



S.E.P.

S.E.S.T.N.M.

T.N.M.

INSTITUTO TECNOLÓGICO DE TOLUCA

**“DIGITALIZACIÓN DE DATOS ANALÓGICOS
USANDO INSTRUMENTACIÓN VIRTUAL”**

**OPCIÓN: III
PROYECTO DE INVESTIGACIÓN**

**QUE PARA OBTENER EL TÍTULO DE:
INGENIERO ELECTRÓNICO**

**PRESENTA:
ISAAC MARTINEZ ESPINOSA
No. CONTROL: 08280147**

**ASESOR:
ING. GABRIEL LÓPEZ REYES**

Metepec, Estado de México, Febrero de 2016

Resumen

El proyecto de investigación propone un método para la adquisición de datos basado en un hardware de medidas, con una arquitectura formada por un solo procesador, complementado con un instrumento virtual.

El hardware de medidas está constituido por un solo IC (circuito integrado por sus siglas en inglés Integrated Circuit), un microcontrolador el cual es un dispositivo programable que contiene los periféricos necesarios para realizar las tareas de conversión de señales y envío de datos.

El instrumento virtual es desarrollado en la plataforma LabVIEW especializada en diseño y control de instrumentos virtuales, esta plataforma además de ser una poderosa herramienta tiene una gran versatilidad, ya que los complementos que esta posee, facilitan el uso de instrumentos genéricos, tal es el caso del sistema que se desarrolla.

El propósito de este proyecto es desarrollar un sistema de adquisición de datos, útil en instrumentación virtual, siendo una plataforma abierta, tal que pueda adecuarse a diversas aplicaciones que estén dentro de su rango de medida.

Abstract

The project presents a method for data acquisition, based on single processor measurements hardware, complemented with a virtual instrument.

The hardware constitute by a single IC, which is a microcontroller, a programmable device with all the necessary peripheral devices build in to successfully complete the signal conversion and data transfer tasks.

The development of the virtual instrument is accomplished using the LabVIEW platform, which specialized on virtual instruments control and design; LabVIEW is not just a powerful platform but also is a very versatile tool with several add-ons useful for generic instruments control.

The purpose of the project is the development of a data acquisition system useful in virtual instrumentation applications, such as an open platform suitable for instrumentation applications in which the variable span range does not exceeds the system range.

Índice de contenidos

Portada	
Resumen	i
Abstract	ii
Índice de contenidos	iii
Índice de figuras	v
Índice de tablas	vii
Introducción	1
Capítulo I. Fundamentos teóricos	4
1.1. Introducción	4
1.2. Instrumentación electrónica	4
1.2.1. Conceptos básicos	4
1.2.2. Errores en la medición	5
1.2.3. Instrumentación virtual	6
1.3. Señales discretas y señales continuas	7
1.4. Teorema de muestreo	8
1.4.1. Reconstrucción de una señal	9
1.4.2. Submuestreo	9
1.5. Convertidores ADC	10
1.5.1. Errores de conversión	13
1.6. Protocolos de comunicación	17
1.6.1. Protocolo USB	17
1.7. Lenguajes de programación	18
1.7.1. Niveles de programación	18
1.7.2. Compiladores	19
1.7.3. Programación modular	19
1.8. Microcontroladores	20

1.8.1. Arquitectura de computadoras	20
1.8.2. Programación de microcontroladores	21
Capítulo II. Método para el desarrollo del sistema de adquisición de datos	22
2.1. Introducción	22
2.2. Diseño del sistema	22
2.3. Elección de los componentes del sistema	23
2.4. Diseño del hardware de medidas	24
2.4.1. Desarrollo del hardware de medidas	25
2.5. Diseño del VI	38
2.5.1. Desarrollo del VI	39
2.6. Metodología de pruebas	42
Capítulo III. Resultados de las pruebas	46
3.1. Introducción	46
3.2. Resultados obtenidos de las pruebas	46
Capítulo IV. Conclusiones	61
4.1. Introducción	61
4.2. Conclusiones	61
4.3. Observaciones	62
4.4. Recomendaciones	63
4.5. La instrumentación dando forma al futuro	64
Glosario	67
Referencias y bibliografía	68
Anexo A. Códigos	71

Índice de figuras

Figura 1.1. Fluctuación de la presión acústica provocada por el habla	8
Figura 1.2. Diagrama a bloques del convertidor básico rampa en escalera	10
Figura 1.3. Diagrama a bloques del convertidor de aproximaciones sucesivas	11
Figura 1.4. Diagrama a bloques del convertidor paralelo	12
Figura 1.5. Diagrama a bloques del convertidor doble rampa	13
Figura 1.6. Error de offset positivo	14
Figura 1.7. Error de offset negativo	14
Figura 1.8. Error de ganancia positivo	15
Figura 1.9. Error de ganancia negativo	15
Figura 1.10. Error de linealidad integral	16
Figura 1.11. Error total de desajuste	16
Figura 1.12. Arquitectura Von Neumann	20
Figura 1.13. Arquitectura Harvard	21
Figura 2.1. Diagrama a bloques de la estructura un sistema de adquisición de datos	22
Figura 2.2. Rutina del microcontrolador	25
Figura 2.3. Diagrama de flujo del código 1	27
Figura 2.4. Diagrama de flujo del código 2	28
Figura 2.5. Circuito 1, esquemático del circuito eléctrico mínimo del sistema	29
Figura 2.6. Diagrama de flujo del código 3	30
Figura 2.7. Diagrama de flujo del código 4	31

Figura 2.8. Diagrama de flujo del código 5	32
Figura 2.9. Diagrama de flujo del código 6	35
Figura 2.10. Circuito 2, esquemático del circuito eléctrico con el aterrizamiento adecuado .	38
Figura 2.11. Señal generada por una salida digital	44
Figura 3.1. Diagrama de tiempos del muestreo realizado por el MCU	46
Figura 3.2. Efecto en la señal reconstruida cuando la frecuencia de la señal original se aproxima a la frecuencia de Nyquist	55
Figura 3.3. Efecto en la señal reconstruida cuando la frecuencia de la señal original es igual a la frecuencia de Nyquist	56
Figura 3.4. Señal reconstruida @ 50Hz	57
Figura 3.5. Señal reconstruida @ 100Hz	57
Figura 3.6. Señal reconstruida @ 10KHz	57
Figura 3.7. Interpolación de los datos de la señal @ 50Hz realizada en Excel	58
Figura 3.8. Interpolación de los datos de la señal @ 100Hz realizada en Excel	58
Figura 3.9. Interpolación de los datos de la señal @ 10KHz realizada en Excel	58
Figura 3.10. Señal reconstruida @ 12KHz	59
Figura 3.11. Interpolación de los datos de la señal @ 12KHz realizada en Excel	59

Índice de tablas

Tabla 3.1. Resultados de las pruebas del código 1 y VI 1 con el circuito 1	46
Tabla 3.2. Resultados de las pruebas del código 2 y VI 1 con el circuito 1	47
Tabla 3.3. Resultados de las pruebas del código 3 y VI 1 con el circuito 1	48
Tabla 3.4. Resultados de las pruebas del código 4 y VI 1 con el circuito 1	48
Tabla 3.5. Resultados de las pruebas del código 5 y VI 2 con el circuito 1	49
Tabla 3.6. Resultados de las pruebas del código 5.1 y VI 2 con el circuito 1	50
Tabla 3.7. Resultados de las pruebas del código 5.1 y VI 3 con el circuito 1	50
Tabla 3.8. Resultados de las pruebas del código 5.1 y VI 4 con el circuito 1	51
Tabla 3.9. Resultados de las pruebas del código 5.1 y VI 5 con el circuito 1	52
Tabla 3.10. Resultados de las pruebas del código 5.1 y VI 3 con el circuito 1	52
Tabla 3.11. Resultados de las pruebas del código 6 y VI 6 con el circuito 1	53
Tabla 3.12. Resultados de las pruebas del código 6.1 y VI 6 con el circuito 2	54
Tabla 3.13. Resultados de las pruebas del código 6.1 y VI 6.1 con el circuito 2	54
Tabla 3.14. Comparación entre los códigos y las mejoras obtenidas	60

Introducción

La observación de fenómenos y procesos a través de instrumentos de medición con la finalidad de obtener datos de su comportamiento es una práctica en la que comúnmente se recurre a la instrumentación electrónica como medio para desarrollar estos instrumentos, gracias al procesamiento de las computadoras tenemos una herramienta poderosa para el análisis y almacenamiento de información, el uso de estas herramientas en combinación con la instrumentación electrónica han llevado a la necesidad de mezclar el mundo analógico y el mundo digital, el resultado es la instrumentación virtual.

La flexibilidad que ofrece un instrumento virtual para el almacenamiento y procesamiento de mediciones o datos, obtenidos del análisis de un fenómeno, no sería posible de no existir una forma de digitalizar datos, esta labor se le confía a un sistema de adquisición de datos, cuyos inicios datan de la década de los setenta, la instrumentación virtual comenzó con la creación del protocolo HP-IB (Hewlett-Packard Instrument Bus) utilizado para controlar dispositivos de prueba y medición, que después se convertiría en GPIB (General Purpose Instrumentation Bus), hasta 1978 cuando fue estandarizado como IEEE-488 (nombre del estándar emitido por la IEEE the Institute of Electrical and Electronics Engineers).

Si bien en la década de los setenta es que se reconoce formalmente a estas prácticas como instrumentación virtual, las bases matemáticas que dan forma a la digitalización de lo que en electrónica entendemos como señales analógicas (en un contexto matemático funciones continuas), se atribuyen a **Harry Nyquist** que en **1928** presenta el documento “**Certain topics in telegraph transmission theory**” en el cual propone el teorema que más tarde sería demostrado por **Claude E. Shannon** en **1949**, dicho teorema es conocido hoy como “**Teorema de muestreo de Nyquist-Shannon**”.

Actualmente entendemos la diferencia entre señales discretas y continuas como parte del día a día en la electrónica, pero sabemos que vivimos en un mundo de fenómenos y procesos continuos (de señales analógicas) que cuando son instrumentados generan una cantidad de datos infinita en el tiempo, a esto se le conoce como “**Big analog data**” este

concepto representa esa cantidad de datos infinitos que se obtienen de digitalizar señales provenientes de los sensores que registrando variables en los procesos o fenómenos, la demanda de datos marca una tendencia en estas prácticas, que se conocen como “**El internet de las cosas (The internet of things)**”.

Este proyecto tiene por objetivo **crear un método para la adquisición de datos a través de instrumentación virtual y sistemas digitales embebidos**, con una arquitectura abierta y escalable, esto se logrará mediante las siguientes actividades:

- Diseño y construcción de un hardware de medidas.
- Diseño y programación del modulo de software correspondiente al instrumento virtual.
- Planeación y ejecución de métodos de mejora continua y evaluación de resultados.

Con base en las siguientes preguntas, se define una hipótesis centrada en instrumentación virtual y sistemas digitales embebidos:

¿Cuál es el método para desarrollar un sistema de adquisición de datos?

¿Es un microcontrolador un elemento adecuado para desarrollar un sistema de adquisición de datos?

¿Cuál es el mejor protocolo de comunicación para un sistema de adquisición de datos?

¿Cuál es la arquitectura de software que requiere un sistema de adquisición de datos?

Hipótesis:

Optimizar un proceso de adquisición de datos con el uso de instrumentación virtual y sistemas digitales embebidos, hasta alcanzar la máxima eficiencia posible.

El proyecto está delimitado en sus diferentes aspectos técnicos como se enlista a continuación:

- En sus mediciones. Se limita a las mediciones que puedan ser obtenidas con las restricciones que impone el convertidor analógico-digital embebido en el microcontrolador seleccionado, es decir, no se utilizaran dispositivos externos.
- En el hardware. Se excluirá la etapa de acondicionamiento de señales de propósito específico y la etapa de sensores.
- En los alcances de la mejora continua. Se limitan las mejoras del sistema a las restricciones que impone el microcontrolador seleccionado, es decir, no se migrara a una arquitectura o dispositivo diferente para continuar el proceso de mejora.
- En la evaluación de resultados se limita a evaluar el rendimiento del sistema en relación a las mejoras aplicadas o a las fallas que se presenten, a fin de conocer los resultados del proceso de mejora continua. No se evaluarán los atributos que caracterizan al instrumento o a la medición (tales como exactitud, precisión, incertidumbre, linealidad, y error), se sabe de las especificaciones del convertidor analógico-digital algunos de los atributos que caracterizan a la medición (resolución y linealidad) sin embargo evaluar dichos atributos no es el propósito central de este proyecto.

Capítulo I. Fundamentos teóricos

1.1. Introducción

En este capítulo se tratan las consideraciones teóricas necesarias para el desarrollo de un sistema de adquisición de datos, teniendo en cuenta que dicho proceso comprende diversas áreas de la electrónica, por lo que solo se tratarán los temas principales relacionados con la instrumentación, digitalización de datos y electrónica digital, las demostraciones matemáticas de dichos principios están disponibles en [1] por lo que no serán tratadas en esta sección.

1.2. Instrumentación electrónica

La incapacidad del humano para evaluar la magnitud de una variable a través de la percepción sensorial hace necesario el uso de instrumentos de medición, el proceso de medición es un proceso de comparación, por lo que un instrumento puede definirse como un medio físico para la determinación de la magnitud de una variable mediante un proceso de comparación, por lo tanto un instrumento electrónico es un instrumento que basa su funcionamiento en principios eléctricos o electrónicos para la determinación de la magnitud de una variable. La instrumentación es una rama de la electrónica dedicada al diseño e implementación de instrumentos y técnicas de medición de variables físicas, valiéndose de principios o fenómenos eléctricos o electrónicos, mismos que se desarrollan en este proyecto.

1.2.1. Conceptos básicos

En esta sección se presentan los atributos que caracterizan una medición y a un instrumento [2], los cuales por su uso son necesarios para el proyecto, los más importantes son:

- Rango de medida. Es el espectro de valores desde el máximo y hasta el mínimo, que el instrumento es capaz de medir.
- Alcance. Es la diferencia algebraica entre los valores superior e inferior del campo de medida.

- Error. Es la desviación a partir del valor real de la variable medida.
- Incertidumbre. Es la dispersión de los valores que pueden ser atribuidos razonablemente al verdadero valor de la variable medida.
- Exactitud. Es la aproximación al valor real de la variable medida.
- Precisión. Es la proximidad entre un grupo de mediciones realizadas al mismo valor de la variable bajo las mismas condiciones.
- Zona muerta. Es la zona en la que un cambio de la variable medida no producirá una salida o cambio en la medición del instrumento.
- Sensibilidad. Es la razón entre el cambio en la salida o la respuesta del instrumento y el cambio en la entrada o la magnitud de la variable medida.
- Repetibilidad. La capacidad de reproducir la misma lectura al medir repetidamente valores idénticos en las mismas condiciones, en el mismo sentido de variación.
- Histéresis. Es la máxima diferencia observada en la lectura del instrumento para el mismo valor cualquiera del campo de medida, cuando la variable recorre la escala completa en sentido ascendente y descendente.
- Resolución. Es el cambio más pequeño en la magnitud de la variable medida que puede ser registrado por el instrumento.
- Linealidad. Es la aproximación de una curva de calibración a una línea recta especificada.
- Estabilidad. Es la capacidad del instrumento para mantener su comportamiento especificado a lo largo de su vida útil.

1.2.2. Errores en la medición

Existen errores inherentes a una medición, dichos errores se pueden evitar o minimizar su efecto según sea el caso, para ello deben conocerse sus causas. A continuación se presenta una clasificación general de los tipos de errores en mediciones eléctricas [3].

Errores gruesos. Generalmente de origen humano, como malas lecturas, ajuste incorrecto, aplicación inapropiada, errores en los cálculos. Estos errores no pueden ser tratados matemáticamente, se pueden evitar procurando buenas prácticas al hacer mediciones [3].

Errores sistemáticos. Se deben a fallas en los instrumentos, como desgaste en los instrumentos, efectos ambientales o mala calibración del instrumento. Estos errores se pueden identificar verificando la estabilidad y repetibilidad del instrumento, en busca de comportamiento errático. Los errores relacionados con el instrumento se pueden evitar comparándolo con un instrumento patrón, y aplicando los factores de corrección después de determinar la cantidad del error instrumental. Los errores por efectos ambientales pueden producirse por humedad, presión atmosférica, campos electromagnéticos, temperatura; para minimizar estos efectos las medidas a tomar son instalación de aire acondicionado, sellado hermético en ciertas partes del instrumento, y blindaje electromagnético. Los errores sistemáticos también se subdividen en estáticos o dinámicos, los errores estáticos ocurren por las limitaciones del dispositivo o de las leyes físicas que gobiernan su comportamiento, los errores dinámicos se producen cuando el instrumento no responde con la rapidez suficiente a los cambios de la variable [3].

Errores aleatorios. Son errores a los que no se les puede atribuir directamente una causa debido a variaciones aleatorias en los parámetros. En experimentos bien diseñados no debe haber muchos errores aleatorios, aunque el instrumento se encuentre bien calibrado y se opere en condiciones adecuadas las lecturas varían ligeramente durante el periodo de observación, esta variación no se puede corregir por ningún método de calibración o de control, ni se puede atribuir una causa hasta realizar una investigación meticulosa, la única forma de compensar este error es tomando más muestras y usar métodos estadísticos para aproximar a su valor real [3].

1.2.3. Instrumentación virtual

Un instrumento virtual o VI (por sus siglas en inglés Virtual Instrument) es un modulo de software que simula el panel frontal de un instrumento, apoyándose en elementos de hardware que son accesibles para la computadora como tarjetas de adquisición DSP en rack basados en VXI PXI o compactRIO, que realizan una serie de mediciones como si se tratara de un instrumento real. Cuando se ejecuta un programa que funciona como un instrumento virtual el usuario ve en la pantalla un panel cuyas funciones son idénticas a las de un

instrumento real. Los comienzos de la instrumentación virtual tienen lugar en la década de los setentas con la introducción del protocolo IEEE-488 o GPIB [4].

Existen varias plataformas especializadas en el diseño y control de instrumentos virtuales estas plataformas son una herramienta de programación de alto nivel o programación grafica con las cuales es posible diseñar y ejecutar el modulo de software que será utilizado como instrumento virtual para realizar mediciones. Algunas de las plataformas más comunes para este propósito son LabVIEW, Agilent-VEE, Cyber tools, Beta instruments manager. Aunque también es posible diseñar el modulo de software en algún otro lenguaje como Java o Visual Basic.

La adquisición de datos es el proceso de medir mediante una computadora fenómenos eléctricos o físicos, un sistema de adquisición de datos consiste en sensores, hardware de medidas y una computadora con software programable [4].

1.3. Señales discretas y señales continuas

Una señal describe un fenómeno físico, generalmente la información contenida en una señal se manifiesta como una variación de una magnitud en función de una variable que forma un patrón determinado, un ejemplo es el habla, cuya señal es una fluctuación de la presión acústica en función del tiempo, en la figura 1.1 se aprecia la fluctuación de la presión acústica como función del tiempo, las señales se representan matemáticamente como funciones de una o más variables independientes [1].

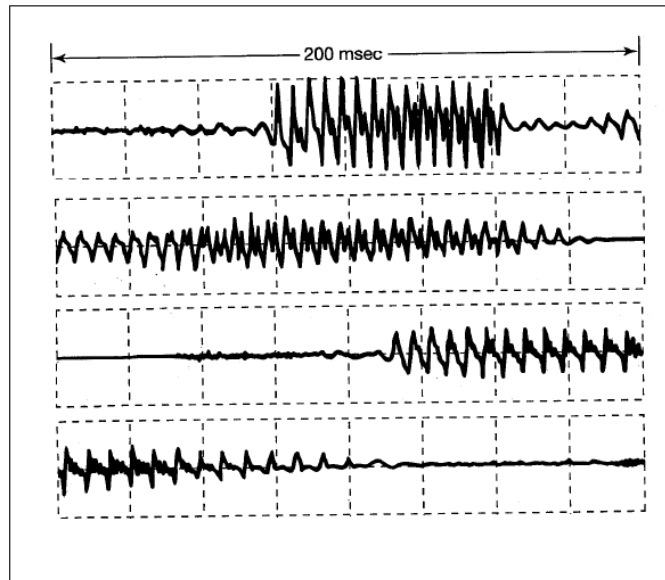


Figura 1.1. Fluctuación de la presión acústica provocada por el habla.

Las señales pueden clasificarse en dos tipos, señales discretas y señales continuas, la señales continuas se caracterizan por que la variable independiente es continua, estas señales se definen para una sucesión continua de valores de la variable independiente, las señales discretas solo están definidas en tiempos discretos es decir la variable independiente solamente toma valores de un conjunto discreto [1].

1.4. Teorema de muestreo

El teorema de muestreo fue postulado por H. Nyquist en 1928 en el documento titulado "Certain topics in telegraph transmission theory". En el cual se postula que una señal continua puede representarse y reconstruirse por completo partiendo de sus muestras tomadas en puntos igualmente espaciados en el tiempo. Gran parte de la importancia del teorema de muestreo se debe a su papel como puente entre las señales continuas y las discretas [1]. El teorema de muestreo dice lo siguiente:

Sea $x(t)$ una señal de banda limitada con $X(j\omega) = 0$ para $|\omega| > \omega_M$. Entonces $x(t)$ se determina unívocamente mediante sus muestras $x(nT), n = 0, \pm 1, \pm 2, \dots$, si

$$\omega_s > 2\omega_M \text{ --- (1)}$$

Donde:

$$\omega_s = \frac{2\pi}{T} \text{ --- (2)}$$

$$\omega_s \equiv \text{frecuencia de muestreo}$$

$$\omega_M \equiv \text{frecuencia de Nyquist}$$

$$T \equiv \text{periodo de muestreo}$$

Dadas estas muestras podemos reconstruir $x(t)$ generando un tren de impulsos periódicos en el cual los impulsos sucesivos tengan amplitudes que corresponden a valores de muestras sucesivas. Este tren de impulsos es entonces procesado a través de un filtro pasa bajas ideal con ganancia T cuya frecuencia de corte sea mayor que ω_M y menor que $\omega_s - \omega_M$. La señal de salida resultante será exactamente igual a $x(t)$.

1.4.1. Reconstrucción de una señal

Un método comúnmente utilizado para reconstruir una señal de manera aproximada a partir de sus muestras es la interpolación, un procedimiento simple de interpolación es el uso de un retenedor de orden cero, otro método es la interpolación lineal mediante la cual puntos de muestras adyacentes se conectan por una línea recta, si bien la interpolación lineal puede dar como resultado una reconstrucción burda de la señal, habrá casos en los que sea suficiente, sin embargo mediante el uso de filtros pasa bajas y métodos de interpolación con polinomios de mayor orden es posible reconstruir las señales de manera más exacta [1].

1.4.2. Submuestreo

El efecto de submuestreo da como resultado frecuencias altas reflejadas en frecuencias más bajas, dado que existe un límite para el cual el teorema de muestreo funciona en relación a la componente de frecuencia de la señal muestreada, la representación univoca de la señal mediante sus muestras será posible **si y solo si la frecuencia de muestreo es mayor que**

el doble de la máxima frecuencia muestreada, de no cumplirse esta condición habrá un traslape y se presentaran los efectos del submuestreo dicho efecto también es conocido como “efecto aliasing”, y de manera análoga a las frecuencias reflejadas se les nombra frecuencias o señales “alias” [1].

1.5. Convertidores ADC

Por sus siglas en ingles Analog to Digital Converter. Los convertidores ADC efectúan la conversión de una señal analógica de voltaje en un número binario equivalente al valor instantáneo de la señal en el momento en que es tomada la muestra. A continuación se presentan los tipos de convertidores y su funcionamiento.

Convertidor rampa en escalera. Su funcionamiento se basa en la comparación de la señal analógica de entrada con una señal de rampa definida. Un pulso de inicio en la lógica de control da inicio a un contador inicializado en cero y a su salida estará el código binario correspondiente al cero esta salida pasara a la entrada de un DAC (convertidor digital-analógico) para poner en cero sus salida esta señal es usada como referencia para un comparador que compara la magnitud de la señal de entrada contra la generada por el contador, cuando el voltaje generado por el DAC sea mayor que la señal de entrada el contador se detendrá de lo contrario aumentara en un bit para seguir generando voltajes en el DAC y continuar el proceso de comparación, cuando el contador se detiene se tiene la conversión de la señal de entrada la cual será la cuenta inmediata anterior a la cuenta actual del contador [5], en la figura 1.2 se muestra el diagrama a bloques del convertidor rampa en escalera.

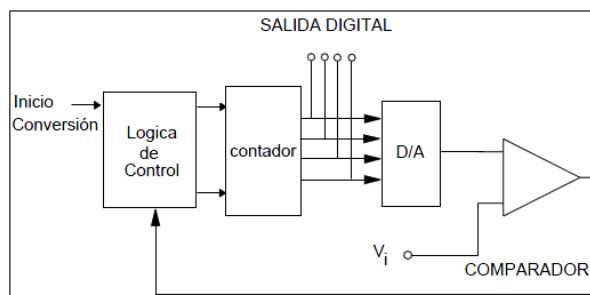


Figura 1.2. Diagrama a bloques del convertidor básico rampa en escalera.

Convertidor de aproximaciones sucesivas. Este convertidor combina una buena precisión con tiempos de conversión rápidos. Su funcionamiento se basa en la comparación de la señal de entrada con una generada por un contador ascendente y descendente. El contador pone el bit más significativo en uno y los demás en cero esta salida del contador pasa a un DAC para generar un voltaje que será comparado con la señal de entrada si el voltaje generado por el DAC es mayor que la señal de entrada el contador cambia a uno el bit inmediato anterior al bit más significativo y los demás a cero y así sucesivamente hasta que encuentra una secuencia tal que resulte menor a la señal de entrada cuando esto ocurre el bit se mantendrá en uno y el bit inmediato anterior cambiara de cero a uno y se repite el proceso hasta haber probado con cada bit del contador en uno [5], en la figura 1.3 se muestra el diagrama a bloques del convertidor de aproximaciones sucesivas.

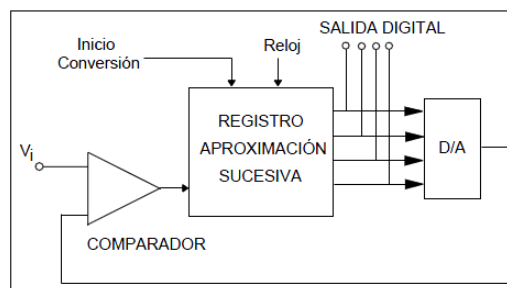


Figura 1.3. Diagrama a bloques del convertidor de aproximaciones sucesivas.

Convertidor paralelo (flash). Una velocidad de conversión rápida es el objetivo principal de esta arquitectura. Su funcionamiento se basa en varios amplificadores operacionales como comparadores dispuestos en paralelo a la entrada de un decodificador a los cuales se aplica la señal a convertir y una señal de referencia que también es aplicada a una red de resistencias en serie del mismo valor, dando como resultado una diferencia de uno en términos del bit menos significativo entre compradores sucesivos, por la complejidad de su arquitectura no son prácticos comparadores con resolución de más de 16bits [5], en la figura 1.4 se muestra el diagrama a bloques del convertidor paralelo.

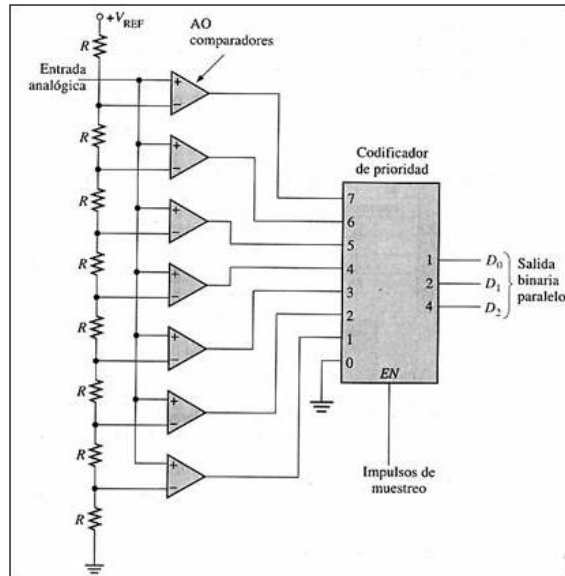


Figura 1.4. Diagrama a bloques del convertidor paralelo.

Convertidor doble rampa. Son comparadores de alta precisión pero con tiempos de conversión lenta su velocidad de muestreo típica es de 10 muestras por segundo, pueden alcanzar resoluciones de 18 a 20 bits y tiene inmunidad al ruido. Este dispositivo consiste en un integrador electrónico para dos señales, la que se va a convertir y una señal de referencia, un switch mantiene conectada en todo momento a una de las dos señales al integrador, otro switch en capacitor lo mantiene conectado o no, la actuación coordinada de estos switches genera a la salida una señal de doble rampa una ascendente y una descendente [5], en la figura 1.5 se muestra el diagrama a bloques del convertidor doble rampa, el cálculo de la señal digitalizada se basa en los tiempos de subida y bajada de acuerdo con la ecuación siguiente:

$$\frac{t_s}{t_m} = \frac{V_{ref}}{V_a} \text{ --- (3)}$$

Donde:

$t_s \equiv$ tiempo de subida

$t_m \equiv$ tiempo de muestreo

$V_{ref} \equiv$ Voltaje de referencia

$V_a \equiv$ Voltaje de la señal analógica

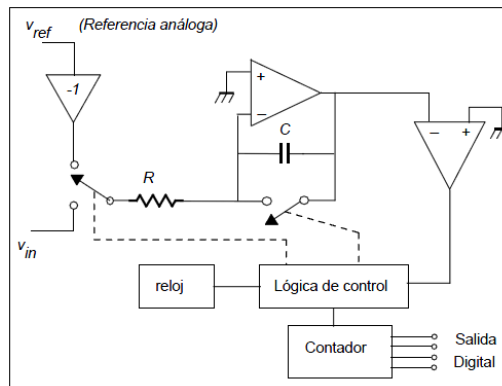


Figura 1.5. Diagrama a bloques del convertidor doble rampa.

Convertidor serie. Este tipo de convertidores entregan el resultado de la conversión como una secuencia de bits en forma serial. Su funcionamiento se basa en un capacitor conmutado [5].

1.5.1. Errores de conversión

Adicionalmente a la pérdida de información inherente al proceso de digitalización mediante un convertidor ADC, existen cierto tipo de condiciones que pueden generar errores en la conversión, este tipo de errores se aprecian como muestras erradas del valor instantáneo de la señal, dado que las muestras son tomadas de manera continua es durante el proceso de reconstrucción de la señal que se agudiza el problema, ahora la serie de muestras tomadas ya no representan unívocamente a la señal original. A continuación se mencionan los tipos de error asociados a la conversión analógico-digital [6].

Error de offset. Se define como la desviación entre la primera transición real y la primera transición ideal. La primera transición es cuando la salida digital del convertidor cambia de cero a uno, en el caso ideal debe tenerse una salida digital de uno cuando la entrada analógica está entre 0.5 LSB y 1.5 LSB, la primera transición en el caso ideal debe ocurrir

cuando la entrada analógica alcance el valor de 0.5 LSB. Las figuras 1.6 y 1.7 representan el caso de error de offset positivo y negativo respectivamente [6].

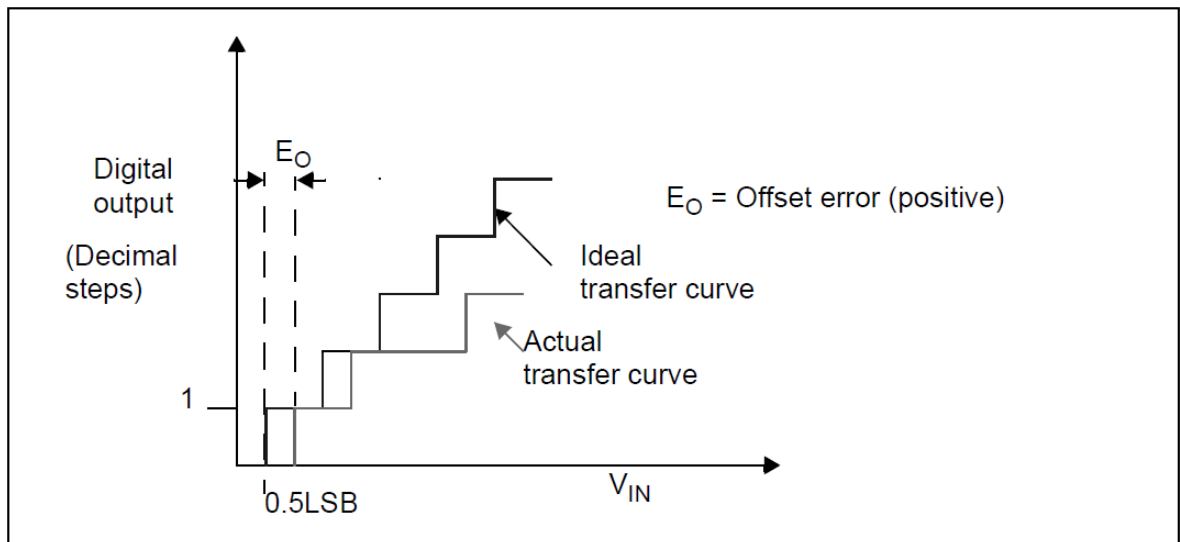


Figura 1.6. Error de offset positivo.

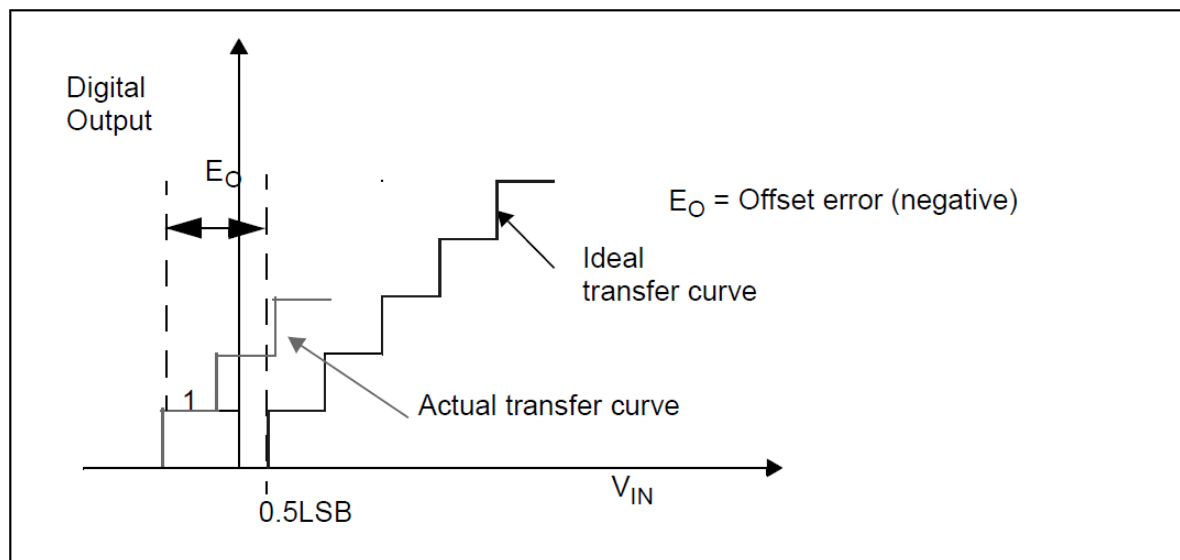


Figura 1.7. Error de offset negativo.

Error de ganancia. Se define como la desviación entre la última transición real y la última transición ideal. La última transición real es el cambio en la salida digital de 3FE a 3FF en un convertidor de 10-bits, en el caso ideal se tiene este cambio cuando la señal analógica toma el valor igual a la diferencia del nivel de referencia positivo menos 0.5 LSB, si el nivel de

referencia positivo fuese 5V entonces este cambio ocurriría a 4997.12mV. Las figuras 1.8 y 1.9 representan el caso de error de ganancia positivo y negativo respectivamente [6].

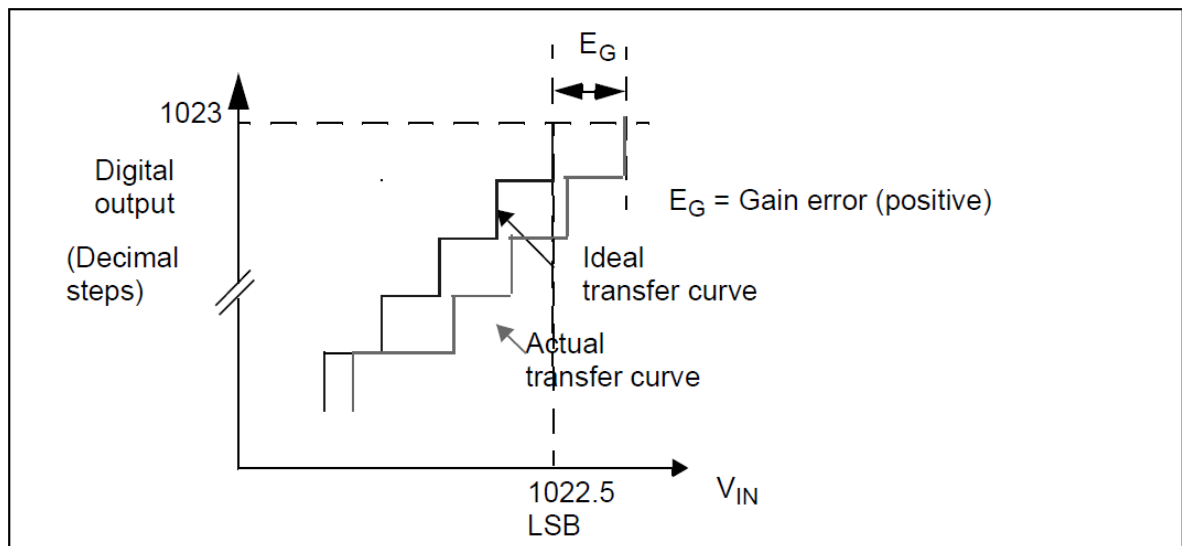


Figura 1.8. Error de ganancia positivo.

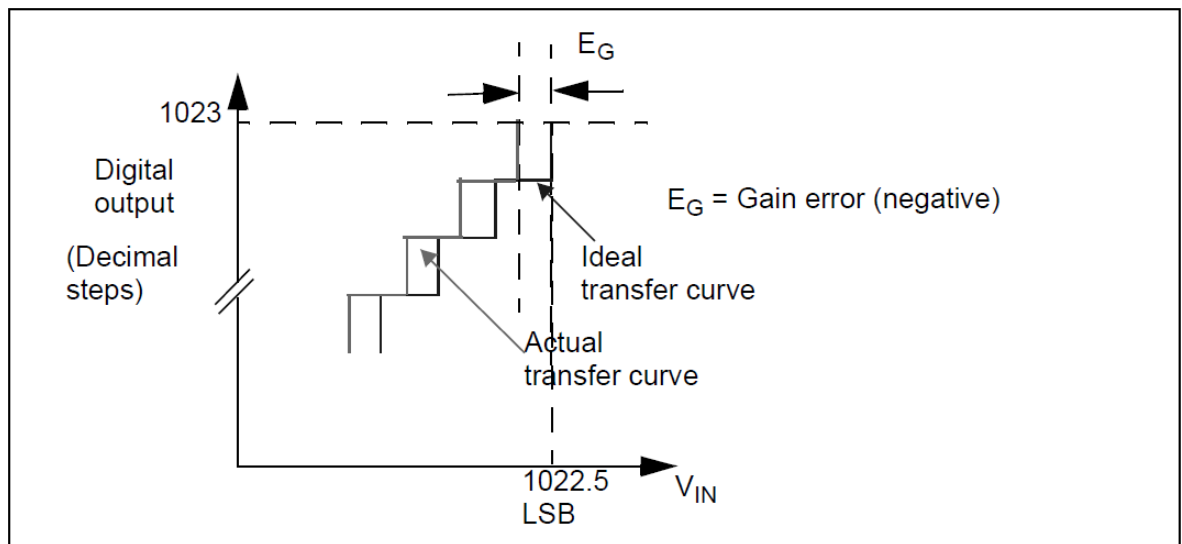


Figura 1.9. Error de ganancia negativo

Error de linealidad diferencial. Es definido como la máxima desviación entre incrementos reales e incrementos ideales. Idealmente un voltaje analógico de entrada de 1 LSB debería causar un cambio en el código digital, si el voltaje requerido para realizar este cambio es mayor a 1 LSB entonces el convertidor tiene error de linealidad diferencial [6].

Error de linealidad integral. Es la máxima desviación entre cualquier transición y el punto final de la línea de correlación. El punto final de la línea de correlación se define como la línea de la curva de la función de transferencia del ADC que conecta a la primera transición con la última transición, el punto final de la línea de correlación se define en base a las transiciones reales por lo tanto no tiene relación con curva de la función de transferencia ideal. La figura 1.10 muestra una representación de este tipo de error [6].

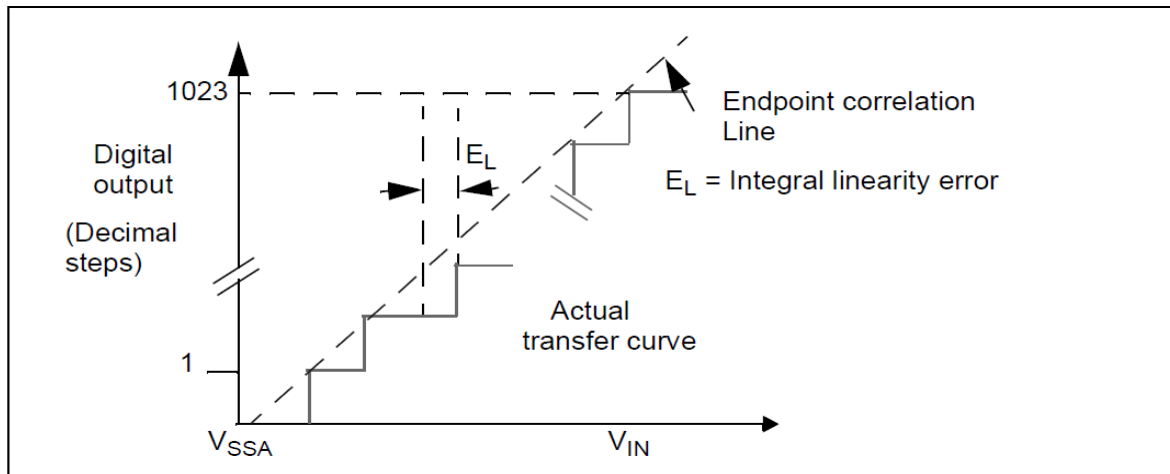


Figura 1.10. Error de linealidad integral.

Error total de desajuste. Se define como la desviación máxima entre la curva de la función de transferencia real e ideal en otras palabras es la máxima desviación registrada entre el valor ideal esperado y el valor real obtenido del ADC para cualquier valor de entrada, La figura 1.11 muestra este tipo de error [6].

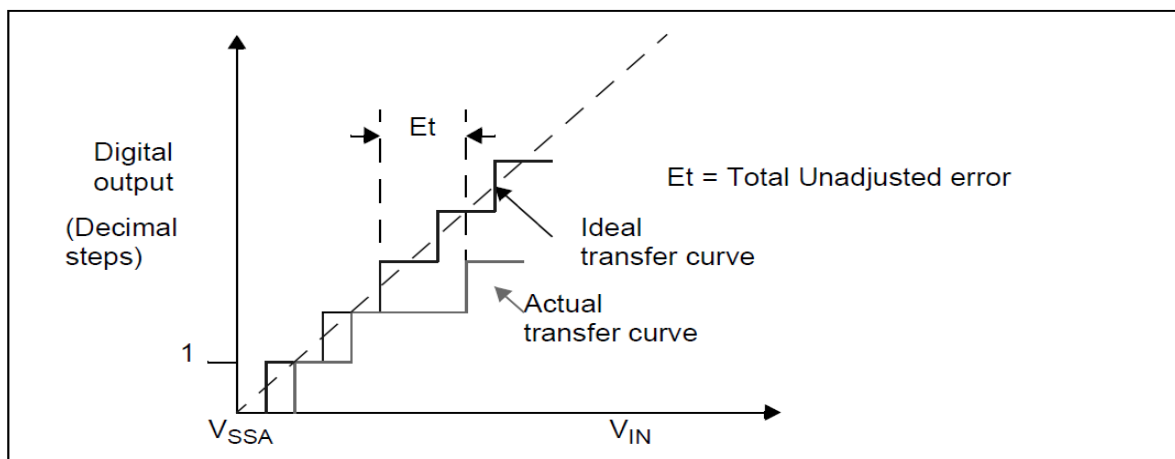


Figura 1.11. Error total de desajuste.

1.6. Protocolos de comunicación

Un protocolo de comunicación es una serie de reglas para mantener la comunicación entre dos entidades, las reglas son un estándar de la semántica, sintaxis y sincronización de la información contenida en el mensaje y en algunos casos también del manejo de errores, el canal por el cual se transmite el mensaje puede ser cualquier variación de una magnitud física, en particular para los fines de estudio de la electrónica, solo se transmite a través de la variación de magnitudes eléctricas. En electrónica los protocolos de comunicación pueden ser implementados por hardware o por software.

1.6.1. Protocolo USB

El protocolo de comunicaciones USB obtiene su nombre de sus siglas en inglés Universal Serial Bus, es un protocolo de comunicación serial. El objetivo de este protocolo es el de proveer un bus estándar, es decir que pueda ser utilizado para la transmisión de datos entre computadoras y dispositivos de cualquier tipo, actualmente existe un organismo regulador del protocolo el cual asigna certificaciones e identificadores a dispositivos y fabricantes, el protocolo fue desarrollado en la década de los noventa por un grupo de compañías que buscaban una forma de conectar periféricos a sus computadoras, la versión 1.0 se publicó en 1996 y en 1998 la versión 1.1 a partir de este momento se volvió una práctica común el uso de dicho bus [7].

El protocolo de comunicación USB estandariza los tipos de conexión, cables, distancia de los cables, tipos de conectores, la topología y la arquitectura lógica necesarias para la utilización del bus. La arquitectura lógica por su nombre en inglés se le conoce como arquitectura pipeline o de tuberías, la topología del bus se basa en tres elementos esenciales Hub, dispositivo y Host o Root Hub los cuales administran las conexiones eléctricas y el flujo de información, la arquitectura pipeline es la ruta lógica y determina el sentido de la información entre dispositivo y hub. Dentro del estándar USB existen diferentes clases y tipos de transferencia apropiadas para diferentes usos o dispositivos, cada una con capacidades y características diferentes [7], los tipos de transferencia son:

- Interrupt
- Control
- Bulk
- Isochronous

Las clases de dispositivos son tan diversos como su aplicación y van desde dispositivos de almacenamiento, de cuidado de la salud y hasta dispositivos de pruebas y mediciones.

1.7. Lenguajes de programación

Un lenguaje de programación es el conjunto de símbolos palabras y reglas que permiten la interacción entre personas y computadoras, estos lenguajes son utilizados para ejecutar algoritmos en una computadora mediante instrucciones de entrada o salida, calculo o manipulación de textos, comparaciones lógica, almacenamiento y recuperación. Los lenguajes de programación son empleados para realizar programas o algoritmos de computadora con el fin de procesar información, dicho programa deberá estar albergado en una memoria de tipo no volátil para que la computadora pueda ejecutarlo cuando sea necesario.

1.7.1. Niveles de programación

Existen tres niveles de programación esta clasificación se asigna a los lenguajes de programación teniendo en cuenta que tan parecidos son al lenguaje humano y que tan cerca están de la lógica de circuitos. A continuación se mencionan los niveles de programación.

Nivel alto. Este nivel es el más alejado de la lógica de circuitos, siempre se usa un compilador y algunos de ellos están hechos para la programación grafica o lenguaje grafico, las ventajas de este tipo de programación son referentes a un entorno de programación fácil de usar.

Nivel medio. Los lenguajes de programación en este nivel siempre requieren un compilador, el lenguaje es un código basado en texto con una sintaxis y caracteres específicos, este tipo de lenguajes combinan un entorno relativamente fácil de usar y una mayor proximidad a la lógica de circuitos que los lenguajes de alto nivel.

Nivel bajo. Los lenguajes de bajo nivel son los más cercanos a la lógica de circuitos o lenguaje máquina, son los más difíciles de programar, la ventaja es referente al control directo sobre los circuitos y módulos del microprocesador por lo que el programa puede ser muy eficiente al aprovechar todos los recursos del microprocesador, a diferencia de los otros dos niveles donde esto no es posible.

1.7.2. Compiladores

Un compilador es un programa que transforma el código escrito en un determinado lenguaje de programación a la lógica de circuitos o lenguaje máquina para que este pueda ser ejecutado por la computadora. A un nivel de programación más alto corresponde una transformación mayor para llegar al lenguaje máquina, en sentido figurado entre más alto el nivel de programación el compilador recorre un camino más largo, por esta razón es que con un lenguaje más bajo existe la posibilidad de analizar y hacer más eficiente el programa.

1.7.3. Programación modular

Cuando un programa es muy extenso o complejo existen diferentes métodos para desarrollarlo uno de ellos es la programación modular la cual consiste en subrutinas, subprogramas o funciones las cuales son llamadas dentro del programa o de la rutina principal para ejecutar un algoritmo específico y el resultado de dicho proceso alimentará a otra subrutina o proceso que le sucede en orden de ejecución, esta técnica es empleada para reducir el nivel de complejidad del programa principal.

1.8. Microcontroladores

Los microcontroladores son dispositivos electrónicos programables que de manera semejante a una computadora poseen una unidad de procesamiento, memorias y periféricos de entrada y salida. El primer microcontrolador se desarrollo en la década de los setentas por el fabricante Intel.

1.8.1. Arquitectura de computadoras

Estos dispositivos igual que una computadora obedecen a dos tipos de arquitecturas de las cuales se deriva métodos diferentes para la programación de los dispositivos. Como se menciona en la sección lenguajes de programación es necesario suministrar a la computadora con las instrucciones para la ejecución de un algoritmo en lenguaje maquina de igual manera deben ser programados los microcontroladores, dado que existen dos arquitecturas diferentes la programación de cada arquitectura es diferente ya que la lógica de los circuitos es diferente por los tanto es necesario conocer las arquitecturas para poder programar estos dispositivos, a continuación se mencionan los dos tipos de arquitecturas principales.

Arquitectura Von Neumann. Esta arquitectura lleva su nombre por el matemático John Von Neumann quien postulo esta arquitectura para el diseño una computadora digital en el año de 1945, su característica principal es la existencia de una memoria única (utilizada tanto como memoria de programa y memoria de datos) y un bus de datos único, en la figura 1.12 se muestra la arquitectura Von Neumann.

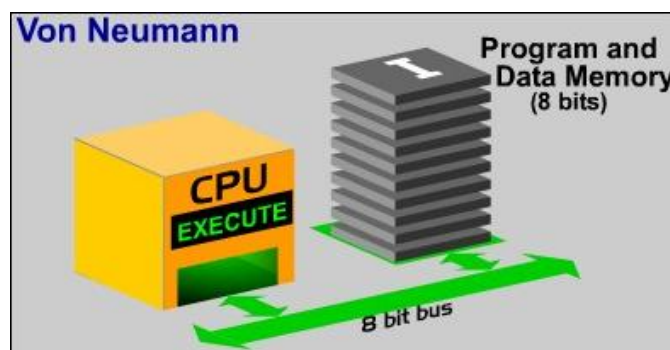


Figura 1.12. Arquitectura Von Neumann.

Arquitectura Harvard. Esta arquitectura es llamada así por la computadora Harvard Mark I, a diferencia de la arquitectura Von Neumann esta posee una memoria de programa independiente de la memoria para datos y cada una con un bus independiente, en la figura 1.13 se muestra la arquitectura Harvard.

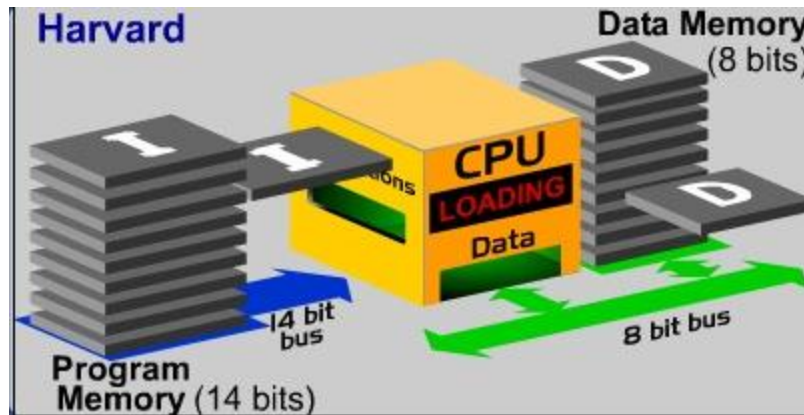


Figura 1.13. Arquitectura Harvard.

1.8.2. Programación de microcontroladores

Dado que existen arquitecturas diferentes existe una serie de instrucciones diferente para cada arquitectura, particularmente para cada dispositivo dado que cada fabricante diseña diferentes arquitecturas. Esta serie de instrucciones es comúnmente conocida como lenguaje ensamblador ó llanamente como ensamblador, no debe confundirse con el lenguaje máquina, y es necesario un compilador para transformar el lenguaje ensamblador a la lógica de circuitos, siendo este lenguaje uno de los lenguajes de más bajo nivel es posible identificar áreas de oportunidad para hacer más eficientes las rutinas ya programadas. Adicionalmente al lenguaje ensamblador hoy en día existen diversos compiladores que utilizan otros lenguajes de nivel medio para programar microcontroladores.

Este capítulo es una referencia de términos y conceptos indispensables para el desarrollo de este proyecto.

Capítulo II. Método para el desarrollo del sistema de adquisición de datos

2.1. Introducción

Este capítulo muestra el desarrollo del proyecto, mediante la documentación de los métodos de diseño, de mejora continua, y de evaluación de resultados, que fueron utilizados para desarrollar el hardware de medidas y el VI.

2.2. Diseño del sistema

Tal como lo plantea la hipótesis se pretende aplicar un proceso de mejora continua para alcanzar la máxima eficiencia posible, por lo que más adelante se establecerán directrices de diseño con el fin de encaminar las decisiones y acciones en aras de obtener la máxima eficiencia posible.

El diseño del sistema parte de las funciones que este debe realizar, en el capítulo anterior se define con precisión que es un sistema de adquisición de datos y el papel que desempeña, con base en dicho fundamento se presenta en la figura 2.1 un diagrama a bloques que muestra de manera general la estructura del sistema de adquisición de datos.

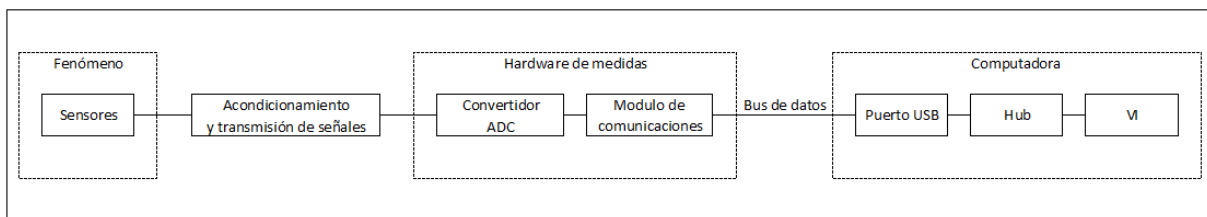


Figura 2.1. Diagrama a bloques de la estructura del sistema de adquisición de datos.

En la figura 2.1 se observa que el sistema debe realizar tres procesos indispensables, que se enlistan a continuación:

1. Convertir las señales analógicas en datos digitales.
2. Transmitir los datos del hardware de medidas a una computadora.

3. Mostrar los datos en un VI.

Estos procesos deben ser realizados haciendo uso de los siguientes elementos y subsistemas:

- Convertidor ADC.
- Modulo de comunicaciones.
- Bus de datos.
- Puerto USB.
- Hub de la computadora.
- Modulo de software del VI.

Como se observa en la figura 2.1 estos subsistemas están contenidos en el hardware de medidas y en la computadora que alberga al VI, y para hacer uso de estos deben seleccionarse de manera específica los dispositivos y las herramientas que se van a emplear, a continuación se muestra la selección que se ha hecho.

2.3. Elección de los componentes del sistema

Hardware de medidas. Para el hardware de medidas son necesarios dispositivos que realicen las tareas que se muestran en la figura 2.1, por lo que se emplea un microcontrolador, ya que es un dispositivo programable que integra los periféricos necesarios para realizar estas tareas mediante la programación de una rutina, dicho dispositivo es un MCU (unidad de microcontrolador por sus siglas en inglés Micro Controller Unit) con matrícula PIC18F4550 del fabricante MICROCHIP, se ha seleccionado ya que están embebidos todos los componentes necesarios, es decir, contiene dentro del mismo encapsulado los dispositivos periféricos necesarios a diferencia de otros microcontroladores a los cuales es necesario añadir módulos de comunicaciones externos para hacer uso de los protocolos de comunicación.

Bus de datos. Es necesario seleccionar un protocolo y una vía de comunicación para hacer posible el envío de datos entre el hardware de medidas y el VI, se utiliza el protocolo USB 2.0 ya que no se limita a un protocolo de comunicación universal sino es también un bus de datos universal que poseen todas las computadoras y hace posible el uso del sistema con cualquier computadora.

Instrumento virtual. Para el desarrollo del VI se utiliza la plataforma LabVIEW 2011 por sus complementos que hacen posible crear un bus de instrumentación para instrumentos genéricos.

2.4. Diseño del hardware de medidas

El hardware como se indica previamente se aprecian sus funciones principales en la figura 2.1, y su desarrollo se presenta a continuación.

Para el diseño de este, se mantiene el enfoque que parte de las funciones del subsistema y considerando la elección previa de los dispositivos, se tienen las siguientes directrices en el diseño del hardware de medidas:

- El hardware de medidas se limita a un solo procesador, a pesar de ser posible la implementación del procesamiento paralelo en donde los procesos se ejecutan en diferentes procesadores, se limitara a un solo microcontrolador y la ejecución de los procesos será secuencial debido a la dependencia de datos.
- El hardware es un dispositivo USB Bus power only, para satisfacer las especificaciones del protocolo y realizar el envío de datos el MCU deberá estar debidamente configurado como un dispositivo USB y cumplir con las especificaciones de tipos de transferencias, las especificaciones del protocolo USB 2.0 están disponibles en [7].
- No existen etapas de acondicionamiento de señal con base en el objetivo del proyecto.

- La velocidad de muestreo debe ser la más rápida posible, todas las rutinas del MCU o cambios aplicados deben ser programados con el objetivo de hacer al hardware más eficiente y mantener o de ser posible incrementar la velocidad de muestreo.

2.4.1. Desarrollo del hardware de medidas

Una vez establecidas las directrices de diseño, se presenta en la figura 2.2 un diagrama de flujo con una rutina que muestra de manera general el funcionamiento del MCU, sin puntualizar en las tareas específicas.

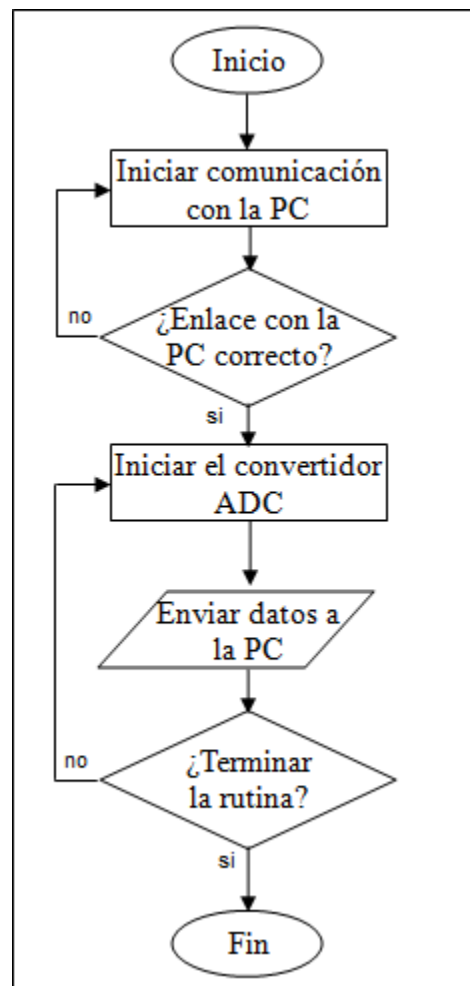


Figura 2.2. Rutina del microcontrolador.

Con en el objetivo de desarrollar un instrumento que pueda ser utilizado en cualquier computadora, se configura el dispositivo en clase HID, por sus siglas en ingles “Human Interface Device”, dicha clase posee la característica de ser identificada por cualquier computadora sin la necesidad de suministrar descriptores y driver específicos para el dispositivo, esto representa una conveniente ventaja en el desarrollo del dispositivo al no tener que ser dotado de descriptores específicos. El dispositivo configurado en dicha clase es identificado por LabVIEW como un bus de instrumentación con el uso de NI-VISA (complemento de LabVIEW). La resolución del convertidor ADC se mantendrá en 8 bits ya que de esa manera solo es necesaria una instrucción para enviar los datos esto en consistencia con las directrices de diseño que pretenden mantener el tiempo de ejecución en el mínimo posible para conseguir la velocidad de muestreo más rápida posible.

Se desarrollan dos enfoques para la rutina del MCU el primero una rutina secuencial y el segundo una rutina que hace uso del servicio de interrupciones. Se plantean inicialmente estos dos enfoques con el fin de encontrar el método más eficiente para realizar las tareas de conversión y envío de datos ya que las rutinas de interrupción son una herramienta para realizar la ejecución de instrucciones en paralelo, el objetivo es encontrar la forma de administrar las tareas para reducir el tiempo de ejecución de la rutina y obtener velocidades de muestreo altas. A continuación se muestran los diagramas de flujo correspondientes a los códigos 1 y 2(disponibles en el anexo A) en las figuras 2.3 y 2.4 respectivamente.

Todos los códigos del MCU que utilizan el tipo de transferencia “Interrupt” (documentados detalladamente en el anexo A) son probados con el VI 1, el cual es tratado más adelante en este capítulo.

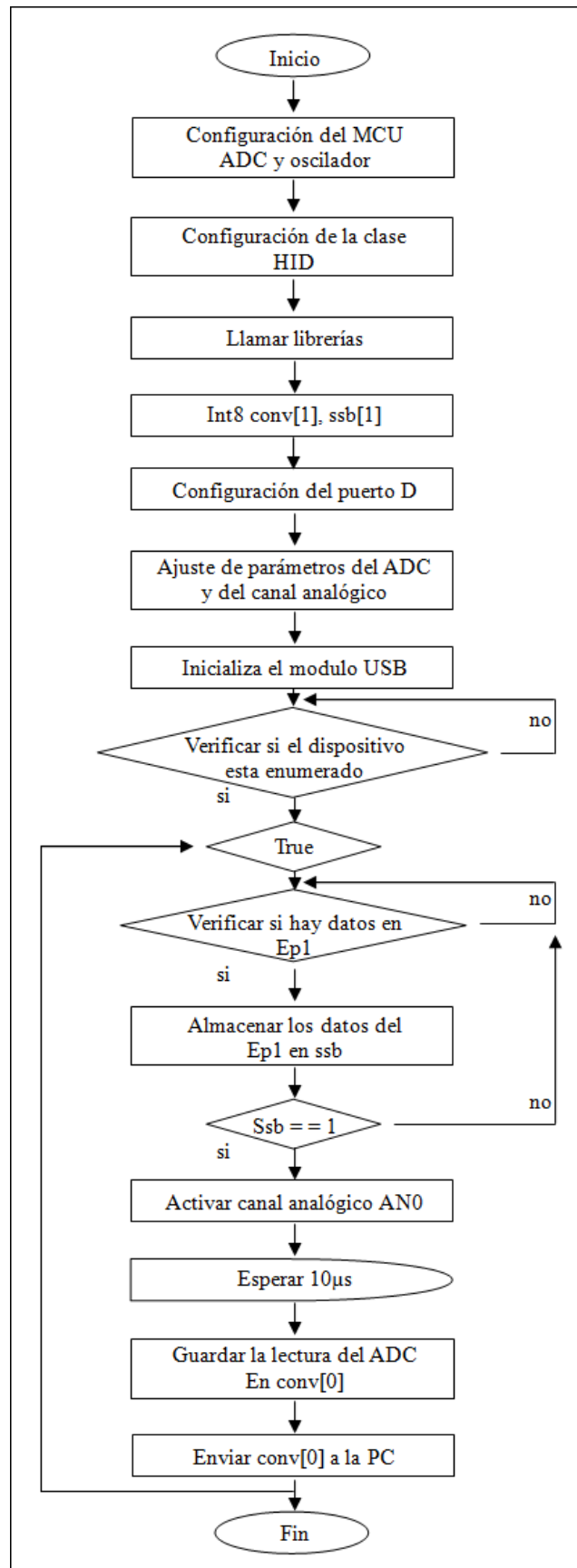


Figura 2.3. Diagrama de flujo del código 1.

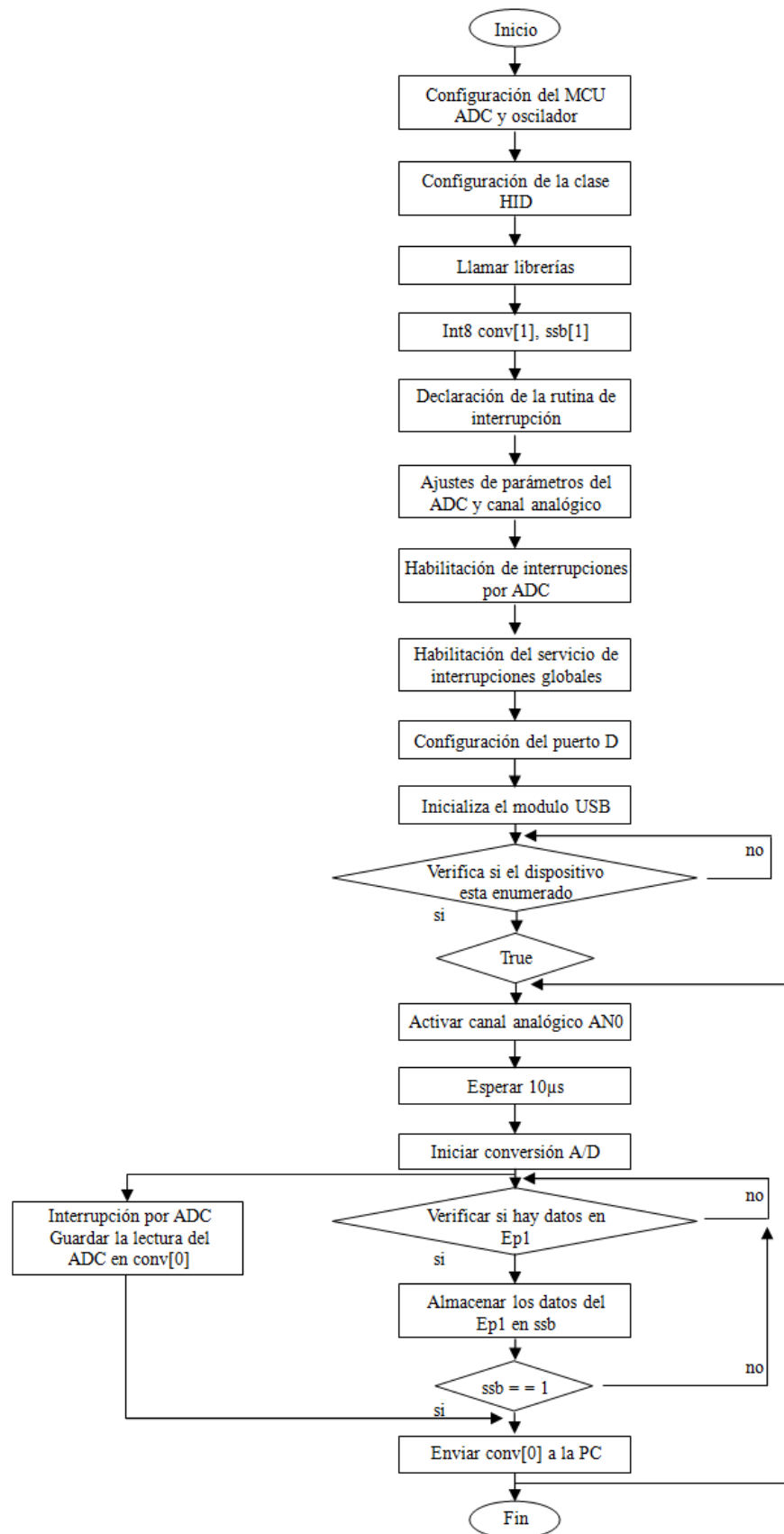


Figura 2.4. Diagrama de flujo del código 2.

El circuito eléctrico (circuito 1) que se ha desarrollado para el sistema de adquisición de datos es un circuito relativamente sencillo esto se debe a que el MCU contiene embebidos todos los periféricos necesarios, por lo tanto los elementos externos que se añaden al circuito son pocos, esto minimiza el espacio y las conexiones eléctricas que requiere, cabe destacar que este circuito es el mínimo necesario para que el sistema funcione, es decir no tiene ningún tipo de acondicionamiento de señal, aislamiento, ó acoplamiento por lo cual solo puede ser usado bajo condiciones en las que se garantice que la señal de entrada no dañara el MCU, el circuito 1 se muestra en la figura 2.5.

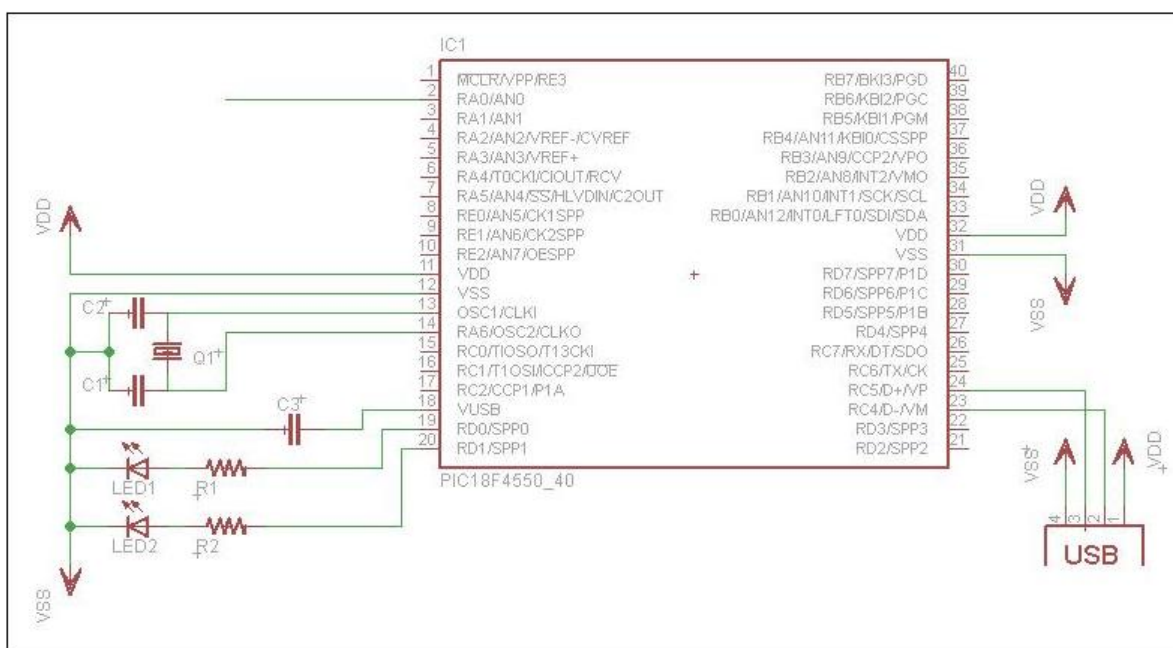


Figura 2.5. Circuito 1, esquemático del circuito eléctrico mínimo del sistema.

A partir de los resultados obtenidos en las pruebas (los resultados están disponibles en el capítulo III, y la metodología de pruebas se encuentra en la sección 2.6 de este capítulo) se decide realizar cambios en la estructura de los programas con el fin de buscar una disminución en el tiempo de ejecución, lo cual dará como resultado una velocidad de muestreo más alta, se mantienen ambos enfoques (con y sin el uso de interrupciones) con el objetivo de encontrar el más eficiente por lo tanto se generan los códigos 3 y 4 (disponibles en el anexo A), a continuación se muestran los diagramas de flujo en la figuras 2.6 y 2.7.

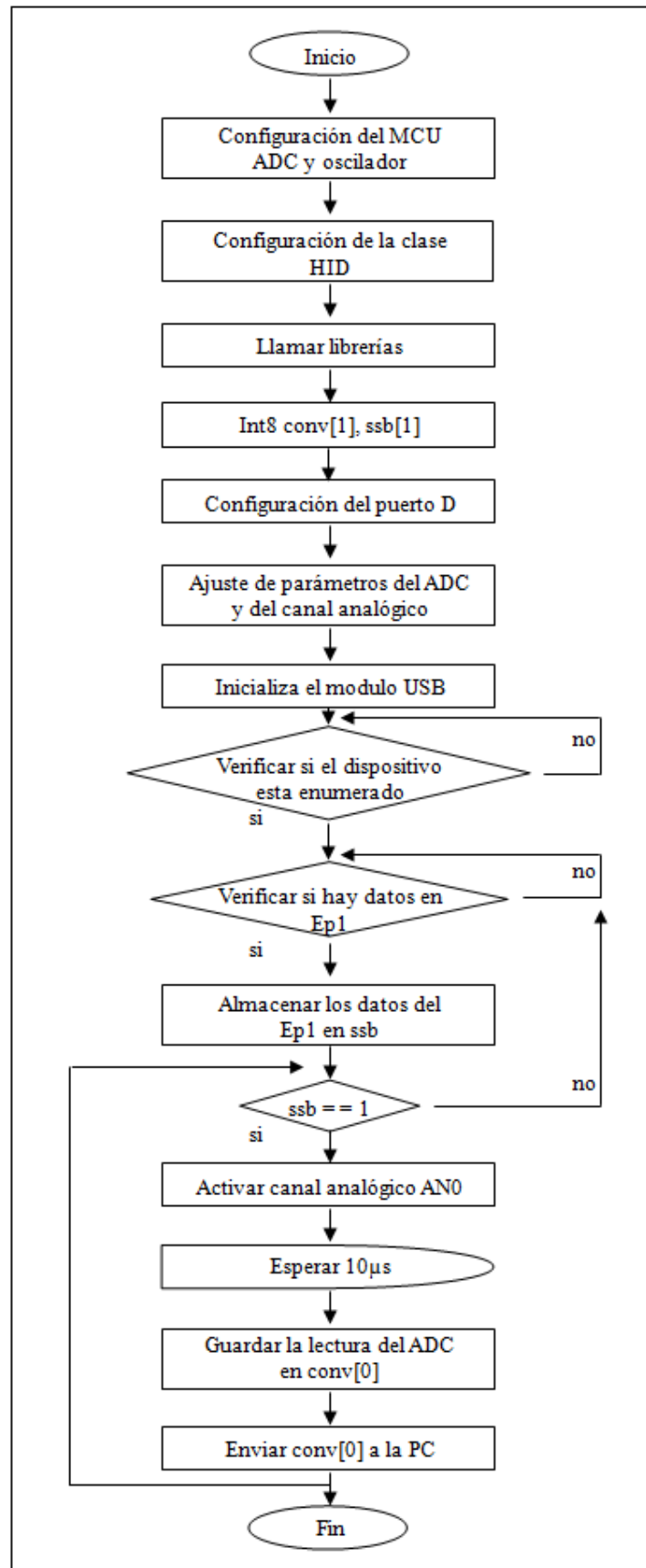


Figura 2.6. Diagrama de flujo del código 3

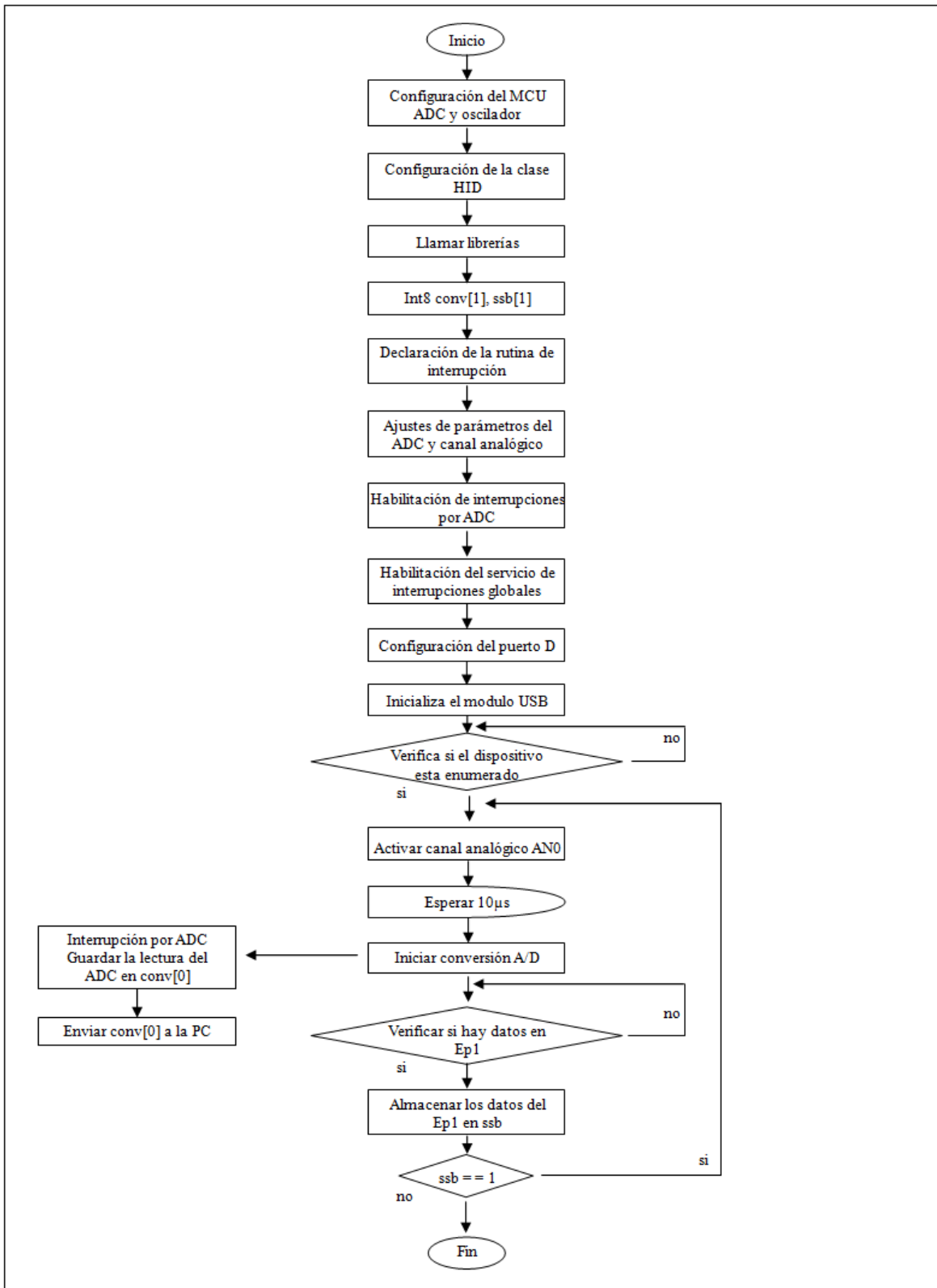


Figura 2.7. Diagrama de flujo del código 4.

De las pruebas realizadas a la rutina del código 4 se concluye que es más eficiente mantener la rutina sin el uso del servicio de interrupciones, por lo cual a partir del código 3 se genera un código mejorado (código 5) que no recurre al uso de interrupciones, su diagrama de flujo se muestra en la figura 2.8.

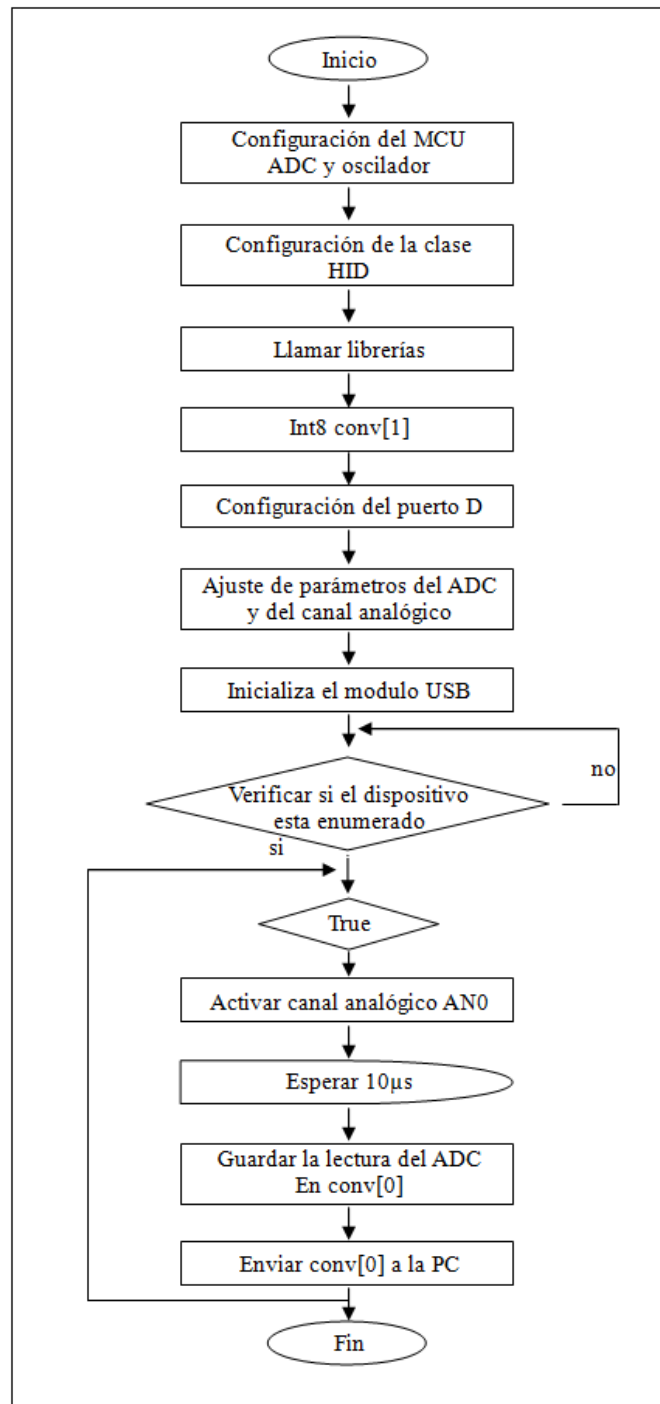


Figura 2.8. Diagrama de flujo del código 5.

Se realiza un análisis minucioso de los tiempos de conversión para el ADC y se toma la decisión de cambiar el tiempo de espera para la conversión de datos de un tiempo de 10µs a 7µs, que implica un incremento en la velocidad de muestreo. El tiempo recomendado por el fabricante es de 10µs para los dispositivos de la familia PIC18, sin embargo gracias al análisis realizado se sabe que el dispositivo que se usa requiere como mínimo un tiempo de 6.35µs, dada la resolución de la instrucción “delay_us()” el tiempo debe ser un número entero por esta razón el tiempo de espera es de 7µs. Este cambio se aprecia en el código 5.1, cuya diferencia con el código 5 es únicamente la disminución del tiempo de espera.

El tiempo de espera es calculado con base en la siguiente ecuación:

$$t_d = t_{acq} + 3t_{ad}^1 \text{ --- (4)}$$

Donde:

$$t_d \equiv \text{tiempo de espera mínimo}$$

$$t_{acq} \equiv \text{tiempo de adquisición mínimo} = 2.45\mu s$$

$$t_{ad} \equiv \text{tiempo de conversión por bit} = 1.3\mu s$$

Cabe destacar que esta ecuación solo refleja el tiempo de espera mínimo requerido por el modulo ADC del MCU; se sabe que la máxima frecuencia de muestreo del ADC es de 48KHz, y adicionalmente a estos tiempos debe sumarse el tiempo de ejecución de del de las instrucciones de la rutina, mas el tiempo que tardan en transferirse los datos desde el MCU hacia la PC y el tiempo que tarda el VI en desplegar y actualizar los datos, la suma de estos tiempos será equivalente al periodo de muestreo del sistema de adquisición de datos y se desataca que se considera el tiempo que tarda el VI en actualizar los datos ya que dicho tiempo influye directamente en el análisis de datos o el proceso de control que se pueda hacer a través del VI, por esta razón se busca optimizar las operaciones de MCU para alcanzar la máxima velocidad de muestreo posible.

¹Se deduce de la ecuación para calcular el tiempo de conversión del ADC que brinda el Fabricante [8].

Una de las razones por las que se ha tomado la decisión de utilizar la clase HID es por los beneficios que ofrece el tipo de transferencia “Interrupt”, ya que el tipo de datos que se obtiene de la conversión del ADC son enviados a la computadora con una sola instrucción.

Cuando el código 5.1 es puesto a prueba con el VI 3, para determinar la máxima frecuencia de muestreo, se aprecian limitaciones en el muestreo de señales. El rendimiento del sistema es mucho menor al esperado (los detalles de dichas limitaciones están disponibles en el capítulo III), y se identifica la solución, que consiste en cambiar el tipo de transferencia al tipo “Bulk”, esto a pesar de los beneficios que ofrece la transferencia “Interrupt”.

Al cambiar a tipo de transferencia “Bulk” serán necesarias dos instrucciones para enviar los datos, a diferencia del tipo de transferencia “Interrupt”.

Por dichas razones se genera el código 6 adecuado para la transferencia tipo “Bulk”, la librería de descriptores que se debe suministrar en la nueva configuración del dispositivo es un librería genérica suministrada con el compilador PIC C, dicha librería está diseñada para otra aplicación, por lo que debe ser sujeta a modificaciones para cumplir con las características del dispositivo (en [7] se detallan las características de los descriptores), el driver requerido para el dispositivo es creado con la herramienta NI-VISA que ofrece National Instruments para dispositivos genéricos, cuya función es crear un driver único para que LabVIEW identifique al dispositivo como un bus de instrumentación, ahora el dispositivo no será clase HID será un bus de instrumentación, adicionalmente a estos cambios dentro de la rutina se deben añadir instrucciones adicionales para convertir el tipo de datos de entero a cadena, la lectura del ADC ahora tendrá resolución de 10 bits ya que el paquete con la información solo requiere una línea de programación para ser enviado, esto cumple con el objetivo de mantener la velocidad de muestreo lo más rápido posible, no se verá afectado el tiempo de conversión ya que, es el mismo tiempo que se requiere para convertir 8 bits. Ya se ha determinado que el uso de interrupciones no es viable en esta aplicación, por lo que este enfoque es descartado. El diagrama de flujo del código 6 se muestra en la figura 2.9.

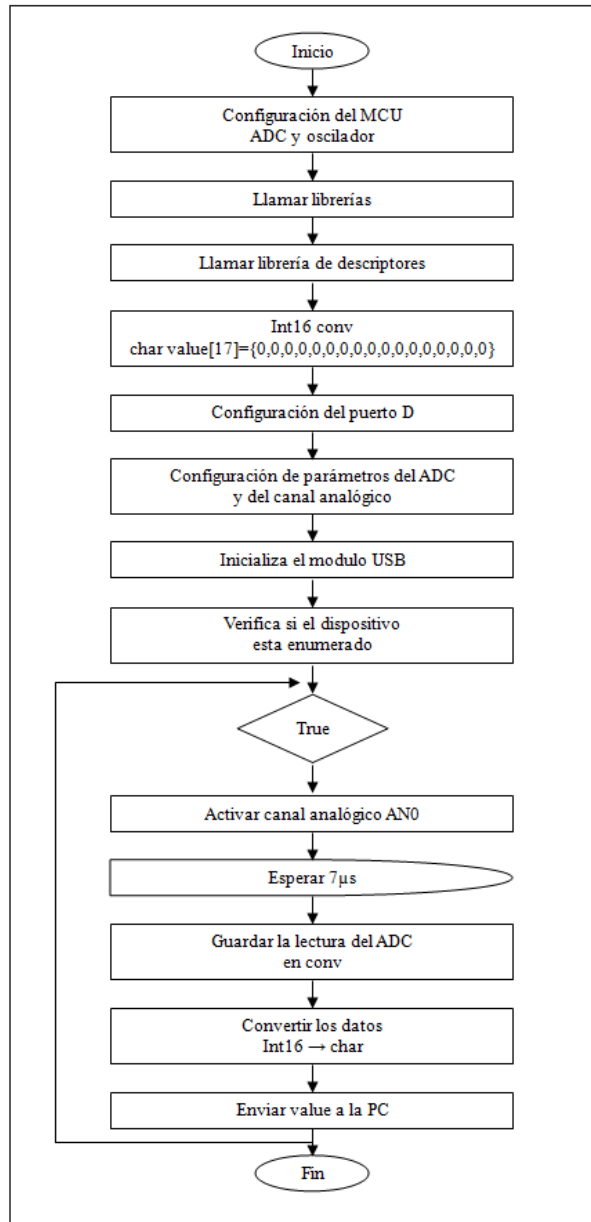


Figura 2.9. Diagrama de flujo del código 6.

Al realizarse una prueba para la estimación de la velocidad de muestreo del sistema, se logra identificar un problema, al iniciarse el análisis para encontrar la causa del problema se plantea la posibilidad de que esté relacionado con un lazo de tierra, se realizan mediciones y se observa que la diferencia de potencial entre la referencia de la fuente de alimentación y la referencia de la señal analógica es de 800 mV, adicionalmente se observan otros efectos que consisten en fallas en la transmisión de datos (interrupciones súbitas) así como reinicios

súbitos del microcontrolador, pero ya que el sistema está siendo probado con instrumentos de laboratorio se descarta la posibilidad de un lazo de tierra, ya que los equipos de laboratorio no pueden generar corrientes de la magnitud necesaria para que los lazos de tierra generen problemas de esta magnitud, por lo cual se analiza la posibilidad de que la puesta a tierra sea inadecuada, esto induciría ruido analógico en los circuitos digitales, y provocaría las fallas que observamos. Si las fallas están relacionadas con el aterrizamiento del ADC, pueden solucionarse implementando las siguientes recomendaciones:

- Separar los planos de tierra.
- Separar las fuentes de alimentación de circuitos analógicos y circuitos digitales.
- La diferencia de potencial entre los pines de alimentación y tierra no debe exceder la especificada por el fabricante.
- Los cristales de cuarzo deben estar rodeados por pistas de tierra.
- El cuerpo metálico del cristal debe ser aterrizado.
- El cristal debe ser colocado cerca del microcontrolador.
- No alternar con salidas digitales el mismo puerto donde se está realizando la conversión ya que esto introducirá ruido en las entradas digitales.
- Evitar altas corrientes en entradas o salidas ya que inducen ruido de alta frecuencia a la fuente de alimentación y puede afectar el resultado de la conversión.
- Conectar los planos de tierra por medio de diodos Schottky contrapuestos o con una pequeña esfera de ferrita.
- El sistema solo debe tener un plano de referencia, dos referencias crearan una antena de dipolo.
- Los planos de tierra deben estar formados por grandes cantidades de cobre.
- Utilizar capacitores de filtrado para la alimentación.
- Utilizar capacitores de filtrado en las entradas analógicas.
- Mantener la resistencia de entrada en los canales analógicos dentro de la magnitud indicada por el fabricante.
- Blindar las entradas analógicas colocando pistas de tierra junto a las de señal analógica, esto reducirá el ruido de “cross talk”.

La solución potencial al problema que se ha detectado implica realizar cambios en el circuito eléctrico con el que desarrolla el proyecto, por lo que se crea el circuito 2, igual que el anterior es implementado en una tablilla de pruebas (protoboard) para verificar su funcionamiento, para lograr la implementación de las recomendaciones se aplican pequeñas adecuaciones, cabe destacar que las recomendaciones anteriores están proyectadas para implementarse en un PCB (circuito impreso por sus siglas en inglés Printed Board Circuit) por esto es que se deben hacer adecuaciones, que consisten en sustituir los planos de tierra por las porciones conductoras de la tablilla, en cuanto a el blindaje no son aplicadas ninguna de la recomendaciones, ya que no es posible conseguirlo en la tablilla de pruebas.

Los cambios que se realizan al circuito 1 son los siguientes:

1. Habilitación del pin Vref- (pin 4), el pin Vref+ (pin 5) no se habilita puesto que se mantiene cortocircuitado por software con los pines VDD (pines 11 y 32).
2. El pin Vref- (pin 4) será ahora la tierra analógica, será conectado eléctricamente con una porción conductora de la tablilla de pruebas, para asemejar las porciones de cobre que actúan como planos de tierra.
3. La señal analógica aplicada a la entrada analógica AN0 (pin 2) es filtrada con un capacitor de 10 nF, dispuesto en paralelo a la fuente de la señal.
4. Se añaden capacitores de filtrado a la fuente de alimentación, un capacitor de 1 μ F que debe ubicarse lo más cerca posible y en paralelo con la fuente de alimentación, dos más de 0.1 μ F y 10 pF en paralelo con la fuente y lo más cerca posible de los pines VDD (pines 11 y 32) y VSS (pines 12 y 31).
5. Los planos de tierra deben conectarse eléctricamente por medio de dos diodos Schottky (matricula BAT42) contrapuestos, para mantener el potencial entre los dos planos de tierra debajo de 0.3 V.

Estas medidas aseguran el correcto aterrizamiento del ADC, esto prevendrá y evitara los errores de conversión ocasionados por ruido analógico en la circuitería digital ó ruido digital en los circuitos analógicos, estos cambios requieren configurar los pines por software lo que da lugar a un nuevo código del MCU (código 6.1) cuya diferencia con el código 6 será la

configuración del pin Vref-. Cabe destacar que tanto el circuito 2 como el código 6.1 pueden implementarse en PCB sin ningún cambio adicional, realizando el diseño del layout apropiado.

El circuito eléctrico (circuito 2) se muestra en la figura 2.10.

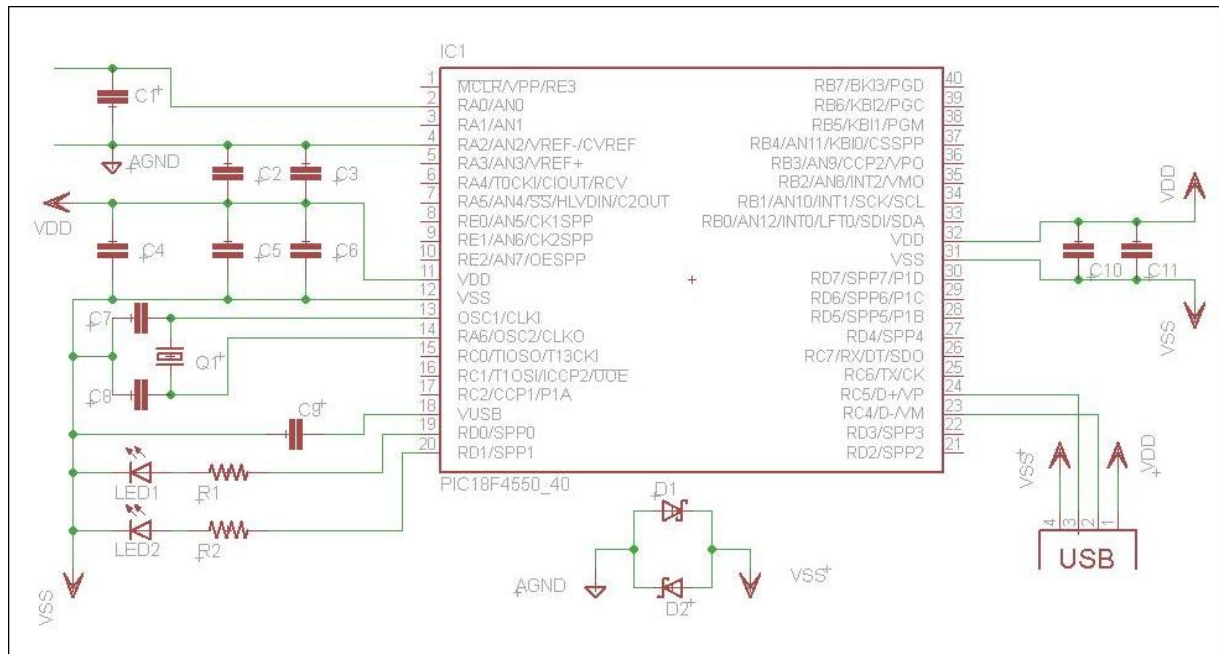


Figura 2.10. Circuito 2, esquemático del circuito eléctrico con el aterrizamiento adecuado.

Al implementarse la separación de planos de tierra se observa que las fallas desaparecen, por lo cual se da por solucionado el problema y se concluye que la puesta a tierra del circuito 1 no era la adecuada, y como se suponía el problema fue causado por ruido analógico en los circuitos digitales.

2.5. Diseño del VI

El modulo de software que se desarrolla en la plataforma LabVIEW es el correspondiente al VI, cabe destacar que los códigos son desarrollados en lenguaje grafico.

El diseño se realiza con base en los objetivos del proyecto, la elección del Bus de datos y la elección de la plataforma para diseño y control de instrumentos virtuales que se han hecho previamente. Y las directrices de diseño serán las siguientes:

- El VI deberá cumplir las características de una arquitectura abierta y escalable tal que pueda ser utilizada en diversas aplicaciones, en otras palabras el VI debe estar abierto a la adición de código generado por el usuario.
- Todos los códigos o cambios aplicados deben ser programados con el objetivo de hacer al VI más eficiente y mantener o de ser posible incrementar la velocidad de muestreo.

2.5.1. Desarrollo del VI

Dado que el VI y el hardware se han desarrollado de manera simultánea cada código del VI que se mencione en esta sección estará ligado a uno o más códigos de las rutinas del MCU, en contraste con la sección anterior en la cual solo se detalla el desarrollo del hardware.

Con el código 1 para el MCU se desarrolla un VI (VI 1) que sirve como estructura básica de los códigos utilizados para dispositivos con transferencias de tipo “Interrup”, este código incluye funciones de entrada y salida de datos, es usado para poner a prueba el funcionamiento de los códigos del MCU 1, 2, 3, y 4. Cabe destacar que el protocolo USB tiene como referencia el Hub para hacer la distinción entre entrada y salida con relación a la dirección de transmisión de los datos, es decir los datos salen del Hub o entran al Hub, hacia o desde el o los dispositivos.

Cuando se pone a prueba el código 5 el VI 1 se modifica, ya que se descarta el uso de un bit de inicio y paro, por lo que las funciones de salida ya no son necesarias en el VI, eliminar esta función también servirá para disminuir el tiempo de ejecución del VI, estos cambios dan lugar al VI 2, con este VI es posible adquirir datos y graficar señales.

En LabVIEW existen dos formas de graficar señales, a través de las funciones “Waveform Graph” y “Waveform Chart”, la diferencia entre ambas es el tipo de datos que se grafican, mientras que “Waveform Chart” grafica punto a punto a medida que la función recibe los datos, “Waveform Graph” grafica un conjunto de datos obtenidos en un intervalo de tiempo, con el VI 2 se utiliza la función “Waveform Chart” ya que los datos recibidos por el VI llegan de a uno, conforme la muestra es tomada, esta función es adecuada para graficar datos que se monitorean en tiempo real; los datos son introducidos de esta manera ya que se ha demostrado que la rutina de muestreo del MCU es más eficiente de esta manera, el VI adquiere y grafica los datos obtenidos pero no los almacena, es decir la variable donde es guardado el dato proveniente del hardware de medidas se sobrescribe a medida que se toma otra muestra. Para utilizar la función “Waveform Graph” es necesario almacenar cada muestra, y agruparlas en un vector, se recurre a esta función ya que teniendo control sobre el eje “x” de la grafica puede verificarse la frecuencia de la señal y la aparición de señales alias siendo esta una herramienta más precisa para determinar la frecuencia de muestreo, a diferencia de la función “Waveform Chart” donde el eje “x” se actualiza continuamente perdiendo la información anterior, en ambas funciones el eje “x” es asignado al tiempo y el eje “y” a la amplitud, por lo tanto se decide someter a modificaciones el VI 2, para poder hacer uso de la función “Waveform Graph”, las modificaciones consisten en la adición de código para generar el vector de datos, estos se logra mediante la función “Shift register” de la cual se puede hacer uso ya que se tiene un ciclo while para ejecutar el VI de manera continua estos cambios generan el VI 3.

Adicionalmente a la generación del vector, se busca la optimizar del código, dado que este se encuentra en su totalidad dentro del ciclo while se ejecutan todas las operaciones en cada iteración del ciclo, con el uso de la estructura “Flat sequence” se pueden separar la operaciones en tres partes como se enlistan a continuación:

1. Apertura de la comunicación con el puerto USB
2. Recepción de datos
3. Clausura de la comunicación con el puerto USB

Con la finalidad de que la porción del código que está contenida dentro del ciclo while sean solo las funciones que reciben los datos, es decir las únicas funciones que se ejecutaran continuamente serán las que reciben los datos, se genera el VI 4.

Al ser ejecutado el VI 4 en sustitución del VI 3 se observa que, la porción del código que recibe los datos solo recupera una muestra, a pesar de que el hardware de medidas continua enviando las muestras de manera continua, esto se debe a que el VI solo obtiene los datos cada que se genera una interrupción, como es especificado en el protocolo USB, en términos del VI debe abrirse la comunicación con el puerto USB cada que se desea recuperar un dato por lo que se genera el VI 5 tratando de mantener el uso de la estructura “Flat sequence” para solucionar este problema, sin embargo al ponerse a prueba este VI se presenta un desbordamiento en la función “VISA Wait on Event” ocasionado por que el evento anterior no fue cerrado, esto quiere decir que la comunicación con el puerto USB debe ser abierta y cerrada para cada interrupción o cada evento, esto significa que no es viable el uso de la estructura “Flat sequence” con este tipo de transferencia, por lo tanto se descarta el uso del VI 4 y 5, y es necesario volver al VI 3.

Al realizarse las pruebas para determinar la frecuencia de muestreo usando el VI 3, se observa que la frecuencia de muestreo es muy baja, esto es ocasionado por el tiempo de ejecución del VI que aumenta con el uso de la función “Shift register”, ya que deben añadirse más registros para almacenar la cantidad de datos necesarios para reconstruir la señal y cada iteración del ciclo while llenara uno de los registros, al aumentar la cantidad de registros el VI requerirá mas iteraciones para reconstruir una señal y por lo tanto más tiempo, esto afectara la capacidad de observar cambios en la variable; por lo que se toma la decisión de cambiar el tipo de transferencia.

Para el tipo de transferencia “Bulk” se desarrolla el VI 6, para este tipo de transferencia es posible usar la estructura “Flat sequence” ya que los datos no se generan por interrupciones, con la experiencia previa del VI 3 se descarta por completo el uso de la función “Shift register” ya que limita la cantidad de muestras y esto afecta la reconstrucción de la señal.

El VI 6 se pone a prueba junto con el código 6 del MCU, en el VI se aprecian las fallas ocasionadas por el ruido analógico, sin embargo el VI es adecuado, y el hardware es el que se somete a un proceso de mejora. De este proceso de mejora se obtiene el código 6.1 con un circuito diferente, el VI 6 sigue siendo adecuado para el sistema.

2.6. Metodología de pruebas

En esta sección se describe el método y los objetivos de aplicar pruebas al sistema. El objetivo principal es conocer el comportamiento del sistema con el fin de evaluar si es necesario realizar cambios que mejoren su funcionamiento y si dichos cambios han tenido el efecto deseado. Las pruebas son realizadas al sistema de manera integral, pero también son probados los subsistemas de manera independiente, de esta manera se limitan las etapas en donde se pueda producir un error. Dichas pruebas se clasifican en tres clases:

- Pruebas del hardware. Las pruebas de hardware tienen como objetivo particular identificar las áreas de oportunidad en el código de la rutina del MCU, y verificar que el dispositivo este correctamente configurado como dispositivo USB.
- Pruebas del software. Estas pruebas tienen como objetivo identificar que la transferencia y conversión de datos sean correctas, e identificar áreas de oportunidad en la arquitectura del VI que se implementa.
- Pruebas del sistema. Estas tienen un objetivo ambicioso, verificar la máxima frecuencia de muestreo, verificar que la arquitectura del VI sea adecuada para la rutina del MCU y comprobar que la arquitectura es la adecuada para procesar los datos provenientes del hardware de medidas.

El método general para realizar pruebas consiste en un ambiente controlado en el que las condiciones en que se prueba el sistema puedan ser replicadas cada vez que se realiza una prueba, un ambiente controlado garantiza que no habrá variables diferentes a la vez anterior que se realizó la prueba y será posible observar el mismo comportamiento, cada prueba deberá revelar de manera clara la fuente o la causa que origina el comportamiento que se

analiza, esto dará certeza de que se ha identificado una o más áreas de oportunidad para mejorar el código.

Las pruebas como el desarrollo se ejecutan simultáneamente, es decir, se debe poner a prueba el binomio de hardware y software. Las pruebas se dividen en dos etapas:

- I. Pruebas primarias, en la prueba primaria se verifica el binomio de hardware y VI, se buscan errores de conversión y errores en la transmisión de datos, es en esta etapa se realizan las pruebas de hardware y de software.
- II. Pruebas finales, las pruebas finales se ejecutan cuando se decide con base en la prueba primaria que el binomio funciona correctamente, y en función de los resultados obtenidos de las pruebas finales se determinara si el objetivo del proyecto ya se ha cumplido, es en esta etapa se realizan las pruebas del sistema.

En las pruebas primarias se verifica que la corriente se encuentre dentro del límite que establece el protocolo USB que se limita a 500 mA como corriente máxima y se mide el intervalo de tiempo que tardan en tomarse las muestras, ya que el muestreo es continuo y las instrucciones de convertir y enviar los datos se encuentran dentro de un ciclo infinito, la mejor manera de hacerlo es medir el tiempo que tarda en volver a ejecutarse la instrucción que envía los datos por el puerto USB, dicho tiempo puede ser estimado ya que se sabe la duración de cada una de las instrucciones, la frecuencia del oscilador y el tiempo de conversión del ADC, pero como se recurre a una programación modular no es posible determinar el tiempo de ejecución de cada modulo de manera exacta, por lo cual estimar el tiempo en función del tiempo de ejecución de cada instrucción no aporta información útil.

Las estructuras de las rutinas del MCU poseen ciclos dentro de los cuales se toma una muestra y la siguiente muestra es tomada cuando el ciclo se vuelve a ejecutar, para conocer en la practica la duración de este tiempo, se añaden instrucciones para poner en estado alto una salida digital cuando inicie el ciclo y devolverla al estado bajo al final del ciclo lo que tendrá como resultado una señal periódica alternando entre estado alto y bajo con flancos

ascendentes que marcan el inicio del ciclo y flancos descendentes que marcan el fin del ciclo y la duración del ciclo será igual a la duración en estado alto entre flanco y flanco, para medir esta señal se emplea un osciloscopio. Por definición el tiempo que dure el ciclo será el equivalente al tiempo de muestreo, ya que el tiempo de muestreo es el tiempo que tarda en tomarse la siguiente muestra. Cabe destacar que este método de medición afecta al sistema ya que se incrementa el tiempo de ejecución del ciclo a medida que se añaden instrucciones dentro de este, por lo tanto el resultado de esta medición no refleja con exactitud el tiempo de ejecución del ciclo while. En la figura 2.11 se muestran los flancos y la duración en estado alto de la salida digital usada para medir el tiempo de ejecución del ciclo while.

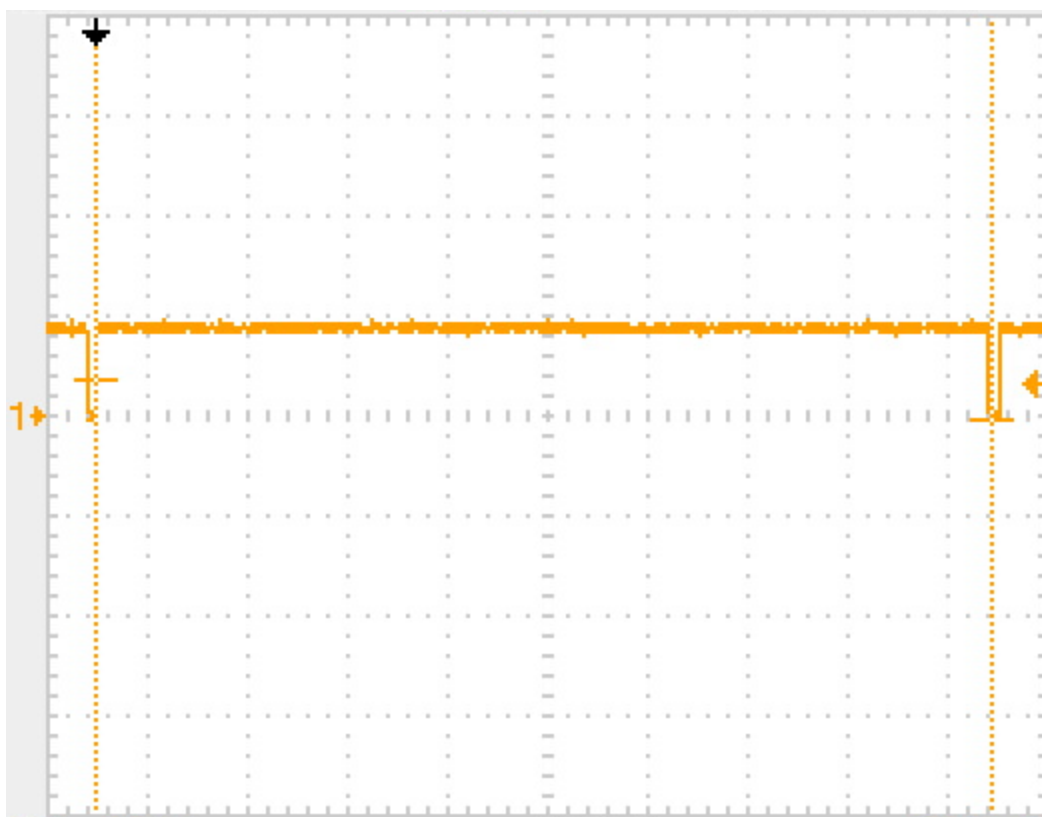


Figura 2.11. Señal generada por una salida digital.

La forma de realizar pruebas del VI es más sencilla ya que LabVIEW ofrece herramientas para la depuración del código, particularmente se busca verificar que los datos que son recibidos son correctos, es decir que no existe error de conversión ni error en la transición de datos, para lo cual se debe controlar los factores que pueden afectar esta prueba, tales como ruido en las señales, la resistencia asociada a la fuente de la señal, la polaridad de la señal,

la amplitud de la señal y la radiación electromagnética, ya que existen factores como la radiación electromagnética que no pueden ser eliminados en su totalidad, las únicas medidas para evitarlo serán las buenas prácticas en el cableado y en la realización de las mediciones, cabe destacar que todas las pruebas son realizadas en una tablilla de prototipos. Para minimizar el efecto indeseado que puedan provocar las señales con ruido se emplea equipo de laboratorio como fuentes de alimentación y generadores de señal los cuales también garantizan una impedancia de salida tal que no supera la requerida en las especificaciones del ADC ya que de lo contrario la conversión se puede ver afectada. La manera de comprobar que la conversión y el envío de datos es correcto se realiza por un proceso de comparación, el dato desplegado en el VI equivale al número binario resultante de la conversión dado que se conoce la función de transferencia del ADC y la magnitud del voltaje que se aplica se podrá realizar la comparación, adicionalmente deberá realizarse la medición a lo largo de todo el rango del ADC es decir desde 0 VDC hasta 5VDC, y verificar que el incremento mínimo sea de 4.8mV correspondiente a la resolución del ADC esto hará evidente si existe algún error de conversión tales errores se mencionan en el capítulo I.

En el siguiente capítulo se presentan en detalle los resultados obtenidos en cada prueba.

Capítulo III. Resultados de las pruebas.

3.1. Introducción

Este capítulo presenta los resultados obtenidos de cada prueba realizada como se describe en el capítulo II.

3.2. Resultados obtenidos de las pruebas

Como se describe en la sección 2.6, para estimar la duración del ciclo while se aplica un método que afecta al sistema, este consiste en la adición de dos instrucciones para generar una salida digital periódica la cual se mide con un osciloscopio, a continuación se presentan los resultados de las mediciones que se realizaron en cada prueba, la figura 3.1 ilustrar como afecta esta medición el tiempo de ejecución del ciclo while.

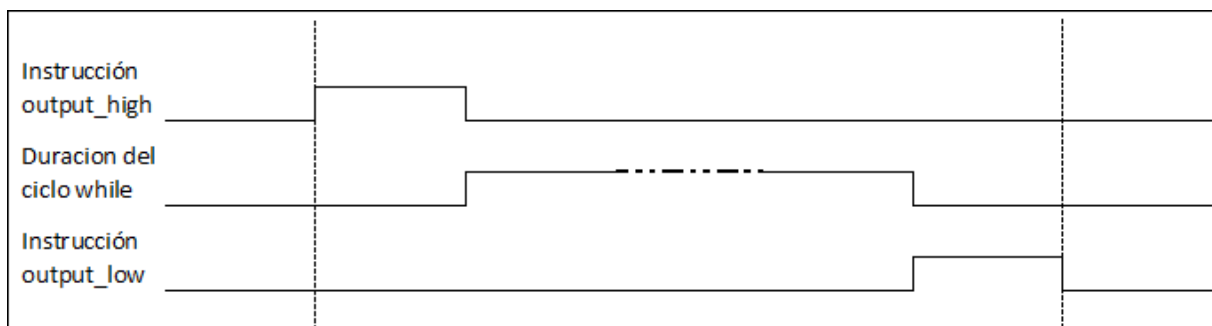


Figura 3.1. Diagrama de tiempos del muestreo realizado por el MCU.

En la tabla 3.1 se muestran los resultados de las pruebas primarias realizadas al código 1 del MCU y al VI 1 con el circuito 1.

Tabla 3.1. Resultados de las pruebas del código 1 y VI 1 con el circuito 1.

Prueba.	Resultado.	Unidades.
Duración del ciclo while	7.8	ms
Corriente del circuito	52	mA
Errores de conversión en la escala completa (0-1024)	No se detecto ninguno	-
Errores de transmisión	No se detecto ninguno	-

Observaciones:

La ejecución del ciclo while toma mucho tiempo, la conversión es correcta en todo el rango. El VI funciona correctamente, no se detectan errores de envío o recepción de datos. Se toman medidas para disminuir el tiempo de ejecución del ciclo while las cuales son puestas a prueba en el código 3, consisten en un cambio en la estructura del programa remplazando las dos estructuras if anidadas por la combinación de una estructura if con un ciclo while. Se decide no ejecutar las pruebas finales ya que se detectan grandes aéreas de oportunidad.

En la tabla 3.2 se muestran los resultados de las pruebas primarias realizadas al código 2 del MCU y al VI 1 con el circuito 1.

Tabla 3.2. Resultados de las pruebas del código 2 y VI 1 con el circuito 1.

Prueba.	Resultado.	Unidades.
Duración del ciclo while	8	ms
Corriente del circuito	52	mA
Errores de conversión en la escala completa (0-1024)	No se detecto ninguno	-
Errores de transmisión	No se detecto ninguno	-

Observaciones:

La ejecución del ciclo while toma mucho tiempo, la conversión es correcta en todo el rango. El VI funciona correctamente no se detectan errores de envío o recepción de datos. Se realizan cambios a la estructura del código para disminuir la duración del ciclo while que consisten en la sustitución del ciclo while por un ciclo do-while y se descartan las estructuras if anidadas, adicionalmente el envío de datos se llevara a cabo en la rutina de interrupción, estos cambios se aprecian en el código 4. Se decide no ejecutar las pruebas finales ya que se detectan grandes aéreas de oportunidad.

En la tabla 3.3 se muestran los resultados de las pruebas primarias realizadas al código 3 del MCU y al VI 1 con el circuito 1.

Tabla 3.3. Resultados de las pruebas del código 3 y VI 1 con el circuito 1.

Prueba.	Resultado.	Unidades.
Duración del ciclo while	100	µs
Corriente del circuito	52	mA
Errores de conversión en la escala completa (0-1024)	No se detecto ninguno	-
Errores de transmisión	No se detecto ninguno	-

Observaciones:

Se observa una reducción significativa del tiempo de ejecución del ciclo while, se aprecia que la reducción del tiempo es debida a los cambios en la estructura del programa. El VI funciona correctamente, no se detectan errores de envío o recepción de datos. Se decide reestructurar el programa para prescindir de una señal de inicio o paro y con esto se eliminan todas las estructuras anidadas, se espera reducir el tiempo de ejecución y hacer el muestreo más eficiente, estos cambios se observan en el código 5. Se decide no ejecutar las pruebas finales ya que aun existen aéreas de oportunidad.

En la tabla 3.4 se muestran los resultados de las pruebas primarias realizadas al código 4 del MCU y al VI 1 con el circuito 1.

Tabla 3.4. Resultados de las pruebas del código 4 y VI 1 con el circuito 1.

Prueba.	Resultado.	Unidades.
Duración del ciclo while	7	ms
Corriente del circuito	52	mA
Errores de conversión en la escala completa (0-1024)	No se detecto ninguno	-
Errores de transmisión	No se detecto ninguno	-

Observaciones:

No se observa que los cambios surtan el efecto deseado, se concluye de manera semejante al código 3 que prescindir de una señal de inicio y paro vuelve el código más eficiente, sin embargo para que el enfoque con el uso del servicio de interrupciones se factible el MCU

debería realizar tareas adicionales a la adquisición de señales, ó en su defecto adquirir más de una señal, por lo que se descarta el uso de este algoritmo y no se ejecutan las pruebas finales.

En la tabla 3.5 se muestran los resultados de las pruebas primarias realizadas al código 5 del MCU y al VI 2 con el circuito 1.

Tabla 3.5. Resultados de las pruebas del código 5 y VI 2 con el circuito 1.

Prueba.	Resultado.	Unidades.
Duración del ciclo while	44.6	μs
Corriente del circuito	52	mA
Errores de conversión en la escala completa (0-1024)	No se detecto ninguno	-
Errores de transmisión	No se detecto ninguno	-

Observaciones:

Como se menciona en el capítulo II se desarrolla el VI 2 de manera análoga al código 5 eliminando secciones de código que ahora son innecesarias, nuevamente se observa una disminución importante del tiempo de ejecución, sin errores de conversión ni errores de recepción de datos, y como se menciona en el capítulo II se aplican cambios que disminuye el tiempo de espera de 10 μs a 7 μs, esto se observa en el código 5.1. Ya que aun existen cambios por hacer se decide no ejecutar las pruebas finales.

En la tabla 3.6 se muestran los resultados de las pruebas primarias realizadas al código 5.1 del MCU y al VI 2 con el circuito 1.

Tabla 3.6. Resultados de las pruebas del código 5.1 y VI 2 con el circuito 1.

Prueba.	Resultado.	Unidades.
Duración del ciclo while	40.2	μs
Corriente del circuito	52	mA
Errores de conversión en la escala completa (0-1024)	No se detecto ninguno	-
Errores de transmisión	No se detecto ninguno	-

Observaciones:

Se aprecia nuevamente una disminución en el tiempo de ejecución, la conversión es correcta en todo el rango. El VI funciona correctamente y no se detectan errores en la recepción de datos. Se toma la decisión de que la rutina del MCU es eficiente como para ejecutar las pruebas finales, para esto se desarrolla un nuevo VI, el binomio de VI2 y el código 5.1 no son aptos para pruebas finales, deberán realizarse con un nuevo VI, para determinar con precisión la aparición de señales alias, como se explica en el capítulo II el VI 3 se desarrolla para realizar estas pruebas.

En la tabla 3.7 se muestran los resultados de las pruebas primarias realizadas al código 5.1 del MCU y al VI 3 con el circuito 1.

Tabla 3.7. Resultados de las pruebas del código 5.1 y VI 3 con el circuito 1.

Prueba.	Resultado.	Unidades.
Duración del ciclo while	40.2	μs
Corriente del circuito	52	mA
Errores de conversión en la escala completa (0-1024)	No se detecto ninguno	-
Errores de transmisión	No se detecto ninguno	-

Observaciones:

El VI funciona correctamente, los cambios aplicados son monitoreados con la herramienta “Probe” y “Highlight execution” de LabVIEW, del estudio de los patrones de diseño de LabVIEW se concluye que debe ser usada la estructura “Flat sequence” con el fin de ejecutar

de manera continua solo la porción de código que recupera los datos, estos cambios serán aplicados en el VI 4. Dadas la nuevas aéreas de oportunidad que se visualizan ahora, aun no es posible ejecutar las pruebas finales, ya que el VI continua en desarrollo.

En la tabla 3.8 se muestran los resultados de las pruebas primarias realizadas al código 5.1 del MCU y al VI 4 con el circuito 1.

Tabla 3.8. Resultados de las pruebas del código 5.1 y VI 4 con el circuito 1.

Prueba.	Resultado.	Unidades.
Duración del ciclo while	40.2	µs
Corriente del circuito	52	mA
Errores de conversión en la escala completa (0-1024)	No es posible determinar si los hay	-
Errores de transmisión	No es posible determinar si los hay	-

Observaciones:

Los cambios aplicados al VI no surten el efecto deseado, no es posible determinar si existen errores de conversión o en la transmisión de datos ya que no es posible monitorear estos datos con el VI, el comportamiento del VI es diferente al que se pronostica, esto debido a la omisión de los principios que rigen el funcionamiento del tipo de transferencia y la clase, con la ayuda de las herramientas de LabVIEW para la depuración del código se confirma que la causa es la omisión de los principios de funcionamiento, y ahora se sabe que el error radica en la estructura del código, se realizan cambios en el código del VI para solucionar el problema, dichos cambios se aplican en el código 5. Nuevamente no es posible ejecutar las pruebas finales ya que el VI no es apto para esto.

En la tabla 3.9 se muestran los resultados de las pruebas primarias realizadas al código 5.1 del MCU y al VI 5 con el circuito 1.

Tabla 3.9. Resultados de las pruebas del código 5.1 y VI 5 con el circuito 1.

Prueba.	Resultado.	Unidades.
Duración del ciclo while	40.2	μs
Corriente del circuito	52	mA
Errores de conversión en la escala completa (0-1024)	No es posible determinar si los hay	-
Errores de transmisión	No es posible determinar si los hay	-

Observaciones:

El VI no funciona correctamente se presenta un desbordamiento en la función “VISA Wait on Event” durante la ejecución y debe ser terminada debido a este error, nuevamente se hace uso de la herramientas para la depuración del código y se concluye que el enfoque que se tiene del patrón de diseño basado en la estructura “Flat sequence” no es viable para usarse con este tipo de transferencia y clase, se decide volver al código 3 para realizar las pruebas finales.

En la tabla 3.10 se muestran los resultados de las pruebas finales realizadas al código 5.1 del MCU y al VI 3 con el circuito 1.

Tabla 3.10. Resultados de las pruebas del código 5.1 y VI 3 con el circuito 1.

Prueba.	Resultado.	Unidades.
Duración del ciclo while	40.2	μs
Corriente del circuito	52	mA
Errores de conversión en la escala completa (0-1024)	No se detecto ninguno	-
Errores de transmisión	No se detecto ninguno	-

Observaciones:

Los cambios no surten el efecto deseado, el VI tiene un comportamiento errático y lento al desplegar las señales. Se estima la frecuencia de muestreo en 7 Hz, pero la prueba no es

concluyente dado que no se puede determinar con exactitud si hay o no presencia de señales alias al incrementar la frecuencia de la señal de entrada a mas de 7 Hz. Se concluye que el tipo de transferencia utilizada no es apta para esta aplicación. Se decide cambiar la transferencia al tipo “Bulk” este cambio se aprecia en el código 6.

En la tabla 3.11 se muestran los resultados de las pruebas primarias realizadas al código 6 del MCU y al VI 6 con el circuito 1.

Tabla 3.11. Resultados de las pruebas del código 6 y VI 6 con el circuito 1.

Prueba.	Resultado.	Unidades.
Duración del ciclo while	42	μs
Corriente del circuito	52	mA
Errores de conversión en la escala completa (0-1024)	No es posible determinar si los hay	-
Errores de transmisión	Se detectan errores en la recepción de datos	-

Observaciones:

Se detectan errores en la recepción de los datos, y comportamiento errático del MCU y del VI, se observa este comportamiento exclusivamente cuando son aplicadas a la entrada analógica señales con forma de onda cuadrada con frecuencias de más de 10 Hz, se analiza el problema y se determina que el comportamiento errático tanto en el MCU como en el VI es el efecto del ruido analógico en los circuitos digitales, en el capítulo II se detallan las medidas tomadas para solucionarlo, estas se aprecian en la implementación del circuito 2 y en el código 6.1 del MCU.

En la tabla 3.12 se muestran los resultados de las pruebas primarias realizadas al código 6.1 del MCU y al VI 6 con el circuito 2.

Tabla 3.12. Resultados de las pruebas del código 6.1 y VI 6 con el circuito 2.

Prueba.	Resultado.	Unidades.
Duración del ciclo while	42	μs
Corriente del circuito	52	mA
Errores de conversión en la escala completa (0-1024)	No se detecto ninguno	-
Errores de transmisión	No se detecto ninguno	-

Observaciones:

El problema ocasionado por el ruido analógico se ha solucionado, no se detectan errores de conversión ni de transmisión el comportamiento del MCU es estable con cualquier forma de onda aplicada a la entrada, el VI funciona correctamente como se esperaba. Se toma la decisión de pasar a la etapa de pruebas finales, se adecua el VI para graficar señales, se mantiene la misma estructura, el único cambio es la adición de funciones para graficar dentro del ciclo while del VI.

En la tabla 3.13 se muestran los resultados de las pruebas finales realizadas al código 6.1 del MCU y al VI 6 con el circuito 2.

Tabla 3.13. Resultados de las pruebas del código 6.1 y VI 6 con el circuito 2.

Prueba.	Resultado.	Unidades.
Duración del ciclo while	42	μs
Corriente del circuito	52	mA
Errores de conversión en la escala completa (0-1024)	No se detecto ninguno	-
Errores de transmisión	No se detecto ninguno	-

Observaciones:

El MCU y el VI funcionan correctamente, la frecuencia de muestreo se estima en 24KHz, se determina que el binomio VI y hardware funcionan correctamente, y que la frecuencia de muestreo es la máxima posible con esta rutina del MCU y la arquitectura del VI, en este punto se da por cumplido el objetivo del proyecto. Para esta última prueba se hace una

evaluación diferente para determinar la frecuencia de muestreo, ya que la aparición de señales alias se presenta de manera gradual, comenzando por la deformación de la señal de manera tal que la señal reconstruida asemeja una señal modulada en amplitud, ya que la componente de frecuencia aun no se ha perdido pero la forma de la onda se ha modificado, esto se aprecia en la figura 3.2, cabe destacar que para generar esta imagen se recurre a un VI ejemplo creado por National Instruments (Resampling a Waveform.vi) el cual muestra mediante una simulación el efecto del remuestreo de una señal, y también el efecto que produce incrementar la frecuencia de la señal muestreada hasta llegar a un valor cercano a la frecuencia de Nyquist, es aquí cuando se pierde una cantidad de información importante y la forma de la onda reconstruida ya no es igual a la forma de la señal original.

La figura 3.2 es producto de una simulación y representa lo que debería ocurrir en la señal reconstruida en cualquier caso en que la frecuencia de la señal original se acerque a la frecuencia de Nyquist.

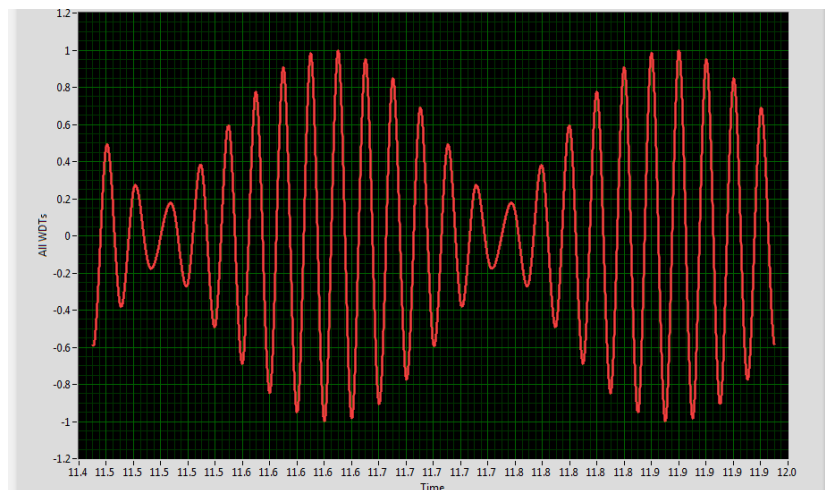


Figura 3.2. Efecto en la señal reconstruida cuando la frecuencia de la señal original se aproxima a la frecuencia de Nyquist.

De esta simulación también obtenemos información de lo que ocurre en la señal reconstruida cuando la frecuencia de la señal original es igual a la frecuencia de Nyquist como se muestra en la figura 3.3, estos son los efectos que se buscan al realizar la última prueba del sistema.

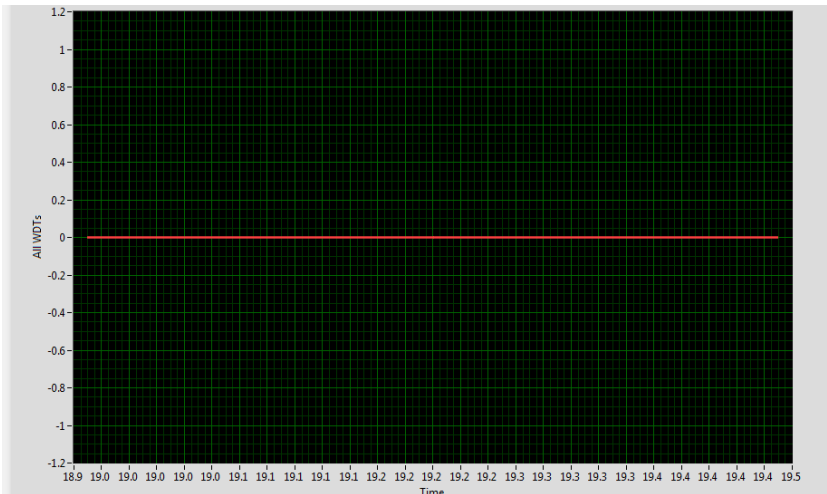


Figura 3.3. Efecto en la señal reconstruida cuando la frecuencia de la señal original es igual a la frecuencia de Nyquist.

Al sobrepasar la frecuencia de Nyquist aparecen señales alias, en las que la información de la componente de frecuencia se ha perdido por efecto del submuestreo y se observara en la señal reconstruida una frecuencia más baja que en la señal original.

Ya que no fue posible determinarse la frecuencia de muestreo con el método que se describen en la sección 2.6, dado que no era viable el uso de la función “Waveform Graph”, se aplico un método que no se describe en la sección 2.6, este consiste en graficar la señal en la función “Waveform Chart” y exportar los datos a una hoja de cálculo de Excel para graficar solo una porción de estos, como se observa en las figuras 3.4, 3.5, 3.6, no es posible determinar la aparición de los efectos mencionados anteriormente haciendo uso de la función “Waveform Chart”, por esta razón son llevados a Excel para buscar los efectos del submuestreo y determinar la frecuencia de muestreo, la cantidad de datos que la función puede exportar se limita a 1024, que es el tamaño de memoria de datos asignado en el VI, de los que solo se extraen 60 datos, la interpolación de esta cantidad limitada de datos afecta la reconstrucción de la señal pero se sabe que el VI está recibiendo muestras continuas y que el hardware de medidas tiene una frecuencia de Nyquist estimada en 12KHz

por lo que se efectúa un barrido de frecuencias hasta llegar a 12KHz, sabiendo que los efectos del submuestreo se presentaran al estar cerca de esta frecuencia. Cabe destacar que la frecuencia que se busca es la frecuencia de Nyquist y por lo tanto la frecuencia de muestreo será el doble de esta.

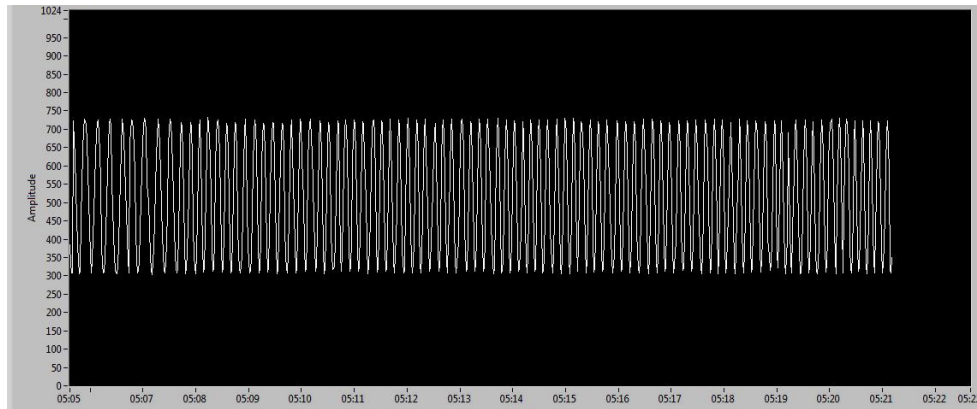


Figura 3.4. Señal reconstruida @ 50Hz.

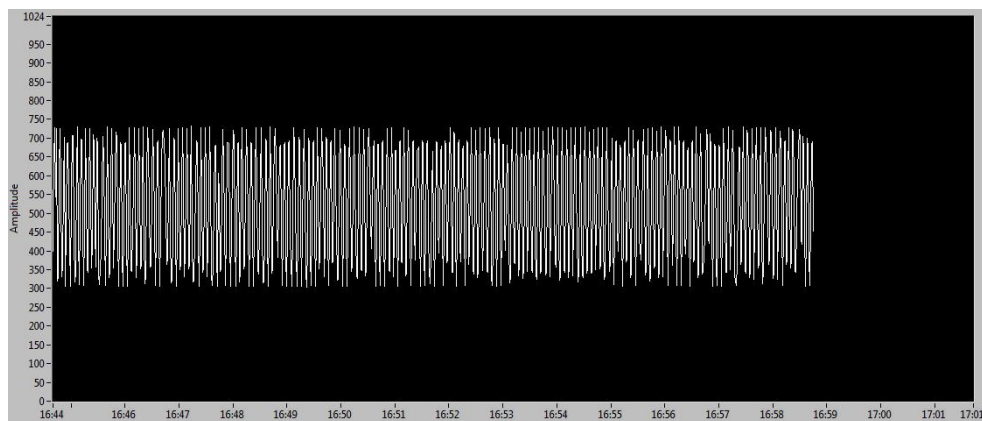


Figura 3.5. Señal reconstruida @ 100Hz.

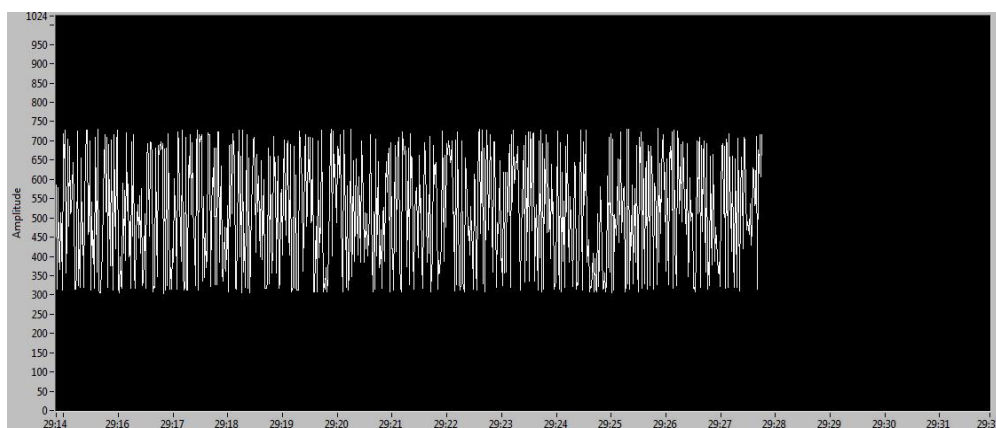


Figura 3.6. Señal reconstruida @ 10KHz.

A continuación se muestran las graficas generadas en Excel de los datos obtenidos de la función “Waveform Chart” en las figuras 3.7, 3.8, y 3.9.

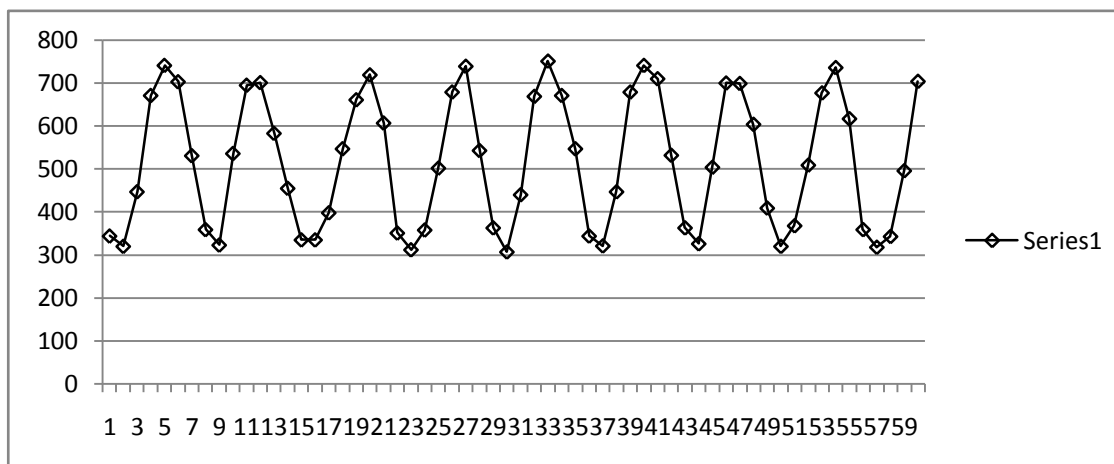


Figura 3.7. Interpolación de los datos de la señal @ 50Hz realizada en Excel.

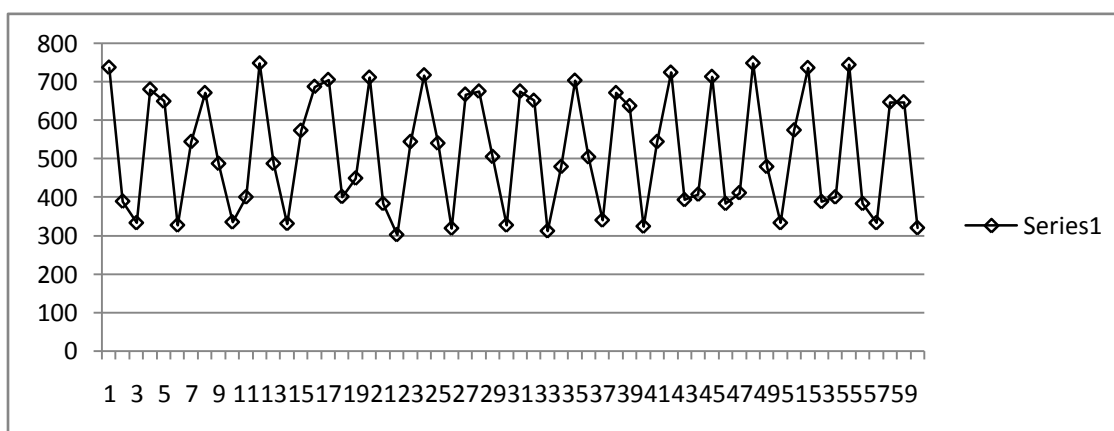


Figura 3.8. Interpolación de los datos de la señal @ 100Hz realizada en Excel.

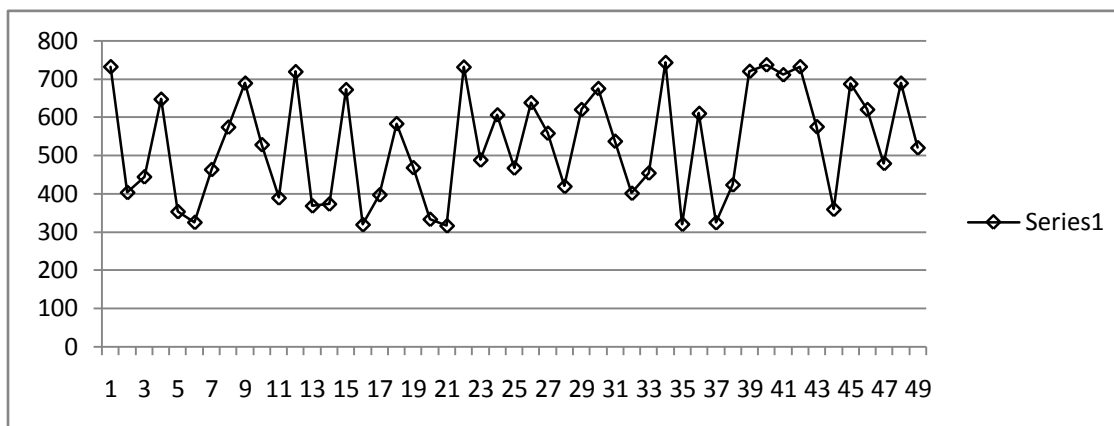


Figura 3.9. Interpolación de los datos de la señal @ 10KHz realizada en Excel.

Del análisis de las graficas se concluye que los efectos del submuestreo no son los que se esperaban (semejantes a los que se muestran en la figura 3.1) debido a la interpolación que realiza la función “Waveform Chart”, sin embargo se logra observar un efecto semejante al que se presenta en la figura 3.2 a una frecuencia de 12KHz, como se esperaba, con esto se concluye que la frecuencia de Nyquist del sistema es igual a la del hardware de medidas. Esto se observa en las figuras 3.10 y 3.11.

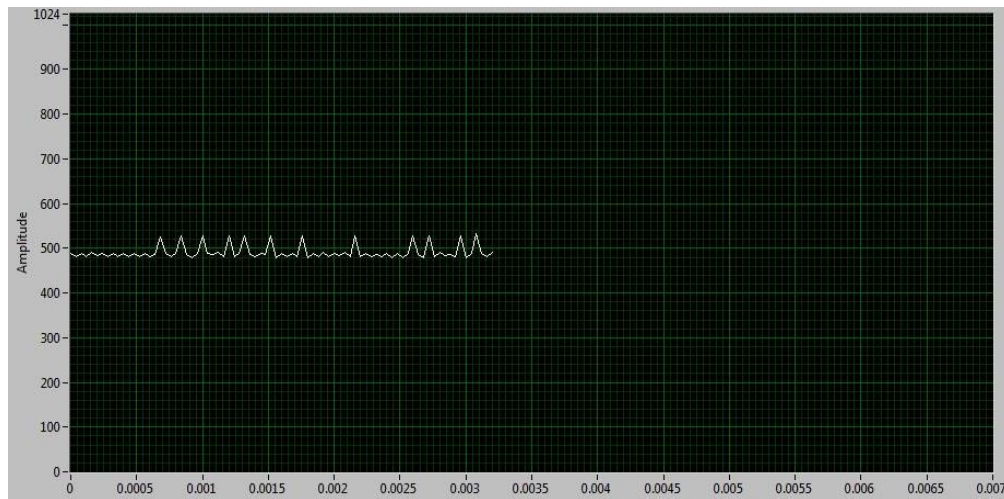


Figura 3.10. Señal reconstruida @ 12KHz

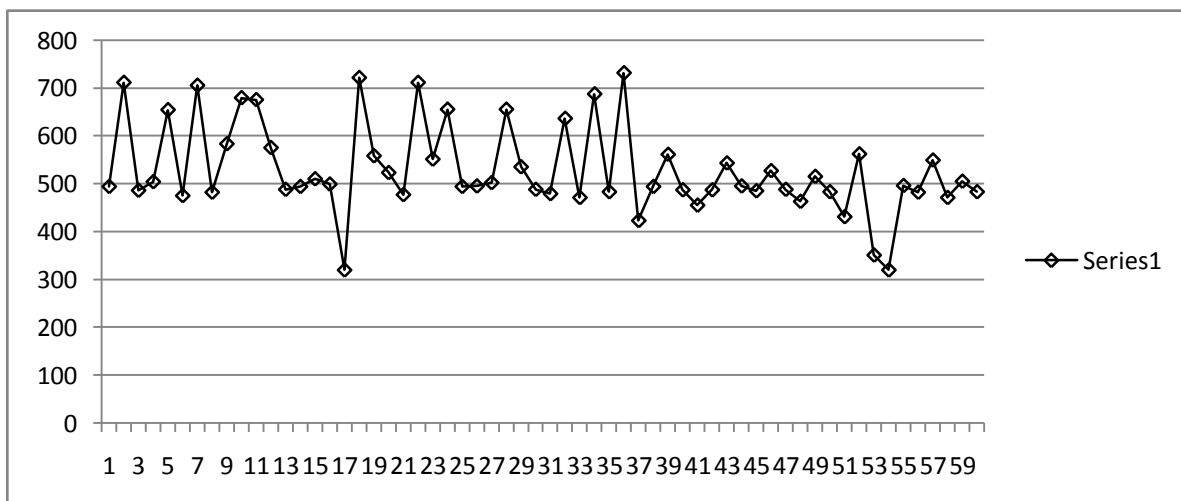


Figura 3.11. Interpolación de los datos de la señal @12KHz realizada en Excel.

En la tabla 3.14 se muestra una comparación entre cada código y VI probado y las mejoras obtenidas.

Tabla 3.14. Comparación entre los códigos y las mejoras obtenidas.

Código del MCU	VI	Circuito eléctrico	Etapas de prueba	Cambios aplicados	Mejoras obtenidas
1	1	1	Primaria	-	-
2	1	1	Primaria	-	-
3	1	1	Primaria	Se sustituyen las estructuras if anidadas por la combinación de la estructuras if y while.	Disminución del tiempo de ejecución de 7.8 ms a 100 μ s.
4	1	1	Primaria	Se sustituye la estructura while por la estructura do-while y se descartan las estructuras if anidadas.	Disminución del tiempo de ejecución de 8 ms a 7ms.
5	2	1	Primaria	Se eliminan el bit de inicio y paro, se descartan todas las estructuras anidadas, se eliminan funciones de entrada y salida en el VI.	Disminución del tiempo de ejecución de 100 μ s a 44.6 μ s
5.1	2	1	Primaria	Se reduce el tiempo de espera de 10 μ s a 7 μ s.	Disminución del tiempo de ejecución de 44.6 μ s a 40.2 μ s.
5.1	3	1	Primaria	Se añade la función "Shift register" al VI.	Es posible usar la función "Waveform Graph".
5.1	4	1	Primaria	Se aplica la estructura "Flat sequence" al VI.	Ninguna.
5.1	5	1	Primaria	Se reubican las funciones dentro de la estructura "Flat sequence".	Se obtienen los datos de manera continua.
5.1	3	1	Final	No hay cambios.	Ninguna.
6	6	1	Primaria	Se cambia el tipo de transferencia y se adecuan los códigos del MCU y el VI.	Mayor resolución del ADC, la posibilidad de usar la estructura "Flat sequence".
6.1	6	2	Primaria	Se adecua el circuito para el aterrizamiento del ADC y se adecua el código del MCU.	Se solucionan los errores ocasionados por ruido analógico.
6.1	6	2	Final	Se adecua el VI para graficar la señales adquiridas y buscar señales alias.	Ninguna.

Capítulo IV. Conclusiones

4.1. Introducción

Este capítulo presenta las conclusiones a las que se ha llegado a lo largo del desarrollo de este proyecto.

4.2. Conclusiones

- De la evaluación de resultados se observa que al reducir el tamaño y complejidad de las estructuras de programación se obtienen rutinas del MCU más eficientes, por lo tanto se concluye que la forma más eficiente de realizar el muestreo es por medio de una estructura secuencial.
- Del análisis de las estructuras implementadas en los códigos del VI y el MCU se observa que es posible tomar muestras en un intervalo de tiempo con en el hardware de medidas y enviar el paquete de datos al VI para ser desplegados, sin embargo este método implica un retraso significativo en la visualización de los datos y no sería posible observar cambios en la variable en tiempo real por lo que se concluye que es más apropiado realizar un muestreo continuo, así como el envío de datos, para tener la posibilidad de observar cambios en la variable con un retraso mínimo.
- Inicialmente se implementa el tipo de transferencia “Interrupt” y al evaluar el rendimiento obtenido se concluye que este tipo de transferencia no es adecuado para esta aplicación.
- La incertidumbre en las mediciones existe, esto es inherente a la naturaleza de cualquier instrumento y los instrumentos virtuales no están exentos de esto, se concluye que la medición a través de un VI es susceptible a mas fuentes de error que un instrumento común, a pesar de esto las ventajas que ofrece un VI superan por mucho a un instrumento real.
- De la mejora aplicada al circuito eléctrico del hardware de medidas se destaca la complejidad para identificar problemas relacionados con el aterrizamiento, dado que inicialmente se suponía que el problema era atribuible a un lazo de tierra, se concluye que para la identificación de problemas el método de mejora continua debe asegurar la correcta implementación de los sistemas, en este caso, de los circuitos eléctricos, para poder centrar la atención en el desarrollo de los códigos.

- En este caso se logro la máxima eficiencia posible en el proceso de muestreo con códigos relativamente sencillos y con pocas instrucciones, sin embargo se plante una arquitectura escalable para el VI, por lo tanto el código del VI que se presenta es el código mínimo necesario para la adquisición de datos con este hardware de medidas y es posible añadir funciones al VI para manipular los datos pero no es posible añadir funciones para realizar un proceso de control por lo que se concluye que este sistema solo es apto para realizar mediciones y deben añadirse instrucciones al código del MCU si se desea controlar algún sistema, pero esto disminuirá la velocidad de muestreo significativamente.
- Las buenas prácticas de programación señalan que el desarrollo de un código debe apegarse a arquitecturas de programación que puedan ser fácilmente identificadas y el código debe ser documentado para que usuarios diferentes tengan la posibilidad de añadir código, se observa que las estructuras de los códigos del MCU y del VI se apegan a estructuras conocidas que pueden ser fácilmente identificadas, por lo que se concluye que el sistema es apto para que cualquier usuario adapte el código para aplicaciones específicas.
- Del cambio aplicado en el circuito eléctrico del hardware de medidas se observa que se soluciona el problema sin embargo no se implementan todas las recomendaciones ya que el circuito fue implementado en una tablilla de pruebas, se concluye que el circuito debe ser implementado en PCB para realizar de manera correcta el aterrizamiento del ADC, adicionalmente a esto deben seguirse la reglas y normas de diseño del layout, el PCB puede ser implementado en una solo capa, pero para obtener el mejor resultado posible debe implementarse un PCB multicapa.

4.3. Observaciones

- Al concretarse la hipótesis planteada se da por cumplido el objetivo del proyecto en su totalidad.
- El proceso de mejora continua que aquí se plantea comienza su implementación al ser parte fundamental del diseño del sistema, las directrices de diseño que se definen tiene como objetivo principal lograr la mejora y la máxima eficiencia del sistema, la metodología de pruebas es el medio para retroalimentar el proceso de mejora y verificar que los resultados que se obtienen conducen a la concreción de los

objetivos, partiendo de la evaluación de resultados se aplican cambios al sistema con base en las directrices de diseño.

- En relación a las preguntas que se plantean para desarrollar la hipótesis se observa que existen diferentes métodos para desarrollar un sistema de adquisición de datos, sin embargo implementar procesos de mejora continua y fijar objetivos es clave para obtener los resultados deseados.
- En tanto a la pregunta de si un microcontrolador es el dispositivo adecuado se destaca que los objetivos son logrados con este, no obstante, es posible obtener mejores resultados migrando a arquitecturas multiprocesador o arquitecturas basadas en FPGA (sistema digital programable por sus siglas en inglés Field Programmable Gate Array), o utilizando dispositivos DSP (procesador digital de señal por sus siglas en inglés Digital Signal Processor) preferentemente arquitecturas que posean módulos DMA (modulo de acceso directo a memoria por sus siglas en inglés Direct Memory Access) y los periféricos para utilizar protocolos de comunicación (es importante que los módulos estén embebidos ya que esto facilita el desarrollo de los códigos y circuitos eléctricos), con estas arquitecturas y dispositivos es posible obtener mejores resultados ya que al hacer uso de estos recursos (como el modulo DMA) se incrementa la eficiencia del hardware de medidas.
- Inicialmente se plante la interrogante de cual es protocolo de comunicación adecuado, se observa que es crítico para el sistema ya que el retraso que provoca la transmisión debe ser considerado si se realizan procesos de control a través de un VI (para realizar procesos de control es recomendable minimizar los tiempos de transmisión tanto en entradas como de salidas), y en cualquier proceso de medición a través de un VI, por lo que el protocolo debe ser seleccionado de acuerdo al uso que se le dé al sistema.

4.4. Recomendaciones

En el desarrollo de proyectos futuros se hacen la siguiente recomendación. El sistema de adquisición de datos debe ser diseñado con objetivos concisos en relaciona su aplicación, es decir, si será usado únicamente para realizar mediciones o también se realizaran procesos de control, ya que esta aplicación implica la necesidad de desarrollar sistemas altamente eficientes.

4.5. La instrumentación dando forma al futuro

A lo largo del desarrollo del proyecto se llega a conclusiones relacionadas a los aspectos técnicos del sistema, sin embargo, también se llega a conclusiones relacionadas con el impacto de esta tecnología y su interacción con otras prácticas, por lo que a continuación el autor presenta su opinión sobre la trascendencia que tendrán la instrumentación y la electrónica en los ámbitos social y económico.

Tanto en el desarrollo científico como en los procesos industriales la necesidad de conocer cómo se desarrollan los fenómenos o procesos demanda datos e información, la instrumentación provee una solución a dicha demanda, pero la necesidad de tener datos más confiables aumentara, aún a sabiendas de que la pérdida de información es inherente al proceso de digitalización, por ello deben desarrollarse dispositivos que acorten la brecha entre el mundo analógico y el mundo digital, ya que las actividades humanas dependerán cada vez mas de esta información; actualmente existen equipos con la capacidad de generar 1GB de información cada segundo, producto de la digitalización de señales, esto es un claro indicio de que la tecnología y los métodos de digitalización continuaran reduciendo dramáticamente la brecha entre el mundo analógico y el mundo digital, pero esto acarrea una serie de problemas adicionales, como el almacenamiento, resguardo, transmisión, clasificación y análisis de dicha información, y no tendremos que esperar a que llegue este futuro, ya que en el presente son un problema real y tangible.

La clasificación y el análisis son pieza clave en el propósito de obtener dicha información, ya que estos dos son los procesos que convertirán los datos y la información en conocimiento y su vez este se convertirá en métodos, tecnologías, técnicas y aparatos que son útiles en las actividades humanas. Las personas dependen de esta información para su vida diaria como es el ejemplo del estado del tiempo, que será consultado para tomar decisiones simples de cómo vestir, y cuyas consecuencias no tendrán mayores implicaciones, en contraste con el ejemplo de un médico, que necesita la ayuda de un instrumento para obtener información del cuerpo humano y poder salvar la vida de una persona. La generación de información precisa y exacta es más necesaria en ciertos aspectos de las actividades humanas y en algunos casos indispensable, pero el uso cotidiano de información aparentemente sencilla se

convertirá en una necesidad, uno de los principales indicadores de esta tendencia es el incremento en el uso de esta información en la vida diaria, que va del uso esporádico al uso cotidiano.

De acuerdo con el periódico español ABC que publico el 13 de febrero de 2015 en su sección de tecnología, un artículo relacionado con el decimo aniversario de “Google Maps”, los usuarios de este servicio en cualquiera de sus modalidades pasaron de diez millones en su primer año a más de mil millones de usuarios activos cada mes en 2015, esto pone de manifiesto la constante demanda de información, y advierten un incremento continuo en el uso cotidiano de dicha información.

El futuro de la instrumentación está directamente relacionado a la demanda de cualquier tipo de información y el medio por el cual es solicitada. El internet de las cosas es un concepto difícil de entender ya que incluye a todos y cada uno de los usuarios de la información los cuales son simultáneamente generadores y consumidores de dicha información, uno de los retos que enfrenta el internet de las cosas es la clasificación de esta información, y se debe a su gran tamaño por a la cantidad de información que alberga, es un reto no solo desde el punto de vista técnico, sino aún más importante, desde la óptica del usuario que es incapaz de discriminar la información que le sirve de la que no. Esta gran cantidad de información desde la perspectiva del prestador de servicios debe ser filtrada para suministrar al usuario la información adecuada según su idioma, edad, género, etnia, nivel de educación, poder adquisitivo, gustos y necesidades, todos estos parámetros son seleccionados por el usuario y monitoreados por el prestador de servicios gracias a la instrumentación de sus actividades, al monitorear los lugares que frecuenta, la cantidad de dinero que gasta, las búsquedas que realiza, la frecuencia con la que se ejerce y los instrumentos que hacen posible este constante monitoreo los llevamos con nosotros día a día, integrados en nuestros teléfonos, autos, computadoras, ropa, y calzado esto muestra que la instrumentación ha pasado de ser una práctica ocasional que se aplicaba solo a ciertas actividades, a una práctica cotidiana que se aplica a casi cualquier actividad de una persona.

Inminentemente la electrónica y todas sus ramas incluida la instrumentación han cambiado al mundo y lo seguirán haciendo, el precedente más cercano a la transformación que estamos observando en las actividades humanas se remonta a la revolución industrial, que data del siglo XVIII y que generó una desestabilización social, sin embargo para la humanidad las tecnologías desarrolladas por la electrónica son un fenómeno nuevo del cual no podemos predecir sus efectos a largo plazo, pero podemos observar efectos importantes en el ámbito económico, la demanda de tecnología y servicios de información son factor clave en la creación y desarrollo de compañías que han llegado a tener un valor semejante al del mercado de valores de un país completo, tal es el caso “Apple” que el 15 de noviembre de 2014 su valor ascendía por encima del valor de toda la bolsa de valores rusa [9], esto demuestra que el impacto económico que ha tenido la electrónica es incluso de mayor trascendencia, que la misma tecnología que se ha desarrollado a través de los años, ya que esta se consume y desecha a la velocidad que se genera una nueva, por el contrario los beneficios económicos que esta nueva revolución tecnológica ha traído siguen rindiendo frutos, pero con base en los precedentes históricos que existen, es de esperarse nuevos cambios, sin embargo la dependencia en la electrónica no se verá afectada, ya que esta se ha convertido en la herramienta y el medio de todos los logros recientes de la humanidad.

La realidad que hoy vivimos es inconcebible sin la electrónica y aún menos sin la instrumentación, por esta razón el poder que ha cobrado la tecnología desarrollada por la electrónica es la clave para establecer el rumbo que tomara la evolución humana en el ámbito social y económico.

Quizás en el futuro la instrumentación sea una herramienta para extender los límites de la percepción sensorial, o para entender cómo funciona el universo, pero cada variable que sea instrumentada generara información, e inminentemente conocimiento y poder. El internet de las cosas acerca la información a cualquier persona en el mundo y se convertirá en una entidad con un gran valor económico que cobre conciencia y adquiera poder, pero darle poder a todos en el mundo es lo que menos deseamos, así que surge una pregunta ¿Encomendara la humanidad a una máquina el resguardo del estatus quo?

Glosario

IC. Circuito integrado (por sus siglas en inglés Integrated Circuit).

HP-IB. nombre del bus de datos (por sus siglas en inglés Hewlett-Packard Instrument Bus).

GPIB. Nombre del bus de datos (por sus siglas en inglés General Purpose Instrumentation Bus).

IEEE-488. Nombre del estándar de comunicación.

IEEE. Organización dedicada a la ingeniería eléctrica y electrónica (por sus siglas en inglés the Institute of Electrical and Electronics Engineers).

VI. Instrumento virtual (por sus siglas en inglés Virtual Instrument).

ADC. Convertidor analógico digital (por sus siglas en inglés Analog Digital Converter).

DAC. Convertidor digital analógico (por sus siglas en inglés Digital Analog Converter).

MCU. Unidad de microcontrolador (por sus siglas en inglés Micro Controller Unit).

NI-Visa. Complemento de LabVIEW.

PCB. Circuito impreso (por sus siglas en inglés Printed Circuit Board).

Cross talk. Nombre con el que se denomina la interferencia electromagnética producida por dos pistas o dos pines contiguos en un circuito impreso.

Layout. Es la forma en que están dispuestos los componentes de un circuito electrónico en la tablilla de un circuito impreso.

FPGA. Sistema digital programable (por sus siglas en inglés Field Programmable Gate Array).

DSP. Procesador digital de señales (por sus siglas en inglés Digital Signal Processor).

DMA. Modulo de acceso directo a memoria (por sus siglas en inglés Direct Memory Access).

Referencias y bibliografía

- [1] Alan V. Oppenheim, Alan S. Willsky, Señales y Sistemas, segunda edición, 1999.
- [2] Antonio Creus Solé, Instrumentación Industrial, sexta edición, alfaomega marcombo, 1997.
- [3] Albert D. Helfrick, William D. Cooper, Instrumentación electrónica moderna y técnicas de medición, Prentice Hall Hispanoamerica, primera edición, 1991.
- [4] Joaquín de Río Fernández, Shahram Sharat-Panahi, David sarria gandul, Antoni Manuel Lázaro, LabVIEW programación para sistemas de instrumentación, primera edición, alfa-omega, 2012.
- [5] Juan Ignacio Huircán, Conversores Análogo-Digital y Digital-Análogo: Conceptos básicos.
- [6] Understanding and minimising ADC conversion errors, application note AN1636, STMicroelectronics, 2003.
- [7] Jan Axelson, USB Complete the developers guide, cuarta edición, Lakeview Research LLC, 2009.
- [8] PIC18F2455/2550/4455/4550 Data Sheet, Microchip Technology Inc., 2009.
- [9] Tyler Durden, Apple is now worth more than the entire Russian stock market, Business insider, www.bussinessinsider.com, 2014.
- [10] CCS C Compiler Manual PCB/PCM/PCH, CCS Inc., Noviembre 2013.
- [11] Corina Schmelkes, Manual para la presentación de anteproyectos e informes de investigación, segunda edición, 1998.
- [12] Dogan Ibrahim, Advanced PIC Microcontroller Projects in C from USB to RTOS with the PIC18F series, primera edición, Elsevier Ltd., 2008.
- [13] Eduardo García Berijo, Compilador C CCS y simulador Proteus para microcontroladores PIC, primera edición, 2008.
- [14] Curtis D. Johnson, Process Control Instrumentation Technology, eighth edition, 2006.

- [15] Antoni Mànuel, Domingo Biel, Joaquim Olivé, Jordi Prat, Francesc J. Sánchez, instrumentación virtual.
- [16] Instrumentación Virtual e Instrumentación Tradicional, application note NI-Tutorial-4757-es, National Instruments, febrero 7, 2008.
- [17] Guía Sobre Control de Instrumentos Por USB, application note NI-Tutorial-4478-es, National Instruments, abril 21, 2009.
- [18] Como Escoger el Bus Adecuado para su Sistema de Medida, application note NI-Tutorial-9401-es, National Instruments, octubre 18, 2012.
- [19] Fundamentos de Control de Instrumentos: Software de control de Instrumentos, application note NI-Tutorial-4478-es, National Instruments, enero 17, 2013.
- [20] Como Escoger el Hardware DAQ adecuado para su Sistema de Medidas, application note NI-Tutorial-13655-es, National Instruments, junio 7, 2013.
- [21] Serial Instrument Control Tutorial, application note NI-Tutorial-2897-en, National Instruments, junio 8, 2011.
- [22] Avoiding Ground Loops, application note NI-Tutorial-3035-en, National Instruments, junio 6, 2013.
- [23] Six Things You Need to Know about USB Instrument Control, application note NI-Tutorial-4878-en, National Instruments, agosto 23, 2013.
- [24] Acquiring an Analog Signal: Bandwidth, Nyquist Sampling Theorem, and Aliasing, application note NI-Tutorial-2709-en, National Instruments, mayo 4, 2015.
- [25] Grounding Considerations for Improved Measurements, application note NI-Tutorial-3394-en, National Instruments, may 4 2015.
- [26] Isolation Types and Considerations when Taking a Measurement, application note NI-Tutorial-3410-en, National Instruments, may 4 2015.
- [27] Analog Sample Quality: Accuracy, Sensitivity, Precision and Noise, application note NI-Tutorial-4018-en, National Instruments, may 4 2015.

[28] Sanjay Pithadia, Shridhar More, Grounding in mixed-signal systems demystified Part 1, Analog Applications Journal, Texas Instruments, 2013.

[29] Sanjay Pithadia, Shridhar More, Grounding in mixed-signal systems demystified Part 2, Analog Applications Journal, Texas Instruments, 2013.

Anexo A. Códigos

Parte 1. Códigos del MCU.

Código 1. Rutina sin el uso del servicio de interrupciones.

```
1  #include <18F4550.h>
2  #fuses HSPLL, NOWDT, NOPROTECT, NOLVP, NODEBUG, USBDIV, PLL5, CPUDIV2, VREGEN, NOMCLR
3  #device adc=8 //configuracion del ADC justificado a la derecha, se desprecian los dos bit mas significativos.
4  #use delay (clock=48000000) //configuracion del oscilador del MCU.
5  #DEFINE USB_HID_DEVICE TRUE //se habilita el uso como dispositivo HID en la libreria usb_dec_hid.h.
6  #DEFINE USB_EP1_TX_ENABLE USB_ENABLE_INTERRUPT //se habilita el tipo de transferencia interrupt.
7  #DEFINE USB_EP1_TX_SIZE 32
8  #DEFINE USB_EP1_RX_ENABLE USB_ENABLE_INTERRUPT
9  #DEFINE USB_EP1_RX_SIZE 32
10 #DEFINE USB_CONFIG_HID_TX_SIZE 4
11 #DEFINE USB_CONFIG_HID_RX_SIZE 4
12 #DEFINE USB_CONFIG_PID 0x0000 //se definen el valor ProductID la importancia de este se puede consultar en [7] .
13 #DEFINE USB_CONFIG_VID 0x0000 //se define el valor VendorID la importancia de este se puede consultar en [7] .
14 #include <pic18_usb.h> //libreria necesaria para usar el modulo y las funciones usb.
15 #include <usb_desc_hid.h> //libreria de descriptores.
16 #include <usb.c> //libreria necesaria para usar el modulo y las funciones usb.
17
18 int8 ssb[1]; //aquí se almacenara la señal proveniente de la pc para iniciar o detener la transmisión de datos.
19 int8 conv[1]; //en esta variable se almacena el valor de la conversión A/D.
20
21 void main() {
22     set_tris_b(0x00); //configura el puerto B como salida.
23     output_b(0x00); //pone el puerto B en 0.
24     output_high(PIN_B0); //pin D0 indica que el dispositivo esta encendido pero no enumerado.
25     setup_adc_ports(AN0|VSS_VDD); //config de canal analogico y voltajes de referencia.
26     setup_adc(ADC_CLOCK_DIV_64); //config de reloj del convertidor.
27     set_tris_d(0x00); //configura el puerto D como salida.
28     output_d(0x00); //pone el puerto D en 0.
29     output_high(PIN_D0);
30     output_low(PIN_D1);
31     usb_init(); //inicializa el modulo usb.
32     usb_wait_for_enumeration(); //espera en un ciclo infinito, sale del ciclo hasta que el dispositivo este enumerado.
33     output_low(PIN_D0); //cambia a estado bajo.
34     output_high(PIN_D1); //cambia a estado alto indicando que el dispositivo ya esta enumerado.
35     while(true) {
36         output_high(PIN_B1); //inicia el ciclo while---B1.
37         trigger: if(usb_kbhit(1)==true) { //verifica si hay datos en el epl.
38             usb_get_packet(1, &ssb[0], 1); //lee los datos y los almacena.
39         }
40         else
41             goto trigger; //si no hay datos espera en el ciclo.
42
43         if(ssb[0]==1) { //si el dato recibido es 1, se interpreta como señal de disparo e inicia el proceso.
44             set_adc_channel(0); //se selecciona el canal analogico.
45             delay_us(10); //Tmin=6.35us, Tmin=Tacq+3Tad, Tacq-minimo=2.45us, 3Tad=tiempo minimo entre una conversion y otra.
46             conv[0]=read_adc(); //inicia la conversion y espera el resultado Tad=1.3us, la conversio de 10-bits dira 11Tad.
47             usb_put_packet(1, &conv[0], 1, USB_DTS_TOGGLE); //envia el paquete con los datos de la conversion A/D.
48         }
49         else
50             goto trigger; //regresa a actualizar los datos enviados por la pc y esperar la señal de disparo.
51
52         output_low(PIN_B1); //termina el ciclo while---B1.
53     }
54 }
```

Código 2. Rutina con el uso del servicio de interrupciones.

```

1  #include <18F4550.h>
2  #fuses HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL5,CPUDIV2,VREGEN,NOMCLR
3  #device adc=8 //configuracion del ADC justificado a la derecha, se desprecian los dos bit mas significativos.
4  #use delay (clock=48000000) //configuracion del oscilador del MCU.
5  #define USB_HID_DEVICE TRUE //se habilita el uso como dispositivo HID en la libreria usb_desc_hid.h.
6  #define USB_EP1_TX_ENABLE USB_ENABLE_INTERRUPT //se habilita el tipo de transferencia interrupt.
7  #define USB_EP1_TX_SIZE 32
8  #define USB_EP1_RX_ENABLE USB_ENABLE_INTERRUPT
9  #define USB_EP1_RX_SIZE 32
10 #define USB_CONFIG_HID_TX_SIZE 4
11 #define USB_CONFIG_HID_RX_SIZE 4
12 #define USB_CONFIG_PID 0x0000 //se definen el valor ProductID la importancia de este se puede consultar en [7] .
13 #define USB_CONFIG_VID 0x0000 //se define el valor VendorID la importancia de este se puede consultar en [7] .
14 #include <pic18_usb.h> //libreria necesaria para usar el modulo y las funciones usb.
15 #include <usb_desc_hid.h> //libreria de descriptores.
16 #include <usb.c> //libreria necesaria para usar el modulo y las funciones usb.
17
18 int8 ssb[1]; //aquí se almacenara la señal proveniente de la pc para iniciar o detener la transmisión de datos.
19 int8 conv[1]; //en esta variable se almacena el valor de la conversión A/D.
20
21 #INT_AD
22 void AD_isr() {
23     conv[0]=read_adc(ADC_READ_ONLY); //rutina de servicio de interrupcion por adc, recupera el dato convertido y lo almacena en una variable.
24 }
25
26 void main() {
27     setup_adc_ports(AN0|VSS_VDD); //seleccion del canal AN0 y voltajes de referencia VSS y VDD.
28     setup_adc(ADC_CLOCK_DIV_64); //config del rezo del convertidor.
29     enable_interrupts(INT_AD); //habilita interrupcion por adc.
30     enable_interrupts(GLOBAL); //habilita servicio de interrupciones globales.
31     set_tris_d(0x00); //configura el puerto D como salida.
32     output_d(0x00); //pone el puerto D en 0.
33     output_high(PIN_D0); //pin D0 indica que el dispositivo esta encendido pero no enumerado.
34     output_low(PIN_D1); //pin D1 indica que el dispositivo esta encendido y enumerado.
35     usb_init(); //inicializa modulo USB.
36     usb_wait_for_enumeration(); //espera en un ciclo infinito, sale del ciclo hasta que el dispositivo este enumerado.
37     output_low(PIN_D0); //cambia a estado bajo.
38     output_high(PIN_D1); //cambia a estado alto indicando que el dispositivo ya esta enumerado.
39     while(true) {
40         set_adc_channel(0); //se activa el canal analogico.
41         delay_us(10); //espera necesaria 3Tad=tiempo minimo entre conversiones, la conversion de 10-bits dura 11Tad.
42         read_adc(ADC_START_ONLY); //solo inicia la conversión, en cuanto la conversion termina y el dato esta listo
43         //se recupera mediante la rutina de interrupción.
44     }
45     trigger: if(usb_kbhit(1)==true) { //verifica si hay datos provenientes de la pc, este dato es de tipo int1, y se interpreta
46         //como bit de inicio o paro.
47         usb_get_packet(1,&ssb[0],1); //recupera el dato y lo almacena.
48     }
49     else
50         goto trigger; //espera en el ciclo hasta que aparezca el bit de inicio.
51
52     if(ssb[0]==1) { //si el bit es 1 se envian los datos (1 es equivalente al bit de inicio).
53         usb_put_packet(1,&conv[0],1,USB_DTS_TOGGLE); //envia los datos a la pc.
54     }
55     else
56         goto trigger; //si el bit es 0 regresa al principio del ciclo para
57         //refrescar el dato proveniente de la pc (0 es equivalente al bit de paro).
58 }
59 }

```

Código 3. Rutina sin el uso del servicio de interrupciones.

```

1  #include <18F4550.h>
2  #fuses HSPLL, NOWDT, NOPROTECT, NOLVP, NODEBUG, USBDIV, PLL5, CPUDIV2, VREGEN, NOMCLR
3  #device adc=8
4  #use delay (clock=48000000)
5  #define USB_HID_DEVICE TRUE
6  #define USB_EP1_TX_ENABLE USB_ENABLE_INTERRUPT
7  #define USB_EP1_TX_SIZE 32
8  #define USB_EP1_RX_ENABLE USB_ENABLE_INTERRUPT
9  #define USB_EP1_RX_SIZE 32
10 #define USB_CONFIG_HID_TX_SIZE 64
11 #define USB_CONFIG_HID_RX_SIZE 64
12 #define USB_CONFIG_PID 0x0000
13 #define USB_CONFIG_VID 0x0000
14 #include <pic18_usb.h>
15 #include <usb_desc_hid.h>
16 #include <usb.c>
17
18 int8 ssb[1];
19 int8 conv[1];
20
21 void main() {
22     set_tris_d(0x00);
23     output_d(0x00);
24     setup_adc_ports(AN0|VSS_VDD);
25     setup_adc(ADC_CLOCK_DIV_64);
26     output_high(PIN_D0);
27     output_low(PIN_D1);
28     usb_init();
29     usb_wait_for_enumeration();
30     output_low(PIN_D0);
31     output_high(PIN_D1);
32
33     trigger: if(usb_kbhit(1)==true){
34         usb_get_packet(1, &ssb[0], 1);
35     }
36     else
37         goto trigger;
38
39     while(ssb[0]==1){
40         output_high(PIN_D2);
41         set_adc_channel(0);
42         delay_us(10);
43         conv[0]=read_adc();
44         usb_put_packet(1, &conv[0], 1, USB_DTS_TOGGLE);
45         output_low(PIN_D2);
46     }
47     goto trigger;
48 }

```

Código 4. Rutina con el uso del servicio de interrupciones.

```

1  #include <18F4550.h>
2  #fuses    HSPLL, NOWDT, NOPROTECT, NOLVP, NODEBUG, USBDIV, PLL5, CPUDIV2, VREGEN, NOMCLR
3  #device   adc=8
4  #use      delay (clock=48000000)
5  #define   USB_HID_DEVICE    TRUE
6  #define   USB_EP1_TX_ENABLE USB_ENABLE_INTERRUPT
7  #define   USB_EP1_TX_SIZE   32
8  #define   USB_EP1_RX_ENABLE USB_ENABLE_INTERRUPT
9  #define   USB_EP1_RX_SIZE   32
10 #define   USB_CONFIG_HID_TX_SIZE 64
11 #define   USB_CONFIG_HID_RX_SIZE 64
12 #define   USB_CONFIG_PID       0x0000
13 #define   USB_CONFIG_VID       0x0000
14 #include  <pic18_usb.h>
15 #include  <usb_desc_hid.h>
16 #include  <usb.c>
17
18 int8  ssb[1];
19 int8  conv[1];
20
21 #INT_AD
22 void AD_isr() {
23     output_high(PIN_D2);
24     conv[0]=read_adc(ADC_READ_ONLY);
25     usb_put_packet(1, &conv[0], 1, USB_DTS_TOGGLE);
26     output_low(PIN_D2);
27 }
28
29 void main() {
30     set_tris_d(0x00);
31     output_d(0x00);
32     setup_adc_ports(AN0|VSS_VDD);
33     setup_adc(ADC_CLOCK_DIV_64);
34     enable_interrupts(INT_AD);
35     enable_interrupts(GLOBAL);
36     output_high(PIN_D0);
37     output_low(PIN_D1);
38     usb_init();
39     usb_wait_for_enumeration();
40     output_low(PIN_D0);
41     output_high(PIN_D1);
42     do {
43         set_adc_channel(0);
44         delay_us(10);
45         read_adc(ADC_START_ONLY);
46         if(usb_kbhit(1)==true){
47             usb_get_packet(1, &ssb[0], 1);
48         }
49     }
50     while(ssb[0]==1);
51 }

```

Código 5. Rutina sin el uso del servicio de interrupciones.

```
1  #include <18F4550.h>
2  #fuses    HSPLL, NOWDT, NOPROTECT, NOLVP, NODEBUG, USBDIV, PLL5, CPUDIV2, VREGEN, NOMCLR
3  #device    adc=8
4  #use      delay (clock=48000000)
5  #define    USB_HID_DEVICE    TRUE
6  #define    USB_EP1_TX_ENABLE USB_ENABLE_INTERRUPT
7  #define    USB_EP1_TX_SIZE    32
8  #define    USB_EP1_RX_ENABLE USB_ENABLE_INTERRUPT
9  #define    USB_EP1_RX_SIZE    32
10 #define    USB_CONFIG_HID_TX_SIZE  64
11 #define    USB_CONFIG_HID_RX_SIZE  64
12 #define    USB_CONFIG_PID          0x0000
13 #define    USB_CONFIG_VID          0x0000
14 #include    <pic18_usb.h>
15 #include    <usb_desc_hid.h>
16 #include    <usb.c>
17
18 int8    conv[1];
19
20 void main() {
21     set_tris_d(0x00);
22     output_d(0x00);
23     setup_adc_ports(AN0|VSS_VDD);
24     setup_adc(ADC_CLOCK_DIV_64);
25     output_high(PIN_D0);
26     output_low(PIN_D1);
27     usb_init();
28     usb_wait_for_enumeration();
29     output_low(PIN_D0);
30     output_high(PIN_D1);
31
32     while(true) {
33         output_high(PIN_D2);
34         set_adc_channel(0);
35         delay_us(10);
36         conv[0]=read_adc();
37         usb_put_packet(1, &conv[0], 1, USB_DTS_TOGGLE);
38         output_low(PIN_D2);
39     }
40 }
```

Código 5.1. Rutina sin el uso del servicio de interrupciones.

```
1  #include <18F4550.h>
2  #fuses    HSPLL, NOWDT, NOPROTECT, NOLVP, NODEBUG, USBDIV, PLL5, CPUDIV2, VREGEN, NOMCLR
3  #device   adc=8
4  #use      delay (clock=48000000)
5  #define   USB_HID_DEVICE    TRUE
6  #define   USB_EP1_TX_ENABLE USB_ENABLE_INTERRUPT
7  #define   USB_EP1_TX_SIZE   32
8  #define   USB_EP1_RX_ENABLE USB_ENABLE_INTERRUPT
9  #define   USB_EP1_RX_SIZE   32
10 #define   USB_CONFIG_HID_TX_SIZE 64
11 #define   USB_CONFIG_HID_RX_SIZE 64
12 #define   USB_CONFIG_PID        0x0000
13 #define   USB_CONFIG_VID        0x0000
14 #include <pic18_usb.h>
15 #include <usb_desc_hid.h>
16 #include <usb.c>
17
18 int8 conv[1];
19
20 void main() {
21     set_tris_d(0x00);
22     output_d(0x00);
23     setup_adc_ports(AN0|VSS_VDD);
24     setup_adc(ADC_CLOCK_DIV_64);
25     output_high(PIN_D0);
26     output_low(PIN_D1);
27     usb_init();
28     usb_wait_for_enumeration();
29     output_low(PIN_D0);
30     output_high(PIN_D1);
31     while(true) {
32         output_high(PIN_D2);
33         set_adc_channel(0);
34         delay_us(7);
35         conv[0]=read_adc();
36         usb_put_packet(1,&conv[0],1,USB_DTS_TOGGLE);
37         output_low(PIN_D2);
38     }
39 }
```

Código 6. Rutina sin el uso del servicio de interrupciones y tipo de transferencia Bulk.

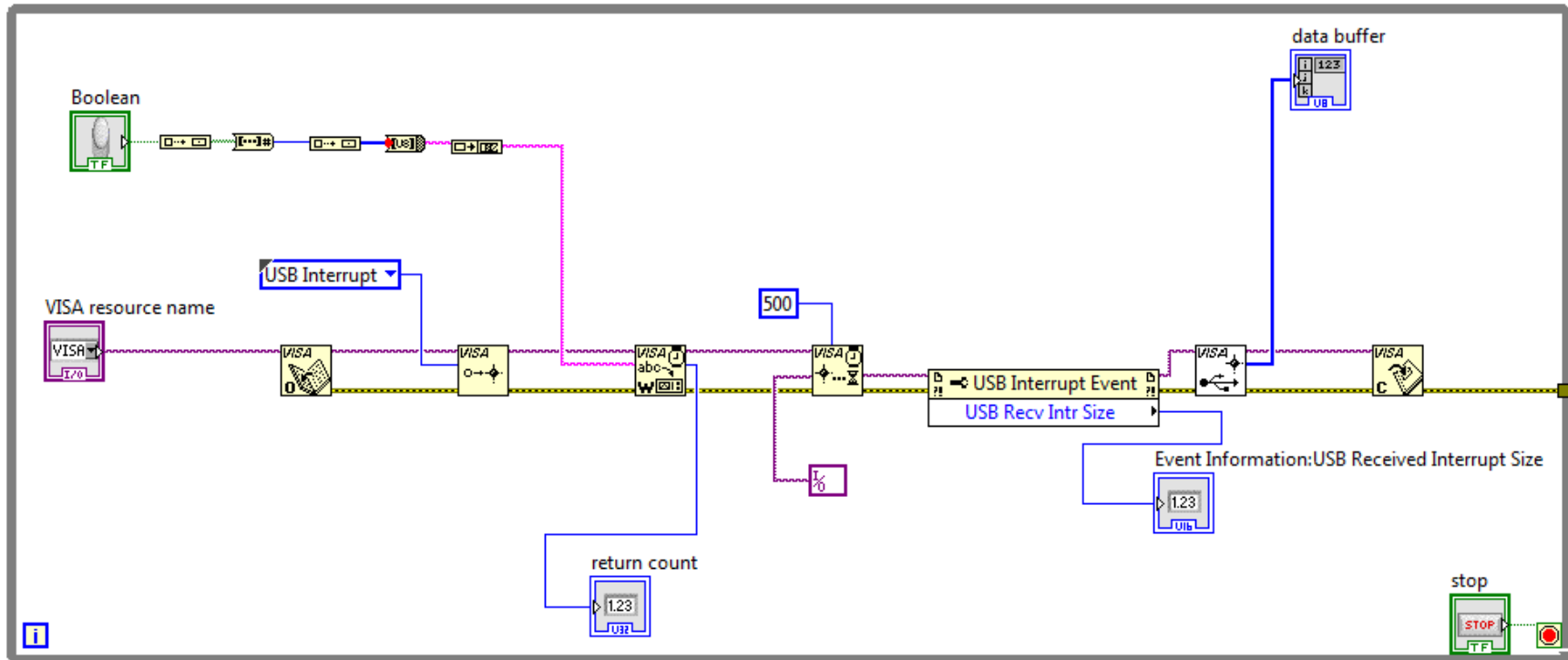
```
1  #include <18F4550.h>
2  #fuses    HSPLL, NOWDT, NOPROTECT, NOLVP, NODEBUG, USBDIV, PLL5, CPUDIV2, VREGEN, NOMCLR
3  #device   adc=10
4  #use      delay (clock=48000000)
5  #include  <pic18_usb.h>
6  #include  <usb_desc_bulk.h>
7  #include  <usb.c>
8
9  int16 conv;
10 char value[17]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
11
12 void main() {
13     set_tris_d(0x00);
14     output_d(0x00);
15     setup_adc_ports(AN0|VSS_VDD);
16     setup_adc(ADC_CLOCK_DIV_64);
17     output_high(PIN_D0);
18     output_low(PIN_D1);
19     usb_init();
20     usb_wait_for_enumeration();
21     output_low(PIN_D0);
22     output_high(PIN_D1);
23     while(true) {
24         set_adc_channel(0);
25         delay_us(7);
26         conv=read_adc();
27         sprintf(value,"%lu",conv);
28         usb_put_packet(1,value,4,USB_DTS_TOGGLE);
29     }
30 }
```

Código 6.1. Rutina sin el uso del servicio de interrupciones, con el pin Vref- habilitado y tipo de transferencia Bulk.

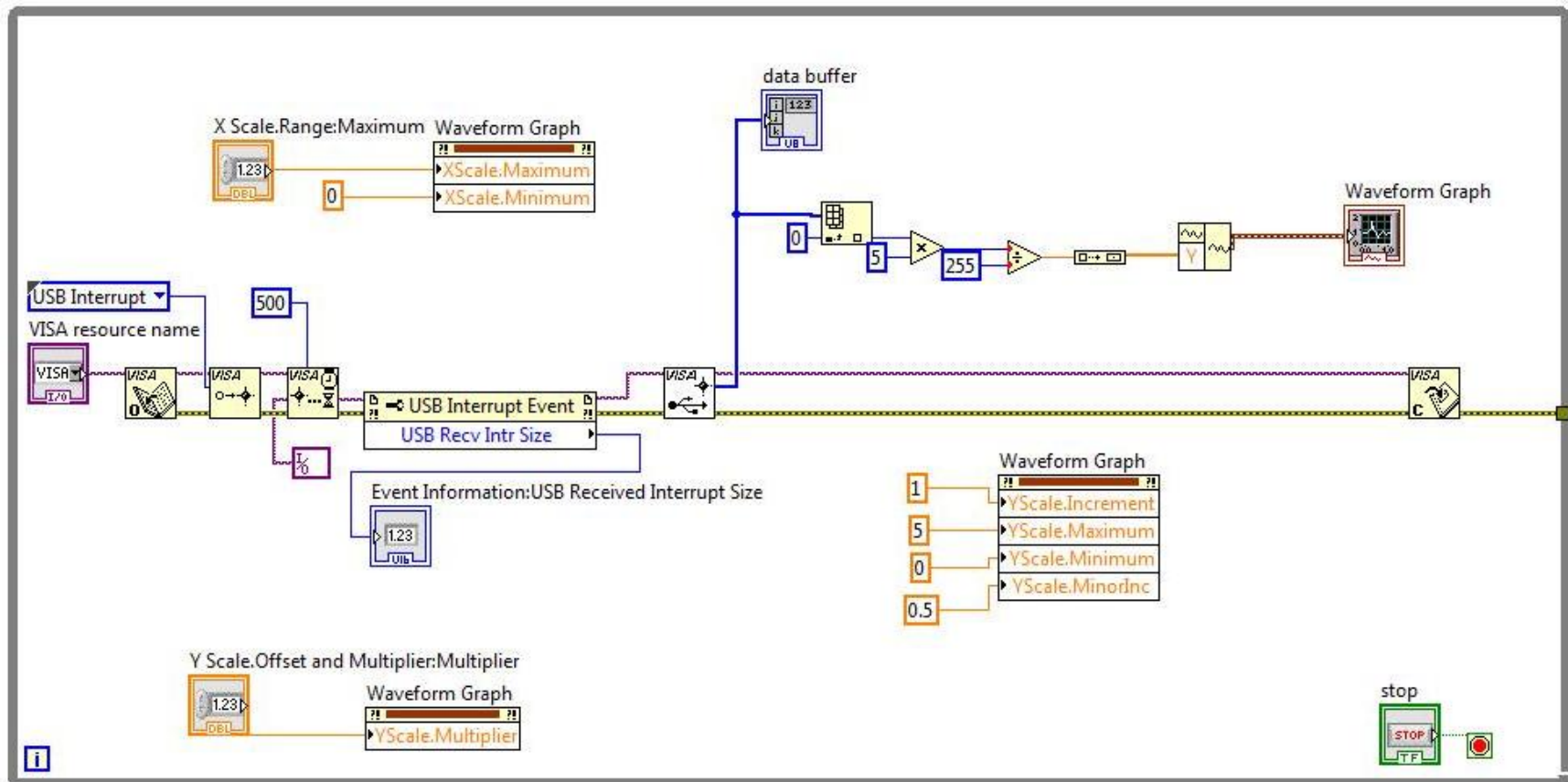
```
1  #include <18F4550.h>
2  #fuses    HSPLL, NOWDT, NOPROTECT, NOLVP, NODEBUG, USBDIV, PLL5, CPUDIV2, VREGEN, NOMCLR
3  #device    adc=10
4  #use      delay (clock=48000000)
5  #include <pic18_usb.h>
6  #include <usb_desc_bulk.h>
7  #include <usb.c>
8
9  int16 conv;
10 char value[17]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
11
12 void main() {
13     set_tris_d(0x00);
14     output_d(0x00);
15     setup_adc_ports(AN0|VREF_VDD);
16     setup_adc(ADC_CLOCK_DIV_64);
17     output_high(PIN_D0);
18     output_low(PIN_D1);
19     usb_init();
20     usb_wait_for_enumeration();
21     output_low(PIN_D0);
22     output_high(PIN_D1);
23     while(true) {
24         set_adc_channel(0);
25         delay_us(7);
26         conv=read_adc();
27         sprintf(value,"%1u",conv);
28         usb_put_packet(1,value,4,USB_DTS_TOGGLE);
29     }
30 }
```

Parte 2. Códigos del VI.

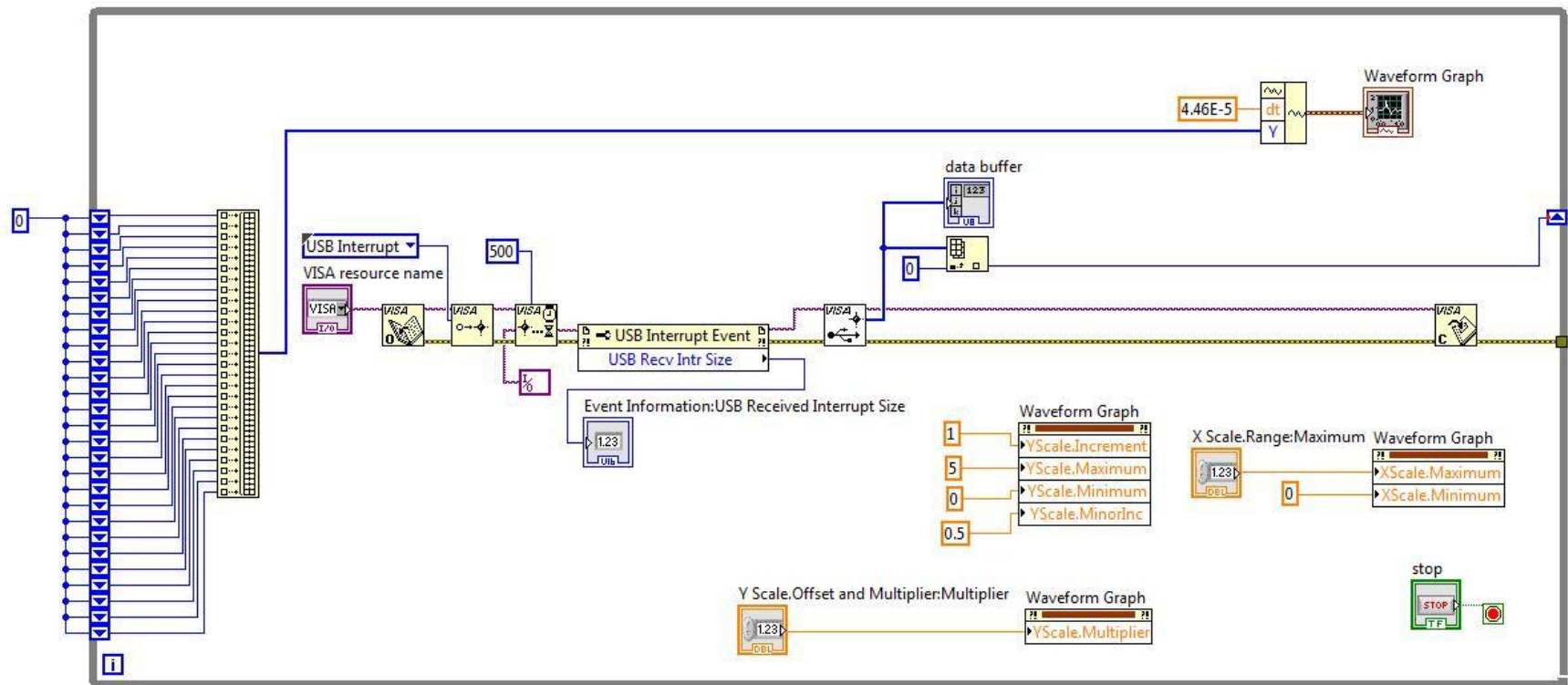
VI 1.



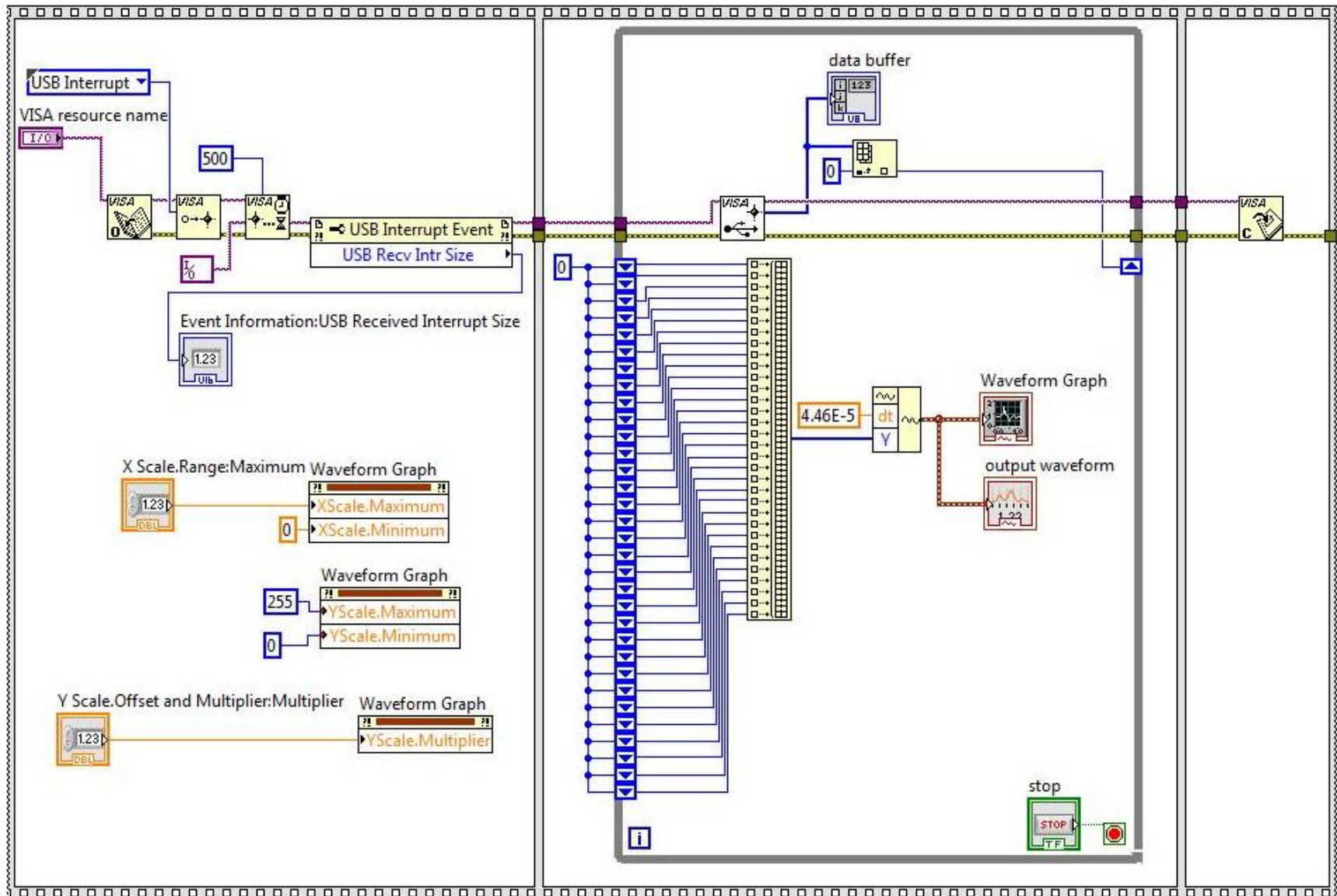
VI 2.



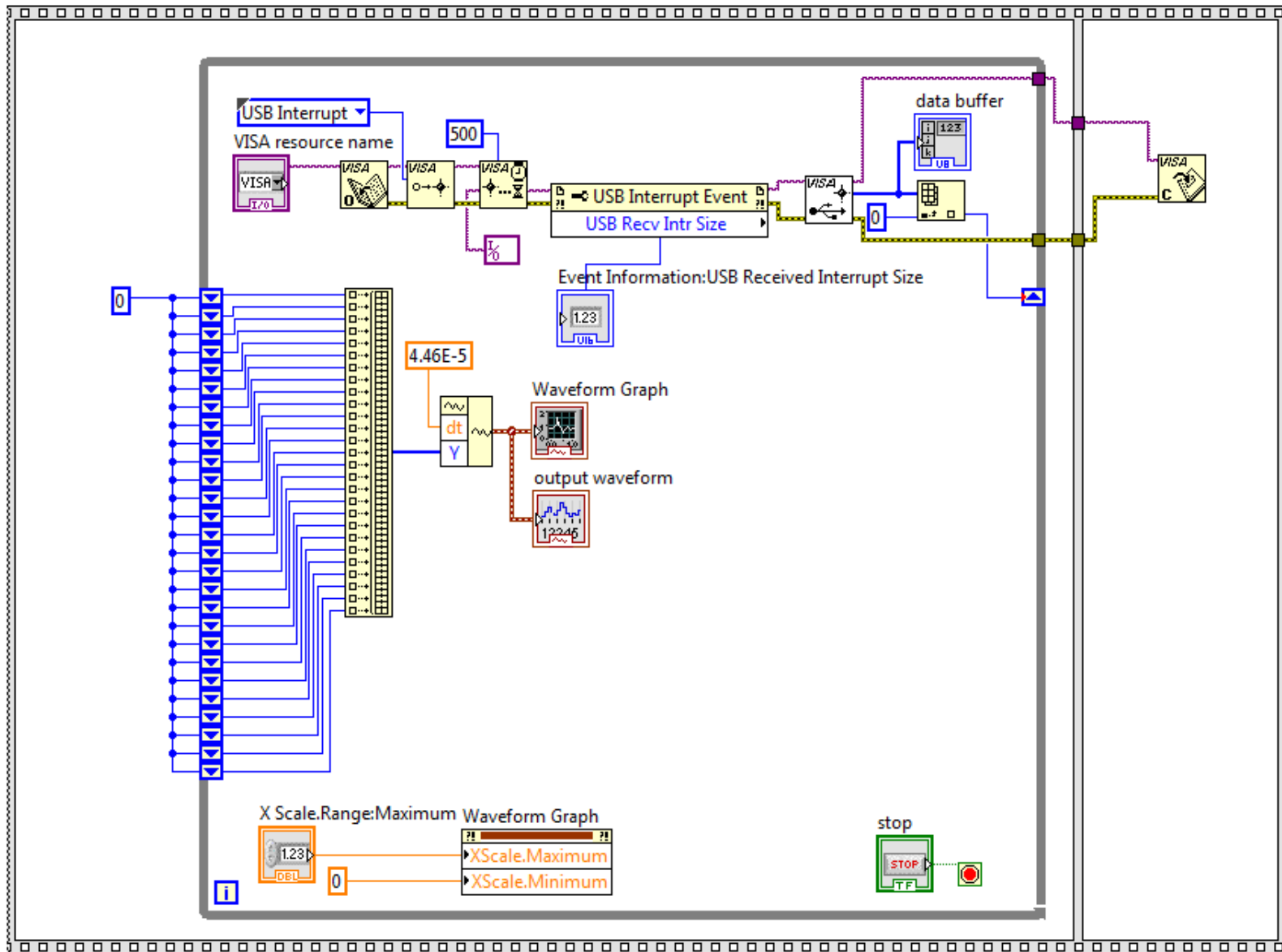
VI 3.



VI 4.



VI 5.



VI 6.

