dicho proceso.

El panel frontal de la aplicación **Leer.vi** tiene las siguientes modificaciones con respecto al panel frontal del subVI **ex_subFileRead.vi**:

- El botón "1st", "*File dialog?*" (cuadro de diálogo de archivos), "*relative time?*" (tiempo relativo) y "*Attemp to Read Generic Text Files*" (Intentar leer archivos de texto genéricos), se encuentran habilitados.
- El delimitador es un caracter ",".

Ítem "Guardar gráfica" del menú "Archivo"

Si se selecciona el ítem "Guardar gráfica" del menú "Archivo" de la Interfaz SAD800-L, en el subVI **ManejoMenú.vi** se entra al caso con la etiqueta "Guardar". En este caso se ejecuta el subVI llamado **guardar.vi** el cual corresponde al subVI **ex_subFileWrite.vi** tomado del VI express **Write LVM.vi** con las siguientes modificaciones en el panel frontal:

- Los botones de "*Reset*", "*File dialog?*" (Cuadro de diálogo de archivo) y "*Ask each iteration*" (Preguntar en cada iteración) están siempre habilitados.
- En la lista "*Column headings*" (Encabezados de columna) se encuentra seleccionada la opción "*One per*" (Uno por columna).

El diagrama de bloques del subVI **guardar.vi** no tiene ningún cambio con respecto al diagrama de bloques del subVI **ex_subFileWrite.vi.**

Ítem "Salir" del menú "Archivo"

Al seleccionar el ítem "Salir" del menú "Archivo" de la Interfaz SAD800-L, en el subVI **ManejoMenú.vi** se entra al caso etiquetado "Salir", el cual contiene la herramienta **Salir de LabView** (*Quit LabView*) que detiene la aplicación y cierra LabView.

Ítems "Imprimir" y "HTML" del menú "Reporte"

Al seleccionar el ítem "Imprimir" ó "HTML" del menú "Reporte" de la Interfaz SAD800-L, se entra al caso etiquetado "Imprimir" o "HTML" del subVI **ManejoMenú.vi**, respectivamente. En cualquiera de los dos casos se ejecuta el subVI **GenReporte.vi**, el cual tiene como entradas la gráfica de los datos en el dominio del tiempo, la gráfica del espectro de las señales y una constante numérica que permite seleccionar si el reporte se debe imprimir (0) o guardar en formato html (1).

Al ejecutarse el subVI **GenReporte.vi** se abre su panel frontal, mostrado en la Figura 44 permitiendo al usuario ingresar los datos que aparecerán en el reporte, como: título del reporte, nombre del autor, nombre de la compañía, comentarios. También permite seleccionar si se desea incluir en el reporte la gráfica de la señal en dominio del tiempo, su espectro en frecuencia, y/o la tabla de datos correspondiente a cada una de las dos gráficas. Si se selecciona el ítem "Imprimir" del menú "Reporte" de la Interfaz, la opción para seleccionar la ruta donde se desea guardar el reporte aparece deshabilitada. La opción para seleccionar la ruta sólo se habilita cuando se hace clic en el ítem "HTML", lo que permite elegir el nombre y la ruta del reporte. Para habilitar y deshabilitar el control que permite seleccionar el nombre y la ruta del reporte, se creó en el diagrama de bloques del subVI **GenReporte.vi** un nodo de

propiedad del control, asignando a la propiedad "Deshabilitar" (*Disable*) los valores numéricos 0 y 2 respectivamente.

Figura 44. Panel frontal del subVI GenReporte.vi, en el cual se ingresan los datos requeridos en el reporte. El panel que se muestra en la figura corresponde al que se muestra seleccionando el ítem Imprimir del menú Reporte. El panel que se muestra cuando se selecciona el ítem HTML del menú Reporte difiere en que la herramienta para escoger la ruta se encuentra habilitada.

Todos los datos que se deben entrar en el subVI **GenReporte.vi**, incluyendo la variable de identificación del tipo de reporte (Imprimir o HTML), pasan a otro subVI llamado **Reporte.vi** que corresponde al VI express **Report.vi**, con las siguientes modificaciones en el diagrama de bloques:

- Todas las constantes del VI express **Report.vi** correspondientes a los datos entrados en el subVI **GenReporte.vi** fueron reemplazados por controles.
- La variable "*Enable*" (Habilitar) fue reemplazada por la constante booleana verdadero.
- Se añadió un control para la variable de tipo de reporte que permite identificar si el reporte se debe imprimir (0) o guardar como archivo html (1).

Ítem "Mostrar ayuda" del menú "Ayuda"

Si se selecciona el ítem "Mostrar ayuda" del menú "Ayuda" de la Interfaz SAD800-L, en el subVI **ManejoMenú.vi** se entra al caso con la etiqueta "Mostrar". En este caso se encuentra el subVI **Control Help Window.vi** con la variable booleana verdadero en su entrada "*show*" permitiendo mostrar la ventana de ayuda de LabView (*Context Help*).

Ítem "Acerca de..." del menú Ayuda

Seleccionando el ítem "Acerca de..." del menú "Ayuda" de la Interfaz SAD800-L, se entra al caso con la etiqueta "Acerca de", en el subVI **ManejoMenú.vi.** En este caso se encuentra el subVI **Acerca.vi**, el cual, al ejecutarse muestra una ventana con información sobre los diseñadores de la aplicación.

LOGROS

- Se construyó una tarjeta de adquisición de datos por bus USB v1.1 con cuatro entradas de tensión analógicas entre ?10V empleando una frecuencia de muestreo de 10kHz por canal, transmitidos al PC a una tasa de 6400bps en tiempo real.

- El sistema de adquisición de datos cuenta con las medidas de protección requeridas para su buen funcionamiento: aislamiento capacitivo, fusibles y resistencias de protección.

- El sistema implementado utiliza el microcontrolador MC68HC908JB8JP de Motorola para manejar y emplear el bus USB v1.1 a la máxima tasa de transferencia efectiva de datos por punto final (endpoint) que corresponde a 800 bytes por segundo.

- Se diseñó un sistema de temporización externa al conversor analógico/digital ADS7825P para trabajar en modo continuo garantizando una frecuencia de muestreo constante para los cuatro canales de entrada. De esta manera, a pesar que el bus USB v1.1 es asíncrono, el sistema se comporta de una forma cuasi-síncrona, aprovechando la condición de sobremuestreo y la independencia del sistema de temporización.

- Se construyó un dispositivo USB de baja velocidad a partir del módulo embebido en el microcontrolador MC68HC908JB8, que trabaja con el estándar 1.1 y cumple con todos los requisitos estipulados por la organización oficial a nivel mundial USB.ORG.

- Se diseñó un dispositivo versátil que además de contar con las ventajas del bus USB, permite al usuario cambiar la frecuencia de muestreo del sistema de

adquisición sin necesidad de cambiar o añadir físicamente otro elemento más al hardware.

- Se entendió completamente el funcionamiento del protocolo USB V1.1 que está embebido a nivel hardware en el microcontrolador MC68HC908JB8JP, el cual se programó para garantizar la máxima transferencia de datos efectiva en la versión 1.1 (800 bytes por segundo).

- Se diseñó la Interfaz gráfica SAD800-L empleando Visual Basic 6.0 y LabVIEW 7.0, para adquirir y graficar los datos provenientes del SAD800-L, respectivamente.

- Se implementaron opciones adicionales para la Interfaz SAD800-L que permiten dar un tratamiento adicional a las señales, generar reportes y almacenar gráficas, con el fin brindar mayor comodidad al usuario final de la Interfaz al momento de manipular los datos adquiridos con el SAD800-L.

- Se desarrolló un sistema de adquisición para una aplicación lenta (f<50Hz) en un prototipo de monitoreo de motores diesel de la Escuela de Ingeniería Mecánica empleando la interfaz de comunicación USB V1.1.

- Se hizo parte de un grupo de investigación para realizar un proyecto interdisciplinario en el que se involucra ingeniería mecánica, diseño industrial e ingeniería electrónica.

CONCLUSIONES Y OBSERVACIONES

- Debido a la pérdida de la base de tiempo de los datos adquiridos con la tarjeta SAD800-L, producida por el almacenamiento de los datos en un archivo binario, se requiere la tasa de transferencia de los datos para visualizarlos correctamente. Las transferencias por interrupción, a pesar de ser periódicas, no aseguran una tasa de transferencia de datos constante para todas las adquisiciones; por esta razón, fue necesario hacer una aproximación de ésta a partir del número de bytes de datos transmitidos y el tiempo de adquisición. Con esta aproximación se obtuvo resultados satisfactorios al momento de graficar los datos, como se muestra en el capítulo 5.

- Aunque los dispositivos de baja velocidad, bajo el estándar USB V1.1, tienen una tasa de transferencia de 1.5Mbps, no transmiten datos a esta velocidad. Para un sistema de adquisición de datos la máxima tasa efectiva teórica es de 800 bytes por segundo, ya que el protocolo permite enviar máximo 8 bytes de datos cada 10 ms, acompañados de información adicional como estatus, control y chequeo de errores. Por esta razón, la versión USB empleada en este proyecto es muy lenta comparada con otras interfaces de comunicación actual (serial, paralelo, entre otros).

- El sistema de adquisición realiza transferencias USB tipo interrupción; esto quiere decir que las peticiones entre el manejador (*host*) y el Sad800-L deben respetar un tiempo de latencia mínimo de 10ms para el envío de datos; razón por la cual la implementación de transferencia isócrona no es posible en esta versión.

- La temporización de un dispositivo USB requiere del uso de un reloj (*timer*) que garantiza los períodos de latencia y peticiones entre el manejador (*host*) y el

dispositivo, permitiendo realizar transferencias en los tiempos estipulados en el estándar USB, que para el caso de la versión 1.1 corresponde a un período de interrupción de 1 ms.

- En transferencias de tipo interrupción, el tiempo de latencia para dispositivos USB 1.1 de baja velocidad varía entre 10 y 255 ms debido a que el manejador (*host*) es libre de transferir datos tan rápido como sea la frecuencia de envío de peticiones. Es decir, las transferencias por interrupción no garantizan una rata precisa de entrega, excepto cuando se tiene la mínima latencia posible, que en este caso es de 10ms, tal como está configurado el sistema.

- El envío de datos a través del bus USB V 1.1 empleando el canal de control (*Control Pipe*) permite disminuir en 10 veces el tiempo de envío entre paquetes de datos (1ms entre paquetes de 8 bytes), pero la tasa de transferencia no es constante. Además no es la forma más eficiente de enviar datos ya que cada transferencia de control con fase de datos requiere un encabezado de 63 bytes (se requieren mínimo 3 transacciones para completar una transferencia de control con datos), mientras las transferencias de interrupción requieren un encabezado de 19 bytes. Es decir, en el diseño de un sistema de adquisición no es adecuado emplear el canal de control para el envío de un flujo constante de datos.

- La implementación de dos canales de comunicación USB de entrada (*Pipe In*) puede aumentar la tasa de transferencia (aprox. 1600 bytes/s), tan solo si el hardware soporta una configuración con dos interfaces activos al mismo tiempo que permita al *host* enviar peticiones de a cada *endpoint* en una misma trama y además debe asegurar que el controlador utilizado soporte esta configuración.

- En el sistema de adquisición USB V1.1 la implementación de dos *endpoints* de entrada duplica la cantidad de datos transmitidos por segundo, pero estos no son transmitidos a una frecuencia uniforme, por lo cual la máxima tasa de

transferencia posible para una aplicación de adquisición de datos por bus USB 1.1 corresponde a 800 bytes/s.

- El manejo de errores en el Sad 800-L se realiza de acuerdo a lo estipulado en el protocolo USB que define en la estructura de los paquetes de datos el envío del código de redundancia cíclica que a nivel de software permite al manejador (*host*) reconocer la validez de los datos transferidos por el dispositivo.

- No es posible controlar en qué orden ni en qué momento se envían los paquetes por el bus USB, programando aplicaciones de alto nivel como lo es la Interfaz USBIO COM del *driver* USBIO. Si se desea tener un mayor control sobre el orden y temporización de los paquetes, se debe realizar la programación de éstos en un nivel inferior.

- El Sad800-L trabaja con un conversor analógico/digital de 16 bits de los cuales se logra una resolución efectiva de 13 bits, debido a que el paso de conversión es del orden de la décima parte del ruido presente en el sistema.

- En resumen, para aplicaciones que no requieran un flujo alto de información y sean de tipo interfaz-humano la versión 1.1 del estándar USB es apropiada, sin embargo, para sistemas de adquisición de datos no lo es, debido principalmente a que es un bus asíncrono, y la tasa de transferencia de datos máxima para baja velocidad es de 800 bytes/s.

RECOMENDACIONES PARA FUTUROS PROYECTOS

- Con el fin de incrementar la velocidad del sistema de adquisición se recomienda migrar a las versiones 1.1 Full Speed y 2.0 High Speed del estándar USB.

- Diseñar un controlador USB que permita tener un mayor control sobre el orden y temporización de los paquetes que son transmitidos por el bus, y que permita el empleo de todas las funciones

- Es necesario seguir explorando todo lo que concierne al desarrollo de dispositivos que se manejan a través del bus USB en otras áreas como dispositivos interfaz-humano; ya que interfaces como la serial y la paralela han ido desapareciendo.

- Diseñar dispositivos de clase fabricante, que emplean transferencias de control con fase de datos (para el caso del controlador USBIO corresponden a las funciones *Vendor In Request* y *Vendor Out Request*), para transmitir información. Estas peticiones deben ser programadas en la aplicación de alto nivel, al igual que el dispositivo. En particular, el controlador USB empleado debe estar programado para responder a este tipo de peticiones.

- Para lograr una visualización de los datos en línea, se recomienda emplear LabVIEW 7.1 y el *driver* VISA 3.1. La versión 7.0 de LabVIEW tiene un error en el VI "`Get USB Interrupt Data`" por lo cual no es posible adquirir los datos almacenados en el buffer de recepción del *host*. Este error fue corregido en la versión 7.1 de LabVIEW.

BIBLIOGRAFÍA

[1] AXELSON, Jan. USB Complete. 2ed. Madison,USA: Lakeview Research, 2001.

[2] COMPAQ COMPUTER CORPORATION, Intel Corporation, Microsoft Corporation y NEC Corporation. Universal Serial Bus Specification, Rev 1.1. Septiembre 23, 1998.

[3] HYDE, John. USB Designed by example. Intel, 2ed. Capítulos 1, 2, 3 y 4.

[4] MOTOROLA. Technical Data – MC68HC908JB8, Rev 2.

[5] --------. USB evaluation board using the MC68HC908JB8, Designer Reference Manual. 2001.

[6] NATIONAL INSTRUMENTS. LabVIEW manual de usuario. http://www.ni.com

?

ANEXO A

A.1 GENERALIDADES DEL *DRIVER* USBIO V1.51

A.1.1 INSTALACIÓN

Al instalar el paquete usbio_el.exe, se crea la carpeta **Thesycon** en el directorio **Archivos de programa**. **Thesycon** contiene una serie de carpetas y subcarpetas, entre ellas **usbio,** donde se encuentran el *driver* usbio_el.sys y el archivo usbio_el.inf. El archivo *.inf* es un archivo en formato ASCII que contiene la información sobre la instalación del *driver* y puede ser modificado en cualquier editor de texto. Su contenido y sintaxis están documentados en el Kit de desarrollo para drivers de Microsoft Windows (*Microsoft Window Driver Development Kit - DDK*).

El archivo *.inf* contenido en el paquete es genérico, y es necesario realizar los cambios enunciados a continuación para que se ajuste al dispositivo que se desea controlar: se deben eliminar las partes del código que corresponden a otro sistema operativo (estas partes están indicadas dentro del archivo), se deben asignar los nombres del fabricante, del dispositivo y del controlador en las propiedades adecuadas y por último, se debe cambiar el número identificador del fabricante y del dispositivo dependiendo de los valores que se haya programado en el microcontrolador. Para instalar el archivo *.inf* en Windows XP (sistema operativo en el que se desarrolló la Interfaz SAD800-L) se debe hacer clic con el botón derecho del ratón al archivo, y seleccionar la opción “instalar”. Esto permite que el sistema reconozca la existencia del *driver* USBIO de Thesycon.

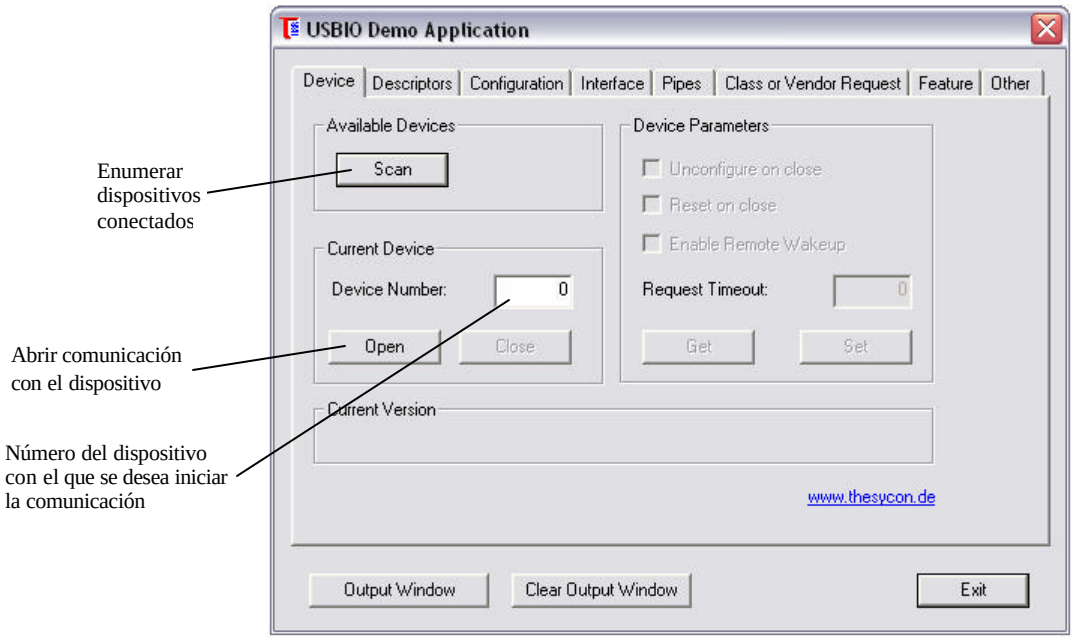
Con este paquete también se instala una aplicación ejecutable para controlar un dispositivo USB (USBIO APPLICATION implementada en Visual C++) con el driver usbio_el.sys, y una carpeta con ejemplos de aplicación que emplean el *driver*

usbio_el.sys para implementar aplicaciones que controlen el dispositivo USB con VisualBasic y Delphi.

A.1.2 APLICACIÓN USBIO (USBIO *APPLICATION*)

La aplicación consta de una serie de pestañas con funciones específicas. En la Figura 1 se puede apreciar la interfaz gráfica de esta aplicación.

Figura 1. Interfaz gráfica de la aplicación ejecutable de USBIO.



Fuente: los autores

Al iniciar la aplicación USBIO, se debe realizar una búsqueda de todos los dispositivos USB que emplean el *driver* USBIO conectados, empleando el botón de búsqueda (*Scan*) para enumerarlos y verificar su disponibilidad. El siguiente paso es escoger el dispositivo de acuerdo al número asignado por el *driver* USBIO durante la enumeración a los dispositivos USB conectados; para un solo dispositivo el número es cero, y se encuentra por defecto en el cuadro de texto de número del dispositivo (*Device Number*). Para varios dispositivos se asignan números del cero en adelante,

en orden ascendente. Para saber qué número corresponde a cada dispositivo, es necesario abrir una comunicación un dispositivo colocando su número en el cuadro indicado[42], y verificar sus descriptores. Se abre una conexión con el dispositivo seleccionado haciendo clic en Abrir (*Open*).

La información de la enumeración y de las demás operaciones puede observarse en una ventana auxiliar de salida (*Output Window*). Esta ventana se observa en la Figura 2.

Figura 2. Ventana auxiliar de salida de la aplicación ejecutable de USBIO

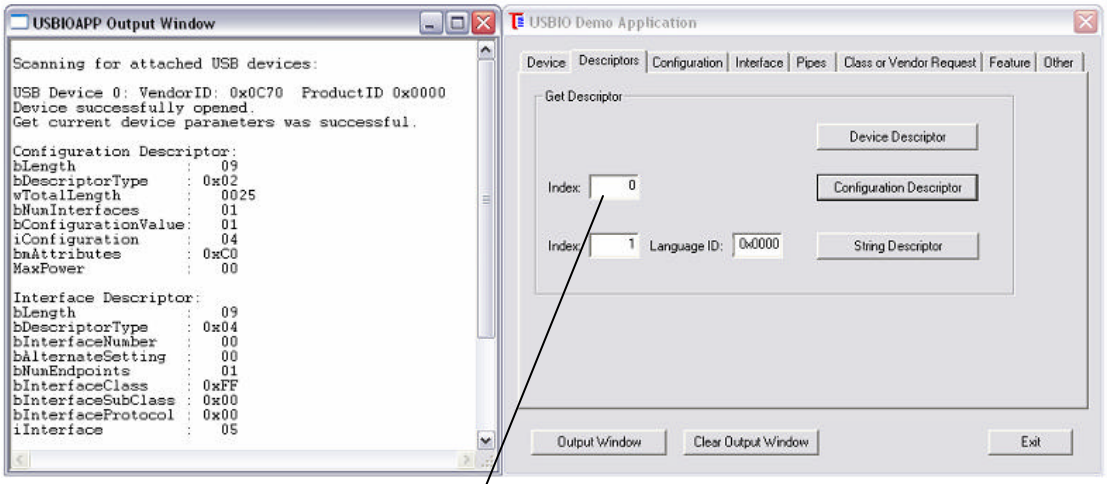
Fuente: los autores.

Una vez abierta la conexión, es posible solicitar los descriptores del dispositivo, de su configuración o su información de fábrica (identificadores del producto y del fabricante), haciendo clic en la pestaña Descriptores (*Descriptors*). Esta información también se visualiza en una ventana auxiliar de salida como se observa en la Figura 3.

El siguiente paso es configurar el dispositivo: en la ventana Configuración (*Configuration*) de la aplicación se hace clic en el botón Configurar (*Set Configuration*). La ventana de configuración y el resultado de esta operación se pueden observar en la Figura 4.

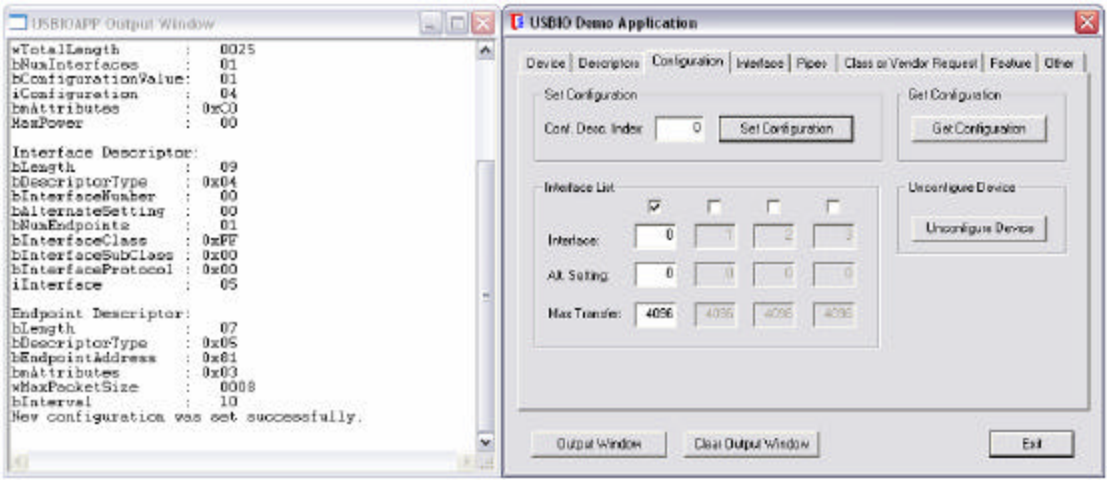
[42] Ver Figura 1 de este Anexo.

Figura 3. Pestaña de los descriptores de la aplicación ejecutable de USBIO.



Fuente: los autores.

Figura 4. Pestaña de configuración de la aplicación ejecutable de USBIO.

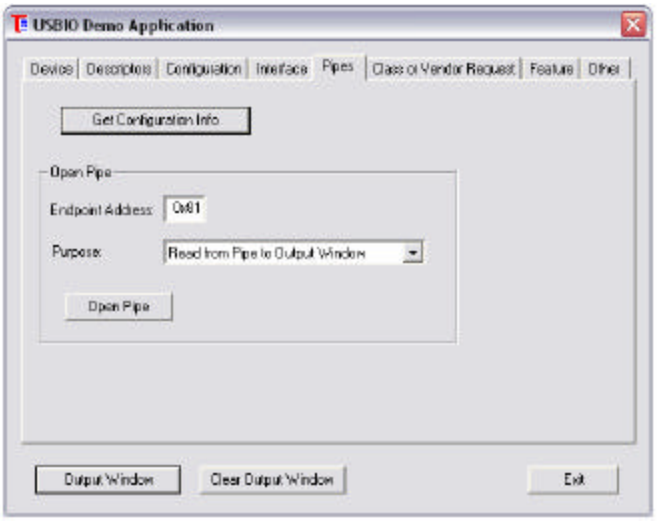


Fuente: los autores.

En este momento el dispositivo debe estar configurado para transmitir datos, pero antes de lograr una comunicación es necesario habilitar un canal, o en otras palabras, abrir un *Pipe*. En la ventana Canales (*Pipes*) aparece por defecto abrir un canal con el *endpoint* 0x81 y la opción de desplegar los datos recibidos en la ventana de salida. Estos valores son adecuados debido a que el *enpoint* empleado en el microcontrolador MC68HC908JB8JP de la tarjeta SAD800-L para enviar los datos tiene también la

dirección 0x81 (dirección por defecto para el primer *endpoint* de entrada). Esta ventana se observa en la Figura 5.

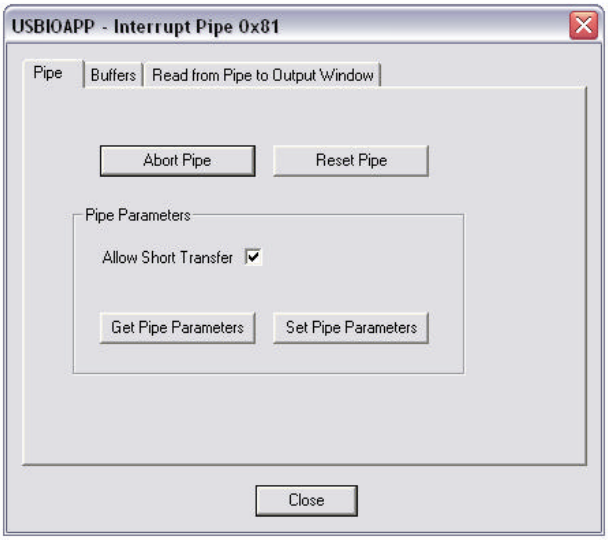
Figura 5. Ventana de configuración de los Pipes de la aplicación ejecutable de USBIO



Fuente: los autores.

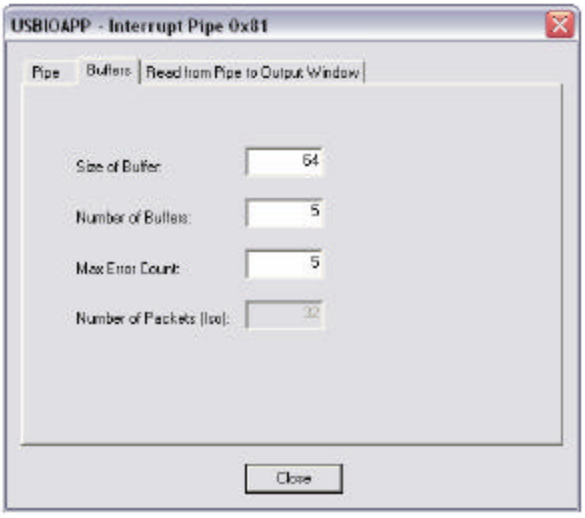
Después de hacer clic en el botón Abrir Canal (*Open Pipe*) se abre la ventana que se observa en la Figura 6 para controlar el canal. Esta ventana contiene 3 pestañas: la primera es Canal (*Pipe*) y contiene los métodos para abortar y reiniciar el canal, así como los de obtener y ajustar sus parámetros; la segunda es *Buffers* (Figura 7) y contiene los parámetros del buffer de llegada de los datos que se encuentra en el computador personal (tamaño en bytes del *buffer*, número de *buffers* y el máximo conteo de errores permitido); y por último se encuentra la ventana de aplicación, que contiene los botones para iniciar y detener la lectura de los datos (ver Figura 8).

Figura 6. Ventana de aplicación para el canal.



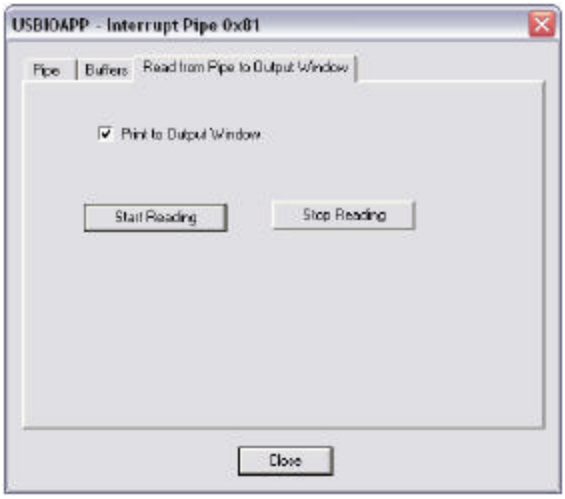
Fuente: los autores.

Figura 7. Pestaña de configuración de los buffers del *PC* correspondientes al canal de lectura de datos abierto (sus valores por defecto se pueden emplear para la tarjeta de adquisición de datos de este proyecto).



Fuente: los autores.

Figura 8. Pestaña de aplicación para el canal de lectura de datos abierto.



Fuente: los autores.

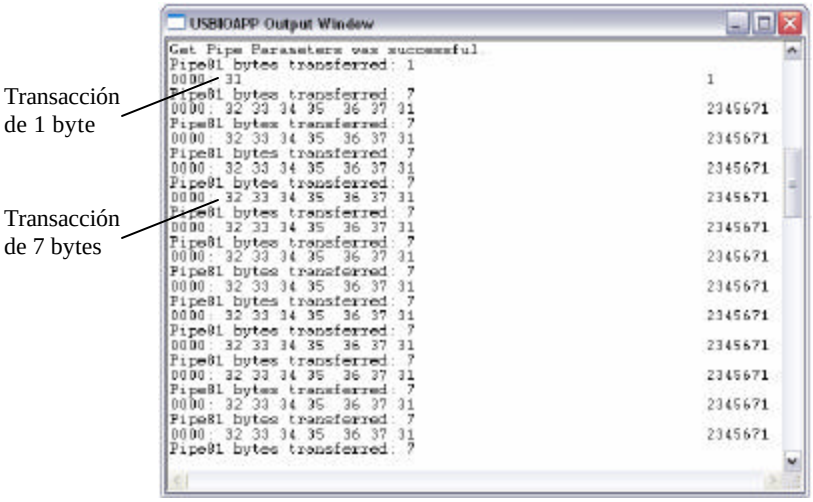
Al hacer clic en el botón Iniciar Lectura (*Start Reading*) se obtiene la lectura de los datos enviados por dispositivo USB conectado. En la Figura 9 se muestra un ejemplo de los datos recibidos por la aplicación cuando se conecta la tarjeta de desarrollo para el microcontrolador MC68HC908JB8JP[43] diseñada en este proyecto, programada con un *firmware* para enviar el código ASCII de los números del 1 al 7 secuencialmente.

Las demás pestañas de la aplicación tienen otras funciones adicionales. La pestaña Interface (*Interface*) se emplea para dispositivos con varias interfaces o modos de operación, para hacer la transición de una a otra. Esta opción no se emplea en la tarjeta de SAD800-L debido a que sólo se requiere un modo de operación (transmitir datos empleando transferencias por interrupción[44]). La pestaña Petición de clase o fabricante (*Class or Vendor Request*) se emplea para realizar peticiones estándar (definidas en las especificaciones USB 1.1) al dispositivo USB, o peticiones definidas por el fabricante. Algunas de estas peticiones están restringidas debido a que la versión del *driver* con que se cuenta es una versión de evaluación (*light*). La pestaña Parámetros (*Feature*) es útil para cambiar parámetros particulares del dispositivo y del endpoint (estos parámetros están definidos en las especificaciones USB 1.1 y

[43] Remitirse al capítulo 2 del libro.
[44] Remitirse a la sección de Tipos de transferencias del aparte 1.3.1 del libro.

corresponden a parámetros de manejo de potencia y respuesta del *endpoint*). La pestaña *"Other"* contiene las opciones de estado ya sea del dispositivo, de la interfaz o del enpoint, y las opciones para configurar el manejo de energía del dispositivo, además del conteo de las tramas recibidas.

Figura 9. Datos recibidos por la aplicación ejecutable de USBIO, cuando se habilita la comunicación con la tarjeta de desarrollo para el microcontrolador MC68HC908JB8JP diseñada en este proyecto.



A.2 GENERALIDADES DEL *DRIVER* VISA 3.1

La versión empleada del *driver* VISA contiene dos herramientas poderosas: el *VISA Driver Development Wizard* y el *VISA Interactive Control* (VISA *IC*), que se encuentran en la carpeta VISA, del directorio de National Instruments en el menú "`Todos los programas`" de "`Inicio`" en Window XP.

El *VISA Driver Development Wizard*, es la herramienta que permite crear el archivo *.inf* que contiene la identificación del fabricante y del producto escrita en el

dispositivo USB, para que el sistema operativo lo reconozca y el *driver* VISA lo pueda controlar[45].

La herramienta VISA *IC* se describe en el siguiente aparte de este anexo.

A.2.1 VISA Control Interactivo (VISAIC)

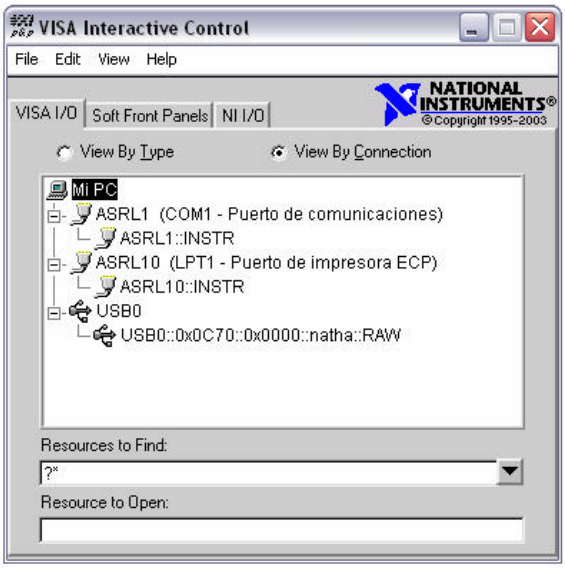
Esta herramienta permite controlar de forma interactiva los dispositivos manejados por el *driver* VISA, empleando el lenguaje de LabView. La herramienta de control interactivo de VISA (VISAIC) se encuentra en la misma ruta que la herramienta *VISA Driver Development Wizard*.

En la ventana principal de la herramienta se pueden observar los dispositivos conectados y disponibles que son manejados por el *driver* VISA como se observa en la Figura 10.

Al hacer doble clic sobre el nombre de un dispositivo se abre una sesión de comunicación con éste, habilitando una ventana nueva que contiene diferentes opciones dependiendo del tipo de dispositivo con el que se pretende iniciar la comunicación. En el caso de un dispositivo USB, al iniciar una sesión con un dispositivo, la herramienta VISAIC de National Instrument configura automáticamente el dispositivo con las opciones por defecto, dejándolo listo para iniciar la comunicación, y a su vez abre una ventana con tres pestañas principales: "*Template*", "*Basic I/O*" e "*Interface I/O*", como se muestra en la Figura 11. Cada una de estas pestañas contiene una serie de pestañas adicionales con funciones específicas; a continuación se describen aquellas que se refieren a un dispositivo USB de baja velocidad.

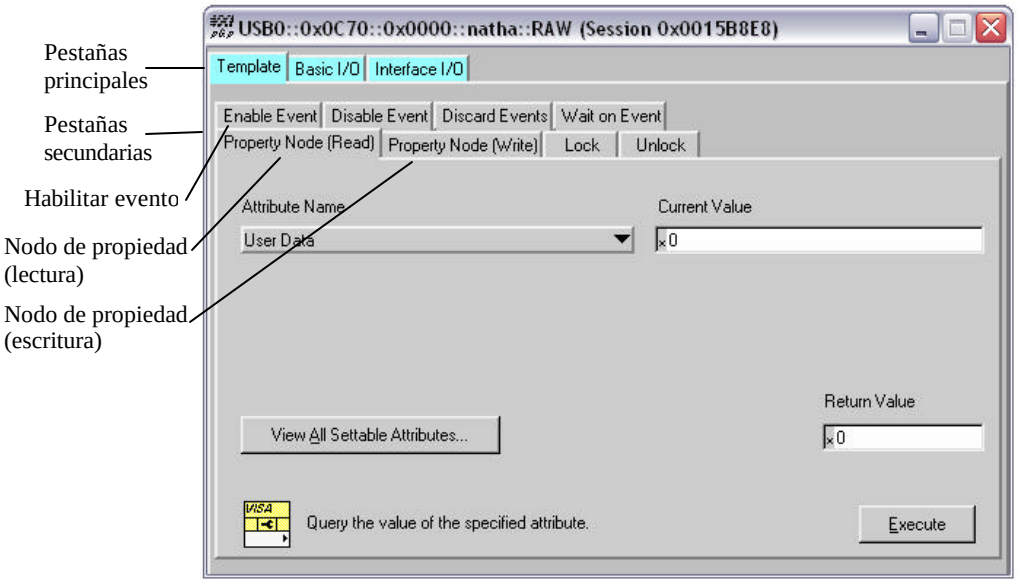
[45] Para mayor información sobre cómo crear e instalar el archivo *.inf*, visitar la página de National Instrument www.ni.com.

Figura 10. Interfaz de la aplicación VISAIC.



Fuente: los autores.

Figura 11. Ventana de una sesión de comunicación abierta con un dispositivo USB.



Fuente: los autores.

o La Pestaña Plantilla (*Template*) se muestra en la Figura 11 y tiene cuatro funciones importantes: "`Nodo de propiedad (lectura)`" y "`Nodo de propiedad (escritura)`", "`Habilitar evento`" y "`Esperar evento`". Los nodos de propiedad de lectura y escritura se pueden emplear para leer

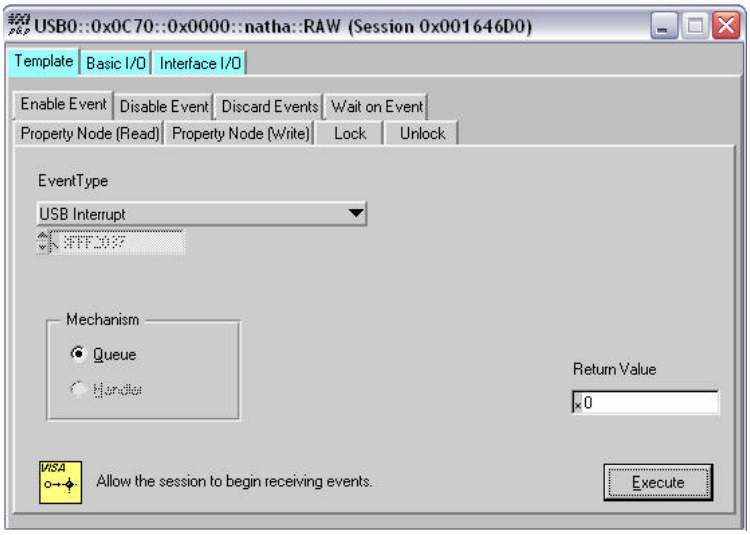
o escribir en las propiedades disponibles para el dispositivo. Las propiedades disponibles dependen del dispositivo y se pueden observar en la ventana desplegable "Nombre de la propiedad" (*Attribute Name*). Estas propiedades se relacionan con el estado del dispositivo, con las configuraciones disponibles y con las propiedades particulares del dispositivo (número de interfaces, clase, subclase, entre otros).

La pestaña "Habilitar evento" sirve para habilitar eventos relacionados con el dispositivo USB y el driver; la pestaña "Esperar evento" permite identificar cuándo ocurre un evento para realizar un procedimiento particular. Estas dos últimas pestañas se emplean para recibir los datos transmitidos por el dispositivo USB empleando transferencias por interrupción, de la siguiente forma: en la pestaña habilitar evento, mostrada en la Figura 12, se selecciona de la lista desplegable "Tipo de Evento" (*Event Type*) la opción Interrupción USB (*USB Interrupt*) y se hace clic en el botón "Ejecutar" (*Execute*). Cuando se habilita el evento "Interrupción USB", el *driver* habilita un *Pipe* (*Canal*) para la transmisión de datos desde el dispositivo USB (tarjeta de adquisición de datos). Para observar los datos se debe acceder a la pestaña "Esperar evento" (*Wait on event*) mostrada en la Figura 13, que se encuentra dentro de la misma pestaña principal "Plantilla" (*Template*).

Cada vez que se desee recibir una transferencia de los datos transmitidos empleando el protocolo USB[46], se debe seleccionar la opción "Interrupción USB" (*USB Interrupt*) en la lista desplegable "Tipo de evento" (*Event Type*) de la pestaña "Esperar evento" (*Wait on event*) y hacer clic en el botón "Ejecutar" (*Execute*). Los datos recibidos y otra información como el tipo de evento, el estado del evento, el tamaño de la transferencia se muestran en el cuadro "Información del evento" (*Event Information*) que se puede observar en la Figura 13.

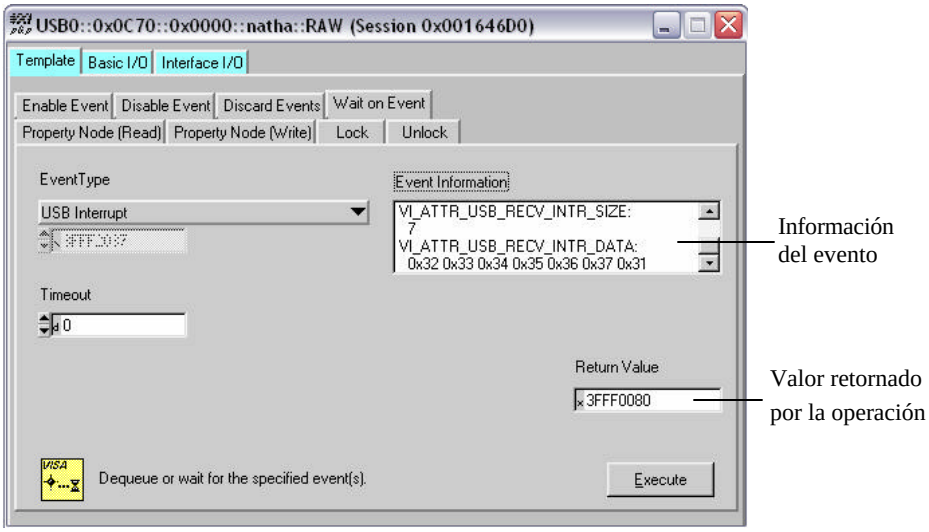
[46] Ver aparte 1.3.1 del capítulo 1.

Figura 12. Pestaña Habilitar evento.



Fuente: los autores.

Figura 13. Pestaña Esperar evento.

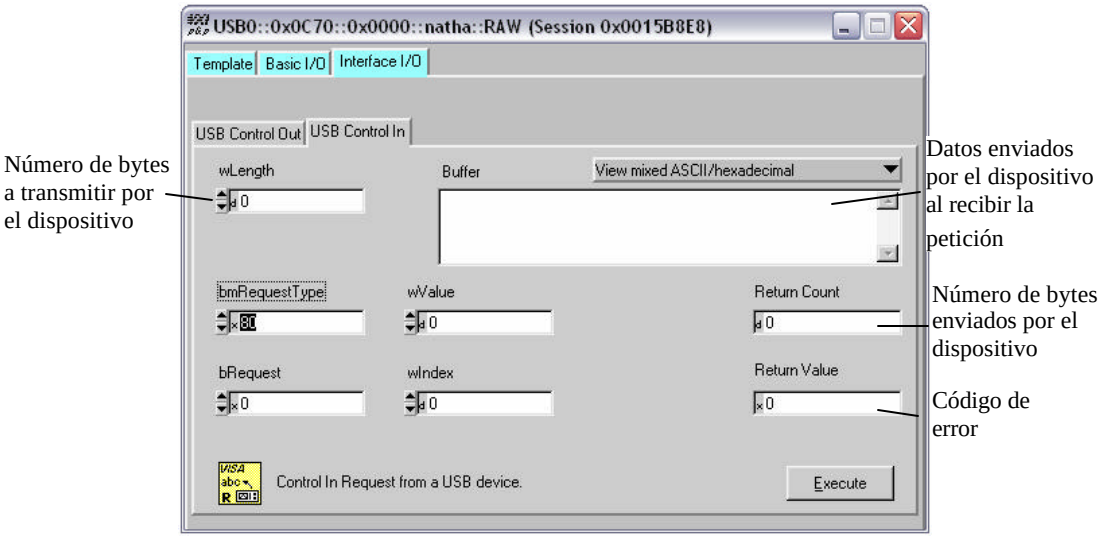


Fuente: los autores

El valor retornado por la operación (*Return Value*) 3FFF0080, indica que la operación fue exitosa y que existe por lo menos otro evento igual al seleccionado para la operación anterior disponible en la sesión. El *driver* VISA soporta máximo 50 eventos en cola por defecto.

o La pestaña de Interfaz de entrada y salida contiene dos funciones importantes: Control USB de salida (*USB Control OUT*) que se emplea para enviar peticiones al dispositivo con datos de salida (datos hacia el dispositivo) y Control USB de entrada (*USB Control IN*) que se emplea para enviar peticiones al dispositivo con datos de entrada (datos hacia el *host*). La pestaña Control USB de entrada se muestra en la Figura 14.

Figura 14. Ventana para controlar las interfaces del dispositivo USB.



Fuente: los autores.

Debido a que el *driver* VISA configura automáticamente los dispositivos USB con las opciones por defecto, las únicas peticiones permitidas para la tarjeta SAD800-L y la tarjeta de desarrollo para el microcontrolador MC68HC908JB8JP son las de control de entrada. Las peticiones de salida se emplean cuando el dispositivo controlado por el *driver* VISA contiene más de una configuración, o más de una interfaz[47]. Un ejemplo de las peticiones de control de entrada se muestra en la Tabla 1.

[47] Ver aparte 9.2.3 de las Especificaciones USB 1.1.

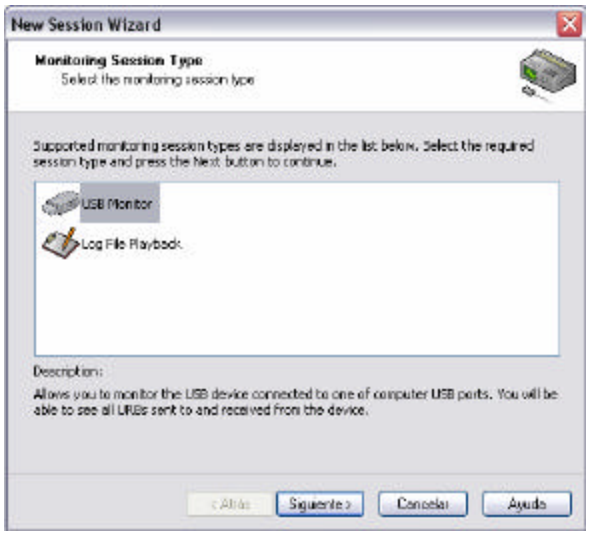
Tabla 1. Peticiones de Control USB de entrada.

PETICIÓN	bmRequestType	bRequest	wValue	wIndex	wLength	Data
GET_STATUS	0x80 (hexa)	0	0	0	2 dec	0
GET_CONFIGURATION	0x80 (hexa)	0x08 (hexa)	0	0	1 dec	0
GET_DESCRIPTOR	0x80 (hexa)	0x06 (hexa)	512 dec (config)	0	32 dec[48]	0

2. MANUAL DEL MONITOR DE PROTOCOLO USB (USB *MONITOR*)

Para iniciar una sesión con el Monitor USB, se debe seleccionar la opción "Nueva `Sesión`" (*New Session*) del menú "`Archivo`" (*File*). Una vez seleccionada, aparece la ventana que se muestra en la Figura 15. En esta ventana se debe seleccionar la opción USB Monitor y hacer clic en el botón "`Siguiente`".

Figura 15. Ventana para iniciar una nueva sesión de monitoreo de protocolo USB

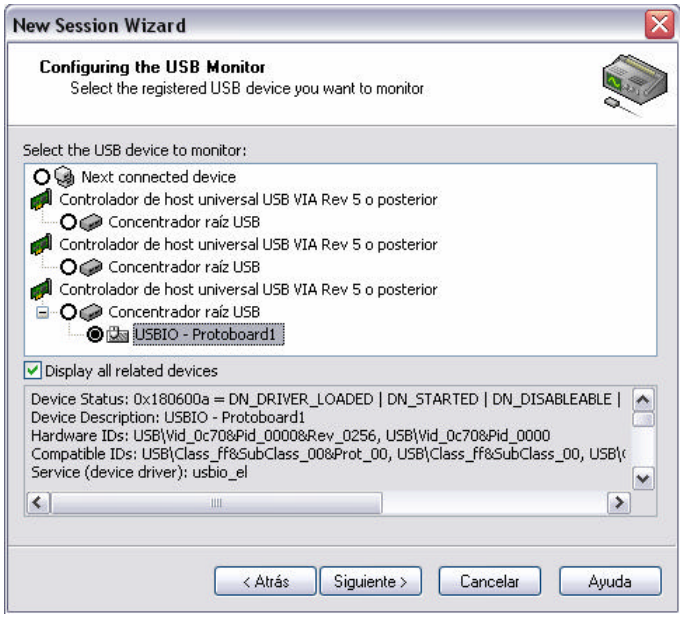


Fuente: los autores.

La siguiente ventana corresponde a la mostrada en la Figura 16, y es la que permite seleccionar el dispositivo USB que se desea monitorear. Una vez seleccionado, se hace clic en "`Siguiente`".

[48] Con este valor el dispositivo envía los descriptores tanto de la configuración, como de la interfaz y del *endpoint* para un dispositivo con una sola configuración y una sola interfaz. Corresponde al largo en bytes de los datos pertenecientes a todos los descriptores del dispositivo que se desean obtener.

Figura 16. Ventana para seleccionar dispositivo USB a monitorear.

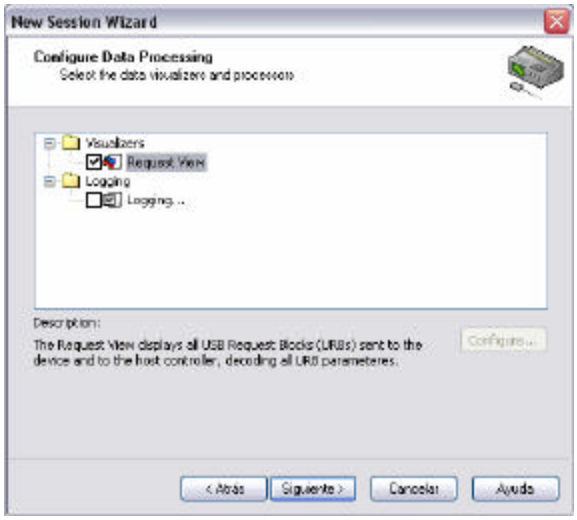


Fuente: los autores.

El último paso es configurar el tipo de datos que se desea procesar, en la ventana de la Figura 17. Se debe seleccionar la opción "`Vista de peticiones`" (*Request View*) y hacer clic en "`Siguiente`".

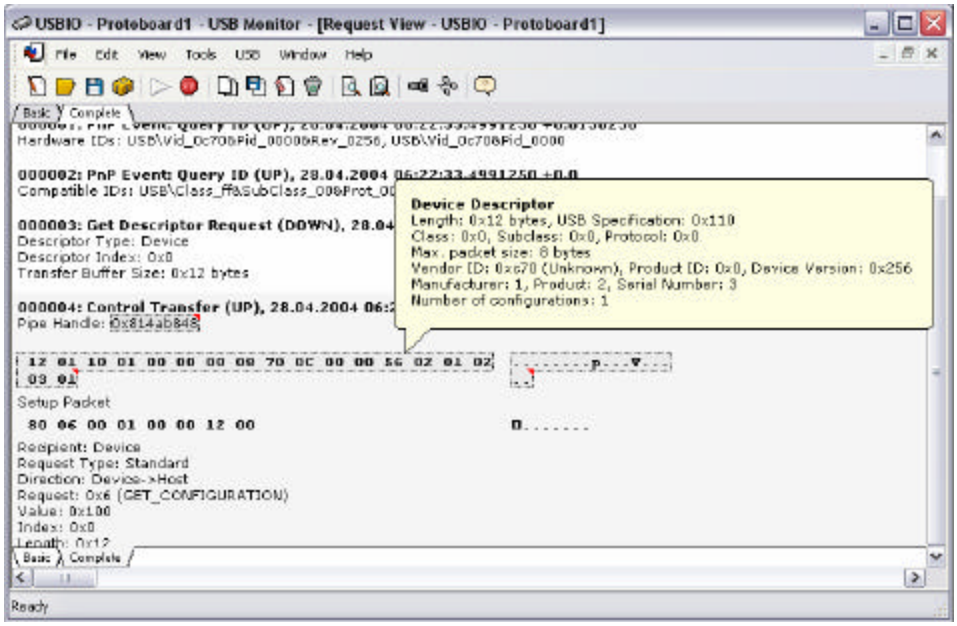
La Figura 18 muestra una sesión iniciada con la tarjeta de desarrollo diseñada en este proyecto. Para observar la información completa de las peticiones se hace clic en la pestaña Completo (*Complete*). Para interpretar la información contenida en la petición de un descriptor, basta con colocar el cursor sobre los bytes transmitidos en la petición y un cuadro de diálogo aparecerá con la información interpretada, como se muestra también en la Figura 51. De esta forma es muy sencillo realizar un análisis del funcionamiento del dispositivo USB con el que se inició la comunicación y del protocolo USB en general.

Figura 17. Ventana de configuración del tipo de datos a procesar.



Fuente: los autores.

Figura 51. Ventana principal de una sesión iniciada con la tarjeta de desarrollo diseñada en este proyecto.



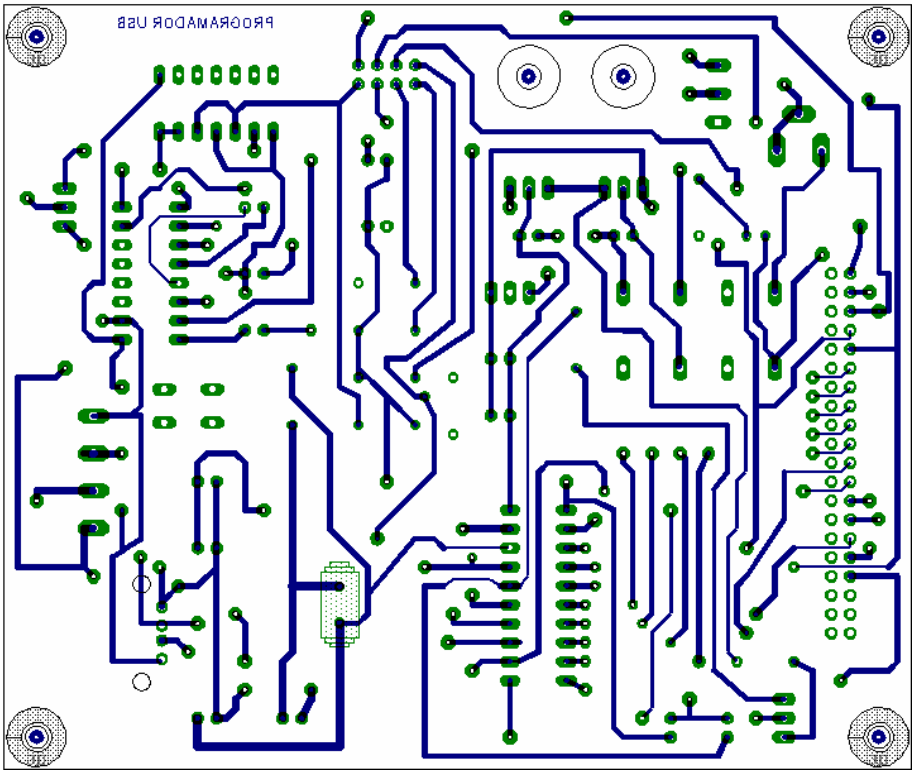
Fuente: los autores.

ANEXO B. DIAGRAMA DE LAS TARJETAS

B.1 DIAGRAMA DE LA TARJETA DE DESARROLLO PARA EL MICROCONTROLADOR MC68HC908JB8JP

En la Figura 1 se muestra el diagrama de la cara inferior de la Tarjeta de Desarrollo.

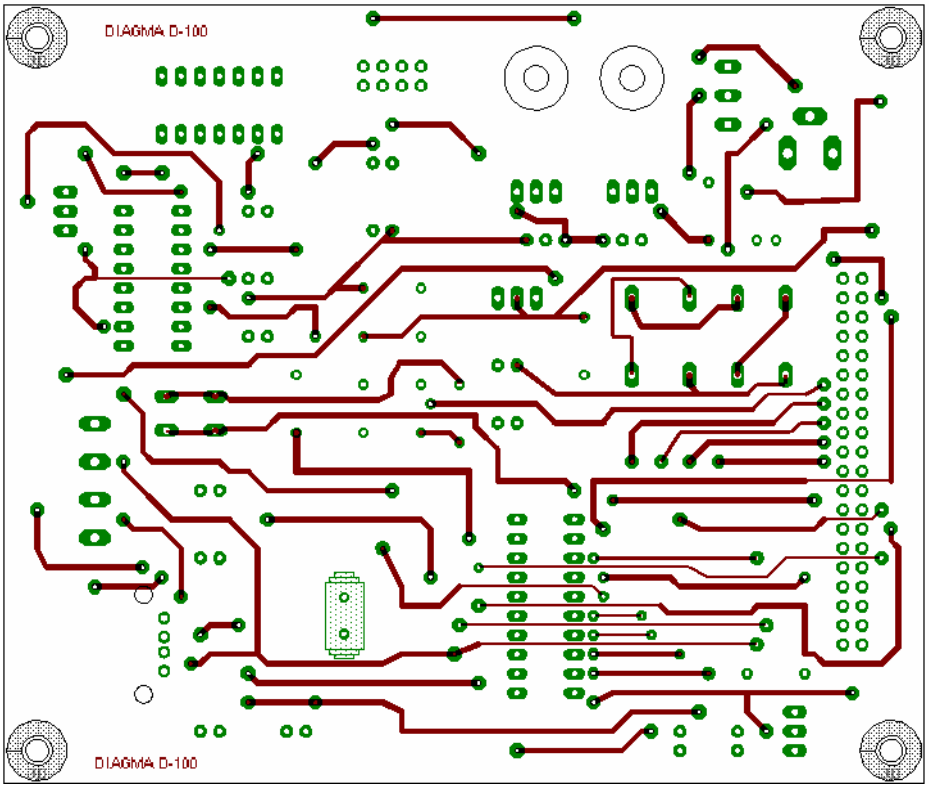
Figura 1. Diagrama de la cara inferior del circuito impreso de la Tarjeta de Desarrollo para el microcontrolador MC68HC908JB8JP



Fuente: los autores

En la Figura 2 se muestra la cara superior de la Tarjeta de Desarrollo.

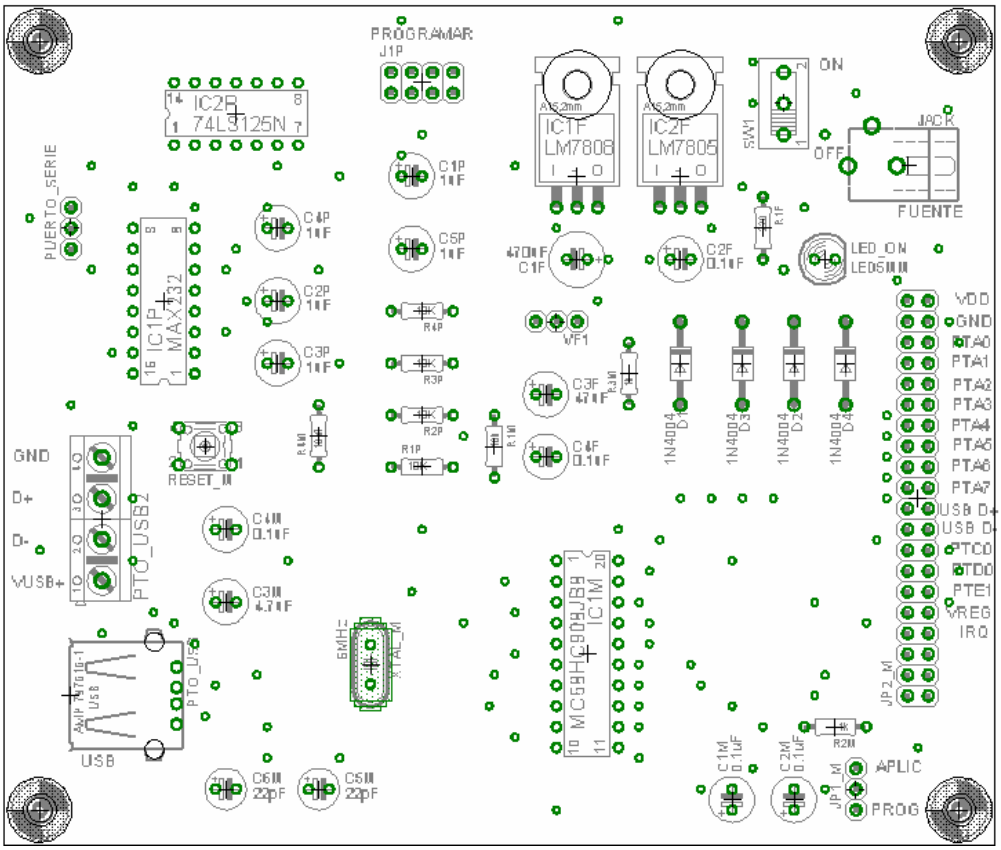
Figura 2. Diagrama de la cara superior del circuito impreso de la Tarjeta de Desarrollo para el microcontrolador MC68HC908JB8JP



Fuente: los autores.

En la Figura 3 se muestra el diagrama de los elementos del circuito impreso de la tarjeta de desarrollo.

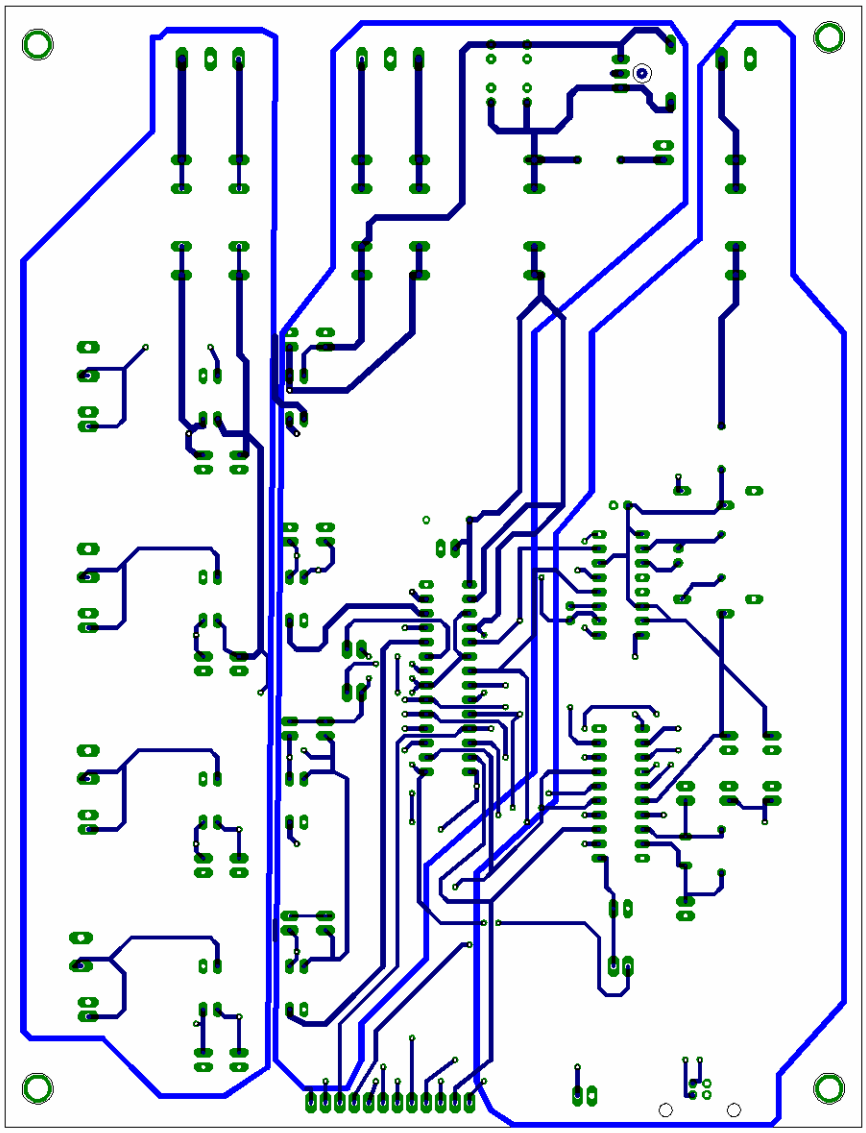
Figura 3. Diagrama de la cara de los elementos del circuito impreso de la Tarjeta de Desarrollo para el microcontrolador MC68HC908JB8JP



B.2 DIAGRAMA DE LA TARJETA SAD800-L

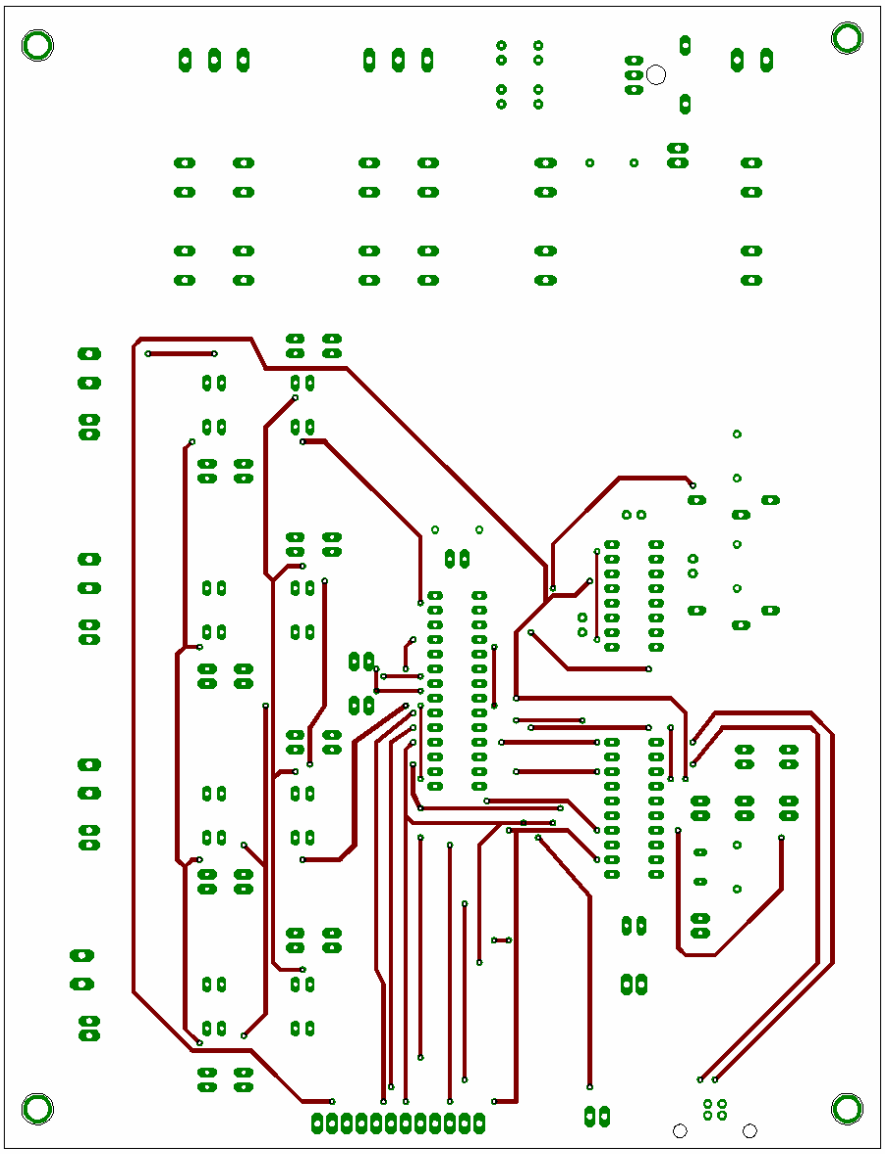
En la Figura 4 se muestra el diagrama de la cara inferior de la Tarjeta SAD800-L

Figura 4. Diagrama de la cara superior de la Tarjeta SAD800-L.



En la Figura 5 se muestra el diagrama de la cara superior de la Tarjeta SAD800-L.

Figura 2. Diagrama de la cara superior de la Tarjeta SAD800-L.



En la Figura 6 se muestra el diagrama de los elementos de la Tarjeta SAD800-L.

Figura 6. Diagrama de los elementos de la Tarjeta SAD800-L.