Desarrollo de un Generador de Señales de Forma de Onda Arbitraria Aplicado a Espectrometría Mössbauer

Víctor Manuel Rico Botero

Universidad del Valle
Facultad de Ciencias Naturales y Exactas
Programa Académico de Física
Santiago de Cali, Colombia
2014

Desarrollo de un Generador de Señales de Forma de Onda Arbitraria Aplicado a Espectrometría Mössbauer

Víctor Manuel Rico Botero

Trabajo de Grado presentado como requisito parcial
para optar al título de Físico

Director:
Otto Vergara García, *Dr. Rer. Nat.*

Universidad del Valle
Facultad de Ciencias Naturales y Exactas
Programa Académico de Física
Santiago de Cali, Colombia
2014

UNIVERSIDAD DEL VALLE
FACULTAD DE CIENCIAS
ACTA DE EVALUACION DE TRABAJO DE GRADO
PROGRAMA ACADEMICO DE FISICA.

JURADO CONFORMADO POR:

DR. OTTO VERGARA GARCÍA
DR. JESÚS ANSELMO TABARES GIRALDO

El día 10 de Febrero de 2014 a las 2:00 P.M. se llevó a cabo la ENTREGA DE LA EVALUACION ( ) SUSTENTACION (**X**) del INFORME FINAL DEL TRABAJO DE GRADO titulado ***"DESARROLLO DE UN GENERADOR DE SEÑALES DE FORMA DE ONDA ARBITRARIA APLICADO A ESPECTROMETRÍA MÔSSBAUER".*** Presentado por el estudiante VICTOR MANUEL RICO BOTERO código 07336470, bajo la dirección del Dr. OTTO VERGARA GARCÍA.

RESULTADO DE LA EVALUACION:

(X) **APROBADO .**
( ) **REPROBADO.**

Se recomienda modificaciones: SI ( ) NO (X).

OBSERVACIONES:

OTTO VERGARA GARCÍA
DIRECTOR

JESÚS ANSELMO TABARES
JURADO

JESUS MARIA CALERO QUINTERO
DIRECTOR
PROGRAMA ACADEMICO DE FISICA

Agradecimientos

Alguna vez Andrea me contó que los seres humanos somos en realidad una combinación de todas las personas con las que nos relacionamos; ahora que estoy un poquito más grande, creo que la combinación de la que ella hablaba debe ser ponderada puesto que hay indudablemente unas personas que influyen en las decisiones y en la forma de ser más que otras. Quizás si este trabajo no lo hubiera hecho yo, lo habría hecho tarde que temprano otro estudiante, pero lo cierto es que en lo personal probablemente no habría sido posible sin mamá, quien siempre ha creido en mí y quien me apoyó en la empresa de hacer carrera profesional sin pensar en dinero y sin cuestionar sí sería rentable a pesar de todas las dificultades que, para bien o para mal, nos ha tocado pasar. Creo que nunca se lo he dicho, pero tengo una admiración grandísima por la humildad con la que siempre ha afrontado todo y el cariño inconfundible que expresa hacia mi. Gracias mamá por cuidarme, por esperarme en las noches y madrugar sin necesidad, por escucharme y por estar siempre conmigo. Tampoco habría sido posible sin las enseñanzas que ha dejado en mí la Universidad y que espero aplicar, recrear y replicar con entusiasmo; no habría sido posible sin el profesionalismo de los profesores, de los funcionarios con los que me topaba a diario y que de una u otra forma trabajaron para que este trabajo se pudiera escribir. Gracias a todos por las sonrisas, y la amable gestión. Tampoco puedo dejar pasar la oportunidad de agradecer a los compañeros sin quienes todo hubiese sido diferente. Finalmente, la realización de este trabajo no habría sido posible sin la valiosa, siempre presta, amable y gentil orientación del profesor Otto Vergara, quien desde los cursos de Programación y Teoría de Circuitos, siempre perfilé como un profesor de verdad, de esos que, como dice Gibrán: *acercan a los estudiantes al umbral de su propia sabiduría*. Gracias profesor por compartir conmigo ese ahínco por la experimentación, por enseñarme a hacer las cosas bien hechas, por mostrarme que el desarrollo de un país como el nuestro depende no de quienes tienen capacidad adquisitiva, sino de aquellos que se desvelan curiosiando y que a partir de imaginación, criterio y una profunda dedicación al trabajo, construyen con sus manos lo que necesitan; además, gracias por recibirme sin titubear en el grupo de investigación.

Resumen

En este trabajo se presenta el diseño de un generador de forma de onda arbitraria, desarrollado principalmente para controlar el movimiento de la fuente radiactiva de un sistema de espectroscopía Mössbauer así como las señales de sincronización de su analizador multicanal. Se incorporó un puerto de comunicaciones USB a través del cual el equipo intercambia datos con una computadora personal, en la que se ejecuta una aplicación que permite al usuario definir la forma matemática que modela el periodo de la forma de onda a reproducir. La evaluación de su funcionamiento se realizó mediante pruebas de calidad con osciloscopio y a través de tres experimentos de espectroscopía Mössbauer en los que se estudió la presencia de hierro en una muestra de referencia, en una muestra de óxido de bulevar del rio Cali y en una muestra de óxido de una estructura metálica del bosque municipal de Palmira que se encuentra en proceso de corrosión.

Abstract

We show in this work the design of an arbitrary waveform generator developed mainly to control the movement of the radioactive source involved on a Mössbauer spectroscopy system and also the synchronization signals of its multichannel analyzer. We added an USB communication port that serves to interchange data with a personal computer. The computer runs an application that allows the user define the mathematical shape of the wave reproduced. We performed two operation test, by examining the quality with a digital oscilloscope and through three experiments of Mössbauer spectroscopy. In these three experiments, we studied the presence of iron in a reference sample, in oxide samples from Cali river's boulevard and from a metallic structure under corrosion process located on Palmira's municipal woodland.

Índice general

Capítulo 1

Introducción

La idea de desarrollar un generador de onda arbitraria surge de la necesidad del Grupo de Metalurgia Física y Transiciones de Fase (GMTF) de la Universidad del Valle, de conectar su nuevo analizador multicanal (el Easy-MCS[1]) al sistema de instrumentación Mössbauer del laboratorio. Esta tarea abarca la generación de una onda triangular que controle el movimiento de la fuente de radiación y la producción de un par de señales de reloj que sincronizan dicho movimiento con la apertura de canales. Pensando en dar solución general al problema y no únicamente en la construcción de un generador de onda triangular, nos propusimos en diseñar un equipo electrónico que fuera capaz de reproducir no solo las formas de onda que se consiguen con los generadores de señales convencionales, sino además cualquier señal periódica cuyo periodo se pueda describir como función del tiempo, lo que resulta relativamente útil para la realización de varios experimentos[2][3] entre los que se incluye la espectrometría Mössbauer[4]. Ese problema fué identificado y solucionado en el trabajo de Sanchez[5], quien junto a sus colegas[6] solucionaron las necesidades de instrumentación Mössbauer en la Universidad del Valle; y aunque no puede negarse la capacidad instrumental de su espectrómetro, la falta de modularidad de algunos de sus componentes y los pasos agigantados con los que ha avanzado la tecnología en los últimos 20 años, dificulta hoy su integración con nuevos sistemas de procesamiento.
El equipo que se describe en este trabajo aporta justamente un granito de arena para la consolidación de un sistema de espectrometría Mössbauer que esté a la vanguardia de los nuevos desarrollos tecnológicos. En él se ha implementado el protocolo de comunicaciones USB -que permite la conexión con computadoras modernas-, se ha usado electrónica reciente y se ha escrito software en lenguajes de programación que dilucidan su funcionamiento.

La mayor parte del desarrollo de hardware que se presenta y que ha si-

do determinante para su felíz término, es fundamentalmente la experiencia adquirida en el trabajo del profesor Sanchez y del trabajo realizado por el Grupo de Instrumentación y Física Aplicada (GIFA), en la automatización de varias prácticas de laboratorio de Física de la Universidad del Valle, en particular por los resultados del trabajo ASLAB (ASistente de LABoratorio)[7] con el que se desarrolla en la actualidad los laboratorios de Física Fundamental I. Resalta además el trabajo de Ojeda[8] quien en 2009 realizó un trabajó con un generador de forma de onda que opera bajo el mismo fundamento del que aquí se presenta.

Para poner en contexto al lector acerca del equipo implementado, conviene recordar que los generadores de señales son instrumentos ampliamente utilizados en laboratorios de docencia y en actividades de investigación y calibración entre otras. Una de las bondades que presentan estos equipos y que explican su masificación, es la capacidad para generar diferencias de potencial que varían en el tiempo con frecuencias que van desde el orden de los Hertz(Hz) hasta los GigaHerthz(GHz); generalmente un mecanismo (en el uso extendido de la palabra), permite conmutar entre diferentes formas de onda que, por lo regular, se limitan a funciones sinusoidales, triangulares, cuadradas y las variantes que de ellas resultan. Aunque estos patrones de señal satisfacen las necesidades de un sinnúmero de aplicaciones, hoy existen en el mercado equipos que ofrecen una solución general a este problema; esos equipos, llamados *Generadores de Forma de Onda Arbitraria*, permiten configurar la curva que se repite periódicamente en la señal y que se extiende en todo su periodo. A diferencia de los generadores se señal convencionales que emplean circuitos específicos basados en osciladores analógicos, los generadores de forma de onda arbitraria emplean una arquitectura de procesamiento digital articulado con un sistema de conversión digital-analógica para realizar esta tarea; con esto, la generación de señales se reduce a un ejercicio de programación arbitrado por un procesador central.

La estrategia metodológica estudiada típicamente para realizar lo anterior es la llamada *Síntesis Digital Directa* (DDS)[9][10], en la que se tabulan y almacenan en una memoria cada uno de los datos numéricos que dan cuenta del patrón de onda a generar; capturando estos datos y escribiéndolos en el puerto de entrada de un conversor digital análogo (DAC) a una tasa definida por el usuario, ha de reproducirse -al menos idealmente- la señal esperada. Sin embargo la situación real que se observa cuando se sintetiza la forma de onda, revela que a pesar que el algoritmo es simple -pues solo involucra movimiento de datos- aparecen efectos no deseados tipo glitch que pueden evitarse de entrada sí se elige un diseño electrónico adecuado.

La aplicación Mössbauer hacia la que se ha orientado este trabajo requiere que el generador de forma de onda tenga unas características muy particulares; por una parte, como la forma de onda controla el movimiento del actuador el cual debe desarrollar aceleración constante[11](Figura 1.1), se debe garantizar que la forma de onda sea lo más continua y rectilínea posible para que el movimiento de la muestra radiactiva sea suave, esto es evitando movimientos bruscos que pueden implicar espectros ruidosos. Por otra parte, la señal controladora del actuador debe estar sincronizada con una señal de disparo (*Trigger*), que marca el inicio de un nuevo periodo, y con una señal de avance (*Advance*) que debe tener tantos ciclos como canales se esperan barrer en el periodo de la forma de onda. Estas dos últimas señales son de tipo cuadrado y las emplea el analizador multicanal, para relacionar un número de cuentas indicado por el detector, con un canal dado.
La figura 1.2 resume en modo gráfico, las partes constitutivas del proyecto, las cuales se discuten en detalle en el capítulo 2. Para darle completez al trabajo, se han dispuesto unos anexos que documentan algunas partes importantes que por extensión y pertinencia, conviene mejor dejarlos al final para el lector interesado. La idea principal de esos anexos, es facilitar -si así se quiere- la reproducción del trabajo o de alguna de sus partes.

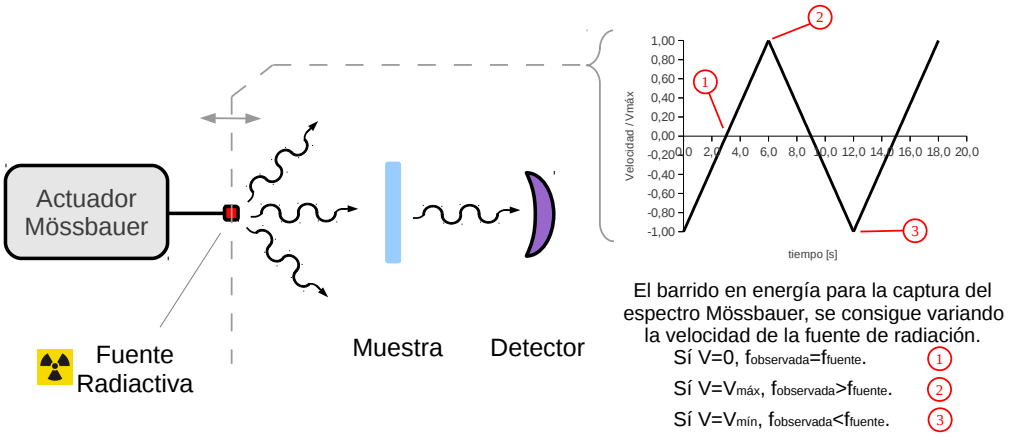


Figura 1.1: Bosquejo experimental típico para el desarrollo de experimentos de espectroscopía Mössbauer.

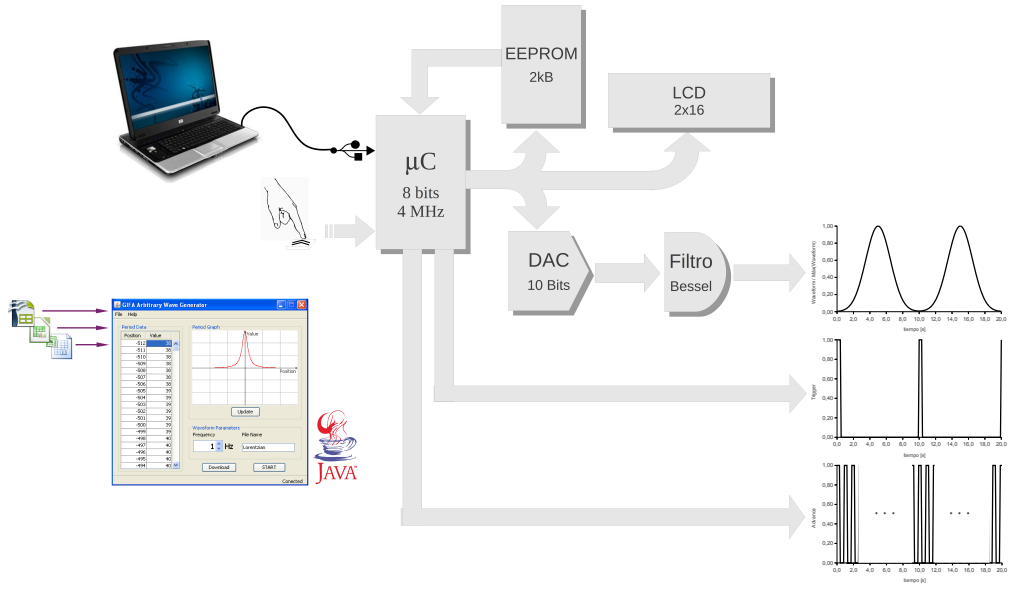


Figura 1.2: Resumen gráfico del trabajo implementado para la realización del generador de forma de onda arbitraria.

Capítulo 2

Prototipo Implementado

El prototipo implementado está formado por dos componentes: una de hardware, que incluye fuente de alimentación, procesador, memoria y un circuito analógico que convierte y adecúa los datos digitales que arroja el procesador, y otra de software con la que se administra el hardware y se realiza la comunicación con una computadora y con el usuario.
El hardware se desarrolló para contener en un solo circuito impreso la mayoría de los componentes electrónicos del equipo; sólo los conectores BNC, el interruptor de encendido/apagado, el transformador y el conector de potencia, se han conectado de forma aérea para reducir la probabilidad de fallas por fatiga mecánica.

2.1. Hardware

2.1.1. Fuente de potencia

El hardware ha sido diseñado con un conjunto de fuentes de potencia que alimenta de forma independiente los circuitos analógicos y digitales. La potencia se obtiene de la red de distribución de electricidad, a través de un transformador monofásico que reduce la tensión de entrada (115 V_{rms} @ 60 Hz) a niveles de 9 V_{rms} y 6 V_{rms} referidos al punto medio del devanado secundario (o tab-central). Los conectores Molex *TR_PRI1* (Figura 2.1b) y *TR_SEC1* (Figura 2.1a) sirven de puente entre los devanados del transformador y la tarjeta electrónica.
Un puente rectificador, seguido del arreglo condensadores → bomba de diodos → regulador, para el caso de las fuentes de +5 V_{DC}, +12 V_{DC} y -12 V_{DC}, convierte la onda sinosoidal del devanado secundario del transformador en los niveles de tensión continua mencionados. La fuente de -5 V_{DC} se ha

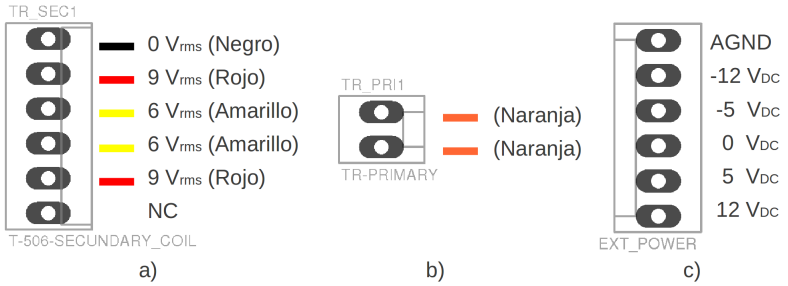


Figura 2.1: Distribución de cables de los conectores molex del transformador. a) devanado secundario, b) devanado primario.

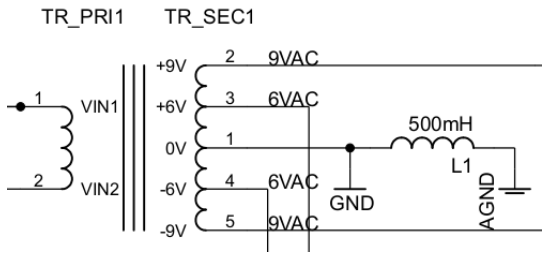


Figura 2.2: Distribución de cables de los conectores molex del transformador. a) devanado secundario, b) devanado primario.

construido a partir del regulador lineal (L7905CV) que "reduce"la tensión de -12 V_{DC} a -5 V_{DC}. Esta última fuente se implementó de la forma que se indica, debido a que la única carga que soporta es el DAC y la máxima corriente que éste demanda es 250μA[12]. Como complemento de la fuente, se ha anexado a cada circuito integrado del equipo, un condensador de 0,1 μF en la vecindad de los pines de alimentación para filtrar armónicos de alta frecuencia que puedan alterar el buen funcionamiento de los componentes. Dado que los circuitos electrónicos conectados a la fuente de potencia son tanto de tipo analógico como digital, y los circuitos digitales generan señales armónicas de alta frecuencia que pueden interferir con el buen funcionamiento de los circuitos analógicos, se ha hecho la distinción entre las tierras de estos circuitos conectando la inductancia L_1 al tab central del transformador como se muestra en la figura 2.2; dado que la impedancia de un inductor ideal responde en el dominio de la frecuencia al modelo $Z_L = j\omega L$[14], los armónicos que se realimenten desde la tierra digital hasta la tierra analógica deben pasar por una impedancia proporcional a su frecuencia, consiguiendo así que los armónicos responsables del ruido electrónico se atenúen.

La tarjeta electrónica se ha provisto además con el conector EXT_POWER (Figura 2.1c) que permite expandir los niveles de tensión DC generados por

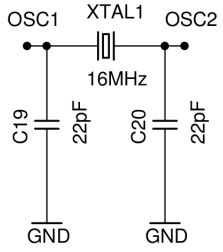


Figura 2.3: Fuente de reloj del sistema de sistema de procesamiento. El cristal XTAL1 es de tipo piezoeléctrico.

la fuente de potencia, a otros equipos. En la figura 5.2 se enseña el circuito esquemático de la fuente de potencia.

2.1.2. Sistema de procesamiento

El núcleo principal del generador de forma de onda arbitraria (refiérase a la página 2 del anexo 5.2) es el microcontrolador PIC18F4550[13], en el cual se procesan eventos de interrupción generados por agotamiento de tiempo, por peticiones del ordenador host o por solicitud del usuario.
El microcontrolador se ha conectado para que opere con un cristal de cuarzo de 16 MHz, que se ha referido a tierra a través de un capacitor cerámico de 22 pF conectado a cada uno de sus pines de conformidad con las indicaciones del fabricante en la hoja de especificaciones[13](Figura 2.3).
La estructura de puertos de entrada y salida digitales del microcontrolador, se han organizado para que el puerto *D* opere como bus bidireccional de datos de 8 bits, el cual es compartido entre todos los dispositivos que lo requieren. Para evitar la competencia de dispositivos por uso simultáneo del mismo recurso, se controla a través del puerto *B* el pin de habilitación de cada chip. A este puerto se han cableado además las teclas *Up* y *Down* que generan interrupciones de tipo pin change en el programa del microcontrolador.
El puerto *E* se ha dedicado exclusivamente para el control de las líneas *enable, RS* y *R/W* de un display de cristal líquido de dos líneas y 16 caracteres.
El puerto A se ha designado como puerto de señales misceláneas; en el bit 0 (RA0) se atiende la señal *Start/Stop* generada por el usuario por pulsación de la tecla "Enter" sobre el panel principal del equipo, mientras que en los bits 4 (RA4) y 5 (RA5) se generan señales cuadradas que se emplean en la aplicación de analizador multicanal Mössbauer externo.
El puerto *C* se ha configurado para operar como puerto de control; a él se han cableado: los pines de lectura y escritura de la memoria, el pin de verificación de estado de la misma (Ready/Busy) y los pines de control de los latch que

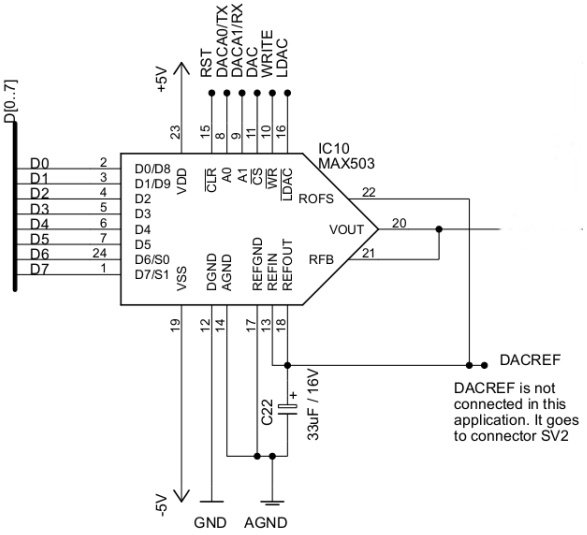


Figura 2.4: Diagrama esquemático del circuito de conversión DAC implementado para el generador de onda arbitraria.

están embebidos en el DAC (Señales DACA0 y DACA1). En la página 2 del anexo 5.2 se enseña el circuito esquemático del sistema de procesamiento del generador de onda arbitraria.

2.1.3. Conversión digital-analógica

Como complemento al sistema de procesamiento, el equipo cuenta con un conversor digital análogo (DAC) que traduce las palabras digitales del microcontrolador, en una diferencia de potencial sobre el conector BNC nombrado *waveform* . Para ello se ha empleado el circuito integrado MAX503CWG, un DAC de 10 bits compatible microprocesador con salida en tensión y referencia interna; ésta característica se refiere a que el DAC cuenta con un conjunto de latch internos que sostienen de forma estable el código digital antes del paso de conversión, disminuyendo así la probabilidad que aparezcan efectos *glitch* en el pin de salida analógica[15]. La salida del DAC se ha configurado en modo bipolar, facultando al DAC la generación de señales entre $\pm 2,048$ V, tensión que se ha dispuesto, para fines de depuración y revisión de la tarjeta, en el pin DACREF del conector header SL1 (ver figura 5.2). La etapa siguiente al conversor DAC amplifica el rango de salida a ± 3 V. La figura 2.4 esquematiza el circuito de conversión digital-analógica implementado. En la figura 5.2 se detallan las conexiones de éste circuito con otras partes del generador de onda arbitraria.

2.1.4. Memoria no volátil paralela

Para desarrollar el algoritmo de síntesis digital directa, se ha cableado una memoria paralela EEPROM -no volátil- al bus de datos del sistema de procesamiento. Esta memoria, de conformidad con la resolución elegida para nuestro DAC, debe ser de 2 kB para obtener el mayor beneficio del conversor, el cual requiere 2 bytes por dato a convertir sobre un total de 1024 datos posibles. El zócalo sobre el que se aloja la memoria y el circuito impreso implementado permite el direccionamiento de un módulo de hasta 32 kB (tipo 28C256), sin embargo, la lógica metodológica de la DDS hace innecesario el uso de una memoria de más de 2^{n+1} bytes cuando se emplea un DAC de n bits (con $8 < n \leq 16$), pues se busca que a cada estado posible de conversión corresponda un dato en memoria, dato que para $8 < n \leq 16$ debe ser de 2 bytes.
Para dar soporte a los 11 bits requeridos por la memoria paralela para su direccionamiento, se han cableado al bus de datos la pareja de latch de lógica transistor-transistor 74HC373 y 74HC374; estos latch sostienen los bytes alto y bajo respectivamente de la posición de memoria que direcciona el microcontrolador. Un flanco ascendente de la señal LAT, cableada al bit 3 del puerto B (RB3), activa el sostenimiento del byte bajo de direcciones, mientras que el estado bajo de esta misma señal activa el sostenimiento del byte alto (ver página 2 del anexo 5.2).

2.1.5. Memoria no volátil serial

Para el almacenamiento de datos diferentes a los empleados en el algoritmo de síntesis digital directa, se ha cableado a los pines del módulo MSSP (Master Synchronous Serial Port) del microcontrolador, la memoria serial no volatil 24LC16 que opera bajo el protocolo I^2C (Figura 2.5). En esta memoria se guardan mensajes como el nombre del último archivo de datos descargado y los textos de las pantallas que se presentan en el display de cristal líquido.

2.1.6. Acondicionamiento de señales de salida

El sistema de conversión digital a análogo, como todo sistema de este tipo, genera un efecto de escalonamiento sobre la señal de salida debido a que los pasos de conversión de datos son discretos. Para retirar este efecto, se ha anexado un filtro Bessel a la salida del DAC, el cual, atenúa los armónicos de frecuencia por encima de cierta frecuencia crítica. Aplicando análisis de corrientes de Kirchoff al nodo V de la figura 2.6a y suponiendo la existencia de

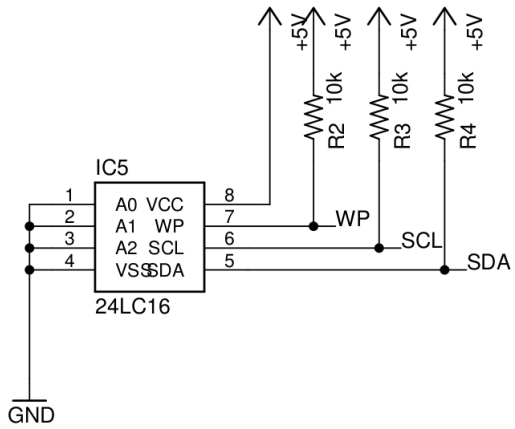


Figura 2.5: Diagrama esquemático de la memoria serial no volatil cableada para el generador de onda arbitraria.

una tierra virtual en la entrada no inversora del amplificador[1], se encuentra que la función de transferencia del filtro[2] es,

$$H(\omega) = \frac{-\frac{R_1}{R_3} + j\omega C \frac{R}{R_3}}{\left(\frac{R}{R_3}\omega C\right)^2 + \left(\frac{R_1}{R_3}\right)^2} \tag{2.1}$$

Donde,

$$R = R_1R_2 + R_2R_3 + R_1R_3 \tag{2.2}$$

El conjunto de componentes pasivos R_1, R_2, R_3 y C dispuestos sobre la tarjeta electrónica, se han seleccionado para generar una frecuencia crítica de 608 Hz; la gráfica de la función de transferencia para los valores de resistencia y capacitancia que se indican en el circuito esquemático 5.2, se enseña en la figura 2.6b.

2.1.7. Comunicaciones

Para realizar la comunicación entre el generador de forma de onda arbitraria y una computadora (*host*) se han implementado dos protocolos de comunicaciones: uno RS-232, desarrollado para comunicaciones con host distantes (más de 5 m pero menos de 20 m), y otro de tipo *Universal Serial Bus* (USB) para host cercanos (menos de 5 m). A pesar que el protocolo RS-232

[1]Ésta suposición se puede hacer debido a que los amplificadores operacionales presentan alta impedancia entre las entradas inversora y no inversora y por tanto, la corriente entre dichos pines tiende a cero.

[2]Ver anexo 5.3

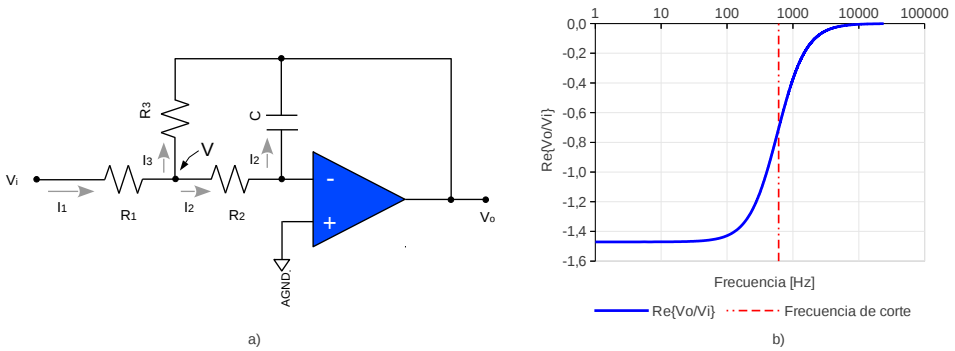


Figura 2.6: a) Circuito esquemático del filtro Bessel, b) gráfica de su función de transferencia.

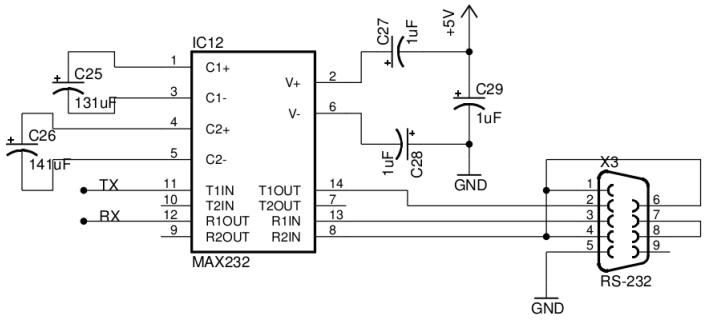


Figura 2.7: Circuito esquemático del puerto de comunicaciones RS232 del generador de forma de onda arbitraria.

puede operar indistintamente para host cercanos y lejanos, se diseñó la tarjeta electrónica con un puerto USB pensando en la compatibilidad con equipos de cómputo recientes, muchos de los cuales no cuentan a la fecha con puerto serie RS-232.

2.1.8. Comunicación con computadora via RS-232

La interfase de comunicaciones RS-232 del generador de forma de onda arbitraria se ha desarrollado con base en el módulo de comunicación sincrónica asincrónica (EUSART) -que se encuentra embebida en el microcontrolador del sistema- y del circuito de acondicionamiento de señales MAX232[16]. Las conexiones corresponden a las sugeridas por el fabricante en la hoja de especificaciones (figura 2.7).

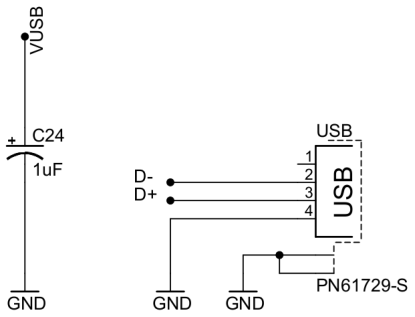


Figura 2.8: Diagrama esquemático de las conexiones del puerto USB implementado en el generador de onda arbitraria.

2.1.9. Comunicación con computadora via USB

La interfase de comunicaciones USB se ha desarrollado a partir del módulo *Universal Serial Bus* que se encuentra embebido en el microcontrolador del sistema. De acuerdo a las directrices estándar para el desarrollo de equipos compatible USB[17], se dispuso sobre la tarjeta electrónica un conector USB tipo B, debido a que el generador de onda arbitraria está en capacidad de operar como equipo esclavo pero no como host, esto es, no ha sido diseñado ni para entregar potencia ni para inicializar otros dispositivos USB. Los detalles técnicos de la configuración, operación y desarrollo del protocolo de comunicaciones USB se desarrollan en el anexo 5.4.
De acuerdo a los requisitos de la especificación USB 2.0, para que el equipo tome potencia desde el puerto USB, es necesario garantizar que la capacitancia neta vista entre el pin *VBus* del microcontrolador y la tierra digital del sistema sea menor a $10\mu F$; esto se ha logrado mediante la conexión de un capacitor de $1\mu F$ entre dichos pines (figura 2.8).

2.1.10. Interfaz hombre-máquina

La capa de hardware a través de la cual el usuario interactúa con el equipo, se ha desarrollado con base en un display de cristal líquido en el que se presenta información acerca de la forma de onda generada, y tres botones que inciden directamente sobre ella, dos de los cuales incrementan y decrementan la frecuencia de acuerdo al botón pulsado y uno tercero empleado como start/stop (figura 2.9). Esta última señal se puede generar alternativamente a través de la aplicación de ordenador que se describe en 2.2.12.
Las teclas para el ajuste del valor de la frecuencia se han cableado a los pines RB7 y RB6, que activan una interrupción cuando se ocurre un cambio en su estado; debido a que ambas teclas generan la misma interrupción, se

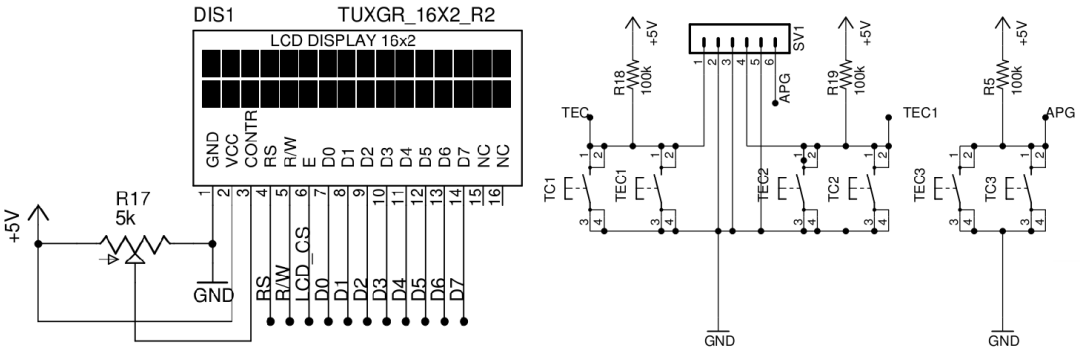


Figura 2.9: Diagrama esquemático de la interfaz hombre-máquina implementada en el generador de onda arbitraria. A la izquierda, conexiones del display de cristal líquido; a la derecha, circuito implementado para las teclas.

ejecuta un algoritmo en el programa principal que detecta cual de ellas se ha pulsado (Para más detalle, remítirse a 2.2.10). La tecla APG (Atención Parada Global) se ha cableado al pin RA0; para revisar sí esta tecla ha sido pulsada, se monitorea regularmente su estado en la rutina principal.

2.1.11. Estructura de soporte

Para soportar la tarjeta electrónica y los componentes eléctricos que la complementan, se construyó una estrutura metálica en lámina cold roll, cuya forma se ha obtenido de procesos de doblado, perforado y soldadura de punto. El transformador se ha anclado a la cara inferior de la estructura mediante tornillos. El circuito impreso descansa sobre unos espaciadores de 2"que generan el espacio suficiente para que el transformador no entre en contacto eléctrico ni con la tarjeta de circuito impreso ni con los componentes electrónicos soldados a ella. Un conjunto de espaciadores extra de 1/2"de longitud, se atornillan sobre los espaciadores de 2"para inmovilizar la tarjeta electrónica y para generar los puntos de descanso de la cubierta principal del equipo (figura 2.10).
Toda la estructura se fabricó en acero para evitar que ondas electromagnéticas generadas por fuentes externas interfieran con la forma de onda producida, al tiempo que reduce la probabilidad que el equipo actúe como fuente de ruido electromagnético. En el anexo 5.2 se enseñan los planos de la estructura de soporte.

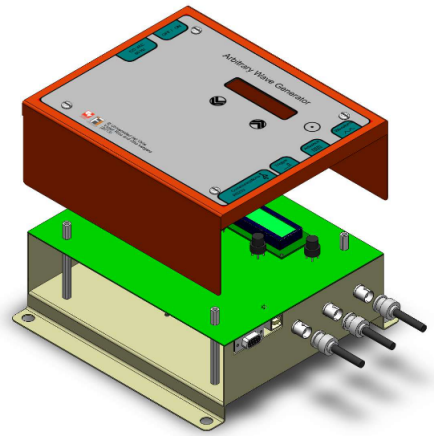


Figura 2.10: Estructura de soporte de la tarjeta electrónica. El conjunto de espaciadores anclan la tarjeta electrónica a la estructura y sirven como punto de apoyo de la tapa.

2.1.12. Conectores de salida

El equipo se ha dotado con tres conectores BNC incrustados en la estructura de soporte. Uno de los conectores opera como terminal de salida de la forma de onda (conector *waveform*) mientras que los otros dos están dedicados a operar como terminales de salida de un par de señales cuadradas de uso exclusivo para el analizador multicanal Mössbauer de la Universidad del Valle[5] (conectores *Trigger* y *Advance*). En el conector Trigger se genera un pulso de 5 V de amplitud cada vez que inicia un nuevo periodo de la forma de onda; en el conector Advance se genera una señal cuadrada de 5 V de amplitud y N pulsos periodo con un ciclo de trabajo de 50 % (figura 2.11); el número N se selecciona a través de un menú en el display del dispositivo y tiene la posibilidad de ser 512 ó 1024.

2.2. Algoritmos

El equipo tiene dos instancias de programación: una para el microcontrolador que se ha dispuesto sobre la tarjeta electrónica (apartados 2.2.1 a 2.2.11) y otra para una computadora personal (apartado 2.2.12) que complementa la idea de generador de onda arbitraria propuesta. La programación del microcontrolador PIC18F4550[13] se ha desarrollado fundamentalmente en lenguaje C; algunas líneas de código se escribieron en lenguaje ensamblador para optimizar los tiempos de procesamiento de rutinas clave para la operación del equipo. Como herramienta de compilación se ha empleado el software CCS C Compiler[20].

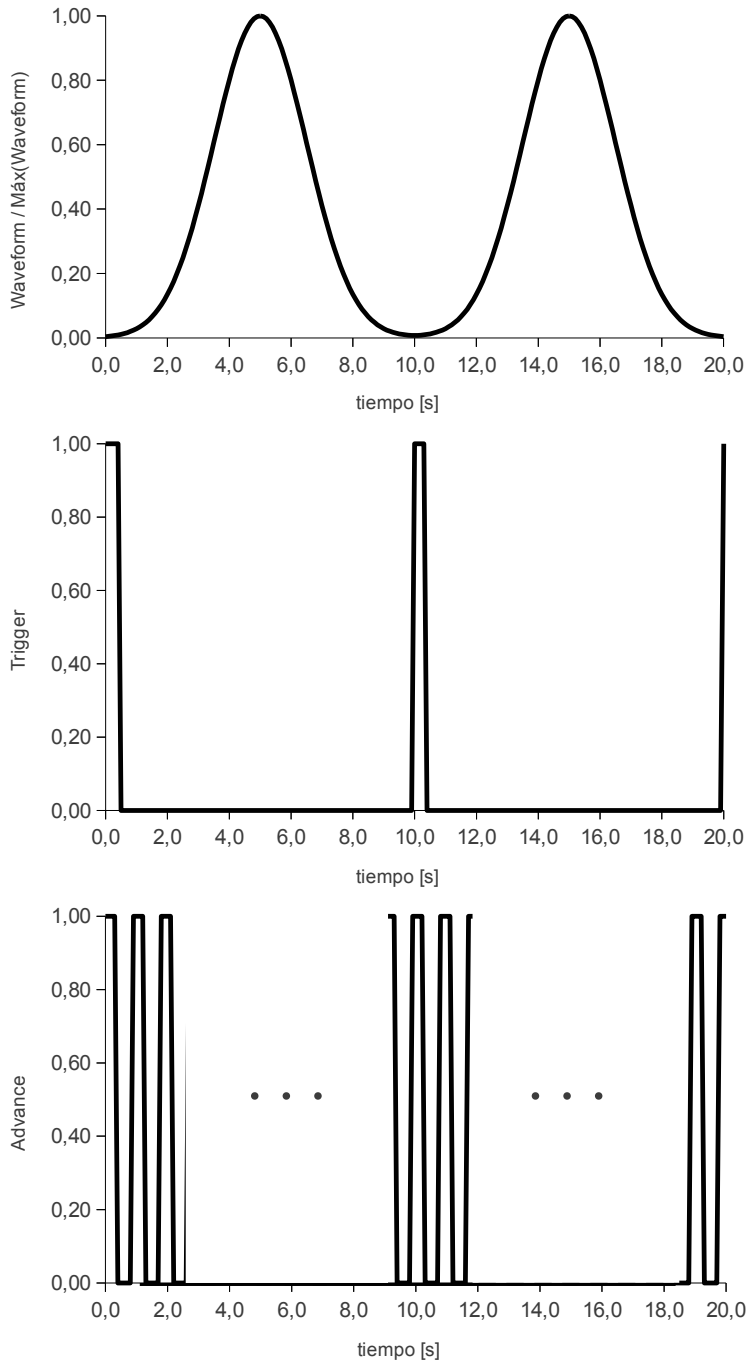


Figura 2.11: Señales de salida en los conectores BNC del generador de forma de onda arbitraria. Por cada periodo de la forma de onda generada (imagen superior), se producen un pulso de la señal trigger (imagen al medio) y n periodos de una señal de reloj con ciclo de trabajo de aproximadamente 50 % (imagen inferior).

2.2.1. Sentencias Preliminares y Rutina de Inicialización

Como se indicó en 2.1.2, el microcontrolador se ha cableado a una fuente de reloj piezoeléctrica externa de 16 MHz; de acuerdo al diagrama reloj del PIC18F4550, haciendo uso de los fuses HSPLL y CPUDIV1 (Figura 5.4), el dispositivo opera a una frecuencia de oscilador (f_{osc}) de 48 MHz. Dado que la frecuencia real de operación del microcontrolador es tan solo una cuarta parte de este valor[13], el ciclo de máquina -es decir el tiempo que consume el microcontrolador en ejecutar una instrucción- es:

$$T_{cm} = \frac{4}{f_{osc}} = \frac{4}{48 \text{ MHz}} \approx 83,3 \text{ ns} \tag{2.3}$$

Para inicializar variables y ejecutar métodos que se corren una sola vez (como por ejemplo la configuración del display de cristal líquido y el USB), se han escrito las líneas 1 a 103 del programa *FTEDDS1305V1.c* que se enseña en el apartado 5.5.2.
Las variables encabezadas con la directiva de compilación `#BYTE` corresponden a definiciones de los puertos del microcontrolador para su uso en lenguaje ensamblador; el número al que se han inicializado los puertos (por ejemplo `#BYTE PORT_A=0x0F80`) corresponde a la dirección física del puerto en el mapa de registros de propósito específico que indica el fabricante en el manual de usuario[3].
En la rutina *inicialization* se define la dirección inicial de los puertos de entrada/salida a través de las sentencias `TRISp=0bddddddddd`, donde p indica el nombre del puerto (p={A,B,C,D,E}) y d la dirección del bit respectivo ($0 \rightarrow$ salida, $1 \rightarrow$ entrada); en la misma rutina también se fijan los valores iniciales de los mensajes que se presentan en el display del equipo, de carga del temporizador y se habilitan las interrupciones de atención por cambio de estado de los pines RB4..7 y desbordamiento del timer 0.

2.2.2. Rutina principal: lazo ejecutivo

La rutina principal del software del microcontrolador está dedicada a vigilar el estado de cada uno de los bits de un registro de tareas (taskRegister) creado exclusivamente para ese fin. Cada bit del registro de tareas tiene asociada una subrutina que se ejecuta cuando dicho bit cambia a estado activo alto; estas tareas se presentan en la figura 2.12. La única forma de activación de esos bits, es a través de un evento de interrupción que puede provenir

[3]Table 5-1 del datasheet PIC18F4550[13].

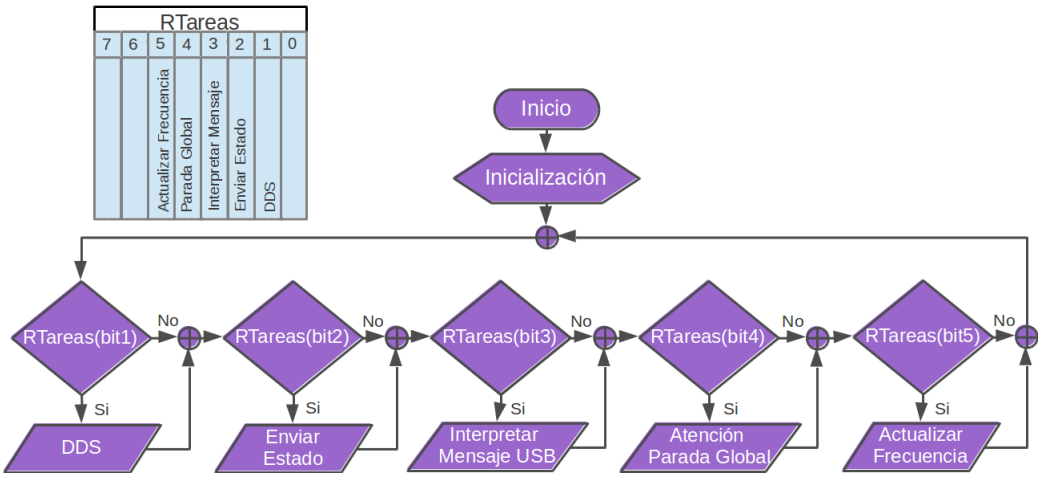


Figura 2.12: Registro de tareas y lazo ejecutivo del programa del microcontrolador

del levantamiento de la bandera de dato válido en el buffer USB, del desbordamiento del timer 0, ó por la pulsación de una de las teclas del equipo. Cada subrutina incluye en su última línea de código, una sentencia que borra el bit del registro de tareas responsable de su ejecución. Dicha activación se consigue mediante la sentencia,

```
bit_set(taskRegister,b);
```

La limpieza de un bit del registro de tareas se consigue mediante la sentencia,

```
bit_clear(taskRegister,b);
```

El parámetro *b* indica en ambos casos el bit del registro de tareas al que se le ha de fijar el estado.
El algoritmo que describe la rutina principal y el lazo ejecutivo se enseña en la figura 2.12. Esta estrategia metológica ha sido empleada por la robustez que exhibe para el manejo de tiempos en algorimos largos. La razón principal de haber desarrollado un algoritmo así y no de otra forma, es que con esta metodología el microcontrolador pasa la mayor parte del tiempo rotando en torno a los bits del registro de tareas, lo que resulta bastante útil para hacer síntesis digital directa como se verá más adelante.
Los siguientes apartados describen las rutinas que se ejecutan tras la activación de cada bit del registro de tareas.

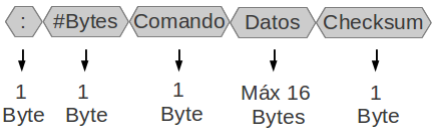


Figura 2.13: Trama del mensaje del protocolo de comunicaciones Intel-Hex implementado. Junto a la cadena de datos a transmitir se anexa un byte de inicio, uno de comando, uno que indica el número de bytes y un byte que revisa la integridad de la información transmitida (checksum).

2.2.3. Rutina de interpretación de mensajes recibidos a través del puerto USB

La transferencia de datos a través del puerto USB se ha implementado de acuerdo a los lineamientos del protocolo de comunicaciones Intel-Hex[21], en el que se enlaza a los datos principales una serie de bytes que informan sobre la cadena transmitida (figura 2.13). La cadena se compone de un byte de inicio (caracter ":"), de un byte que da cuenta de la longitud en bytes que componen el campo de datos (#Bytes), de un byte de comando que se usa para interpretar el(los) dato(s) transmitido(s), de la secuencia de datos a transmitir y de un byte de verificación (checksum) en el que se guarda el byte menos significativo de la suma de los bytes: #Bytes, Comando y Datos; este último dato permite detectar problemas asociados al enlace host-dispositivo. La interpretación del mensaje recibido a través del puerto USB se realiza por comparación del byte de comando; dependiendo del comando recibido, se activa uno u otro bit del registro de tareas o se corre una rutina específica. La figura 2.14 enseña el algoritmo que se ejecuta cuando hay un dato válido en el bufer USB.
En el caso particular de la recepción de una cadena con comando 4, el paquete de datos recibido corresponde a la dirección (dos bytes) del primero de 16 valores de memoria a leer; mediante la ejecución de un ciclo de 16 lecturas consecutivas de la memoria paralela del sistema (ver 2.2.5) se agrupan los bytes de datos en un vector (message) y se envían al host a través de la instrucción en lenguaje c,

```
usb_puts(1,message,16,1);
```

En esta instrucción, el primer parámetro indica el número del endpoint a usar, *message* corresponde a la cadena de datos a transmitir, el tercer parámetro relaciona el número de bytes del vector *message* que efectivamente se transmitirán y el último parámetro indica el tiempo de espera (en ms) entre la transmisión de cada dato.

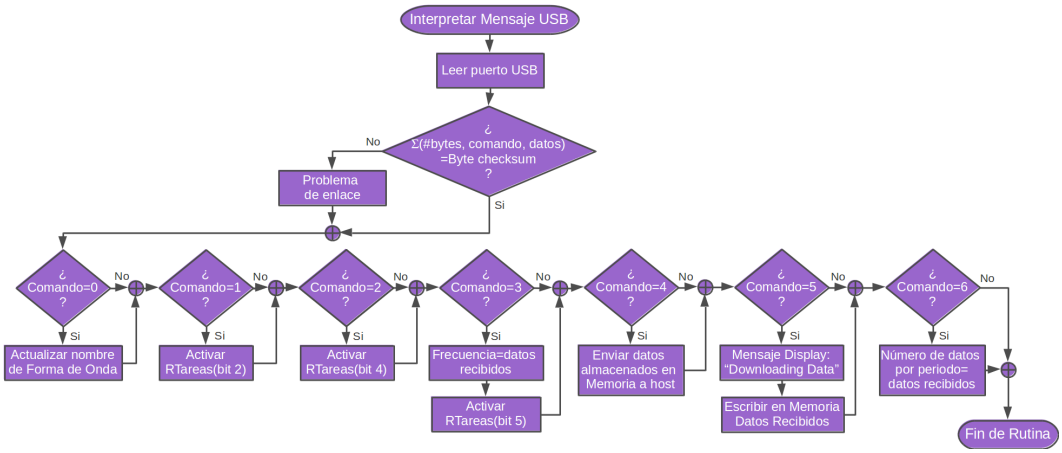
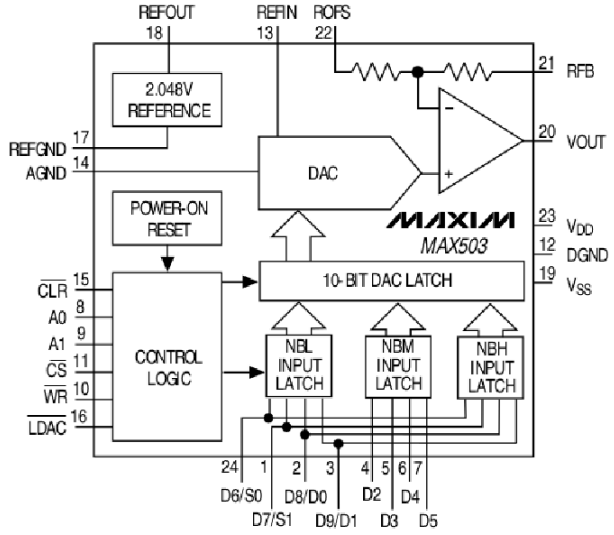


Figura 2.14: Algoritmo de Interpretación de mensajes recibidos a través del puerto USB. En algunos casos, un bit del registro de tareas se activa en función del comando recibido.

2.2.4. Subrutina de Escritura en DAC

En la figura 2.15 se muestra el diagrama funcional del conversor digital análogo MAX503CWG dispuesto en la tarjeta electrónica del equipo. Los 10 bits de entrada del DAC, están distribuidos en 3 nibbles (NBL, NBM y NBH) cuyo valor se sostiene de forma estable a través de un conjunto de latch que se controlan con los pines A0 y A1. Fué justamente esta característica (la de poder sostener mediante latch el valor previo al paso de conversión) además de estar configurado en una red R-2R y la ventaja de compatibilidad del DAC con el microprocesador de 8 bits de esta aplicación, el principal criterio para elegir esta referencia y este fabricante de otros que se encuentran en el mercado[22].
Para acoplar los 10 bits de resolución del DAC con los 8 bits del bus de datos, el fabricante ha asignado a algunos de los pines del circuito integrado doble función; éste es el caso de los bits D_6, D_7, D_8 y D_9 que se comparten con los bits S_0, S_1, D_0 y D_1 respectivamente. La función que desempeña cada pin depende del código se selección de latch {A0, A1}, así para el valor A0=A1=0 los bits D_7 y D_6 del bus de datos corresponden a los bits S_1 y S_0 del DAC. Como se indica en la hoja de características técnicas[12], el MAX503 es la versión de 10 bits del conversor digital análogo de 12 bits MAX530; por esta razón se debe movilizar al puerto de entrada del MAX503 un dato de 12 bits, dos de los cuales (S0 y S1), no tienen una función específica asignada y el fabricante recomienda garantizar que su valor sea siempre cero.
La movilización de un dato a los latch del DAC requiere dos tiempos de programación, en uno de los cuales se fija el valor de los nibbles NBL y NBM y en otro el valor del nibble NBH. La tabla 2.1 enseña la relación existente



A0	A1	Latch
0	0	NBL, NBM (S0, S1, D0-D5)
0	1	NBL (S0=0,S1=0, D0, D1)
1	0	NBM (D2-D5)
1	1	NBH (D6-D9)

Figura 2.15: Diagrama funcional del conversor digital análogo MAX503CWG. Fuente: Maxim[12].

entre los bits del bus de datos y los de la interfaz digital del DAC; para fijar el valor de los nibbles NBL (D_1, D_0, S_1, S_0) y NBM (D_5, D_4, D_3, D_2) se asigna el estado activo bajo a los pines A0 y A1 y se mueve al bus de datos el valor que corresponde a cada uno de estos bits, esto es: S_1=S_0=0 y D_5, D_4, D_3, D_2, D_1, D_0 con el valor que corresponda. Para fijar el valor del nibble NBH (D_9, D_8, D_7, D_6) se asigna el estado activo alto a los pines A0 y A1 y se mueve al bus de datos el valor correspondiente.

Tabla 2.1: . Relación de bits Bus de Datos / DAC

	Bits							
Bus de Datos	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
DAC	S_1	S_0	D_5	D_4	D_3	D_2	D_1	D_0
DAC	D_7	D_6					D_9	D_8

Debido a que los bits S_1 y S_0 están ubicados en medio de la palabra a escribir en el DAC, es necesario enmascarar el byte de cada movimiento de datos para garantizar que los bits se asignen de acuerdo a lo esperado. Sí el dato a escribir sobre el DAC es $\texttt{0b}D_9D_8D_7D_6D_5D_4D_3D_2D_1D_0$, una operación AND de su byte bajo con el número `0b00111111` arroja el valor $\texttt{0b00}D_5D_4D_3D_2D_1D_0$; al mover este resultado al bus de datos y fijar en estado activo bajo los bits A0 y A1, se fija el valor de los nibbles NBM y NBL. De forma similar, la suma del byte alto del dato a escribir con el byte bajo enmascarado con `0b11000000` ($\texttt{0b}D_7D_6D_5D_4D_3D_2D_1D_0$ `&` `0b11000000`), genera el dato que se debe mover al

bus de datos para fijar el valor del nibble NBH.
Todas estas operaciones se hacen de acuerdo a la secuencia y tiempos mínimos entre instrucciones que define el fabricante del conversor digital análogo en la hoja de especificaciones técnicas; estos tiempos se muestran en la figura 2.16. La figura 2.17 resume en un algoritmo la secuencia de pasos antes descrita y constituye la base de la estructura de las líneas de código programadas para la operación del DAC (rutina `writeDAC()` del programa *FTEDDS1305V1.c*).

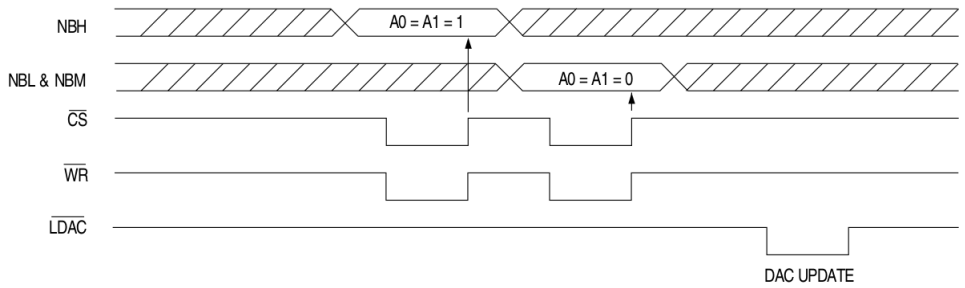


Figura 2.16: Diagrama de tiempos de escritura del conversor digital análogo MAX503CWG. Fuente: Maxim[12].

2.2.5. Subrutina de lectura de memoria paralela

La lectura de los datos almacenados en la memoria paralela no volatil (E^2PROM) se consigue a través del protocolo de señales digitales que indica el fabricante en la hoja de especificaciones técnicas[23]. Esa secuencia se reproduce con fines documentativos en la figura 2.18. Para leer uno de los 2048 bytes disponibles en la memoria, es necesario fijar primero su dirección a través de los latches IC7 e IC8; esto se consigue mediante la secuencia: byte bajo de direcciones a bus de datos → sostener → byte alto de direcciones a bus de datos → sostener. Una vez la dirección está sostenida se generan secuencialmente las señales digitales de chip enable ($\overline{\texttt{CE}}$) y output enable ($\overline{\texttt{OE}}$) al tiempo que se sostiene en estado activo alto el pin correspondiente a la habilitación de escritura ($\overline{\texttt{WR}}$). En principio, la caracterización hecha por el fabricante para este modelo de memoria indica que la lectura de un dato ($t_{\texttt{RC}}$) no tarda menos de 200 ns, siendo así el ciclo de máquina de 83,3 ns (ec. 2.3) apenas apropiado para realizar esta tarea (figura 2.20a).

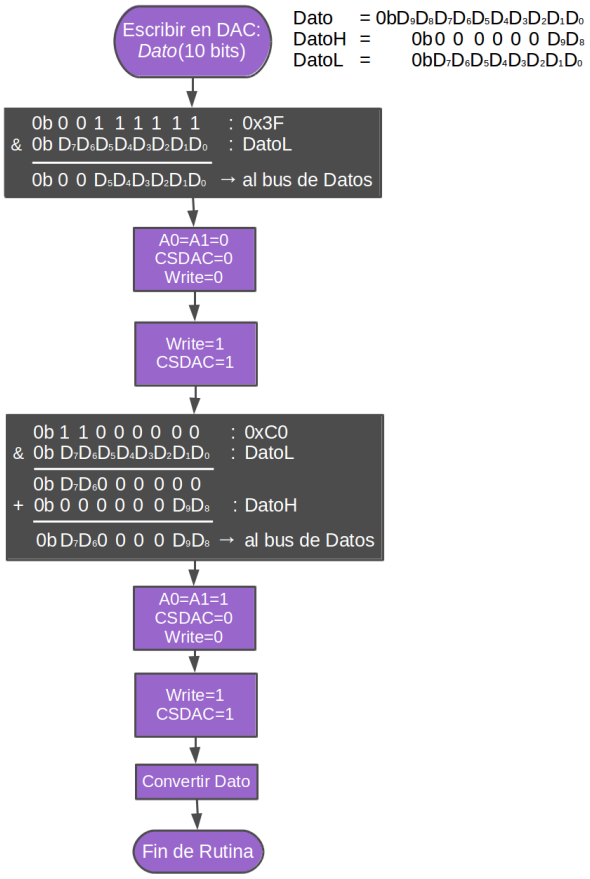


Figura 2.17: Algoritmo de escritura de datos en el DAC.

2.2.6. Subrutina de escritura de memoria paralela

Al igual que la rutina de lectura de memoria paralela, la escritura de datos en la E^2prom se realiza de acuerdo al protocolo que indica el fabricante en la hoja de especificaciones técnicas [23]. Este tipo de memoria cuenta con dos métodos de escritura: uno sosteniendo la señal $\overline{WE}$ ($\overline{WR}$ en el diagrama de la página 2 del anexo 5.2) y manipulando el estado del pin $\overline{CE}$ y otro que opera de forma inversa, sosteniendo $\overline{CE}$ y manipulando $\overline{WE}$, siendo este último el método por que se optó. La figura 2.19 enseña el diagrama de tiempos para la escritura de un dato.
Al igual que en la subrutina de lectura (2.2.5), primero se debe fijar la dirección del segmento de memoria a escribir (haciendo uso de los latches IC7 e IC8) luego se mueve el valor a escribir al bus de datos y se cambia a estado activo alto de forma consecutiva los pines correspondientes a $\overline{CE}$, $\overline{OE}$ y $\overline{WE}$. A diferencia de la lectura, el tiempo del ciclo de escritura es significativamente

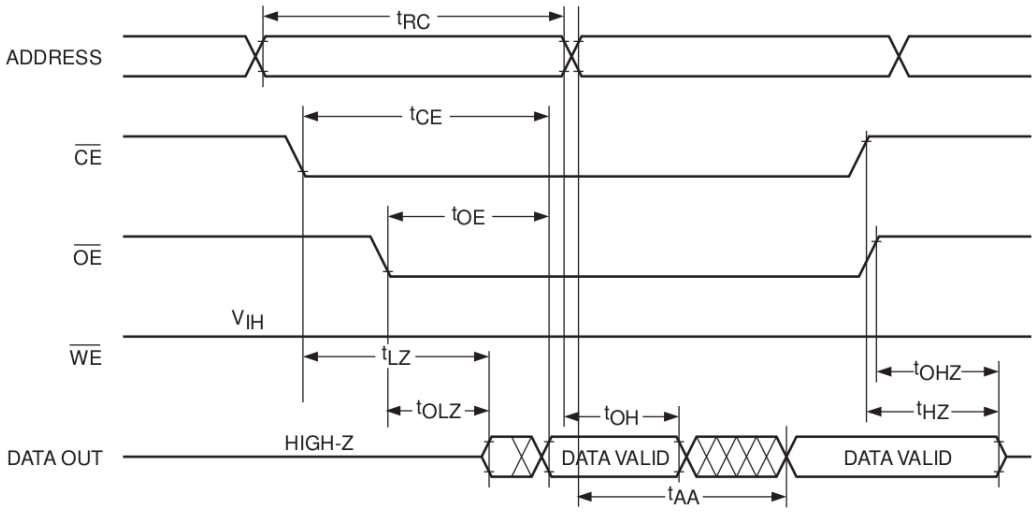


Figura 2.18: Ciclo de lectura de la memoria EEprom CAT28C17AP. Fuente: Catalyst[23].

largo (máx 10 ms) comparado con el tiempo que le toma al microcontrolador ejecutar una instrucción (ecuación 2.3). En el caso particular de la memoria CAT28C17AP, se cuenta con el pin Ready/Busy (`RDY/BSY`) que informa sí la memoria está o no ocupada escribiendo un dato. Para el uso de memorias tipo AT28C64 o 28C256 es necesario ejecutar un ciclo de retardo que garantice que el tiempo mínimo de escritura se de, pues este tipo de memorias no cuentan con el pin `RDY/BSY`.
Debido a que los tiempos son excesivamente largos, cuando se escribe un dato se visualiza sobre el display el mensaje `"Downloading Data"` mientras la escritura está en proceso. Esta rutina se ejecuta únicamente por petición de la aplicación de ordenador a través de la rutina de interpretación de mensajes enviados por el USB. La figura 2.20b resume mediante un algoritmo la secuencia de pasos antes descrita.

2.2.7. Subrutina de síntesis digital directa (DDS)

Esta es la rutina responsable de la generación de formas de onda en el conector *waveform* de la estructura de soporte y es en ella donde se realiza el algoritmo de síntesis digital directa que titula este trabajo. La rutina hace uso de las subrutinas de lectura de memoria paralela y escritura del DAC. Llegado a este punto es recomendable referirse al capítulo 2 para comprender la estructura lógica de los algoritmos implementados. Previo al desarrollo de esta rutina se debe contar con una tabla de datos escrita en la memoria E^2prom coherente con la forma de onda a reproducir.

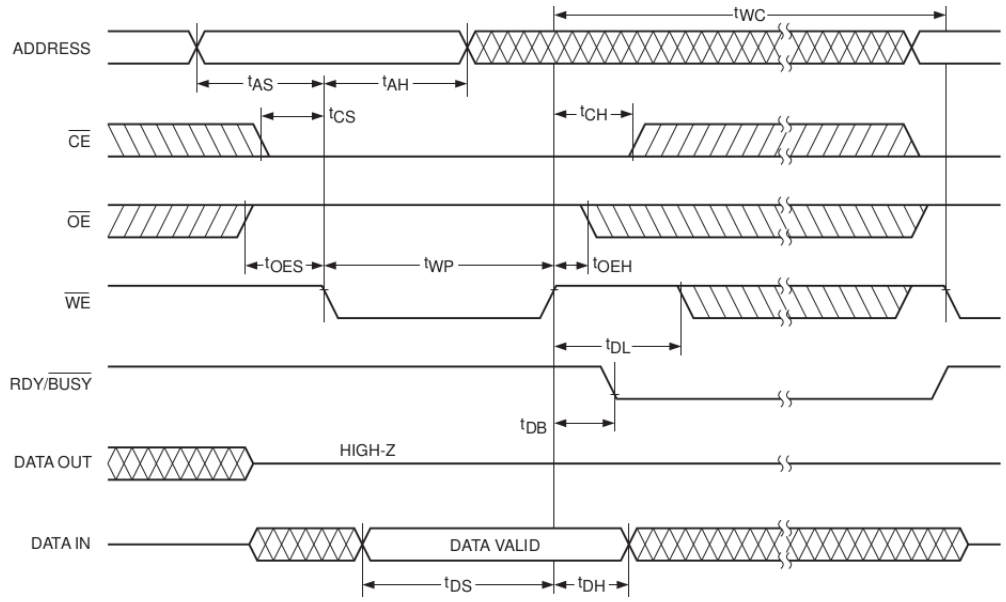


Figura 2.19: Ciclo de escritura de la memoria EEprom CAT28C17AP. Fuente: Catalyst[23].

La rutina se ejecuta cada vez que se produce el desbordamiento del timer0; cuando este evento ocurre, se pone en alto la bandera del registro de tareas que corresponde a esta rutina y en el lazo ejecutivo se hace el llamado para su inicio.
El algoritmo comienza por conmutar el estado del bit 5 del puerto A (RA5, el cual controla la señal Advance); sí el estado de la señal Advance es activo bajo, la rutina entra en el proceso de transferencia de datos desde la memoria al DAC que caracteriza la síntesis digital directa. En caso contrario, la rutina ejecuta tantas instrucciones NOP (no operation) como ciclos de máquina tiene el proceso de transferencia de datos. Esto se hace con el fin de compensar los caminos del condicional y para procurar que la señal Advance presente un ciclo de trabajo de 50 %. En realidad uno de los estados (el alto o el bajo) puede durar más de lo esperado debido al tiempo de espera establecido por el timer; en particular cuando el usuario ajusta un valor de frecuencia bajo (es decir, fija indirectamente el desbordamiento del timer0 a un tiempo largo), el ciclo de trabajo de la señal Advance puede reducirse aproximadamente hasta el 6 %.
Para el proceso de transferencia de datos desde la memoria al DAC, primero se leen las posiciones consecutivas de memoria que contienen el byte alto y el byte bajo del número a escribir y luego se escribe el dato (de 10 bits) en el conversor digital análogo. Por cada ciclo realizado, se incrementa en una unidad el puntero de memoria de modo que el próximo ciclo apunte a la

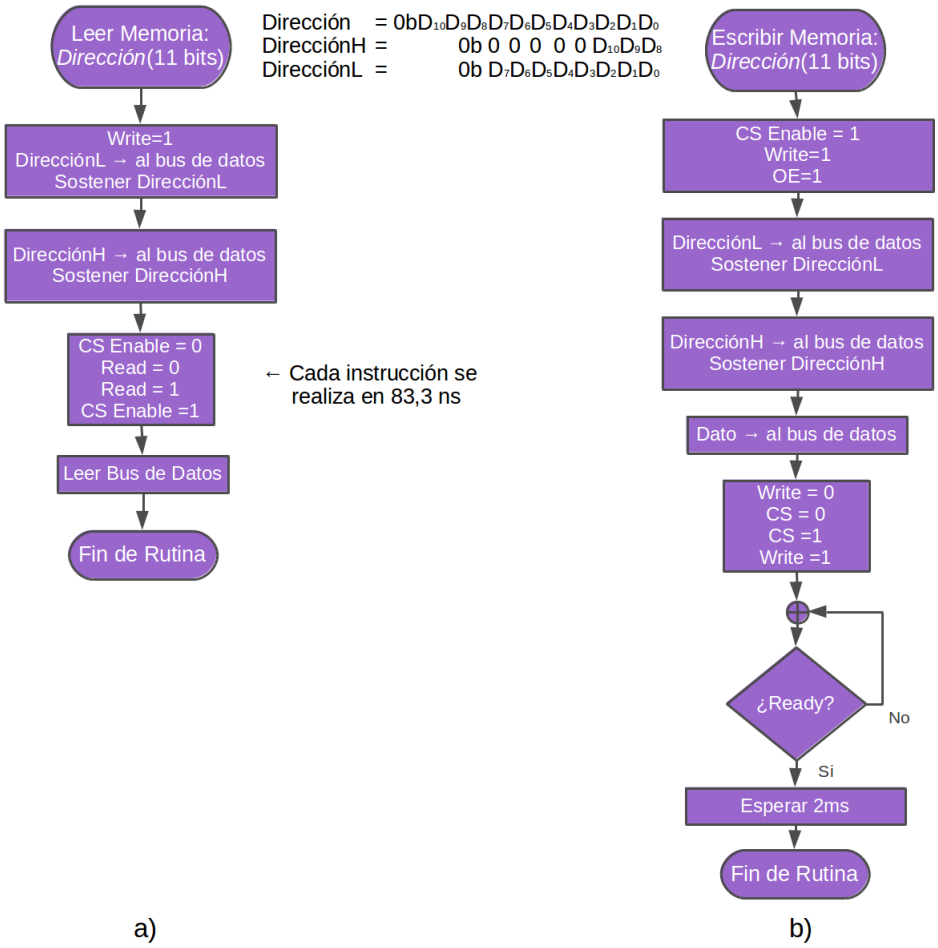


Figura 2.20: Algoritmos de lectura (a) y escritura (b) de la memoria EEprom CAT28C17AP.

posición de memoria siguiente. El puntero inicia siempre con el valor cero y termina apuntando a la posición de memoria $2n$, donde n relaciona el número de puntos con el que se ha construido la forma de onda. El flujograma que resume los pasos antes descritos se enseña en la figura 2.21.

2.2.8. Rutina de verificación del estado del botón APG

Una de las tareas que atiende el microcontrolador con tanta prioridad como la de ejecutar el algoritmo de síntesis digital directa, es la de revisar el estado de la señal de atención parada global, la cual opera para efecto práctico como un botón de parada de emergencia. Esta señal se produce por pulsación de la tecla `enter` sobre el panel principal del equipo o del botón `Start/Stop` en la aplicación de computadora. Cuando uno de estos eventos

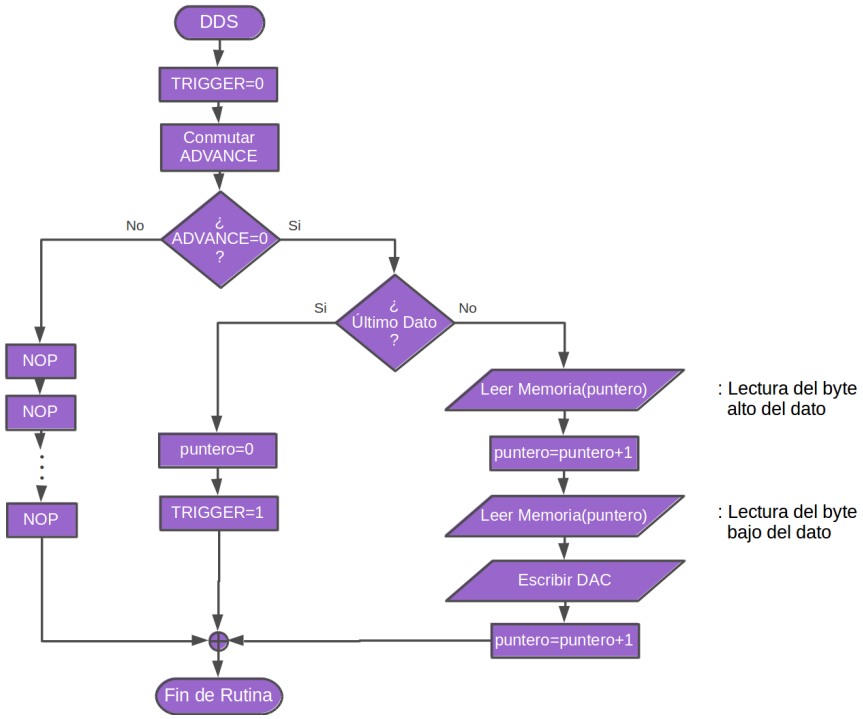


Figura 2.21: Algoritmo de la rutina de Síntesis Digital Directa (DDS).

ocurre conmuta el estado de la bandera `APG`, una variable de un bit definida únicamente para este objetivo. Sí la bandera APG está baja se deshabilita la interrupción por desbordamiento del timer0, en caso contrario se habilita. Además se escribe sobre el display el mensaje `Start?` o `*` según el estado de la bandera (bajo o alto respectivamente).
La figura 2.22 enseña el algoritmo que modela el funcionamiento de esta rutina.

2.2.9. Rutina de interrupción por desbordamiento del timer0

El programa hace uso del timer0 del microcontrolador para habilitar la tarea de ejecución de la rutina de síntesis digital directa (apartado 2.2.7). Independientemente de la tarea que se esté procesando, el programa responde a la petición de atención a la interrupción por desbordamiento de este timer la cual pone en alto el bit 1 del registro de tareas (ver figura 2.12) y recarga el registro del timer0 a su valor inicial. Este timer, el cual es un contador que se incrementa cada vez que se ejecuta un determinado número de ciclos de máquina, pone en alto su bandera de desbordamiento cuando el valor de

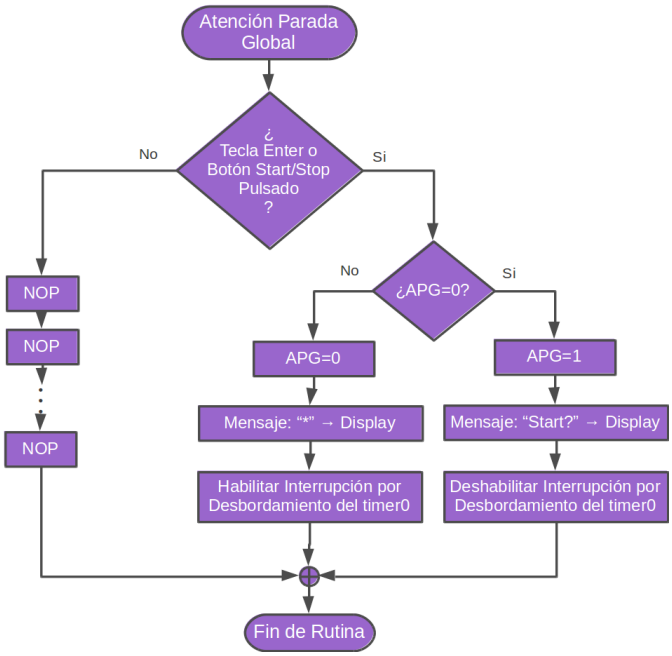


Figura 2.22: Algoritmo de la rutina de Atención Parada Global (APG).

su contador sobrepasa el valor máximo permitido; el tiempo que tarda en ocurrir un evento de desbordamiento está dado por la ecuación,

$$t_{desb} = T_{cm} P_s (C_{max} - \texttt{TMR0}) \tag{2.4}$$

donde P_s indica el número de ciclos de máquina que ocasionan un incremento del registro `TMR0`, C_{max} es una constante que determina la cuenta máxima a la que puede llegar el contador del timer0, y T_{cm} es el tiempo que tarda en evolucionar un ciclo de máquina (ver ecuación 2.3).

2.2.10. Rutina de interrupción por cambio de estado de uno de los pines RB7..4

De forma similar al desbordamiento del timer0, el programa también responde a la petición de atención a interrupciónes por cambio de estado de algunos de los pines de entrada del puerto b, específicamente de los bit 6 y 7. Esto ocurre únicamente cuando se encuentra pulsada una de las teclas de incremento o decremento de la frecuencia sobre el panel principal del equipo y generan, como es de esperarse, un cambio en la frecuencia de la señales que se observan en los conectores de salida.
Cuando es pulsada una de estas teclas, el puntero de direcciones de programa

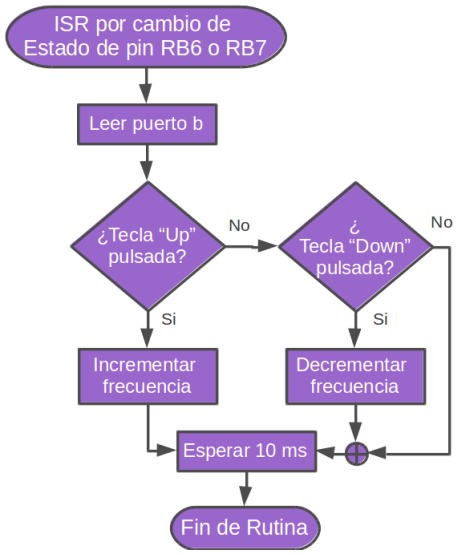


Figura 2.23: Algoritmo de la rutina de Atención Parada Global (APG).

pasa a la rutina de atención a la interrupción (ISR), determina qué tecla generó la interrupción y en función de ello incrementa o decrementa el valor de la variable `frequency` que controla indirectamente el valor de precarga del timer0. La figura 2.23 enseña el algoritmo que se ejecuta cuando un evento de este tipo ocurre.

2.2.11. Subrutina de actualización de la frecuencia de la forma de onda

Esta rutina se ejecuta por pulsación de las teclas `Up/Down` del panel principal del equipo o por la actualización del estado del *spinner* de frecuencia en la aplicación de computadora (ver 2.2.12); ésto último, se realiza por medio de un mensaje enviado desde el equipo host hacia el generador de onda arbitraria a través del puerto USB, razón por la cual se debe garantizar su enlace. Cuando un evento de este tipo ocurre, se actualiza el valor del registro `frequency` en función del cual se es escribe el valor de la frecuencia correspondiente en el display y se calcula el valor de precarga del timer0 que controla el tiempo de desbordamiento del mismo.
Dado que el usuario ajusta un valor de frecuencia (f) de la forma de onda —equivalente a un periodo $T = 1/f$— y que por cada periodo se generan n pulsos de la señal advance, se sigue el siguiente procedimiento matemático para determinar el valor de precarga del timer:

– un periodo de la señal Advance tiene una duración de $T_{adv} = 1/(nf)$, pero como la interrupción debe ocurrir en los hemiciclos de esta señal, se tiene que el tiempo de desbordamiento del timer debe ser,

$$t_{desb} = \Big(\frac{1}{2}\Big)\frac{1}{nf} \tag{2.5}$$

– de acuerdo con las ecuaciones 2.3 y 2.4 se tiene entonces que,

$$\Big(\frac{1}{2}\Big)\frac{1}{nf} = \frac{4}{f_{osc}}P_s\Big(C_{max} - \texttt{TMR0}\Big) \tag{2.6}$$

– despejando `TMR0`,

$$\texttt{TMR0} = C_{max} - \frac{1}{8nP_s}\Big(\frac{f_{osc}}{f}\Big) \tag{2.7}$$

El término no constante de la ecuación 2.7 permite deducir el rango de valores que puede tomar el registro `TMR0` y por ende la resolución con la que se debe configurar el timer; ya que $f_{osc} = 48$ MHz, que el número de canales (y puntos por periodo) puede ser 512 o 1024, y que el rango de frecuencias requerido es de 1 a 30 Hertz —de acuerdo a las especificaciones del espectrómetro Mössbauer[5]—, el registro `TMR0` podrá tomar valores comprendidos entre,

$$\Big(\frac{48\times10^6}{8\times1\times1024\times30} \approx 195\Big) > \texttt{TMR0} > \Big(\frac{48\times10^6}{8\times1\times512\times1} \approx 11719\Big) \tag{2.8}$$

De este intervalo se deduce que es necesario configurar el timer en modo de 16 bits y por tanto, el valor de la cuenta máxima (C_{max}) será 65535. En el cálculo de la ecuación anterior se ha fijado el prescaler en 1 buscando que el timer tenga la mayor resolución posible, pues el incremento del contador se produce cada P_s ciclos de máquina.

2.2.12. Aplicación de computadora

El punto de contacto del usuario con los datos de la forma de onda a reproducir, se lleva a cabo en una aplicación para computadora personal escrita específicamente para complementar la programación de la tarjeta electrónica antes descrita. La aplicación se desarrolló en lenguaje Java[24] bajo la plataforma de programación NetBeans[25] en su distribución 7.3.1. La licencia de uso se ha concebido como de software libre estando cualquier usuario

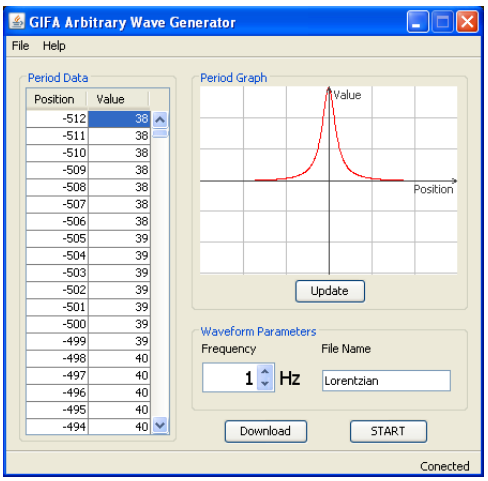


Figura 2.24: Pantallazo de la aplicación de computadora desarrollada como complemento a la programación de la tarjeta electrónica del generador de onda arbitraria.

del programa en libertad de copiar y modificar su contenido. La elección de este lenguaje sobre otros disponibles, obedece a dos razones: por una parte por la compatibilidad de la interfaz gráfica con varios sistemas operativos (Linux, Solaris, Machintosh y Windows) y por otra parte porque las rutinas para el uso del USB ya han sido escritas por varias personas en la internet, que desinteresadamente las han compartido y a quienes los autores extienden su sincero agradecimiento. El único requisito es tener la máquina virtual de Java instalada en la computadora donde se ejecute, herramienta de software que se distribuye Oracle[24] en la red de forma gratuita.

La aplicación de computadora, como es característico de los programas escritos en Java, está orientada a objetos y responde esencialmente a eventos tipo clic sobre los componentes dispuestos en la ventana principal, la cual contiene una tabla de datos editable por el usuario, un gráfico que previsualiza la forma de onda, una barra de menú desde la que se llaman ventanas emergentes y unos controles que modifican el valor de algunas variables de programación del generador de onda arbitraria, siempre que los equipos esten enlazados a través del puerto USB. La figura 2.24 enseña un pantallazo de aplicación de computadora ejecutándose bajo el sistema operativo Microsoft Windows XP.

La función principal de la aplicación es descargar a través del puerto USB los n valores de las ordenadas que reproducen la gráfica de la forma de onda a generar; los valores de las abscisas se desprecian debido a que cada ordenada está asociada a una posición de la memoria no volatil instalada en la tarjeta electrónica y a que el valor de las abcisas queda indirectamente determina-

do en función de la frecuencia ajustada por el usuario. En ningún momento —salvo para graficar la curva de la función Lorentziana que se muestra en cuanto se inicializa la ventana principal— la aplicación se dedica a calcular relaciones abcisas-ordenadas, razón por la cual es necesario procesar los datos en una aplicación externa[4] o ingresarlos manualmente. Éste proceso se consigue haciendo clic en el menú *File/Import* a partir del cual emerge una ventana auxiliar que solicita un archivo de datos separados por comas (extensión *.csv*) que debe haber sido preparado con las siguientes especificaciones:

• La primera línea debe contener la cadena de códigos ASCII:
`Arbitrary Wave Generator File`.
• La segunda línea debe indicar el número de puntos (n) de la forma de onda a reproducir.
• Las n líneas restantes deben contener números enteros comprendidos entre 0 y 1023 (ambos extremos incluidos) correspondientes a los valores de las ordenadas de la forma de onda.

Dado que el conversor digital análogo que se encuentra instalado en la tarjeta electrónica ha sido configurado en modo bipolar[5], al número cero (0) le corresponderá el valor mínimo de la excursión de señal mientras que al número mil veintitrés (1023) le corresponderá el máximo. Dicha excursión se ha calculado mediante los componentes pasivos del filtro Bessel (figura 2.6a) en ± 3 V [6]. Un ejemplo ilustrativo del contenido de un archivo con el cual se podría[7] reproducir una forma de onda diente de sierra de 10 puntos por periodo, se muestra a continuación:

```
Arbitrary Wave Generator File
10
0
114
227
341
455
```

[4]Una hoja de cálculo, por ejemplo, constituye una buena opción.
[5]Refiérase al apartado 2.1.3 para ampliar esta información.
[6]Refiérase a la parte 2 del anexo 5.3 para el detalle del cálculo.
[7]El calificativo condicional del verbo *poder* es debido a que se debe garantizar que el número de datos y el valor de frecuencia ajustado sean suficientes para sostener la suavidad de la forma de onda; en caso contrario el resultado puede ser similar al que se presenta en la figura 3.1.

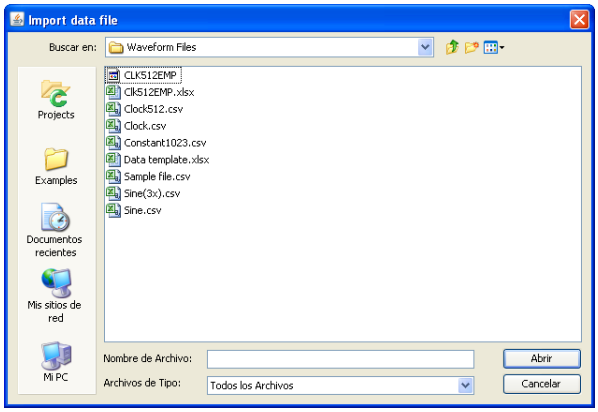


Figura 2.25: Ventana emergente para cargar archivos `.csv` a la aplicación de computadora.

```
568
682
796
909
1023
```

Una vez cargado el archivo y después de oprimir el botón de confirmación, la aplicación escribe en la tabla de la ventana principal los datos y se debe poder previsualizar su gráfica. La figura 2.25 documenta la ventana emergente mencionada.

Mediante una ventana emergente similar a la de la figura 2.25 pero a través de la acción *File/Export*, se puede además exportar en un archivo *.csv* los valores cargados en la tabla de datos, lo que resulta útil cuando el usuario edita directamente la tabla de valores. La aplicación se ha dotado además con un algoritmo que permite "subir"los datos de la memoria E^2prom de la tarjeta electrónica al equipo host en un archivo *.csv* a través de la selección del menú *File/Upload*; éste archivo sigue los mismos lineamientos antes descritos y puede ser utilizado para evaluar, por ejemplo, el funcionamiento de la memoria o para hacer una copia de seguridad de los datos allí almacenados. En este punto la aplicación envia un mensaje al microcontrolador a través del puerto USB, que incluye: un código de petición de datos y los bytes bajo y alto de la primera dirección de memoria a leer; el programa del microcontrolador levanta entonces una bandera para que se ejecute esa tarea y cuando se realiza, envia en una sola trama los datos que se encuentran almacenados

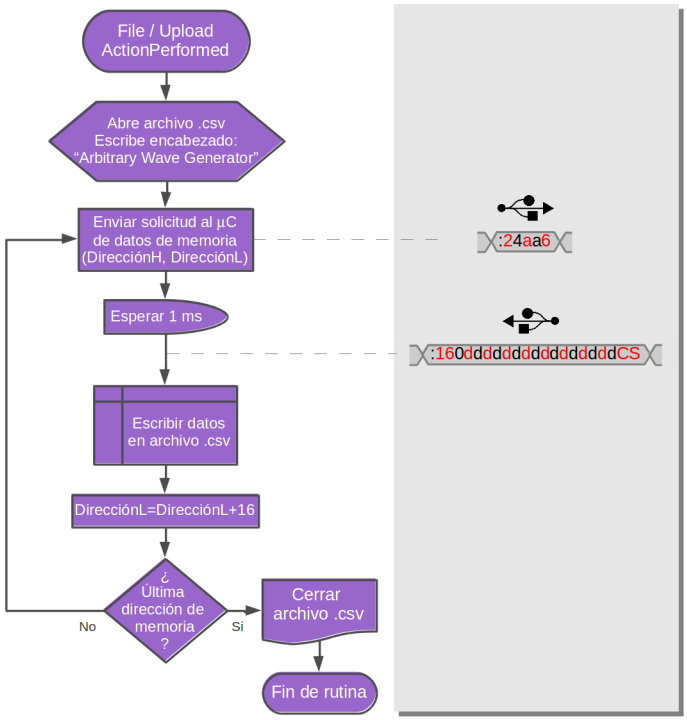


Figura 2.26: Algoritmo de lectura de todos los datos almacenados en la memoria E^2prom desde la aplicación de computadora. Este rutina se ejecuta cuando el usuario pulsa el menú `File/Upload`.

desde la dirección de memoria indicada en la petición, hasta la dirección que se encuentra en un offset de 16; 128 repeticiónes consecutivas de este proceso incrementando en 16 cada vez la dirección inicial del segmento de memoria a leer, permiten barrer todo su contenido y capturar la totalidad de los datos allí almacenados. La figura 2.26 resume en un diagrama de flujo el algoritmo antes descrito.

Por otra parte, cuando el usuario oprime el botón *Download*, se ejecuta una rutina que transmite a través del puerto USB toda la información necesaria para que la forma de onda sea reproducida. Los datos se transmiten por lo menos en tres paquetes, los cuales contienen información sobre el número de datos a alojar en memoria, los datos en sí mismos y una cadena de texto que los describe y que se presenta en el display de cristal líquido, para que el usuario relacione la forma de onda observada con los datos grabados en la memoria E^2prom; ésta última cadena se debe ingresar en el campo de texto *File Name* dispuesto sobre la ventana principal.
Al igual que la rutina que se ejecuta cuando se hace clic en el menú *File/Upload*, los datos se descargan en bloques de 16 bytes correspondientes a 8 valores de ordenadas de la tabla de la ventana principal; debido a que los valores

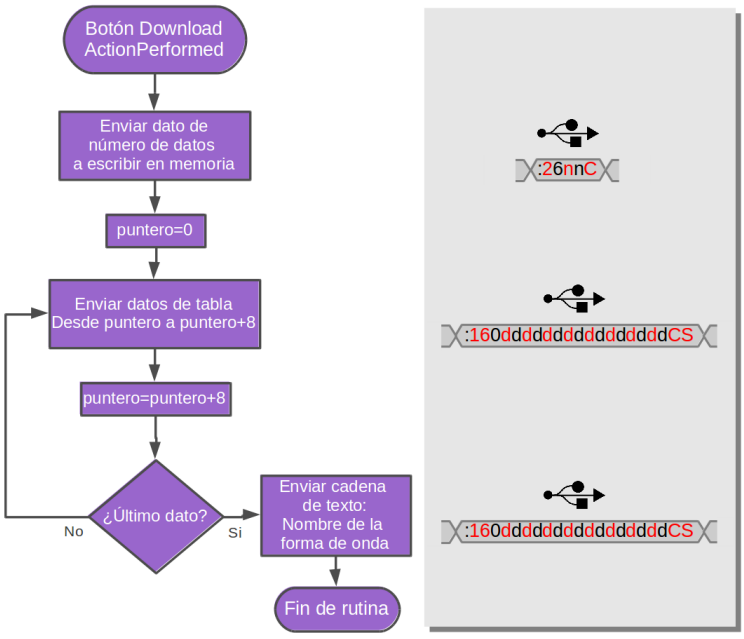


Figura 2.27: Algoritmo de transmisión de datos para escritura en la memoria E^{2}prom desde la aplicación de computadora. Este rutina se ejecuta cuando el usuario pulsa el botón `Download`.

escritos están comprendidos en el rango de números enteros [0-1023], cada número (lo necesite o no) se empaqueta bajo una presentación de 2 bytes[8]. Cuando se terminan de enviar los 8 primeros valores, se envian los 8 siguientes y así susecivamente hasta el número de datos reportado por el usuario en la segunda línea del archivo *.csv*. Cada cadena de datos enviada se acompaña de un código de comando que en el protocolo Intel-Hex, le indica al procesador que debe escribir la información transmitida en la memoria E^{2}prom; de no ser así el sistema colapsaría pues el microcontrolador no cuenta con un sistema de almacenamiento embebido lo suficientemente grande para alojar toda esa información. La figura 2.27 esquematiza el algoritmo que se ejecuta cuando se oprime el botón *Download*.

Oprimiendo el botón *Start*, el usuario define sí se debe o no detener la producción de la forma de onda; este botón es el equivalente virtual del botón de atención parada global (APG) dispuesto físicamente sobre el panel principal del equipo, y su función es la misma: conmutar el estado de la bandera APG la que indirectamente habilita o deshabilita las interrupciones de desbordamiento del timer0 de la programación del microcontrolador.

El botón *Update* se ha dispuesto únicamente para actualizar la gráfica de

[8]Si uno de los datos a transmitir por el USB es por ejemplo el número 2 (cuya representación requiere tan sólo un byte), la cadena se organiza para enviar ese número en dos bytes, es decir 02 (un byte para el cero y otro para el 2).

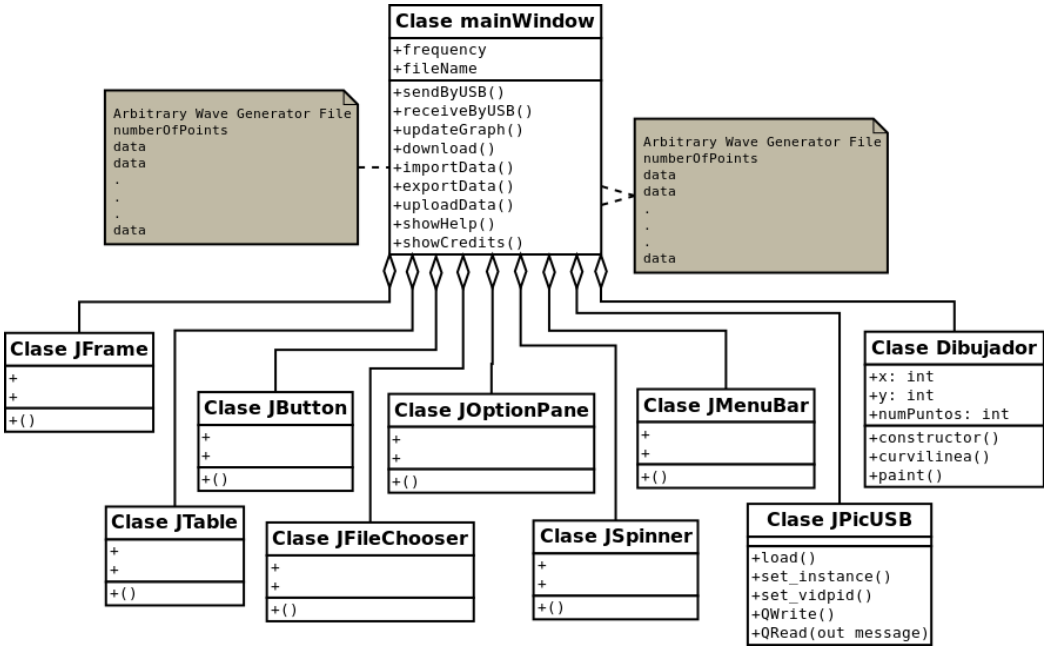


Figura 2.28: Diagrama UML de la aplicación de computadora. Los objetos que componen la ventana principal se muestra en un modelo de agregación; sólo se presentan los atributos y operaciones de mayor interés.

la forma de onda; su uso se hace necesario cuando el usuario edita manualmente una entrada de la tabla de datos de la ventana principal de la aplicación. Adicionalmente, se ha dispuesto el menú *Help* que da acceso a un pequeño tutorial de la aplicación (*Help/How to use?*) y a los créditos del trabajo (*Help/About*).
El diagrama UML[28] que se presenta en la figura 2.28, resume la relación de objetos y clases y los métodos usados en la aplicación de computadora. Resulta de especial interés la clase *JPicUSB*, desarrollada por Geronimo Oñativia[26] y la clase *dibujador* desarrollada por uno de los autores en un trabajo previo[27]. La aplicación se generó como un ejecutable Java (extensión *.jar*), siendo necesario hacer doble clic sobre el archivo sí el usuario se encuentra bajo el sistema operativo Microsoft Windows, o ejecutar el comando

```
java -jar GIFAFTEDDS201305V1.jar
```

sí se encuentra en una terminal del sistema operativo linux.

Capítulo 3

Resultados

Para evaluar el desempeño del generador de forma de onda arbitraria, se han realizado pruebas comparativas del comportamiento observado de la tarjeta electrónica respecto al esperado, y pruebas experimentales haciendo uso de uno de los espectrómetros Mössbauer (AMCMB-96) del Grupo de Metalurgia Física y Teoría de Transciones de Fase (GMTF). En este último punto, el generador de forma de onda arbitraria controló las señales *Waveform, Trigger y Advance* que manejan el transductor de velocidad y el mecanismo de sincronización del analizador multicanal.
Las pruebas de escritorio se realizaron fundamentalmente con uno de los osciloscopios Rigol DS1102E del Grupo de Instrumentación y Física Aplicada, el cual permite la captura 600 puntos con una exactitud de[31],

$$\delta t = \pm\Big(1\ Sample\_Interval + 50\ ppm \times Reading + 0,40\ ns\Big) \tag{3.1}$$

$$\delta y = \pm 3\,\% \times Reading \tag{3.2}$$

con $Sample\_Interval = \frac{1}{500\ MHz}$

3.1. Pruebas de escritorio

3.1.1. Suavización de la forma de onda

El efecto suavizador del filtro Bessel, se ha determinado aplicando a su entrada una señal cuyo periodo presenta saltos bruscos en tensión. Dado que el filtro atenúa los armónicos de alta frecuencia, es de esperar que un cambio discreto de niveles de tensión a su entrada se visualicen en la salida como una señal continua que se traza aproximadamente a través de la diagonal de los estados que componen el salto; en realidad, esta curva corresponde a la

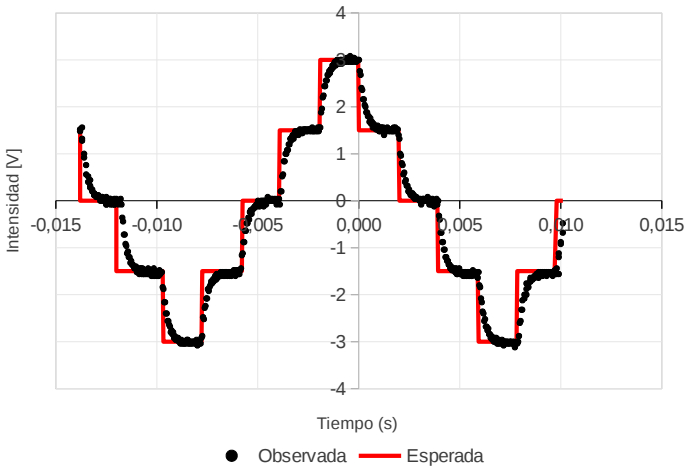


Figura 3.1: Respuesta del filtro Bessel ante una entrada tipo escalón. El efecto neto del filtro es suavizar la forma de onda.

carga y descarga del capacitor C23 (figura 5.2) y se describe por una función exponencial que es creciente o decreciente dependiendo de si se pasa de un estado bajo a uno alto o vicerversa respectivamente. La figura 3.1 ilustra este efecto. A menor escala, es decir, con tiempos de transición más cortos entre estados, el efecto de suavización tiende a ser más lineal.

3.1.2. Análisis de distorsión total armónica

Una prueba de especial interés para evaluar la calidad de onda producida, es la medición de la distorsión total armónica que introduce el equipo cuando se configura para generar algún patrón periódico. Si se considera el equipo como una caja negra en cuya entrada el usuario peticiona una forma de onda cualquiera, es de esperar que esa forma de onda se reproduzca sin mayores diferencias; en particular si la onda corresponde a una función cosenoidal del tipo $y = \cos(\omega t)$, el espectro de Fourier a la salida de la caja debe mostrar sólo el armónico fundamental ubicado en $f_1 = \frac{\omega}{2\pi}$.
Esto puede determinarse cuantitativamente mediante el cálculo de la llamada distorsión total armónica (DTH), la cual examina la contribución de las amplitudes de los armónicos no fundamentales observados en el espectro, respecto a la amplitud del armónico fundamental[29]; la expresión matemática que modela este estadístico es,

Tabla 3.1: Resultados del análisis de distorsión total armónica.

n	f	a_n	b_n	$I_n = \sqrt{a_n^2 + b_n^2}$	$\left(\frac{I_n}{I_1}\right)^2$
DC	0	-0,016			
1	12,7	3,004	0,130	3,006803	
2	25,4	0,034	0,020	0,039205	0,000170
3	38,2	0,000	0,000	0,000000	0,000000
4	50,9	0,000	0,000	0,000000	0,000000
5	63,6	-0,018	0,000	0,017505	0,000034
6	76,3	0,000	0,000	0,000000	0,000000
7	89,1	0,000	0,000	0,000000	0,000000
8	101,8	0,000	0,000	0,000000	0,000000
9	114,5	0,000	0,000	0,000000	0,000000
10	127,2	0,000	0,000	0,000000	0,000000
				$\Sigma =$	0,000204
				DTH=	1,42795 %

$$\%DTH = \sqrt{\sum_{n=2} \left(\frac{I_n}{I_1}\right)^2} \times 100 \tag{3.3}$$

con $I_n \equiv \sqrt{a_n^2 + b_n^2}$ y a_n y b_n las intensidades de las componentes cosenoidal y senoidal respectivamente que resultan del análisis de Fourier.
La figura 3.2 enseña las curvas observada, esperada y su gráfica de diferencias para la producción de una señal cosenoidal; en la figura 3.3 y la tabla 3.1, se muestran además los resultados analíticos de la síntesis de Fourier de la señal observada realizada a partir de una hoja de cálculo[30].

3.1.3. Señales de sincronización

Por otra parte, la figuras 3.4 y 3.5 muestran las señales *Waveform*, *Trigger* y *Advance* capturadas con el osciloscopio tras configurar el generador de forma de onda arbitraria para que produzca una señal triangular de 512 puntos. Las medidas experimentales de los periodos de las señales triangular y Advance para diferentes frecuencias se muestran en la tabla 3.2. Para contar el número de pulsos *Advance* que se reproducen por cada periodo de la señal

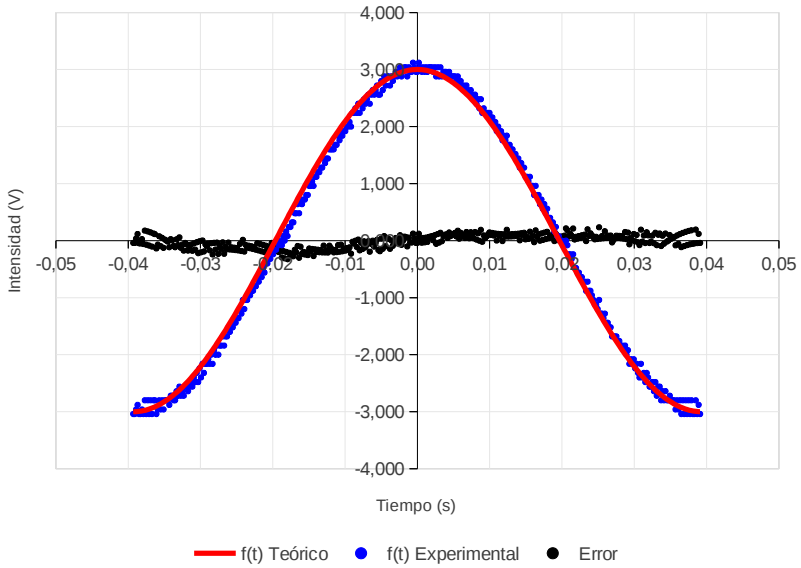


Figura 3.2: Curvas de funciones cosenoidales esperada y observada a la salida del generador de forma de onda arbitraria; la curva de error (en negro) se ha calculado como la diferencia punto a punto entre las dos curvas.

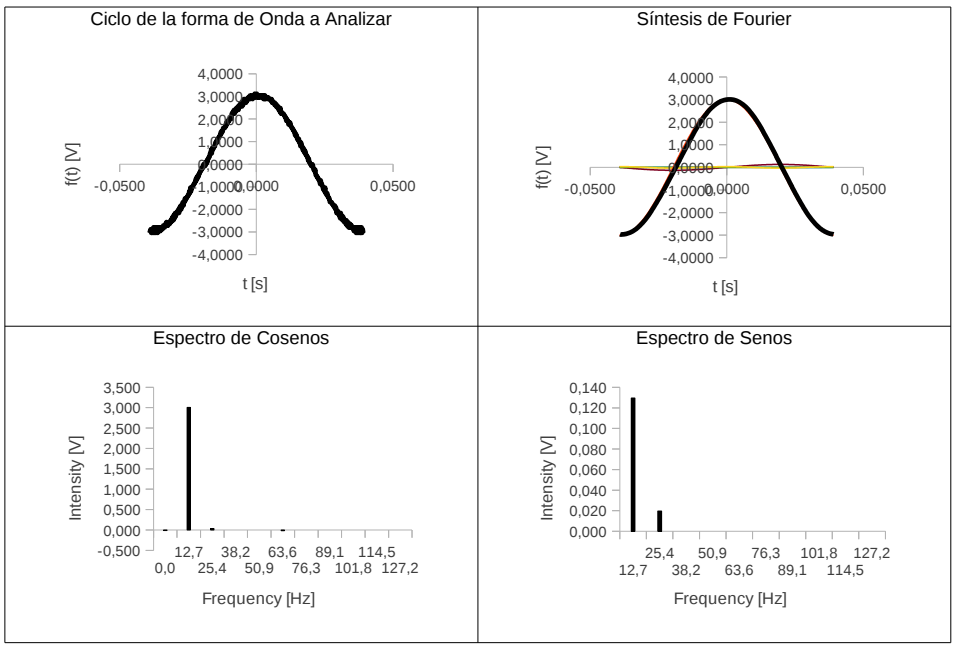


Figura 3.3: Resultados de la síntesis analítica de Fourier de los datos observados de la función cosenoidal de la figura 3.2. La suma de las funciones armónicas (en color) que se aprecian en el recuadro *Síntesis de Fourier* conforman la función sintética que se presenta (en negro).

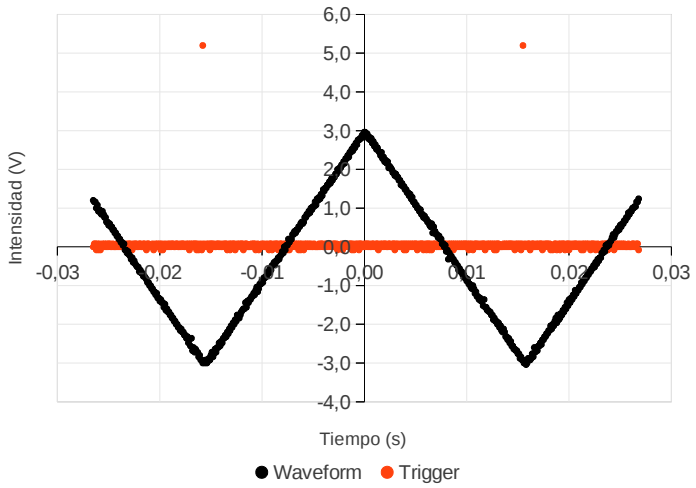


Figura 3.4: Señales *Waveform* y *Trigger* producidas con el generador de forma de onda arbitraria. La señal *Trigger* sólo se produce al inicio de cada periodo.

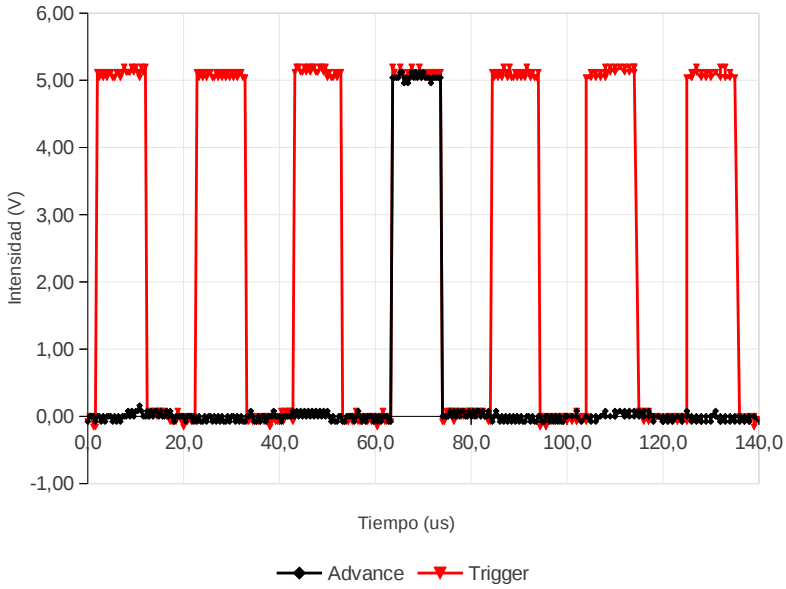


Figura 3.5: Señales *Trigger* y *Advance* producidas con el generador de forma de onda arbitraria. La imagen corresponde a una captura de pantalla del osciloscopio; se ha hecho un acercamiento horizontal para detallar la fase de las señales; el gráfico se ha reproducido con puntos y líneas para una mejor apreciación.

Waveform, se calculó su densidad (η) como la razón del periodo de la forma de onda (T_{wave}) al periodo de la señal *Advance* (T_{ADV}). La incertitumbre en el cálculo de la densidad η se determinó de acuerdo a la regla convencional de propagación de errores,

$$\eta = \frac{T_{wave}}{T_{ADV}} \rightarrow \delta\eta = \frac{1}{T_{ADV}}\delta T_{wave} + \frac{T_{wave}}{T_{ADV}^2}\delta T_{ADV} \tag{3.4}$$

Las incertidumbres δT_{wave} y δT_{ADV} se han obtenido de la hoja de especificaciones del osciloscopio[31] mediante las ecuaciones 3.1 y 3.2.

Tabla 3.2: Medidas experimentales de los periodos de las señales triangular y advance. El valor de frecuencia de la primera columna corresponde al valor ajustado en el generador de forma de onda arbitraria; los tiempos que se reportan en las columnas 2 y 3 han sido medidos con el osciloscopio RIGOL DS1102E[31].

f (Hz)	$(T_{wave}$	$\pm$	$\delta T_{wave})$(ms)	$(T_{ADV}$	$\pm$	$\delta T_{ADV})(\mu$s)	η	$\pm$	$\delta\eta$
2	504	$\pm$	0,025	980	$\pm$	0,052	514,29	$\pm$	0,053
10	102	$\pm$	0,005	199	$\pm$	0,013	512,56	$\pm$	0,063
22	48	$\pm$	0,002	94	$\pm$	0,007	510,64	$\pm$	0,078
30	35	$\pm$	0,002	69	$\pm$	0,006	507,24	$\pm$	0,088

3.1.4. Prueba de linealidad

Para que el generador de forma de onda arbitraria opere como controlador del movimiento del transductor de velocidad del espectrómetro Mössbauer, resulta conveniente analizar la linealidad de la onda triangular de la figura 3.4. Calculando la regresión lineal de las rectas creciente y decreciente de esa gráfica por el método de los mínimos cuadrados[32] se encuentraron los resultados que se presentan en la tabla 3.3.

Tabla 3.3: Parámetros de regresión lineal para las rectas creciente y decreciente de la figura 3.4.

Parámetro	Creciente	Decreciente
Pendiente(m)	386,302	-384,372
Punto de corte (b)	2,966	3,012
Coeficiente de Regresión (R^2)	0,999350	0,999387

3.2. Pruebas de funcionamiento con espectrómetro Mössbauer

El desempeño del generador de forma de onda arbitraria como controlador del movimiento del trasductor de velocidad y del mecanismo de sincronización del analizador multicanal Mössbauer, se evaluó a través de cinco experimentos en los que se evidenciaron espectros de emisión y absorción de radiación γ. La señal *waveform* se conectó al driver de potencia del espectrómetro Mössbauer AMCMB-96[5] que, mediante un sistema PID, permite amplificar, realimentar y controlar el movimiento del transductor de velocidad[11]; por otra parte las señales *Trigger* y *Advance* se conectaron al analizador multicanal *Easy-MCS* del GMTF, que junto al software *Ortec MCS-32* [1] permite la captura de datos digitalizados por el espectrómetro. El diagrama 3.6 detalla las conexiones realizadas. En todos los experimentos, la fuente de radiación, la muestra a estudiar y el detector, se alinearon formando un solo eje óptico, configurando el diseño experimental típico de esta técnica[33] (Figura 1.1) y se ajustó tanto el analizador multicanal como el generador de onda arbitraria para operar en modo de 512 canales.

3.2.1. Espectro de ^{57}Co

La fuente de radiación usada fué ^{57}Co, la cual decae por captura electrónica en ^{57}Fe exitado y emite posteriormente rayos X y γ de diferentes energías[34]. Haciendo incidir directamente la radiación emitida, es decir ejecutando el experimento sin muestra alguna, se puede obtener el espectro característico de la fuente. Para ello basta con conectar sólo la señal advance al analizador multicanal y esperar el tiempo suficiente para que la curva característica se evidencie. La figura 3.7 enseña los espectros de radiación esperado[34] y observado para la fuente de ^{57}Co después de aproximadamente 24 horas de captura de datos. El espectro observado registró los picos principales característicos de la fuente de ^{57}Co. Como se indica en las líneas esperadas, el espectro observado debería reportar un pico muy intenso de rayos X ubicado en 6,40391 keV[34], seguido de un pico de rayos γ de 14,4keV[34] que es particularmente útil en espectroscopía Mössbauer de transmisión [33]. Debido a que el barrido de canales (*SCA Sweep mode* en el software Ortec) arroja una curva de número de cuentas contra tensión aplicada en el detector, fue necesario recalcular el valor de las abscisas para presentarlas en unidades de energía. Para determinar la escala, se calculó la diferencia de posiciones de los picos de rayos X y γ antes descritos y se comparó con la diferencia

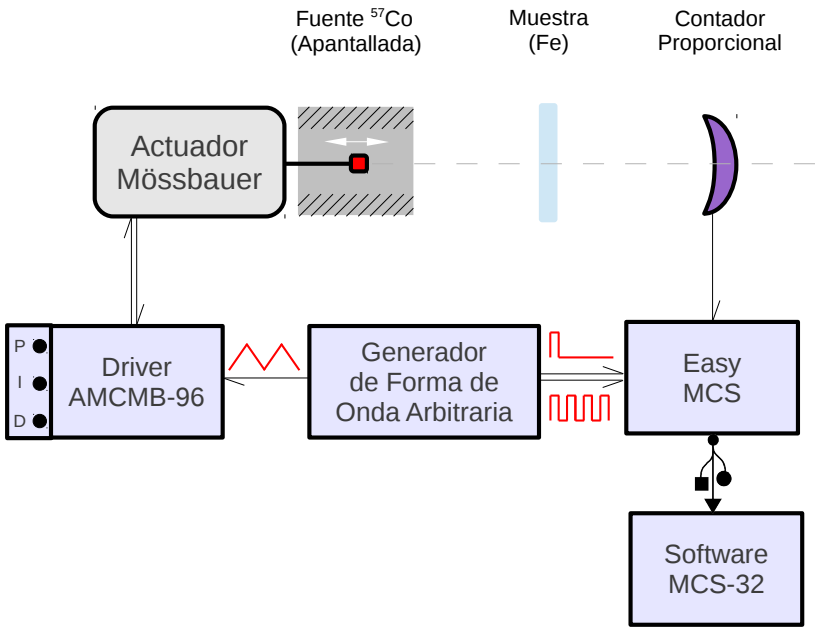


Figura 3.6: Diagrama de conexiones realizadas entre equipos para desarrollar los experimentos de espectrometría Mössbauer con el generador de onda arbitraria.

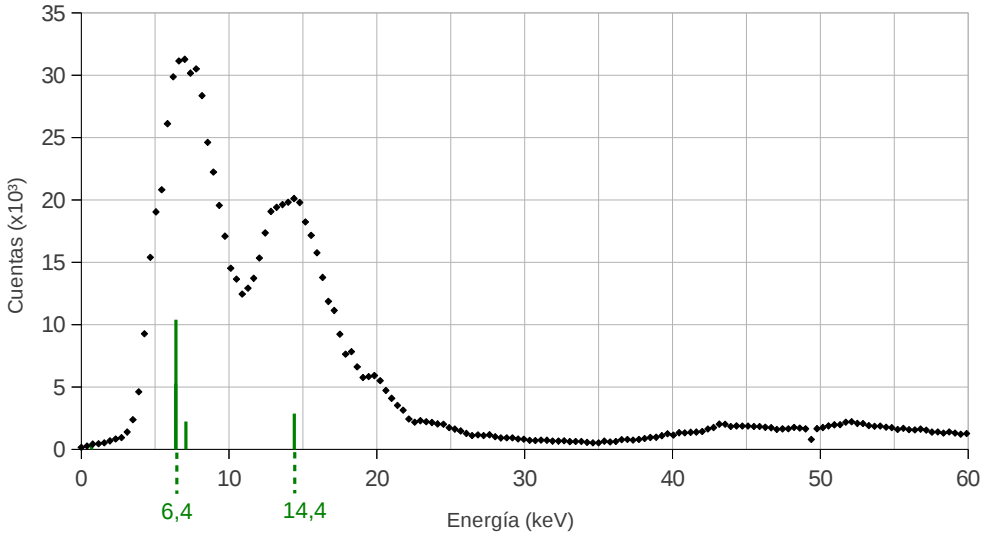


Figura 3.7: Espectros de radiación esperado[34] y observado para la fuente de ^{57}Co.

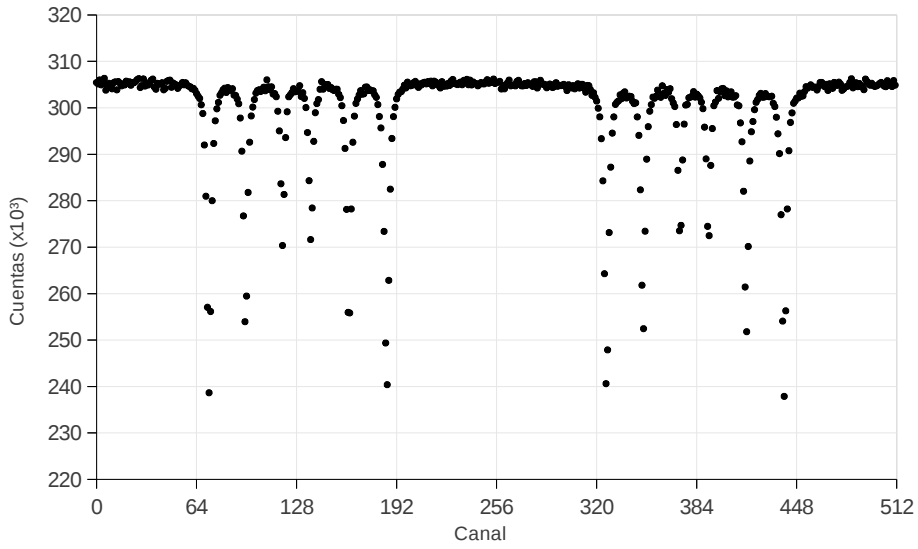


Figura 3.8: Espectrograma observado para la muestra de calibración.

correspondiente al espectro de líneas esperado.

3.2.2. Espectros Mössbauer de muestras que contienen hierro

El primer experimento de espectrometría Mössbaer realizado haciendo uso del generador de forma de onda arbitraria, fué el de una muestra patrón de propiedad del GMTF. De acuerdo a la información que publica el fabricante en su página web[35] con esta muestra se debe evidenciar un espectro de absorción Mössbauer de seis picos (sextete), característico de su composición. El espectro observado y su curva de refinamiento se enseña en las figuras 3.8 y 3.9a.
Una prueba de linealidad de la forma de onda triangular producida lo constituye la simetría del espectro observado, pues con el semiciclo creciente se barren las mismas energías que con el semiciclo decreciente, pero los datos se almacenan en canales diferentes[33].

La segunda clase de experimentos se realizaron con muestras cuya composición contiene algún porcentaje de hierro. La primera muestra fué de hierro natural la cual ha sido estudiada con anterioridad por miembros del GMTF y que se encontraba en el laboratorio de espectrometría Mössbauer en el momento de realizar la prueba. La segunda muestra se obtuvo del bulevar

del rio Cali cuyas barandas, construidas en acero corten, desarrollan al contacto con la atmósfera una gruesa película de óxido que proteje al material de la corrosión[36]. La tercera muestra se obtuvo de una estructura metálica ubicada en la pista de bicicross del bosque municipal de Palmira; esta estructura abandonada, ha desarrollado una gruesa capa de óxido que, contrario a lo que ocurre con las barandas del bulevar del rio Cali, ha permitido la corrosión del material al punto que su deterioro puede calificarse como de pérdida total.

Las muestras se prepararon mediante maceración en mortero hasta obtener un polvo fino que luego se distribuyó lo más uniformemente posible sobre una cinta de enmascarar que se usó como portamuestras. La masa neta de cada muestra se indica en la tabla 3.4.

Los datos brutos que resultaron de las pruebas de espectrometría Mössbauer indicadas anteriormente, se presentan en las figuras 3.9b) c) y d). El doblez de los datos se realizó con un programa en hoja de cálculo desarrollado por Aguirre[37].

La muestra de óxido de hierro del bulevar del rio Cali, se sometió además a una prueba de difracción de rayos X para confirmar la presencia de hierro. Los resultados del refinamiento del difractograma se presentan en la figura 3.10[38][39]; de acuerdo a lo indicado por Burckhardt y Echeverri[36], el acero corten se fabrica como una aleación de hierro con nickel, cromo y cobre de los cuales el último elemento es probablemente el responsable del color rojo-anaranjado característico de este acero y que lo hace tan llamativo para aplicaciones arquitectónicas. Esa hipótesis se confirmaró al observar que era necesario la inclusión de fases de estos elementos para refinar los picos relevantes observados en el difractograma. La prueba se realizó con el difractómetro de rayos X *Panalytical X'pert Pro*; el tiempo de barrido para el rango de ángulos de difracción elegido se calculó en dos horas; las fracciones de fase en peso que se encontraron en este ensayo fueron: (Cr_2O_5) $\rightarrow$ 4,496 % $\pm$ 0,346 %, (Fe_2O_4) $\rightarrow$ 62,769 % $\pm$ 0,234 % y (Fe_3O_4) $\rightarrow$ 32,736 $\pm$ 0,308.

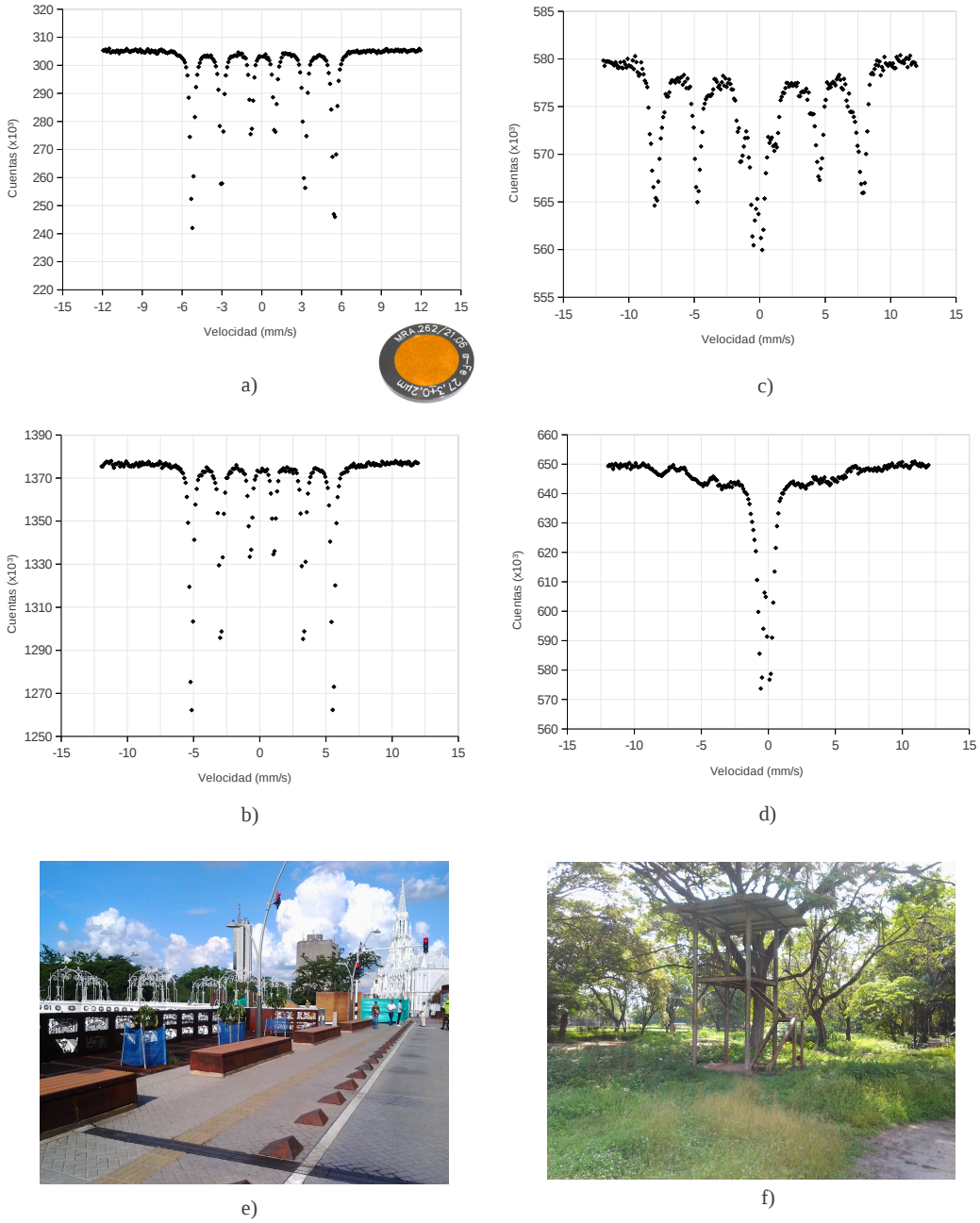


Figura 3.9: Resultados del refinamiento de los espectros Mössbauer obtenidos. a) Muestra de calibración. b) Muestra de hierro natural. c) Muestra del bulevar del rio Cali. d) Muestra de estructura metálica del bosque municipal de Palmira. e) Fotografía del bulevar del rio Cali del día en que se tomó la muestra. f) Ídem para la muestra de la estructura metálica del bosque municipal de Palmira.

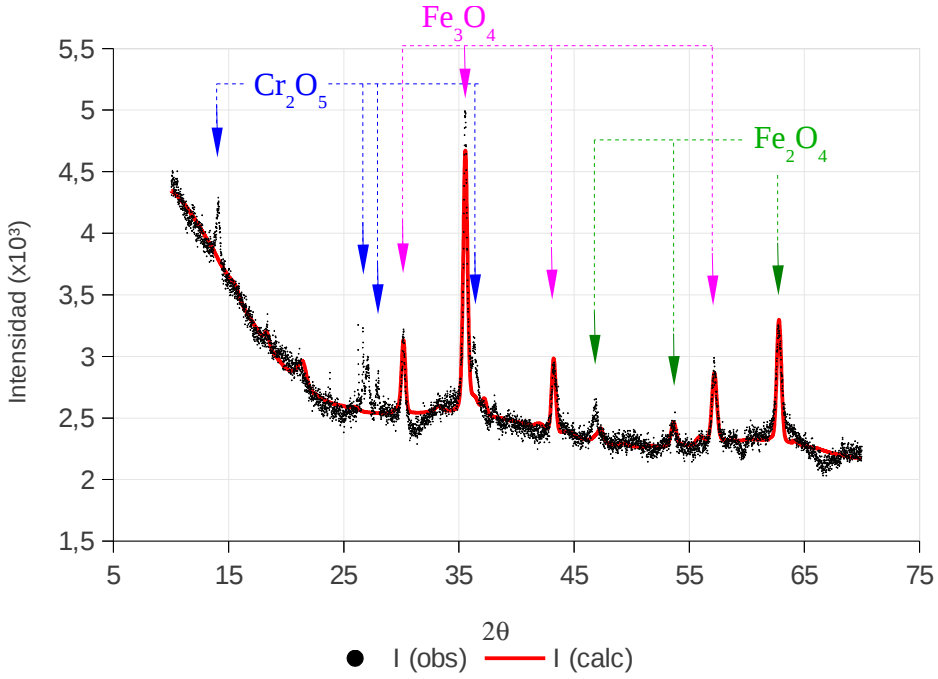


Figura 3.10: Difractograma de rayos x de la muestra del bulevar del rio Cali. El ajuste de la curva teórica a la datos experimentales confirma la presencia de magnetita (y por ende de hierro) en el objeto bajo estudio.

Tabla 3.4: Parámetros espectrales obtenidos del análisis Mössbauer. IS: Desplazamiento Isomérico, QS: Desdoblamiento Cuadrupolar. El factor de velocidad (NC) para todos los experimentos se calculó -a partir de la muestra de calibración- en 0,093382

Muestra	Masa (mg)
Hierro Natural	≈150,0
Bulevar del rio Cali	141,0
Estructura del Bosque	140,2

Capítulo 4

Discusión de resultados y conclusiones

El diseño interior del conversor digital analógico MAX503CWG con la inclusión de latches y una interfaz de 8 bits, tienen la ventaja de hacerlo compatible con microprocesadores como el usado, pero la desventaja que la generación de un valor de tensión requiere de muchos pasos de programación, lo que sacrifica su rapidez. Así, la generación de señales analógicas a través de la técnica de síntesis digital directa, sólo resulta apropiada para la producción de formas de onda de baja frecuencia como las requeridas por la aplicación Mössbauer. Para que éste parámetro tome valores mayores a los presentados en la tabla de especificaciones (anexo 5.1), es necesario contar con un sistema de procesamiento más veloz. Probablemente con un microprocesador como los de las computadoras personales actuales (cuya frecuencia de operación es del orden de 10^9 Hz) o con un procesador digital de señales (DSP), se pueden alcanzar frecuencias cercanas a las generables con circuitos analógicos.

Por otra parte, el generador de forma de onda arbitraria puede en teoría producir señales periódicas de frecuencia tan baja como el valor del registro `TMR0` lo permita. En ese escenario y considerando $P_s = 1$ se obtiene, de conformidad con la ecuación 2.7, el valor de frecuencia $f = 0,0894$ Hz, sin embargo ese valor no resulta práctico puesto que el filtro Bessel a la salida del DAC no cuenta con la capacitancia suficiente para sostener señales escalón durante un tiempo tan largo como 11,1846 s y si lo tuviera permitiría el paso de armónicos de alta frecuencia que desmejorarían la suavidad de la forma de onda. Por esta razón se ha definido como mínima frecuencia 1 Hz, resolución que resulta suficiente para hacer experimentos en expectrometría Mössbauer como lo afirma Sanchez[5].

Un efecto secundario de la suavización del filtro Bessel sobre la forma de onda a la salida del DAC, es que crestas con picos agudos como los que se esperan en una señal triangular, se vean redondos cuando se observa al osciloscopio la señal del conector de salida *Waveform*. Esto es debido a la carga y descarga del capacitor C23 que ocurre cuando la señal esperada alcanza su valor máximo.

El porcentaje de distorsión total armónica que se presenta en la tabla 3.1 se justifica sí se tiene en cuenta que en la síntesis de Fourier de la figura 3.3 aparecieron armónicos senoidales no nulos. En principio, sí una forma de onda armónica presenta un ángulo de fase diferente de cero, la descomposición de Fourier da origen tanto a coeficientes a_n como b_n y por tanto resulta significativamente relevante garantizar en el análisis de distorsión total armónica los datos experimentales sean simétricos, de modo que una de las dos sucesiones (a_n o b_n) se anule y el estadístico $\%DTH$ tienda a su mínimo valor; esta condición se alcanza mediante traslaciones temporales de la forma de onda observada, acción que se puede lograr directamente en el osciloscopio o mediante el procesamiento de datos en una hoja de cálculo, pero su efectividad depende siempre de la resolución de la escala de tiempo del osciloscopio usado. Sin embargo, si se compara el valor de distorsión total armónica reportado en este trabajo con el que se especifica en equipos comerciales como el BK Precision 4017[40] (DTH $< 3\,\%$), el Unisource FG-8102[41] (DTH $< 1\,\%$), el Protek B8003FD[42] (DTH entre 0,2 % y 2,5 %) y los generadores de forma de onda arbitraria Rigol de la serie DG1000[43](DTH $< 0{,}2\,\%$), se puede afirmar que la síntesis de la forma de onda reproducida es exitosa.

El valor alcanzado por los coeficientes de correlación de la regresión lineal (R^2) que se presentan en la tabla 3.3 y la similitud del valor absoluto de las pendientes de las rectas creciente y decreciente de la forma de onda triangular, sugieren que el algoritmo programado en el microcontrolador emplea el mismo tiempo en procesar los pasos consecutivos de tensión que observa el usuario en el conector *waveform*. Una consecuencia de esto, es la simetría observada en el espectro Mössbauer de la figura 3.8.

El espectro de radiación γ de la figura 3.7 y los resultados cualitativos y cuantitativos del espectro Mössbauer correspondiente a la muestra de calibración, así como a los espectros de las muestras bajo estudio, sugieren que el generador de forma de onda arbitraria puede ser usado como controlador del movimiento de la fuente de radiación en experimentos de espectrometría Mössbauer.

Como prueba de verificación y de acuerdo a lo publicado por E. Burckhardt y Echeverri[36], la muestra del óxido en las barandas del bulevar del rio Cali evidencia, de acuerdo a los resultados de espectrometría Mössbauer y de difracción de rayos X, presencia de por lo menos una fase de hierro.

Capítulo 5

Anexos

5.1. Especificaciones Generales

Características físicas

Largo (mm)	Ancho (mm)	Alto (mm)	Masa (kg)
208	206	80	1,823

Características eléctricas

Alimentación:	120 V_{rms}, 60 Hz	
Potencia:	30 W	
Tensión en Conectores de Salida:		
Waveform	Advance	Trigger
±3 V (Máx)	5 V (TTL)	5 V (TTL)

Características metrológicas

Rango de frecuencias:	$\left(1 \text{ a } \frac{30720}{n^{\dagger}}\right)$ Hz
Exactitud en frecuencia:	±1 Hz
DTH:	1,43 %

†n:=número de puntos por periodo

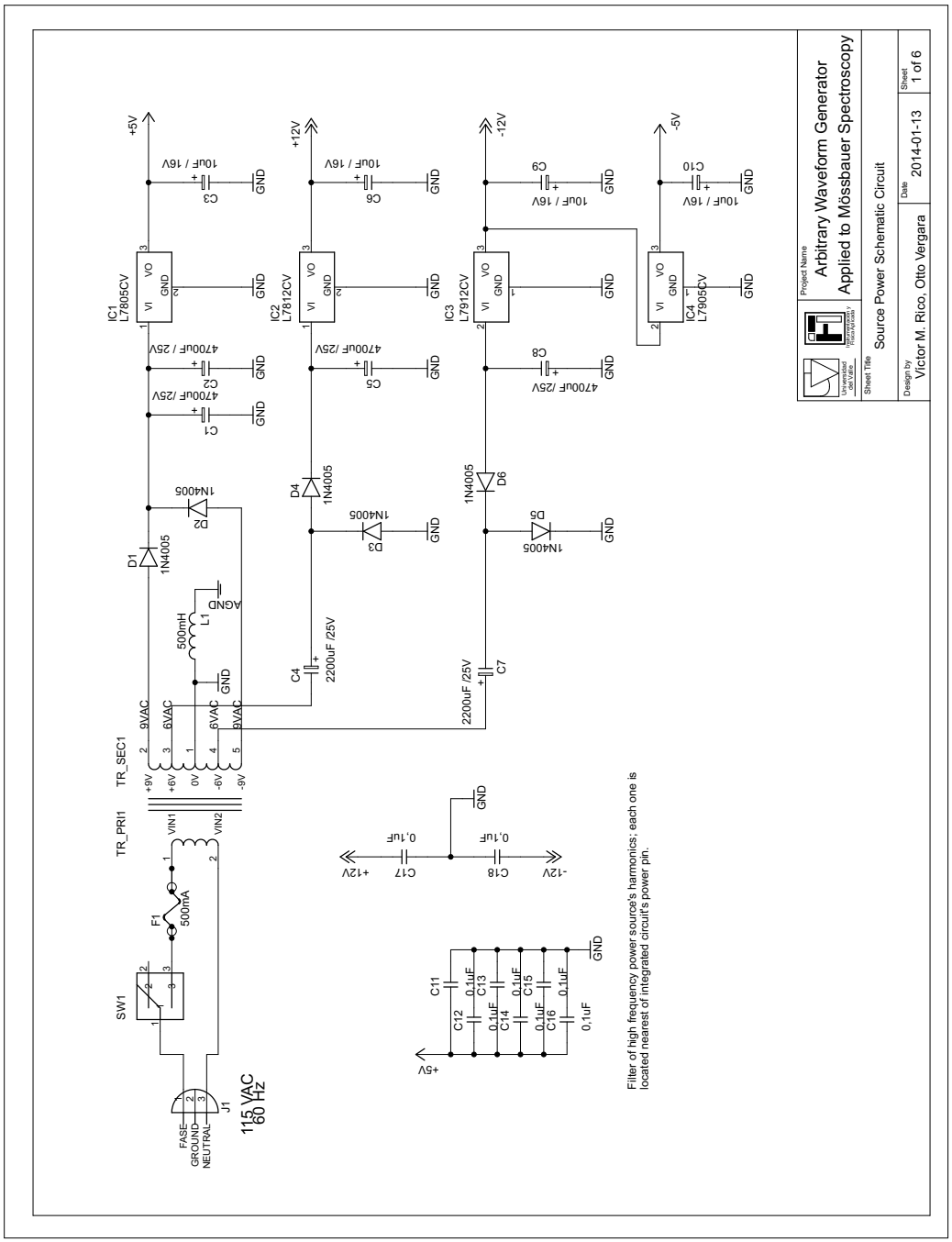
Características computacionales

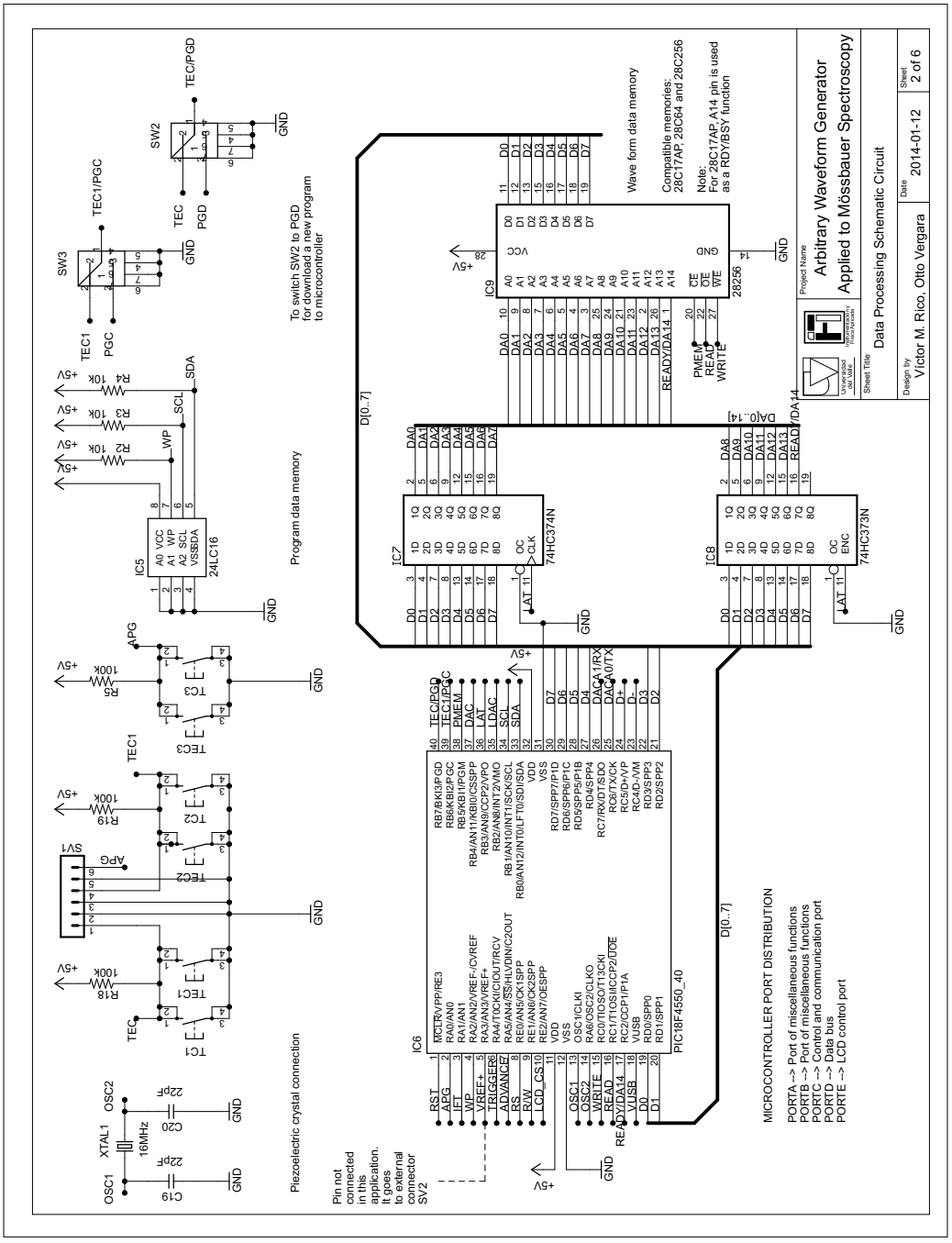
Puerto de comunicaciones:	USB 2.0 y RS-232
Memoria interna:	2 kB
Máximo número puntos por periodo:	1024
Número de bytes por punto:	2

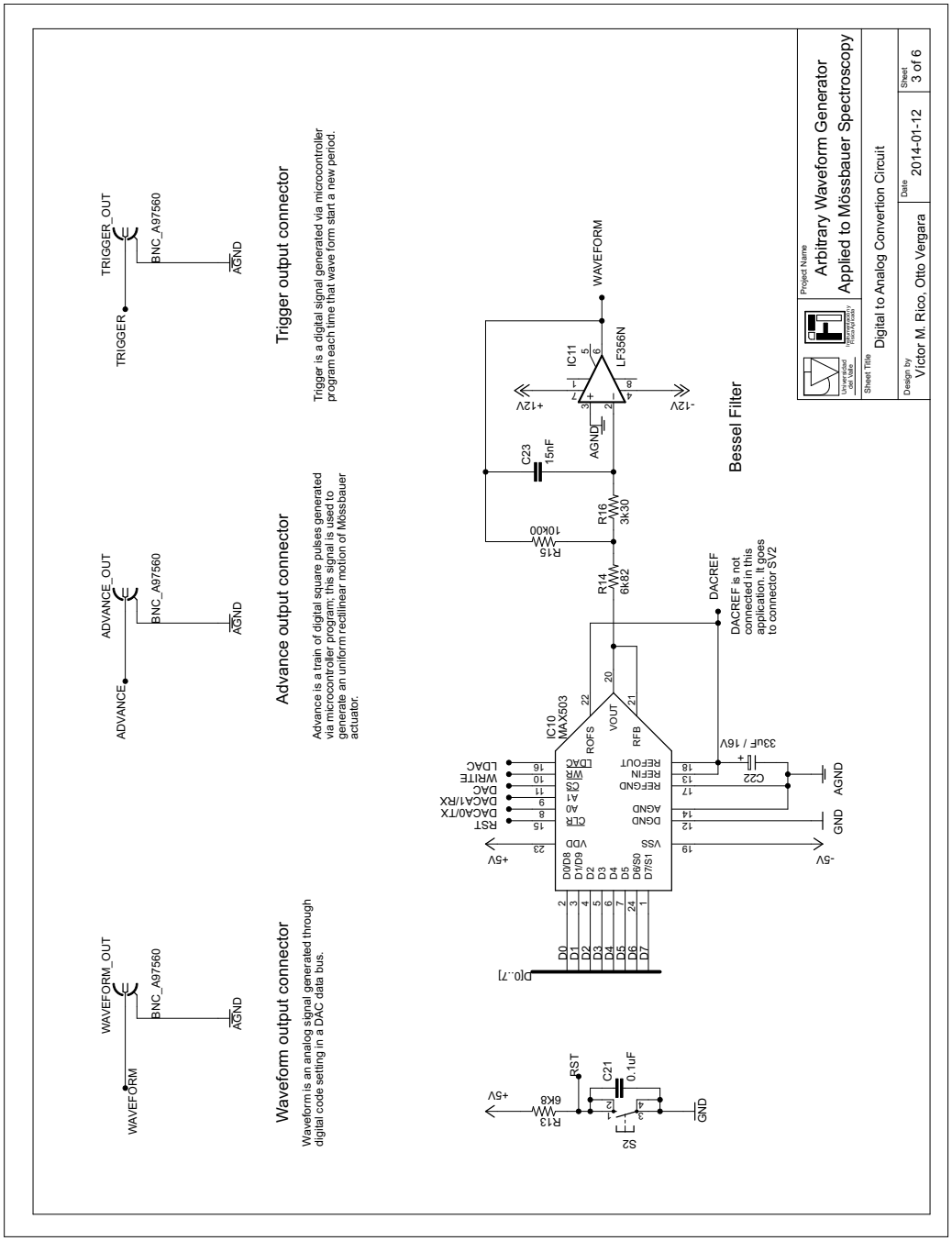
Prerequisitos para ejecutar la aplicación de computadora

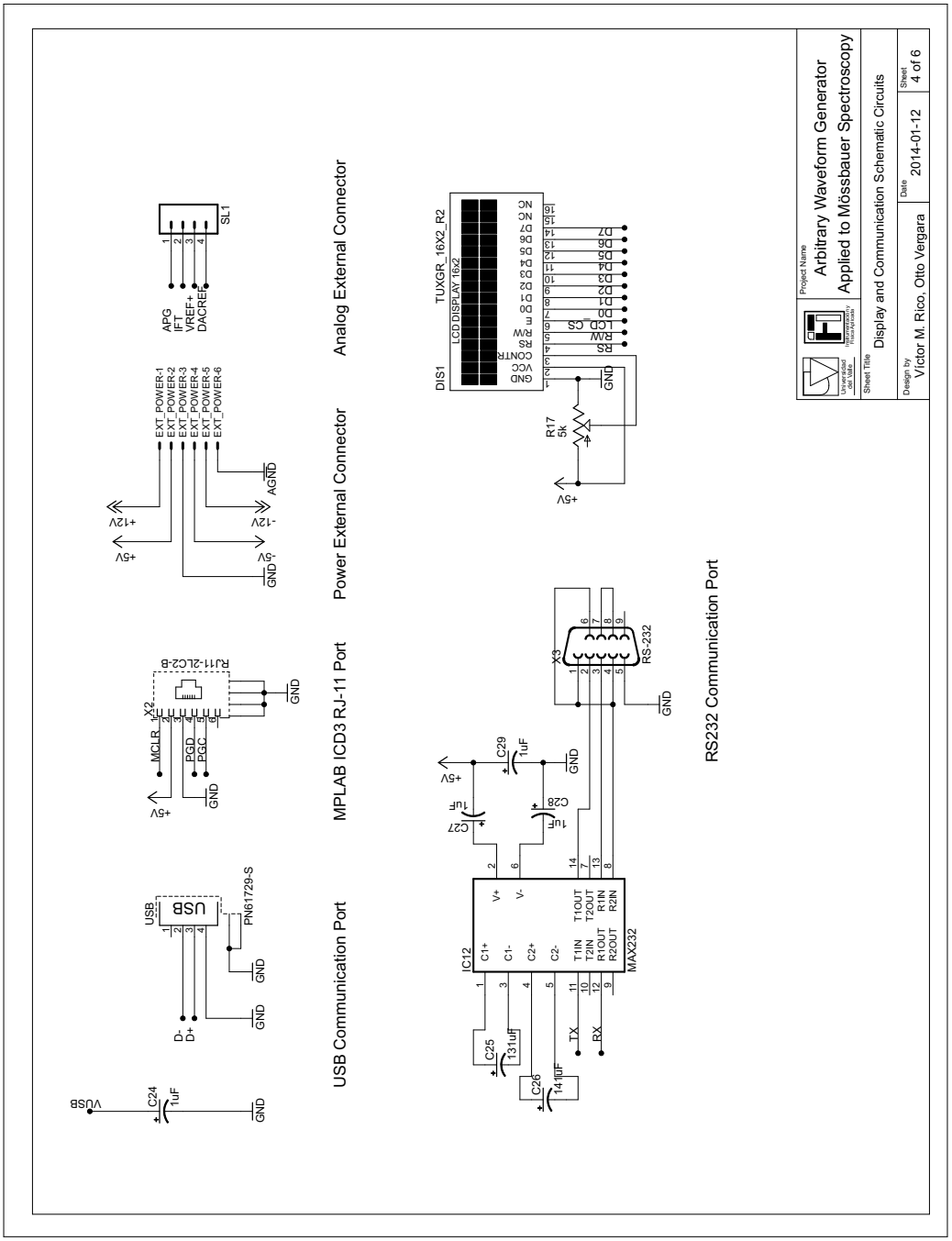
Computadora:	IBM-compatible con al menos un puerto USB disponible y 512 MB de memoria
Sistema operativo:	Microsoft Windows Xp (32 bits)
Software adicional:	JAVA 7.0 o superior Hoja de cálculo o equivalente con opción de generar archivos *.csv*

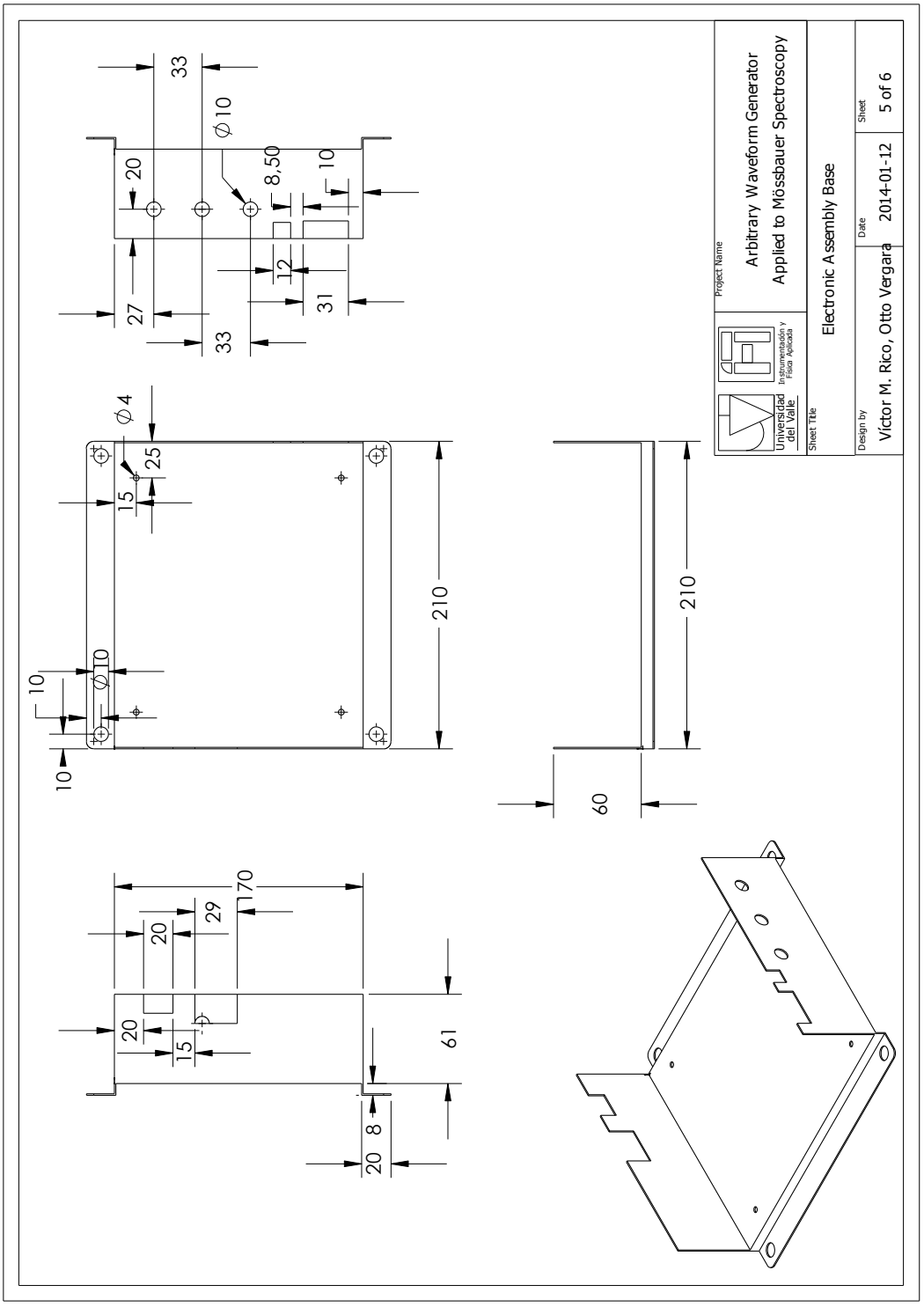
5.2. Planos Esquemáticos

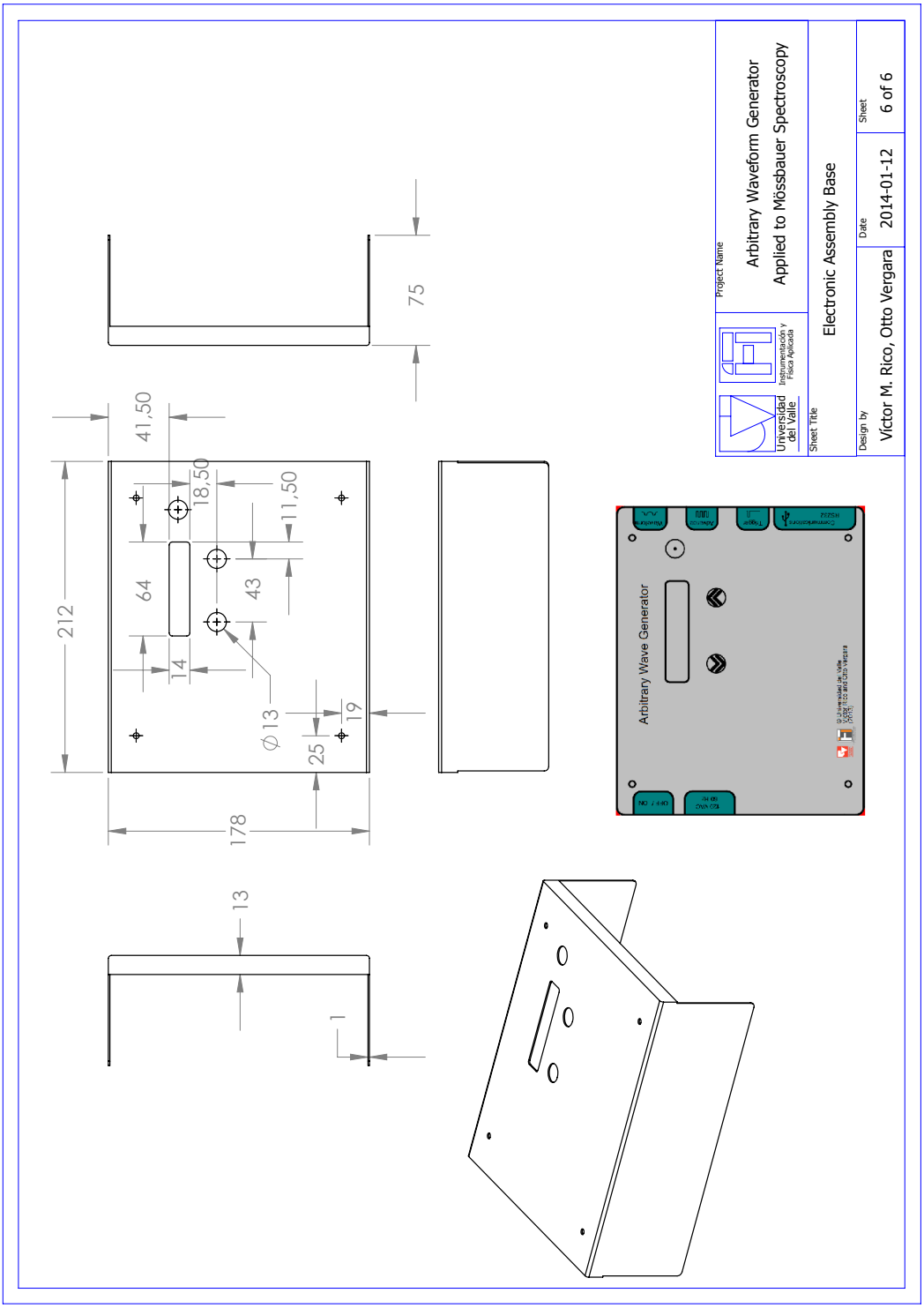

Project Name
Arbitrary Waveform Generator Applied to Mössbauer Spectroscopy
Sheet Title
Source Power Schematic Circuit
Design by
Victor M. Rico, Otto Vergara
Date
2014-01-13
Sheet
1 of 6
Universidad de Costa Rica
Instrumentación Electrónica
TR_PRI1
TR_SEC1
SW1
F1
500mA
J1
FASE
GROUND
NEUTRAL
115 VAC
60 Hz
IC1
L7805CV
IC2
L7812CV
IC3
L7912CV
IC4
L7905CV
L1
500mH
D1
D2
D3
D4
D5
D6
1N4005
C1
C2
C3
C4
C5
C6
C7
C8
C9
C10
C11
C12
C13
C14
C15
C16
C17
C18
2200uF /25V
4700UF /25V
10UF / 16V
0,1uF
+5V
+12V
-12V
-5V
GND
AGND
Filter of high frequency power source's harmonics; each one is located nearest of integrated circuit's power pin.


Arbitrary Waveform Generator
Applied to Mössbauer Spectroscopy
Data Processing Schematic Circuit
Victor M. Rico, Otto Vergara
2014-01-12
2 of 6
Wave form data memory
Program data memory
Compatible memories:
28C17AP, 28C64 and 28C256
Note:
For 28C17AP, A14 pin is used
as a RDY/BSY function
To switch SW2 to PGD
for download a new program
to microcontroller
Piezoelectric crystal connection
Pin not
connected
in this
application.
It goes
to external
connector
SV2
MICROCONTROLLER PORT DISTRIBUTION
PORTA --> Port of miscellaneous functions
PORTB --> Port of miscellaneous functions
PORTC --> Control and communication port
PORTD --> Data bus
PORTE --> LCD control port
PIC18F4550_40
74HC374N
24LC16
28256


Project Name
Arbitrary Waveform Generator Applied to Mössbauer Spectroscopy
Sheet Title
Digital to Analog Convertion Circuit
Design by
Victor M. Rico, Otto Vergara
Date
2014-01-12
Sheet
3 of 6
Universidad del Valle
Instrumentación Electrónica
Trigger output connector
Trigger is a digital signal generated via microcontroller program each time that wave form start a new period.
TRIGGER_OUT
BNC_A97560
TRIGGER
AGND
Advance output connector
Advance is a train of digital square pulses generated via microcontroller program; this signal is used to generate an uniform rectilinear motion of Mössbauer actuator.
ADVANCE_OUT
BNC_A97560
ADVANCE
AGND
Waveform output connector
Waveform is an analog signal generated through digital code setting in a DAC data bus.
WAVEFORM_OUT
BNC_A97560
WAVEFORM
AGND
Bessel Filter
WAVEFORM
IC11
LF356N
+12V
-12V
AGND
C23
15nF
R16
3k30
R15
10k00
R14
6k82
DACREF
DACREF is not connected in this application. It goes to connector SV2
IC10
MAX503
ROFS
VOUT
RFB
REFOUT
REFIN
REFGND
DGND
AGND
VSS
VDD
CLR
A0
A1
WR
CS
LDAC
D0D8
D1D9
D2
D3
D4
D5
D6S0
D7S1
33uF / 16V
C22
+5V
-5V
GND
AGND
RST
DAC0/TX
DAC1/RX
DAC
WRITE
LDAC
D[0..7]
D0
D1
D2
D3
D4
D5
D6
D7
R13
6K8
C21
0.1uF
S2
RST
+5V
GND


Project Name
Arbitrary Waveform Generator Applied to Mössbauer Spectroscopy
Sheet Title
Display and Communication Schematic Circuits
Design by
Victor M. Rico, Otto Vergara
Date
2014-01-12
Sheet
4 of 6
Analog External Connector
Power External Connector
MPLAB ICD3 RJ-11 Port
USB Communication Port
RS232 Communication Port
SL1
APG
IFT
VREF+
DACREF
EXT_POWER-1
EXT_POWER-2
EXT_POWER-3
EXT_POWER-4
EXT_POWER-5
EXT_POWER-6
+12V
+5V
-5V
GND
-12V
AGND
X2
RJ11-2C2-B
MCLR
PGD
PGC
USB
PN61729-S
D-
D+
VUSB
C24
1uF
DIS1
TUXGR_16X2_R2
LCD DISPLAY 16x2
GND
VCC
CONTR
RS
R/W
E
D0
D1
D2
D3
D4
D5
D6
D7
LCD_CS
R17
5k
IC12
MAX232
V+
V-
C1+
C1-
C2+
C2-
T1IN
T2IN
R1OUT
R2OUT
T1OUT
T2OUT
R1IN
R2IN
C25
131uF
C26
141uF
C27
1uF
C28
1uF
C29
1uF
TX
RX
X3
RS-232


Project Name
Arbitrary Waveform Generator
Applied to Mössbauer Spectroscopy
Universidad del Valle
Instrumentación y Física Aplicada
Sheet Title
Electronic Assembly Base
Design by
Víctor M. Rico, Otto Vergara
Date
2014-01-12
Sheet
5 of 6
33
Ø10
20
8,50
10
12
27
31
33
Ø4
25
15
210
10
Ø10
10
210
60
170
29
20
20
15
61
20
8


Project Name
Arbitrary Waveform Generator
Applied to Mössbauer Spectroscopy
Universidad del Valle
Instrumentación y Física Aplicada
Sheet Title
Electronic Assembly Base
Design by
Víctor M. Rico, Otto Vergara
Date
2014-01-12
Sheet
6 of 6
75
41,50
18,50
11,50
64
43
212
14
Ø 13
19
25
178
13
1
Arbitrary Wave Generator

5.3. Cálculo de la función de transferencia del filtro Bessel

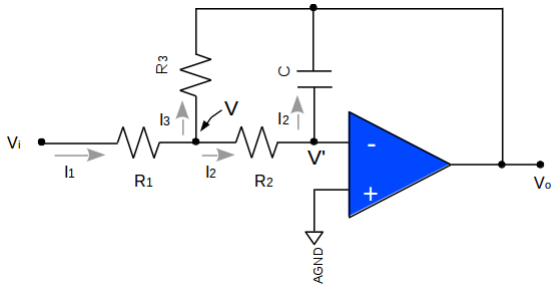


Figura 5.1: Diagrama esquemático del filtro Bessel implementado a la salida del conversor digital analógico.

Parte 1:

Para encontrar la función de transferencia del circuito esquemático de la figura 5.1, debe inicialmente aplicarse la ley de corrientes de Kirchoff al nodo V,

$$I_1 = I_2 + I_3 \tag{5.1}$$

Las corrientes I_1 e I_3 pueden calcularse a partir de la ley de Ohm como,

$$I_1 = \frac{V_i - V}{R_1} \tag{5.2}$$

$$I_3 = \frac{V - V_o}{R_3} \tag{5.3}$$

Si se supone que la impedancia entre los pines inversor y no inversor del amplificador operacional es lo suficientemente grande como para considerar que la corriente entre ellos tiende a cero, entonces ambos pines estarán al mismo potencial y por tanto, el nodo V' estará conectado virtualmente a tierra. Bajo esta hipótesis, la corriente I_2 está dada por,

$$I_2 = \frac{V}{R_2} \tag{5.4}$$

Dado que la corriente hacia la entrada no inversora del amplificador operacional tiende a cero, entonces, la corriente a través de la resistencia R_2 es aproximadamente igual a la corriente que carga al capacitor C. En términos de la capacitancia C, la corriente I_2 puede calcularse como,

$$I_2 = \frac{0 - V_o}{Z_c} = -j\omega C V_o \tag{5.5}$$

Igualando las ecuaciones 5.4 y 5.5 se obtiene una expresión para el potencial V,

$$V = -j\omega C R_2 V_o \tag{5.6}$$

Reemplazando 5.6 en 5.1 se obtiene,

$$\frac{V_i}{R_1} + j\omega C V_o \frac{R_2}{R_1} = -j\omega C V_o - j\omega C V_o \frac{R_2}{R_3} - \frac{V_o}{R_3} \tag{5.7}$$

Agrupando los términos con factor común $j\omega C V_o$ y despejando la razón V_o/V_i se llega a,

$$\frac{V_o}{V_i} = -\frac{1}{j\frac{\omega C}{R_3}\Big(R_1R_2 + R_2R_3 + R_1R_3\Big) + \frac{R_1}{R_3}} \tag{5.8}$$

Sustituyendo la suma de productos de resistencias en el denominador de la ecuación 5.8 por R,

$$R \equiv R_1R_2 + R_2R_3 + R_1R_3 \tag{5.9}$$

y multiplicando y dividiendo por el conjugado del denominador, se obtiene finalmente la expresión,

$$\frac{V_o}{V_i} = \frac{-\frac{R_1}{R_3} + \frac{R}{R_3}\omega C j}{\Big(\frac{R}{R_3}\omega C\Big)^2 + \Big(\frac{R_1}{R_3}\Big)^2} \tag{5.10}$$

que corresponde a la función de transferencia del filtro Bessel que se discute en el capítulo 3.

Parte 2:
Para que la excursión de señal a la salida del filtro Bessel esté comprendida en ±3 V y considerando $\omega = 0$ (el mejor de los casos, pues como se muestra en la figura 2.6a el filtro es paso-bajo) se tiene que para la parte real de la ecuación 5.10,

$$\frac{V_o}{V_i} = -\frac{R_3}{R_1} \tag{5.11}$$

Teniendo en cuenta que la salida del conversor digital análogo —etapa de hardware previa al filtro— genera tensiones entre ±2,048 V y fijando la resistencia $R_3 = 10$ kΩ se obtiene,

$$R_1 = \frac{2,048}{3,000} \times 10\ k\Omega \approx 6,826\ k\Omega \tag{5.12}$$

5.4. Configuración y operación del puerto de comunicación USB con PIC18F4550

El USB es un protocolo de comunicación serial que emplea cuatro hilos para establecer la conexión entre dos equipos electrónicos, uno de los cuales es denominado como *host*. De los cuatro hilos de conexión dos, $D+$ y $D-$, se usan para transmitir datos en forma de pulsos de tensión diferencial, mientras que los otros dos, $V+$ y GND, se usan para proveer potencia desde uno de los equipos hacia el otro; es precisamente esta característica, además de su rapidez, la que impulsó su masificación desde la última década del siglo XX. Una lectura recomendada que resume brevemente las principales características del protocolo de comunicación USB, es la que hace Microchip[13] en la sección *Overview of USB* de la hoja de especificaciones técnicas del microcontrolador PIC18F4550. Los términos más relevantes para efecto de la descripción que aquí se hace son:

Tipo de Potencia:
La configuración del puerto USB responde a dos tipos de configuración de potencia: *self-power*, para el caso en que el equipo USB cuente con su propia fuente de alimentación y *bus-power*, para el caso en que el equipo tome potencia directamente del bus[1]. El estándar USB está diseñado para proveer hasta 100 mA (@ 5 V) de corriente continua por dispositivo, y puede llegar a entregar hasta 500 mA (@ 5 V) en condiciones especiales.
Para garantizar que la configuración self-power tome potencia únicamente de la fuente de alimentación del equipo y no del bus USB, Microchip recomienda adjuntar una impedancia lo suficientemente alta al pin V+ (ó V_{Bus}) como para no drenar corriente del mismo (figura 5.2). Aunque el usuario puede optar por no conectar lo pines de potencia del bus a la alimentación del dispositivo electrónico desarrollado, es necesario acoplar las tierras de los equipos conectados para que se encuentren al mismo potencial. Como lo discute García-Breijo[18], cuando hay disponibilidad de pines, se puede conectar el pin de alimentación V+ a una las interrupciones externas del microcontrolador para detectar el enlace al bus, lo que puede resultar relativamente útil para algunas aplicaciones.

[1]Esta última configuración es apropiada para dispositivos como memorias de almacenamiento masivo (o *memory stick*) donde no resulta práctico contar con una fuente de potencia integrada.

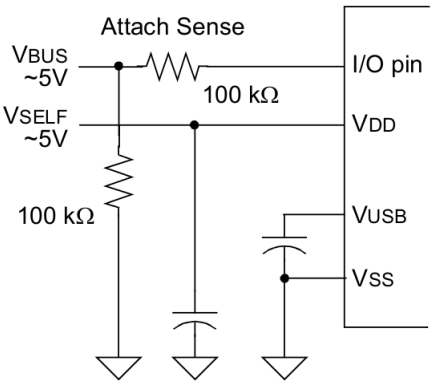


Figura 5.2: . Conexiones externas recomendadas para configurar un dispositivo USB en modo self-power. Fuente: Microchip[13]

.

Tipo de Transferencia:
El puerto USB soporta cuatro tipos de transferencia de datos: *isochronous, bulk, interrupt* y *control*. *Isochronous* y *Bulk* son apropiados para transmitir grandes cantidades de datos. El primero no garantiza la integridad de los datos pero si su tiempo de entrega, mientras que el segundo si garantiza la integridad pero no el tiempo de entrega; para algunas aplicaciones, como por ejemplo la transmisión de audio, puede resultar relativamente útil emplear una transferencia tipo isochronous, pues el usuario raramente percibe el error de unos cuantos bytes sobre el total transmitido. En transferencias tipo *Interrupt* y *Control*, la información es entregada en pequeños paquetes y se asegura la integridad de los mismos. Este último par de tipos de transferencia son los únicos disponibles cuando los equipos USB operan en modo *Low-Speed*, como es el caso de los periféricos de las computadoras -o dispositivos de interfaz humana (HID)- donde la información transmitida es escasa y significativamente espaciada en el tiempo comparada con los tiempos de procesamiento de la máquina.

Rapidez del Bus:
En general el bus USB soporta dos tipos de rapideces: *Full-Speed* y *Low-Speed*. En el PIC18F4550 la rapidez se selecciona por hardware, agregando una resistencia de pull-up entre una referencia de 3,3 V (que la provee el microcontrolador en el pin V_{USB}) y $D+$ para full-speed ó $D-$ para low-speed. La conexión de estas resistencias puede controlarse por programación habilitando o deshabilitando los fuses FSEN y UPUEN de acuerdo a la figura 5.3; de forma alternativa se pueden conectar, sí el diseñador de hardware así lo decide, resistencias de pull-up externas que realicen la misma función. Anexo

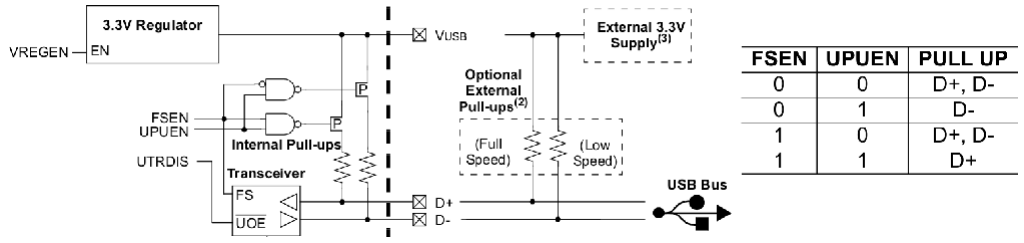


Figura 5.3: Diagrama esquemático del circuito de configuración de rapidez del bus USB. Fuente: Microchip[13]

.

a esto es necesario activar el fuse VREGEN que habilita el funcionamiento del regulador de 3,3 V.

Por otra parte, los dispositivos USB están estructurados en tres niveles de programación a los que se accede dependiendo del tipo de tarea a realizar sobre el puerto. La capa más alta es la de *configuración*, en ella se determinan las especificaciones de potencia. Las capas más bajas, llamadas *endpoint*, es donde se ubican directamente los datos a transmitir o transmitidos. Entre la capa de configuración y los endpoint hay una capa intermedia llamada *Interfase* en la que se realiza la conexión física entre el host y el dispositivo.

Para que la comunicación opere adecuadamente, es necesario que los mensajes enviados desde y hacia el microcontrolador esten sincronizados en frecuencia con la computadora host. Para el estándar 2.0 implementado en este trabajo, dicha frecuencia se ha establecido en 48 MHz. Debido a que los cristales piezoeléctricos que se comercializan regularmente alcanzan frecuencias de hasta aproximadamente 20 MHz, Microchip ha implementado a su interior un multiplicador de frecuencia configurable por el usuario a través de programación, y con el que se puede alcanzar la frecuencia requerida. Para ello refiérase a la figura 5.4 en la que se enseña la fuente de reloj del microcontrolador PIC18F4550; la fuente de oscilador primario, después de pasar por una compuerta schmitt trigger en la que rectifica la cuadratura de la señal de reloj, se lleva a un multiplexor (PLL Prescaler) en el que se debe seleccionar un divisor de frecuencia de modo que, como resultado de dicha división, se obtengan exactamente 4 MHz. Anexo a esto, el microcontrolador cuenta con dos multiplicadores de frecuencia fijos seguidos uno a continuación del otro, con los que se deben obtener frecuencias de 96 MHz y 96/2 MHz exactamente. Direccionando la señal de 96/2 MHz a través de los fuses USBDIV y FSEN se define entonces la frecuencia a la que operará el puerto USB. Así,

para un cristal piezo eléctrico de 16 MHz -como el que se ha dispuesto en la tarjeta electrónica de este trabajo- y haciendo uso de los fuses PLLDIV4, USBDIV y FSEN se obtiene la fuente de reloj de 48 MHz que se buscan.

Una aspecto importante que se debe resaltar de la figura 5.4, es que no por el hecho de sincronizar el puerto USB a 48 MHz se tiene necesariamente la máquina ejecutando instrucciones a esta rapidez; la frecuencia de operación de la máquina se obtiene de la manipulación de los fuses CPUDIV y los fuses característicos según el tipo de oscilador empleado.

Ya en la programación, el código escrito para el microcontrolador debe responder a varios requisitos que establece el protocolo. Cuando un dispositivo se conecta al bus, el equipo host (una computadora por ejemplo) entra en un proceso de enumeración con el que pretende identificarlo; en ese proceso, el host interroga al dispositivo acerca de su consumo de potencia, del tipo de transferencia de datos, protocolo e información descriptiva, debiendo el dispositivo contar con respuestas adecuadas dentro de tiempos adecuados para que la conexión se realice exitosamente; de ahí que es tan importante la sincronización en frecuencia descrita anteriormente.
Toda esta información se almacena en unos vectores, llamados descriptores, que deben cargarse dentro del programa principal. Los descriptores más importantes que deben manipularse para configurar un dispositivo USB son: el *descriptor de dispositivo*, en el que se provee información general del fabricante (vendor identification -VID-), código del dispositivo (product identification -PID-), serial y clase de dispositivo (dispositivo de audio, de almacenamiento masivo, de comunicaciones o dispositivo de interfaz humana -HID-); el *descriptor de configuración*, que provee información acerca de los requisitos de potencia; el *descriptor de endpoint*, que identifica el tipo de transferencia; el *descriptor de interfaz*, que da cuenta del número de endpoints usados, y el *descriptor de cadena*, en el que se almacenan cadenas de texto que se muestran en el host cuando el dispositivo se conecta y que ayudan a su identificación por parte del usuario.

Para programar estas directrices, se ha adjuntado al programa principal el archivo *header.h*, que contiene las líneas de configuración del puerto usb y que puede emplearse como plantilla para otros proyectos. Para la discusión que sigue refiérase al código que se indica en el apartado 5.5.1.
El archivo empieza por definir en un vector de 32 componentes al descriptor de configuración (líneas 5 a 35). La respuesta a la petición de potencia hecha por el equipo host se consigue mediante las declaraciones de las líneas 12 y 13. En la línea 12 se codifica un byte que en el bit 6 indica sí el dispositivo

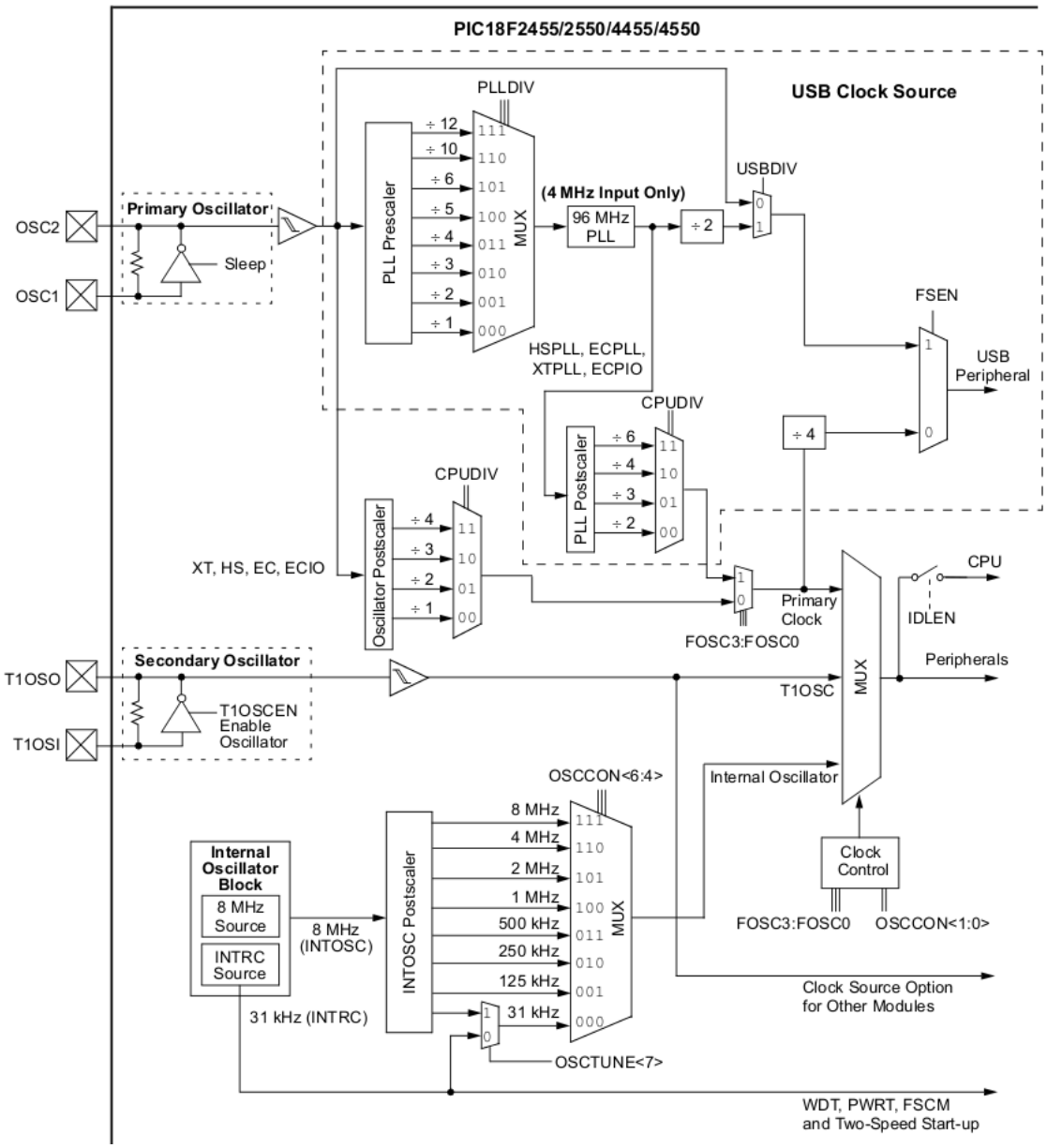


Figura 5.4: Diagrama esquemático de la fuente de reloj del microcontrolador PIC18F4550. Fuente: Microchip[13]

.

opera con fuente de alimentación propia (activo alto para self-power) ó si se alimenta del bus (activo bajo para bus-power). La línea 13 es un byte que establece la máxima potencia requerida por el dispositivo en caso que sea configurado en modo bus-power; dado que la máxima cuenta que se puede obtener con un byte es 255, el cálculo se realiza en términos de la mitad de la máxima corriente a suministrar; así para un valor de 100 mA se debe indicar un valor de $50_{(10)}$ =0x$32_{(16)}$.
El descriptor de dispositivo se define a partir de un vector de 18 componentes (líneas 41 a 56). Para identificar al equipo, es necesario inicializar los parámetros VID (línea 49) y PID (línea 50) con un número único que registra el fabricante en el usb forum[19]; ambos parámetros son números de 16 bit, así, con el registro de un VID se tiene acceso a 65536 PID cada uno de los cuales es asignable a un tipo de equipo desarrollado por el fabricante. Para fines académicos, Microchip permite el uso de los códigos VID=0x$04D8_{(16)}$ y PID=0x$000B_{(16)}$ con los que se pueden operar prototipos y desarrollos experimentales preliminares. Para posicionar equipos en el mercado, se requiere adquirir estos códigos mediante comercio electrónico.
El descriptor de cadena, que contiene los mensajes que presenta la computadora cuando el dispositivo se conecta al usb, se define a partir de un vector de no menos de 8 componentes (líneas 59 a 107). Cada carácter de la cadena de texto a presentar debe pasarse como una componente del vector y siguiendo el protocolo que se enseña en las líneas 66 a 110.
Sí el equipo se conecta a una computadora que opera bajo el sistema operativo Microsoft Windows[2] se inicializa el asistente de instalación de hardware nuevo, el cual solicita al usuario el archivo *mchpusb.inf* que contiene información acerca de su operación (figura 5.5). Este archivo se puede reciclar de la configuración de otro dispositivo usb (como por ejemplo de la instalación de una impresora), debiéndose editar los campos `[DeviceList]` y las cadenas de personalización del dispositivo para facilitar su indexación en el administrador de dispositivos (figura 5.6). Estos campos se deben editar en correspondencia con la información programada en el microcontrolador como se indica a continuación:

```
...
[DeviceList]
%DESCRIPTION%=DriverInstall, USB VID_04D8&PID_000B
...

[Strings]
```

[2]Las pruebas de conexión se ha realizado para Windows XP sin ningún inconveniente.

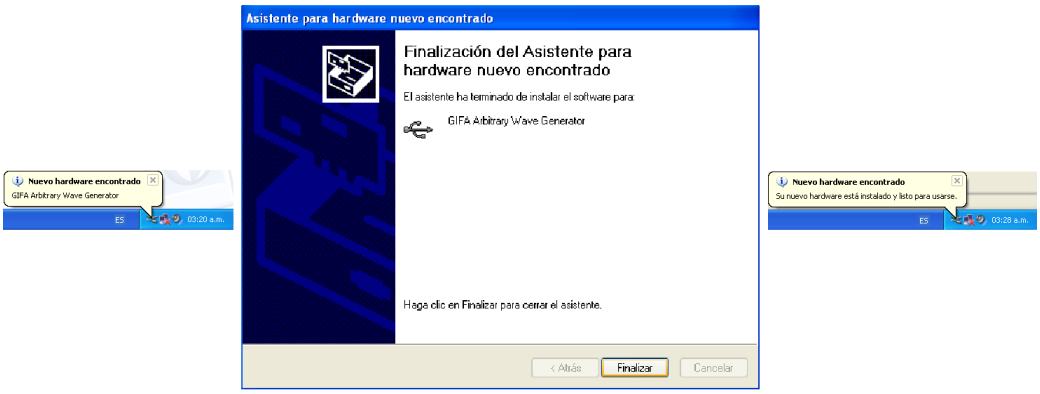


Figura 5.5: Instalación del equipo en el sistema operativo Microsoft Windows

.

```
DEVICEMANAGERCATEGORY="Laboratory Instruments"
MFGFILENAME="mchpusb"
MFGNAME=‘‘Universidad del Valle"
INSTDISK="Microchip Technology, Inc. Installation Disc"
DESCRIPTION="GIFA Arbitrary Wave Generator"
```

Finalmente para inicializar y transmitir datos por el puerto usb, el compilador CCS[20] suministra un conjunto de librerías que facilitan la comunicación entre la computadora y el microcontrolador[18]. Dos de esas librerías -empleadas en este trabajo- son suficientes para conseguir la enumeración por parte del host y enviar datos en ambos sentidos. La primera, llamada *pic18_usb.h*, es el driver de la capa de hardware de la familia de microcontroladores PIC18F4550 y la segunda, llamada *usb.c*, es la controladora de las interrupciones usb y del *usb setup request* del endpoint0. Algunas de las funciones más importantes de esas librerías son:

usb_init(): Inicializa el hardware usb y habilita la interrupción correspondiente.
usb_task(): reinicia el usb stack y el periférico.
usb_enumerated(): devuelve el valor verdadero (true) sí el dispositivo ha sido enumerado por el host. En ese caso, el dispositivo entra queda habilitado para enviar y recibir datos.
usb_kbhit(): devuelve el valor verdadero sí hay un dato nuevo en el buffer rx del usb.
usb_puts(a, message, b, c): transmite datos desde el microcontrolador; el

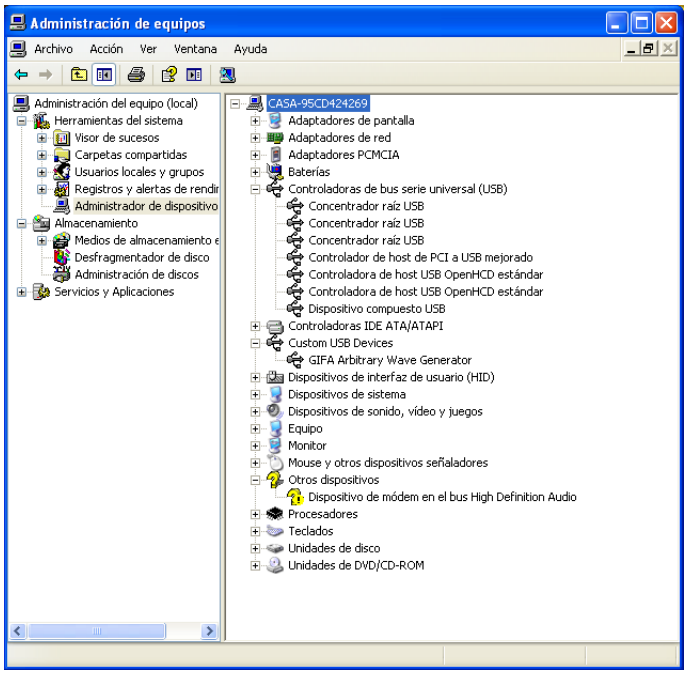


Figura 5.6: Administrador de dispositivos de Microsoft Windows. El generador de onda arbitraria se lista como uno de los equipos instalados correctamente

.

parámetro *a* indica el endpoint a usar en la transferencia; *message* es un vector de bytes que contiene los datos a transmitir; el parámetro *b* es un entero no signado que indica el número de datos a transmitir, el cual puede eventualmente ser la dimensión del vector message; el parámetro *c* , es un entero no signado que indica el tiempo (en milisegundos) empleado para la transmisión de cada paquete. Este último parámetro es particularmente crítico en el tipo de transfencia isochronous.

usb_get_packet(d, recbuf, Lenbuf): captura datos transmitidos hacia el microcontrolador; el parámetro *d* indica el tiempo de retardo entre captura de datos (en ms); el parámetro *recbuf* es un vector entero cuya dimensión se indica en el parámetro *Lenbuf*, el cual es un entero no signado de 8 bits.

5.5. Programas

5.5.1. header.h

```
1   IFNDEF __USB_DESCRIPTORS__
2   #DEFINE __USB_DESCRIPTORS__

3   #include <usb.h>

4   #DEFINE USB_TOTAL_CONFIG_LEN 32
5   char const USB_CONFIG_DESC[] = {  //Descriptor de configuración
6   USB_DESC_CONFIG_LEN,
7   USB_DESC_CONFIG_TYPE,
8   USB_TOTAL_CONFIG_LEN,0,
9   1,
10  0x01,
11  0x00,
12  0xC0,
13  0x32,
14  USB_DESC_INTERFACE_LEN,
15  USB_DESC_INTERFACE_TYPE,
16  0x00,
17  0x00,
18  2,
19  0xFF,
20  0xFF,
21  0xFF,
22  0x00,
23  USB_DESC_ENDPOINT_LEN,
24  USB_DESC_ENDPOINT_TYPE,
25  0x81,
26  0x02,
27  USB_EP1_TX_SIZE,0x00,
28  0x01,
29  USB_DESC_ENDPOINT_LEN,
```

```
30  USB_DESC_ENDPOINT_TYPE,
31  0x01,
32  0x02,
33  USB_EP1_RX_SIZE,0x00,
34  0x01,
35  };
36  #define USB_NUM_HID_INTERFACES 0
37  #define USB_MAX_NUM_INTERFACES 1

38  #if (sizeof(USB_CONFIG_DESC) !=
    USB_TOTAL_CONFIG_LEN)
39  #error USB_TOTAL_CONFIG_LEN
40  #endif

41  char const USB_DEVICE_DESC[] ={     //Descriptor de dispositivo
42  USB_DESC_DEVICE_LEN,
43  0x01,                               //Constante de dispositivo
44  0x10,0x01,                          //Versión usb en bcd
45  0x00,                               //Código de clase
46  0x00,                               //Código de subclase
47  0x00,                               //Código de protocolo
48  USB_MAX_EP0_PACKET_LENGTH,          //Máximo tamaño del paquete para
                                        //el endpoint 0
49  0xD8,0x04,                          //VID
50  0x0B,0x00,                          //PID
51  0x01,0x00,                          //Serial
52  0x01,                               //Índice de la cadena del
                                        //descriptor del fabricante
53  0x02,                               //Índice de la cadena del
                                        //descriptor del producto
54  0x00,                               //Índice de la cadena del
                                        //descriptor del número serial
55  USB_NUM_CONFIGURATIONS              //Número de posibles configuraciones
56  };

57  const char
    USB_STRING_DESC_OFFSET[]={0,4,20};
58  #define USB_STRING_DESC_COUNT
    sizeof(USB_STRING_DESC_OFFSET)

59  char const USB_STRING_DESC[]={      //Descriptor de cadena
60  4,                                  //Índice de la longitud de cadena
```

```
61  USB_DESC_STRING_TYPE,   //Tipo de descriptor 0x03 (STRING)
62  0x09,0x04,
63
64  16,                     //Índice de longitud de la cadena
65  USB_DESC_STRING_TYPE,   //Tipo de descriptor 0x03 (STRING)
66   'F',0,
67   'T',0,
68   'E',0,
69   'D',0,
70   'D',0,
71   'S',0,
72   ' ',0,
73                           //Cadena 2 -->Nombre del dispositivo
74   18,                     //Índice de longitud de cadena
75   USB_DESC_STRING_TYPE,  //Tipo de descriptor 0x03 (STRING)
76   'F',0,
77   'T',0,
78   'E',0,
79   'D',0,
80   'D',0,
81   'S',0,
82   ' ',0,
83   'A',0,
84   'r',0,
85   'b',0,
86   'i',0,
87   't',0,
88   'r',0,
89   'a',0,
90   'r',0,
91   'y',0,
92   ' ',0,
93   'W',0,
94   'a',0,
95   'v',0,
96   'e',0,
97   ' ',0,
98   'G',0,
99   'e',0,
100  'n',0,
101  'e',0,
102  'r',0,
```

```
103  'a',0,
104  't',0,
105  'o',0,
106  'r',0,
107  };
108  #ENDIF
```

5.5.2. FTEDDS1305V1.c

```
/********************* PROJECT SUMMARY ***********************/
Project : Arbitrary Wave Generator Based on Direct Digital
Synthesis
By : Víctor Rico and Otto Vergara, GIFA - Universidad del Valle
Last Revision : 2013-10-30

Abstract : This program manage hardware resources to generate a
periodic waveform whose period is described by data stored in
an EEPROM. Code has routines to show messages in a 16x2 LCD and
attend interrups based on external events and timer overflow.
Communication is done via RS232(when host and equipment are
so distant -more than 5 m-) and USB (when are near one to
one). Communication between hardware and host is done through
software GIFAFTEDDS201305.jar wrote in Java. For Microsoft
Windows is necesary ensure that library jpicusb.inf is located
inside of Java's application folder.

/*************** GENERAL DEFINES AND LIBRARIES***************/
 1   #include <18F4550.h>
 2   #fuses HSPLL,NOWDT,NOPROTECT,
     NOLVP,NODEBUG                        // Defines fuses for uC
 3   #fuses USBDIV,PLL4,CPUDIV1,VREGEN    // Defines fuses for USB
 4   #use fast_io(b)                      //
 5   #use delay(clock=48000000)           // Specifies clock
                                          // speed
 6   #use i2c(MASTER, sda=PIN_B0,
     scl=PIN_B1,FORCE_SW)                 // i2c port configuration
 7   #include <flex_lcd.c>                // lcd library
 8   #define USB_HID_DEVICE FALSE         // Use of HID directives
                                          // disabled
```

```
9   #define USB_EP1_TX_ENABLE          // Turn on EP1(EndPoint1)
    USB_ENABLE_BULK                    // for IN bulk/interrupt transfers
10  #define USB_EP1_RX_ENABLE          // Turn on EP1(EndPoint1)
    USB_ENABLE_BULK                    // for OUT bulk/interrupt transfers
11  #define USB_EP1_TX_SIZE 32         // Size to allocate for the tx
                                       // endpoint 1 buffer
12  #define USB_EP1_RX_SIZE 32         // Size to allocate for the rx
                                       // endpoint 1 buffer
13  #include <pic18_usb.h>             // Microchip PIC18Fxx5x hardware
                                       // layer for CCS's PIC USB driver
14  #include "header.h"                // USB Settings and descriptors by
                                       // this device
15  #include <usb.c>                   // Handles usb setup tokens and
                                       // get descriptor reports
16  #define WP       31746             // pin_a2
17  #define TRIGGER  31748             // pin_a4
18  #define ADVANCE  31749             // pin_a5
19  #define LDAC     31754             // pin_b2
20  #define LAT      31755             // pin_b3
21  #define CSDAC    31756             // pin_b4
22  #define CSPMEM   31757             // pin_b5
23  #define WRITE    31760             // pin_c0
24  #define READ     31761             // pin_c1
25  #define READY    31762             // pin_c2
26  #define DACA0    31766             // pin_c6
27  #define DACA1    31767             // pin_c7
/************************** VARIABLES  *******************************/
28  char          _addressLow=0;       // Byte low of memory address
29  char          address_Low=0;       //
30  char          _addressHigh=0;      // Byte high of memory address
31  char          address_High=0;      //
32  int8          byteHigh;            //
33  int8          byteLow;             //
34  int8          lowData=0;           //
35  signed int8   highData=0;          //
36  const  int8   Lenbuf = 32;         //
37  int8          recbuf[Lenbuf];      //
38  int8          freq_digit0;         //
39  int8          freq_digit1;         //
40  int8          freq_digit2;         //
41  char          frequency=1;         //
```

```
42  char           correctedFrequency=1;  //
43  int8           taskRegister=0;        //
44  int8           checksum=0;            //
45  unsigned byte  g, s;                  // cycle's index
46  byte           message[16];           //
47  byte           message2[16];          //
48  byte           waveformName[16];      //
49  boolean        auxA=true;             //
50  boolean        auxB=false;            //
51  byte           aux=0;                 //
52  int16          numberOfData=512;      // number of data per period
53  int16          n=0;                   //
54  int16          number=0;              //
55  signed int8    inc=1;                 //
56  signed int8    incCounter=0;          //
57  int16          counter=0xFE01;        // 0xFE01; it is the second
                                          // complement of (65535-511+1)
58  int16          counterCharge=0xFE01;  // 0xFC01-->1024, 0xFE01-->512;
59  int            channelsFactor=4;      // channelsFactor=2 ->1024 chann
                                          // channelsFactor=4 ->512 chann
60  boolean        APG=false;             //
61  byte           ADV=0;                 //
62  int16          valueT0=65455;         //
63  #BYTE          PORT_A=0x0F80          //
64  #BYTE          PORT_B=0x0F81          //
65  #BYTE          PORT_C=0x0F82          //
66  #BYTE          PORT_D=0x0F83          //

/*****************************  ROUTINES  ************************/
67  void inicialization()                 //
68  {                                     //
69     lcd_init();                        // LCD inicialization
70     lcd_putc("Triangle");              // String to LCD's line 1
71     lcd_gotoxy(1,2);                   // Define cursor position
72     lcd_putc("F= 1Hz");                // String to LCD's
73     lcd_gotoxy(10,2);                  // Define cursor position
74     lcd_putc("Start?");                // String to LCD's
75     SET_TRIS_A(0x01);                  // TRISA=0b 0000 0001: 0=Out
                                          // PA0->APG
76     SET_TRIS_B(0xC0);                  // TRISB=0b 1100 0000: 0=Out
                                          // PB7->key up, PB6->key down
```

```
77     SET_TRIS_C(0x04);                // TRISC=0b 0000 0100: 0=Out
                                        // LCD control port
78     SET_TRIS_D(0x00);                // TRISD=0b 0000 0000: 0=Out
                                        // data bus direction
79     output_high (DACA0);             // DAC A0 Initial condition
80     output_high (DACA1);             // DAC A0 Initial condition
81     output_high (CSDAC);             // DAC CS Initial condition
82     output_high (WRITE);             // Initial condition of WRITE
83     output_high (LDAC);              //
84     delay_ms(500);                   // Stabilization time
85     usb_init();                      //
86     usb_task();                      //
87     SET_TIMER0(valueT0);             //
88     SETUP_TIMER_0(T0_INTERNAL        //
                  | RTCC_DIV_1);        // Default PS=1
89     enable_interrupts(INT_RB);       //
90     enable_interrupts(GLOBAL);       //
91     output_low(WP);                  //
92     i2c_start();                     //
93     i2c_write(0x01);                 //
94     i2c_write(.ª");                  //
95     i2c_stop();                      //
96     int8 ojo;                        //
97     i2c_start();                     //
98     i2c_write(0x01);                 //
99     ojo=i2c_read();                  //
100    i2c_stop();                      //
101    lcd_gotoxy(14,1);                //
102    lcd_putc(ojo);                   //
103 }

104 int8 readMemory(char address_Low,   //
                    char address_High)  //
105 {                                   //
106    output_high (WRITE);             //
107    output_d(address_Low);           //
108    output_high (LAT);               //
109    output_d(address_High);          //
110    output_low (LAT);                // Address Latched
111    SET_TRIS_D(0xFF);                // Port D as an input
112    output_low(CSPMEM);              //
113    output_low(READ);                //
```

```
114    output_high(READ);           //
115    output_high(CSPMEM);         //
116    return(input_d());           //
117 }                               //
118 void writeMemory(char aLow,     // Task 0: Is activated by
          char aHigh, byte data)   // host through download button
119 {                               //
120    output_high (CSPMEM);        // Memory's CS to High
121    output_high (WRITE);         // WRITE to high
122    output_high (READ);          // READ to high
123    output_d (aLow);             //
124    output_high (LAT);           // LAT to High
125    output_d (aHigh);            //
126    output_low (LAT);            // Address Latched
127    output_d(data);              //
128    output_low (WRITE);          //
129    output_low (CSPMEM);         //
130    output_high (CSPMEM);        //
131    output_high (WRITE);         //
132    while(!READY) {}             // Wait whereas memory is busy
133    delay_ms(10);                // Time to ensure data writing
134    bit_clear(taskRegister,0);   //
135 }

136 void writeDAC()                 //
137 {                               //
138    byteHigh=(lowData&0xC0)      // Two MSB of byteLow are caught
             +(highData);          // ByteLow and added with byteHigh
139    byteLow=lowData&0x3F;        // Two MSB are cleared
140    SET_TRIS_D(0x00);            // Port D as an output
141    output_d(byteHigh);          // Data for NBH
142    #asm                         //
143    BSF PORT_C,6                 //
144    BSF PORT_C,7                 //
145    BCF PORT_B,4                 //
146    BCF PORT_C,0                 //
147    BSF PORT_C,0                 //
148    BSF PORT_B,4                 //
149    #endasm                      //
150    output_d(byteLow);           // Data for NBM and NBL
151    #asm                         //
152    BCF PORT_C,6                 //
```

```
153    BCF PORT_C,7                          //
154    BCF PORT_B,4                          //
155    BCF PORT_C,0                          //
156    BSF PORT_C,0                          //
157    BSF PORT_B,4                          //
158    BCF PORT_B,2                          //
159    BSF PORT_B,2                          //
160    #endasm                               //
161 }

162 void DDS()                               // Task 1: Is activated
163 {                                        // by timer0's overflow
164    #asm                                  //
165    BCF PORT_A,4                          // Trigger cleared
166    BTG PORT_A,5                          // Advance Signal
167    BTG ADV,0                             //
168    #endasm                               //
169    if(bit_test(ADV,0))                   //
170    {                                     //
171       if(n<numberOfData)                 //
172       {                                  //
173          highData=readMemory(            //
            address_Low, address_High);      //
174          address_Low++;                  //
175          lowData=readMemory(             //
            address_Low, address_High);      //
176          writeDAC();                     //
177          n++;                            //
178          if (address_Low<255)            //
179             address_Low++;               //
180          else                            //
181          {                               //
182             address_Low=0;               //
183             address_High++;              //
184          }                               //
185       }                                  //
186       else                               //
187       {                                  //
188          address_Low=0;                  //
189          address_High=0;                 //
190          n=0;                            //
```

```
191          #asm                        //
192          BTG PORT_A,4;               // Trigger Signal
193          #endasm                     //
194       }                              //
195       bit_clear(taskRegister,1);     //
196    }                                 //
197    else                              //
198    {                                 //
199       #asm                           //
200       NOP                            //
201       NOP                            //
202       NOP                            //
203       NOP                            //
204       NOP                            //
205       NOP                            //
206       NOP                            //
207       NOP                            //
208       NOP                            //
209       NOP                            //
210       NOP                            //
211       NOP                            //
212       NOP                            //
213       NOP                            //
214       NOP                            //
215       NOP                            //
216       NOP                            //
217       NOP                            //
218       NOP                            //
219       NOP                            //
220       NOP                            //
221       NOP                            //
222       NOP                            //
223       NOP                            //
224       NOP                            //
225       NOP                            //
226       NOP                            //
227       NOP                            //
228       NOP                            //
229       NOP                            //
230       NOP                            //
231       NOP                            //
232       NOP                            //
```

```
233       NOP  //
234       NOP  //
235       NOP  //
236       NOP  //
237       NOP  //
238       NOP  //
239       NOP  //
240       NOP  //
241       NOP  //
242       NOP  //
243       NOP  //
244       NOP  //
245       NOP  //
246       NOP  //
247       NOP  //
248       NOP  //
249       NOP  //
250       NOP  //
251       NOP  //
252       NOP  //
253       NOP  //
254       NOP  //
255       NOP  //
256       NOP  //
257       NOP  //
258       NOP  //
259       NOP  //
260       NOP  //
261       NOP  //
262       NOP  //
263       NOP  //
264       NOP  //
265       NOP  //
266       NOP  //
267       NOP  //
268       NOP  //
269       NOP  //
270       NOP  //
271       NOP  //
272       NOP  //
273       NOP  //
274       NOP  //
```

```
275       NOP  //
276       NOP  //
277       NOP  //
278       NOP  //
279       NOP  //
280       NOP  //
281       NOP  //
282       NOP  //
283       NOP  //
284       NOP  //
285       NOP  //
286       NOP  //
287       NOP  //
288       NOP  //
289       NOP  //
290       NOP  //
291       NOP  //
292       NOP  //
293       NOP  //
294       NOP  //
295       NOP  //
296       NOP  //
297       NOP  //
298       NOP  //
299       NOP  //
300       NOP  //
301       NOP  //
302       NOP  //
303       NOP  //
304       NOP  //
305       NOP  //
306       NOP  //
307       NOP  //
308       NOP  //
309       NOP  //
310       NOP  //
311       NOP  //
312       NOP  //
313       NOP  //
314       NOP  //
315       NOP  //
316       NOP  //
```

```
317       NOP  //
318       NOP  //
319       NOP  //
320       NOP  //
321       NOP  //
322       NOP  //
323       NOP  //
324       NOP  //
325       NOP  //
326       NOP  //
327       NOP  //
328       NOP  //
329       NOP  //
330       NOP  //
331       NOP  //
332       NOP  //
333       NOP  //
334       NOP  //
335       NOP  //
336       NOP  //
337       NOP  //
338       NOP  //
339       NOP  //
340       NOP  //
341       NOP  //
342       NOP  //
343       NOP  //
344       NOP  //
345       NOP  //
346       NOP  //
347       NOP  //
348       NOP  //
349       NOP  //
350       NOP  //
351       NOP  //
352       NOP  //
353       NOP  //
354       NOP  //
355       NOP  //
356       NOP  //
357       NOP  //
358       NOP  //
```

```
359       NOP          //
360       NOP          //
361       NOP          //
362       NOP          //
363       NOP          //
364       NOP          //
365       NOP          //
366       NOP          //
367       NOP          //
368       NOP          //
369       NOP          //
370       NOP          //
371       NOP          //
372       NOP          //
373       NOP          //
374       NOP          //
375       NOP          //
376       NOP          //
377       NOP          //
378       NOP          //
379       NOP          //
380       NOP          //
381       NOP          //
382       NOP          //
383       NOP          //
384       NOP          //
385       NOP          //
386       NOP          //
387       NOP          //
388       NOP          //
389       NOP          //
391       NOP          //
392       NOP          //
393       NOP          //194 Not operation
394       #endasm
395    }
396 }

397 void sendState()   // Task 2: Is activated by
398 {                  // host
399    message[0]=99;  //
```

```
400    usb_puts(1,message,1,1);                  // Send the number 99 via
401    bit_clear(taskRegister,2);                // endpoint1
402 }                                            //

403 void interpretUSBMessage()                   // Task 3: Is activated
404 {                                            // by host via USB
405    int i=1;                                  //
406    usb_get_packet(1, recbuf, Lenbuf);        //
407    for(i=1;i<recbuf[1]+3;i++)                //
408    checksum=checksum+recbuf[i];              //
409    if (checksum!=recbuf[recbuf[1]+3])        //
410    {                                         //
411       lcd_gotoxy(15,1);                      //
412       lcd_putc(¡");                          //
413    }                                         //
414    checksum=0;                               //
415    switch(recbuf[2])                         //
416    {                                         //
417       case 0:                                // Update name of
418          for(i=1;i<recbuf[1];i++)            // waveform
419             waveformName[i-1]=recbuf[i+2];   //
420          lcd_gotoxy(1,1);                    //
421          for(i=1;i<recbuf[1];i++)            //
422             lcd_putc(waveformName[i-1]);     //
423          lcd_gotoxy(1,2);                    //
424          lcd_putc("F= 1Hz ");                //
425          lcd_gotoxy(10,2);                   //
426          lcd_putc("Start?");                 //
427          break;                              //
428       case 1:                                // Send state to pc
429          bit_set(taskRegister,2);            // application
430          break;                              //
431       case 2:                                // Start / Stop command
432          bit_set(taskRegister,4);            //
433          break;                              //
434       case 3:                                // Ajust frequency
435          frequency=recbuf[3];                //
436          bit_set(taskRegister,5);            //
437          break;                              //
438       case 4:                                // Send memory's data
439          _addressLow=recbuf[3];              // to pc application
```

```
440          _addressHigh=recbuf[4];          //
441          for(i=0;i<16;i++)                // command 4 send
442             message2[i]=readMemory(       // packets of 16
               i+_addressLow,_addressHigh);   // bytes
443          usb_puts(1,message2,16,1);       //
444          break;                           //
445       case 5:                             // Write Memory
446          lcd_gotoxy(1,1);                 //
447          lcd_putc("Downloading ");        //
448          lcd_gotoxy(1,2);                 //
449          lcd_putc("Data ");               //
450          for(g=0;g<16;g++)                //
451          {                                //
452             s=g+16*recbuf[19];            //
453             writeMemory(s,recbuf[20],     //
                recbuf[g+3]);                 //
454          }                                //
455          bit_set(taskRegister,0);         //
456          break;                           //
457       case 6:                             // Set the number of
458          numberOfData=((int16)(           // data per period
              recbuf[3]))*100+                //
              (int16)(recbuf[4]);             //
459          break;                           //
460    }                                      //
461 }

462 void globalStop()                         // Task 4: Is activated
463 {                                         // via host or APG button
464    if(recbuf[3]==0)                       // Stop generation.
465    {                                      //
466       lcd_gotoxy(10,2);                   //
467       lcd_putc("Start?");                 //
468       APG=true;                           //
469       disable_interrupts(INT_TIMER0);     //
470    }                                      //
471    if(recbuf[3]==1)                       // Start generation
472    {                                      //
473       lcd_gotoxy(10,2);                   //
474       lcd_putc("* ");                     //
475       APG=false;                          //
```

```
476       enable_interrupts(INT_TIMER0);      //
477    }                                      //
478    bit_clear(taskRegister,4);             //
479 }                                         //
480 void updateFrequency()                    // Task 5: is activated
481 {                                         // either by host though
482    correctedFrequency=(char)(             // frequencySpinnerButton
         -0.01071495*frequency*frequency+     // or by user via up/down
         1.96652536*frequency+0.04723371);    // button
483    freq_digit2=correctedFrequency/100;    //
484    freq_digit1=((correctedFrequency-      //
         freq_digit2*100)/10);                //
485    freq_digit0=(correctedFrequency-       //
         freq_digit2*100-freq_digit1*10);     //
486    lcd_gotoxy(4,2);                       //
487    if(freq_digit2!=0)                     //
         lcd_putc(freq_digit2+0x30);          //
488    if(freq_digit2!=0 || freq_digit1!=0)   //
         lcd_putc(freq_digit1+0x30);          //
489    lcd_putc(freq_digit0+0x30);            //
490    lcd_putc("Hz ");                       //
491    valueT0=(int16)(65535-                 //
         6000000/(numberOfData*frequency));   //
492    bit_clear(taskRegister,5);             //
493 }

494 void APGScanner()                         //
495 {                                         //
496    auxA=bit_test(PORT_A,0);               //
497    if(!auxA)                              //
498    {                                      //
499       if(!APG)                            //
500       {                                   //
501          APG=true;                        //
502          lcd_gotoxy(10,2);                //
503          lcd_putc("Start?");              //
504          disable_interrupts(INT_TIMER0);  //
505       }                                   //
506       else                                //
507       {                                   //
508          APG=false;                       //
```

```
509          lcd_gotoxy(10,2);              //
510          lcd_putc("* ");                //
511          enable_interrupts(INT_TIMER0); //
512       }                                 //
513       while(!auxA)                      //
514          auxA=bit_test(PORT_A,0);       //
515       delay_ms(20);                     // Anti-bouncing time
516    }                                    //
517 }

/*************** INTERRUPTIONS (HANDLERS)    *****************/
518 #INT_RB                                 //
519 void changePortB_isr()                  //
520 {                                       //
521    aux=input_b();                       //
522    aux&=0xC0;                           // Keep just PB7 and PB6
523    if(aux==0x40)                        // Key up pushed
524       if(frequency<255)                 //
525           frequency++;                  //
526    if(aux==0x80)                        // Key down pushed
527       if(frequency>1)                   //
528           frequency--;                  //
529    disable_interrupts(INT_RB);          //
530    delay_ms(100);                       // Keyboard stabilization
531    enable_interrupts(INT_RB);           // time
532    bit_set(taskRegister,5);             //
533 }                                       //

534 #INT_TIMER0                             //
535 void timer0Overflow()                   //
536 {                                       //
537    SET_TIMER0(valueT0);                 //
538    bit_set(taskRegister,1);             //
539 }

/************************** MAIN ROUTINE    ********************/
540 void main()                             //
541 {                                       //
542    inicialization();                    // Port configuration
543    while (TRUE)                         //
544    {                                    //
```

```
545         if(bit_test(taskRegister,1))    //
              DDS();                        //
546         if(bit_test(taskRegister,2))    //
              sendState();                  //
547         if(usb_enumerated())            // It line research if
              if (usb_kbhit(1))             // there is a new data
                  interpretUSBMessage();    // in USB's rx buffer
548         APGScanner();                   //
549         if(bit_test(taskRegister,4))    //
              globalStop();                 //
550         if(bit_test(taskRegister,5))    //
              updateFrequency();            //
551     }                                   //
552 }
/************************ END OF PROGRAM   ********************/
```

5.6. Listado de Partes y Proveedores

Tabla 5.1: Listado de partes y proveedores

Ítem	Referencia	Descripción	Proveedor	Valor (2013)
1	TR506	Transformador	GIFA-UniValle	$18000 [COP]
2	74HC373	Latch	GIFA-UniValle	$2000 [COP]
3	74HC374	Latch	GIFA-UniValle	$2000 [COP]
4	28C17AP	Memoria de 2 kB	GIFA-UniValle	$9100 [USD]
5	PIC18F4550	Microcontrolador de 8 bits	EM[A1]	$12000 [COP]
6	MAX232	Driver RS-232	EM[A1]	$3100 [COP]
7	24LC16	Memoria serial	EM[A1]	$1200 [COP]
8	LF356	Amplificador operacional	EM[A1]	$900 [COP]
9	L7805CV	Regulador 5V	EM[A1]	$700 [COP]
10	L7812CV	Regulador 12V	EM[A1]	$700 [COP]
11	L7905CV	Regulador -5V	EM[A1]	$700 [COP]
12	L7912CV	Regulador -12V	EM[A1]	$700 [COP]
13	TUXGR_16X2_R2	Display 16 caracteres, 2 líneas	EM[A1]	$13000 [COP]
14	- -	Fusible 250 mA	EM[A1]	$200 [COP]
15	BBC BOBINA	Inductor	EM[A1]	$500 [COP]
16	MS-307	Interruptor	EM[A1]	$1000 [COP]
17	KS-101	Conector potencia 110 V	EM[A1]	$900 [COP]
18	1N4004	Diodo Rectificador	EM[A1]	$100 [COP]
19	- -	Capacitor $1\mu F$	EM[A1]	$100 [COP]
20	R 3k3	Resistencia Precisión	EM[A1]	$200 [COP]
21	- -	Capacitor $6800\mu F$	ETC[A2]	$3000 [COP]
22	Molex 6 pines	Conector	ETV[A3]	$800 [COP]
23	- -	Terminal para cable	ETV[A3]	$100 [COP]
24	MAX503CWG	DAC 10 bits	Digikey[A4]	$6,73 [USD]
25	EG4791-ND	SWITCH PUSH SPST-NO 0.01A 35V	Digikey[A4]	$0,62 [USD]
26	8425K-ND	STDOFF HEX M/F 6-32 2.000”L ALUM	Digikey[A4]	$0,763 [USD]
27	8414K-ND	HEX STANDOFF 6-32 ALUMINUM 1/2”	Digikey[A4]	$0,52 [USD]
28	H702-ND	MACHINE SCREW PAN PHILLIPS 2-56	Digikey[A4]	$0,0732 [USD]
29	H212-ND	HEX NUT 3/16”2-56	Digikey[A4]	$0,0353 [USD]

Tabla 5.2: Listado de partes y proveedores (continuación)

Ítem	Referencia	Descripción	Proveedor	Valor (2013)
30	H154-ND	MACHINE SCREW PAN SLOTTED 6-32	Digikey[A4]	$0,0228 [USD]
31	H220-ND	HEX NUT 5/16"6-32	Digikey[A4]	$0,0265 [USD]
32	COD0020270	Circuito impreso	Microcircuitos[A5]	$83333 [COP]
33	- -	Soporte para circuito impreso	Rolma[A6]	$30000 [COP]
34	- -	Tapa metálica	VAG[A7]	$41000 [COP]
35	- -	Adhesivo	CITE[A8]	$20000 [COP]

[A1] Electrónica Megacentro. Cr 6 16-45 Local 024A. Santiago de Cali-Colombia. (+57) (2) 8801693.
[A2] Electrónica Trade Center. Cr 100 11-90 Local 246. Santiago de Cali-Colombia. (+57) (2) 3744933.
[A3] Electrónicas TV & Video. Cl 31 24-07. Palmira-Colombia. (+57) (2) 2739885.
[A4] Digikey Corporation. 701 Brooks Ave South. Thief River Falls-USA. www.digikey.com
[A5] Microcircuitos. Cr 24 5-90. Santiago de Cali-Colombia. (+57) (2) 5189577.
[A6] Iluminación y Metaleléctrica S.A.S. Cl 18 4-69. Santiago de Cali-Colombia. (+57) (2) 8831092.
[A7] VAG Ingeniería. Cl 42 34C-11. Palmira-Colombia. (+57) (2) 2703228.
[A8] Circuitos Impresos Teclados de Membrana. Cl 28 11A-22. Santiago de Cali-Colombia. (+57) (2) 4421024.

Bibliografía

[1] Ametek, inc. Ortec Easy-MCS Multichannel Scaler. http://www.ortec-online.com (2013).

[2] M. de la Cruz, P. López. *Sistema de test automático para equipos de vía férrea*. REE **May** (2009).

[3] M. J. Madero *et. al.* Plataforma para la caracterización del efecto memoria en circuitos no lineales de microondas. Universidad de Sevilla (2004).

[4] P. Gütlich, E. Bill, A. Trautwein. Mössbauer Spectroscopy And Transition Metal Chemestry. Springer (2011).

[5] A. Sanchez. Tesis Doctoral: *Diseño y Construcción de un Analizador Multicanal Mössbauer y su Aplicación al Estudio de Sistemas Metálicos*. Universidad del Valle (1997).

[6] A. Sanchez, A. Simar, O. Vergara. Microgenio: Sistema microprocesado para automatización y control. V encuentro de informática universitaria (1988).

[7] O. Vergara. *ASLAB - ASistente de LABoratorio*. Interfaz electrónica para la realización de prácticas de laboratorio de física Fundamental I. Universidad del Valle. (Sin publicar) (2009).

[8] O. Ojeda. *Diseño y construcción del primer prototipo de un generador de funciones*. Tesis de Físico. Universidad del Valle (2009).

[9] E. Murphy, C. Slattery. All About Direct Digital Synthesis. Analog Dialog **38-08** (2004).

[10] X. Solans. *Introducción práctica a la síntesis digital directa*. www.qsl.net (2001).

[11] R. R. Rodriguez. *Construcción de un transductor de velocidad y su aplicación al estudio de la serie de aleaciones Fe, Mn, Al.* Tesis de Físico. Universidad del Valle (1997).

[12] Maxim, MAX503 Data Sheet: 5 V, low-power, parallel input, voltage-output, 10 bit DAC. Rev 0 (1994).

[13] Microchip Technology Inc. PIC18F2455/2550/4455/4550 Data Sheet: 28/40/44-Pin, High-Performance, Enhanced Flash, USB Microcontrollers with nanoWatt Technology. Rev DS39632E (2009).

[14] W. Hayt, J. Kemmerly. *Relaciones fasoriales para R, L y C* en Análisis de Circuitos en Ingeniería. Quinta Edición. Mc Graw Hill (1994).

[15] Intersil Corporation. *Understanding Glitch in a High Speed D/A Converter*. Technical Brief TB325.1 (1999).

[16] Maxim, MAX232 Data Sheet: Dual EIA-232 Drivers/Receivers. SLLS047L (Rev March 2004).

[17] Requirements and Recomendations for USB Products with Embedded Host and/or Multiple Receptacles. Rev. 1.0. www.usb.org (2004)

[18] E. García-Breijo. *Universal Serial Bus* en Compilador C CCS y simulador Proteus para microcontroladores PIC. Alfaomega (2008).

[19] USB Implementers Forum. http://www.usb.org/developers/vendor/ (2013).

[20] Custom Computer Services Inc. CCS C Compiler (2008)

[21] Intel Inc. *Hexadecimal Object File Format Specification*. Revision A. http://www.microsym.com/content/index.php?pid=4&id=25 (1988).

[22] Digikey. Digital to Analog Converters. www.digikey.com/product-search/en/integrated-circuits-ics/data-acquisition-digital-to-analog-converters-dac/2556292?k=dac (2013).

[23] Catalyst Semiconductor, Inc. CAT28C17AP 16K-Bit CMOS Parallel E^2PROM. Doc No 25034-00 2/98 (1998).

[24] Oracle. Java: computer programming language www.java.com (2013).

[25] Oracle. NetBeans: integrated development environment (IDE) https://netbeans.org/ (2013).

[26] Geronimo Oñativia. JPicUSB: *Java—PIC communication class.* http://sourceforge.net/projects/jpicusb/ (2009).

[27] Víctor M. Rico, Richard Aguirre. *Implementación de un sistema de simulación de la dinámica molecular basado en colisiones bidimensionales elásticas no relativistas de discos rígidos* (sin publicar). Disponible en: https://sites.google.com/site/cienciaalestilovmricob/software (2011).

[28] Stephen R. Schach. *Análisis y diseño orientado a objetos con UML y el proceso unificado.* Mc Graw Hill (2004).

[29] D. Shmilovitz. *On the definition of total harmonic distortion and its effect on measurement Interpretation.* IEEE Transactions on power delivery **20**, **1** (2005).

[30] Víctor M. Rico. *Construcción de formas de onda con el teorema de Fourier.* http://sites.google.com/site/cienciaalestilovmricob/tcircuitos (2009).

[31] RIGOL Technologies, Inc. DS1000E, DS1000D Series digital osciloscopes (2013)

[32] D. Baird, J. Castro. *Experimentación: una introducción a la teoría de mediciones y al diseño de experimentos. 2 edición.* Prentice-Hall Hispanoamericana (1991).

[33] A. Bohórquez G. *Introducción a la espectroscopía Mössbauer.* 2° Curso de espectroscopía Mössbauer. Principios básicos, Instrumentación y aplicaciones. Universidad del Valle (1996).

[34] V. P. Chechev, N. K. Kuzmenko. ^{57}Co Table de Radionucléides. Laboratoire National Henri Becquerel. http://www.nucleide.org/DDEP_WG/DDEPdata.htm (2001).

[35] RITVERC GmbH. Isotope products. www.ritverc.com (2013).

[36] E. Burckhardt, J. M. Echeverri. *Proyecto: paseo de la avenida Colombia.* Departamento Administrativo de Planeación Municipal de Santiago de Cali. http://planeacion.cali.gov.co/Publicaciones/Patrimonio%20Proyectos%20en%20Estudio/Proyecto%20Soterramiento%20Avenida%20Colombia/MEMORIA/Memoria%20nov%2017.pdf (2010).

[37] R. Aguirre. "Busca punto de foldeo", versión 4. Programa de VBA MS Excel para encontrar el punto de doblez en espectros Mössbauer (sin publicar) (2013).

[38] A.C. Larson and R.B. Von Dreele, "General Structure Analysis System (GSAS)", Los Alamos National Laboratory Report LAUR 86-748 (2000).

[39] B. H. Toby, EXPGUI, a graphical user interface for GSAS, J. Appl. Cryst. 34, 210-213 (2001).

[40] BK Precision. Function Generators Models 4017A & 4040A (Datasheet). http://www.bkprecision.com (2012).

[41] Unisource Corporation. 2 MHz Function Generator FG-8102 (Datasheet). http://www.tequipment.net/Unisource/FG-8102-OpenBox/ (2014).

[42] Protek. B8003FD Series DDS Function Generators (Datasheet). http://www.tequipment.net/ProtekB8003FD.html (2014).

[43] Rigol Technologies, Inc. DG1000 Series Dual-Channel Function/ Arbitrary Waveform Generator (Datasheet). www.rigol.com (2012).