Capítulo 2

Generalidades

En este capítulo se expondrán los conceptos teóricos que fueron necesarios para realizar este proyecto, si bien no se abarcaran todos y los que se mencionen no se harán con la suficiente profundidad que estos ameritan, servirán como una guía para una mejor comprensión de los capítulos de diseño.

Los temas abarcados se enfocan en el diseño del hardware, aunque si bien al final se hace mención a ciertas generalidades acerca del lenguaje C#, el cual fue utilizado en el diseño del programa de cómputo.

2.1. Adquisición de señales analógicas

Los temas que se han seleccionado intentan abarcar todo el proceso de adquisición de señales analógicas. Estos temas abarcan desde la teoría de muestreo, el diseño del acondicionamiento y el funcionamiento del convertidor analógico digital, este último permite la representación de las señales analógicas de manera binaria.

2.1.1. Muestreo de señales

En esta sección analizaremos a grandes rasgos el proceso de digitalización de una señal analógica y las consideraciones necesarias para realizar este proceso de manera correcta.

La digitalización es el proceso de convertir una señal analógica a un formato digital. En un formato digital la información puede ser guardada, comprimida, procesada y analizada de forma más sencilla, práctica y rápida que en su formato analógico, de ahí que se prefiera el manejo de la información en un formato digital.

La digitalización de las señales analógicas se hace a través de una conversión analógica–digital en el que la variable analógica continua es cambiada tratando de preservar su contenido esencial. Aun así la señal digitalizada sólo es una aproximación de la señal que representa.

La digitalización ocurre en dos partes, primeramente la señal es discretizada, es decir se toman muestras de la señal en intervalos regulares de tiempo, la frecuencia de estos intervalos se denomina frecuencia de muestreo. Cada lectura es llamada una muestra de la señal analógica y en esta fase se considera que ésta tiene una precisión infinita. La siguiente parte es la etapa de cuantización, en el cual la señal discretizada se establece en niveles de amplitud fijos (niveles de cuantización). En general, estas etapas pueden ocurrir al mismo tiempo aunque son conceptualmente distintas.

La frecuencia de muestreo (f_s) tiene una importancia significativa en asegurar que la señal digitalizada pueda ser posteriormente reconstruida. El teorema de muestreo de Nyquist-Shannon establece que, para que la reconstrucción de una señal sea posible a través de sus muestras, esta señal debe de ser muestreada por lo menos al doble de la frecuencia más alta que contenga[1]. Denotando la máxima frecuencia de la señal analógica a digitalizar como f_m, el teorema de muestreo requiere que:

$$f_s \geq 2f_m \tag{2.1}$$

Todas las señales de entrada con frecuencias menores a $f_s/2$ son sujetas a ser digitalizadas. Si hay una porción de la señal de entrada con frecuencias que se encuentren por arriba de $f_s/2$, esa porción será reflejada dentro del ancho de banda de interés (entre DC y $f_s/2$) con su amplitud preservada. Este fenómeno se conoce como *aliassing* de la señal y nos hace imposible discernir la diferencia entre una señal de frecuencia baja (abajo de $f_s/2$) y una de alta frecuencia (arriba de $f_s/2$).

El fenómeno de *aliasing* de una señal en el dominio de la frecuencia se ilustra en la figura 2.1. Como se puede apreciar en la parte izquierda de la figura, cinco segmentos de bandas de frecuencia son identificados. El segmento $N = 0$ se extiende desde DC hasta la mitad de la frecuencia de muestreo. Dentro del ancho de banda de esta región, el sistema de muestreo registrará de manera fiel el contenido de la señal analógica de entrada. En los segmentos en donde $N > 0$, el contenido de frecuencias de la señal analógica será registrado por el sistema de muestreo en el ancho de banda del segmento $N = 0$. Matemáticamente, las señales de alta frecuencia a la entrada del convertidor A/D (f_{ent}) serán reflejadas de acuerdo a la siguiente expresión:

[1] **Kester W.**, *Analog-Digital Conversion*, pág. 2.27.

$$f_{aliasing} = \mid f_{ent} - N f_s \mid \tag{2.2}$$

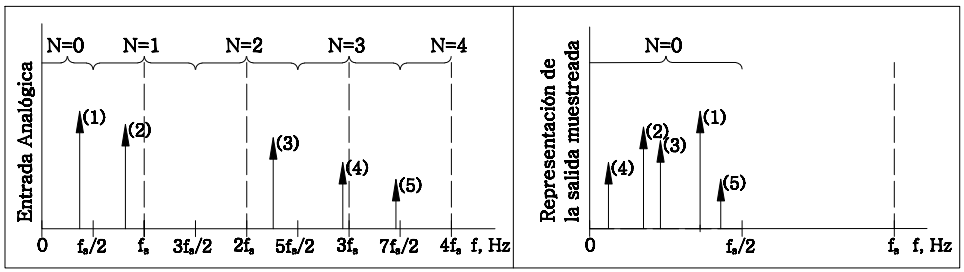

Fig. 2.1. Efecto *aliasing* en el dominio de la frecuencia.

El fenómeno de *aliasing* puede ser eliminado, o significativamente reducido, usando un filtro analógico paso bajas antes de la entrada del convertidor A/D. Este concepto se ilustra en la figura 2.2. En la gráfica, el filtro paso bajas atenúa la segunda porción de la señal de entrada a una frecuencia (2). Consecuentemente esta señal no será reflejada en la salida final muestreada.

Existen dos regiones ilustrados en la figura 2.2. La región a la izquierda que está entre DC y $f_s/2$ se encuentra dentro de la región de la banda de paso del filtro. La segunda región, la cual está sombreada, ilustra la banda de transición del filtro. Dado que esta región de transición es mayor a $f_s/2$, las señales dentro de esta banda de frecuencias serán reflejadas a la salida del sistema de muestreo. Los efectos de este error pueden ser minimizados al mover la esquina de frecuencias más abajo que $f_s/2$ o incrementando el orden del filtro. En ambos casos la mínima ganancia del filtro en la banda de atenuación a la frecuencia de $f_s/2$ debe ser menor a la razón señal a ruido (SNR) del sistema de muestreo.

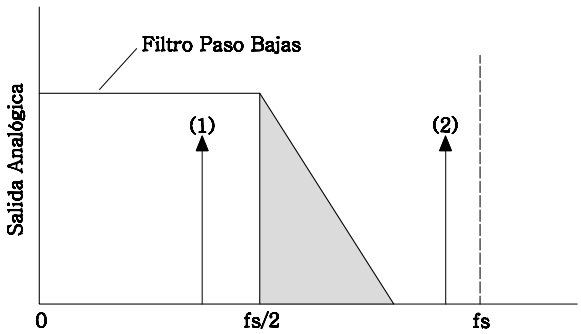

Fig. 2.2. Uso de un filtro *antialiasing*.

Esta razón señal a ruido puede ser calculada para un convertidor A/D mediante la ecuación 2.3[2].

$$SNR = (1.763 + 6.02B)[dB] \tag{2.3}$$

En donde B es el número de bits del convertidor analógico digital.

2.1.2. Diseño de filtros analógicos

Los métodos de aproximación de un filtro ideal más comúnmente utilizados son el Butterworth, Bessel y el Chebyshev. De éstos el más empleado en el diseño de filtros *anti–aliasing* es la aproximación de Butterworth. Esto se debe a que sus características de respuesta plana en la región de paso permiten que la señal no presente una distorsión al ser digitalizada, distorsión que posteriormente derive en un error de medición o tenga que ser corregido posteriormente una vez digitalizada la señal. Por esta razón nos enfocaremos únicamente a la teoría de diseño de este tipo de filtro.

El filtro Butterworth

En la figura 2.3 se muestra una respuesta de magnitud de un filtro Butterworth pasobajas. Su respuesta exhibe una transición que decrece de manera monótona con todos los ceros de transición en $\omega = \infty$, lo que lo hace un filtro totalmente de polos.

En el diseño del filtro Butterworth se deben especificar cuatro parámetros. Estos parámetros se muestran en la figura 2.4 y se listan enseguida:

A_p = atenuación en dB en la banda de paso
A_s = atenuación en la banda de rechazo
f_p = frecuencia a la cual ocurre A_p
f_s = frecuencia a la cual ocurre A_s

La magnitud de la función de transferencia al cuadrado para un filtro Butterworth de orden N-ésimo está dada por[3],

$$| T(j\omega) |^2 = \frac{1}{1 + \omega^{2N}} \tag{2.4}$$

Por ejemplo, la función de transferencia de un filtro Butterworth de segundo orden está determinada por

[2] **Baker B.**, *Nota de diseño analógico ADN010: Predict the Repeatability of Your ADC to the BITB*, Microchip.
[3] **Savant Jr., et al**, *Diseño Electrónico, Circuitos y Sistemas*, pág. 669.

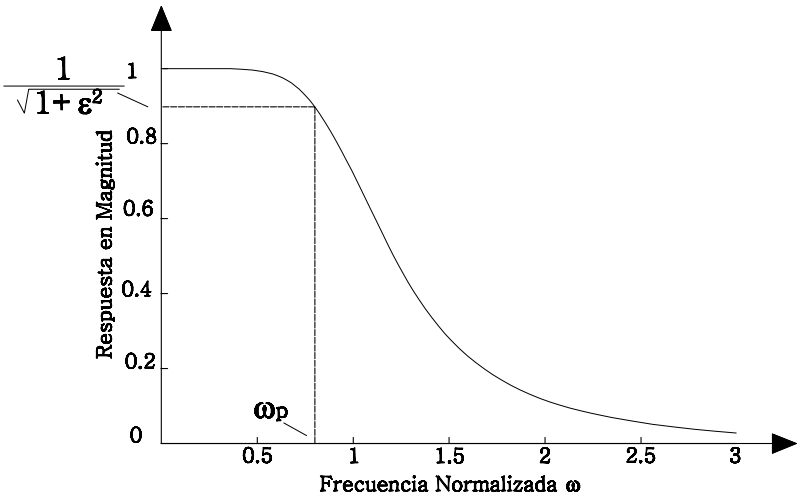

Fig. 2.3. Respuesta en magnitud de un filtro Butterworth.

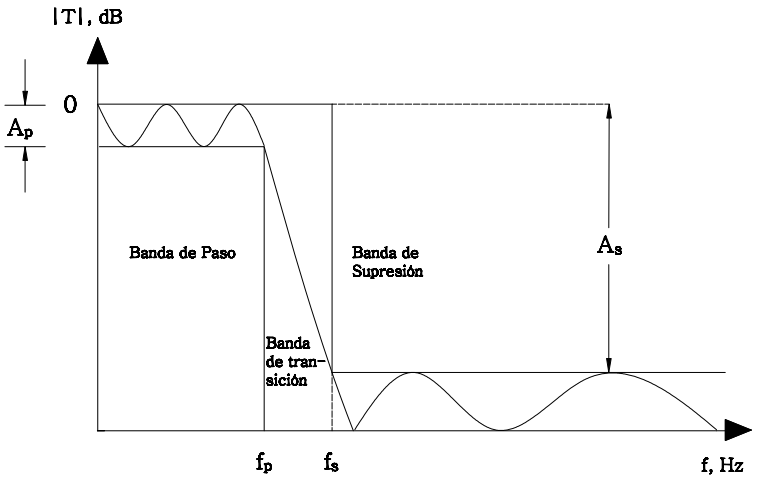

Fig. 2.4. Parámetros de diseño de un filtro.

$$H(s) = \frac{1}{s^2 + \sqrt{2}s + 1} \tag{2.5}$$

Si $s = j\omega$ se sustituye en la ecuación 2.5, la magnitud de la función compleja resultante es la dada por la ecuación 2.4 con $n = 2$.

La ecuación 2.4 es la expresión general para la magnitud al cuadrado de la función de transferencia del filtro Butterworth. Cuando el orden del filtro, N, aumenta, la pendiente de la región de transición se vuelve más inclinada.

De hecho, la pendiente es de $20N$ dB/década donde N es el orden del filtro. Esta observación permite encontrar el orden del filtro requerido cuando se especifica la pendiente.

Suponiendo que las especificaciones de un filtro indican la magnitud en dos puntos, esto es, la magnitud en $\mid H_1 \mid$ a la frecuencia de ω_1 y la magnitud de $\mid H_2 \mid$ a la frecuencia de ω_2. Se encuentra el orden requerido por el filtro de acuerdo con la ecuación 2.4. Sustituyendo los dos valores en esta ecuación, se obtiene

$$\left(\frac{\omega_2}{\omega_1}\right)^N = \sqrt{\frac{\mid H_2 \mid^{-2} -1}{\mid H_1 \mid^{-2} -1}} \tag{2.6}$$

De la ecuación 2.6, despejando el orden del filtro N obtenemos que

$$N = \frac{\log\left(\sqrt{\frac{|H_2|^{-2}-1}{|H_1|^{-2}-1}}\right)}{\log \omega_2/\omega_1} \tag{2.7}$$

De la ecuación 2.7 denotamos $\mid H_1 \mid$ como A_p, ω_1 como ω_p, $\mid H_2 \mid$ como A_s y ω_2 como ω_s. Puesto que A_p es el número de dB debajo de 0 dB, el valor de $\mid H_1 \mid$ es $10^{-A_p/20}$. Otras cantidades se transforman de manera similar. Así,

$$\mid H_1 \mid^{-2} = 10^{+0.1A_p} \tag{2.8}$$

Por lo tanto, el orden del filtro Butterworth (N_B) está dado por el resultado de la ecuación 2.9, en caso de que el resultado sea fraccional se toma el siguiente número entero.

$$N_B = \frac{\log {}^{\epsilon_2}\!/_{\epsilon_1}}{\log {}^{f_s}\!/_{f_p}} \tag{2.9}$$

donde

$$\epsilon_1 = \sqrt{10^{A_p/10} - 1} \tag{2.10}$$

y

$$\epsilon_2 = \sqrt{10^{A_s/10} - 1} \tag{2.11}$$

Los polos de un filtro Butterworth de orden N se determina con la construcción gráfica mostrada en la figura 2.5[4]. Observe que los polos quedan en un circulo de radio $\omega_p(1/\epsilon)^{1/N}$ separados por ángulos iguales de π/N, con el primer polo en un ángulo $\pi/2N$ a partir del eje imaginario.

[4] **Sedra A. y Smith C.**, *Circuitos Microelectrónicos*, pág. 1093.

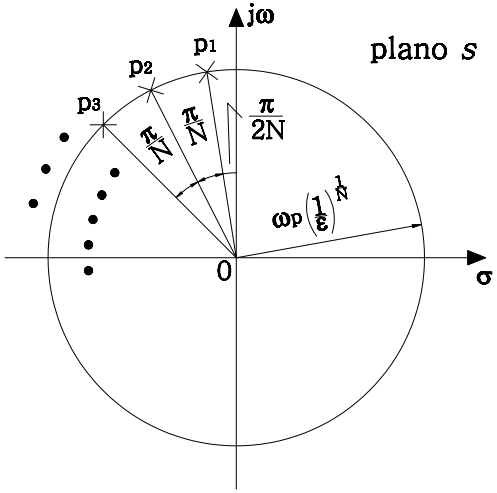

Fig. 2.5. Construcción gráfica para determinar los polos de un filtro Butterwoth.

Como todos los polos están a una distancia radial igual del origen, tienen la misma frecuencia $\omega_0 = \omega_p(1/\epsilon)^{1/N}$. Una vez que se encuentran los N polos p_1,p_2,...,p_N, la función de transferencia se escribe

$$H(s) = \frac{K\omega_0^N}{(s-p_1)(s-p_2)...(s-p_N)} \tag{2.12}$$

donde K es una constante igual a la ganancia DC requerida por el filtro.

En resumen, para encontrar la función de transferencia Butterworth que cumpla con las especificaciones de transmisión requeridas se realiza el siguiente procedimiento:

1. Se determina ϵ_1 y ϵ_2 con las ecuaciones 2.10 y 2.11
2. Se determinar el orden requerido por el filtro como el valor entero más alto del resultado de la ecuación 2.9
3. Se usa la figura 2.5 para determinar los polos del filtro Butterworth
4. Con los polos se determina el polinomio correspondiente a la función $T(s)$

2.1.3. Implementación de filtros analógicos activos

Los filtros activos usan una combinación de un amplificador, una o tres resistencias y uno o dos capacitores para implementar filtros de uno o dos polos.

Filtros de polo único

La respuesta de un filtro de un polo único activo es similar a la de un filtro de un solo polo pasivo, con la ventaja de que el filtro activo, proporciona aislamiento entre etapas. Un ejemplo de su implementación y respuesta se pueden observar en la figura 2.6.

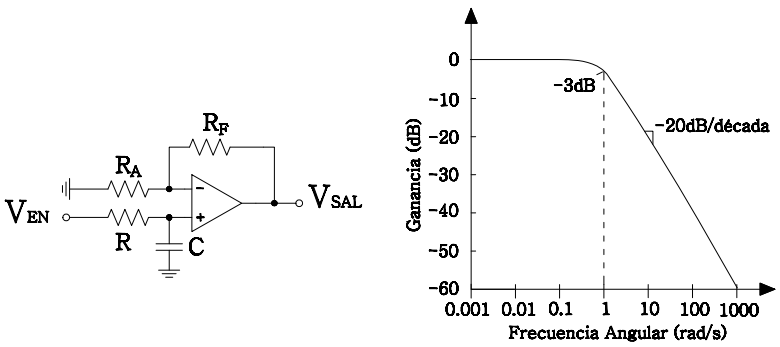

Fig. 2.6. Respuesta en frecuencia de un filtro de polo único.

La función de transferencia del filtro mostrado en la figura 2.6 está dada por la ecuación 2.13.

$$H(s) = \frac{1 + R_F/R_A}{1 + sRC} \tag{2.13}$$

Filtros de polo doble

Topología fuente de tensión controlada por tensión

Este tipo de configuración es mejor conocida como la configuración Sallen-Key. En la figura 2.7 se muestra la implementación de esta topología con una ganancia unitaria en DC. En la implementación mostrada en la figura los polos del circuito están determinados por los valores de R_1, R_2, C_1 y C_2 mediante la siguiente expresión:

$$H(s) = \frac{1}{1 + C_2(R_1 + R_2)s + C_1C_2R_1R_2s^2} \tag{2.14}$$

Topología realimentación múltiple

La topología de un filtro de segundo orden de realimentación múltiple se muestra en la figura 2.8. La ganancia en DC del filtro está dada por la relación

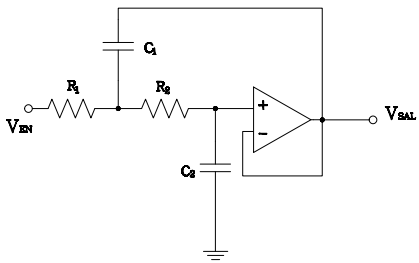

Fig. 2.7. Topología Sallen-Key para un filtro de segundo orden.

entre R_1 y R_2. Sus polos están determinados por los valores de R_1, R_3, a través de la siguiente función de trasferencia:

$$H(s) = \frac{-1/R_1R_3C_5C_6}{s^2C_2C_1 + sC_1(1/R_1 + 1/R_2 + 1/R_3) + 1/(R_2R_3C_2C_1)} \tag{2.15}$$

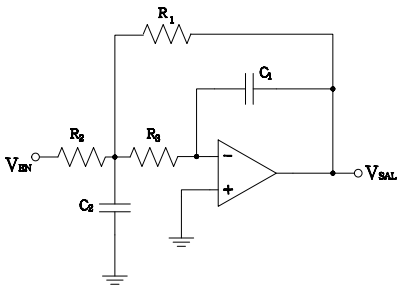

Fig. 2.8. Topología de Realimentación Múltiple para un filtro de segundo orden.

2.1.4. Amplificadores operacionales

Dejando de lado al transistor, el amplificador operacional es el bloque básico de construcción de muchas aplicaciones analógicas. Funciones fundamentales como la de amplificación, aislamiento de carga, inversión, suma y resta de señales pueden ser implementadas con el uso de un amplificador operacional.

Debido a la restricción de no disponer de una fuente de alimentación negativa o la cantidad de componentes necesarios para implementar esta fuente,

no es posible el uso de amplificadores de alimentación bipolar en aplicaciones embebidas.

Cuando se trabaja con amplificaciones de alimentación unipolar cierto tipo de consideraciones de diseño deben ser hechas.

Circuito buffer o seguidor

El circuito *buffer* es útil para resolver problemas con el acople de impedancias y aislar circuitos de potencia de circuitos sensibles. Este circuito puede ser implementado usando un amplificador de fuente unipolar como se muestra en la figura 2.9. En este circuito, como en todos los circuitos con amplificadores operacionales, un capacitor C de *by pass* debe ser usado para evitar que el amplificador pueda llegar a oscilar[5] y eliminar el posible ruido proveniente de otros circuitos. La ganancia de este circuito es unitaria.

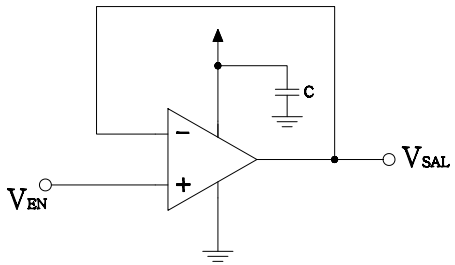

Fig. 2.9. Circuito *buffer*.

Este circuito posee un comportamiento lineal sobre el ancho de banda del amplificador. La única restricción que tiene la señal de entrada del circuito, es que ésta no debe violar los límites de modo común y el rango de variación de salida del amplificador (*output swing*). En caso de querer alimentar cargas de baja impedancia es necesario que las especificaciones del amplificador seleccionado estipulen que éste puede entregar la corriente necesaria a la salida.

Circuitos amplificadores

Si se desea darle ganancia a una señal analógica, dos tipos fundamentales de circuitos amplificadores pueden ser utilizados. El circuito de la figura 2.10 produce una señal amplificada sin inversión a la salida.

[5] **Baker B.**, *AN682: Using single supply Operational Amplifiers in Embedded Systems,* pág. 1.

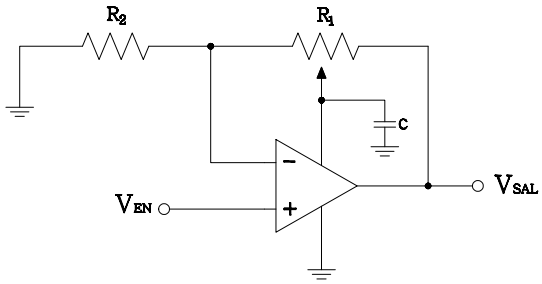

Fig. 2.10. Circuito amplificador no inversor.

La ganancia que este circuito aplica a la señal de entrada está dado por la ecuación 2.16.

$$V_{\mathrm{SAL}} = \left(1 + \frac{R_1}{R_2}\right) V_{\mathrm{EN}} \tag{2.16}$$

Valores típicos de las resistencias usadas en circuitos con fuente unipolar están por arriba de los 2 [kΩ] para R_1[6]. El valor de las resistencias se escoge tomando en consideración el valor de la ganancia deseado para el circuito. Para fijar la ganancia, adicionalmente, deben ser tomados en cuenta el nivel de ruido del amplificador y el *offset* de la señal de entrada.

De nueva cuenta, este circuito tiene restricciones en cuanto al margen de las señales de entrada y de salida. La entrada de la terminal no inversora está restringida al índice de rechazo de modo común del amplificador. El margen de salida del amplificador limita la amplitud de la señal de salida, por lo que si se observan cortes de la señal, se debe disminuir la ganancia del amplificador.

La configuración inversora de un amplificador operacional alimentado con una fuente unipolar se puede lograr a través del circuito mostrado en la figura 2.11. Como se puede apreciar en este circuito, los circuitos inversores que utilizan amplificadores de alimentación unipolar requieren de un divisor de tensión que eleve la señal de salida para mantenerla entre el nivel de alimentación negativa (generalmente tierra) y el nivel positivo.

La salida de este circuito está dado por la ecuación 2.17.

$$V_{\mathrm{SAL}} = -\left(\frac{R_1}{R_2}\right) V_{\mathrm{EN}} + \left(1 + \frac{R_1}{R_2}\right) V_{\mathrm{SESGO}} \tag{2.17}$$

[6] Ibid, pág. 2.

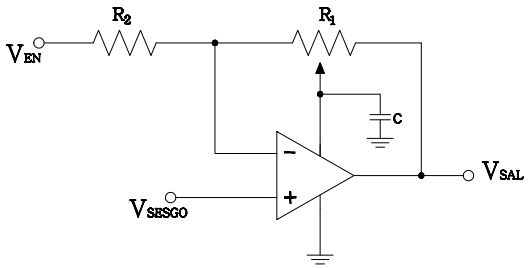

Fig. 2.11. Circuito amplificador inversor.

2.1.5. El convertidor analógico/digital

Un convertidor analógico digital (convertidor A/D) es un dispositivo que convierte una señal continua en números digitales discretos proporcionales a la magnitud de la señal.

Existen diversas formas de implementar un convertidor A/D. Cada una de ellas tiene ventajas y desventajas, la elección de uno u otro tipo de convertidor depende de la aplicación en donde se vaya a emplear. En seguida se describirá el convertidor A/D por aproximaciones sucesivas, el cual fue utilizado en este proyecto.

Si bien hay muchas variaciones para su implementación, la arquitectura básica de un convertidor por aproximaciones sucesivas es sencilla y se muestra en la figura 2.12[7].

Un convertidor A/D de aproximaciones sucesivas utiliza un comparador para rechazar intervalos de tensiones hasta, eventualmente, establecerse en un valor. Estas comparaciones se realizan entre la tensión de entrada y la tensión de salida de su convertidor digital analógico (CDA), él cual es alimentado con el actual valor aproximado. A cada paso de este proceso, el valor binario de cada aproximación es almacenado en un registro de aproximaciones sucesivas (RAS). El convertidor A/D utiliza una tensión de referencia para realizar las comparaciones. El valor de tensión de este referencia (V_{REF}) es el máximo valor de tensión que puede ser convertido sin saturar la salida del convertidor A/D.

El proceso de conversión se realiza de la siguiente manera:

La tensión analógico de entrada (V_{EN}) es mantenido en un circuito de muestreo y retención (*sample/hold*). Para implementar el algoritmo de búsqueda, el registro de N-bits es primeramente establecido a su escala media

[7] **Maxim**, *Application Note 1080: Understanding SAR ADCs.*

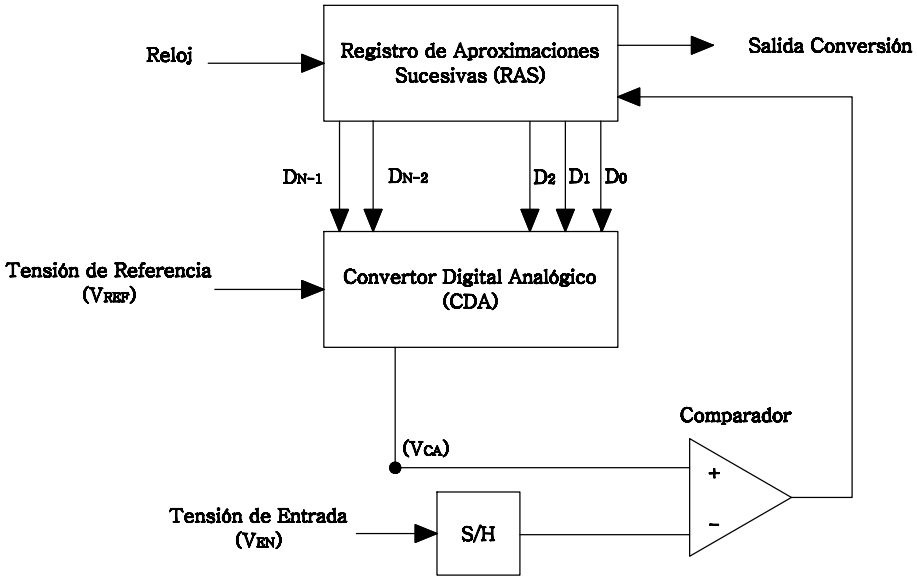

Fig. 2.12. Diagrama de un convertidor de aproximaciones sucesivas.

(esto es, 100...00), donde el bit más significativo se establece en 1. Esto fuerza la salida del CDA (V_{CDA}) volverse $V_{REF}/2$, posteriormente se realiza una comparación entre la salida del CDA y la tensión de entrada. En caso de que la tensión de salida del convertidor sea mayor o igual a V_{EN}, se determina que el bit más significativo debe de ser uno, pero si $V_{EN}<V_{CDA}$, se debe memorizar un cero en el registro de salida. En el siguiente pulso de reloj se efectúa una segunda comparación del V_{CAD} correspondiente a la palabra 110...00, si la comparación anterior había sido positiva ó 010...00 en caso contrario, la salida del comparador determina que valor debe de memorizarse en el bit de peso $V_{REF}/4$, creándose de esta forma la palabra de salida digital una vez realizadas las "n" comparaciones sucesivas; donde "n" es el número de bits de resolución del convertidor A/D.

Dado que cada comparación se realiza en un pulso de reloj, la frecuencia de muestreo del convertidor A/D es la frecuencia del reloj del convertidor dividida entre el número de bits de resolución de éste.

2.2. Adquisición de pulsos digitales

La adquisición de un pulso digital se puede realizar a través de cualquier puerto de entrada de un circuito digital. Dado que la velocidad de recepción de este tipo de circuitos suele ser muy rápida, en caso de que la señal que se recibe contenga ruido, el sistema digital podría interpretar que más de un pulso ha sido enviado, esto es especialmente importante al recibir pulsos provenientes de fuentes mecánicas debido al efecto rebote que generan.

2.2.1. Rebote mecánico

El efecto del rebote mecánico presenta un repentino prendido y apagado de un interruptor en un corto tiempo, debido a las vibraciones mecánicas producidas durante el contacto como se observa en la figura 2.13. El rebote mecánico puede causar un malfuncionamiento de una entrada de un contador o de una terminal de *reset*. Usualmente, el rebote mecánico es eliminado con un filtro pasobajas, pero para asegurar una mayor inmunidad del circuito se puede utilizar una combinación de un filtro pasobajas y un *buffer* con Schmitt Trigger[8].

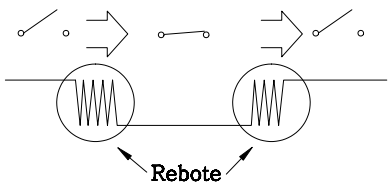

Fig. 2.13. Efecto de rebote en un interruptor mecánico.

Un *buffer* con Schmitt Trigger cambia su estado de salida cuando la tensión en su entrada sobrepasa cierto nivel (V_{u+}); la salida no vuelve a bajar cuando la entrada baja a ese nivel, sino cuando ésta llega a un nivel distinto más bajo que el primero (V_{u-}). A este efecto se le conoce como ciclo de histéresis y se ilustra en la figura 2.14. Los niveles de tensión de los *buffers* varían de acuerdo al modelo y a la tensión con la cual sea alimentado el circuito integrado. Como un ejemplo para el modelo de *buffer* 74LVC1G17 sus valores se muestran en la tabla 2.1[9].

[8] **Maxim**, *Aplication Note 287: Switch bounce and other dirty little secrets.*
[9] **NXP semiconductors**, *74LVC1G17, Single Schmitt trigger buffer.* Hoja de datos del producto.

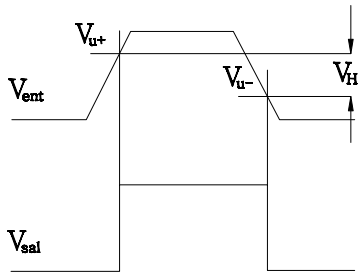

Fig. 2.14. Efecto de histéresis.

Tensión de Alimentación [V]	**Tensión de Umbral Positivo (V_{u+}) [V]**	**Tensión de Umbral Negativo (V_{u-}) [V]**	**Tensión de Histéresis (V_H) [V]**
1.8	1.0	0.6	0.4
3.0	1.5	1.0	0.5
5.5	2.5	1.8	0.7

Tabla 2.1. Valores de tensión del *buffer* 74LVC1G17.

En la figura 2.15 se muestra la propuesta de un circuito para eliminar el rebote mecánico. Como se muestra en el diagrama, el circuito se implementa con un filtro pasivo pasobajas y un *buffer* digital con Schmitt Trigger, adicionalmente, el diodo en el circuito permite la descarga del capacitor para evitar que el valor máximo a la entrada del *buffer* sea excedido.

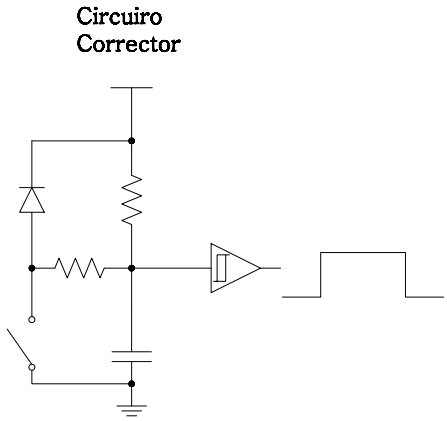

Fig. 2.15. Circuito corrector del efecto rebote.

2.3. Memorias de almacenamiento

El almacenamiento de datos se puede realizar a través de dos tipos de memoria; estos son de tipo volátil o no volátil. Las memorias volátiles abarcan diversos tipos de *Random-Access Memories* (RAM) como las *Dynamic Random-Acess Memories* (DRAM) y las *Static Random-Access Memories* (SRAM). Este tipo de memorias de almacenamiento temporal requieren constantemente estar energizadas para retener la información guardada en ellas.

Las memorias no volátiles son aquellas que retienen la información incluso si el dispositivo es desenergizado. Dentro de esta categoría se encuentran las memorias construidas con base en transistores (flash, *Electrically Erasable Programmable Read-Only Memory* (EEPROM), *Erasable Programmable Read-Only Memory* (EPROM), etc.) y otros tipos de memorias de almacenamiento magnético y óptico.

En el caso del diseño de un *datalogger*, se requiere para el almacenamiento de datos, de memoria no volátil, de tamaño pequeño, reutilizables y de bajo consumo, lo cual descarta las de guardado magnético y óptico. Se requiere también que la memoria elegida sea práctica en su uso, lo anterior descarta a la memorias EPROM debido a que su borrado es a base de luz ultravioleta y las flash dado que su modo de operación es con base en bloques de bits y no de bits individuales. Si bien las memorias EEPROM no tienen la capacidad de almacenamiento de otros tipos de memorias, incluidas las tipo flash, son lo suficientemente densas para almacenar los datos esperados por el *datalogger*. A continuación se describirá su estructura, uso y restricciones.

2.3.1. Memorias EEPROM

Las memorias EEPROM son dispositivos de almacenamiento no volátil, que como su nombre lo indica, pueden ser borrados y escritos de manera eléctrica. Son usados en computadoras y otros dispositivos electrónicos para almacenar una cantidad relativamente pequeña de información.

Las memorias EEPROM al igual que las memorias flash y EPROM son construídas con base en arreglos de *Floating Gate* MOSFETS (FGMOSFETS), gracias a su capacidad de almacenar carga por largos periodos de tiempo sin necesidad de estar alimentadas. En la figura 2.16 se muestra el corte de un transistor FGMOSFET. Como se puede apreciar el óxido de silicio rodea complemente la compuerta flotante, por lo que su carga permanece aislada. La carga en la compuerta flotante puede ser modificada al aplicar tensión eléctrica sobre la

fuente, el drenaje, el cuerpo y la compuerta de control. Esto permite que el transistor funcione como una memoria para un bit de datos.

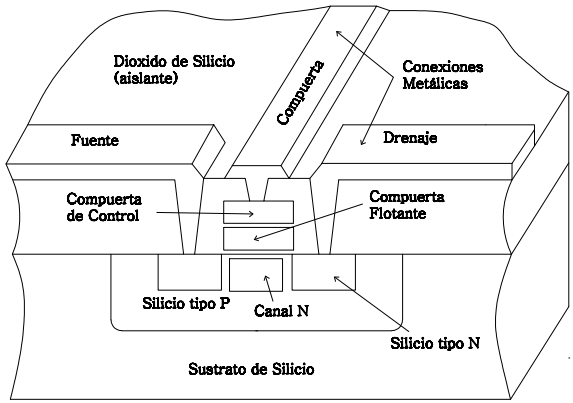

Fig. 2.16. Diagrama interno de un FGMOSFET.

Los tipos de interfaces de comunicación más comunes para este tipo de memorias son el SPI, la I^2C y la 1-WIRE. Estas interfaces requieren entre uno y cuatro terminales de control por lo que la memoria puede ser encapsulada en empaques de ocho terminales o menos.

Las memorias EEPROM típicamente operan en tres fases; fase de código de operación, fase de direccionamiento y fase de datos. La fase de código de operación es generalmente los primeros ocho bits que se envían a la memoria, seguida de ocho o 24 bits de direccionamiento, dependiendo de la cantidad de celdas de memoria que ésta posea, y finalmente, en la fase de datos, se trasmite o recibe la información a leer o escribir.

Cada memoria tiene códigos de operación distintos pero siempre poseerán códigos para la lectura y escritura de datos (en el caso del I^2C, por ejemplo, estos son inherentes al protocolo).

Los modos de falla de las memorias EEPROM en cuanto a retención de datos se deben a dos factores: número de ciclos de escritura y tiempo de retención de datos.

Durante el proceso de reescritura de las memorias, la capa de óxido que rodea a la compuerta flotante gradualmente acumula electrones. El campo eléctrico de estos electrones atrapados se suma a la de los electrones en la compuerta flotante, descendiendo el umbral de tensión entre un cero y un uno almacenado. Después de un número determinado de ciclos de reescritura, las diferencias se vuelven demasiado pequeñas para ser reconocidas y la celda

queda encasillada en un estado programado, es en ese momento que la falla por ciclos de escritura ocurre. El fabricante típicamente especifica un número de reescrituras por celda antes de que esto ocurra, normalmente este valor es de cien mil o un millón de veces.

Por otra parte, los electrones insertados en la compuerta flotante pueden fugarse a través del aislamiento, esto ocurre especialmente si hay un incremento en la temperatura, lo cual lleva a la celda a entrar en un estado de borrado. El fabricante también especifica un tiempo mínimo de retención de datos que típicamente es de diez años.

2.4. Transmisión de información a una computadora personal

Dado que las computadoras personales (PC) son elementos muy estandarizados en su arquitectura, la manera más fácil de realizar una comunicación con una PC es a través de la implementación de uno o varios de los protocolos de comunicación que ésta ya tenga presentes.

Un protocolo de comunicación es un conjunto de reglas estándar para la conexión, comunicación y transferencia de información entre una computadora y otro dispositivo. El protocolo puede ser implementado por software, hardware o una combinación de ambos. Los tipos de comunicación definidos por estos protocolos pueden ser de tipo síncrono o asíncrono.

Los protocolos que se estudiarán en esta sección son aquellos que se implementaron en el *datalogger*. Estos son el protocolo RS-232 y el protocolo USB.

2.4.1. Estándar de comunicación RS-232

El estándar RS-232, definido a comienzos de los años 60, especifica las tensiones, los tiempos y la función de las señales eléctricas; además, el protocolo de intercambio de información y los conectores mecánicos para realizar una comunicación serial asíncrona entre dos equipos que implementen el estándar. La palabra serial significa que la información es enviada bit a bit a través del *bus* de comunicación, la palabra asíncrona nos dice que la información no es enviada en espacios de tiempo predefinidos y la transferencia de datos puede empezar en cualquier momento y es tarea del receptor detectar cuando un mensaje comienza y termina.

Flujo de datos del RS-232

Con una comunicación síncrona, un reloj o una señal de disparo debe de estar presente para indicar el principio de cada transferencia. La ausencia de esta señal de reloj en una comunicación asíncrona, como la definida por el estándar RS-232, hace al canal de comunicación asíncrono más barato de operar pero con la "desventaja" de que el receptor puede empezar a recibir la información en un tiempo inadecuado. En caso de que esto suceda una resincronazación de la señal es necesaria con un costo en tiempo, adicionalmente, toda información recibida en el tiempo de resincronización se pierde. Para recuperar la información perdida es necesario que el receptor detecte los bytes de información faltantes y pida que éstos sean retransmitidos. Otra desventaja es que son requeridos bits extras en el flujo de datos para indicar el comienzo y el fin de la información útil. Estos bits extras utilizan un ancho de banda adicional.

Los bits son enviados con una frecuencia predefinida, el *buad rate*. Ambos el trasmisor y el receptor deben de ser programados para usar la misma frecuencia de transferencia de los bits. Después de recibir el primer bit, el receptor calcula en qué momento el siguiente bit de datos debe de ser recibido. Éste revisará el valor de tensión de la línea de información en ese momento.

Para poder ser enviada la información de manera serial, ésta debe de ser separada en palabras de información. En la figura 2.17 se muestra la representación de una palabra de información del estándar RS-232. La longitud de esta palabra es variable, en una PC esta longitud puede ser seleccionada entre cinco y ocho bits. Esta longitud es la longitud neta de información en cada palabra. Para una trasmisión apropiada son agregados ciertos bits con propósitos de sincronización y de detección de errores, los cuales son: el bit de inicio, el o los bits de paro y el bit de paridad. Es importante que tanto el receptor como el trasmisor utilicen la misma cantidad de bits, de otra manera el mensaje puede no ser interpretado de manera correcta o incluso no ser reconocido.

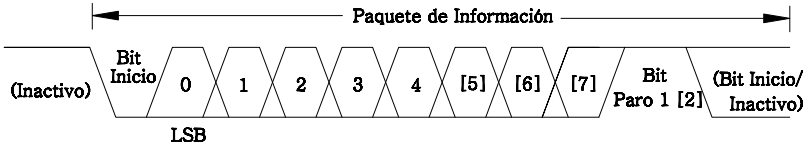

Fig. 2.17. Diagrama de tiempo de la comunicación del RS-232.

Con el estándar RS-232, la tensión de línea puede tener dos estados. El estado encendido es conocido como marca y el estado de apagado como espacio. Ningún otro estado es posible, cuando la línea no es usada, el estado es mantenido en marca.

Los tipos de bits que conforman el flujo de información son los siguientes:

El bit de inicio es un bit de atención para que el receptor se dé cuenta que un paquete de información está por ser enviado. Este bit de inicio siempre es identificado como un espacio. Debido que cuando la línea no es usada, ésta se mantiene en marca, el bit de inicio es fácilmente reconocido por el receptor.

A continuación del bit de inicio los bits de datos son enviados. Un valor de uno causa que la línea vaya a un nivel de marca, el valor de bit cero es representado mediante un nivel de espacio. El bit menos significativo siempre es enviado primero.

Para propósitos de detección de error, es posible adicionar un bit extra que indica la paridad de los bits de datos enviados. El transmisor calcula el valor de este bit dependiendo de la información que va a enviar, el receptor realiza el mismo cálculo y revisa si el valor del bit de paridad corresponde al valor calculado.

Supongamos que el receptor no se da cuenta de un bit de inicio debido al ruido de la línea de trasmisión. Así que éste comienza la recepción en el siguiente valor de espacio que encuentra en los bits de información, esto causaría que información confusa llegara al receptor. Para evitar esta situación, un mecanismo de resincronazación se encuentra presente y hace uso del enmarcamiento que los paquetes de información poseen.

Un enmarcamiento de los paquetes de información significa que todos los bits de datos y de paridad están contenidos en un marco entre un bit de inicio y un bit o bits de paro. El periodo de tiempo entre el bit de inicio y los bits de paro es una constante definida por el *baud rate* y el número de bits de datos y el bit de paridad. El bit de inicio siempre tiene un valor de espacio y el bit de paro de marca. Si el receptor detecta un valor de espacio donde un bit de paro debiera estar presente en la línea, éste reconoce que ha ocurrido un error en la sincronización. El dispositivo entonces trata de resincronizarse con los nuevos bits recibidos.

Para el proceso de resincronización, el receptor busca de la información recibida un par de bits de paro e inicio válidos. Esto funciona siempre que haya una gran variación en el patrón de las palabras de datos. Por ejemplo, si palabras de ceros son enviadas repetidamente el proceso de sincronización no es posible.

Propiedades físicas del estándar RS-232

El estándar RS-232 describe un método de comunicación capaz de intercambiar información en diferentes ambientes. Esto tiene un impacto en los niveles máximos de tensión de las señales, los conectores, etc. En la definición original del estándar fue establecida una velocidad máxima de 20 [kbps]. En la actualidad dispositivos como la UART 16550A alcanzan velocidades hasta de 1.5 [Mbps].

Tensiones

El nivel de tensión de la señal en las terminales del estándar RS-232 puede tener dos estados. Un bit alto, o marca, es identificado por una tensión negativa y para un bit bajo, o espacio, se utiliza un valor positivo. Los voltajes límites están mostrados en la tabla 2.2.

Nivel	**Capacidad del Transmisor** [V]	**Capacidad del Receptor** [V]
Nivel espacio (0)	+5 ... +15	+3 ... +25
Nivel marca (1)	–5 ... –15	–3 ... –25
Indefinido	—	–3 ... +3

Tabla 2.2. Valores de tensión del estándar RS-232.

Cables y Conectores

Según el estándar del RS-232, la distancia máxima del cable de conexión utilizado en la comunicación es aquel con una capacitancia menor o igual 2500 [pF]. Utilizando un cable estándar esta capacitancia se alcanza entre los 15 y 20 metros, pero si, por ejemplo, se utiliza cable UTP CAT-5 con una capacitancia de 52 [pF/m] esta distancia se incrementa considerablemente.

El conector originalmente definido para el estándar RS-232 es el conector DB25, el cual posee 25 terminales para realizar la conexión serial. Debido al gran tamaño de este conector, en los equipos de cómputo que aún poseen este tipo de comunicación se suele encontrar el conector DB9, el cual realiza una conexión con sólo nueve terminales y elimina las terminales del conector DB25 que realizan funciones raramente usadas. En la figura 2.18 se muestra la distribución de las terminales de un conector DB9 macho y hembra, y en la tabla 2.3 se muestra el nombre de dichas terminales y su dirección.

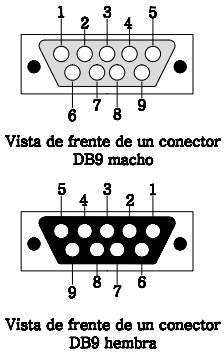

Fig. 2.18. Terminales del conector DB9.

Núm. de Terminal	Nombre	Dirección	Abreviatura
1	*Data Carrier Detect*	Ent	DCD
2	*Received Data*	Ent	RD
3	*Trasmitted Data*	Sal	TD
4	*Data Terminal Ready*	Sal	DTR
5	*Common Ground*	-	GND
6	*Data Set Ready*	Ent	DSR
7	*Request To Send*	Sal	RTS
8	*Clear To Send*	Ent	CTS
9	*Ring Indicator*	Ent	RI

Tabla 2.3. Señales del estándar RS-232 en un conector DB9.

En caso de que no se necesiten señales de control de flujo entre los dispositivos, una comunicación RS-232 se puede realizar con sólo tres terminales, la de recepción de datos (RD), la de transmisión (TD) y la referencia de las señales (GND).

2.4.2. El estándar USB

El estándar USB se ha convertido en el puerto de comunicación entre la computadora y dispositivos externos más usado en la actualidad. Dentro de las características de este estándar están el permitir a los dispositivos una conexión con la computadora sin la necesidad que ésta deba ser reinicializada y la posibilidad de alimentar a los dispositivos conectados a través del mismo *bus*.

Los tres componentes principales del estándar USB son: el anfitrión (*host*), el *hub* y los dispositivos (*devices*). Los dispositivos son el punto final del sistema como ratones, impresoras, etc. Múltiples dispositivos pueden conectarse a un solo puerto USB a través de un *hub*. El dispositivo anfitrión es el centro de la comunicación USB.

El protocolo USB define las características necesarias para la interconexión de los dispositivos con el anfitrión USB y esto incluye los conectores, los cables, la topología y las capas de comunicación.

Flujo de datos del USB

El estándar USB emplea para la transmisión de sus paquetes de información la codificación *Non Return to Zero Inverted* (NRZI). En la figura 2.19 se muestra un ejemplo de este tipo de codificación. La codificación NRZI representa un uno lógico manteniendo el nivel de la línea de datos y un cero lógico mediante el cambio en el nivel de ésta. Una cadena de ceros causa que la línea de datos cambie a cada bit trasmitido mientras que una cadena de unos causa un período sin transición de la línea.

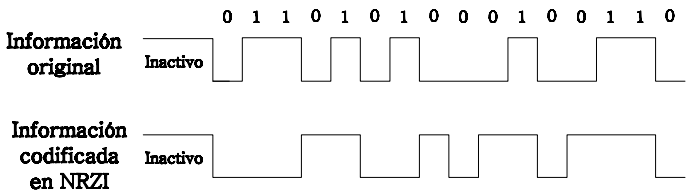

Fig. 2.19. Codificación NRZI.

Topología

La topología se basa en un modelo de conexión estrella escalonada, con el anfitrión en la punta de la pirámide y los *hubs* ejes en el centro de cada conexión con los dispositivos. El anfitrión siempre tiene un camino directo de datos hacia cada uno de los dispositivos.

El anfitrión

El anfitrión controla el estado del *bus* y de los dispositivos conectados a éste. Sólo puede existir un anfitrión en una red USB. Las siguientes son funciones del anfitrión:

- Detectar cuando un dispositivo es conectado y desconectado
- Manejar el control de flujo de datos del *bus*
- Suministrar energía a los dispositivos que así lo soliciten

Dispositivos

Los dispositivos pueden ser descritos de acuerdo a la clase a la cual pertenecen. Una clase es un grupo de dispositivos que tienen similar funcionalidad e implementación, por ejemplo: monitores, dispositivos de audio, de almacenamiento masivo, impresoras, dispositivos de interfaz humana (teclado, ratones, etc.). Esto permite el desarrollo de un controlador genérico para toda una clase de dispositivos y que el usuario no deba de instalar un controlador específico para cada uno. Si un dispositivo tiene más funcionalidades que las presentes en este controlador genérico, un controlador específico debe ser instalado para su funcionamiento.

Todos los dispositivos USB tienen un descriptor que incluye el proveedor del equipo, el identificador del producto y el número de configuraciones que el dispositivo tiene. De esta manera el anfitrión conoce cuantos descriptores de configuración debe de pedir. Los descriptores de configuración incluyen el número y el tipo de interfaces o funciones que el dispositivo posee. Después de recibir todos los descriptores de configuración, el anfitrión pide los descriptores de interfaz. El descriptor de interfaz le dice al anfitrión el tipo de características y qué clase de características el dispositivo tiene.

Un dispositivo puede pertenecer a clases distintas, por ejemplo, si se conecta una cámara web con micrófono, el dispositivo manejara una clase para el audio y otra para el video.

El protocolo de comunicación

El protocolo de comunicación está definido por las rutinas que determinan como los dispositivos interactúan en el *bus*. En el protocolo USB la comunicación entre el anfitrión y los dispositivos conectados al *bus* se realiza a través de paquetes. Una transacción es un grupo de estos paquetes que realizan una función útil y éstas son siempre iniciadas por el anfitrión.

Un ejemplo de la comunicación entre un anfitrión y un dispositivo se puede observar en la figura 2.20. Como se observa en la figura, el anfitrión utiliza un método de encuesta para obtener información de un dispositivo. La comunicación inicia con el anfitrión enviando un paquete *token*. El paquete *token*

es un tipo de paquete sólo usado por el anfitrión USB que informa a un dispositivo que el anfitrión desea realizar una acción sobre éste. Si una petición de información es enviada con el paquete *token*, el dispositivo responde con un paquete de datos o un paquete indicando que no hay información por enviar. Si no existe información, el anfitrión envía un paquete *token* al siguiente dispositivo.

El paquete *token* también puede indicar al dispositivo que espere por información. Si éste es el caso, un paquete de datos será enviado por el anfitrión, al que el dispositivo responderá con un mensaje de confirmación, ya sea que la operación haya tenido éxito (Enterado) o no (No Enterado).

El estándar USB fue diseñado para manejar transacciones a velocidades bajas, medias y altas que corresponden 1.5 [Mbps], 12 [Mbps] y 480 [Mbps] respectivamente y se les refiere como modos de señalización.

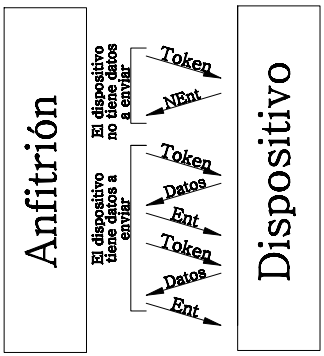

Fig. 2.20. Ejemplo de comunicación entre un anfitrión USB y un dispositivo.

Cables y conectores

Los cables consisten en dos líneas de alimentación y dos líneas para la trasmisión de información, las señales de datos son diferenciales. La máxima distancia permitida de los cables es de 5 [m]. La razón principal de esto es que el máximo retardo que la señal puede tener es alrededor de 1500 [ns]. Si los comandos de un dispositivo no son recibidos en ese tiempo, el anfitrión determina que estos fueron perdidos.

El estándar especifica varios tipos de conectores y sus receptáculos, algunos tienen la intensión de servir para los anfitriones (conector tipo A) y otros para los dispositivos (tipo B), de modo que no se pueda conectar dispositivo con dispositivo y anfitrión con anfitrión. En la figura 2.21 se puede apreciar las

terminales de cada tipo de conector y en la tabla 2.4 el nombre de cada terminal y su función.

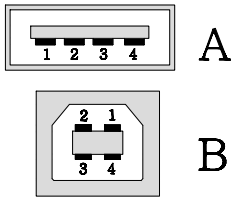

Fig. 2.21. Terminales de los conectores USB A y B.

Terminal	**Nombre**	**Descripción**
1	VCC	+5 [V] de CD
2	D-	Datos -
3	D+	Datos +
4	GND	Tierra

Tabla 2.4. Terminales de los conectores del estándar USB.

2.5. El lenguaje de programación C#

C# (pronunciado en inglés *si sharp*) es un lenguaje de programación orientado a objetos y de escritura segura. C# tiene sus raíces en la familia de lenguajes C y su escritura es parecida a los lenguajes C, C++ y Java. C# fue un lenguaje desarrollado por Microsoft como parte de su plataforma .NET y posteriormente fue estandarizado por el estándar Ecma (ECMA-334) y el estándar ISO/EIC (ISO/IEC 23270). La versión actual del lenguaje es la 3.0 y fue lanzada en conjunto con el .NET Framework 3.5 en 2007.

Dentro de las características de este lenguaje de programación están:

- Incluye programación orientada a componentes, si bien C# es un lenguaje de programación orientado a objetos.
- Contienen un colector de basura automático que libera la memoria utilizada por objetos sin uso.

- Tiene un diseño de escritura segura, lo que hace imposible de leer en variables no inicializadas, arreglos fuera de sus fronteras y hacer *cast* de variables sin revisión.
- Los apuntadores de memoria sólo pueden ser usados en bloques marcados como inseguros, un programa con código inseguro requiere de permisos apropiados para ser ejecutado.
- Incluye un tipo de variable booleano.
- No permite múltiple herencia, si bien las clases pueden ser implementadas con múltiples interfaces.
- Actualmente el lenguaje posee 77 palabras reservadas.

Un ejemplo de código en C# es:

```
using System;
class Hello
{
   static void Main()
        {
      Console.WriteLine("Hello, World");
   }
}
```

Si bien existen compiladores de C# como Mono, desarrollado por Novell, para crear aplicaciones en los distintos sistemas operativos como Linux, Solaris, Mac OS X y Windows, para la realización de este proyecto se utilizó la versión desarrollada por Microsoft, bajo su ambiente Framework .NET contenida dentro de Visual Studio 2008.

El Framework .NET es una estructura de soporte definida, mediante el cual otro proyecto de software puede ser organizado y desarrollado. Este incluye un gran número de herramientas incluyendo interfaz de usuario, conectividad con bases de datos, criptografía, desarrollo de aplicaciones web, algoritmos numéricos y comunicación en red. Su librería de clases es usada para que en combinación con código propio se puedan desarrollar aplicaciones que después puedan ser ejecutadas bajo el ambiente Windows.

Una vez presentados en este capítulo los conceptos teóricos que se consideraron relevantes para entender el desarrollo de este proyecto, el siguiente capítulo se enfoca en estudiar los componentes de hardware que componen el *datalogger*.