# PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

## FACULTAD DE CIENCIAS E INGENIERÍA


# DISEÑO DE UN SISTEMA DE ADQUISICIÓN DE DATOS UTILIZANDO EL PROTOCOLO USB EN UN MICROCONTROLADOR AVR

Tesis para optar el Título de **Ingeniero Electrónico,** que presenta el bachiller**:**

**Richard Armando Nole Calle**

**ASESOR: José Daniel Alcántara Zapata**

Lima, noviembre del 2013

## RESUMEN

Existen diversas tecnologías de comunicación con una computadora para la adquisición de datos, entre los más comunes se encuentran: PCI, USB, Ethernet, Firewire, puerto serial, etc. Dentro de ellos, el USB destaca por su configuración automática, bajo costo y facilidad de uso. A pesar de existir varios sistemas de adquisición de datos en el mercado, su uso se ve restringido debido a sus altos costos y por poder usarlo sólo con las aplicaciones y drivers del proveedor.

En el presente trabajo se diseña un sistema de adquisición de datos con interface USB utilizando un microcontrolador Atmel de la familia AVR XMEGA, buscando en todo momento obtener la máxima tasa de transferencia posible. Para ello se desarrolla una aplicación en el microcontrolador que permita leer datos adquiridos de cuatro canales del ADC del microcontrolador. Asimismo, se desarrolla una clase USB propietaria que define cómo se van a trasferir los datos, qué tipo de transferencias USB se van a usar y cuál va a ser su máxima tasa de transferencia posible. En el lado de la computadora, se desarrolla una aplicación en lenguaje C que permita a la computadora poder comunicarse con el microcontrolador a través del bus USB.

Para el desarrollo del firmware del microcontrolador se tomó como base el framework USB que provee Atmel (ASF 3.1.3, Atmel Software Framework), y posteriormente fue implementado en la tarjeta de evaluación XMEGA – A3BU XPLAINED con una frecuencia de CPU de 32MHz. Se obtuvo que en promedio se puede transmitir datos a 8.46Mbps usando un alto nivel de optimización del compilador. Se concluye que si se quiere obtener mejores tasas de transferencia se debe mejorar una serie de factores como: MIPS del microcontrolador y optimización del framework USB de Atmel.

Finalmente, se recomienda portar este trabajo a microcontroladores que soporten el modo “Alta Velocidad” (del inglés *High Speed*) del USB 2.0, cuya velocidad por bit es de 480Mbps, así como desarrollar las etapas de preprocesamiento de las señales: amplificación, filtrado, e aislamiento de las señales que se quieran enviar por la interfaz USB.

**TEMA DE TESIS PARA OPTAR EL TÍTULO DE INGENIERO ELECTRÓNICO**

| | | |
|---|---|---|
| Título | : | Diseño de un sistema de adquisición de datos utilizando el protocolo USB en un microcontrolador AVR. |
| Área | : | Electrónica |
| Asesor | : | José Daniel Alcántara Zapata |
| Alumno | : | Richard Armando Nole Calle |
| Código | : | 20070251 |
| Fecha | : | 04/11/13 |

**Descripción y Objetivos**

En la actualidad, con el desarrollo de la tecnología, la adquisición de datos y la comunicación entre computadoras y periféricos en general juegan un rol importante. Para ello, se ha desarrollado interfaces que permitan esta comunicación como por ejemplo: RS-232, USB, Firewire, PCI, etc. Dentro de ellos, el USB destaca por su configuración automática, bajo costo y facilidad de uso.

Si bien en varias aplicaciones es suficiente el uso de un puerto serial, una interface USB ofrece versatilidad además de alcanzar velocidades de transferencia más altas. De esta forma, se benefician tanto usuarios como diseñadores.

La presente tesis tiene como objetivo el diseño de un sistema de adquisición de datos con interface USB utilizando un microcontrolador Atmel de la familia AVR XMEGA. Esto permitirá que se puedan desarrollar aplicaciones en las cuales se requieran transferir datos a una computadora a altas velocidades y que sea económico y sencillo de usar.

El diseño del sistema propuesto incluye el análisis de otras interfaces de computadora (PCI, Ethernet, etc), el desarrollo del firmware que permita al microcontrolador enviar datos de acuerdo al protocolo USB 2.0, así como la implementación de un software de aplicación que permita a la computadora comunicarse con este dispositivo a través del puerto USB.

**TEMA DE TESIS PARA OPTAR EL TÍTULO DE INGENIERO ELECTRÓNICO**

Título : Diseño de un sistema de adquisición de datos utilizando el protocolo USB en un microcontrolador AVR.

**Índice**

# INDICE

# INDICE DE FIGURAS

# GLOSARIO

**ACK**: Paquete de datos indicando una confirmación de recepción de datos.

**DMAC**: Del inglés *Direct Memory Access Controller*. Periférico de un microcontrolador que permite transferir datos entre memorias y periféricos, por lo que permite reducir la intervención del CPU.

**Endpoint**: Buffer que almacena varios bytes en el dispositivo USB. Los datos almacenados en el endpoint pueden ser datos recibidos por el host o datos en espera para ser transmitidos hacia el host.

**EOF**: Del inglés *End of Frame* (Fin de trama).

**EOP**: Del inglés *End of Paquete* (Fin de paquete).

**Full Speed**: Velocidad del bus USB a 12Mbps.

**High Speed**: Velocidad del bus USB a 480Mbps.

**Host**: Computadora donde el controlador de host USB está instalado.

**Paquete:** Conjunto de datos que se transmiten en grupo. Los paquetes contienen tres elementos: información de control (dirección de la transmisión, tamaño de paquete, etc.), datos útiles y bits de error de corrección.

**Pipe**: Una abstracción lógica que asocia un endpoint en el dispositivo USB y el software en el host.

**Start of Frame (SOF)**: Es la primera transacción de cada trama que permite a los endpoints identificar cuándo empieza una nueva trama en el bus USB.

**Transferencia Bulk**: Uno de los cuatro tipos de transferencia USB. Las transferencias tipo Bulk son usadas en transferencias que pueden usar el

ancho de banda disponible, y a su vez pueden ser retrasadas hasta que haya ancho de banda disponible. Es utilizado por ejemplo en dispositivos como impresoras y escáneres.

**Transferencia de Control**: Las transferencias de control permiten enviar mensajes de configuración, comandos y status entre el host y el dispositivo USB.

**Transferencia de Interrupción**: Este tipo de transferencia se caracteriza por transferir pequeñas cantidades de datos y no periódicos. Es utilizado por ejemplo en dispositivos como el mouse y el teclado.

**Transferencia Isócrona**: Son transferencias en tiempo real útiles cuando los datos debe ser transmitidos a una tasa de transferencia constante y donde errores ocasionales en los datos pueden ser tolerados. Un ejemplo de uso de transferencia isócrona es reproducir música en tiempo real.

# INTRODUCCIÓN

En la actualidad la interfaz USB es capaz de adaptarse a un amplio rango de periféricos: teclado, impresora, escáner, etc. Asimismo, es adecuado para sistemas de adquisición de datos tanto de propósito general como de uso específico.

El presente trabajo propone el diseño de una interfaz USB personalizada para aplicaciones que requieran alta tasa de transferencia de datos y que la integridad de los datos transferidos esté garantizada. Para ello, se hace uso de la transferencia Bulk, uno de los cuatro tipos de transferencia en el protocolo USB. De esta manera, se tiene un mayor control de cómo se transmiten los datos del dispositivo USB al software de aplicación.

El presente trabajo de tesis está dividido en cuatro capítulos, en el primero de ellos se hace una introducción a los sistemas de adquisición de datos y a la interfaz USB, comparándolo con otras tecnologías y explicando las ventajas que ofrece respecto a estas. El segundo capítulo está enfocado en el estudio del protocolo USB, analizando sus diferentes elementos y tipos de transferencia que ofrece, además se presentarán avances en el desarrollo de interfaces de entrada para computadora. El tercer capítulo está dedicado al desarrollo del firmware para el sistema de adquisición de datos así como la implementación del programa de aplicación en el lado de la computadora. Finalmente en el cuarto capítulo se muestran las pruebas realizadas y se evalúan los resultados.

# CAPÍTULO 1

# ACERCA DE LOS SISTEMAS DE ADQUISICIÓN DE DATOS

## 1.1 Sistemas de Adquisición de datos

En la actualidad, encontramos una gran variedad de sistemas de adquisición de datos que permiten extraer información a partir de diversas fuentes: Ondas de sonido, luz, temperatura, presión; convirtiéndolas a señales digitales para ser procesadas y visualizadas a través de una interfaz de usuario.

Hoy en día empresas como Texas Instruments, Freescale Semiconductor, Measurement Computing, entre otros, ofrecen un variado rango de sistemas de adquisición en términos de número de canales analógicos, resolución del ADC, forma de comunicación con la computadora, etc. que se ajustan a diversas aplicaciones.

## 1.2 Interfaces de Computadora

A través de los años se han desarrollado diferentes tipos de interfaces para poder transmitir datos entre un periférico y la computadora. Así, inicialmente se tenían los puertos serial y paralelo a través de los cuales se podía controlar impresoras, módems, etc. Sin embargo, esto implicaba tener diferentes tipos de puertos para cada dispositivo así como drivers personalizados.

Posteriormente se diseñaron otras interfaces que alcanzaban velocidades más altas. Tal es el caso de los estándares Firewire, USB, Gigabit Ethernet y PCI Express.

La tabla 1 muestra una comparación entre distintas interfaces de conexión.

**Tabla 1: Comparación de interfaces de computadora [1].**

| Interfaz | Formato | Velocidad máxima (bps) | Uso típico |
|---|---|---|---|
| **RS-232** | Serial | 20k | Módem, mouse, instrumentación |
| **Puerto Paralelo** | Paralelo | 8M | Impresoras, escaners, discos externos |
| **RS-485** | Serial | 10M | Adquisición de datos y sistemas de control |
| **USB** | Serial | 1.5M, 12M, 480M (Depende de la versión) | Mouse, teclado, disco duro, módem, audio |
| **Fire Wire 800 (IEEE-1394)** | Serial | 800M | Stream de video y audio digital de alta resolución |
| **Gigabit Ethernet** | Serial | 1G | Acceso a archivos, audio y video a través de una red |
| **Thunderbolt** | Serial | 20G(10G por cada canal) | Stream de audio/video digital de alta resolución |

De especial interés es la interfaz USB pues además de sus altas velocidades presenta otros beneficios para los usuarios finales tales como: versatilidad (puede ser usado con distintos periféricos), bajo costo, configuración automática, etc.

## 1.3 Problemática

En la actualidad existen distintas tecnologías para poder adquirir datos y transmitirlos hacia alguna computadora para su posterior procesamiento, visualización o almacenamiento. Asimismo, las interfaces entre periféricos y computadoras han ido mejorando a través de los años en términos de velocidad, facilidad de uso, versatilidad, etc.

Sin embargo, siempre existe la necesidad de contar con un dispositivo que permita adquirir datos a alta velocidad y que además sea económico y sencillo de usar. Debido a esto, el presente trabajo busca desarrollar un sistema que permita adquirir datos haciendo uso de la interfaz USB en un microcontrolador AVR. Así, se trabajará con el protocolo USB para aprovechar las tasas de transferencia que ofrece esta interfaz, de tal manera que el resultado final del presente trabajo sea una mejor performance respecto a enfoques más tradicionales de abordar el problema (como usar un chip conversor USB - RS232).

# CAPÍTULO 2

# TECNOLOGÍAS DE ADQUISICIÓN DE DATOS

## 2.1 Estado del arte

### 2.1.1 Presentación del asunto de estudio

En la actualidad encontramos una gran variedad de sistemas de adquisición de datos los cuales permiten adquirir señales de distintos tipos como: audio, corriente, video, etc. Adicionalmente, estos equipos pueden contar con un software de aplicación que permite almacenar y analizar los datos.

Por otro lado, debido a la necesidad de contar con una interface que permita una alta transferencia de datos así como facilidad de uso para el usuario final, cada vez la interfaz USB se hace más popular para transmitir los datos del dispositivo a la computadora (host) respecto a otros protocolos (Firewire, eSATA, etc.)

Por tal motivo, el dispositivo o interfaz USB presenta un amplio campo de acción a nivel comercial. La gran diversidad de aplicaciones es uno de sus muchos beneficios, pues en vez de tener conectores diferentes para cada periférico, la interfaz USB hace viable adaptarlo a distintas necesidades.

Por lo tanto, el presente estudio busca el desarrollo de un sistema de adquisición de datos con interfaz USB para la comunicación con el computador, evaluando hasta qué tasa de transferencia es posible llegar.

### 2.1.2 Estado de la investigación

A continuación se hace una evaluación de las principales tecnologías de comunicación con la computadora así como de tarjetas de adquisición en el mercado.

#### 2.1.2.1 Interfaces de computadora

**A. Firewire**

IEEE1394, también conocido como Firewire (Apple Inc), I.Link (Sony) y Lynx (Texas Instruments) fue desarrollado inicialmente por la compañía Apple como reemplazo al bus SCSI (del inglés “Small Computer System Interface”)[2].

Los dispositivos que usan la interface Firewire soportan velocidades de transferencia entre 400-800Mbps, lo cual lo hace ideal para la transferencia de de audio o video en tiempo real.

**Figura 2.1 Conector Firewire de 4 pines (izquierda) y 6 pines (derecha)**

Una ventaja que ofrece este estándar frente al USB es su topología Peer-to-Peer, lo cual permite que dos dispositivos se comuniquen entre sí sin la intervención de una computadora para gestionar el bus.

La desventaja de esta tecnología es que no es común encontrarla en las PCs; si bien se encuentran usualmente en las computadoras de Apple, es mucho más común que una computadora cuente con interfaz USB.

## B. PCI Express

El PCI Express (abreviado como PCIe) es una evolución del bus PCI, reemplazando el bus paralelo con un bus serial de alta velocidad. Provee una conexión full-duplex y con señalización diferencial para interconectar dispositivos.

A diferencia del PCI tradicional, el cual comparte ancho de banda con todos los dispositivos en el bus, con PICe cada dispositivo recibe un ancho de banda dedicado.

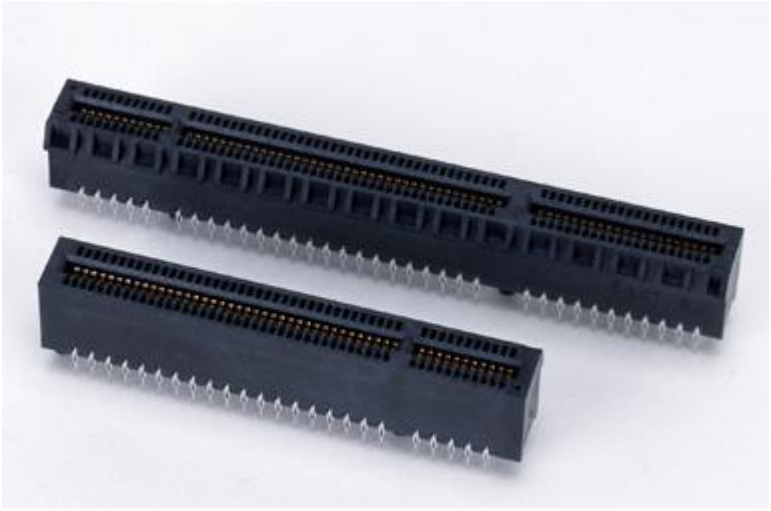


**Figura 2.2 Conectores PCIe**

El estándar PCIe está en constante desarrollo y mejora. PCIe 1.0 fue presentado por Intel en 2004 con tasas de datos por dirección y por línea

de 250 MB/s. Le siguió la revisión PCIe 2.0, la cual estuvo disponible en 2007, duplicando la tasa de datos de PCIe 1.0 de 250MB/s a 500MB/s. Posteriormente, la revisión 3.0 estuvo disponible en 2010, la cual provee una tasa de 1GB/s[3].

Como se puede apreciar, debido a su alto ancho de banda, PCIe es un bus idóneo para sistemas de adquisición de datos que requieran un alto rendimiento. Estas tarjetas, sin embargo, tienen la desventaja de tener un costo bastante alto.

### C. Ethernet

El estándar IEEE802.3 (popularmente conocido como Ethernet) hace referencia al conjunto de tecnologías usadas en las redes de área local (LAN). A través de los años se han definido varias tasas de transferencia de datos a través de fibra óptica o cable de par trenzado:

- 10Base-T Ethernet: 10Mbps
- Fast Ethernet:100 Mbps
- Gigabit Ethernet: 1000 Mbps
- 10-Gigabit Ethernet: 10 Gbps
- 100-Gigabit Ethernet: 100 Gbps

Existen varias razones por las cuales el estándar Ethernet es una elección popular cuando se quiere conectar computadoras o sistemas embebidos a una red [4]:

- Versatilidad: Si bien Ethernet es usado comúnmente para conectar computadoras a una red, éste estándar puede transferir cualquier tipo de data, desde mensajes cortos hasta archivos. Asimismo, Ethernet puede aprovechar otros protocolos de más alto nivel como TCP/IP (Ethernet pertenece a la capa 1 y 2 del modelo OSI).

- Rapidez: Ethernet soporta velocidades de 10Mbps hasta 100Gbps.

- Alcanza largas distancias.

Con respecto a la adquisición de datos, la interfaz Ethernet es ideal cuando se quiere extender el alcance y obtener mediciones remotas.

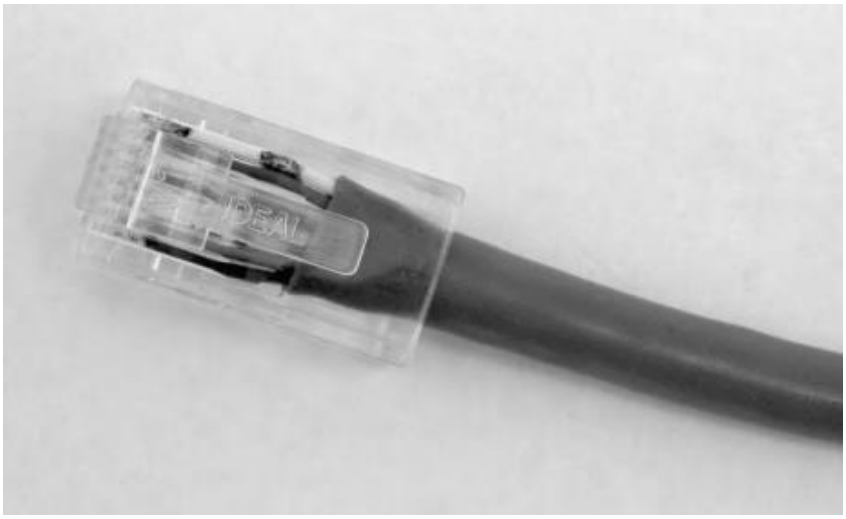


**Figura 2.3 Cable Ethernet [4].**

Sin embargo, Ethernet no es la solución para todos los sistemas. Para ciertas aplicaciones, existen soluciones más baratas y más simples de implementar. A continuación se listan unas cuantas limitaciones:

- Costo: Si se quiere mantener el costo al mínimo, la interfaz USB es una solución más adecuada
  .
- Limitaciones de tiempo real: Ethernet no garantiza que la transferencia de datos ocurran con un retardo mínimo o que ocurra dentro de un tiempo determinado. A veces la red puede estar congestionada debido a que otras computadoras están usando el ancho de banda.

## D. USB

El estándar USB (del inglés *Universal Serial Bus*) fue diseñado para comunicarse con varios tipos de periféricos. Cuando la revisión 1.0 estaba en desarrollo, se tenía puertos seriales, paralelos, puertos joystick, tipo midi, entre otros; pero todos eran dedicados para un uso específico.

El estándar ha ido mejorando las tasas de transferencia a través de los años, inicialmente en la revisión 1.0 podía transferir 1.5 Mbits/s y actualmente con los puertos USB 3.1 se puede llegar a transferencias de 10Gbits/s. Actualmente existen 4 velocidades de transferencia:

- Velocidad Baja (Low Speed): 1.5Mbps
- Velocidad Completa (Full Speed): 12Mbps
- Alta Velocidad (High Speed): 480Mbps
- Super Alta Velocidad (SuperSpeed): 10Gbps

Lo que hace atractivo usar USB para sistemas de adquisición de datos es que estos se benefician de la simplicidad, portabilidad, así como por el bajo costo por interface. Asimismo, ofrece la ventaja de poder energizar el sistema por el bus [1].

**Figura 2.4 Tipos de conectores USB.**
**De izquierda a derecha: tipo A, tipo B, mini-A, mini-B [1].**

No obstante, se debe tener en cuenta que la interface USB puede ser poco práctica para cierto tipo de aplicaciones:

- Velocidad: si bien USB es rápido, ya se ha visto que hay otras interface mucho más veloces como Gigabit Ethernet, PCI Express, etc.
- Distancia: USB fue diseñado como un bus de expansión, y se espera que el periférico esté relativamente cerca de la computadora. Si se quisiera trabajar a grandes distancias, existen otras soluciones como RS-485 o Ethernet.

### 2.1.2.2 Tarjetas de adquisición de datos

Un gran número de fabricantes ofrecen productos de adquisición de datos para PCs. La mayoría de productos cuentan con algunas de las siguientes características:

- Bus: Puede ser USB, Wi-fi o Ethernet, los cuales ofrecen portabilidad al producto. También se puede usar el bus PCI, el cual se instala en una ranura de la computadora.

- Razón de muestreo: Número de muestras por unidad de tiempo que se toma de una señal. Dependiendo del dispositivo, se puede muestrear en el orden de los kS/s (del inglés *kilo samples per second)* hasta los MS/s (del inglés *mega samples per second)*.

- Resolución: Número de bits que determinan la cantidad de niveles en que puede ser dividida la señal analógica.

- Canales de Entrada/Salida: Canales analógicos o digitales de entrada y salida que proporciona el sistema. En algunos productos es posible aumentar el número de canales a través de módulos de expansión.

- Acondicionamiento de señales: Circuito para acondicionar la señal antes de ser digitalizada. Dependiendo del modelo del producto, algunos cuentan con circuito de acondicionamiento integrado y otros no.

En la tabla 2 se muestran algunas tarjetas comerciales de adquisición de datos de bajo costo y de propósito general.

**Tabla 2: Tarjetas de adquisición de datos de bajo costo y de propósito general.**

| | USB-6008[5] | USB-6009[5] | DT9812[6] | DI710[7] | DI720[7] | DT301[8] |
|---|---|---|---|---|---|---|
| Entradas analógicas | 8 | 8 | 8 | 16 | 32 | 16 |
| Resolución(bits) | 12 | 14 | 12 | 14 | 16 | 12 |
| Frecuencia de muestreo (muestras/segundo) | 10kS/s | 48kS/s | 100kS/s | 4.8kS/s | 250kS/s | 225kS/s |
| Rango de voltaje de entrada(V) | +/-10V | +/-10V | +/-10V | +/-10V | +/-10V | +/-10V |
| Interface de comunicación con PC | USB | USB | USB | Ethernet | Ethernet | PCI |
| Precio | $210 | $360 | $305 | $599 | $1590 | $765 |
| Fabricante | National Instruments | National Instruments | Data Translation | DATAQ Instruments | DATAQ Instruments | Data Translation |

### 2.1.3 Síntesis sobre el asunto de estudio

Los nuevos requerimientos de obtener señales del mundo físico a mayores velocidades de transferencia generan la necesidad de trabajar con interfaces como las mencionadas en la sección anterior. Dentro de este ámbito, el puerto USB surge como una alternativa eficiente, pues presenta altas velocidades de transferencia, es de bajo costo y flexible para un gran número de aplicaciones usando un mismo tipo de conector.

Por tal motivo, el presente estudio propone desarrollar un sistema de adquisición de datos que se comunique con la computadora a través del puerto USB para que sea el precedente de trabajos posteriores en el desarrollo de sistemas con conexión USB a velocidades más altas.

Un punto importante que cabe resaltar es que el diseño va a depender del chip que contiene el controlador USB a utilizar, pues presentan diferentes componentes y limitaciones de hardware según la familia a la cual pertenezcan y la empresa que los desarrolla.

Finalmente, se obtendrán conclusiones a partir de los resultados obtenidos, los cuales serán de utilidad para trabajos futuros.

## 2.2 Protocolo USB

### 2.2.1 Definiciones Pertinentes

De acuerdo al documento “Universal Serial Bus Specification” [9], el protocolo USB se divide en una serie de capas físicas y lógicas que permitan un mejor entendimiento del protocolo y, a su vez, enfocarse en diferentes aspectos del desarrollo de un dispositivo USB.
Para poder entender la estructura del protocolo, es necesario definir previamente algunos términos:

- Terminal (en inglés *Endpoint*): Son buffers del dispositivo USB donde se almacena datos para transmitir o se recibe datos de la computadora (“host” en terminología USB).

- Canal de comunicación (en inglés *Pipe*): Una abstracción lógica que representa la asociación entre un endpoint en el dispositivo y el endpoint en el host.

- Dispositivo Lógico USB: Abstracción visto desde el host como una colección de endpoints. Proveen una capacidad específica para el host, por ejemplo: imprimir, adquirir data, mover puntero del mouse, etc. También llamado “interface lógica”.

- Función: Colección de interfaces. Se presenta cuando el dispositivo USB tiene múltiples interfaces lógicas. Por ejemplo: una impresora multifuncional que puede funcionar como escáner, fotocopiadora, etc. Por lo general los dispositivos USB tienen sólo una interface.

La figura 2.5 muestra cómo es el flujo de comunicación en el bus USB haciendo uso de los enpoints y pipes:

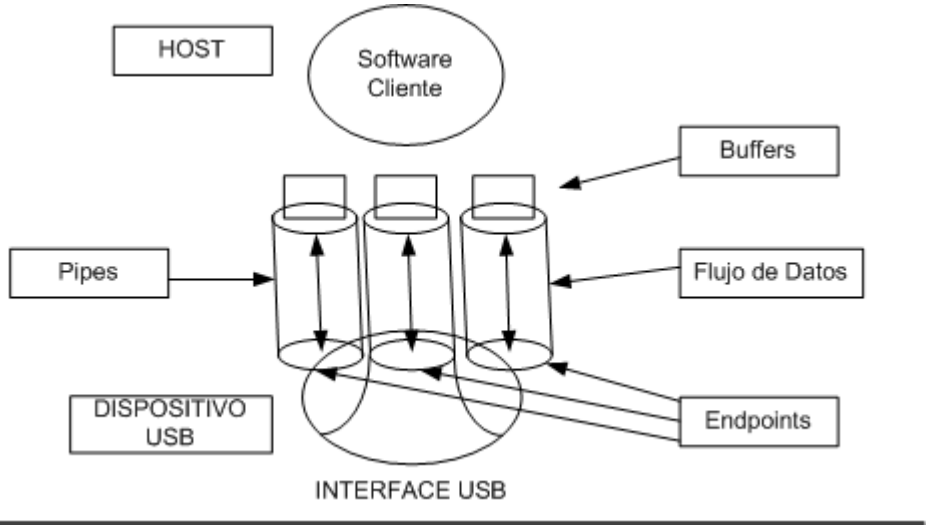



**Figura 2.5 Capas de flujo de datos entre el host y el dispositivo [9].**

La figura 2.6 muestra una vista más detallada de las capas del protocolo a través de las cuales fluye la comunicación entre host y dispositivo:

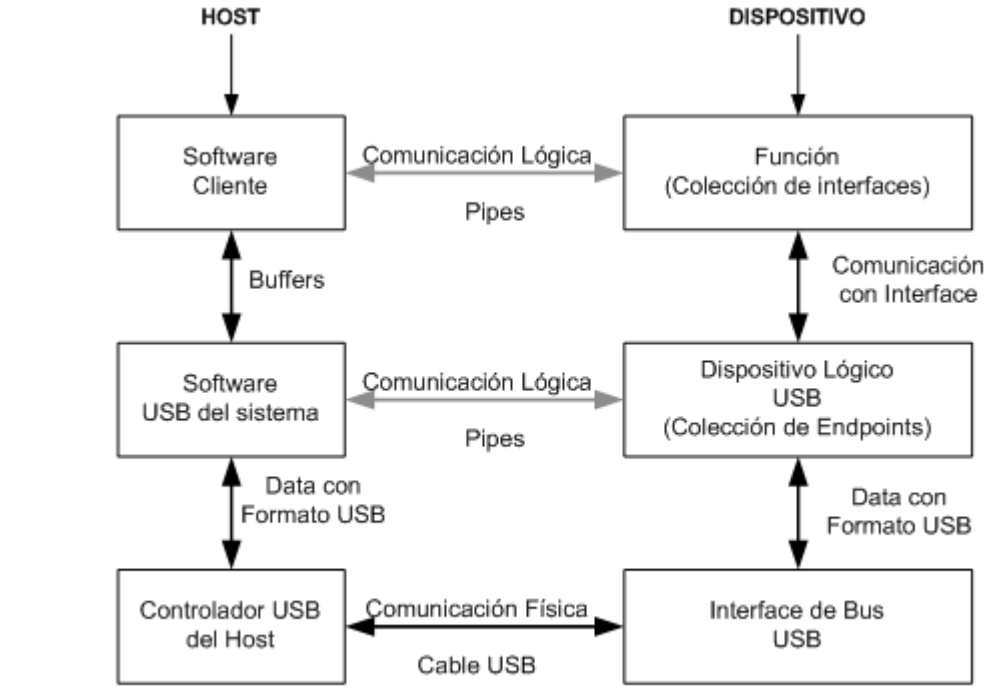



**Figura 2.6 Capas de flujo de datos entre el host y el dispositivo [9].**

### 2.2.2 Tipos de Transferencia USB

De acuerdo al propósito de la data a transmitir, su prioridad y ancho de banda asignado, la especificación USB 2.0 define cuatro tipo de transferencias: Control, Avalancha (en inglés Bulk), Interrupción e Isócrona. Cada transferencia determina las características de cómo se transmite la información [9]:

A. Transferencia de control: Generalmente usada para transferencia de comandos de configuración así como de estados del dispositivo. La entrega de la información es sin pérdidas. El endpoint 0 usa este tipo de transferencia.

B. Transferencia bulk (de avalancha): Estas transferencias pueden alcanzar mayores velocidades respecto a las otras transferencias pues ocupa todo el ancho de banda del bus si es que éste está disponible. Ésta es

usada para cuando se requiere transmitir una gran cantidad de datos. Si el bus está ocupado con otras transacciones, esta transferencia tiene que esperar.

C. Transferencia de interrupción: Destinado para dispositivos que requieren la atención de forma periódica. Por ejemplo, dispositivos como el teclado y mouse usan este tipo de transferencia.

D. Transferencia isócrona: En este caso la data es continua y en tiempo real, además asegura que la información llegue dentro de cierto tiempo límite; sin embargo, esta transferencia no garantiza la ausencia de errores en la recepción. Como ejemplos se tiene la transferencia de audio o video en tiempo real.

Con estos cuatro tipos de transferencias, número de bytes de los endpoints, y dirección de flujo de los datos, es que se va a definir las velocidades de transferencias máximas que se pueden alcanzar, qué será capaz de transmitir nuestro dispositivo, etc. Por ejemplo, para un dispositivo USB se pueden crear 7 endpoints repartidos de la siguiente manera:

- 1 de control
- 2 de transferencia Interrupción(1 IN y 1 OUT)
- 2 de transferencia Isócrona (1 IN y 1 OUT)
- 2 de transferencia Bulk (1 IN y 1 OUT)

### 2.2.3 Clases USB

La mayoría de dispositivos USB existentes comparten características con otros dispositivos USB. Por ejemplo, todos los teclados (independientemente de la marca) envían caracteres por el puerto USB; lo mismo para las impresoras. Por

ello, el USB-IF (del inglés *USB Implementers Forum*) definió las clases USB, donde se agrupan dispositivos que comparten ciertos atributos. Las principales clases son [1]:

A. CDC (del inglés *Communication Device Class*): Esta clase permite emular el comportamiento de los puertos RS-232, permitiendo crear un puerto COM virtual.

B. HID (del inglés *Human Interface Devices*): Esta clase básicamente abarca dispositivos que son usados por personas para controlar la operación de alguna parte del computador. Algunos ejemplos de dispositivos que pertenecen a esta clase son: teclado, mouse, joystick.

C. MSC (del inglés *Mass Storage Class*): Es la clase asociada con los dispositivos de almacenamiento. Algunos de los dispositivos que pertenecen a esta clase son: memorias USB, discos duros externos, reproductores de música, celulares, cámaras digitales, etc.

Estas clases se diferencian unas de otras, entre otras cosas, por tener endpoints con parámetros asociados ya establecidos. Por ejemplo, la clase HID (usada para teclados y ratones) posee un endpoint de control (todos los dispositivos independientemente de la clase tienen este endpoint) y un endpoint de interrupción. La clase MSC tiene 2 endpoints Bulk: 1 IN y 1 OUT. La clase CDC también posee 2 endpoints Bulk al igual que la MSC.

A pesar de que usan la transferencia tipo Bulk, las clases MSC y CDC presentan ciertas limitaciones. La clase CDC está limitada por la tasa de baudios “virtual” con la que debe comunicarse, la cual en la mayoría de los hosts es 115200 bps. Por otro lado, la clase MSC (usada en memorias de almacenamiento USB) transfiere la data usando usando comandos SCSI (Small Computer System Interface) lo cual hace que su tasa de transferencia de data útil disminuya.

Finalmente, se tiene la clase USB personalizada (del inglés *USB Vendor Class*), la cual permite modificar varios parámetros dentro del protocolo como por ejemplo: tipos de transferencia, número de endpoints, formato en que se entrega la data útil, etc. Dado que es más flexible en el sentido de que se puede modificar varios parámetros, es necesario también que el desarrollador trabaje con un driver propietario y desarrolle su propia aplicación para que el host pueda comunicarse con el dispositivo.

En el presente trabajo se usará este tipo de clase personalizada para aprovechar las tasas de transferencia USB mandando la data útil de manera "cruda" y que a través de una aplicación se pueda recibir esta data. También se analizará hasta qué velocidad se ha podido llegar con el microcontrolador que se va a usar, porque si bien la especificación USB indica un máximo teórico de 12 Mbps para la velocidad "Full Speed", este en realidad es la velocidad de cada bit, mas no la tasa de transferencia de datos útiles.

## 2.3 Objetivos

### 2.3.1 Objetivo General

Diseñar un sistema de adquisición de datos utilizando el protocolo USB en un microcontrolador AVR.

### 2.3.2 Objetivos Específicos

- Diseñar una clase USB personalizada usando como base el framework para USB que provee Atmel que permita transmitir datos así como responder a una serie de comandos personalizados.

- Desarrollo de una aplicación en lenguaje C que permita al sistema operativo comunicarse con el dispositivo USB para adquirir los datos.

- Buscar en todo momento obtener la máxima tasa de transferencia posible.

# CAPÍTULO 3

# DISEÑO DEL SISTEMA DE ADQUISICIÓN DE DATOS

## 3.1 Consideraciones de diseño

La clase USB desarrollada está pensada para obtener la máxima tasa de transferencia posible de datos útiles para que pueda ser utilizada en una aplicación de adquisición de datos. Así, se considerará como entrada al sistema un buffer que está siendo constantemente actualizado con datos muestreados. Cualquier paso previo involucrado en la adquisición de datos (atenuación, filtros, etc.) no forma parte del desarrollo de la presente tesis.

Como ya se mencionó previamente, se busca la tasa de transferencia más alta posible. De acuerdo al capítulo 9 de la documentación USB 2.0 [9], el tipo de transferencia que permite transmitir datos a alta velocidad de manera fiable (siempre y cuando el bus esté disponible) es la transferencia tipo Bulk.

El endpoint tipo Bulk va a ser el que por el que se transfiera los datos muestreados. Dado que se aspira a alcanzar la máxima tasa de transferencia posible, es necesario hace un cálculo de cuánto es la tasa de transferencia máxima a la que se puede llegar con este tipo de transferencia teóricamente. Para ello, se analiza la trama USB para saber la cantidad de bytes de data útil que quedan disponibles después de descontar las cabeceras del protocolo:

El tráfico del protocolo USB está dividido en tramas (en inglés “frames”) de 1ms tal como se muestra en la figura 3.1

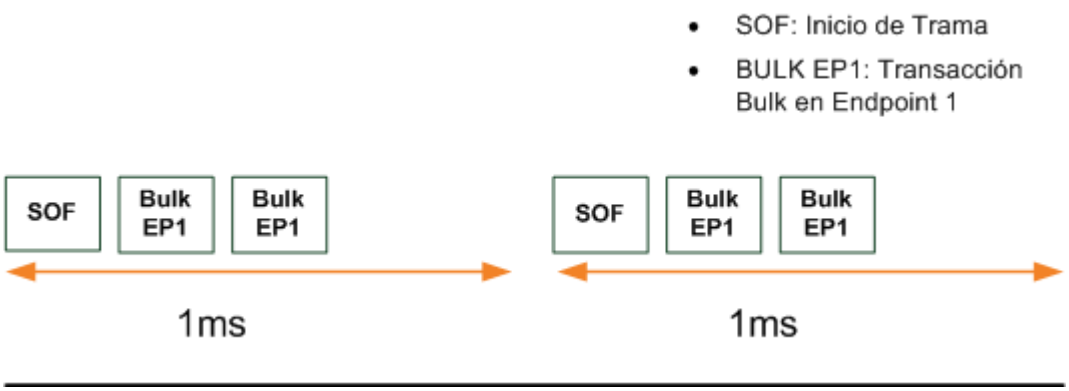



**Figura 3.1 Trama del protocolo USB.**

Donde cada transacción Bulk tiene el formato descrito por la figura 3.2:

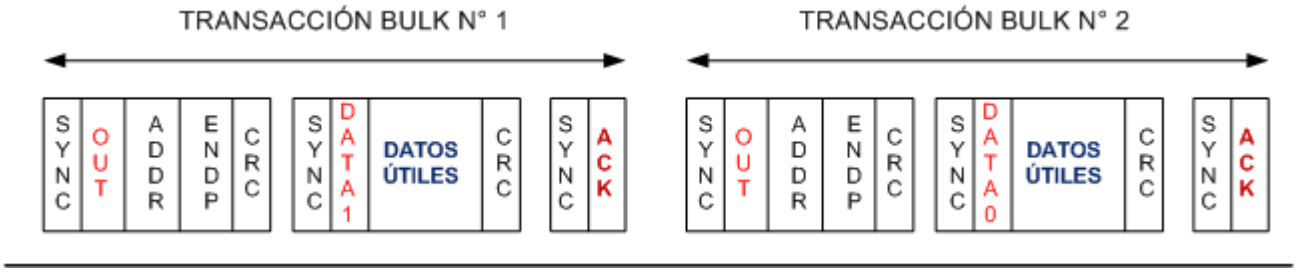



**Figura 3.2 Formato de transacciones tipo Bulk. En la figura, se muestran 2 transacciones.**

Según la especificación, el máximo número de bytes dentro de la sección de Data Útil (*Payload Data* en la figura 7) por transacción es de 64 bytes. Si a eso se le agrega los bytes de cabecera: SYNC (Sincronización), Dirección, CRC (chequeo de error), ACK (Reconocimiento) se obtiene un total de 77 bytes por transacción Bulk (64 de data útil y 13 de cabecera).

Por otro lado, el clock de la interface USB transmite bits a una tasa de 12Mbps o 1.5Mbytes/segundo. Entonces:

- Número de bytes posibles en una trama de 1ms: 1.5Mbytes/1000 = 1500 bytes.
- Número máximo de transacciones bulk: 1500/77 = 19 transacciones.
- Número máximo de bytes de data útil en una trama: 64*19 = 1216 bytes.

Por tanto, la máxima tasa de transferencia de data útil teóricamente es de: 1216 bytes/1 ms = **9.7 Mbps**.

### 3.2 Diagrama de Bloques del Sistema

La aplicación del presente trabajo consiste de un ADC que lee valores de señales de voltaje de 4 canales, estos valores son transferidos a la interfaz USB, y la aplicación USB se encarga de transferir los datos a la computadora (host) a través de los endpoints y pipes. La figura 3.3 muestra el diagrama de bloques del sistema.

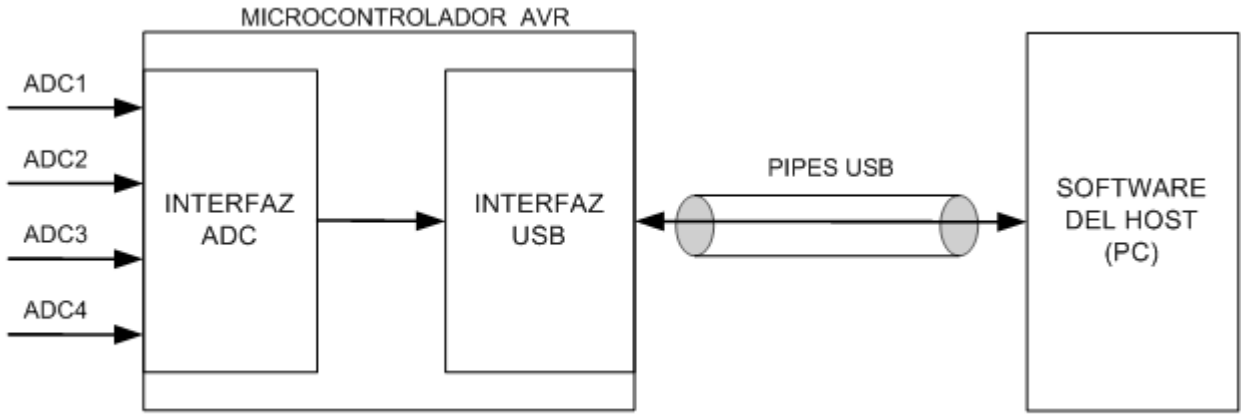



**Figura 3.3 Diagrama de Bloques del Sistema.**

Posteriormente, cuando se vea el diseño del protocolo personalizado para la comunicación con la computadora, se analizará a detalle cuántos canales de comunicación o “pipes” y qué tipo de endpoints usará la interfaz USB del microcontrolador.

## 3.3 Hardware del Sistema de Adquisición de Datos

La elección de un microcontrolador para esta aplicación está determinada por los siguientes criterios:

- Interfaz USB: Este requerimiento es indispensable, dado que se usa para transferir datos a la computadora.
- Frecuencia de muestreo (ADC)
- Resolución (ADC)
- Número de canales (ADC)
- Frecuencia de operación de CPU: Relacionado con la velocidad de procesamiento del microcontrolador.
- Costo

La tabla 3 muestra una comparación entre diferentes microcontroladores con interface USB.

**Tabla 3: Comparación de microcontroladores que cuentan con interfaz USB**

| Parámetro | ATxmega256A3BU | AT90USB1287 | AT90USB162 | ATmega32u2 | PIC18F4550 |
|---|---|---|---|---|---|
| **Frecuencia de operación** | 32MHz | 16MHz | 16MHz | 16MHz | 48MHz |
| **Frecuencia de muestreo(ADC)** | 2MSPS | 15kSPS | - | - | 200kSPS |
| **Resolución(ADC)** | 12 bits | 10 bits | - | - | 10 bits |
| **N° canales(ADC)** | 4 | 8 | - | - | 13 |
| **Costo** | $7.3 | $13.1 | $3.7 | $4.5 | $5.26 |

De la tabla se puede observar que el AT90USB162 y el ATmega32u2 no cuentan con ADC. Por otro lado, el AT90USB1287 sí cuenta con ADC interno, pero su frecuencia de muestreo es de 15kSPS y su resolución máxima es de 10 bits, por lo que se podría adquirir datos a una velocidad de:

15000 muestras/segundo * 10 bits/muestra = 150kbps

Por lo que no se aprovecharía la tasa de transferencia de la interfaz USB cuya tasa de transferencia es de 12Mbps. La misma limitación presenta el PIC18F4550. Sin embargo, se puede ver que el ATxmega256A3BU sí puede generar datos en el orden de los megabits por segundo además de tener un costo razonable con respecto a los otros, por lo que se elige este microcontrolador. El ADC del ATxmega256a3bu puede llegar a estas tasas de muestreo pues la tasa de muestreo es igual a la frecuencia del ADC. Esto es posible gracias a la forma de diseño de su ADC, que contiene un pipeline de 12 etapas de tal manera que en un ciclo puede acquirir una muestra, y en el siguiente ciclo puede generar el siguiente bit de la primera muestra y a la vez acquirir una segunda muestra.

El sistema de adquisición de datos está basado en la tarjeta de desarrollo XMEGA-A3BU Xplained (figura 3.4) la cual contiene el microcontrolador ATXMEGA256A3BU de Atmel.

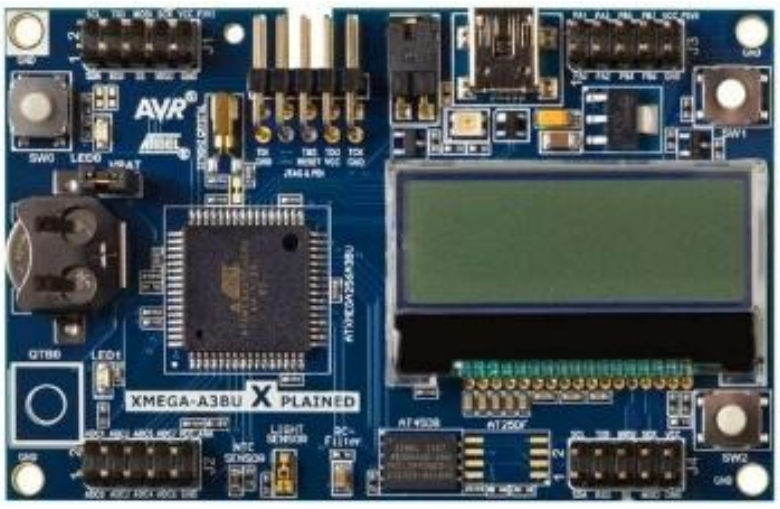



**Figura 3.4 Tarjeta de Evaluación XMEGA-A3BU Xplained [10].**

Por otro lado, la tabla 4 muestra las características de la interfaz USB obtenidas de la hoja de datos del microcontrolador ATXMEGA256A3BU:

**Tabla 4: Parámetros del módulo USB del ATXMEGA256A3BU.**

| | |
|---|---|
| Velocidad máxima: | 12Mbps |
| Número de endpoints: | 31 |
| Tamaño máximo de data útil por endpoint: | 32 bytes |
| Tipos de transferencia permitidos: | Control, Interrupción, Bulk, Isócrona |
| Clock requerido: | 48 MHz |

Si bien se eligió este microcontrolador, no quiere decir que no se puedan usar otros: Existen otros de la misma familia y de otras familias que cuentan con interfaz USB (AT90USB, AT32UC3). Esto es posible gracias al uso de un stack USB, el cual es un conjunto de librerías que permite abstraer las capas de hardware (registros específicos de cada microcontrolador) como la que genera la trama de paquetes USB (Packet Layer) para que el desarrollador pueda enfocarse en el desarrollo de las capas más altas del protocolo USB.

## 3.4 Diseño del Firmware

### 3.4.1 Arquitectura del Firmware

La figura 3.5 muestra la arquitectura del firmware en el microcontrolador.

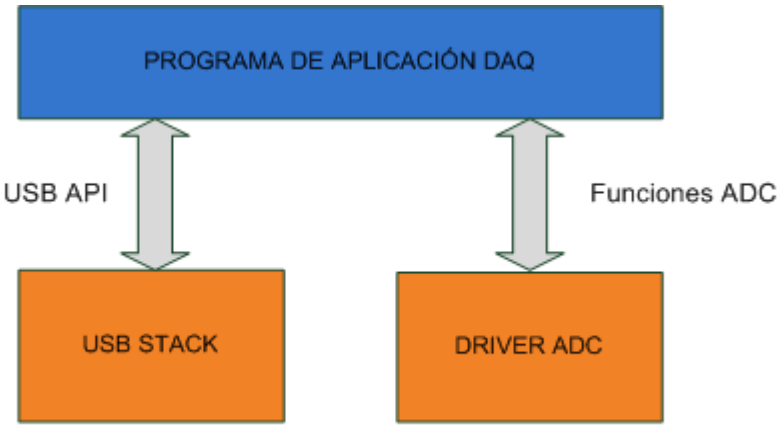



**Figura 3.5 Arquitectura del firmware en el microcontrolador AVR.**

Como ya se mencionó en la sección anterior, el stack USB va a permitir abstraer las capas más bajas del protocolo USB 2.0. Algunas modificaciones que se le va a hacer al stack USB son:

- Número de endpoints a usar, dirección y tipo de transferencia en cada uno.
- Indicar si el dispositivo soporta “Low Speed”, “Full Speed” o “High Speed”.
- En la capa más alta (“Application Layer”), definir el protocolo de comunicación que se va a dar con el host a través de los endpoints. En la sección 3.4.3 se dará una explicación más detallada al respecto.

El programa de aplicación principal va a poder inicializar y configurar la interfaz USB a través de un API (del inglés *Application Programming Interface*) tal como lo muestra la figura 13. El USB API es un conjunto de funciones que permite a la aplicación principal poder comunicarse con el stack USB.

El driver del ADC por otro lado va a permitir configurar y controlar el módulo ADC desde la aplicación principal.

### 3.4.2 Elección del Stack USB

Tal como se vio en la figura 13, el sistema necesita de un stack USB que gestione las capas más bajas del protocolo USB. Los siguientes stacks que se describen a continuación han sido diseñados exclusivamente para los microcontroladores AVR que poseen un controlador USB dentro del chip:

- FreakUSB Stack: Firmware simple de código abierto para ciertos microcontroladores AVR. Posee implementada sólo la clase CDC (del inglés *Communication Device Class*) [11].

- LUFA Stack: Esta solución puede ser implementada en cualquier dispositivo AVR que incorpore un controlador USB [12].

- AVR USB Stack: Tal como su nombre lo indica, es el que provee Atmel para la línea de microcontroladores AVR que soportan USB [13].

Para el presente trabajo se optó por trabajar con el AVR USB Stack debido a la facilidad de uso del API que provee Atmel así como el soporte para clases USB personalizadas.

### 3.4.3 Configuración y Selección de Endpoints

Si bien en las secciones anteriores ya se ha definido qué stack USB se va a usar y cómo va a interactuar con el programa principal, aún falta definir ciertos parámetros en el protocolo USB que permitan la comunicación con la computadora:

- Número de endpoints.
- Dirección de cada endpoint. Puede ser IN: de dispositivo a host; o OUT: de host a dispositivo.
- Tipo de transferencia de cada endpoint. Como ya se mencionó en el capítulo 2, existen cuatro tipos de transferencias: control, bulk, interrupción e isócrona.

Dado que bulk es la que permite un alto ancha de banda y garantiza que no va a haber pérdidas en la transmisión, la transferencia tipo bulk es la opción más adecuada para el endpoint que se encargue de adquirir los datos del ADC.
Si bien la transferencia tipo Bulk va a servir para leer los datos que se esté transfiriendo, se necesita otro tipo de transferencia que sirva para que el host mande órdenes de control al dispositivo como por ejemplo: empezar y/o detener transmisión, cambiar de canal, etc. Para este tipo de órdenes no se

necesita un alto ancho de banda, además de que no van a ocurrir de manera continua sino esporádica. El tipo de transferencia que se adecua a este tipo de solicitudes es la de tipo Interrupción.  Asimismo, es necesario también que por cada comando o solicitud del host, el dispositivo USB envíe un dato de respuesta o status. Para este último se utilizará un endpoint de tipo interrupción.

Finalmente, todos los dispositivos USB independientemente de su aplicación, cuentan con un enpoint de control, el cual sirve para la configuración inicial del dispositivo. La figura 3.6 muestra la dirección y tipo de pipes, los cuales usan los endpoints descritos anteriormente.

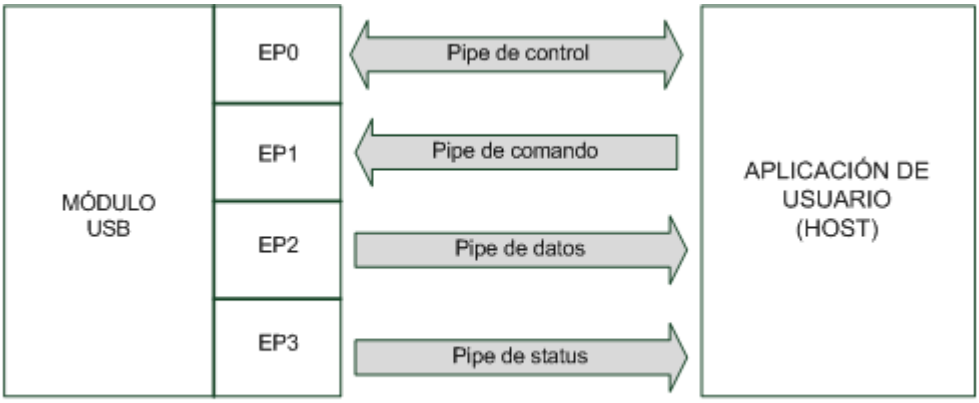



**Figura 3.6 Transmisión de datos del sistema DAQ a la computadora a través de endpoints.**

### 3.4.4 Protocolo personalizado para la capa de aplicación del stack USB

Es necesaria la implementación de un protocolo de comunicación entre host y el dispositivo USB usando los diferentes endpoints de la interfaz USB para que no se mezclen comandos, datos y respuesta de estatus. La comunicación entre el host y el dispositivo va a consistir de dos etapas:

- En la primera etapa, el host envía un comando al dispositivo.
- En la seguna etapa, el dispositivo envía una serie de bytes que pueden ser datos en respuesta a la solicitud del comando de la primera etapa, o un byte de estatus.

La tabla 5 y 6 muestran el formato usado para los comandos que envían el host así como el formato de respuesta que envía el microcontrolador. Notar que el número de parámetros depende de cada comando, puede haber comandos que no requieran parámetros.

**Tabla 5: Formato de los comandos del host (Endpoint 1)**

| Byte | Descripción |
|---|---|
| 1 | Comando |
| 2 | Parámetro 1 |
| 3 | Parámetro 2 |
| 4 | Parámetro 3 |
| ⋮ | ⋮ |

**Tabla 6: Formato de respuesta del dispositivo (Endpoint 3)**

| Byte | Descripción |
|---|---|
| 1 | Comando al que se responde |
| 2 | Parámetro 1 |
| 3 | Parámetro 2 |
| 4 | Parámetro 3 |
| ⋮ | ⋮ |

Los comandos implementados para el ADC son: Empezar lectura, detener lectura, configurar ADC, y leer su configuración actual (número de canales, frecuencia de muestreo, resolución, etc). Los comandos se encuentran en la sección “Anexos” de forma más detallada.

### 3.4.5 Diagramas de Flujo del Programa en el Microcontrolador

El programa principal consiste básicamente en capturar datos del ADC, y esos valores transferirlos por el bus USB hacia el host usando la transferencia tipo Bulk. Asimismo, la computadora en cualquier momento puede mandar comandos de control tales como: detener lectura de datos, cambiar de canal, etc. Para este tipo de solicitudes al dispositivo USB, se hace uso de la transferencia tipo Interrupción. El programa desarrollado en el microcontrolador se puede dividir en las siguientes partes:

- Aplicación principal.
- Función de la interfaz USB cada vez que hay una transferencia, ya sea de escritura o lectura.
- Función del ADC cada vez que se hace una lectura.

En las siguientes secciones se muestran y explican los diagramas de flujo por separado de cada una de estas partes.

#### 3.4.5.1 Aplicación Principal

El diagrama de flujo para la aplicación principal se muestra en la figura 3.7. Lo que se hace en primer lugar es inicializar el sistema, lo cual consite en configurar los parámetros iniciales como:

- clock del sistema
- puertos del microcontrolador como entradas o salidas
- Periféricos tales como Timer, ADC y USB.

La aplicación posteriormente se queda esperando hasta que el dispositivo USB esté enumerado, lo cual sucede cuando el host ha detectado un dispositivo conectado, le ha asignado un driver y el dispositivo ya está listo para usarse.

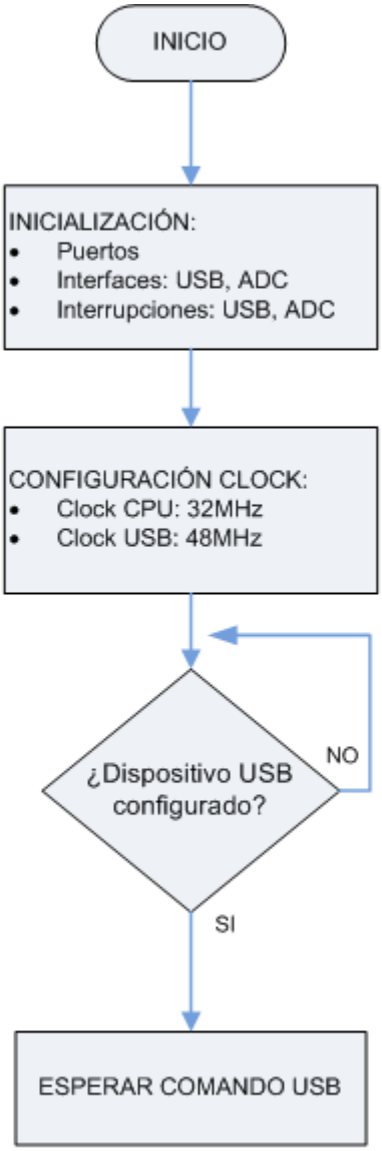



**Figura 3.7 Diagrama de Flujo del Programa Principal para transferir datos por USB.**

### 3.4.5.3 Función del ADC para lectura de datos

En la figura 3.8 se muestra el diagrama de flujo cada vez que se entra a la interrupción del ADC. Si la acquisición de datos ha sido iniciada por un comando USB, entonces el ADC está operativo en modo carrera libre (del inglés *Free Running*).

Cada vez que el programa entra a la interrupción verifica si el buffer del ADC de 128 bytes está lleno; si no está lleno entonces se almacena la lectura del ADC en el buffer, se incrementa el contador que representa el número de lecturas de ADC en el arreglo y el programa sale de la interrupción. Si el buffer ya está lleno, primero se guarda el resultado de la conversión en la segunda mitad del arreglo Buffer_ADC; es decir, que se graba en la posición 64 del arreglo (recordar que su tamaño es de 128 bytes). Seguidamente se llama a la función USB que transmite los bytes del buffer en las posiciones 0-63. Por último se retorna de la interrupción.

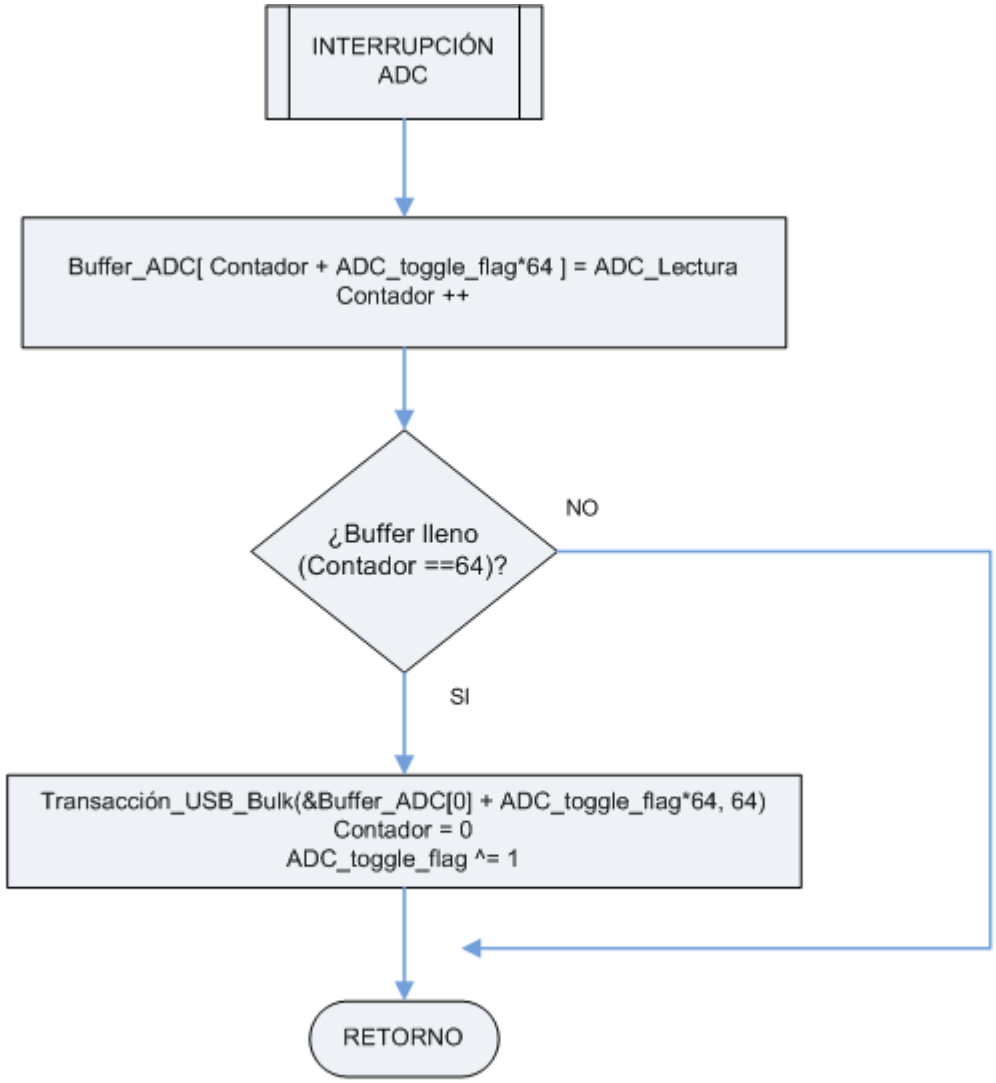



**Figura 3.8 Diagrama de flujo del programa de interrupción del ADC**

### 3.4.5.2 Función de la interfaz USB

En la figura 3.9 se muestra el diagrama de flujo cada vez que hay una transferencia USB. Al inicio, el programa identifica si el tipo de transferencia es tipo Bulk o Interrupción.

Si la transferencia es de tipo Bulk, significa que se ha terminado de transferir los 64 bytes previamente enviados del buffer llamado Buffer_ADC.

Si la transferencia es de tipo Interrupción, significa que se ha recibido un comando enviado por la computadora, por lo que se lee este comando y se ejecuta la acción correspondiente, la cual puede ser:

- Empezar lecturas en el ADC
- Terminar lecturas en el ADC
- Leer estatus del dispositivo USB

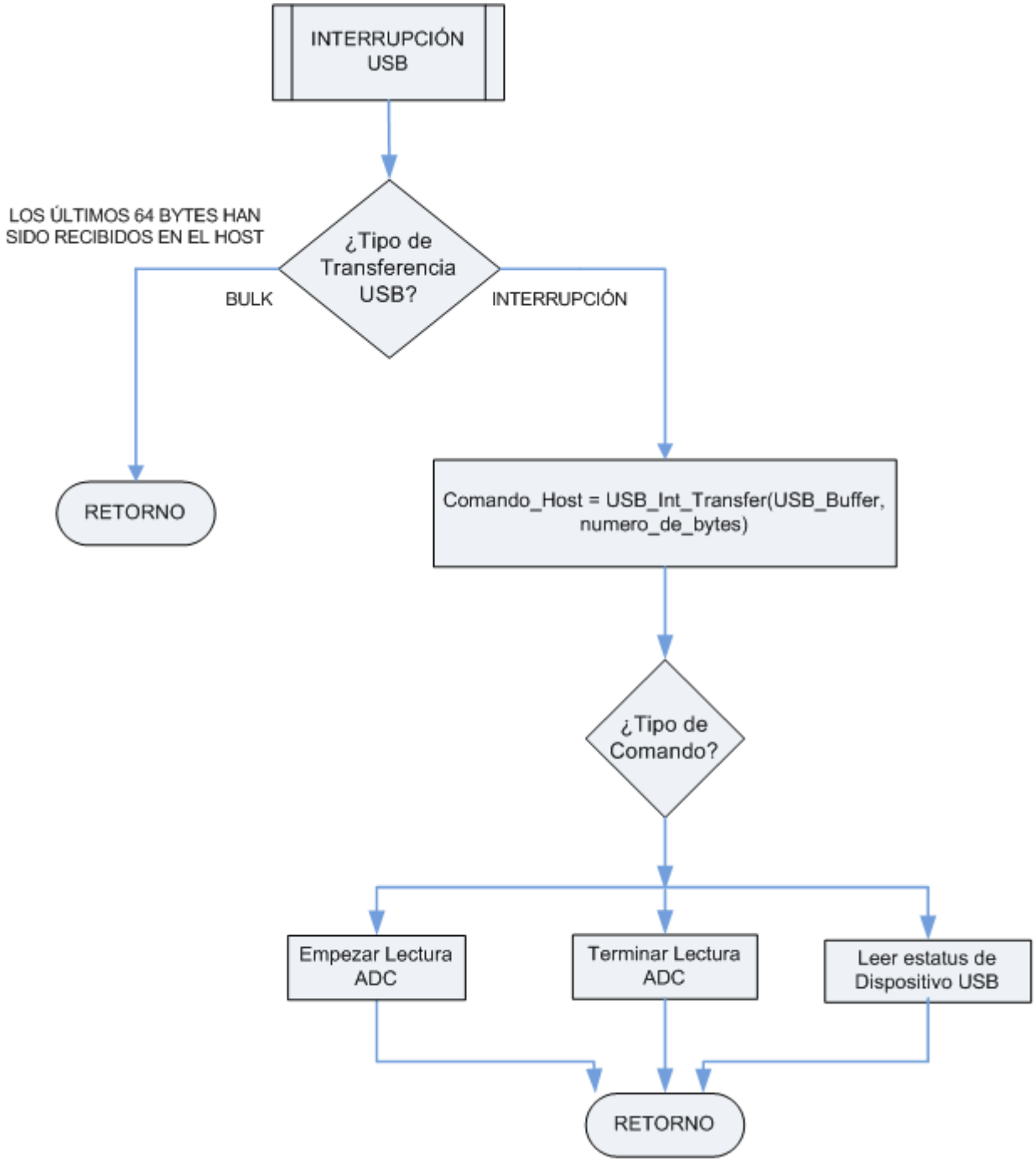



**Figura 3.9 Diagrama de flujo del programa que gestiona las transferencias USB**

## 3.5 Desarrollo del Software de Aplicación

### 3.5.1 Gestión del dispositivo USB con Libusb

Dado que se está diseñando una clase personalizada de USB, Windows solicitará un driver para el dispositivo al momento de conectarlo. Para ello, se utilizará libusb, la cual es una librería hecha en C que permite trabajar con dispositivos USB en distintos sistemas operativos.

Cuando se conecta el dispositvo por primera vez, Windows abrirá una ventana que solicita el driver. Para ello, libusb provee el archivo libusb.sys. Sin embargo, el archivo ".sys" que es el driver en binario por sí solo no es suficiente, sino que además se tiene que crear un archivo INF, el cual describe al sistema operativo el driver y el dispositivo que controla.

Para poder trabajar con el dispositivo USB desde la librería Libusb se hace uso de dos estructuras en lenguaje C [14]:

- **usb_device**: Representa un dispositivo USB que está o estaba conectado al sistema. Usando esta estructura como referencia, se puede determinar cierta información del dispositivo (número de versión, nombre del fabricante, etc). Notar que tener esta referencia no necesariamente significa que el dispositivo sea usable en cualquier momento: puede que se haya desconectado o que algún programa o driver ya esté usando el dispositivo.

- **usb_dev_handle**: Con esta estructura se puede apuntar al dispositivo con el que se quiere operar en algún momento dado. Todas las operaciones se deben hacer con esta estructura pues, a diferencia de la estructura anterior, asegura que se tiene el control del dispositivo en un momento dado.

La tabla 7 describe los argumentos de las funciones usados en el diagrama de flujo del software de la aplicación (figura 15).

**Tabla 7: Parámetros usados en el software de aplicación.**

| Parámetro | Tipo | Función |
|---|---|---|
| **Dev** | estructura usb_dev_handle | Apunta al dispositivo con el que se quiere operar. |
| **VID (Vendor ID)** | uint8_t | Sirve para identificar el dispositivo que se quiere encontrar. |
| **PID(Product ID)** | uint8_t | Permite identificar el dispositivo que se quiere encontrar. |
| **Dirección_Endpoint_Interrupción** | uint8_t | Número que identifica al endpoint INT. |
| **Dirección_Endpoint_Bulk** | uint8_t | Número que identifica al endpoint BULK. |
| **Buffer_Tx** | Arreglo tipo uint8_t | Arreglo con datos para enviar al dispositivo USB. |
| **Buffer_Rx** | Arreglo tipo uint8_t | Arreglo con datos recibidos del dispositivo USB. |
| **Adc_Flag** | Booleano | Determina si se ha empezado el proceso de transferencia de datos hacia la computadora. |
| **Timeout** | uint8_t | Tiempo (en ms) de espera para realizar operación de escritura. |

### 3.5.2 Diagrama de flujo del software de aplicación

La figura 3.10 muestra el diagrama de flujo del software de aplicación hecho en C usando la librería Libusb.

Como se puede observar, primero se inicializa la librería libusb, seguidamente se busca si algún dispositivo conectado al bus USB coincide con el VID y el PID de nuestro dispositivo. Si se encontró, entonces se inicializa el dispositivo USB usando como argumento la variable *dev* que es un puntero a una estructura del tipo *usb_dev_handle*.

Seguidamente, el programa va  a mostrar en pantalla los comandos disponibles para interactuar con el dispositivo USB, los cuales pueden ser cuatro:

- Empezar transferencia: Envía el comando para habilitar el ADC a través del pipe de Interrupción de salida, abre el pipe tipo Buk para la recepción de lecturas del ADC y pone en 1 la variable ADC_flag para poder almacenar los datos entrantes en el buffer de recepción.

- Terminar transferencia: Envía el comando para deshabilitar el ADC a través del pipe de Interrupción de salida y finaliza el programa.

- Leer estatus: Envía el comando para el estatus del dispositivo USB por el pipe de Interrupción de salida y abre el pipe de Interrupción de entrada para recibir el byte de estatus que el dispositivo reciba.

- Configurar ADC: Envía el comando de configuración del ADC a través del pipe de Interrupción de salida, escribe en *Buffer_Tx* la configuración que desea en cuanto a número de canales, frecuencia de muestreo, etc; y transmite esta configuración también por el pipe de Interrupción de salida.

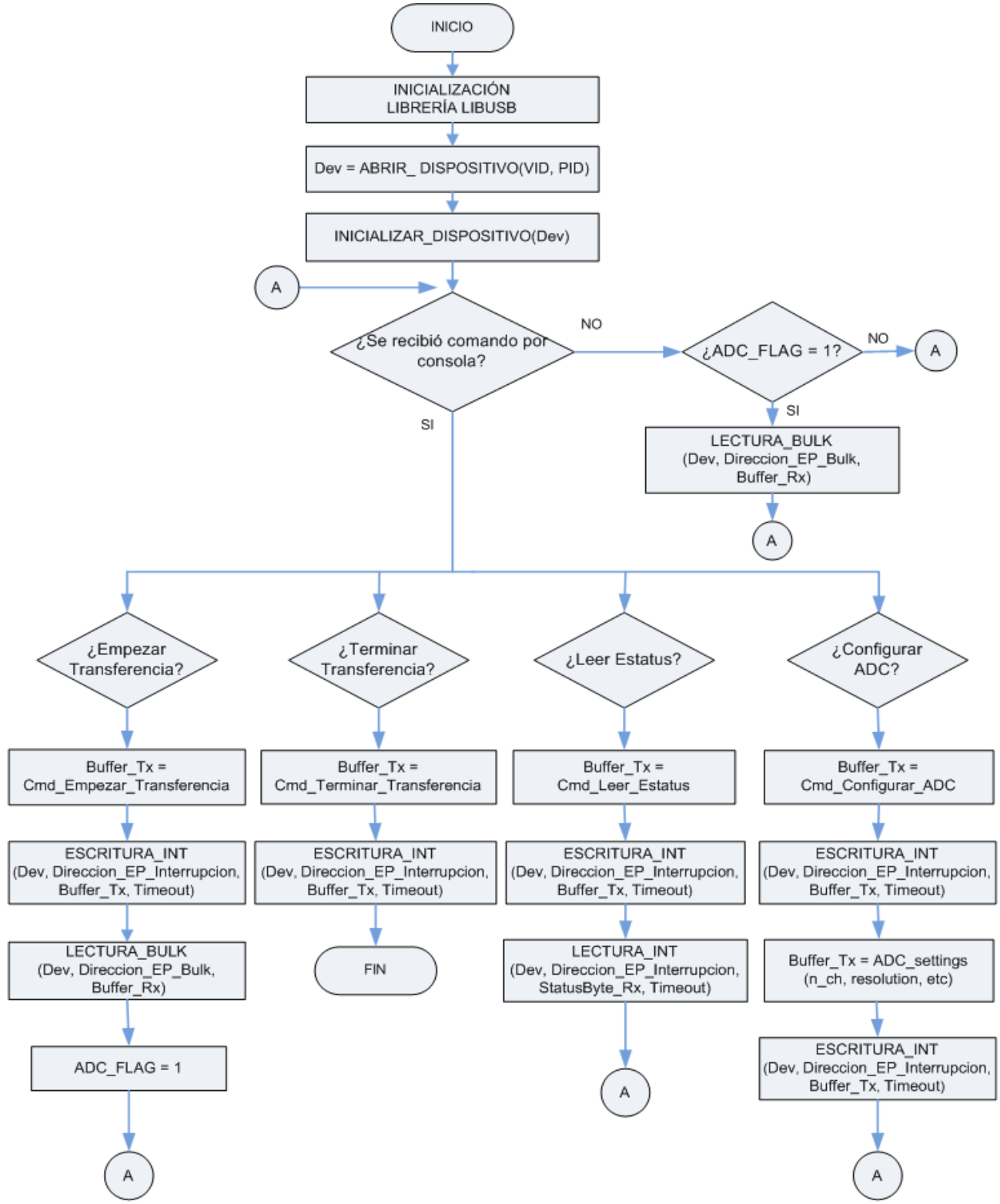



**Figura 3.10 Diagrama de flujo del programa de aplicación en el host.**

# CAPÍTULO 4

# RESULTADOS

## 4.1 Consideraciones de Implementación

**Plataforma de implementación:** Se utilizará la tarjeta de evaluación XMEGA – A3BU XPLAINED de Atmel. El microcontrolador de esta tarjeta es el ATXMEGA256A3BU.

**Herramientas de desarrollo:** El entorno de desarrollo Atmel Studio 6.0 y el compilador AVR-GCC 4.6.2. La versión del Framework que provee Atmel para trabajar con interfaces USB es el ASF 3.1.3.
Para los procesos de programación y depuración se hizo uso de la interfaz JTAG a través de la herramienta JTAGICE3 de Atmel. El lenguaje usado para la implementación es C.

## 4.2 Descripción de la Implementación

Con el objetivo de obtener las tasas de transferencia de la clase USB desarrollada en el presente trabajo se hizo uso del Timer 0 de 16 bits que posee el microcontrolador. Así, el timer empieza la cuenta cuando se llama a la función que empieza la transferencia por USB hacia la computadora. El host notificará al dispositivo USB de que se terminó la transmisión de datos a través de un paquete ACK (del inglés *Acknowledgement*) llamando al vector de interrupción USB_TRNCOMPL, momento en el cual el valor del registro de cuenta del timer es leído y visualizado en la pantalla LCD de la tarjeta de desarrollo.

Conociendo el número de bytes transmitidos y el tiempo que demora la transmisión a través del número de ciclos del timer, se puede calcular la tasa de transferencia de data útil. La frecuencia de cuenta del Timer 0 es de 12MHz.

## 4.3 Detección del Dispositivo USB

La primera vez que se conecta la tarjeta de desarrollo al host, el sistema operativo muestra una ventana que indica que se ha detectado un nuevo dispositivo. Después de haber generado el correspondiente archivo .INF e instalar el driver (libusb-win32), el dispositivo estará listo para comunicarse con la computadora tal como lo muestra la figura 4.1.

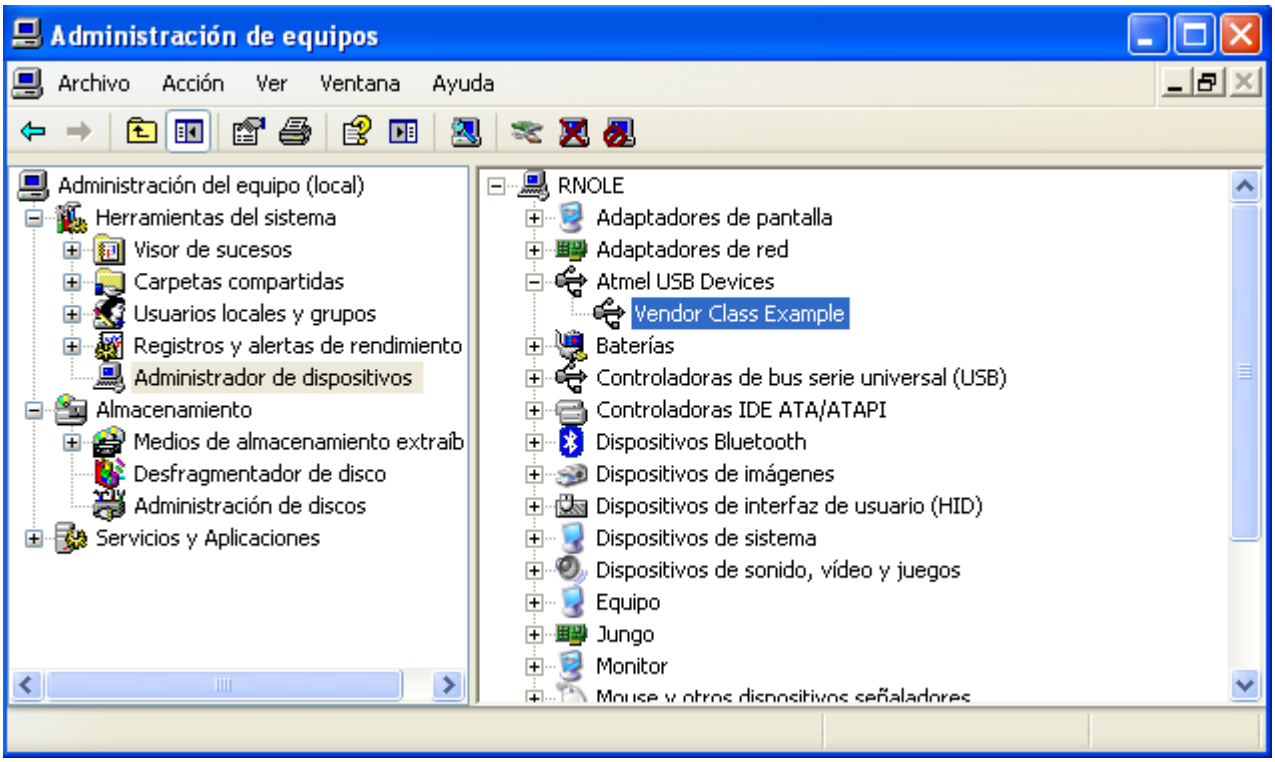



**Figura 4.1 Detección del nuevo dispositivo USB.**

## 4.4 Software para la interacción con el dispositivo USB

El presente software, desarrollado en C, permite la interacción del usuario desde la PC a través de línea de comandos ya sea para almacenar datos o configurar el ADC. La figura 4.2 muestra la ejecución del programa. Como se puede observar, el programa inicialmente muestra información del dispositivo USB con el que se va a trabajar; después, a través del teclado, se puede empezar la lectura de datos o configurar el ADC.

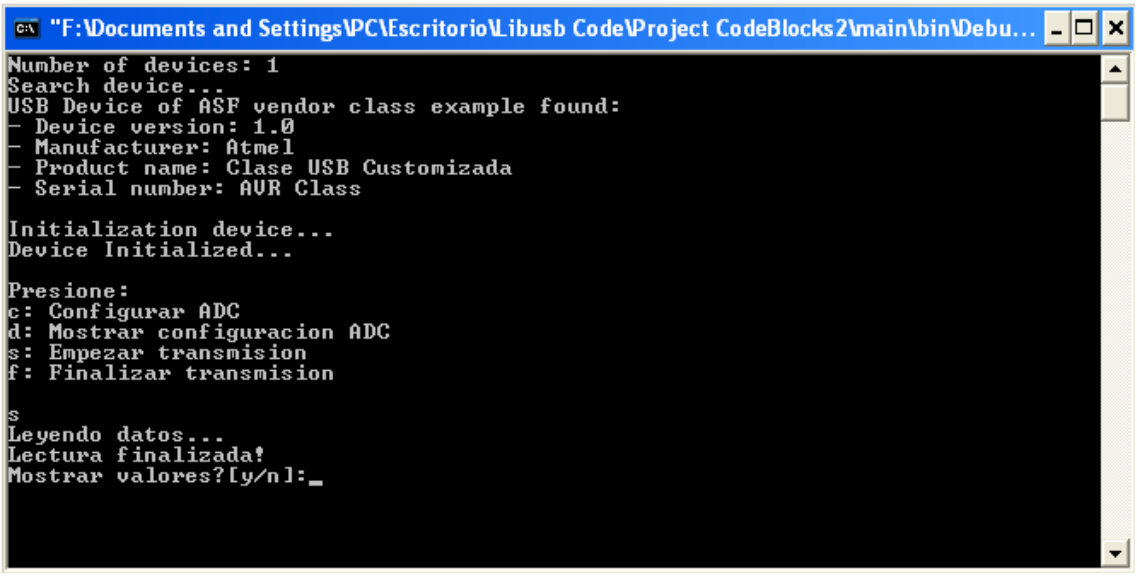



**Figura 4.2 Ejecución del programa de aplicación usando Libusb. Se puede observar información del dispositivo USB conectado así como la lista de comandos disponibles. En este caso, se ejecutó el comando para empezar la lectura de datos del ADC.**

Cuando la operación termina, el usuario tiene la opción de ver los valores que han sido guardados en un archivo en formato CSV.

## 4.5 Análisis de Tasas de Transferencia

Por el lado del host, se ejecuta el programa en C en la computadora para empezar la transmisión de datos. Por el lado del dispositivo USB, se programa el microcontrolador con el archivo .hex usando para el firmware distintos niveles de optimización del compilador (O0, O1, O2, O3).
El número de cuentas del temporizador aparecerá en la pantalla LCD. La tabla 8 muestra las tasas de transferencia obtenidas. Estos son valores promedio de las pruebas que se hicieron transfiriendo datos de un buffer de 64 bytes cien veces.

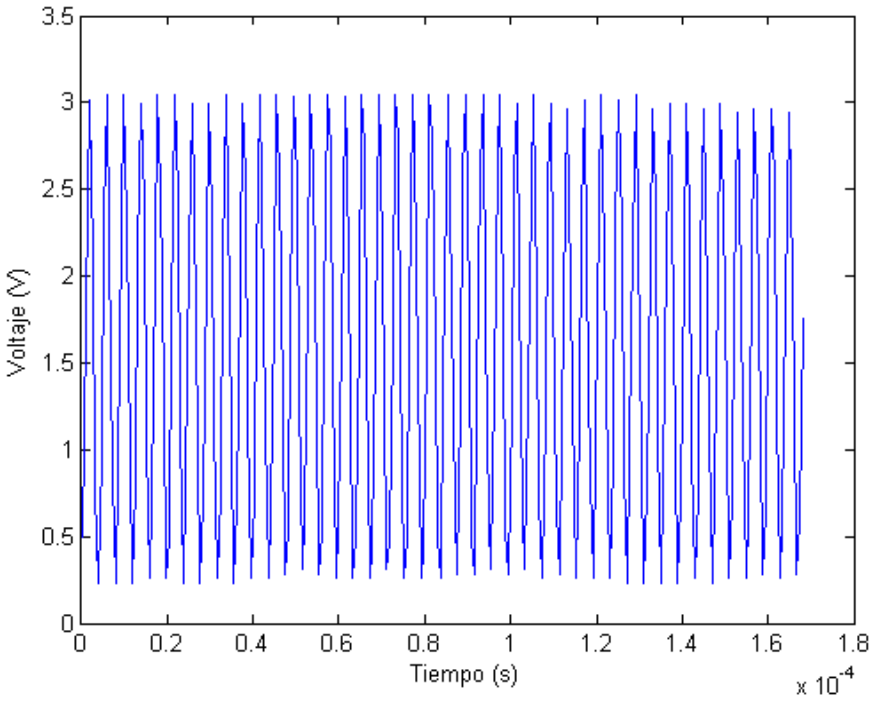
**Tabla 8: Tasas de transferencias obtenidas de la interfaz USB.**

| | Ciclos de Timer 0 | Tiempo(us) | Tasa de transferencia (Mbps) |
|---|---|---|---|
| **Compilador no optimizado(O0)** | 884 | 71.14 | 7.19 |
| **Compilador optimizado(O3)** | 725 | 60.45 | 8.46 |

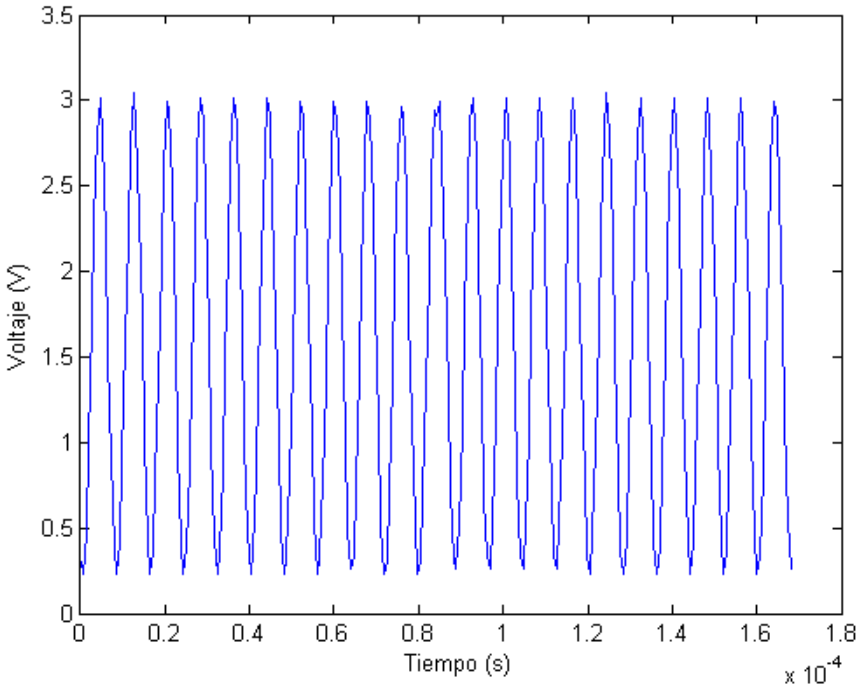
En la tabla se puede ver que hay una diferencia de velocidades de transferencia cuando el código es optimizado con el compilador. Lo cual indica que la optimización del código que gestiona el protocolo USB es un factor importante para obtener mayores velocidades de transferencia.

Asimismo, dado que es posible alcanzar velocidades de 8Mbps, el ADC puede adquirir datos a una frecuencia de muestreo de 1MSPS y 8 bits de resolución en modo free-running (carrera libre). Notar que dado que se está usando solo un canal para la prueba, la tasa de muestreo efectiva para este canal es de 1MSPS. Sin embargo, si se usaran por ejemplo los 4 canales disponibles, la tasa de muestreo efectiva para cada canal sería: 1MSPS/4 = 250kSPS.

La figura 4.3.a muestra una señal sinusoidal de 250kHz obtenida de un generador de señales muestreada a 1MSPS y 8 bits de resolución. Se puede apreciar que la señal reconstruida no es perfecta, pero esto es debido a la frecuencia de muestreo (sólo 4 veces la frecuencia de la señal). Si se quisiera obtener mejores resultados, sería recomendable que la frecuencia de muestreo sea 8 veces la frecuencia de la señal, tal es el caso de la figura 4.3.b donde la señal es de 125kHz.




**(a)**




**(b)**

**Figura 4.3 Señales muestreadas a 1MSPS: (a) Señal de 250kHz, (b) Señal de 125kHz.**

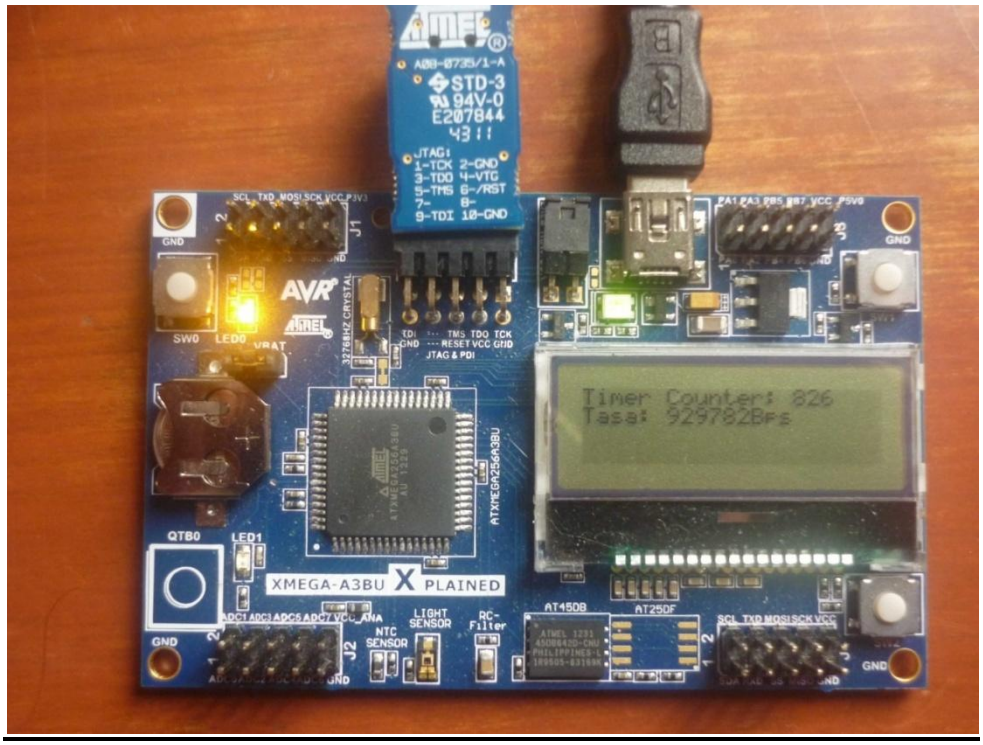



**Figura 4.4 Transmisión de datos usando la Tarjeta de Evaluación XMEGA-A3BU Xplained. En el LCD se observa el número de cuentas del Timer 0 por cada trasferencia así como la tasa de transmisión equivalente en bytes por segundo. El LED amarillo a la izquierda indica que se está transfiriendo datos.**

En el presente capítulo, las pruebas mostradas han permitido validar el funcionamiento tanto del hardware como el software del sistema de acquisición de datos. Asimismo, las tasas de transferencia obtenidas muestran que no fue posible llegar al máximo teórico que nos permite la transferencia tipo Bulk (9.7 Mbps), sino que en promedio se obtuvo 8.46Mbps.

# CONCLUSIONES

En esta tesis se ha desarrollado un sistema de acquisición de datos como una alternativa de bajo costo a sistemas comerciales que usan controladores USB. El desarrollo de la clase USB personalizada ha permitido un mayor control de cómo se transfieren los datos y aprovechar las tasas de transferencia que ofrece la transferencia tipo Bulk. Asimismo, las pruebas realizadas en el capítulo 4 han permitido validar el funcionamiento del sistema. Se puede concluir que:

1. La clase USB desarrollada para el sistema de adquisición de datos en el presente trabajo hace uso de transferencias tipo Bulk e Interrupción, obteniendo una tasa de transferencia promedio de 8.46Mbps cuando se usa un alto nivel de optimización en el compilador (O3).

2. Se logró el objetivo de que una aplicación desarrollada en lenguaje C en la computadora pueda comunicarse con el dispositivo USB. Si se quiere hacer uso de esta aplicación en cualquier computadora que use Windows, se debe contar con el archivo INF generado en el presente trabajo. Por otro lado, trabajos posteriores pueden enfocarse en la programación de una interfaz gráfica de usuario que permita mejor interacción con el usuario.

3. Si se quisiera que el microcontrolador se dedique a otras tareas aparte de acquirir datos, se debería implementar la transferencia de datos del ADC al USB a través de un DMA, pues en el sistema propuesto el uso de interrupciones requiere una mayor intervención por parte del CPU.

4. Si se compara el único trabajo anterior relacionado con USB desarrollado en la Pontificia Universidad Católica del Perú [15], se afirma que el presente trabajo ofrece una mejora en términos de tasas de transferencia obtenidas. El trabajo previo realizado en la referencia [15], al trabajar con la clase USB HID y con la especificación USB 1.1, puede alcanzar teóricamente una tasa máxima de 384Kbps.

## RECOMENDACIONES

1. De los resultados obtenidos se puede observar que no se ha alcanzado la máxima velocidad que las transferencias tipo Bulk pueden alcanzar. Sería ideal hacer esta implementación en microcontroladores que trabajan a mayores frecuencias como los de 32 bits para obtener mejores tasas de transferencia y a la vez tener mayor número de instrucciones de procesamiento disponibles por cada byte.

2. Si se quisiera obtener mayor velocidad por bit que 12Mbps y seguir trabajando con la especificación USB 2.0, se sugiere hacer la implementación en microcontroladores que soporten el modo “Alta Velocidad” (del inglés *High Speed*) del USB 2.0, cuya velocidad por bit es de 480Mbps.

3. Para un trabajo posterior desarrollar de las etapas de preprocesamiento de las señales: amplificación, filtrado, e aislamiento de las señales que se quieran enviar por la interfaz USB.

## BIBLIOGRAFÍA


[1] AXELSON, Jan.
2005 *USB Complete: Everything you need to know to develop USB peripherals*. Tercera edición. Madison: Lakeview Research.

[2] 1394 TRADE ASSOCIATION. *What is Firewire?.*
Fecha de consulta: setiembre de 2013
<http://www.1394ta.org/consumers/WhatIsFireWire.html>

[3] BUDRUK, Ravi; ANDERSON, Don y SHANLEY, Tom.
2003 *PCI Express System Architecture*. Primera edición. Massachusetts: Addison Wesley.

[4] AXELSON, Jan.
2003 *Embedded Ethernet and Internet Complete*. Primera edición. Madison: Lakeview Research, 2003.

[5] NATIONAL INSTRUMENTS. *Low-Cost, Bus-Powered Multifunction DAQ for USB.* Fecha de consulta: Julio de 2013.
< http://sine.ni.com/ds/app/doc/p/id/ds-218/lang/es >

[6] DATA TRANSLATION. *Low-Cost USB DAQ Modules.*
Fecha de consulta: Julio de 2013
<http://www.datatranslation.com/products/dataacquisition/usb/low-cost/>

[7] DATAQ INSTRUMENTS. *Ethernet Data Acquisition Products for any Application and Budget.* Fecha de consulta: Julio de 2013
<http://www.dataq.com/c_en/index.htm>

[8] DATA TRANSLATION. *DT300 Series.* Fecha de consulta: Julio de 2013
<http://www.datatranslation.com/products/dataacquisition/pci/DT300series/>

[9] USB IMPLEMENTERS FORUM.
2000 *Universal Serial Bus Specification*. Consulta: Julio de 2013. <www.usb.org/developers/docs/usb_20.pdf>

[10] ATMEL CORPORATION.
2012 *Atmel AVR1923: XMEGA – A3BU Xplained Hardware User Guide*. Fecha de consulta: Abril de 2013. <http://www.atmel.com/Images/doc8394.pdf>

[11] WANG, Christopher.
2009 *FreakUSB Open Source USB Device Stack*. Fecha de consulta: Abril de 2013. <http://www.freaklabs.org/freakusb/html/index.html>

[12] CAMERA, Dean.
2012 *LUFA (Lightweight USB Framework for AVRs) Library.* Fecha de consulta: Abril de 2013 <http://www.fourwalledcubicle.com/files/LUFA/Doc/120219/html/index.html>

[13] ATMEL CORPORATION.
2012 *Atmel AVR4030: Atmel Software Framework – Reference Manual*. Fecha de consulta: Marzo de 2013. <http://www.atmel.com/Images/doc8432.pdf >

[14] ERDFELT, Johannes.
2012 *Libusb Developers Guide*. Consulta: Agosto de 2013 <http://libusb.sourceforge.net/doc>

[15] CANSAYA HERRERA, Him Cuper.
2005 *Desarrollo de una interfaz USB para el control de estaciones de radio HF y VHF para comunicación de datos*. Tesis de licenciatura en Ciencias e Ingeniería con mención en Ingeniería Electrónica. Lima: Pontificia Universidad Católica del Perú, Facultad de Ciencias e Ingeniería.