UNIVERSIDAD DEL VALLE DE GUATEMALA

FACULTAD DE CIENCIAS Y HUMANIDADES
DEPARTAMENTO DE INGENIERÍA ELECTRONICA

Afinación automática de instrumento musical de cuerda por medio de control digital basado en análisis frecuencial

LUCAS A. PORTELLI

GUATEMALA
NOVIEMBRE, 2003

Afinación automática de instrumento musical de cuerda por medio de control digital basado en análisis frecuencial

UNIVERSIDAD DEL VALLE DE GUATEMALA

FACULTAD DE CIENCIAS Y HUMANIDADES
DEPARTAMENTO DE INGENIERÍA ELECTRONICA

Afinación automática de instrumento musical de cuerda por medio de control digital basado en análisis frecuencial

Trabajo presentado para optar al título de Ingeniero Electrónico en grado de Licenciatura

LUCAS A. PORTELLI

GUATEMALA
NOVIEMBRE, 2003

A mi familia:
Stella, Jorge y Nacho

Mi gloria es vivir tan libre
Como el pájaro del Cielo,
No hago nido en este suelo
Ande hay tanto que sufrir;
Y Naides me ha de seguir
Cuando yo remuento vuelo.

Soy gaucho, y entiéndanló
Como mi lengua lo explica,
Para mí la tierra es chica
Y pudiera ser mayor,
Ni la víbora me pica
Ni quema mi frente el Sol.

José Hemández, 1872. *El Gaucho Martín Fierro*

Vo. Bo. :

(f) _____
Dr. Manuel López
Asesor

Tribunal:

(f) _____
Dr. Manuel López
Asesor

(f) _____
Ing. Julio Vásquez

(f) _____
Ing. Gonzalo Palarea

Fecha de aprobación: 15 de enero de 2004

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

RESUMEN

Para la creación de esta aplicación se siguió un orden lógico en la construcción e investigación.

Para la demostración práctica del sistema de afinación se utilizó una guitarra bajo de cuatro cuerdas tipo "precision" el cual es un estilo ampliamente utilizado en el medio que caracteriza.

El primer paso fue el del diseño y construcción del Hardware de excitación. Se prestó especial cuidado en la durabilidad y consistencia del mismo, debido a que éste formaría la parte mecánica más utilizada tanto para la investigación del fenómeno de vibración de las cuerdas, como para la aplicación final. Este sistema se diseñó de tal manera que tanto los factores ambientales tales como cambios de temperatura o humedad no tuvieran mayor efecto en el desempeño del mismo. El sistema también fue diseñado de manera que sus partes móviles tuvieran el menor desgaste posible, y en el caso en el que el desgaste se torne inminente, el diseño permitirá el fácil reemplazo de esta pieza en particular.

Para el sistema de excitación también fue necesaria la creación de un control digital, el cual se vale de detectores ópticos para controlar la posición de los excitadores, de manera que no interfieran con la vibración de las cuerdas. Con este aditamento, el sistema de excitación se volvió independiente de las demás aplicaciones, es decir, la PC únicamente se limita a mandar comandos de cuándo excitar una cuerda, mientras que el control de los excitadores queda limitado al sistema de excitación exclusivamente.

Una vez diseñado el sistema de excitación, se procedió a la realización de un estudio para determinar el método a utilizar para procesar las señales provenientes del instrumento de cuerda. Esta tarea constituyó la utilización de técnicas de ventaneo, normalización y cálculo del espectro frecuencial. También fue necesaria la utilización de librerías de Windows las cuales permiten tener acceso a hardware como la tarjeta de sonido, reloj del sistema, etc.

El método determinado consta de dos partes principales que son: acercamiento por frecuencia principal y aproximación final por armónicas.

La combinación de ambos métodos teóricamente lleva a una afinación para la cual el oído humano no es capaz de percibir el error remanente con respecto a una afinación matemáticamente prefecta.

Luego, se diseñó un control digital unificado en Visual Basic 6.0, el cual se encarga de la adquisición, procesamiento y despliegue de datos y del control de actuadores y excitadores. Este control digital también cuenta con una interfaz gráfica para el usuario la cual es simple y requiere de poco o ningún entrenamiento previo del individuo que haga uso de ella.

Finalmente, se diseñaron actuadores mecánicos con servo-motores modificados para rotación continua. Estos son controlados desde la PC, y permiten la corrección de la tensión de las cuerdas.

El resultado principal de este trabajo fue la aplicación automática capaz de afinar una "guitarra bajo" hasta el punto en el que el oído humano no percibe la diferencia de frecuencia con respecto a una afinación matemáticamente perfecta. Por lo tanto, la precisión de la afinación resultante es, para el oído humano, de un 100%.

I. INTRODUCCIÓN

La automatización de un proceso conlleva estudios preliminares acerca de temas que describen fenómenos que ocurren en el mismo. Para la creación de una aplicación, que afine un instrumento musical de cuerda automáticamente, es necesario estudiar el servosistema que define a la acción de afinar tradicional, y luego, se deben reemplazar las partes del mismo para incluir algún tipo de trabajo que sea realizado por el ser humano.

Debido a la naturaleza de la comparación de parámetros que realiza el ser humano en la afinación de un instrumento musical, se vislumbra la posibilidad de la emulación de este procedimiento. Como consecuencia de esta automatización, se encontrarán relaciones y modelos matemáticos, que servirán para optimizar el proceso. Esta optimización conducirá a un procedimiento y a un sistema aún más simple que los originales, es decir, que constará de menos elementos y etapas, y que será llevado a cabo en un intervalo de tiempo menor que el mismo procedimiento realizado por el ser humano.

En el capítulo titulado "Hardware" se determinarán los elementos que servirán para suplantar al ser humano en su expresión mecánica y física, en las partes del proceso en las que se requiera. Se expondrán los métodos utilizados para determinar los tipos de actuadores, sensores, y la configuración de los mismos con respecto al instrumento. Se presentará la arquitectura digital necesaria para controlar la parte mecánica y los algoritmos de control autónomo de esta parte. El hecho de tener algoritmos de control ajenos al programa principal de afinación, vuelve a la parte mecánica autosuficiente con respecto a rutinas de seguridad e inicialización de la aplicación.

En el capítulo titulado "Análisis frecuencial y parámetros del control digital", se presentará el estudio realizado para determinar todos los factores y métodos necesarios para llevar a cabo la afinación de un instrumento musical de cuerda. Debe prestarse especial atención al método de afinación Portelli, el cual utiliza las armónicas generadas por el instrumento musical de cuerda, para llevar a cabo la determinación del factor de corrección en frecuencia. Esté método asegura una afinación "perfecta", ya que se espera que la frecuencia principal que presente el instrumento, tenga un error menor al error perceptible por el oído humano. Cabe mencionar que en el método de afinación Portelli, se utilizan recursos que son inferiores a los necesarios si se llevara a cabo una afinación tradicional.

En el último capítulo "Servosistema" se muestra la manera en la que se unifica todo lo desarrollado en los capítulos anteriores, para finalmente llegar a la aplicación terminada. Se definen las bases para el algoritmo de control general y se tocan temas tales como seguridad y autosuficiencia de la aplicación.

En los Anexos se puede encontrar el código utilizado para el control digital de la aplicación terminada. También se muestra información adicional que fue utilizada para el desarrollo de la aplicación final.

II. ANTECEDENTES

La afinación de un instrumento musical de cuerda se lleva a cabo por medio de la variación de la tensión de las cuerdas, parámetro ligado directamente a la frecuencia fundamental de oscilación de las mismas. Dicha tensión es ajustada por medios mecánicos tales como clavijas y engranajes. Sobre estas simples máquinas, la cuerda se enrosca o desenrosca variando consecuentemente la tensión que presenta y así la frecuencia fundamental de oscilación.

El método tradicional de afinación consiste en la comparación entre la frecuencia fundamental de oscilación de un diapasón y la frecuencia actual de la cuerda. Un diapasón consiste en una vara bifurcada de metal, la cual al ser golpeada, vibra con una frecuencia fundamental característica. Esta frecuencia característica no es función de la fuerza con que se aplica este golpe, siempre y cuando esta fuerza se encuentre dentro de un rango tal permitido por el diseñador de dicho instrumento.

La comparación entre las dos frecuencias es realizada por la persona que desea afinar el instrumento, para luego actuar sobre éste, enroscando o desenroscando la cuerda según se requiera. La destreza auditiva del individuo que realiza esta tarea se vuelve un factor determinante del éxito de este método.

Actualmente, existen dispositivos electrónicos que son capaces de realizar dicha comparación frecuencial internamente, para luego indicar al usuario si éste debe aumentar o disminuir la tensión en la cuerda. La comparación se lleva a cabo entre un patrón interno definido y la señal de entrada, proveniente del instrumento musical.

En ambos casos, la afinación tradicional y la afinación asistida por dispositivos electrónicos de comparación, el individuo juega el papel de controlador y actuador del *Servosistema*$_{\varphi}$ [φ] (figura 1)

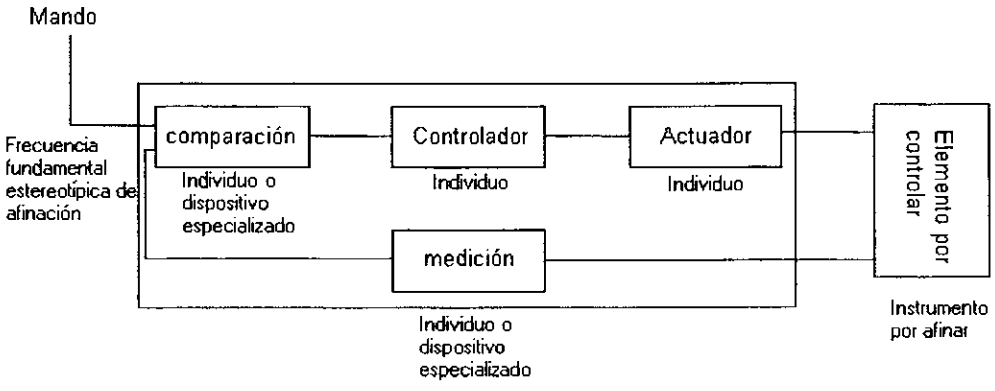


Figura 1: Servosistema formado en el proceso de afinación de un instrumento musical de cuerda.

[φ] Los servosistemas son controles de posición que generan una señal de colocación a raíz de una señal retroalimentada y comparada con el mando.

En este tipo de procesos, el individuo lleva a cabo una tarea simple pero repetitiva, la cual, dada la naturaleza del proceso, puede ser llevada a cabo por algún dispositivo automático. Por esto, se vislumbra la posibilidad de la automatización del proceso de afinación del instrumento musical de cuerda.

Para esto, se debe crear un dispositivo tal que sea capaz de excitar el instrumento, realizar una comparación interna con un estereotipo definido de frecuencias y que basándose en esto, sea capaz de modificar la tensión de la cuerda como ésta lo requiera. De esta manera, el usuario interactúa en el proceso para definir la afinación que debe ser alcanzada por la aplicación automática, convirtiéndose únicamente en un espectador del proceso.

III. OBJETIVOS

Se toma como el objetivo principal, la implementación de una aplicación "automática" la cual sea capaz de afinar una guitarra eléctrica. El término automática se refiere a que la interacción del usuario con la aplicación se limite a colocar una señal de mando, es decir, la afinación deseada, para que el dispositivo actúe sobre el instrumento de manera que sea alcanzada dicha afinación.

La implementación de dicha aplicación se basará en un control digital. Este control digital debe ser implementado a base de un algoritmo que sea capaz de realizar el análisis necesario en el régimen de la frecuencia de la respuesta del instrumento musical a una excitación propiciada por un actuador mecánico.

A. Objetivo General:

- Implementar un dispositivo tal, que mediante un control digital y actuadores mecánicos, sea capaz de medir, clasificar y corregir la afinación de una guitarra eléctrica.

B. Objetivos específicos:

- Implementar un dispositivo, que pueda ser accionado electrónicamente, el cual sea capaz de excitar invariablemente el instrumento a afinar.

- Crear un algoritmo que realice un análisis al régimen frecuencial de la respuesta del instrumento musical a la excitación propiciada por el dispositivo que se describió en el inciso anterior. Dicho análisis tiene por objeto detectar, clasificar y corregir, por medio de actuadores mecánicos, la afinación del instrumento de manera que se alcance la afinación deseada por el usuario (mando).

- Implementar los actuadores mecánicos mencionados en el inciso anterior.

- Implementar una interfaz que permita al usuario colocar la afinación deseada y que despliegue información referente al estado del proceso.

IV. Hardware

A. Diseño del sistema actuador para una guitarra bajo de cuatro cuerdas tipo "Precision"

1. **Determinación del torque necesario en cada actuador.** La necesidad de la determinación del torque radica en tener una especificación concreta la cual sirva para elegir una actuador adecuado. La relación que describe al torque es:

$$\overline{T} = \overline{F} \bullet \overline{d}$$

donde $\overline{F}$ es el vector que describe la fuerza y $\overline{d}$ la distancia según se muestra en la figura 2

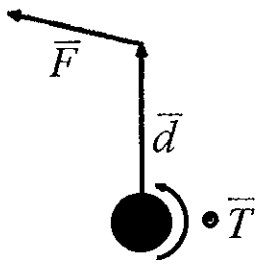


Figura 2: Diagrama de torque.

Para realizar la medición se utilizó una masa conocida (120g) y una vara de plástico de poca flexibilidad (la cual tenía un peso despreciable) de 30cm de largo. Dichos elementos fueron configurados en una máquina simple descrita por la figura 3.

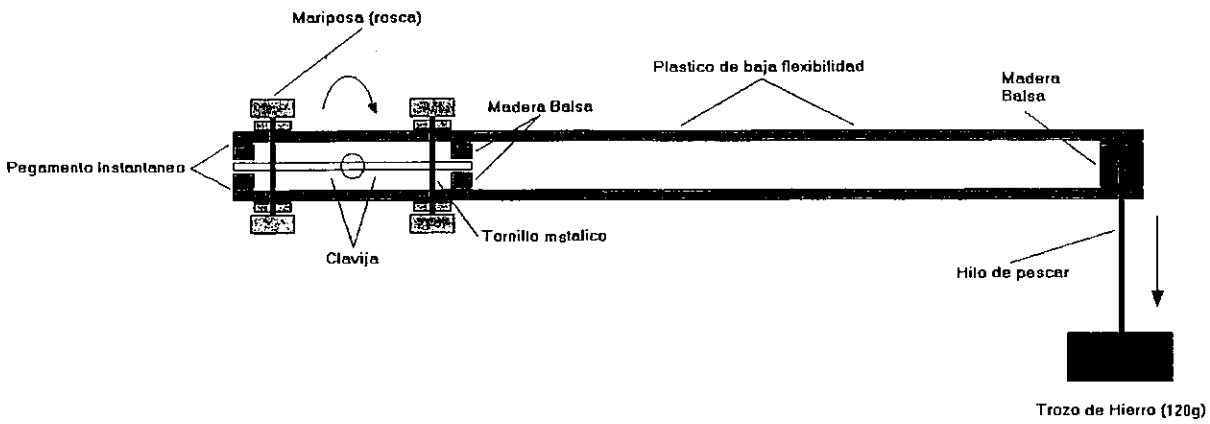


Figura 3: Máquina simple utilizada en la determinación del torque necesario para actuar sobre la cuarta cuerda de guitarra bajo tipo "Precision".

El análisis se basó en la cuerda cuatro debido a que es la que presenta mayor tensión, y por lo tanto mayor torque para modificar la misma. El valor obtenido corresponderá al mayor torque necesario para actuar sobre la afinación del instrumento.

Entonces, el torque aplicado a la clavija cuando la vara está perpendicular al suelo y en estado estacionario es

$$\overline{T} = \overline{F} \bullet \overline{d} = (0.120Kg)(30cm) = 3.60Kg \bullet cm$$

La metodología para obtener el torque del actuador consiste en soltar la masa cuando se encuentra como se observa en la figura 4.

El torque encontrado a través de esta medición contemplará el hecho de que el actuador tenga que mover una carga desde su estado estacionario hasta cierta velocidad constante. Es decir, el torque medido será suficiente para obtener una aceleración angular tal que produzca el movimiento de la clavija desde su estado estacionario.

Esto es debido a que el torque decrece a medida que la clavija rota, puesto que el ángulo entre la fuerza y el vector de la distancia aumenta. Entonces, el torque medido en el estado estacionario es el torque máximo que se pueda medir durante todo el proceso. Véase figura 4

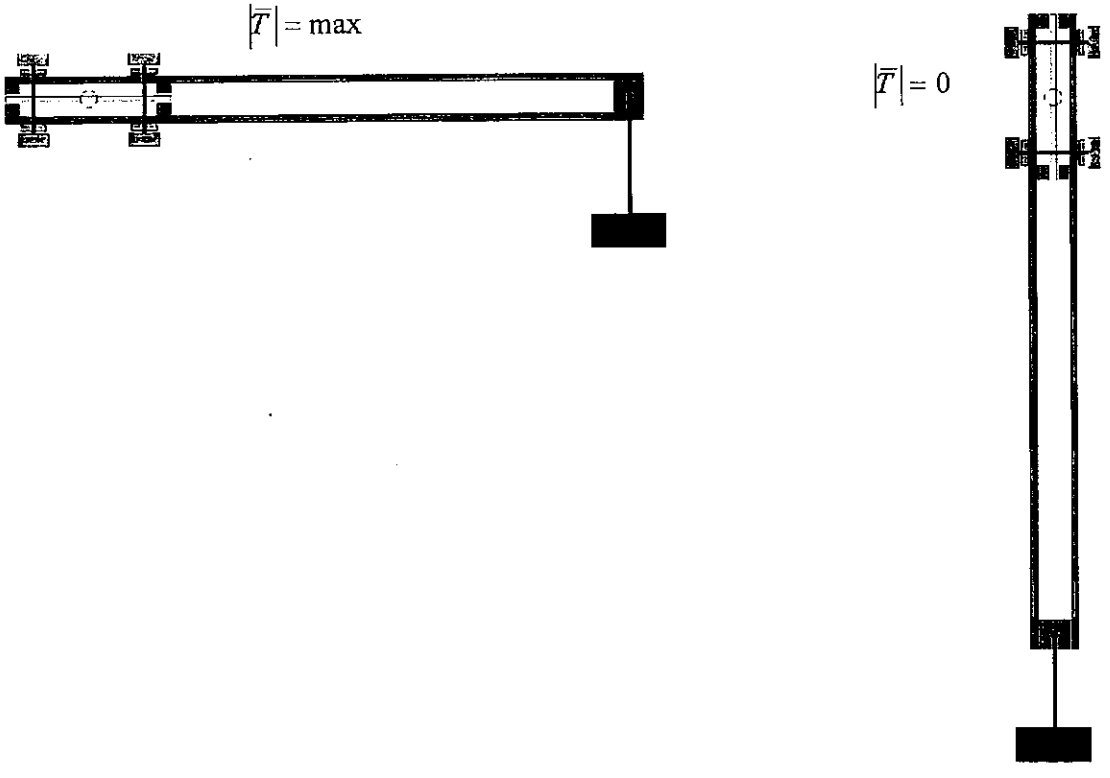


Figura 4: Metodología utilizada en la determinación del máximo torque necesario para actuar sobre la afinación de una guitarra bajo tipo "Precision".

Entonces, el actuador a utilizar debe tener un torque superior a los $3.60 Kg \bullet cm$

2. **Determinación del tipo de actuador a utilizar.** El actuador a utilizar debe tener las siguientes características:

- Torque superior al especificado por el experimento anterior
- Capaz de controlar la distancia angular recorrida

Para ello se utilizó un Servo-Motor modificado para rotación continua el cual cumple con los dos requisitos (ver anexo C). Las especificaciones del actuador se muestran en la tabla 1

Tipo de Actuador	Servo-Motor
Tamaño (mm)	$63.0X32.0X61.6$
Peso (g)	136
Velocidad (°/s)	214.28 ~(4.8V)
Torque (Kg•cm)	13.00 ~(4.8V)
Voltaje nominal (V)	4.8~6.0

Tabla 1: Especificaciones del servo-motor seleccionado.

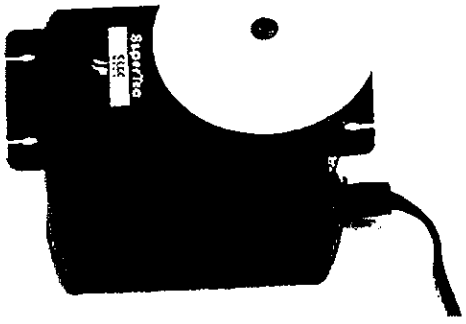


Figura 5: Servo-motor utilizado como actuador.

La utilización de este actuador, implica un valor de torque el cual triplica el mínimo necesario. Esto no mostró ningún efecto contraproducente en el desempeño del sistema de afinación.

3. **Interfaz entre la clavija y el actuador.** Estos elementos están compuestos de dos materiales diferentes. La clavija típicamente está fabricada de hierro cromado, y el actuador de plástico. Además, la clavija no es un elemento fabricado para una rotación continua, por ello se espera que su eje no esté centrado.

Debido a estos inconvenientes es probable que haya daños al plástico que compone el actuador si se une directamente con la clavija de la guitarra bajo. Es de esperar que en el peor de los casos, la clavija no rote cuando se active el actuador.

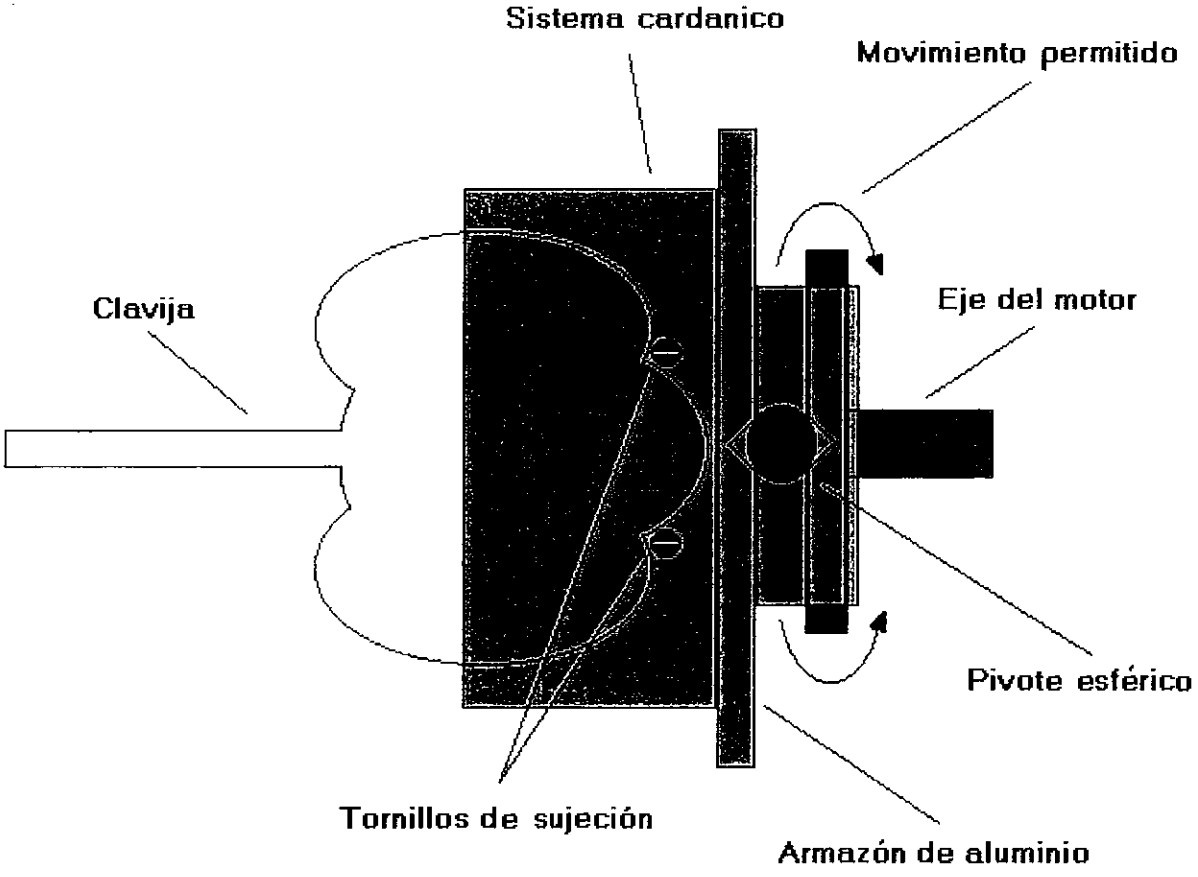


Figura 6: Interfaz entre el actuador y la clavija.

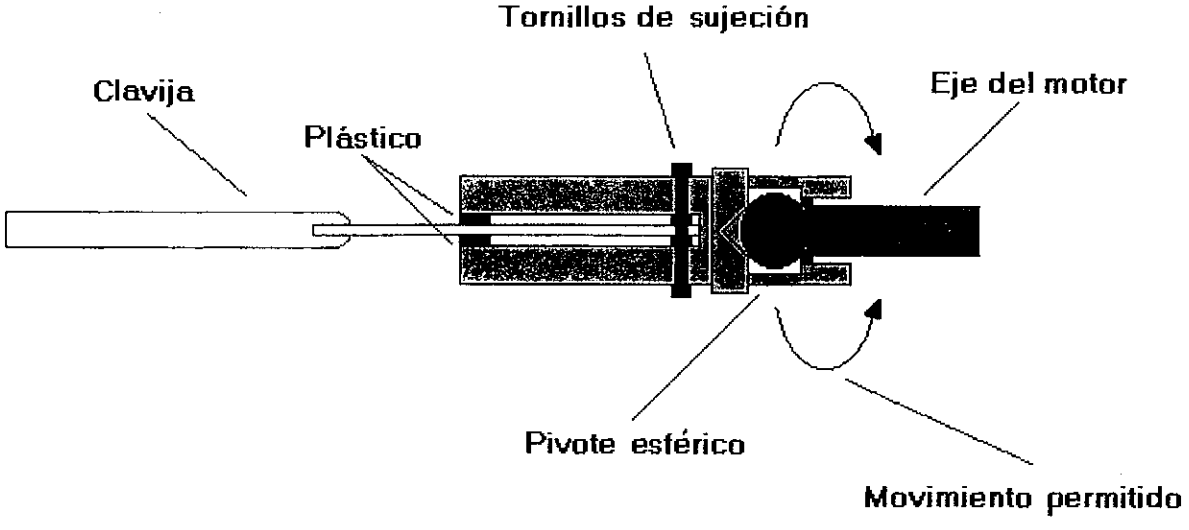


Figura 7: Interfaz entre el actuador y la clavija (vista lateral).

Es necesario la utilización de una interfaz la cual sea capaz de contener estas posibles anomalías en la construcción de la guitarra bajo, de manera que no induzcan fuerzas en sentidos y planos para los que el actuador no está diseñado.

Por ello se atizará una interfaz basada en un eje cardanico, el cual permitirá dos grados de libertad, permitiendo que las fuerzas inducidas se transfieran directamente hacia el armazón del motor. Ver figuras 6 y 7.

Debido a que existirá movimiento en el eje cardanico, éste estará sujeto a la clavija de la guitarra fijamente por presión la cual será ejercida por dos tornillos de sujeción.

El eje cardanico está compuesto al fin por un pivote esférico el cual está fabricado de acero cromado. Se lubricará con grasa fina.

La unión del sistema cardanico con el eje del motor se hará mediante hule debido a que ésta debe ser una unión elástica, pero firme. Esto tiene la ventaja de poder renovar el hule periódicamente para mantener el grado de elasticidad deseado.

4. **Colocación de los actuadores.** Los actuadores deben ser colocados de manera que la única interacción que estos puedan tener con la clavija es de darle un movimiento rotacional. Es decir, que estos no deben estar fijos al marco de la guitarra bajo si es que ambos están fijos a un tercer elemento. Tomaremos el tercer elemento como el actuador contiguo. Entonces bastará con que los actuadores estén sujetos entre sí, sin ninguna fijación al cuerpo de la guitarra bajo. Vea la figura 7

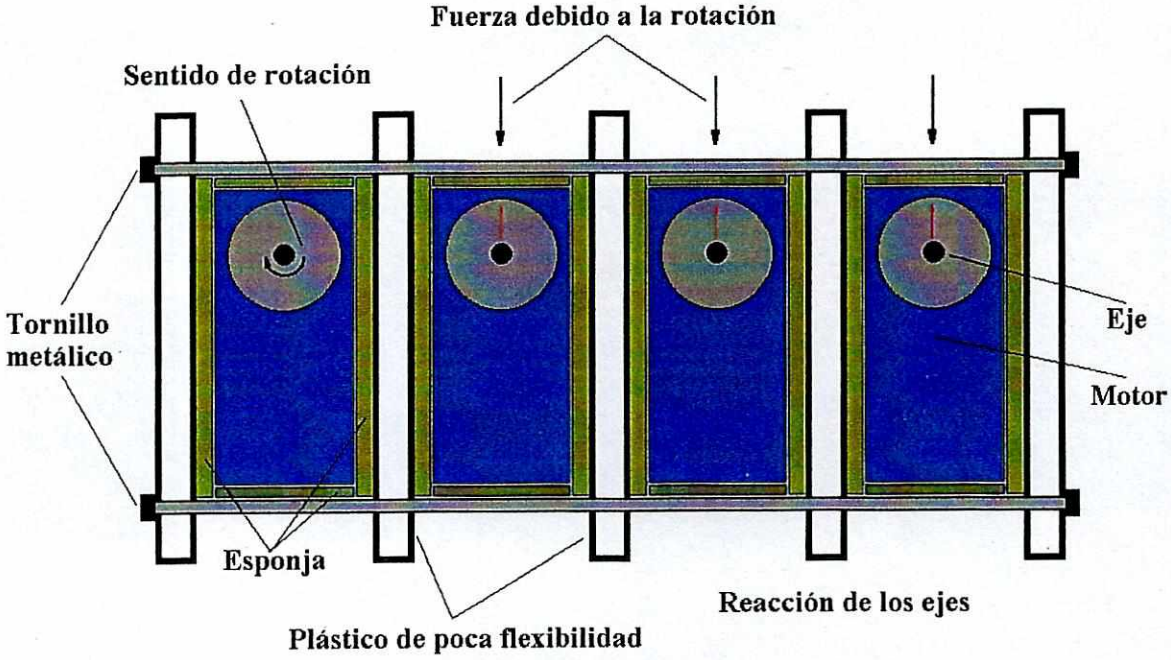


Figura 8: Sujeción de los actuadores.

Debido a que las imperfecciones del eje de la clavija serán transmitidas a la carcasa de cada motor, estos se tendrán movimientos alrededor de su posición inicial al activarse. Esto puede ser permitido, siempre y cuando no tengan movimiento rotacional. La sujeción de los motores entre sí debe estar entonces amortiguada, de manera que esto se cumpla.

Para esto se utilizó esponja como interfaz entre el plástico de poca flexibilidad y la carcasa de los motores. Este material logró cumplir con lo especificado, evitando el movimiento rotacional y a la vez permitiendo movimientos cortos en otros ejes.

B. Diseño del sistema de excitación para las cuerdas de una guitarra bajo de tipo "precision"

1. **Determinación de la estrategia de excitación de la cuerda.** Para excitar la cuerda de la guitarra bajo se realizó una emulación mecánica de una mano humana accionando una púa sobre la cuerda(una púa es un instrumento comúnmente utilizado para esta tarea, consta de un plástico delgado y flexible. ver figura 9).

Figura 9: Púa.

Esto se logró mediante un Servo-motor modificado para rotación continua (ver anexo C), el cual contaba con un instrumento en su eje, descrito en la figura 10.

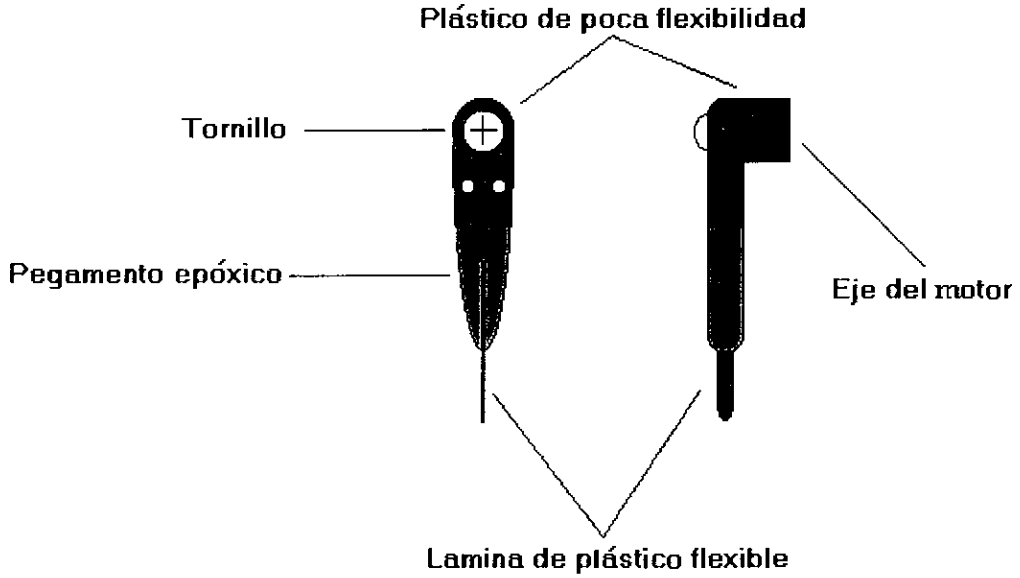


Figura 10: Instrumento utilizado para la excitación de la cuerda.

El funcionamiento de esta máquina simple consiste en llevar a la cuerda hasta un punto en el cual la fuerza ejercida a causa de la tensión de la misma vence la rigidez de la lámina de plástico flexible. En este punto, la lámina regresa a su posición original. Ver figura 11.

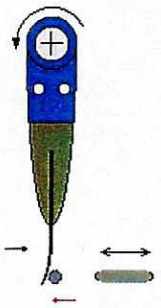


Figura 11: Descripción del funcionamiento del instrumento de excitación.

Debido a la flexibilidad del plástico, se utiliza una fracción mínima de torque de la que se utilizaría si la lámina fuera rígida. De hecho, si el plástico no cediera, existe la posibilidad de llegar al máximo de tensión para el que la cuerda esté diseñada, rompiéndola. Este sistema emula de manera satisfactoria la acción que realizaría una mano humana, causando en la cuerda, una excitación similar.

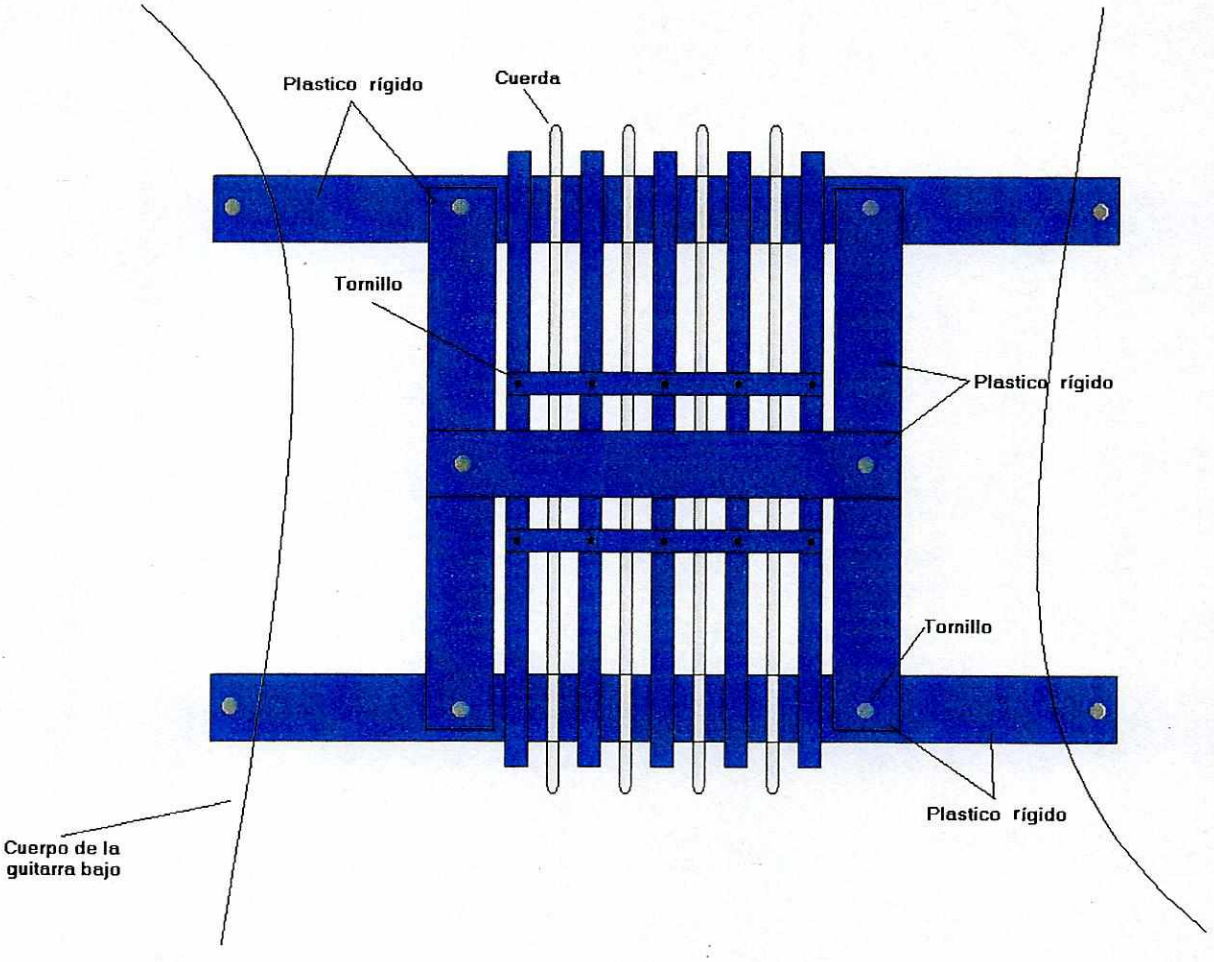


Figura 12: Estructura utilizada para el anclaje del sistema de excitación de las cuerdas.

Se utilizaron servo-motores estándar con un torque de 2.8 Kg/cm. Esta especificación se obtuvo mediante prueba y error. No se consideró necesario análisis previo debido a que el torque necesario es mucho menor al necesario en los actuadores.

2. **Configuración y ubicación de los motores excitadores.** Los motores excitadotes fueron ubicados sobre el cuerpo de la guitarra bajo. Esto es debido a que ésta es el área con mayor superficie del instrumento, lo que permite un anclaje firme. La estructura utilizada para esto se muestra en la figura 11

Se utilizó un plástico rígido el cual es típicamente usado para la construcción de modelos a escala. El diseño de esta pieza permite removerlo fácilmente de la guitarra bajo. Este no es un diseño invasivo ni destructivo, es decir, que la colocación u operación de este en el instrumento no conlleva ninguna modificación o daño al mismo. La configuración de los excitadores sobre esta estructura se muestra en la figura 13

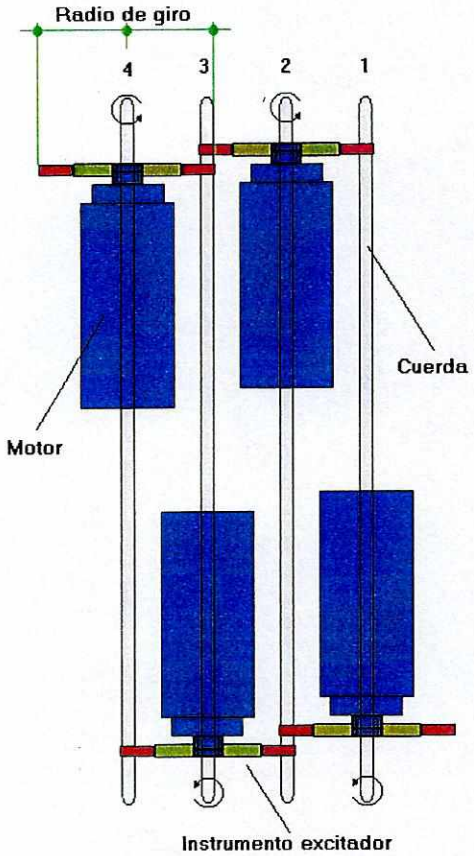


Figura 13: Configuración de los excitadores.

Nótese que los motores debieron ser colocados de manera que el radio de giro del instrumento de excitación no interfiriera con el motor contiguo. También debieron colocarse en diferente ubicación con respecto a la longitud de la cuerda debido a que los motores eran más anchos que la separación entre las cuerdas.

Estas diferencias de ubicación crean diversos efectos en las armónicas generadas por cada cuerda. Esto se tratará en la sección de análisis de la señal generada por una cuerda de guitarra bajo.

C. Control de los sistemas actuador y excitador.

1. Determinación del mando necesario para los servo-motores actuadores.

Una forma de utilizar los motores modificados para rotación continua es accionándolos por tiempos fijos. Estos recorrerán distancias angulares iguales si el tiempo de accionamiento y el voltaje de alimentación no cambia. El problema se reduce entonces a deducir el tiempo de accionamiento mínimo de los mismos.

Los actuadores deben ser capaces de tener un paso angular menor o igual que el mínimo cambio necesario en la afinación del instrumento. Este paso mínimo debe ser determinado mediante un análisis que relacione el cambio en la frecuencia normal con respecto al cambio en la posición angular en la clavija.

La distancia de cada cuerda que se enrosca sobre el engranaje conectado al tornillo sin fin de la guitarra bajo, es dependiente del radio de la misma. La frecuencia normal de oscilación que cada una presente dependerá de la tensión en cada cuerda, la cual, a su vez, depende de esta distancia. Por lo tanto, el cambio en la tensión no será el mismo para cuerdas de radio diferente a las cuales se les ha aplicado un mismo cambio en la posición angular de sus clavijas.

Esto indica que los análisis deben ser realizados para cada cuerda en particular. Los datos correspondientes a estos análisis se muestran en la tabla 2

	Cuerda 1			Cuerda 2		
Δ Grados	Frecuencia inicial	Frecuencia Final	Δ Frecuencia normal	Frecuencia inicial	Frecuencia Final	Δ Frecuencia normal
22.5	98.14-	-		64.94	65.1	0.16
45	98.14-	-		64.94	65.42	0.48
90	98.14	98.375	0.235	64.94	66.4	1.46
180	98.14	102.53	4.39	64.94	69.82	4.88
360	98.14	106.685	8.545	64.94	76.66	11.72
	Cuerda 3			Cuerda 4		
Δ Grados	Frecuencia inicial	Frecuencia Final	Δ Frecuencia normal	Frecuencia inicial	Frecuencia Final	Δ Frecuencia normal
22.5	51.75	52.5	0.75	41.2	41.6	0.4
45	51.75	53.71	1.96	41.2	43.3	2.1
90	51.75	55.17	3.42	41.2	46.32	5.12
180	51.75	61.03	9.28	41.2	51.92	10.72
360	51.75	66.87	15.12	41.2	58.2	17

Tabla 2: Relación de cambio en la frecuencia normal con respecto al cambio angular en la clavija.

Se utilizará un paso de 0.1Hz, por razones expuestas en la sección de análisis de la señal generada por una cuerda de guitarra bajo. Este paso mínimo debe ser logrado en la cuerda cuatro debido a que es la cuerda que presenta mayor cambio en la frecuencia normal por cambio en la posición angular.

Debido a que el ángulo en cuestión es pequeño, se puede asumir que la relación existente en la cuerda cuatro es aproximadamente:

$$\frac{0.4Hz}{22.5Grados} \cong 0.0177\,{}^{Hz}/_{Grado}$$

Entonces, el paso necesario debe ser de:

$$\left(0.1\frac{Hz}{Paso}\right)\left(\frac{1}{0.0177}\frac{Grados}{Hz}\right) \cong 5.625\frac{Grados}{Paso}$$

lo cual será redondeado a 5G/P, que significa que el Servo-motor debe ser capaz de realizar menos de :

$$\left(360\frac{G}{Vuelta}\right)\left(\frac{1}{5}\frac{P}{G}\right) = 72\frac{Pasos}{Vuelta}$$

2. **Implementación del paso para el servo motor actuador.** Para implementar un paso debe definirse primero un mando. El mando propuesto inicialmente fuel el de un pulso rectangular, tal como se muestra en la figura 14.

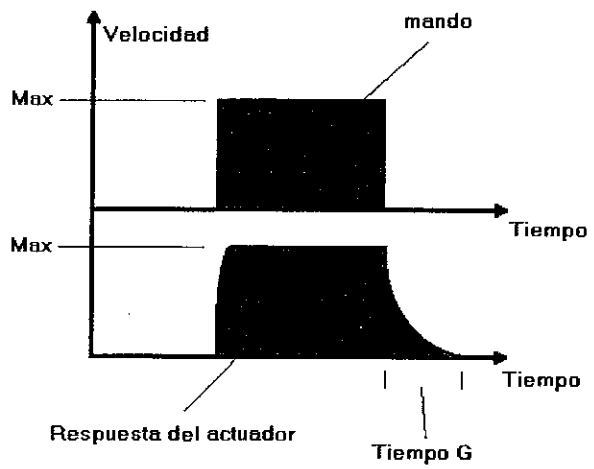


Figura 14: Pulso rectangular y respuesta del actuador.

Uno de los principales problemas fue el de la presencia de inercia en los servo-motores. Debido a la inercia, la distancia angular recorrida se volvía dependiente de la tensión de la cuerda (tiempo G). Es decir, que cuando que el tiempo G disminuye a medida que la tensión en la cuerda aumenta.

Para solucionar este problema, se modificó el mando de manera que la respuesta del actuador fuera lo más independiente posible de la tensión de la cuerda. El nuevo mando y su respuesta se muestran en la figura 15

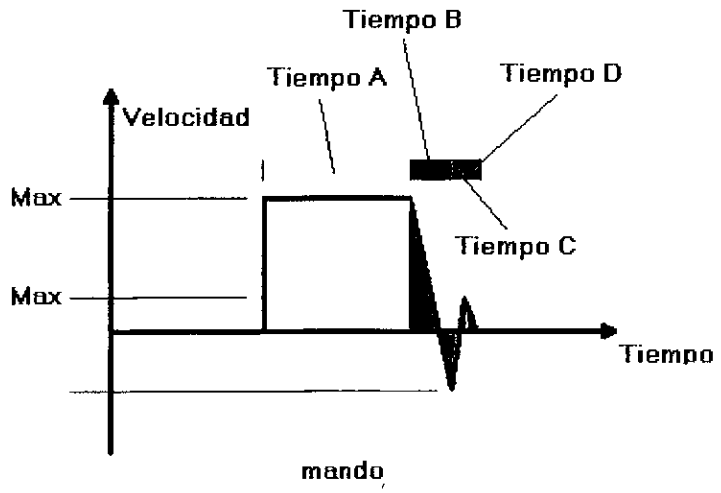


Figura 15: Mando modificado.

Finalmente, para implementar un paso de cinco grados se determinaron los tiempos A, B, C y D por prueba y error, manteniendo la relación que se muestra en la figura 15. Estos tiempos (listados en la tabla) corresponden a 70 pasos por vuelta, lo cual es aproximadamente cinco grados por paso.

Tiempo	Magnitud	Duración (ms)
A	100%	120
B	-20%	30
C	10%	30
D	0%	30

Tabla 3: Tiempos y magnitudes correspondientes al mando modificado.

La magnitud de la velocidad fue modificada variando el ciclo de trabajo del pulso de control del servo-motor entre 1ms y 2ms. 2ms corresponde a 100% y 1ms a –100%, siendo 1.5ms 0%.

3. Utilización de la función timeGetTime de la librería winmm.dll de Windows. Se utilizará esta función debido a que los controles de tiempo no proporcionan la exactitud necesaria, debido a que su periodo depende de la cantidad de tareas que el sistema operativo esté realizando en ese momento.

Esta función devuelve el tiempo del sistema con unidades en milisegundos. El tiempo del sistema corresponde al tiempo desde que se inicializó Windows.

Para la utilización de esta función en Visual Basic 6.0 debe declararse en una clase con el siguiente comando:

Private Declare Function timeGetTime Lib "Winmm.dll" () As Long

Cada llamada a esta función devolverá un número mayor que cero. Este número, para ser utilizable, debe referenciarse de alguna manera. Para esto se debe tener una variable la cual guarde una lectura inicial la cual servirá de referencia relativa a las mediciones futuras y así poder medir tiempos relativos.

4. **Detección de posición para los servo-motores excitadores.** Debido que los servo-motores han sido modificados para rotación continua, no existe una retroalimentación de su posición angular y, por lo tanto, ésta no es una variable controlable. Debido a esto, podrían interferir con la oscilación de las cuerdas, causando lecturas y acciones incorrectas por parte del servosistema afinador.

La solución implementada consiste en la utilización de detectores ópticos para el acuse de la ubicación del instrumento excitador. Dichos detectores fueron colocados de manera que detectaran el paso del instrumento excitador cuando éste estuviera a 180 grados de la cuerda. La configuración se muestra en la figura 16

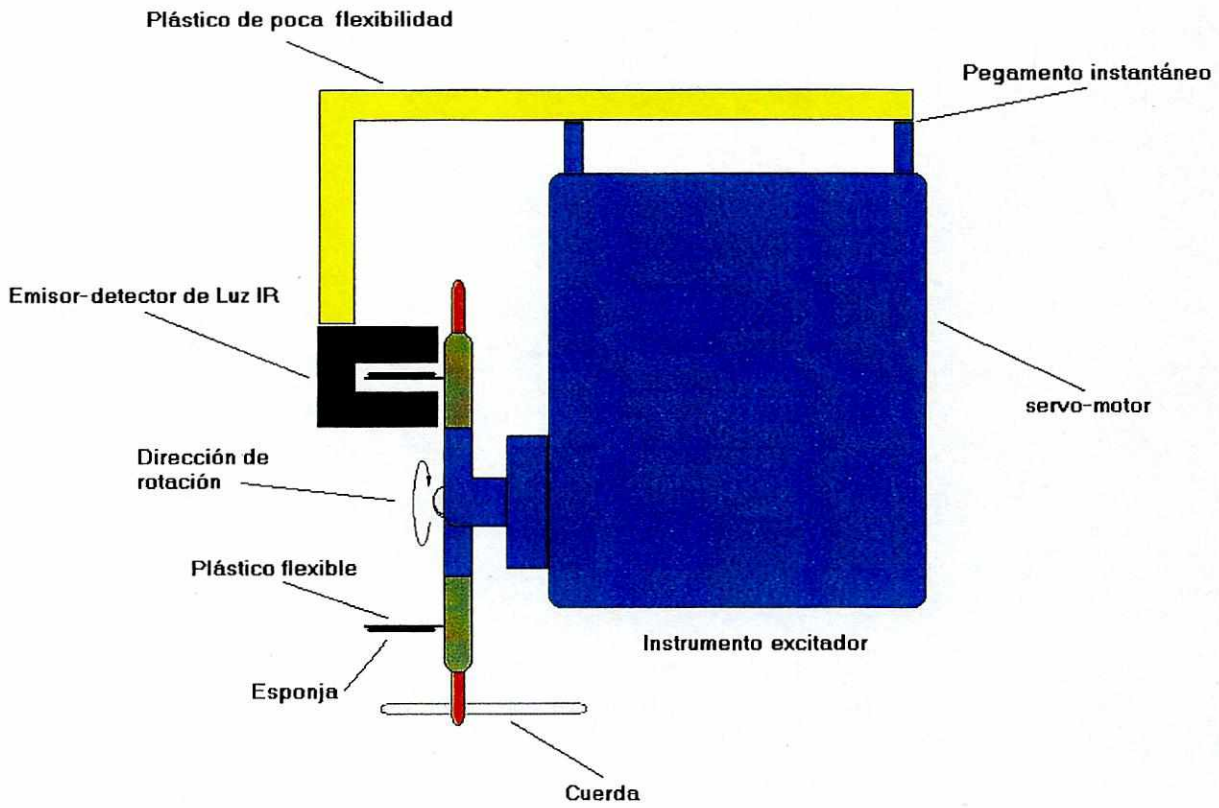


Figura 16: Configuración del detector óptico.

Como detector-emisor óptico se utilizó el H21A1 fabricado por Fairchild (Ver anexo E). El circuito equivalente se muestra en la figura 17.

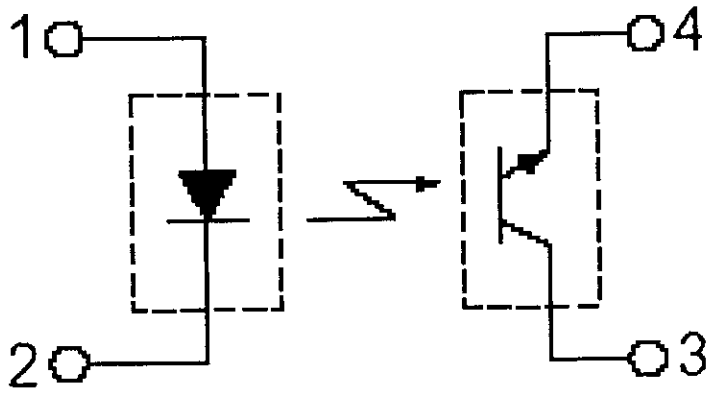


Figura 17: Circuito equivalente del HA21A1.

Los valores adecuados correspondientes a las resistencias para el led y para el colector del transistor deben ser determinados de manera que este se sature y funcione únicamente como un switch. Para esto, se sabe del Anexo E que el valor máximo de corriente en el diodo es de 50ma, por ello se elige que la corriente en el diodo sea de 25mA. Para permitir que esta corriente fluya por el diodo (sabiendo que la caída de voltaje en el mismo será de alrededor de 1.7V y que la alimentación será de 5V), la resistencia en el ánodo del diodo debe tener una magnitud de

$$\frac{(5V-1.7V)}{25mA} = 132\Omega \approx 130\Omega$$

Se sabe por el anexo E que el valor de la corriente de colector correspondiente a 25mA en el led es de 1.5mA (en saturación). Se utilizará 1mA de corriente de colector, lo cual garantiza que el transistor este saturado, y para ello la magnitud de la resistencia de colector debe ser de al menos:

$$\frac{(5V)}{1mA} = 5000\Omega \approx 5.1k\Omega$$

Estos valores fueron aproximados a resistencias comerciales.

5. **Arquitectura para el control de los actuadores y excitadores.** Se utilizó un módulo de generación de PWM para accionar los actuadores y excitadotes (Ver anexo F). También se utilizó un microcontrolador el cual decodifica las señales provenientes de los 4 detectores ópticos. El objetivo principal del microcontrolador es el de mantener a los motores excitadores en posición, e inicializar el módulo de generación de PWM. Véase el esquema en la figura 18.

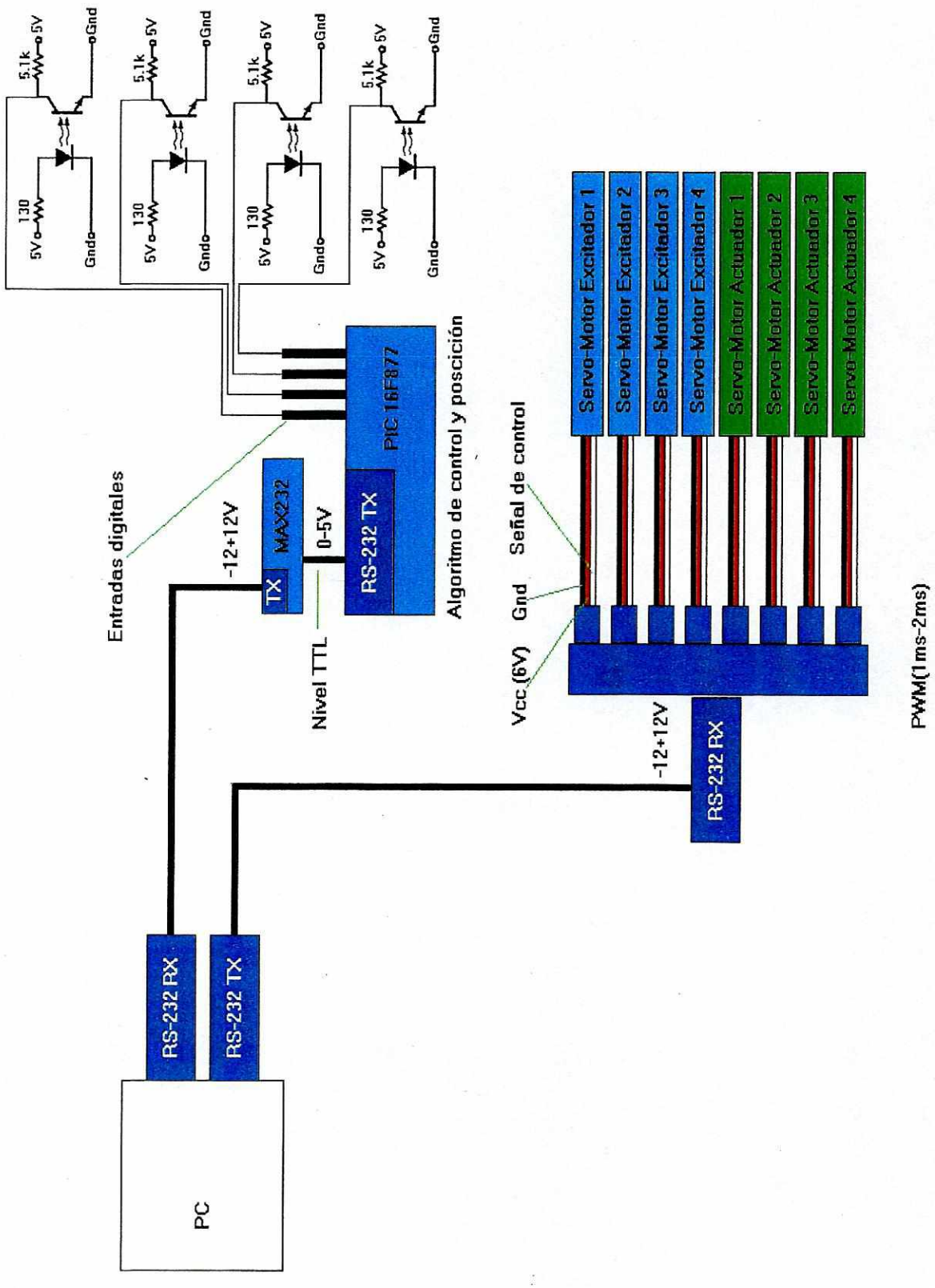


Figura 18: Arquitectura para el control de los actuadores y excitadores.

Las especificaciones de la conexión serial se muestran en la tabla 4

Puerto utilizado	1
Baud Rate	19200
Handshaking	No
Bits de datos	8
Paridad	No
Stop Bits	1

Tabla 4: Especificaciones de la conexión serial.

Para que sea posible que el microcontrolador pueda mandar comandos hacia el módulo de PWM, existe un algoritmo en la PC el cual reenvía los datos provenientes del mismo. Este algoritmo se basa en la detección de un evento generado por el puerto serial. Una de las razones por la que ocurre este evento es por la recepción de un Byte completo. Por ello, cuando se detecta un evento, y se corrobora que corresponde a la recepción de un Byte completo, el algoritmo lo reenvía automáticamente.

El módulo también puede recibir comandos de la PC. La PC tiene la tarea de manejar los actuadores.

No se esperan colisiones en le puerto serial, debido a razones que se explican en el proceso de afinación.

6. Inicialización de los actuadores y excitadores. Antes de que el módulo de PWM pueda empezar a recibir comandos que tengan que ver con el proceso de afinación, todos los servo-motores deben estar en una posición inicial predeterminada.

En el caso de los motores excitadores, estos deben estar a 180 grados de su cuerda objetivo. Las razones son explicadas en el Proceso de afinación. En el caso de los actuadores, estos deben estar apagados para evitar lesiones permanentes al instrumento.

Para lograr esto, el microcontrolador lleva a cabo un análisis de las señales obtenidas en las entradas digitales conectadas a los sensores. El algoritmo básico de inicialización se muestra en la figura 19

Nótese que una vez realizadas las ubicaciones de los cuatro excitadores, existe una doble verificación de su posición. Esta doble garantía no es absolutamente necesaria; sin embargo, toma en cuenta errores que se puedan dar en la comunicación serial por diversas causas.

Fue necesaria la implementación de una rutina específica para detener los servo-motores excitadores debido a que estos también presentaban inercia en la rotación. Esta inercia tiene el mismo efecto que se muestra en la figura 14, con la diferencia que el tiempo G no es variable. Se diseñó entonces, experimentalmente (por prueba y error), un mando para el cual la respuesta del motor en cuanto a posición angular fue lo más exacta posible. La función que describe este mando se muestra en la figura 20.

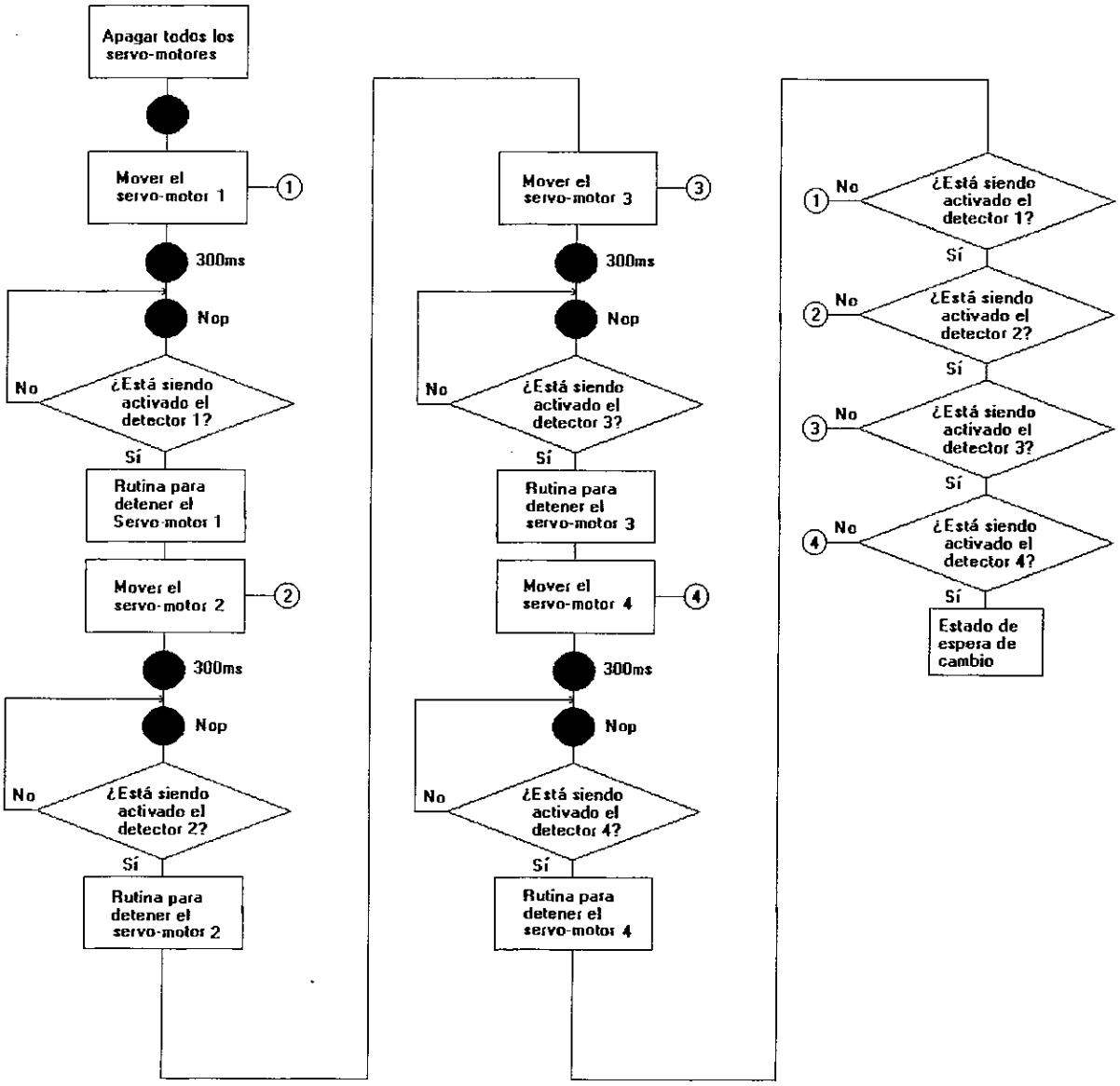


Figura 19: Algoritmo de inicialización.

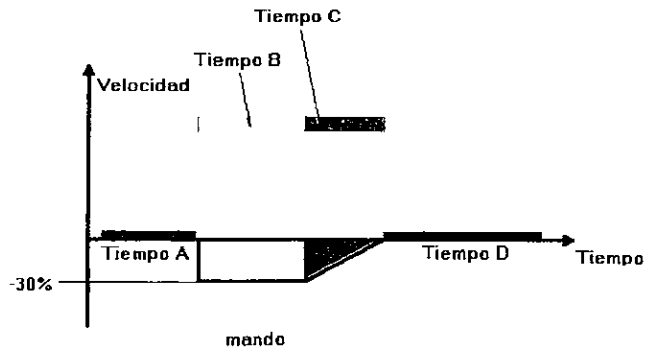


Figura 20: Mando de corrección para los servo-motores excitadores.

Con respecto al tiempo A, éste fue específicamente calculado para actuar en conjunto con el mando, de manera que en este caso hubo un sólo cambio en la dirección antes de apagar el tren de impulsos de control.

El tiempo D fue colocado de manera que si se llega a dar más de una llamada consecutiva a la función de detener el motor, ésta actúe como una corrección en sentido contrario de la rotación habitual. Este punto se ilustrará mejor luego de describir el algoritmo de estado de espera de cambio. Este se muestra en la figura 21

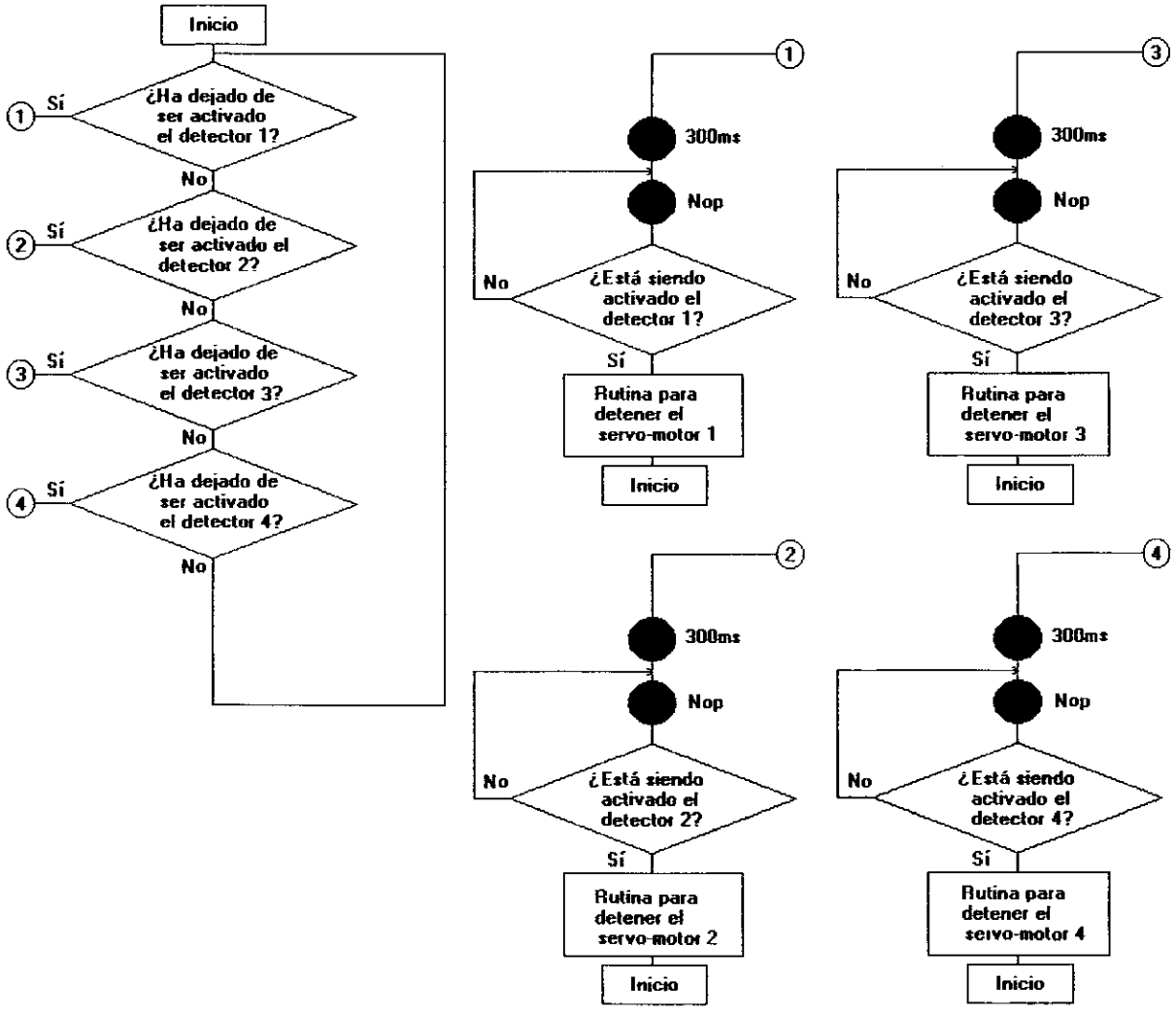


Figura 21: Algoritmo de estado de espera de cambio.

Este algoritmo hace un monitoreo de las entradas digitales que están conectadas a los sensores. Cuando detecta que algún instrumento de excitación no está en su posición inicial, ingresa en una rutina para detectar el regreso este a dicho estado. Los módulos de tiempo de 300ms retrasan la entrada a esta rutina, garantizando así que el instrumento excitador ha salido por completo del alcance del detector y así evitar errores de interpretación. Al detectar el regreso del instrumento de excitación a la posición inicial, realiza la rutina para parar el servo-motor.

Esta rutina tiene una doble garantía, al igual que la de inicialización. Esta consiste en que la rutina para parar el motor es llamada varias veces consecutivamente hasta que el detector sea activado. Típicamente, esta rutina debería ser llamada una sola vez, pero en caso de algún tipo de error en la comunicación serial, o algún problema en los sensores (es decir que estos no funcionasen por alguna razón), la rutina sería llamada consecutivamente.

El efecto de llamar esta rutina consecutivamente genera pasos pequeños (~10 grados) en el sentido inverso de rotación, y de esta manera el error que haya existido será corregido automáticamente por esta parte del hardware, llevando el instrumento de excitación a su posición inicial. En el caso de un mal funcionamiento en los detectores, el algoritmo en el microcontrolador será incapaz de detectar esta condición, pero un observador diagnosticará el problema inmediatamente debido a este síntoma característico. La tabla muestra los valores específicos de estos tiempos.

Tiempo	Magnitud	Duración (ms)
A	Apagado	60
B	-30%	30
C	0%	20
D	Apagado	300

Tabla 5: Especificaciones del mando de corrección para los servo-motores excitadores.

7. **Fuentes de voltaje.** Se utilizaron tres fuentes de voltaje con una alimentación común. La primera consta de un regulador de 5V del cual se obtiene la alimentación para todos los emisores-detectores, el microcontrolador, el reloj del microcontrolador y el led indicador del estado de la inicialización. La segunda consta de un regulador de 8V para la alimentación del control intemo del modulo generador de PWM. La tercera consta de un regulador de 6V para todos los servo-motores. La capacidad necesaria para cada regulador se calculó sumando todas las corriente que demandaba cada circuito. La tabla 6 muestra las especificaciones de cada regulador. La figura 22 muestra el diagrama del circuito de alimentación.

Marca	Voltaje	Corriente máxima
ANK7808c	5V	1A
HA17805	8V(variable)	1A
LM338K	6V(variable)	4A

Tabla 6: Especificación de los reguladores de voltaje utilizados.

Usualmente se recomienda un Ampere por cada servo-motor que esté conectado, para lo que se necesitaría un regulador de ocho Amperes. En la sección de proceso de afinación se explica el porque de la selección de un regulador de cuatro Amperes.

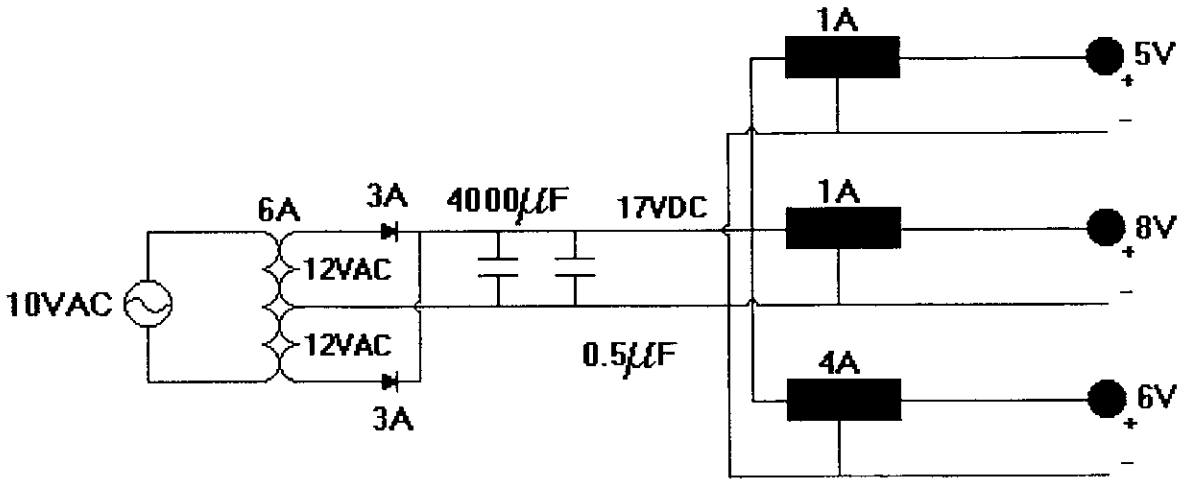


Figura 22: Circuito de alimentación.

V. Análisis frecuencial y parámetros del control digital

A. Determinación del comportamiento de una cuerda que vibra mediante el análisis de las soluciones de la ecuación diferencial parcial que lo describe

1. Ecuación de la cuerda que vibra.

$$\boxed{\frac{\partial^2 y}{\partial t^2} = a^2 \frac{\partial^2 y}{\partial x^2}} \qquad \text{ecuación (1)}$$

Esta ecuación es aplicable a pequeñas vibraciones transversales de una cuerda flexible tensa, tal como la cuerda de un instrumento música, ubicada inicialmente en el eje x y puesta en movimiento (ver figura 23). La función $y(x,t)$ es el desplazamiento de cualquier punto x de la cuerda en un tiempo t.

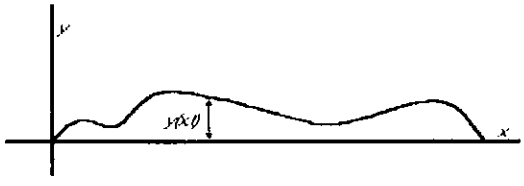


Figura 23: Cuerda con posición inicial descrita por f(x).

La constante $a^2 = \tau/\mu$, donde τ es la tensión (constante) en la cuerda y μ es la masa (constante) por unidad de longitud de la cuerda. Se supone que no actúan fuerzas externas sobre la cuerda y que esta vibra únicamente debido a su elasticidad.

2. Desarrollo de la solución de la ecuación para un caso específico.

a. Planteamiento. Se considera una cuerda de longitud L entre los puntos $(0,0)$ y $(L,0)$ en el eje x. En el tiempo $t = 0$ tiene una forma dada por $f(x)$, $0 < x < L$, y se suelta partiendo del reposo. Se busca una ecuación que describa el desplazamiento en un tiempo posterior.

Siendo la ecuación de la cuerda la descrita en la sección anterior, se consideran las restricciones que exige el problema:

$$0 < x < L \; ; \; t > 0$$

Lo que deriva en las expresiones:

$$y(0,t) = y(L,t) = 0 \ ; \ t > 0$$

Dado que la fórmula inicial de la cuerda está dada por $f(x)$,

$$y(x,0) = f(x) \ ; \ 0 < x < L$$

Dado que la velocidad inicial de la cuerda es cero,

$$y_t(x,0) = 0 \ ; \ 0 < x < L$$

Siendo éste un típico problema de valor límite, se procede definiendo una función

$$y = XT$$

Entonces, $XT'' = a^2 X''T$ o $T''/a^2T = X''/X$

Llamando la constante de separación $-\lambda^2$, se tiene

$$T'' + \lambda^2 a^2 T = 0 \qquad X'' + \lambda^2 X = 0$$

y $T = A_1 sen(\lambda at) + B_1 \cos(\lambda at)$ $X = A_2 sen(\lambda x) + B_2 \cos(\lambda x)$

Una solución es entonces dada por

$$y(x,t) = XT = (A_1 sen(\lambda at) + B_1 \cos(\lambda at))(A_2 sen(\lambda x) + B_2 \cos(\lambda x))$$

b. Aplicación de las condiciones iniciales. Aplicando la condición $y(0,t) = 0$, $A_2 = 0$. Entonces:

$$y(x,t) = B_2 sen(\lambda x)(A_1 sen(\lambda at) + B_1 \cos(\lambda at)) = sen(\lambda x)(A sen(\lambda at) + B \cos(\lambda at))$$

Aplicando la condición $y(L,t) = 0$, se tiene

$$sen(Lx)(A sen(\lambda at) + B \cos(\lambda at)) = 0$$

Lo que restringe $\lambda = m\pi/L$, puesto que $sen(Lx) = 0$

Como también se tiene que $\frac{\partial y(x,0)}{\partial t} = 0$ (velocidad inicial nula)

$$\frac{\partial y(x,t)}{\partial t} = sen(\lambda x)(A\lambda a \cos(\lambda at) - B\lambda a sen(\lambda at))\Big|_{t=0}$$

lo que implica que $A = 0$, y por lo tanto

$$y(x,t) = Bsen\left(\frac{m\pi x}{L}\right)\cos\left(\frac{m\pi at}{L}\right)$$

También es necesario satisface la condición $y(x,0) = f(x)$. Esto implica la superposición de las soluciones:

$$y(x,t) = \sum_{m=1}^{\infty} B_m sen\left(\frac{m\pi x}{L}\right)\cos\left(\frac{m\pi at}{L}\right)\Bigg|_{t=0}$$

Entonces: $$y(x,0) = f(x) = \sum_{m=1}^{\infty} B_m sen\left(\frac{m\pi x}{L}\right)$$

Resolviendo para la constante B_m por series de Fourier, se tiene

$$B_m = \frac{2}{L}\int_0^L f(x)sen\left(\frac{m\pi x}{L}\right)dx$$

Entonces, el resultado final es:

$$y(x,t) = \sum_{m=1}^{\infty}\left(\frac{2}{L}\int_0^L f(x)sen\left(\frac{m\pi x}{L}\right)dx\right)sen\left(\frac{m\pi x}{L}\right)\cos\left(\frac{m\pi at}{L}\right) \quad \text{ecuación (2)}$$

Lo cual corresponde a una solución del problema específico.

c. Correlación de la solución específica con la frecuencia de oscilación. Los términos en esta serie representan los módulos naturales o normales de vibración. La frecuencia f_m del módulo normal m se obtiene del término que contiene $\cos(m\pi at/L)$ y es dado por

$$2\pi f_m = \frac{m\pi a}{L} \qquad \text{o} \qquad f_m = \frac{ma}{2L} = \frac{m}{2L}\sqrt{\frac{\tau}{\mu}}$$

Las frecuencias presentes son entonces múltiplos enteros de la frecuencia más baja f_1. Esto implica que la cuerda deberá vibrar por segmentos, es decir por módulos normales, como se ve en la figura 24.

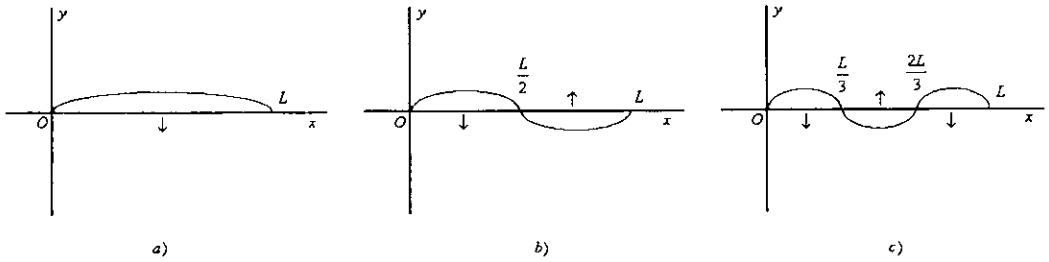


Figura 24: Modos de oscilación.

a) Primera harmónica (Modulo Fundamental)
b) Segunda harmónica (Primer Sobre-tono)
c) Tercera harmónica (segundo Sobre-tono)

Spiegel, M. 1976

La representación matemática de cada harmónica corresponde a un termino de los que integran la ecuación (2). La constante que se multiplica por cada uno de ellos es dependiente únicamente de la forma inicial $f(x)$ que presente la cuerda dado que la longitud L es invariante. Esta constante esta dada por

$$B_m = \frac{2}{L}\int_0^L f(x) sen\left(\frac{m\pi x}{L}\right) dx$$

Este resultado implica que ***una cuerda que vibra presentará un grupo infinito de frecuencias múltiplos enteros de la frecuencia mas baja*** f_1***, y que la amplitud relativa con la que se registre cada una de estas frecuencias, será dependiente únicamente de*** $f(x)$ ***, es decir, de la forma que la cuerda tenga en*** $t = 0$.

El hecho de tener un distinto énfasis armónico para cada forma inicial de la cuerda, se denomina timbre.

d. Consideraciones físicas adicionales. En este análisis en particular, no se tomaron en cuenta propiedades físicas de una cuerda real, tales como volumen, masa por unidad de volumen y fricciones varias que ocurran durante el movimiento. Tampoco se tomaron en cuenta ondas que viajen paralelamente a la longitud de la cuerda ni resonancia que esta pueda tener con el medio.

Se intentará probar que estas contribuciones físicas son despreciables para el objetivo de la investigación mediante el análisis practico, sin embargo, se espera tener mediciones a causa de estas contribuciones.

B. Convención musical de tonos y afinación

Actualmente existen dos convenciones de afinación. Las así llamadas Convención Americana de Afinación y la Convención Internacional de Afinación.

La Convención Americana de Afinación consiste en centrar la nota **A4** en 440Hz(la cual es la utilizada en este trabajo), mientras que en la Convención Internacional de Afinación, la nota **A4** corresponde a 435 Hz. Ambos corresponden a una "escala cromática igualmente templada". Matemáticamente esto significa que cada nota está relacionada a la nota previa por un factor que corresponde a la duodécima raíz de 2.

$$nota(n+1) = \left(\sqrt[12]{2}\right)[nota(n)]$$

Es decir, que el cociente entre las frecuencias de dos notas sucesivas es 1.05946309436. Para el caso específico de contar la décimo segunda frecuencia consecutiva, es decir

$$nota(n+12) = \left(\left(\sqrt[12]{2}\right)^{12}\right)[nota(n)]$$

La nota original de referencia se duplica. Este intervalo de doce notas intermedias, llamadas semitonos es llamado octava. Entonces los semitonos de una octava especifica se pueden encontrar multiplicando o dividiendo la nota original por la duodécima raíz de dos. El término "igualmente templada" se debe al hecho de la utilización de este factor de separación entre cada semi-tono. Existen otras agrupaciones de semitonos, tales como quintas o terceras. Las quintas corresponden a un cociente de 1.5 entre notas las cuales se encuentran separadas por siete pasos cromáticos. Para estas agrupaciones, el uso del factor de la duodécima raíz de dos no satisface exactamente la frecuencia esperada. Por ejemplo, G se encuentra siete pasos cromáticos sobre G, y el factor que las relaciona en ambas convenciones de afinación es

$$\left(\sqrt[12]{2}\right)^{7} = 1.49830707688$$

lo cual no está exactamente "templado"en 1.5. Por esto, la convención del uso de la duodécima raíz de dos es "igualmente templada" solo para las octavas.

Las doce notas musicales que comprenden una octava son C , $C^{\#}$, D , $D^{\#}$, E , F , $F^{\#}$, G , $G^{\#}$, A , $A^{\#}$ y B. La frecuencia de referencia es A_4, esto significa que es la **décima nota de la cuarta octava**.

La afinación de todos los instrumentos de cuerda convencionales se encuentra entre las primeras ocho octavas, es decir desde C_0 (16.35Hz) hasta B_8 (7902.13Hz). (véase el anexo B para la lista completa de las ocho primeras octavas y sus respectivas frecuencias)

1. **Percepción humana de cambios en la frecuencia audible.** Cada semitono a su vez se divide en céntimos, según la expresión

$$nota(n+1) = \left(\sqrt[1200]{2}\right)[nota(n)]$$

Es decir, cada semitono está compuesto de 100 céntimos. Para el oído humano, la diferencia mínima detectable se encuentra entre los 10 y 15 céntimos, dependiendo de factores específicos del individuo.

Se encuentra la relación del cambio mínimo detectable en frecuencias adyacentes, modificando la relación anterior así

$$f_2 - f_1 = \left(\sqrt[1200]{2}\right)f_1 - f_1$$

Luego, despejando $nota(n)$, se tiene

$$\Delta f = \left(\left(\sqrt[\frac{C}{1200}]{2}\right) - 1\right) f$$

La figura 25 muestra cómo para frecuencias mayores, la diferencia perceptible es cada vez mayor. Esto significa que el oído humano es más perceptible a cambios en frecuencias bajas que en frecuencias altas.

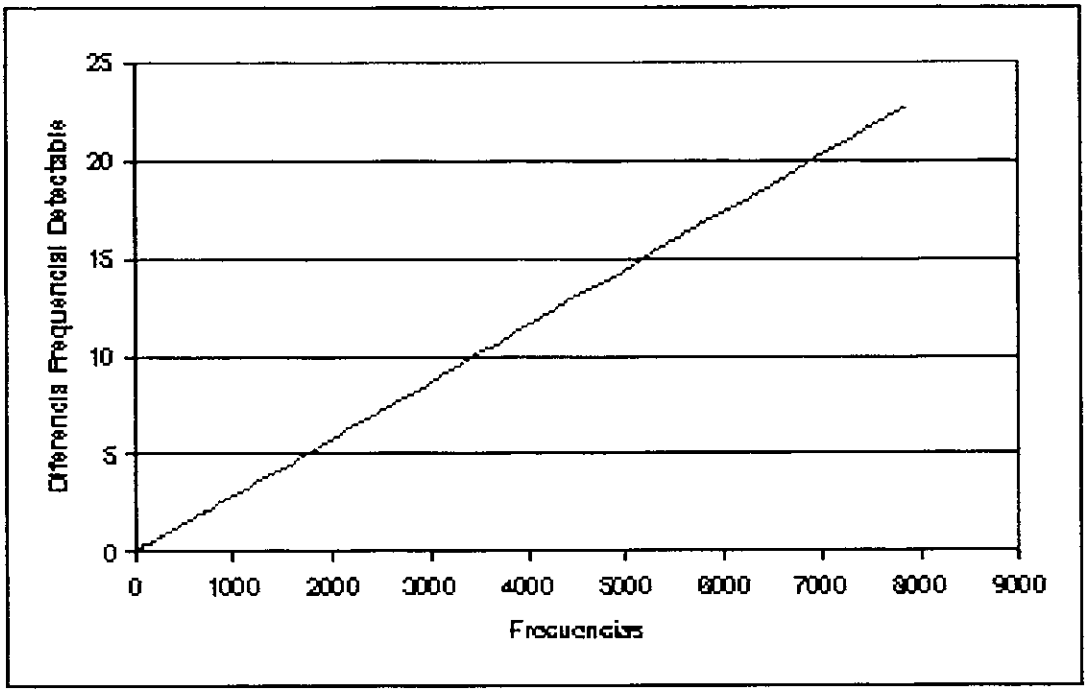


Figura 25 : Diferencia frecuencial perceptible.

C. Adquisición de datos utilizando una tarjeta de sonido estándar mediante el uso de la librería winmm.dll de Windows

La librería de Windows winmm.dll es comúnmente utilizada por aplicaciones multimedia, ya que en ella se encuentran facilidades de utilización de hardware. Con esta librería, las aplicaciones no deben ser diseñadas para el manejo de un hardware específico, al contrario, las aplicaciones deben ser diseñadas para la utilización de esta librería, la cual se encarga de negociar directamente con el software específico del dispositivo, tal como se muestra en la figura 26. De esta manera, las aplicaciones pueden ser heterogéneas en cuanto al uso de hardware.

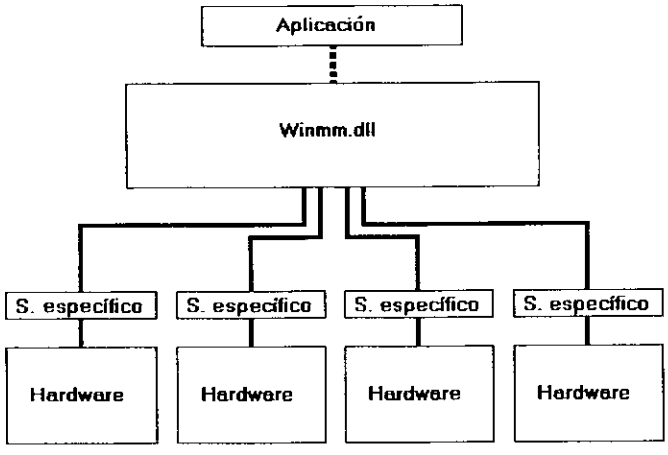


Figura 26: Utilización de hardware específico por medio de la librería winmm.dll (S. = Software).

En el caso de la utilización de la tarjeta de sonido de la PC, sea ésta insertable o integrada, winmm.dll contiene funciones que permiten la adquisición de datos, la especificación de una velocidad de muestreo, el tiempo en el que se debe hacer un muestreo, etc.

La lista de funciones se puede encontrar en el Anexo A.

1. Utilización de las funciones de winmm.dll con Visual Basic 6.

Visual Basic necesita ciertas declaraciones de las funciones a utilizar. Las declaraciones tienen esta estructura

XX1 Declare Function XX2 Lib "winmm" (XX3, XX4, ...) As XX5

En donde:

- *XX1 Determina el alcance de la función (private, public). Esto determina si la función puede ser utilizada en alguna otra forma o modulo.*
- *XX2 Nombre de la función a declara*
- *XX3, XX4, ... Declaración de variables que requiere la función a utilizar*
- XX5 Tipo de variable que devuelve la función a utilizar

2. Funciones que se utilizaron para la adquisición de datos.

Función: waveInOpen

Descripción: Abre el dispositivo de entrada de ondas para adquisición de datos.

Requiere:

- Variable en la que se guarda el número característico del dispositivo, asignado por el sistema operativo.
- Cadena en la que se guarda la descripción del dispositivo de hardware.
- El formato de los datos (puede ser normal (PCM), o con alguna compresión.
- Número de canal abierto.
- Número de muestras por segundo.
- Número de bits por muestra.
- Forma de empaquetado de los bits por muestra.

El software específico del hardware de las tarjetas de sonido tiene ciertos valores predeterminados para el número de muestras por segundo, de manera que el número solicitado debe estar dentro de ese grupo de valores. Estos valores, para una tarjeta *sound Blaster Live! Value* en formato PCM se muestran en la tabla 7.

Frecuencia de muestreo (Hz)	Máximo de Bits por muestra
8000	8 y 16
11025	8 y 16
12000	8 y 16
16000	8 y 16
22050	8 y 16
24000	8 y 16
32000	8 y 16
44100	8 y 16
48000	8 y 16

Tabla 7: Valores predeterminados para el número de muestras por segundo.

El número de bits por muestra está limitado por las características específicas del hardware. Para las tarjetas de sonido, generalmente el máximo de bits por muestra es de 16.

Función: waveInStart

Descripción: Empieza la adquisición de datos.
Requiere:

- Variable en la que se guarda el número característico del dispositivo, asignado por el sistema operativo.

Esta función empieza la adquisición de datos en la parte de hardware y software especifico. Esta función debe llamarse previamente a comenzar la adquisición de datos a nivel de la aplicación.

Función: waveInPrepareHeader

Descripción: Prepara un buffer para el dispositivo de adquisición.

Requiere:

- Variable en la que se guarda el número característico del dispositivo, asignado por el sistema operativo.
- Puntero a un buffer de datos.
- Número de Bytes que se deben llenar del buffer.

Esta función prepara el buffer de datos que utilizará el software especifico. El tiempo total de muestreo debe deducirse de la relación entre muestras por segundo f_m, número de muestras requeridas M, tamaño del buffer, el número de Bytes que deben llenar el buffer B.

$$T = \frac{f_m}{M} = \frac{f_m}{A_B}$$

Donde A_B es la relación de Bytes por muestra, es decir, si cada muestra tiene 16 bits, entonces se requieren 2 Bytes por muestra.

Función: waveInAddBuffer

Descripción: Empieza el llenado del buffer.

Requiere:

- Variable en la que se guarda el número característico del dispositivo, asignado por el sistema operativo.
- Puntero a un buffer de datos.
- Número de Bytes que se deben llenar del buffer.

Esta función empieza el llenado del buffer según las especificaciones introducidas con las funciones waveInOpen, waveInStart y waveInPrepareHeader. La función devuelve un puntero a una variable la cual sirve para monitorear el estado del proceso de llenado del buffer.

En la figura 27 se presenta una fracción de código escrito en Visual Basic, el cual espera a que concluya el llenado de un buffer para luego proseguir con otras funciones del programa.

```
Call waveInAddBuffer(DevHandle, VarPtr(Wave), Len(Wave))
Do
 DoEvents
Loop Until ((Wave.dwFlags And WHDR_DONE) = WHDR_DONE) Or DevHandle = 0
```

Figura 27: Código utilizado para el llenado del buffer.

Aquí, *WHDR_DONE* representa una constante que corresponde a la finalización del llenado del buffer.

Función: waveInUnprepareHeader

Descripción: Invierte la acción de waveInPrepareHeader

Requiere:

- Variable en la que se guarda el número característico del dispositivo, asignado por el sistema operativo.
- Puntero a un buffer de datos.
- Número de Bytes que se deben lle nar del buffer.

Esta función libera el buffer de datos, y debe ser llamada luego de haberlo llenado con la función waveInAddBuffer.

Función: waveInReset

Descripción: Detiene la adquisición de datos.

Requiere:

- Variable en la que se guarda el número característico del dispositivo, asignado por el sistema operativo.

Esta función detiene la adquisición de datos por parte del dispositivo de sonido, además si existe algún buffer en proceso de llenado de datos, este es devuelto inmediatamente a la aplicación. Su efecto no es permanente, es decir, el dispositivo puede reanudar la adquisición de datos.

Función: waveInClose

Descripción: Detiene permanentemente la adquisición de datos.

Requiere:

- Variable en la que se guarda el número característico del dispositivo, asignado por el sistema operativo.

Esta función detiene la adquisición de datos permanentemente por parte del dispositivo de sonido; además, si existe algún buffer en proceso de llenado de datos, éste es devuelto inmediatamente a la aplicación.

Función: waveInGetNumDevs

Descripción: Devuelve el número de dispositivos de adquisición presentes.

Requiere:

- Ningún parámetro.

Esta función devuelve el número de dispositivos presentes para la adquisición de datos. Si el valor devuelto es cero, puede significar algún error en el proceso.

Función: waveInGetDevCaps

Descripción: Devuelve la descripción del dispositivo de adquisición.

Requiere:

- Número comprendido entre 1 y el número devuelto por waveInGetNumDevs.
- Puntero a cadena
- Número de caracteres de la cadena

Esta función devuelve la descripción del dispositivo de adquisición de datos en una cadena de caracteres. En esta descripción puede estar la marca, el modelo y alguna otra información de interés.

D. Análisis frecuencial de una señal de un instrumento musical de cuerda utilizando la Transformada Rápida de Fourier (FFT)

La Transformada Discreta de Fourier es la herramienta básica que se utilizará para calcular el espectro frecuencial de la señal generada por el instrumento de cuerda.

$$G(f)\Big|_{f=n/T} \approx \sum_{k=0}^{N-1} g(k\Delta t) e^{-j(2\pi/N)nk} \Delta t \qquad \text{ecuación (3)}$$

En donde $t = k\Delta t$, $f = n/T$ y $\Delta t = T/n$. Siendo t el tiempo continuo, Δt el inverso de la frecuencia de muestreo, N el número total de muestras, n el índice de muestra y T el tiempo durante el cual se tomaron las muestras.

Esta transformación puede calcularse de una manera eficiente. Esta manera eficiente es La Transformada Rápida de Fourier(FFT) y no se discutirá aquí su desarrollo.

La resolución en el dominio de la frecuencia depende del tiempo durante el cual se tomaron muestras, así

$$\Delta f = \frac{n+1}{T} - \frac{n}{T} = \frac{1}{T}$$

La amplitud de la señal proporcionada por una cuerda que vibra decae exponencialmente. Debido a este decaimiento, llegará el punto en el que la señal será igual o menor que el ruido (Tómese en cuenta que se pretende no utilizar ningún amplificador entre el instrumento musical y la tarjeta de sonido de la PC). Entonces, debe encontrarse un T máximo en el que la señal de entrada sea mayor que el ruido.

Para determinar el rango en el que se encontrará este tiempo se analizará la señal proporcionada por el instrumento de cuerda, de manera que se determine el tiempo en el cual la relación señal a ruido sea mayor que 1.

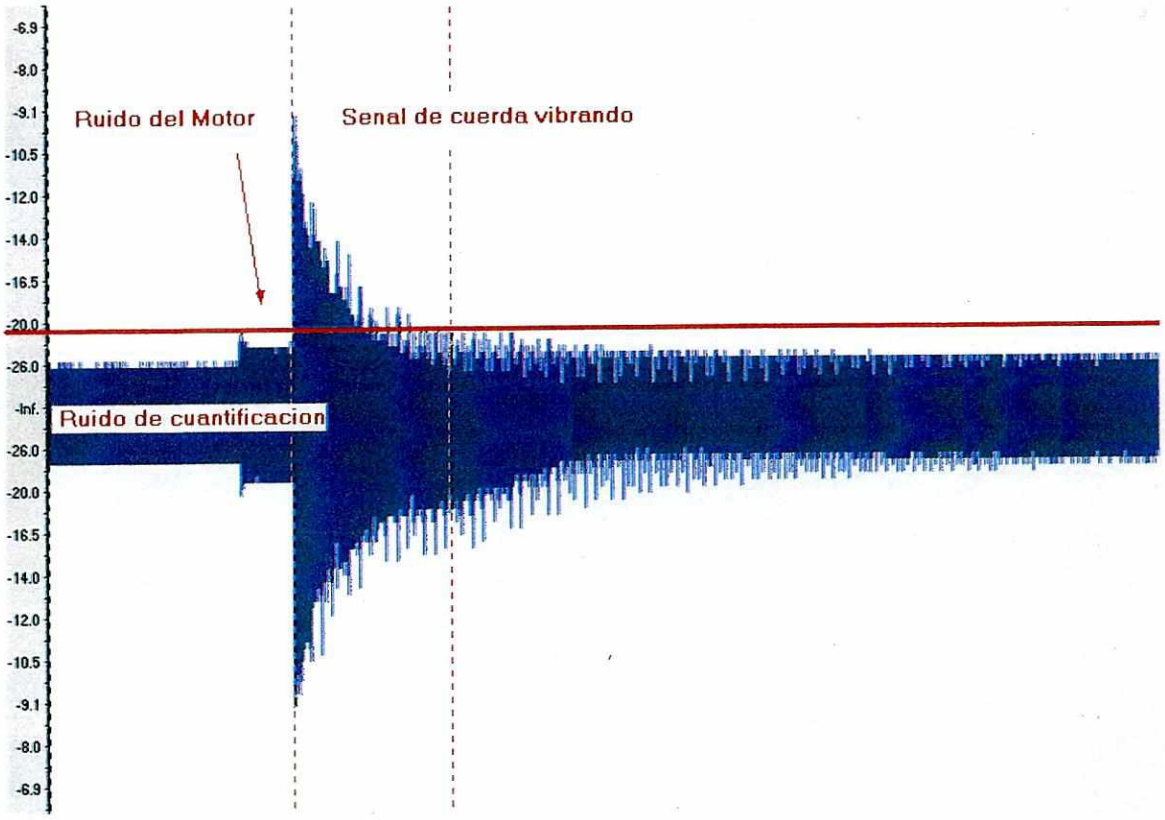


Figura 28: Señal obtenida de la guitarra bajo al utilizar el sistema de excitación descrito en el capitulo I.

La figura 28 corresponde a una de cuatro muestras tomadas (una de cada cuerda de una guitarra bajo). En todas se consideró el tiempo en el que la señal sobrepasaba el ruido de cuantificación y el ruido del motor actuador, siendo la que se muestra en la figura 28 la de menor duración con alrededor de 2.2 segundos. Las cuerdas se encontraban afinadas como se muestra en la tabla 8.

Cuerda	$(Señal/Ruido) \geq 1$ (Seg.)
1 (G)	2.206
2 (D)	2.740
3 (A)	5.062
4 (E)	6.594

Tabla 8: Afinación y relación señal-ruido > 1.

Se observa que el tiempo de duración de la señal sobre el ruido, es decir, la amplitud de la oscilación depende de la masa por unidad de longitud de la cuerda así como de la tensión que está presente.

Se realizará, entonces, una investigación preliminar para determinar la viabilidad de la detección efectiva de la frecuencia normal de oscilación o alguna armónica de esta para que así sea posible la afinación de una guitarra bajo mediante el análisis frecuencial. Se utilizará para éste análisis un tiempo máximo de adquisición de datos de 2.2 segundos, por ser éste el menor tiempo en el cual la señal proveniente de la guitarra bajo se mantiene sobre el ruido (de cuantificación y de actuadores) como se muestra en la tabla 8.

El tiempo de adquisición de datos debe elegirse también de acuerdo al número de muestras que debe ser ingresado al algoritmo de cálculo de la Transformada Rápida de Fourier. Este debe ser una potencia de dos.

Las variables que pueden ser modificadas para alcanzar el tiempo de muestreo deseado son:

- El tamaño del Buffer de datos
- La frecuencia de muestreo

Las posibles frecuencias de muestreo se observan en la tabla 7. La frecuencia máxima que se puede llegar a querer analizar correspondería a la frecuencia más alta de la 8va Octava, es decir B_8 (7902.13Hz). Para cumplir este requerimiento, la frecuencia de muestreo debe ser de al menos

$$f_m \leq 7902.13Hz * 2 = 15804.26Hz$$

Por lo tanto se elegirá de la tabla 7 la frecuencia de muestreo de 16000Hz.

Teniendo ya la frecuencia de muestreo fija, se debe dimensionar el buffer de datos de manera que el tiempo de muestreo sea una potencia de dos así:

$$T = \frac{T.Buffer}{f_m}$$

Entonces,

$$T = \frac{32768m}{16000m/s} = 2.048seg$$

De esta manera, el tamaño del buffer corresponde a una potencia de dos y el tiempo durante el cual se toman las muestras no excede los 2.2 segundos.
En el régimen de la frecuencia se tendrá una resolución dada por:

$$\Delta f = \frac{n+1}{T} - \frac{n}{T} = \frac{1}{T} = \frac{1}{2.048s} = 0.48828125\text{Hz}$$

En la librería Winmm.dll la unidad de dimensionamiento del buffer se encuentra en Bytes. Debido a que se utilizará una resolución de 16 bits, el tamaño del buffer se convierte en $32768 * 2 = 65536$. Estos datos se encuentran resumidos en la tabla 9.

Descripción	Dato
Frecuencia de muestreo	16000 m/s
Tiempo de muestreo	2.048s
Tamaño del buffer de datos(winmm.dll)	65536
Resolución Frecuencial	0.48828125Hz

Tabla 9: Datos correspondientes a la adquisición de datos con la librería winmm.dll.

1. **Utilización de ventanas en el dominio del tiempo.** El algoritmo de la Transformada Rápida de Fourier asume que el grupo de datos que se ingresa corresponde a un periodo de una señal infinita, por lo tanto el primer y el último dato de esta secuencia deben coincidir. Debido a la forma en que se adquieren estos datos, se espera que esta coincidencia se dé únicamente en una despreciable minoría de casos. Cuando esta coincidencia no se da, el algoritmo de la Transformada Rápida de Fourier da como resultado contribuciones frecuenciales inexistentes en el régimen de la frecuencia de la señal original. Este efecto se puede observar significativamente en el caso del análisis de señales compuestas de frecuencias discretas. Esto se muestra en la figura 29, en donde se analizó el espectro frecuencial de la suma de tres señales sinusoidales con fases diferentes de modo que el primer y el último dato difirieran en magnitudes diferentes. Siendo (Fase 1) el caso con mayor diferencia y (Fase 7) el caso con la menor.

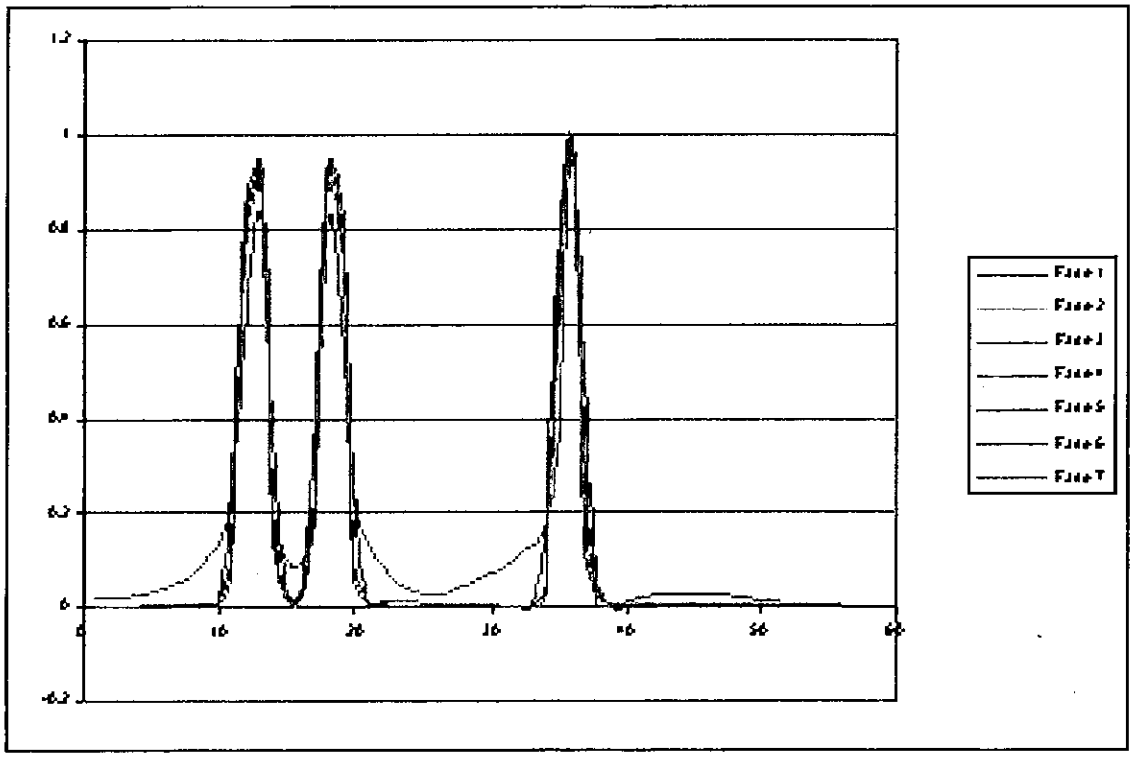


Figura 29: Espectros correspondientes a señales compuestas por la suma algebraica de 3 sinusoides. Las sinusoides que componen cada señal tiene un desfase tal que el primer y el último punto de tales señales difiere en un mayor grado para cada señal.

Se Observa en la figura 29 que el efecto corresponde a ensanchar el espectro en la base de los picos de frecuencia presente en la señal original. Estas contribuciones frecuenciales corresponden a una distorsión de la señal original, ocasionada por la utilización del algoritmo de la Transformada Rápida de Fourier, la cual tiene como objetivo que el primer y el ultimo punto de esta señal original coincidan.

La falta de coincidencia del primer y ultimo punto de la señal a analizar también puede ocurrir a causa de la presencia de una envolvente exponencial decreciente en la señal original.

a. Otra perspectiva. Se sabe que para obtener el espectro de una señal la cual se es generada a partir del producto de otras señales, basta con realizar la convolución de sus espectros. El espectro correspondiente a una ventana rectangular corresponde a la función $snc(x)$. El espectro correspondiente a la señal exponencial decreciente se muestra en la figura 30 y las señales en el dominio del tiempo que corresponden a estos espectros se muestran en la figura 31.

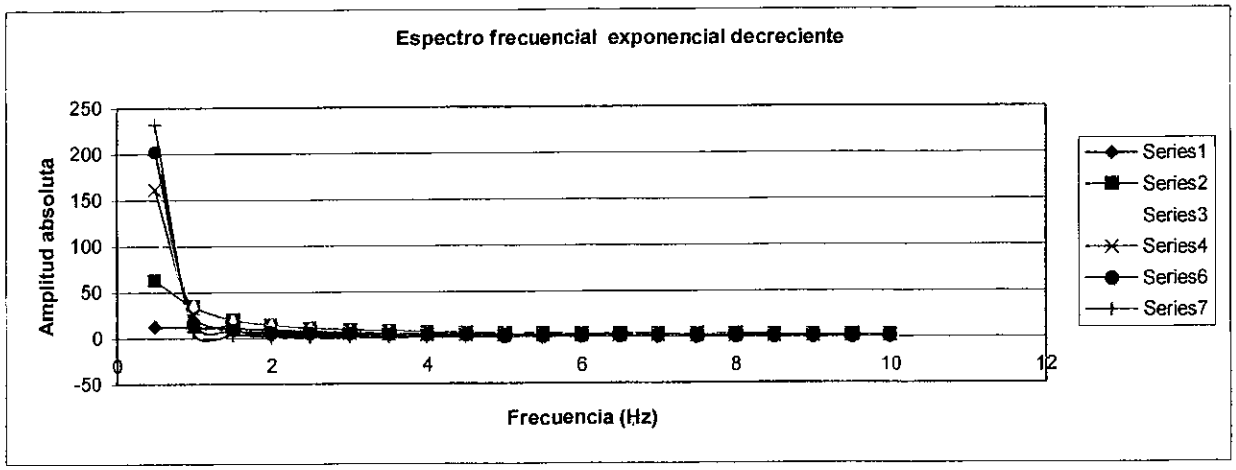


Figura 30: Espectro de una señal exponencial decreciente.

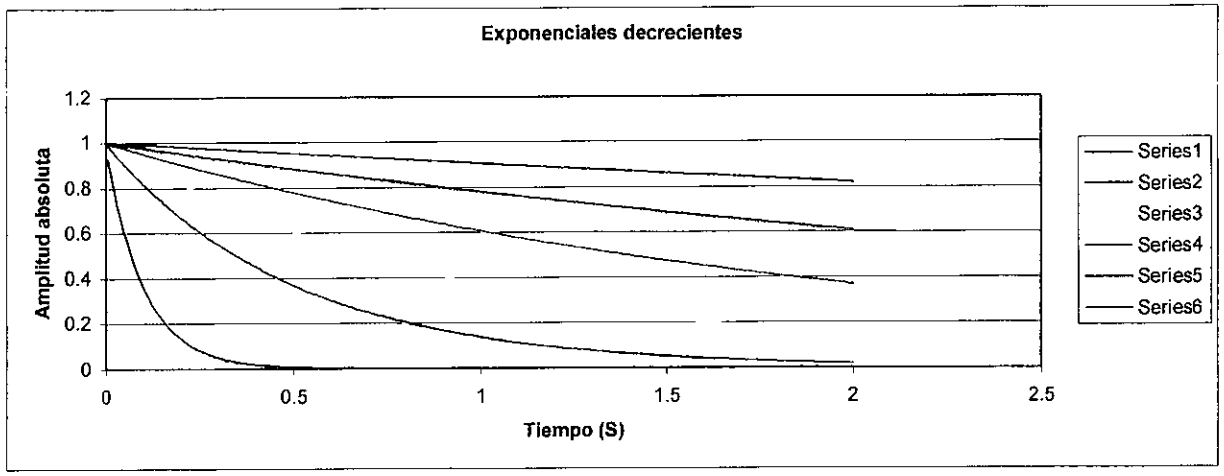


Figura 31:Señales en el dominio del tiempo que corresponden a los espectros de la figura 30.

Estas figuras demuestran claramente que la amplitud absoluta del espectro de éste tipo de señales depende de la rapidez con que la señal decaiga en el dominio del tiempo, siendo mayor la amplitud espectral mientras menor sea la rapidez del decaimiento en el dominio del tiempo.

Sin la necesidad de un análisis extensivo, se puede concluir que la convolución de estos dos espectros con un tono puro ocasionaran un ensanchamiento en el espectro del mismo.

b. Solución practica. Para la reducción de este efecto, se considera la utilización de ventanas en el dominio del tiempo. La figura 32 muestra las 4 ventanas más comunes. La figura 33 muestra cada una de las 4 ventanas aplicadas a la señal que presenta la menor concordancia en su punto inicial y final de la figura 29.

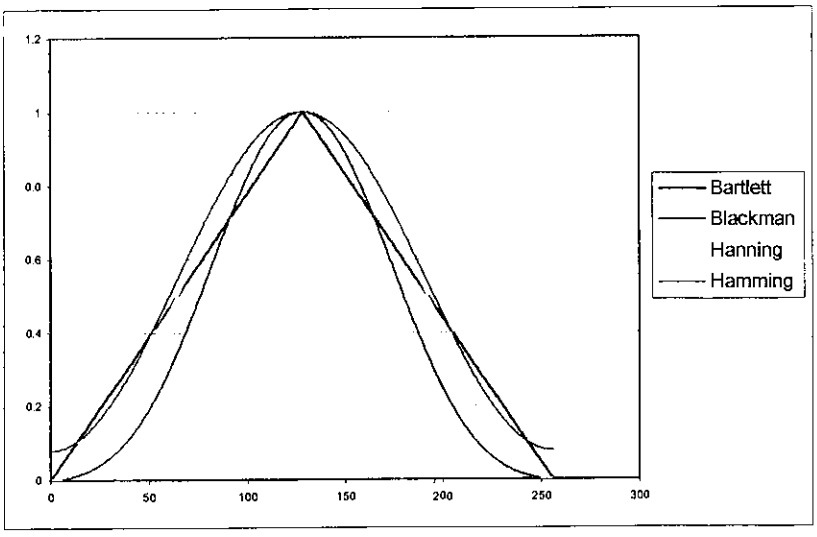


Figura 32: Ventanas más comúnmente utilizadas.

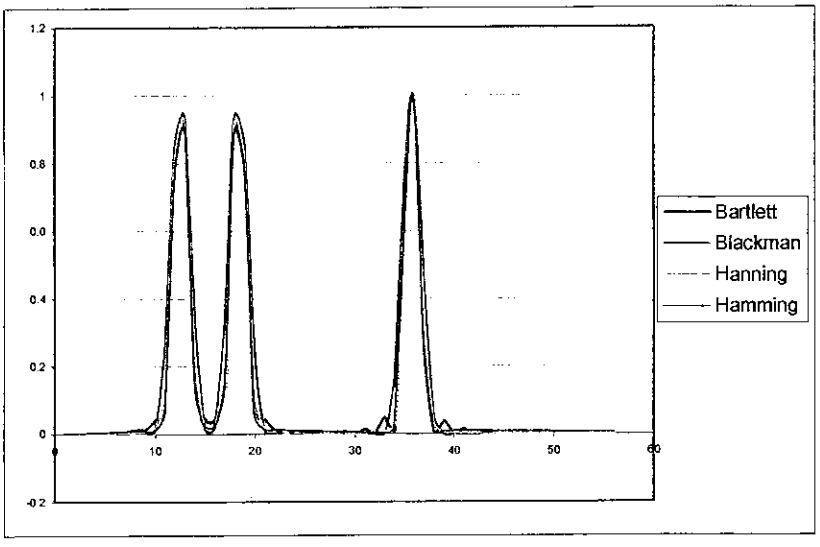


Figura 33: Espectro obtenido de la señal que presenta menor coincidencia en los puntos inicial y final. Este espectro se presenta en la figura 29, luego de la utilización de las 4 diferentes ventanas sobre estos datos.

Se puede observar claramente que la ventana de Hamming es la más efectiva en la eliminación del ensanchamiento en la base de los picos de frecuencia.

Se utilizará entonces una ventana de Hamming a la señal generada por la guitarra bajo para descartar de antemano cualquier distorsión en el espectro introducida por la utilización algoritmo de la Transformada Rápida de Fourier en una señal finita

2. Determinación de la viabilidad de la detección de frecuencias asociadas con la afinación del instrumento de cuerda. El objeto de la realización del análisis frecuencial es el de determinar la viabilidad de la detección certera de la frecuencia fundamental de oscilación. También se espera poder detectar con éxito las armónicas que corresponden a la oscilación normal, a manera de poder realizar una afinación que dependa de una frecuencia alta. Esto es importante, dado que el oído humano tiene menor sensibilidad a cambios de frecuencia a medida que se aumenta la misma (Ver figura 25). Por ello, la realización de una afinación a frecuencias altas garantiza que todas las armónicas a frecuencias menores se encuentran afinadas dentro del rango en el que no es detectable para el oído humano.

Las figuras 34, 35, 36 y 37 muestran los espectros obtenidos de cada una de las 4 cuerdas de la guitarra bajo con una Transformada Rápida de Fourier. Las muestras fueron tomadas con las especificaciones que se muestran en la tabla 9.

El objetivo de la obtención del espectro es la detección de un grupo de frecuencias específico. Por esto, Los espectros fueron normalizados puesto que solo es importante la relación entre las amplitudes del mismo y no su valor absoluto.

Cada figura corresponde al promedio de 14 grupos de datos provenientes de la misma cuerda. Este número esta por encima del mínimo que se requiere para la mayoría de los análisis estadísticos que se podrían realizar.

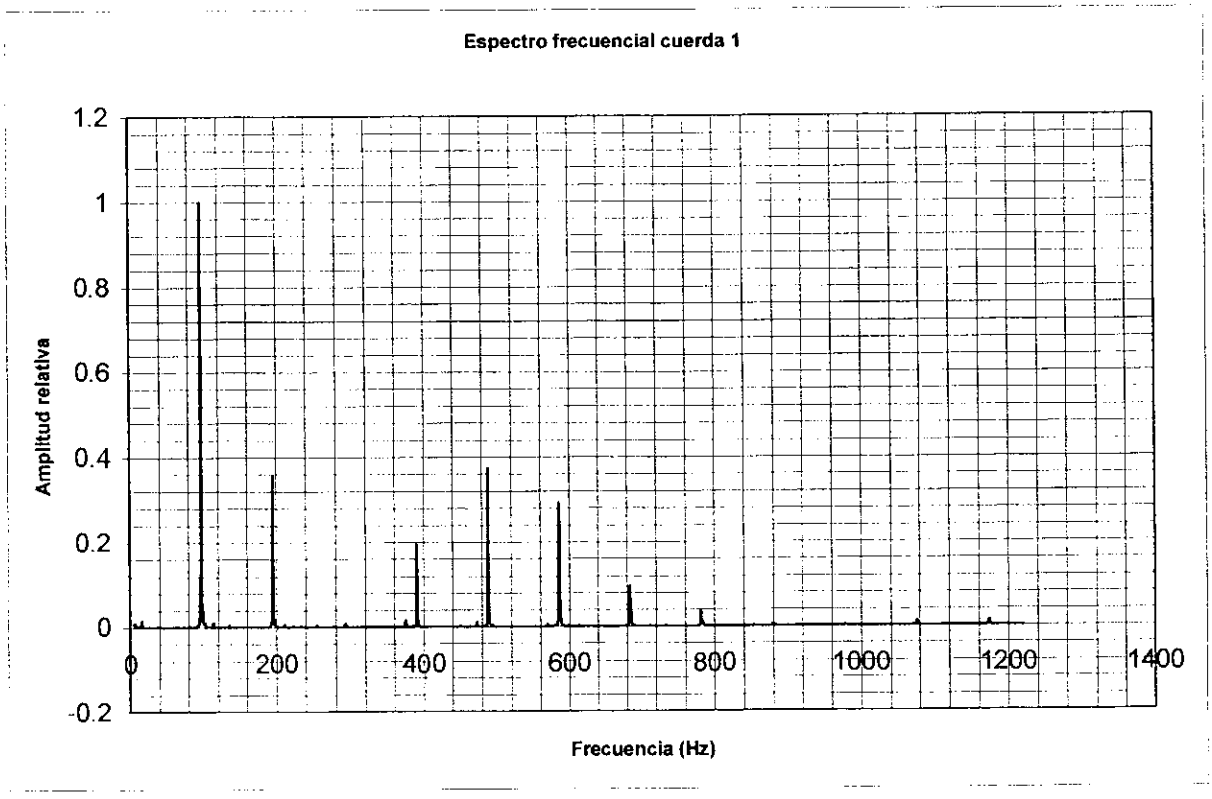


Figura 34: Espectro de los datos obtenidos de la cuerda 1 con las especificaciones de la tabla 9.

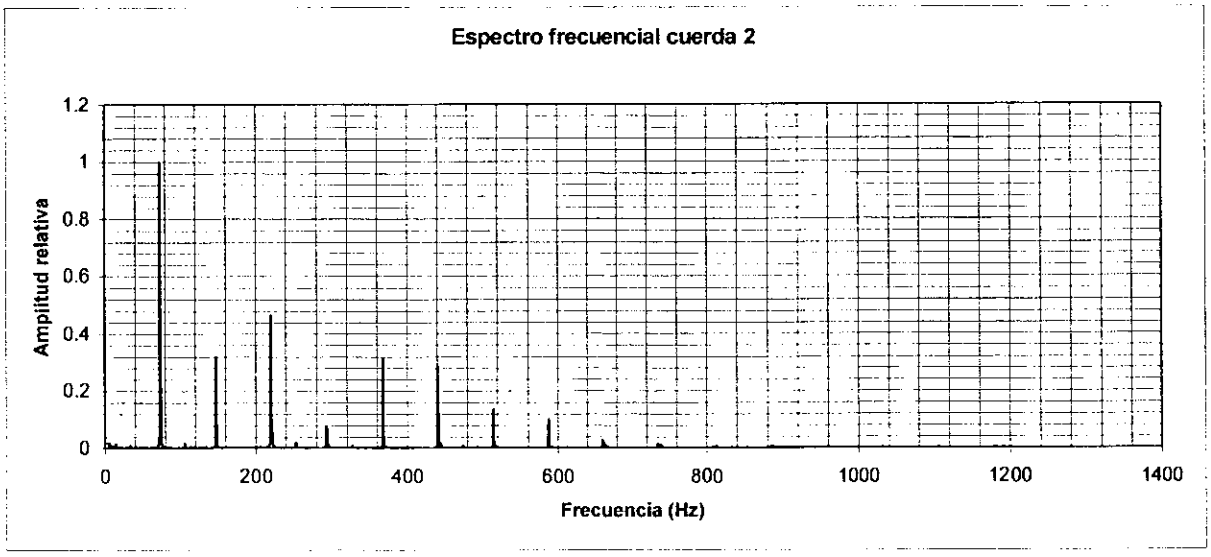


Figura 35: Espectro de los datos obtenidos de la cuerda 2 con las especificaciones de la tabla 9.

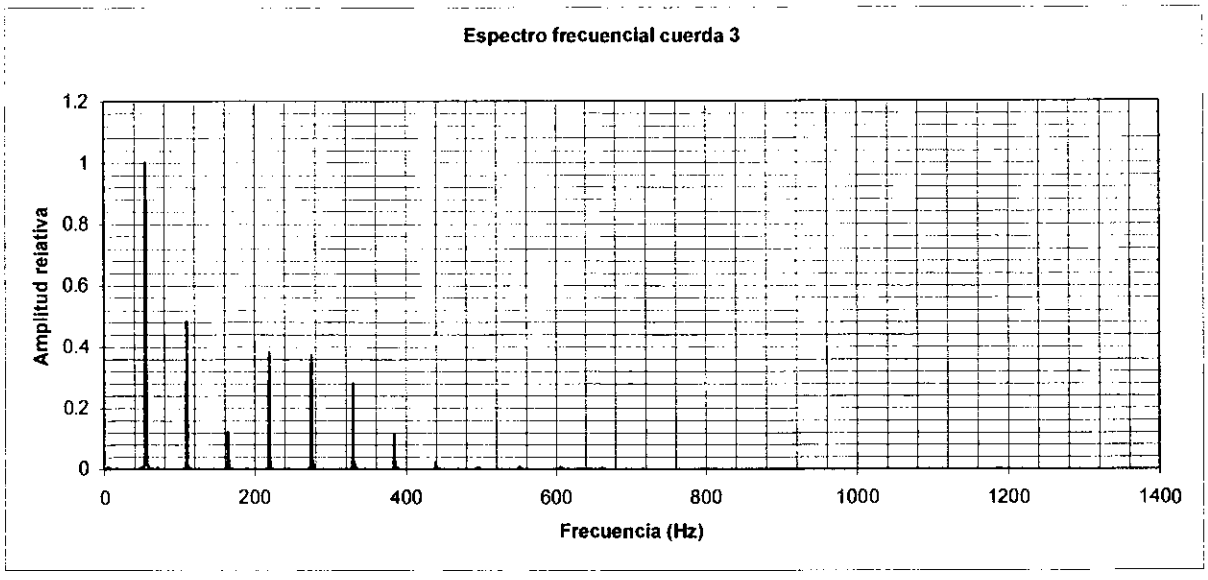


Figura 36: Espectro de los datos obtenidos de la cuerda 3 con las especificaciones de la tabla 9.

Se ha tomado en cuenta desde el valor que corresponde a 5Hz debido a que se detectó una gran contribución debido a pequeñas señales DC que formaban parte de la señal original. Estas señales DC se atribuyen a los efectos de la cuantificación. El hecho de no haber tomado en cuenta el espectro frecuencial correspondiente a los primeros 5Hz, en el proceso de normalización de la señal producida por la guitarra bajo, descartó casi por completo la evidencia de la existencia de una componente DC en los datos en el dominio del tiempo. Este fue un punto clave en el análisis de espectro frecuencial, dado que los picos registrados en los primeros 5Hz tuvieron una magnitud absoluta de al rededor de 6000000 mientras que el pico más grande pasados los 5Hz se registraba al rededor de 10.

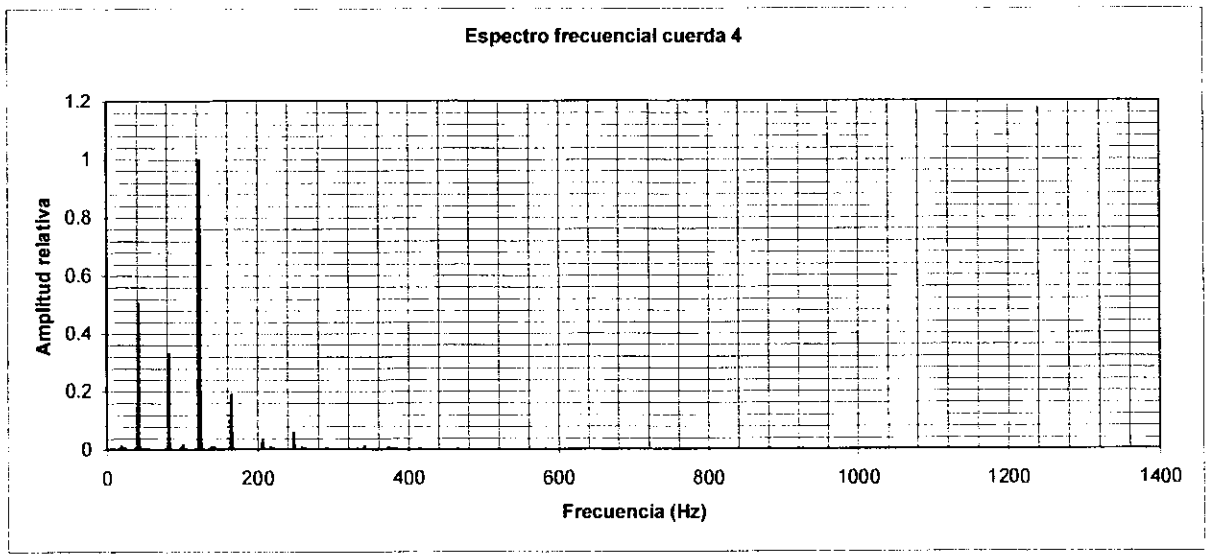


Figura 37: Espectro de los datos obtenidos de la cuerda 4 con las especificaciones de la tabla 9.

Se puede observar también que el espectro frecuencial de todas las cuerdas decrece a menos del 1% pasada cierta frecuencia límite (se denominará frecuencia Portelli). Estas frecuencias se muestran en la tabla 10

Cuerda	Frecuencia P (Hz)
1	1184
2	740
3	457
4	343

Tabla 10: Frecuencias Portelli.

Esto indica que es innecesario que el espectro analizado sea mayor de 1200Hz. Por lo tanto ninguna contribución frecuencial será tomada en cuenta pasado este valor. Esto indica que la guitarra bajo sólo tiene contribuciones frecuenciales entre las primeras cinco a seis octavas. Por ello no tiene caso el análisis de octavas superiores.

La afinación aproximada de cada cuerda para el análisis de las figuras 34, 35, 36 y 37 se muestra en la tabla 11

Cuerda	Afinación	Rango de la frecuencia normal
1	*G*	73.42-138.59Hz
2	*C*	55.00-103.83Hz
3	*A*	41.2-77.78Hz
4	*E*	30.87-58.27Hz

Tabla 11: Afinación presente en las cuerdas cuyos espectros frecuenciales se muestran en las figuras 34, 35, 36 y 37.

Cada cuerda debería tener una armónica en cada múltiplo entero de la frecuencia normal de oscilación. Para corroborar este aspecto de la teoría se marcaron todos los puntos donde deberían aparecer las armónicas correspondientes a las primeras cinco octavas para cada afinación. Esto se muestra en la figura 38, 39,40 y 41.

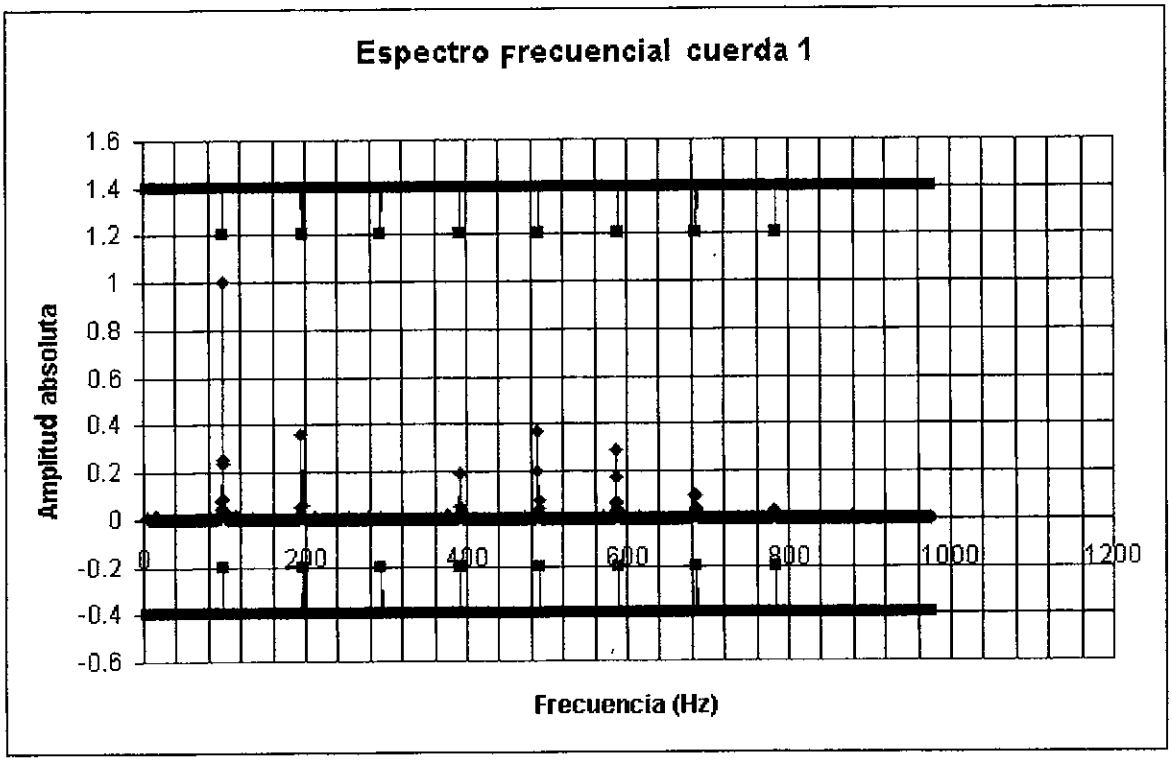


Figura 38: Espectro de la señal obtenida al excitar la cuerda 1. La señal roja indica donde se deberían encontrar las armónicas debido a la afinación que se observa en la tabla 11.

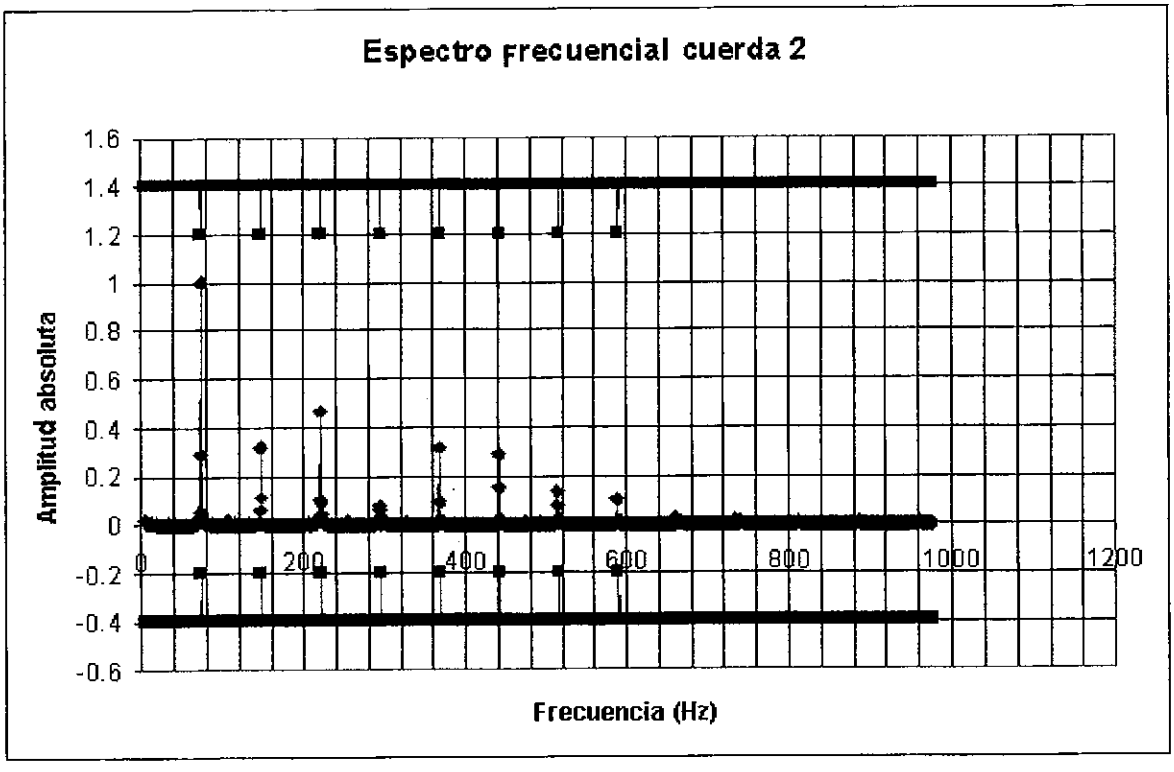


Figura 39: Espectro de la señal obtenida al excitar la cuerda 2. La señal roja indica donde se deberían encontrar las armónicas debido a la afinación que se observa en la tabla 11.

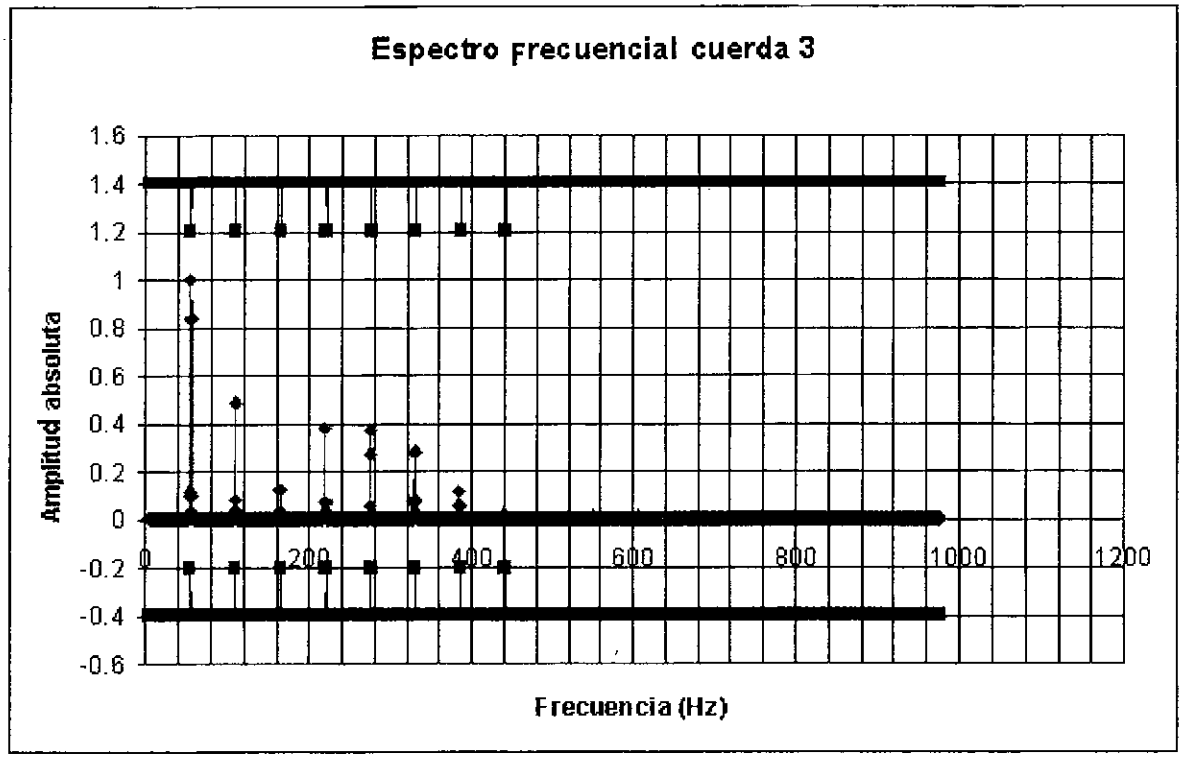


Figura 40: Espectro de la señal obtenida al excitar la cuerda 3. La señal roja indica donde se deberían encontrar las armónicas debido a la afinación que se observa en la tabla 11.

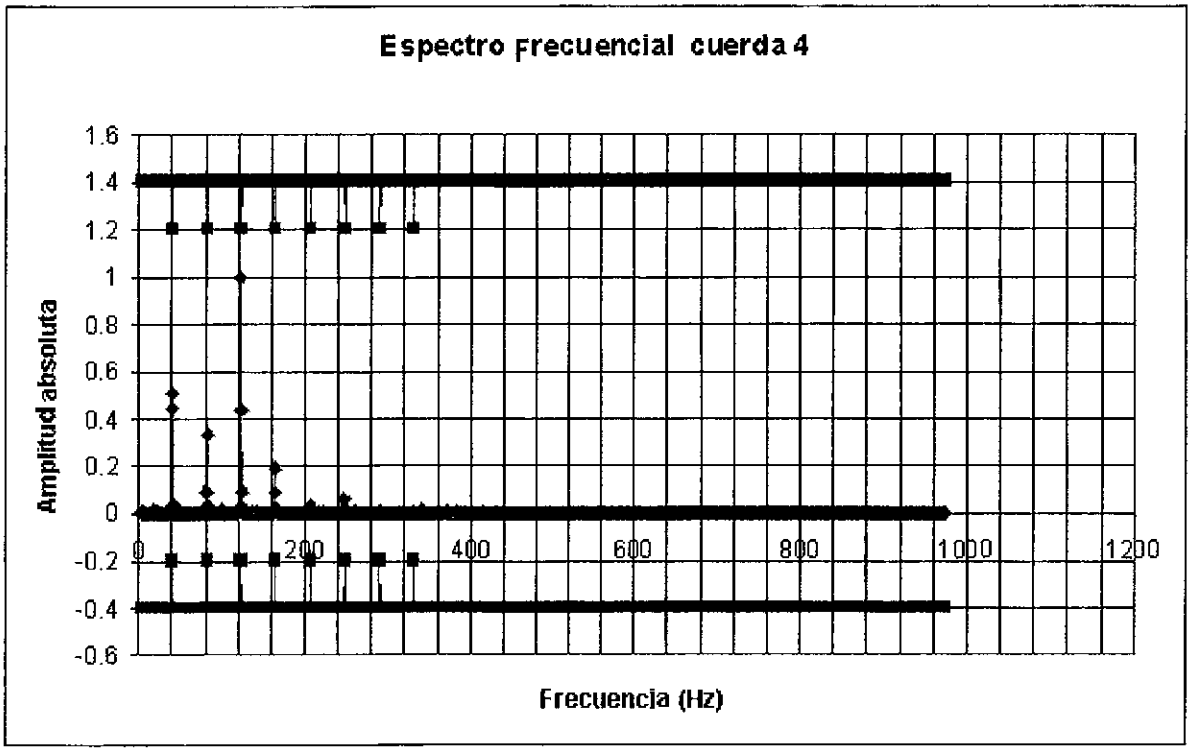


Figura 41: Espectro de la señal obtenida al excitar la cuerda 1. La señal roja indica donde se deberían encontrar las armónicas debido a la afinación que se observa en la tabla 11.

Según estos datos, las armónicas están presentes en donde era esperado. Nótese que las amplitudes de los conjuntos de armónicas correspondientes a las cuerdas 1 y 3 se parecen (considerando una escala diferente). También esto ocurre con las cuerdas 2 y 4. véase las figuras 41 y 42.

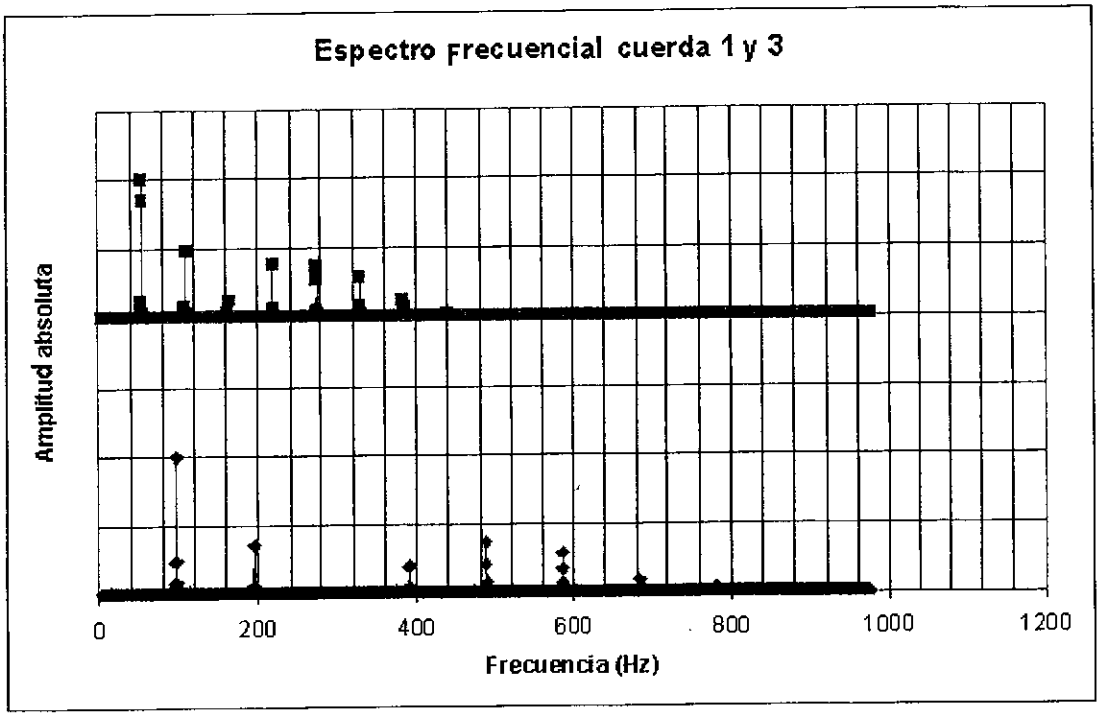


Figura 42: Similitud entre los espectros de las cuerdas 1 y 3.

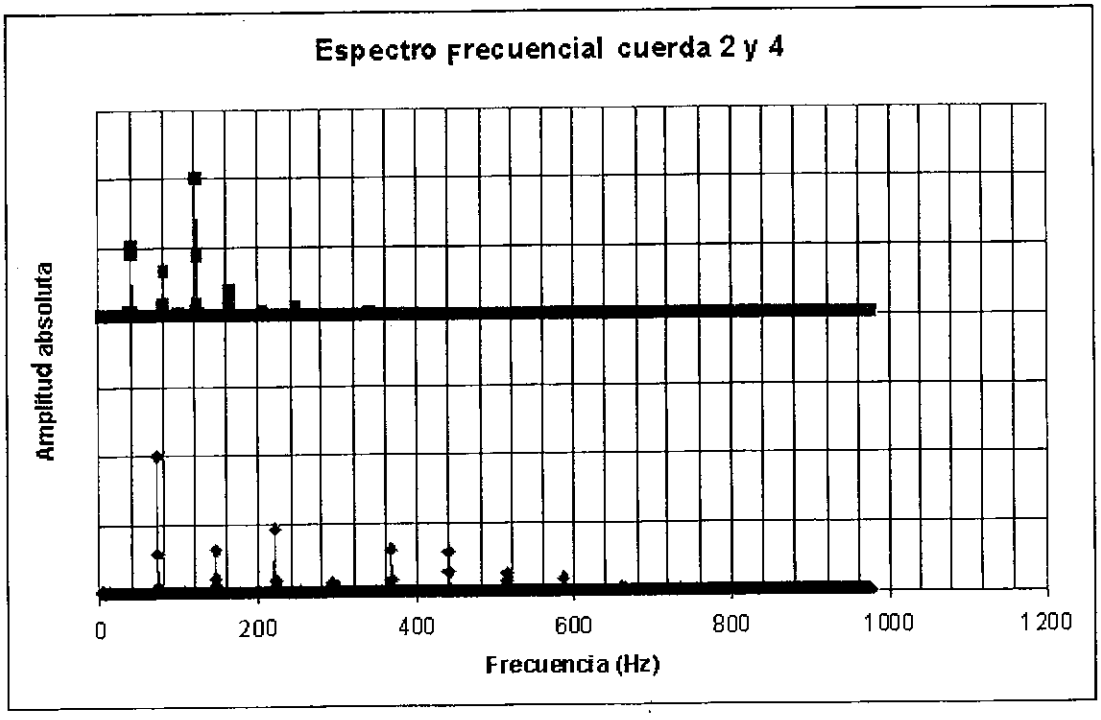


Figura 43: Similitud entre los espectros de las cuerdas 2 y 4.

Esta coincidencia corrobora lo pronosticado por la teoría. La causa fundamental de la similitud de estos espectros es que las parejas de cuerdas mencionadas fueron excitadas en el mismo punto físico. Esto quiere decir que una única función describía la forma que cada cuerda tenía antes de que se diera la vibración y, por lo tanto, ambas tendrían características similares de oscilación.

E. Estrategia de afinación (Método de afinación Portelli)

Todas las cuerdas poseen una frecuencia de amplitud relativa máxima la cual puede ser asociada con la frecuencia normal de oscilación. Esta frecuencia será utilizada para realizar un primer acercamiento, el cual tendrá como limite la resolución máxima del espectro frecuencial (0.48828125Hz). Luego, la afinación dependerá de comparaciones hechas con una armónica de orden mayor. Esto garantiza que las harmónicas de orden menor estén dentro del rango en el cual el oído humano no percibe cambios en frecuencia.

En la tabla 12 se puede ver la relación entre frecuencias y grado de percepción del oído humano a cambios frecuenciales.

Existe entonces una frecuencia mínima (frecuencia M) para la cual la diferencia perceptible por el oído humano supera los 0.48828125Hz. Esta frecuencia corresponde a los 87.31Hz. Por lo tanto, la afinación por frecuencias armónicas debe llevarse a cabo en frecuencias que estén por encima de esta frecuencia mínima.

Debe encontrarse, entonces, el rango para cada cuerda en el cual aparezca una única armónica, con una frecuencia mayor a 87.31Hz. Los rangos escogidos se normalizarán, convirtiendo así a esta armónica en la frecuencia máxima relativa de cada rango. En la tabla 12 se pueden observar estos rangos.

cuerda	Armónica (frecuencia > 87.31Hz)
1	$(98.00*2)\pm 49Hz$
2	$(73.42*3)\pm 36.71Hz$
3	$(55.00*3)\pm 25Hz$
4	$(41.2*3)\pm 21.1Hz$

Tabla 13: Rangos en los cuales se espera encontrar solamente una armónica

Nótese que estos rangos tienen una amplitud igual a la frecuencia normal de oscilación de cada cuerda. Esto se debe a que como cada armónica está separada por este valor, entonces el rango resultante siempre contendrá únicamente 1 armónica.

Nótese también que los rangos no son fijos, sino que dependen de la frecuencia normal que se requiere en la cuerda. Esta medida es necesaria, ya que si se definen rangos fijos, existe la posibilidad de que se encuentren dos armónicas en el mismo rango, siendo así los cálculos de normalización erróneos.

Estos Rangos se denominarán "rangos Portelli".

Analizando la tabla 12 se llega a la conclusión:

$$RangoPortelli = \left(fN * NE - \frac{fN}{2}, fN * NE + \frac{fN}{2} \right)$$

ecuación (4)

en donde

fN = frecuencial normal
NE = número entero positivo

El número entero puede elegirse de acuerdo a la amplitud relativa que se quiera manejar en el espectro frecuencial para el cálculo de aproximación final. En el caso de la tabla 13 se eligió para las cuerdas 2, 3 y 4 la segunda armónica, mientras que para la primera se eligió la primera, ya que la segunda tenía una amplitud relativa despreciable.

Se presenta en la tabla 14 el resumen de especificaciones necesarias para la adquisición y análisis de la señal proveniente de una guitarra bajo con el objetivo de realizar una afinación de la misma por medio del análisis del espectro frecuencial.

Descripción	Dato
Frecuencia de muestreo	8000 m/s
Tiempo de muestreo	2.048s
Tamaño del buffer de datos(winmm.dll)	32768
Resolución frecuencial	0.48828125Hz
Frecuencia mínima requerida en el espectro	29.14Hz
Frecuencia máxima requerida en el espectro	256.71Hz
Ancho de banda por analizar	227.57Hz

Tabla 14: Especificaciones de la adquisición de datos utilizados en la aplicación automática para la afinación de una guitarra-bajo utilizando el método de afinación Portelli.

Además, debe tenerse en cuenta la ecuación (4), la cual debe usarse una vez el error de afinación en cada cuerda sea menor de

$$\frac{0.48828125\text{Hz}}{2} = 0.244140625\text{Hz}$$

Por lo tanto el margen correspondiente a cada cuerda es suficiente para una afinación "perfecta" para la percepción del oído humano.

Nota(440Hz)	Frecuencia normal de oscilación	Diferencia perceptible
C_1	32.7	0.189448
$C^{\#}_1/D^{b}_1$	34.65	0.200713
D_1	36.71	0.212648
$D^{\#}_1/E^{b}_1$	38.89	0.225293
E_1	41.2	0.238689
F_1	43.65	0.252882
$F^{\#}_1/G^{b}_1$	46.25	0.267919
G_1	49	0.283851
$G^{\#}_1/A^{b}_1$	51.91	0.300729
A_1	55	0.318612
$A^{\#}_1/B^{b}_1$	58.27	0.337557
B_1	61.74	0.35763
C_2	65.41	0.378895
$C^{\#}_2/D^{b}_2$	69.3	0.401426
D_2	73.42	0.425296
$D^{\#}_2/E^{b}_2$	77.78	0.450585
E_2	82.41	0.477378
F_2	87.31	0.505765
$F^{\#}_2/G^{b}_2$	92.5	0.535839
G_2	98	0.567702
$G^{\#}_2/A^{b}_2$	103.83	0.601459
A_2	110	0.637224
$A^{\#}_2/B^{b}_2$	116.54	0.675115
B_2	123.47	0.715259
C_3	130.81	0.757791
$C^{\#}_3/D^{b}_3$	138.59	0.802851
D_3	146.83	0.850591
$D^{\#}_3/E^{b}_3$	155.56	0.90117
E_3	164.81	0.954757
F_3	174.61	1.011529
$F^{\#}_3/G^{b}_3$	185	1.071678
G_3	196	1.135403
$G^{\#}_3/A^{b}_3$	207.65	1.202918
A_3	220	1.274447
$A^{\#}_3/B^{b}_3$	233.08	1.35023
B_3	246.94	1.430518
	261.63	1.515581
$C^{\#}_4/D^{b}_4$	277.18	1.605703
D_4	293.66	1.701183
$D^{\#}_4/E^{b}_4$	311.13	1.80234
E_4	329.63	1.909513
F_4	349.23	2.023059
$F^{\#}_4/G^{b}_4$	369.99	2.143356
G_4	392	2.270806
$G^{\#}_4/A^{b}_4$	415.3	2.405836
A_4	440	2.548894
$A^{\#}_4/B^{b}_4$	466.16	2.700459
B_4	493.88	2.861037
C_5	523.25	3.031163
$C^{\#}_5/D^{b}_5$	554.37	3.211405
D_5	587.33	3.402365
$D^{\#}_5/E^{b}_5$	622.25	3.604681
E_5	659.26	3.819026
F_5	698.46	4.046117
$F^{\#}_5/G^{b}_5$	739.99	4.286712
G_5	783.99	4.541613
$G^{\#}_5/A^{b}_5$	830.61	4.811671
A_5	880	5.097788
$A^{\#}_5/B^{b}_5$	932.33	5.400918
B_5	987.77	5.722074

Tabla 12: Frecuencias normales de oscilación y cambio perceptible asociado.

F. Ejemplo de aplicación del método de afinación por armónicas (Método de afinación Portelli)

La tabla 15 muestra las particularidades a tomar en cuenta para la adquisición y análisis de la señal proveniente de cada cuerda en particular para la afinación E, A, D, G.

Cuerda	Rango de frecuencia principal	Rango de frecuencia armónica para afinación fina	Octava correspondiente a la frecuencia principal	Octava correspondiente a la frecuencia armónica de afinación fina	Percepción máxima en el rango de frecuencia armónica para afinación fina	Percepción mínima en el rango de frecuencia armónica para afinación fina
1	73.42-138.59Hz	146.83-245.0Hz	2-3	3-4	1.60570264628159Hz	0.8505913471252
2	55.00-103.83Hz	183.56-256.97Hz	2-1	3-4	1.80234028132101Hz	0.954756505768381
3	41.2-77.78Hz	140.0-190.0Hz	2-1	2-3	1.35022959922159Hz	0.715259214643408
4	92.5-174.61Hz	102.5-147.4Hz	2-3	2-3	1.01152928196104Hz	0.535838971550738

Tabla 15: Especificaciones para la adquisición y análisis de la señal proveniente de una guitarra-bajo mediante el método de armónicas.

Se debe realizar una aproximación previa mediante la comparación del mando (E, A, D, G) con la frecuencia principal. Nótese que mediante esta aproximación no es posible llegar a una precisión menor de 0.244140625Hz debido a las especificaciones de la toma de datos (véase tabla 14).

Luego de que el error sea menor de 0.244140625Hz, el mando debe cambiar a un múltiplo entero del mando para la frecuencia Principal. Este número entero debe ser tal que el mando quede dentro del Rango de Frecuencia Armónica para Afinación Fina correspondiente a cada cuerda. A partir de ese momento, se tomará como respuesta del sistema del Rango de Frecuencia Armónica para Afinación Fina correspondiente a cada cuerda, normalizado para la frecuencia mayor encontrada en tal.

La ventaja de este método radica en que el error remanente, luego de haber pasado la barrera de los 0.244140625Hz para una armónica, no es perceptible para el oído humano. Es decir, el error esta dado, para armónicas de orden menor por la siguiente relación:

$$\textit{Relación de error Portelli: } \mathrm{E(k)} = \mathrm{E(n)} \bullet 2^{k-n} \quad ; \quad \mathrm{n} \geq \mathrm{k}$$

Es decir, el error de una armónica es la mitad del error que presenta la armónica inmediata superior. Entonces, si se afina una armónica de orden superior, todas las armónicas de orden inferior tendrán el error de esta armónica superior dividido por potencias de dos según el orden de armónica que les corresponda.

Por ejemplo, si en una afinación por armónica, se tiene un error de 0.48828125Hz para la tercera armónica, entonces, para la frecuencia principal, se tiene un error de

$$\mathrm{E(1)} = \mathrm{E(4)} \bullet 2^{1-4} = 0.06103515625\mathrm{Hz}$$

Entonces, si la tercera armónica estaba dentro del umbral de percepción humana, también lo estarán todas las armónicas de orden inferior.

Para que la afinación por armónicas se pueda llevar a cabo es necesario que el espectro frecuencial relativo para el rango utilizado sea suficientemente significativo para la armónica que se espera encontrar como frecuencia dominante. Para ilustrar este punto, se han obtenido todos los rangos de frecuencia armónica para afinación fina mostrados en la tabla 14 (las graficas para encontrar la frecuencia principal para estos casos corresponden a 34, 35, 36 y 37). Véase figuras 43, 44, 45 y 46.

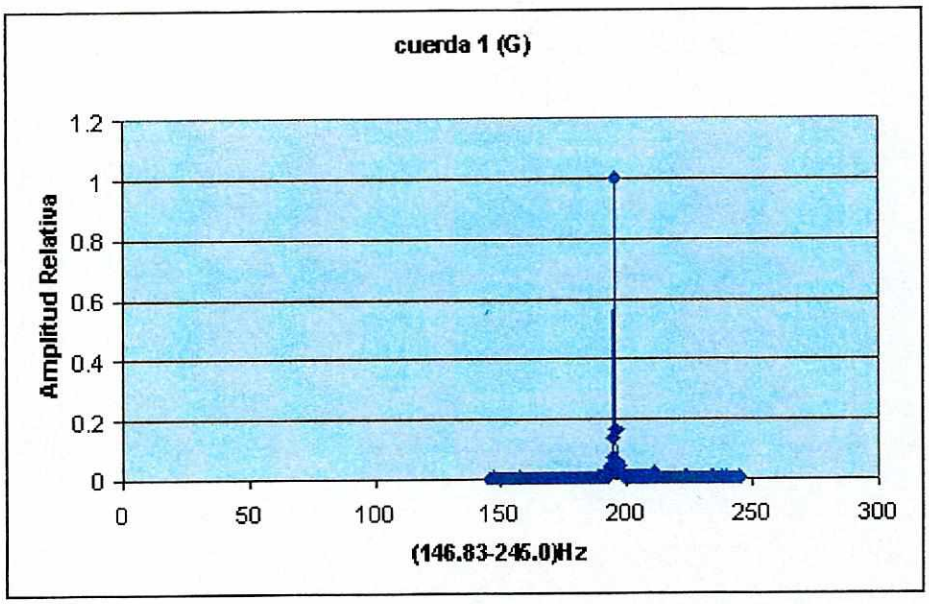


Figura 44: Espectro frecuencial relativo a la primera armónica de la cuerda 1.

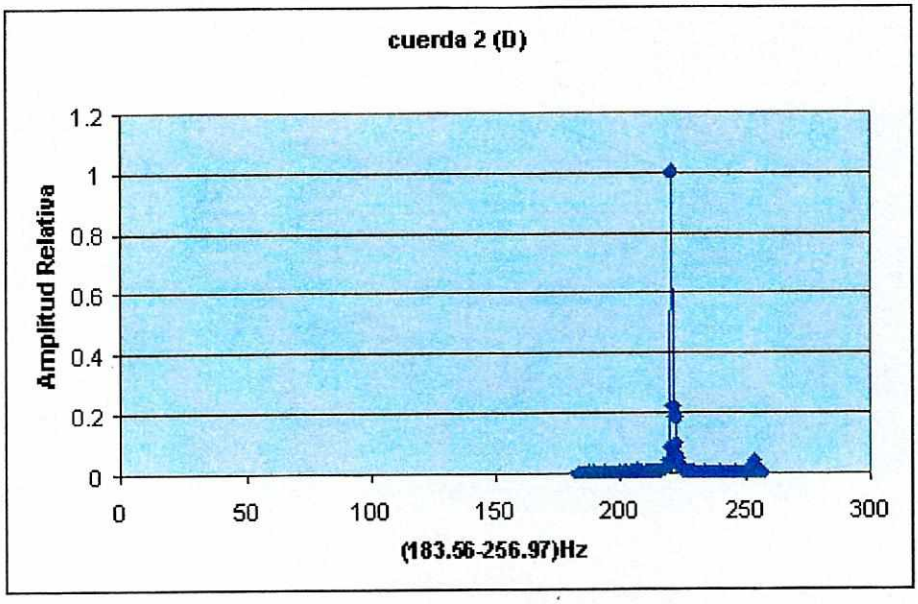


Figura 45: Espectro frecuencial relativo a la segunda armónica de la cuerda 2.

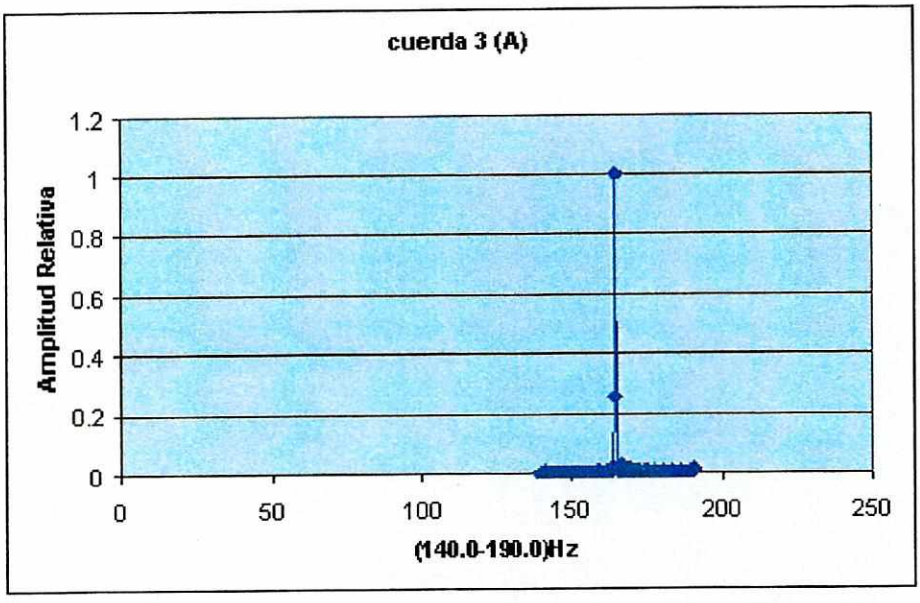


Figura 46: Espectro frecuencial relativo a la segunda armónica de la cuerda 3.

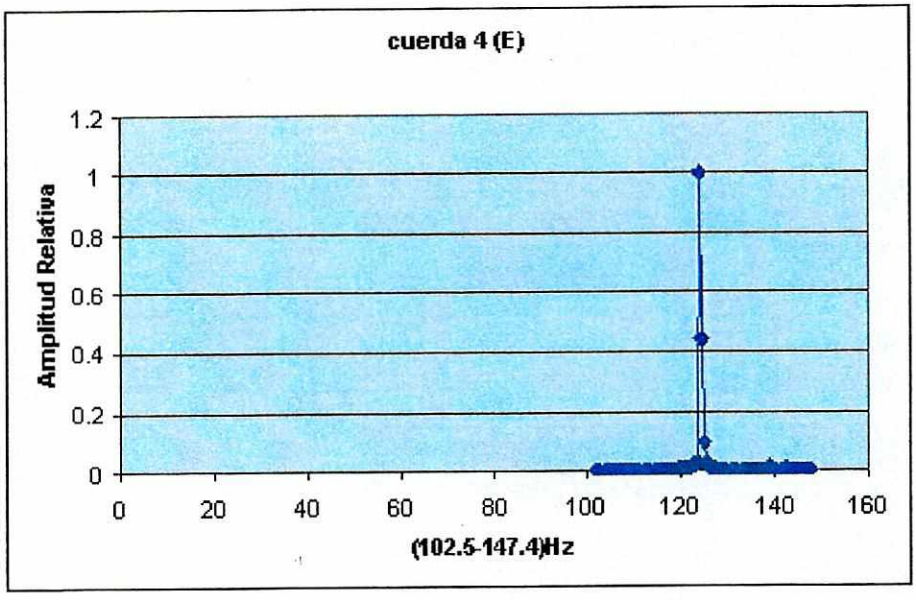


Figura 47: Espectro frecuencial relativo a la segunda armónica de la cuerda 4.

Nótese que los espectros frecuenciales relativos de armónicas son en efecto gráficas fáciles de utilizar, debido a que todos los picos vestigiales no sobrepasan el 5% de amplitud relativa. Entonces el algoritmo de detección y clasificación de la armónica basta con la detección de una amplitud relativa mínima.

VI. Servosistema

A. Proceso de afinación.

En esta sección se unificará todo el material presentado en las secciones anteriores para crear un único algoritmo el cual sea capaz de controlar todo el proceso de afinación. Un diagrama básico del servosistema a controlar se muestra en la figura 47.

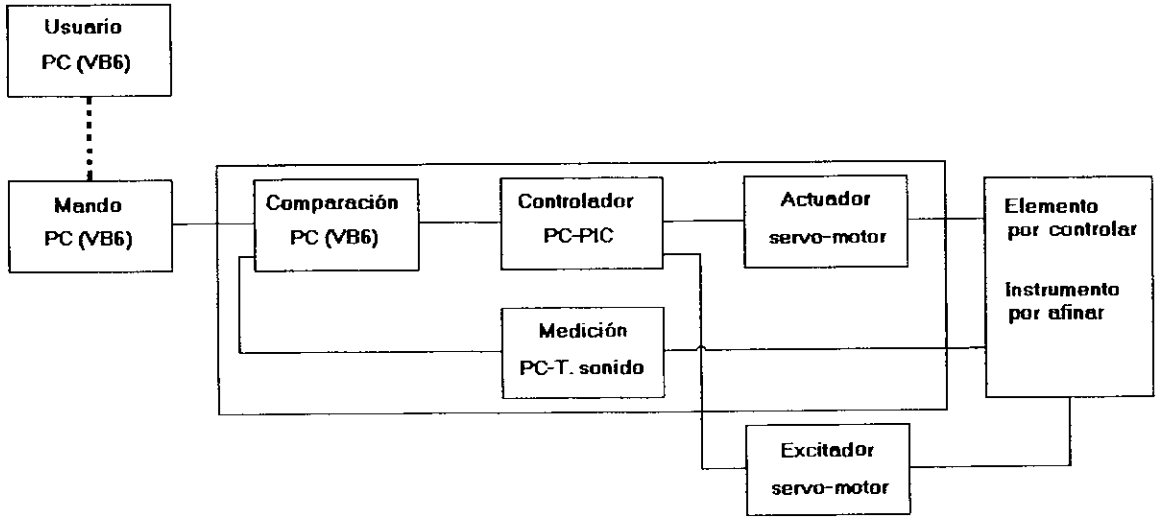


Figura 48: Servosistema que corresponde a la aplicación para la afinación automática de una guitarra-bajo.

El algoritmo de control fue diseñado de manera tal que la única intervención necesaria de parte del usuario sea la colocación de la afinación deseada.

El proceso de afinación se divide en:

- Excitación de la cuerda objetivo.
- Muestreo de la señal producida.
- Procesamiento de los datos.
- Comparación con el mando.
- Determinación de cantidad de pasos y sentido de corrección.
- Corrección.

Los excitadotes son independientes de la PC con respecto al control de la posición en la que se encuentran. Esto significa que para llevar a cabo el proceso de excitación de la cuerda objetivo, la PC únicamente debe enviar un comando serial al modulo PWM, el cual corresponda al de mover el motor objetivo a toda su velocidad en sentido antihorario (La lista de comandos se encuentra en el Anexo F).

Este simple comando garantiza que la cuerda será excitada y luego el instrumento excitador volverá a su posición original. [1]El muestreo de la señal producida empezará 460ms después de que el comando para activar el servo-motor excitador haya sido enviado(este valor fue determinado experimentalmente por prueba y error). Así, las muestras corresponderán, en su mayoría, a la señal generada por la cuerda vibrando.

Esta es una forma de minimizar la detección de frecuencias que no corresponden a la respuesta del instrumento musical.

El muestreo de la señal tendrá una duración de 2.048 segundos. Por lo tanto, si existió algún error en la transmisión de datos, ya habrán pasado 2.508 segundos para la corrección del mismo.

B. Cálculo práctico de la FFT.

El arreglo devuelto por la librería winmm.dll será ingresado a una rutina de calculo de la FFT. Esta rutina se encuentra definida dentro de una librería de tipo dll. Esta medida tiene como consecuencia el aumento de la velocidad de procesamiento de los datos.

Para utilizar esta rutina, debe definirse la función en Visual Basic 6 de la siguiente manera:

```
Declare Sub FFTDouble Lib "FFT.dll" Alias "fft_double" _
 (ByVal NumSamples As Long, ByVal InverseTransform As Boolean, _
 RealIn As Double, _
 ImagIn As Double, _
 RealOut As Double, _
 ImagOut As Double)
```

Figura 49: Declaración de la librería FFT.dll .

Esta rutina tiene definidos todos los arreglos en forma de puntero. La ventaja de esto radica en que no está definida la longitud máxima que debe tener el arreglo de datos que debe procesar. Esto permite el calculo de la FFT a cantidades realmente grandes de datos.

1. **Procesamiento de los datos.** El arreglo de datos devuelto por la función FFTDouble será modificado, de manera que los primeros 10 datos serán transformados en cero. Esto se debe a la gran contribución frecuencial introducida por la envolvente exponencial de los datos. [2]

Del conjunto de datos restante sólo será tomado en cuenta los primeros 1600, lo cual corresponde a los primeros 800Hz. Esto es debido a que ninguna de las cuerdas presenta contribuciones frecuenciales significativas sobre esa frecuencia. [2]

---

[1] ***Esto se explica más a fondo en "Control de los sistemas actuador y excitador".***

[2] ***Esto se explica más a fondo en "Determinación de la viabilidad de la detección de frecuencias asociadas con la afinación del instrumento de cuerda".***

Luego se localizará el dato de mayor magnitud en el arreglo, para luego normalizarlo con respecto al mismo. Esta frecuencia debe estar en el rango correspondiente a la frecuencia principal pronosticada para cada cuerda.. Este rango se muestra en la tabla 14. Si no se detecta una frecuencia dentro de este rango, el proceso de afinación volverá a tomar los datos.

Habiendo localizado la frecuencia principal, se obtendrá un error que corresponde a la diferencia entre la frecuencia normal deseada y la obtenida. Si la frecuencia obtenida no corresponde al rango de la frecuencia normal, pero sí al rango de la frecuencia principal, deberá ser dividida por números enteros para realizar la comparación requerida. Esto no maximizará el error debido a que luego se realizará una aproximación final por medio de análisis en las armónicas.

La aproximación final por armónicas se llevará a cabo cuando el error en la frecuencia normal sea menor que 0.244140625Hz. Esto es debido a que se vuelve imposible la comparación objetiva por debajo de este valor, puesto que la definición que se obtiene de la FFT corresponde al doble de ese valor y en el rango de frecuencias donde aparecen estas frecuencias normales, el oído humano es capaz de percibir cambios de hasta 0.17Hz. (CUERDA 4) Por esto, el proceso de afinación debe trasladar la comparación a porciones del espectro en donde se pueda contar con un error final mayor a 0.244140625Hz. [3]

C. Determinación del número de pasos y sentido de corrección.

De la correcta determinación del número de pasos depende del éxito del proceso de afinación. Para determinar dicho número de pasos, se utilizó la expresión:

$$Pasos = C\text{int}(A(\Delta f - Tolerancia)) \qquad \textbf{Ecuación (4)}$$

En donde A corresponde a un valor determinado de la tabla 2, por medio de una regresión lineal a los datos. Este valor resultó no tener la exactitud requerida y, por lo tanto, se ajustó por prueba y error para cada cuerda. La función $C\text{int}(\ )$ aproxima el argumento a un número entero. Esto es necesario, ya que el evento físico de generar pasos es entero. Existe la posibilidad que el entero convertido sea 0. En ese caso se realizará únicamente un paso.

Los valores utilizados para A en las diferentes cuerdas se muestran en la tabla 13

Cuerda	A (Incrementando tensión)	A (Decrementando tensión)
1	11.5	8
2	7	6.5
3	5	5
4	5	3

Tabla 12: Valores de A para las diferentes cuerdas.

[3] ***Esto se explica más a fondo en "Estrategia de afinación".***

Nótese que existen ganancias para los dos sentidos posibles. Esto es debido a la dependencia vestigial de la distancia angular recorrida por el actuador a la tensión que hay en la cuerda. Por esto se deben realizar más pasos cuando se aumenta la tensión que cuando se disminuye para obtener un cambio de frecuencia similar. Se realizaron pruebas experimentales aumentando y disminuyendo la amplificación en la formula de conversión de pasos. En el caso de la disminución, esto tuvo como efecto el volver el proceso de afinación mas lento, es decir, era requerido que se actuara sobre el instrumento una mayor cantidad de veces, pero se lograba finalmente la afinación correcta. En el caso del aumento de la variable, el servosistema se volvía finalmente inestable, y debía ser abortado el proceso de afinación para evitar daños al instrumento. Los valores mostrados en la tabla 13 corresponden al tiempo óptimo de afinación.

El sentido de la corrección se obtiene simplemente analizando el signo que acompaña a Δf .

D. Control de errores.

Los errores son parte de la operación normal de un sistema automático. Por esto todas las rutinas de control del mismo deben contemplar la posibilidad de que exista algún error. Errores pueden ser causados por:

- Rotura de un instrumento excitador.
- Mala conexión de un detector.
- Error en las comunicaciones seriales.
- Interacción humana.
- Otros.

Es imperativo que el control del servosistema pueda, de alguna manera, compensar el error ocurrido. En los casos en donde el error no pueda ser compensado, debe evitar a toda costa que el instrumento sea dañado por la interpretación errónea de los datos.

El control digital en cuestión consta de una serie de rutinas, las cuales no permiten que el error ocurrido trascienda a través de todo el servosistema, protegiendo así al instrumento y permitiendo que el proceso de afinación continúe normalmente. No existe ninguna rutina que aborte por completo el proceso.

1. Rutinas de seguridad.

a. Microcontrolador. El microcontrolador cuenta con una rutina accionada por un reloj interno, la cual manda un comando para apagar los servo-motores en caso de que el valor de los sensores no cambie en siete segundos. Esta medida está diseñada para proteger al instrumento de un exceso de tensión en sus cuerdas a causa de algún error en la transmisión de algún comando hacia o en el módulo PWM. También el microcontrolador es el encargado de mantener los instrumentos excitadores en una posición de 180 grados con respecto a la posición de las cuerdas. Esto vuelve al sistema ajeno a la PC en un sistema independiente.

b. Programa principal de afinación. Este programa cuenta con varias rutinas de seguridad.

- Rutina de control de rangos de frecuencia.
- Número de pasos máximo. Rutinas de control de tiempo.
- Medidas de seguridad varias.

1) Rutina de control de rangos de frecuencia. Esta rutina revisa si la frecuencia normal, y la frecuencia principal detectadas se encuentran dentro del rango correspondiente a cada cuerda. Si no es así, entonces se llama a una nueva adquisición de datos.

2) Número de pasos máximo. El número de pasos por un actuador es controlado a través de la diferencia máxima de frecuencia, que deriva del control de rangos de frecuencia. De esta manera, el número de pasos no excederá el límite fijado para cada cuerda.

3) Rutinas de control de tiempo. Entre comandos que requieran una acción, es decir, entre correcciones a la tensión de una cuerda o entre excitaciones de las mismas, existen tiempos fijados de manera que nunca se ejecuten dos eventos de este tipo. Estos tiempos también actúan en caso de la llamada consecutiva a alguna función que requiera una acción, separando las llamadas, así la acción se ejecuta consecutivamente sin interferencia de llamadas anteriores o posteriores.

4) Medidas de seguridad varias. Estas medidas consisten en validaciones de datos en partes claves del proceso de control. La mayoría de las validaciones se lleva a cabo en la comunicación serial.

VII. Conclusiones

- El torque mínimo necesario para un actuador modificador de tensión de una cuerda de guitarra bajo tipo "precision" es de $3.60 Kg \bullet cm$
- La utilización de un eje cardanico y esponja en la carcasa del servo-motor actuador contrarrestan los efectos de las anomalías en la construcción de la clavija de una guitarra bajo tipo "precision" cuando a ésta se le aplica una rotación continua.
- Es suficiente, para evitar la rotación continua de los actuadores colocados en las clavijas de una guitarra bajo como se muestra en la figura 8 , la sujeción entre sí de los mismos.
- Una forma de lograr la emulación mecánica de una mano humana tocando una cuerda con una púa, es la que se muestra en la figura 11
- La mejor posición para colocar instrumentos para la excitación de las cuerdas de una guitarra-bajo tipo precision es sobre el cuerpo de la misma.
- La inercia presente en un servo-motor modificado para rotación continua puede ser reducida considerablemente cambiando la señal de mando.
- La amplitud relativa con la que se registre cada una de las frecuencias generadas por una cuerda que vibra, será dependiente únicamente de la forma que la cuerda tenga justo antes de que comience la vibración en la misma.
- Factores tales como volumen, masa por unidad de volumen, fricciones, ondas que viajen paralelamente a la longitud de la cuerda y resonancia que esta pueda tener con el medio generan contribuciones frecuenciales despreciables en el espectro frecuencial relativo.
- El espectro relativo a causa de la contribución DC de la señal a causa de la cuantización se vuelve despreciable sobre los 5Hz
- Las contribuciones frecuenciales de la señal generada por una guitarra bajo se vuelven despreciables sobre los 1200Hz
- Cada cuerda de una guitarra bajo cuenta con una frecuencia de oscilación de amplitud relativa máxima (frecuencia principal) la cual puede o no corresponder a la frecuencia normal.
- El rango del espectro frecuencial de la señal generada por una guitarra bajo en el cual sólo existirá una armónica está dado por:

$$RangoArmonica = \left(fN * NE - \frac{fN}{2}, fN * NE + \frac{fN}{2} \right)$$

en donde

fN = Frecuencial normal
NE = Número entero

- La utilización de un espectro con resolución de 0.48828125Hz permite la afinación de una guitarra bajo para la cual el oído humano no es capaz de percibir el error remanente con respecto a una afinación matemáticamente prefecta.

VIII. Recomendaciones

Manteniendo los mismos métodos y orientaciones presentados en este trabajo, futuras investigaciones pueden orientarse a la optimización del procedimiento de procesamiento de datos con la finalidad de implementar esta aplicación completamente dentro de un microcontrolador.

Para ello se propone se investigue acerca del espectro frecuencial relativo obtenido por la Transformada Del Coseno, en la cual se manejan únicamente números reales, de manera que el procesamiento se vuelve menos complicado.

También se proponen investigaciones orientadas a la optimización de las partes mecánicas de la aplicación.

IX. Bibliografía

Dorf,R, R. Bishop.1998. *Modern Control Systems*. EEUU, Addison Wesley. 856pp.

Nilsson, j. S., Riedel. 2001. *Circuitos Eléctricos.* México, Prentice Hall. 1029pp.

Oppenheim, A. 2000. *Tratamiento de señales en tiempo discreto.* México, Prentice Hall. 873pp.

Sedra, A. 1998. *Circuitos Mircoelectrónicos.* México, Oxford. 1237pp.

ANEXO A

Funciones de la de la librería winmm.dll

Private Declare Function waveInOpen Lib "winmm.dll" (lphWaveIn As Long, ByVal uDeviceID As Long, lpFormat As WAVEFORMAT, ByVal dwCallback As Long, ByVal dwInstance As Long, ByVal dwFlags As Long) As LongDeclare Function mixerGetDevCaps Lib "winmm dll" Alias "mixerGetDevCapsA" (ByVal uMxId As Long, ByVal pmxcaps As MIXERCAPS, ByVal cbmxcaps As Leng) As Long

Declare Function mixerOpen Lib "winmm.dll" Alias "mixerOpen" (phmx As Long, ByVal uMxId As Long, ByVal dwCallback As Long, ByVal dwInstance As Long, ByVal fdwOpen As Leng) As Leng

Declare Function mixerClose Lib "winmm.dll" Alias "mixerClose" (ByVal hmx As Long) As Long
Declare Function mixerMessage Lib "winmm.dll" Alias "mixerMessage" (ByVal hmx As Long, ByVal uMsg As Long, ByVal dwParam1 As Leng, ByVal dwPsram2 As Long) As Leng
Declare Function mixerGetLineInfe Lib "winmm.dll" Alias "mixerGetLineInfoA" (ByVal hmxobj As Long, pmxl As MIXERLINE, ByVal fdwInfe As Long) As Long
Declare Function mixerGetID Lib "winmm.dll" Alias "mixerGetID" (ByVal hmxobj As Long, pumxID As Long, ByVal fdwId As Leng) As Long
Declare Function mixerGetLineControls Lib "winmm.dll" Alias "mixerGetLineControlsA" (ByVal hmxobj As Leng, pmxlc As MIXERLINECONTROLS, ByVal fdwControls As Long) As Long
Declare Function mixerGetControlDetails Lib "winmm.dll" Alias "mixerGetControlDetailsA" (ByVal hmxobj As Long, pmxcd As MIXERCONTROLDETAILS, ByVal fdwDetails As Leng) As Long
Declare Function mixerSetControlDetails Lib "winmm.dll" Alias "mixerSetControlDetailsa" (ByVal hmxobj As Long, pmxcd As MIXERCONTROLDETAILS, ByVal fdwDetails As Leng) As Leng
Declare Function joyGetPosEx Lib "winmm.dll" Alias "joyGetPosEx" (ByVal uJoyID As Long, pji As JOYINFOEX) As Leng
Declare Function midiStreamOpen Lib "winmm.dll" Alias "midiStreamOpen" (phms As Long, puDeviceID As Leng, ByVal cMidi As Long, ByVal dwCallback As Long, ByVal dwInstance As Long, ByVal fdwOpen As Long) As Leng

Declare Function midiStreamClose Lib "winmm dll" Alias "midiStreamClose" (ByVal hms As Leng) As Leng
Declare Function midiStreamProperty Lib "winmm.dll" Alias "midiStreamProperty" (ByVal hms As Long, lppropdata As Byte, ByVal dwProperty As Leng) As Leng
Declare Function midiStreamPosition Lib "winmm.dll" Alias "midiStreamPosition" (ByVal hms As Long, lpmmt As MMTIME, ByVal cbmmt As Long) As Long
Declare Function midiStreamOut Lib "winmm.dll" Alias "midiStreamOut" (ByVal hms As Leng, pmn As MIDIHDR, ByVal cbmh As Leng) As Long
Declare Function midiStreamPause Lib "winmm.dll" Alias "midiStreamPause" (ByVal hms As Long) As Leng
Declare Function midiStreamRestart Lib "winmm.dll" Alias "midiStreamRestart" (ByVal hms As Leng) As Leng
Declare Function midiStreamStop Lib "winmm.dll" Alias "midiStreamStop" (ByVal hms As Long) As Long
Declare Function midiConnect Lib "winmm.dll" Alias "midiConnect" (ByVal hmi As Long, ByVal hmo As Long, pReserved As Any) As Long
Declare Function midiDisconnect Lib "winmm.dll" Alias "midiDisconnect" (ByVal hmi As Leng, ByVal hmo As Long, pReserved As Any) As Leng
Declare Function CloseDriver Lib "winmm.dll" Alias "CloseDriver" (ByVal hDriver As Long, ByVal lParam1 As Long, ByVal lParam2 As Long) As Leng
Declare Function OpenDriver Lib "winmm.dll" Alias "OpenDriver" (ByVal szDriverName As String, ByVal szSectionName As String, ByVal lParam2 As Leng) As Leng
Declare Function SendDriverMessage Lib "winmm dll" Alias "SendDriverMessage" (ByVal hDriver As Long, ByVal message As Leng, ByVal lParam1 As Long, ByVal lParam2 As Long) As Long
Declare Function DrvGetModuleHandle Lib "winmm dll" Alias "DrvGetModuleHandle" (ByVal hDriver As Leng) As Leng
Declare Function GetDriverModuleHandle Lib "winmm dll" Alias "GetDriverModuleHandle" (ByVal hDriver As Long) As Long
Declare Function DefDriverProc Lib "winmm dll" Alias "DefDriverProc" (ByVal dwDriverIdentifier As Leng, ByVal hdrvr As Long, ByVal uMsg As Long, ByVal lParam1 As Long, ByVal lParam2 As Long) As Leng

Declare Function mmsystemGetVersion Lib "winmm.dll" Alias "mmsystemGetVersion" () As Long
Declare Sub OutputDebugStr Lib "winmm.dll" Alias "OutputDebugStr" (ByVal lpszOutputString As String)
Declare Function waveOutGetNumDevs Lib "winmm dll" Alias "waveOutGetNumDevs" () As Long
Declare Function waveOutGetDevCaps Lib "winmm.dll" Alias "waveOutGetDevCapsA" (ByVal uDeviceID As Leng, lpCaps As WAVEOUTCAPS, ByVal uSize As Long) As Leng
Declare Function waveOutGetVolume Lib "winmm dll" Alias "waveOutGetVeiume" (ByVal uDeviceID As Leng, lpdwVolume As Long) As Long
Declare Function waveOutSetVeiume Lib "winmm.dll" Alias "waveOutSetVolume" (ByVal uDeviceID As Long, ByVal dwVeiume As Leng) As Long
Declare Function waveOutGetErrorText Lib "winmm.dll" Alias "waveOutGetErrorTextA" (ByVal err As Long, ByVal lpText As String, ByVal uSize As Leng) As Long
Declare Function waveOutOpen Lib "winmm.dll" Alias "waveOutOpen" (lphWaveOut As Long, ByVal uDeviceID As Long, lpFormat As WAVEFORMAT, ByVal dwCallback As Long, ByVal dwInstance As Long, ByVal dwFlags As Long) As Long

Declare Function waveOutClose Lib "winmm dll" Alias "waveOutClose" (ByVal hWaveOut As Long) As Long
Declare Function waveOutPrepareHeader Lib "winmm dll" Alias "waveOutPrepareHeader" (ByVal hWaveOut As Long, lpWaveOutHdr As WAVEHDR, ByVal uSize As Long) As Long
Declare Function waveOutUnprepareHeader Lib "winmm dll" Alias "waveOutUnprepareHeader" (ByVal hWaveOut As Long, lpWaveOutHdr As WAVEHDR, ByVal uSize As Long) As Long
Declare Function waveOutWrite Lib "winmm.dll" Alias "waveOutWrite" (ByVal hWaveOut As Long, lpWaveOutHdr As WAVEHDR, ByVal uSize As Long) As Long
Declare Function waveOutPause Lib "winmm.dll" Alias "waveOutPause" (ByVal hWaveOut As Long) As Long
Declare Function waveOutRestart Lib "winmm.dll" Alias "waveOutRestart" (ByVal hWaveOut As Long) As Long
Declare Function waveOutReset Lib "winmm.dll" Alias "waveOutReset" (ByVal hWaveOut As Long) As Long
Declare Function waveOutBreakLoop Lib "winmm.dll" Alias "waveOutBreakLoop" (ByVsi hWaveOut As Long) As Long
Declare Function waveOutGetPosition Lib "winmm.dll" Alias "waveOutGetPosition" (ByVal hWsveOut As Long, lpInfo As MMTIME, ByVal uSize As Leng) As Leng
Declare Function waveOutGetPitch Lib "winmm.dll" Alias "waveOutGetPitch" (ByVal hWaveOut As Leng, lpdwPitch As Long) As Long
Declare Function waveOutSetPitch Lib "winmm.dll" Alias "waveDutSetPitch" (ByVal hWaveOut As Long, ByVal dwPitch As Leng) As Long
Declare Function waveOutGetPlaybackRate Lib "winmm.dll" Alias "waveOutGetPlaybackRate" (ByVal hWaveOut As Leng, lpdwRate As Long) As Long
Declare Function waveOutSetPlaybackRate Lib "winmm dll" Alias "waveOutSetPlaybackRate" (ByVsi hWaveOut As Long, ByVal dwRate As Long) As Long
Declare Function waveOutGetID Lib "winmm.dll" Alias "waveOutGetID" (ByVal hWaveOut As Leng, lpuDeviceID As Long) As Leng
Declare Function waveOutMessage Lib "winmm.dll" Alias "waveOutMessage" (ByVal hWaveOut As Long, ByVal msg As Long, ByVal dw1 As Long, ByVal dw2 As Long) As Long
Declare Function waveInGetNumDevs Lib "winmm dll" Alias "waveInGetNumDevs" () As Long
Declare Function waveInGetDevCaps Lib "winmm.dll" Alias "waveInGetDevCapsA" (ByVal uDeviceID As Long, lpCaps As WAVEINCAPS, ByVal uSize As Leng) As Leng
Declare Function waveInGetErrorText Lib "winmm.dll" Alias "waveInGetErrorTextA" (ByVsi err As Long, ByVal lpText As String, ByVal uSize As Leng) As Long
Declare Function waveInOpen Lib "winmm.dll" Alias "waveInOpen" (lphWaveIn As Long, ByVal uDeviceID As Long, lpFormat As WAVEFORMAT, ByVal dwCallback As Long, ByVal dwInstance As Long, ByVal dwFlags As Long) As Long

Declare Function waveInClose Lib "winmm.dll" Alias "waveInClose" (ByVal hWaveIn As Long) As Long
Declare Function waveInPrepareHeader Lib "winmm.dll" Alias "waveInPrepareHeader" (ByVal hWaveIn As Long, lpWaveInHdr As WAVEHDR, ByVal uSize As Long) As Long
Declare Function waveInUnprepareHeader Lib "winmm dll" Alias "waveInUnprepareHeader" (ByVsi hWaveIn As Leng, lpWaveInHdr As WAVEHDR, ByVal uSize As Long) As Long
Declare Function waveInAddBuffer Lib "winmm.dll" Alias "waveInAddBuffer" (ByVal hWaveIn As Long, lpWaveInHdr As WAVEHDR, ByVal uSize As Long) As Leng
Declare Function waveInStart Lib "winmm.dll" Alias "waveInStart" (ByVal hWaveIn As Leng) As Long
Declare Function waveInStop Lib "winmm.dll" Alias "waveInStop" (ByVal hWaveIn As Long) As Leng
Declare Function waveInReset Lib "winmm.dll" Alias "waveInReset" (ByVal hWaveIn As Long) As Leng
Declare Function waveInGetPosition Lib "winmm.dll" Alias "waveInGetPosition" (ByVal hWaveIn As Leng, lpInfo As MMTIME, ByVal uSize As Leng) As Long
Declare Function waveInGetID Lib "winmm.dll" Alias "waveInGetID" (ByVsi hWaveIn As Long, lpuDeviceID As Long) As Long
Declare Function waveInMessage Lib "winmm.dll" Alias "waveInMessage" (ByVal hWaveIn As Long, ByVal msg As Long, ByVal dw1 As Long, ByVsi dw2 As Long) As Leng
Declare Function midiOutGetDevCaps Lib "winmm.dll" Alias "midiOutGetDevCapsA" (ByVal uDeviceID As Long, lpCaps As MIDIOUTCAPS, ByVal uSize As Leng) As Long
Declare Function midiOutGetVolume Lib "winmm.dll" Alias "midiOutGetVolume" (ByVal uDeviceID As Long, lpdwVolume As Long) As Long
Declare Function midiOutSetVsIume Lib "winmm dll" Alias "midiOutSetVolume" (ByVal uDeviceID As Long, ByVal dwVolume As Long) As Long
Declare Function midiOutGetErrorTsxt Lib "winmm.dll" Alias "midiOutGetErrorTextA" (ByVal srr As Long, ByVal lpText As String, ByVal uSize As Long) As Long
Declare Function midiOutOpen Lib "winmm.dll" Alias "midiOutOpen" (lphMidiOut As Long, ByVal uDeviceID As Long, ByVal dwCsilback As Long, ByVal dwInstance As Leng, ByVal dwFlags As Long) As Leng

Declare Function midiOutClose Lib "winmm.dll" Alias "midiOutClose" (ByVal hMidiOut As Leng) As Leng
Declare Function midiOutPrepareHeader Lib "winmm.dll" Alias "midiOutPrepareHeader" (ByVal hMidiOut As Long, lpMidiOutHdr As MIDIHDR, ByVal uSize As Long) As Long
Declare Function midiOutUnprepareHeader Lib "winmm.dll" Alias "midiOutUnprepareHeader" (ByVal hMidiOut As Long, lpMidiOutHdr As MIDIHDR, ByVal uSize As Long) As Long
Declare Function midiOutShortMsg Lib "winmm.dll" Alias "midiOutShortMsg" (ByVal hMidiOut As Leng, ByVal dwMsg As Long) As Long
Declare Function midiOutLongMsg Lib "winmm.dll" Alias "midiOutLongMsg" (ByVal hMidiOut As Long, lpMidiDutHdr As MIDIHDR, ByVal uSize As Long) As Long
Declare Function midiOutReset Lib "winmm.dll" Alias "midiOutReset" (ByVal hMidiOut As Long) As Leng
Declare Function midiOutCachePatches Lib "winmm.dll" Alias "midiOutCachePatches" (ByVal hMidiOut As Leng, ByVal uBank As Long, lpPatchArray As Long, ByVal uFlags As Leng) As Leng
Declare Function midiOutCacheDrumPatches Lib "winmm.dll" Alias "midiOutCacheDrumPatches" (ByVal hMidiOut As Long, ByVal uPatch As Long, lpKeyArray As Leng, ByVal uFlags As Leng) As Leng

Declare Function midiOutGetID Lib "winmm.dll" Alias "midiOutGetID" (ByVal hMidiOut As Long, lpuDeviceID As Long) As Leng
Declare Function midiOutMessage Lib "winmm.dll" Alias "midiOutMessage" (ByVal hMidiOut As Long, ByVal msg As Long, ByVal dw1 As Long, ByVal dw2 As Long) As Leng
Declare Function midiInGetNumDevs Lib "winmm dll" Alias "midiInGetNumDeve" () As Long
Declare Function midiInGetDevCaps Lib "winmm dll" Alias "midiinGetDevCapsA" (ByVal uDeviceID As Long, lpCaps As MIDIINCAPS, ByVal uSize As Long) As Long
Declare Function midiInGetErrorTsxt Lib "winmm dll" Alias "midiInGetErrorTextA" (ByVal err As Leng, ByVal lpText As String, ByVal uSize As Leng) As Long
Declare Function midiInOpen Lib "winmm.dll" Alias "midiInOpen" (lphMidiIn As Leng, ByVal uDeviceID As Leng, ByVal dwCallback As Leng, ByVal dwInstance As Leng, ByVal dwFlags As Long) As Long
Declare Function midiInClose Lib "winmm.dll" Alias "midiInClose" (ByVal hMidiIn As Long) As Leng
Declare Function midiInPrepareHeader Lib "winmm dll" Alias "midiInPrepareHeader" (ByVal hMidiIn As Long, lpMidiInHdr As MIDIHDR, ByVal uSize As Long) As Long
Declare Function midiInUnprepareHeader Lib "winmm dll" Alias "midiInUnprepareHeader" (ByVal hMidiIn As Leng, lpMidiInHdr As MIDIHDR, ByVal uSize As Long) As Long
Declare Function midiInAddBuffer Lib "winmm.dll" Alias "midiInAddBuffer" (ByVal hMidiIn As Leng, lpMidiInHdr As MIDIHDR, ByVal uSize As Long) As Long
Declare Function midiInStart Lib "winmm dll" Alias "midiInStart" (ByVal hMidiIn As Long) As Long
Declare Function midiInStop Lib "winmm.dll" Alias "midiInStop" (ByVal hMidiIn As Long) As Long
Declare Function midiInReset Lib "winmm dll" Alias "midiInReset" (ByVal hMidiIn As Long) As Long
Declare Function midiInGetID Lib "winmm dll" Alias "midiInGetID" (ByVal hMidiIn As Long, lpuDeviceID As Long) As Long
Declare Function midiInMessage Lib "winmm.dll" Alias "midiInMessage" (ByVal hMidiIn As Leng, ByVsi msg As Long, ByVal dw1 As Leng, ByVal dw2 As Leng) As Long
Declare Function auxGetNumDevs Lib "winmm.dll" Alias "auxGetNumDevs" () As Long
Declare Function auxGetDevCaps Lib "winmm.dll" Alias "auxGetDevCapsA" (ByVal uDeviceID As Long, lpCaps As AUXCAPS, ByVal uSize As Long) As Leng
Declare Function auxSetVolume Lib "winmm.dll" Alias "auxSetVsiume" (ByVal uDeviceID As Long, ByVal dwVsiume As Long) As Leng
Declare Function suxGetVolume Lib "winmm.dll" Alias "auxGetVolume" (ByVal uDeviceID As Long, lpdwVsiume As Long) As Long
Declare Function auxOutMessage Lib "winmm.dll" Alias "auxOutMessage" (ByVal uDeviceID As Long, ByVal msg As Long, ByVal dw1 As Leng, ByVal dw2 As Long) As Long
Declare Function timeGetSystemTime Lib "winmm dll" Alias "timeGetSystemTime" (lpTime As MMTIME, ByVal uSize As Long) As Long
Declare Function timeGetTime Lib "winmm.dll" Alias "umeGetTime" () As Long
Declare Function umeSetEvent Lib "winmm.dll" Alias "timeSet(Event" (ByVal uDelay As Long, ByVsi uResolution As Long, ByVal lpFunction As Leng, ByVal dwUser As Leng, ByVal uFlags As Long) As Long

```
Declare Function timeKillEvent Lib "winmm.dll" Alias "timeKillEvent" (ByVal uID As Long) As Long
Declare Function timeGetDevCaps Lib "winmm.dll" Alias "timeGetDevCaps" (lpTimeCaps As TIMECAPS, ByVal uSize As Long) As Long
Declare Function timeBeginPeriod Lib "winmm.dll" Alias "timeBeginPeriod" (ByVal uPeriod As Long) As Long
Declare Function timeEndPeriod Lib "winmm.dll" Alias "timeEndPeriod" (ByVal uPeriod As Long) As Long
Declare Function joyGetThreshold Lib "winmm.dll" Alias "joyGetThreshold" (ByVal id As Long, lpuThreshold As Long) As Long
Declare Function joyReleaseCapture Lib "winmm.dll" Alias "joyReleaseCapture" (ByVal id As Long) As Long
Declare Function joySetCapture Lib "winmm.dll" Alias "joySetCapture" (ByVal hwnd As Long, ByVal uID As Long, ByVal uPeriod As Long, ByVal bChanged As Long) As Long
Declare Function joySetThreshold Lib "winmm.dll" Alias "joySetThreshold" (ByVal id As Long, ByVal uThreshold As Long) As Long
Declare Function mmioStringToFOURCC Lib "winmm.dll" Alias "mmioStringToFOURCCA" (ByVal sz As String, ByVal uFlags As Long) As Long
Declare Function mmioOpen Lib "winmm.dll" Alias "mmioOpenA" (ByVal szFileName As String, lpmmioinfo As MMIOINFO, ByVal dwOpenFlags As Long) As Long
Declare Function mmioRename Lib "winmm.dll" Alias "mmioRenameA" (ByVal szFileName As String, ByVal SzNewFileName As String, lpmmioinfo As MMIOINFO, ByVal dwRenameFlags As Long) As Long

Declare Function mmioClose Lib "winmm.dll" Alias "mmioClose" (ByVal hmmio As Long, ByVal uFlags As Long) As Long
Declare Function mmioRead Lib "winmm.dll" Alias "mmioRead" (ByVal hmmio As Long, ByVal pch As String, ByVal cch As Long) As Long
Declare Function mmioWrite Lib "winmm.dll" Alias "mmioWrite" (ByVal hmmio As Long, ByVal pch As String, ByVal cch As Long) As Long
Declare Function mmioSeek Lib "winmm.dll" Alias "mmioSeek" (ByVal hmmio As Long, ByVal lOffset As Long, ByVal iOrigin As Long) As Long
Declare Function mmioGetInfo Lib "winmm.dll" Alias "mmioGetInfo" (ByVal hmmio As Long, lpmmioinfo As MMIOINFO, ByVal uFlags As Long) As Long
Declare Function mmioSetInfo Lib "winmm.dll" Alias "mmioSetInfo" (ByVal hmmio As Long, lpmmioinfo As MMIOINFO, ByVal uFlags As Long) As Long
Declare Function mmioSetBuffer Lib "winmm.dll" Alias "mmioSetBuffer" (ByVal hmmio As Long, ByVal pchBuffer As String, ByVal cchBuffer As Long, ByVal uFlags As Long) As Long
Declare Function mmioFlush Lib "winmm.dll" Alias "mmioFlush" (ByVal hmmio As Long, ByVal uFlags As Long) As Long
Declare Function mmioAdvance Lib "winmm.dll" Alias "mmioAdvance" (ByVal hmmio As Long, lpmmioinfo As MMIOINFO, ByVal uFlags As Long) As Long
Declare Function mmioSendMessage Lib "winmm.dll" Alias "mmioSendMessage" (ByVal hmmio As Long, ByVal uMsg As Long, ByVal lParam1 As Long, ByVal lParam2 As Long) As Long
Declare Function mmioDescend Lib "winmm.dll" Alias "mmioDescend" (ByVal hmmio As Long, lpck As MMCKINFO, lpckParent As MMCKINFO, ByVal uFlags As Long) As Long
Declare Function mmioAscend Lib "winmm.dll" Alias "mmioAscend" (ByVal hmmio As Long, lpck As MMCKINFO, ByVal uFlags As Long) As Long
Declare Function mmioCreateChunk Lib "winmm.dll" Alias "mmioCreateChunk" (ByVal hmmio As Long, lpck As MMCKINFO, ByVal uFlags As Long) As Long
Declare Function mciSendCommand Lib "winmm.dll" Alias "mciSendCommandA" (ByVal wDeviceID As Long, ByVal uMessage As Long, ByVal dwParam1 As Long, ByVal dwParam2 As Any) As Long

Declare Function mciGetCreatorTask Lib "winmm.dll" Alias "mciGetCreatorTask" (ByVal wDeviceID As Long) As Long
Declare Function mciGetDeviceID Lib "winmm.dll" Alias "mciGetDeviceIDA" (ByVal lpstrName As String) As Long
Declare Function mciGetDeviceIDFromElementID Lib "winmm.dll" Alias "mciGetDeviceIDFromElementIDA" (ByVal dwElementID As Long, ByVal lpstrType As String) As Long
Declare Function mciGetErrorString Lib "winmm.dll" Alias "mciGetErrorStringA" (ByVal dwError As Long, ByVal lpstrBuffer As String, ByVal uLength As Long) As Long
Declare Function mciGetYieldProc Lib "winmm" Alias "mciGetYieldProc" (ByVal mciId As Long, pdwYieldData As Long) As Long
Declare Function mciSetYieldProc Lib "winmm" Alias "mciSetYieldProc" (ByVal mciId As Long, ByVal fpYieldProc As Long, ByVal dwYieldData As Long) As Long
Declare Function midiOutGetNumDevs Lib "winmm" Alias "midiOutGetNumDevs" () As Integer
Declare Function mmioInstallIOProcA Lib "winmm" Alias "mmioInstallIOProcA" (ByVal fccIOProc As String * 4, ByVal pIOProc As Long, ByVal dwFlags As Long) As Long
```

ANEXO B

A. Frecuencias para escala igualmente templada

$A_4 = 440$ Hz
Velocidad del sonido = 345 m/s

ota	Frecuencia(Hz)	Longitud de onda(cm)
C_0	16.35	2100.
$C^{\#}_0/D^{b}_0$	17.32	1990.
D_0	18.35	1870.
$D^{\#}_0/E^{b}_0$	19.45	1770.
E_0	20.60	1670.
F_0	21.83	1580.
$F^{\#}_0/G^{b}_0$	23.12	1490.
G_0	24.50	1400.
$G^{\#}_0/A^{b}_0$	25.96	1320.
A_0	27.50	1250.
$A^{\#}_0/B^{b}_0$	29.14	1180.
B_0	30.87	1110.
C_1	32.70	1050.
$C^{\#}_1/D^{b}_1$	34.65	996.
D_1	36.71	940.
$D^{\#}_1/E^{b}_1$	38.89	887.
E_1	41.20	837.
F_1	43.65	790.
$F^{\#}_1/G^{b}_1$	46.25	746.
G_1	49.00	704.
$G^{\#}_1/A^{b}_1$	51.91	665.
A_1	55.00	627.
$A^{\#}_1/B^{b}_1$	58.27	592.
B_1	61.74	559.
C_2	65.41	527.
$C^{\#}_2/D^{b}_2$	69.30	498.
D_2	73.42	470.
$D^{\#}_2/E^{b}_2$	77.78	444.
E_2	82.41	419.
F_2	87.31	395.
$F^{\#}_2/G^{b}_2$	92.50	373.
G_2	98.00	352.
$G^{\#}_2/A^{b}_2$	103.83	332.
A_2	110.00	314.
$A^{\#}_2/B^{b}_2$	116.54	296.
B_2	123.47	279.
C_3	130.81	264.
$C^{\#}_3/D^{b}_3$	138.59	249.
D_3	146.83	235.
$D^{\#}_3/E^{b}_3$	155.56	222.
E_3	164.81	209.
F_3	174.61	198.
$F^{\#}_3/G^{b}_3$	185.00	186.
G_3	196.00	176.
$G^{\#}_3/A^{b}_3$	207.65	166.
A_3	220.00	157.
$A^{\#}_3/B^{b}_3$	233.08	148.
B_3	246.94	140.
C_4	261.63	132.
$C^{\#}_4/D^{b}_4$	277.18	124.
D_4	293.66	117.
$D^{\#}_4/E^{b}_4$	311.13	111.
E_4	329.63	105.
F_4	349.23	98.8
$F^{\#}_4/G^{b}_4$	369.99	93.2
G_4	392.00	88.0
$G^{\#}_4/A^{b}_4$	415.30	83.1
A_4	440.00	78.4

$A^{\#}_4/B^{b}_4$	466.16	74.0
B_4	493.88	69.9
C_5	523.25	65.9
$C^{\#}_5/D^{b}_5$	554.37	62.2
D_5	587.33	58.7
$D^{\#}_5/E^{b}_5$	622.25	55.4
E_5	659.26	52.3
F_5	698.46	49.4
$F^{\#}_5/G^{b}_5$	739.99	46.6
G_5	783.99	44.0
$G^{\#}_5/A^{b}_5$	830.61	41.5
A_5	880.00	39.2
$A^{\#}_5/B^{b}_5$	932.33	37.0
B_5	987.77	34.9
C_6	1046.50	33.0
$C^{\#}_6/D^{b}_6$	1108.73	31.1
D_6	1174.66	29.4
$D^{\#}_6/E^{b}_6$	1244.51	27.7
E_6	1318.51	26.2
F_6	1396.91	24.7
$F^{\#}_6/G^{b}_6$	1479.98	23.3
G_6	1567.98	22.0
$G^{\#}_6/A^{b}_6$	1661.22	20.8
A_6	1760.00	19.6
$A^{\#}_6/B^{b}_6$	1864.66	18.5
B_6	1975.53	17.5
C_7	2093.00	16.5
$C^{\#}_7/D^{b}_7$	2217.46	15.6
D_7	2349.32	14.7
$D^{\#}_7/E^{b}_7$	2489.02	13.9
E_7	2637.02	13.1
F_7	2793.83	12.3
$F^{\#}_7/G^{b}_7$	2959.96	11.7
G_7	3135.96	11.0
$G^{\#}_7/A^{b}_7$	3322.44	10.4
A_7	3520.00	9.8
$A^{\#}_7/B^{b}_7$	3729.31	9.3
B_7	3951.07	8.7
C_8	4186.01	8.2
$C^{\#}_8/D^{b}_8$	4434.92	7.8
D_8	4698.64	7.3
$D^{\#}_8/E^{b}_8$	4978.03	6.9

ANEXO C

A. Modificación de un Servo-Motor para rotación continua

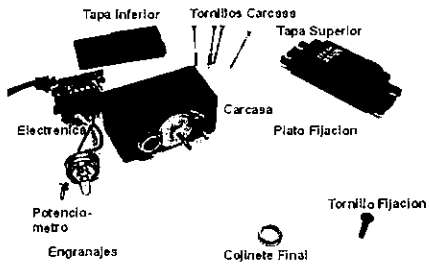


figura C-1 Servo-motor Desarmado Completamente.

1) Desmontar el servo, aflojando los cuatro tornillos que fijan la carcasa y desmontar los diferentes elementos, tal y como aparece en la figura C-1.

Figura C-2. Detalle del tope que hay que cortar.

2) Localizar el engranaje principal, el que tiene el eje estriado, y proceder a cortar el tope limitador (Figura C-2) con la ayuda de un cúter o unos alicates de corte. Ver figura C-3.

Figura C-3. Posible métodos de corte.

3) Montar todo el engranaje en la carcasa superior con la ayuda de la figura C-4. Hay que procurar no eliminar la grasa lubricante. Evitar que queden restos de nylon entre los dientes de los engranajes.

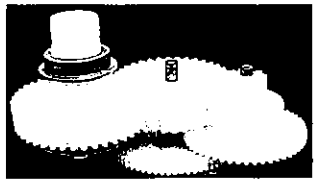


Figura C-4. Montaje de los engranajes.

4) Ahora hay que reemplazar el potenciómetro de control por otro de igual valor (Figura C-5) o o bien por dos resistencias de 2K7. El potenciómetro tiene la ventaja de permitir ajustar el punto neutral del servo al valor deseado. Hay que tener en cuanta el color de los cables especialmente el conectado a la patilla central.

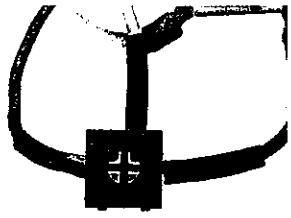
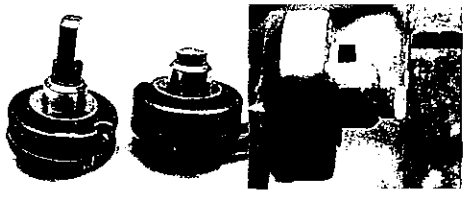


Figura C-5. Conexión de un nuevo potenciómetro.

5) Otra posibilidad es la de utilizar el mismo potenciómetro (Figura C-6), pero cortando su eje con una sierra para que no sea alterado por el engranaje al girar. Esta solución tiene la ventaja de no tener que soldar nada. El eje puede cortarse fácilmente con la ayuda de un tornillo de fijación, tal y como muestra la figura 7.



Figuras C-6 y C-7. Detalle de modificación del potenciómetro original.

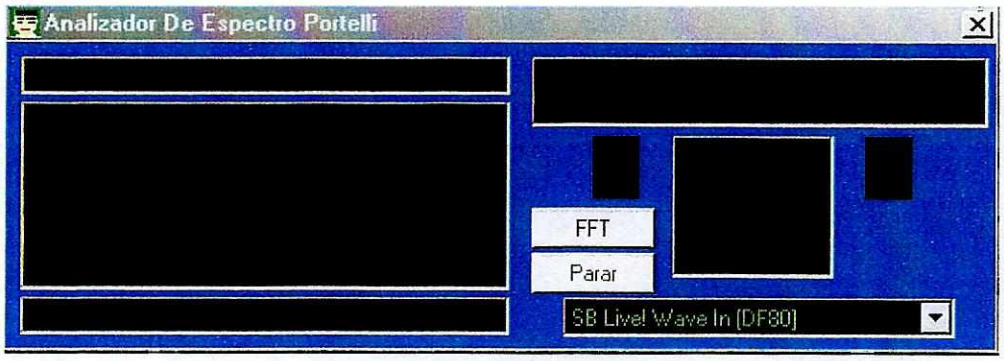
6) Volver a montarlo todo. Hay que tener en cuenta que la posición en la que haya quedado el potenciómetro es la que marcará el valor neutral, por lo que antes de montar el servo es conveniente conectarlo al circuito de control de servos y enviar los pulsos correspondientes a la posición de reposo. A continuación se girará el eje del potenciómetro ya cortado, hasta que este se pare. Luego se monta el potenciómetro en su sitio con cuidado de no mover el eje.

ANEXO D

Control digital implementado en Visual Basic 6.0

Forma 1: Analizador de espectro

Este módulo ejecuta la FFT a los datos, presentando el espectro en forma gráfica, la frecuencia principal de oscilación y la nota musical a la que esta corresponde.



Código fuente:

```
Option Explicit


Const NumSamples = 65536 '32768
Const SPS = 16000
Private DevHandle As Long

Private Visualizing As Boolean
Private Divisor As Long

Private ScopeHeight As Long

Private Type WaveFormatEx
   FormatTag As Integer
   Channels As Integer
   SamplesPerSec As Long
   AvgBytesPerSec As Long
   BlockAlign As Integer
   BitsPerSample As Integer
   ExtraDataSize As Integer
End Type

Private Type WaveHdr
   lpData As Long
   dwBufferLength As Long
   dwBytesRecorded As Long
   dwUser As Long
   dwFlags As Long
   dwLoops As Long
   lpNext As Long
   Reserved As Long
End Type

Private Type WaveInCaps
   ManufacturerID As Integer
   ProductID As Integer
   DriverVersion As Long
   ProductName(1 To 32) As Byte
   Formats As Long
   Channels As Integer
   Reserved As Integer
End Type
```

```
Private Const WAVE_INVALIDFORMAT = &H0&          '/* invalid format */
Private Const WAVE_FORMAT_1M08 = &H1&            '/* 11.025 kHz, Mono,   8-bit
Private Const WAVE_FORMAT_1S08 = &H2&            '/* 11.025 kHz, Stereo, 8-bit
Private Const WAVE_FORMAT_1M16 = &H4&            '/* 11.025 kHz, Mono,   16-bit
Private Const WAVE_FORMAT_1S16 = &H8&            '/* 11.025 kHz, Stereo, 16-bit
Private Const WAVE_FORMAT_2M08 = &H10&           '/* 22.05  kHz, Mono,   8-bit
Private Const WAVE_FORMAT_2S08 = &H20&           '/* 22.05  kHz, Stereo, 8-bit
Private Const WAVE_FORMAT_2M16 = &H40&           '/* 22.05  kHz, Mono,   16-bit
Private Const WAVE_FORMAT_2S16 = &H80&           '/* 22.05  kHz, Stereo, 16-bit
Private Const WAVE_FORMAT_4M08 = &H100&          '/* 44.1   kHz, Mono,   8-bit
Private Const WAVE_FORMAT_4S08 = &H200&          '/* 44.1   kHz, Stereo, 8-bit
Private Const WAVE_FORMAT_4M16 = &H400&          '/* 44.1   kHz, Mono,   16-bit
Private Const WAVE_FORMAT_4S16 = &H800&          '/* 44.1   kHz, Stereo, 16-bit
Private Const WAVE_FORMAT_BOLA = &HFFF&          '/* bola


Private Const WAVE_FORMAT_PCM = 1

Private Const WHDR_DONE = &H1&
Private Const WHDR_PREPARED = &H2&
Private Const WHDR_BEGINLOOP = &H4&
Private Const WHDR_ENDLOOP = &H8&
Private Const WHDR_INQUEUE = &H10&

Private Const WIM_OPEN = &H3BE
Private Const WIM_CLOSE = &H3BF
Private Const WIM_DATA = &H3C0

Private Declare Function waveInAddBuffer Lib "winmm" (ByVal InputDeviceHandle As Long, ByVal WaveHdrPointer As
Long, ByVal WaveHdrStructSize As Long) As Long
Private Declare Function waveInPrepareHeader Lib "winmm" (ByVal InputDeviceHandle As Long, ByVal WaveHdrPointer As
Long, ByVal WaveHdrStructSize As Long) As Long
Private Declare Function waveInUnprepareHeader Lib "winmm" (ByVal InputDeviceHandle As Long, ByVal WaveHdrPointer
As Long, ByVal WaveHdrStructSize As Long) As Long

Private Declare Function waveInGetNumDevs Lib "winmm" () As Long
Private Declare Function waveInGetDevCaps Lib "winmm" Alias "waveInGetDevCapsA" (ByVal uDeviceID As Long, ByVal
WaveInCapsPointer As Long, ByVal WaveInCapsStructSize As Long) As Long

Private Declare Function waveInOpen Lib "winmm" (WaveDeviceInputHandle As Long, ByVal WhichDevice As Long, ByVal
WaveFormatExPointer As Long, ByVal CallBack As Long, ByVal CallBackInstance As Long, ByVal Flags As Long) As Long
Private Declare Function waveInClose Lib "winmm" (ByVal WaveDeviceInputHandle As Long) As Long

Private Declare Function waveInStart Lib "winmm" (ByVal WaveDeviceInputHandle As Long) As Long
Private Declare Function waveInReset Lib "winmm" (ByVal WaveDeviceInputHandle As Long) As Long
Private Declare Function waveInStop Lib "winmm" (ByVal WaveDeviceInputHandle As Long) As Long



Dim Active_Excel As Object
Dim Active_Worksheet As Object
Dim Active_Workbook As Object


Sub InitDevices()

Set Active_Excel = CreateObject("Excel.Application")
Set Active_Workbook = Active_Excel.Workbooks.Add
Set Active_Worksheet = Active_Workbook.Worksheets.Add

  Dim Caps As WaveInCaps, Which As Long
  DevicesBox.Clear
  For Which = 0 To waveInGetNumDevs - 1
    Call waveInGetDevCaps(Which, VarPtr(Caps), Len(Caps))
    If Caps.Formats And WAVE_FORMAT_4S16 Then '16-bit mono devices
      Call DevicesBox.AddItem(StrConv(Caps.ProductName, vbUnicode), Which)
    End If

    Next
```

```
    If DevicesBox.ListCount = 0 Then
      MsgBox "You have no audio input devices!", vbCritical, "Ack!"
      End
    End If
    DevicesBox.ListIndex = 0
  End Sub

  Private Sub Form_Load()
    Call InitDevices

     ScopeBuff.Width = Scope.ScaleWidth
    ScopeBuff.Height = Scope.ScaleHeight
    ScopeBuff.BackColor = Scope.BackColor

    ScopeHeight = Scope.Height
  End Sub

  Private Sub Form_Unload(Cancel As Integer)
    If DevHandle <> 0 Then
      Call DoStop
      Cancel = 1
      If Visualizing = True Then
        quittimer.Enabled = True
      End If
    End If

  Set Active_Excel = Nothing
  Set Active_Workbook = Nothing
  Set Active_Worksheet = Nothing
  End Sub

  Private Sub QuitTimer_Timer()
  Unload Me
  End Sub

  Public Sub StartButton_Click()

    Static WaveFormat As WaveFormatEx
    With WaveFormat
      .FormatTag = WAVE_FORMAT_PCM
      .Channels = 1
      .SamplesPerSec = SPS '16000
      .BitsPerSample = 16
      .BlockAlign = (.Channels * .BitsPerSample) \ 8
      .AvgBytesPerSec = .BlockAlign * .SamplesPerSec
      .ExtraDataSize = 0
    End With

    Debug.Print "waveInOpen:"; waveInOpen(DevHandle, DevicesBox.ListIndex, VarPtr(WaveFormat), 0, 0, 0)
    If DevHandle = 0 Then
      Call MsgBox("Wave input device didn't open!", vbExclamation, " ")
      Exit Sub
    End If
    Debug.Print " "; DevHandle
    Call waveInStart(DevHandle)

    StopButton.Enabled = True
    StartButton.Enabled = False
    DevicesBox.Enabled = False


    If Afinando = False Then

    Call Proceso

    End If

  End Sub


  Public Sub StopButton_Click()
    Call DoStop
  End Sub
```

```
Public Sub DoStop()
  Call waveInReset(DevHandle)
  Call waveInClose(DevHandle)
  DevHandle = 0
  StopButton.Enabled = False
  StartButton.Enabled = True
  DevicesBox.Enabled = True
End Sub


Private Sub MandarAExcel(Ind() As Double, Outd() As Double, outI() As Double, amp() As Double, freq() As Double)
Active_Excel.Visible = True
    Dim ping As Long
    'For ping = 1 To NumSamples / 2
    'For ping = 130 To 260


    'Active_Worksheet.Cells(ping, 1).Value = Ind(ping)
    'Active_Worksheet.Cells(ping, 2).Value = Outd(ping)
    'Active_Worksheet.Cells(ping, 3).Value = outt(ping)
    'Active_Worksheet.Cells(ping - 129, 5).Value = amp(ping)
    'Active_Worksheet.Cells(ping - 129, 4).Value = freq(ping)

    'DoEvents

    'Next


    'For ping = 261 To 522


    'Active_Worksheet.Cells(ping, 1).Value = Ind(ping)
    'Active_Worksheet.Cells(ping, 2).Value = Outd(ping)
    'Active_Worksheet.Cells(ping, 3).Value = outI(ping)
    'Active_Worksheet.Cells(ping - 260, 7).Value = amp(ping)
    'Active_Worksheet.Cells(ping - 260, 6).Value = freq(ping)

    'DoEvents

    'Next



    For ping = 1 To NumSamples / 4


     Active_Worksheet.Cells(ping, 2).Value = amp(ping)
     Active_Worksheet.Cells(ping, 1).Value = freq(ping)

    DoEvents

    Next

End Sub

Public Sub Proceso()

  Visualizing = True

  Static X As Long
  Static XX As Long
  Static Wave As WaveHdr

  Static InDatas(0 To (NumSamples / 2) + 1) As Integer

  Static InData(0 To (NumSamples / 2) + 1) As Double
  Static OutData(0 To (NumSamples / 2) + 1) As Double
  Static InDatai(0 To (NumSamples / 2) + 1) As Double
  Static OutDatai(0 To (NumSamples / 2) + 1) As Double
  Static AmpData(0 To (NumSamples / 2) + 1) As Double
  Static IndiceFrecuencia(0 To (NumSamples / 2) + 1) As Double
  Static nota(0 To (NumSamples / 2) + 1) As Double

  Dim III As Double
```

```
Dim maximo As Double
Dim maximoindice As Double
Dim maximoTotalindice As Double
Dim PI As Double
PI = 3.1415926

Dim Armonica1 As Double
Dim Armonica2 As Double
Dim Armonica3 As Double
Dim Armonica4 As Double


Armonica1 = 2
Armonica2 = 3
Armonica3 = 3
Armonica4 = 3


Dim maximoar1 As Double
Dim maximoar2 As Double
Dim maximoar3 As Double
Dim maximoar4 As Double

Dim MaximoTotalArIndice1 As Double
Dim MaximoTotalArIndice2 As Double
Dim MaximoTotalArIndice3 As Double
Dim MaximoTotalArIndice4 As Double



    For III = 1 To (NumSamples / 2)

    IndiceFrecuencia(III) = III / 2.048

    Next



  Do
    Wave.lpData = VarPtr(InDatas(0))
    Wave.dwBufferLength = NumSamples
    Wave.dwFlags = 0
    Call waveInPrepareHeader(DevHandle, VarPtr(Wave), Len(Wave))

    Call waveInAddBuffer(DevHandle, VarPtr(Wave), Len(Wave))
    Do
     DoEvents
    Loop Until ((Wave.dwFlags And WHDR_DONE) = WHDR_DONE) Or DevHandle = 0

    If DevHandle = 0 Then Exit Do
    Call waveInUnprepareHeader(DevHandle, VarPtr(Wave), Len(Wave))

    If Afinando = True Then
    GENERAL.Timer3 = False
    GENERAL.ListoAdquirir.ZOrder
    GENERAL.Timer4 = True
    End If

    For III = 1 To (NumSamples / 2)

    InData(III) = CDbl(InDatas(III))

    Next

    For III = 1 To (NumSamples / 2)

    InData(III) = 0.54 - 0.46 * Cos(CDbl(((2 * PI * InData(III)) / (NumSamples / 2))))

    Next


    Call FFTDouble((NumSamples / 2), False, InData(1), InDatai(1), OutData(1), OutDatai(1))

    For III = 1 To (NumSamples / 2) + 1
```

```
        AmpData(III) = Abs(OutData(III) * OutData(III) + OutDatai(III) * OutDatai(III))

    Next
     maximo = 0

    For III = 1 To (NumSamples / 2) + 1

        If AmpData(III) > maximo Then
           If ((III > 32) And (III < 500)) Then
         maximo = AmpData(III)
         maximoTotalindice = III

       End If
       End If

    Next


      maximoar1 = 0

    For III = 2 * CDbl(Armonica1 * Afinacion.LabelFrecuencial - (Afinacion.LabelFrecuencial / 2)) To 2 *
CDbl(Armonica1 * Afinacion.LabelFrecuencial + (Afinacion.LabelFrecuencial / 2))

        If AmpData(III) > maximoar1 Then
          maximoar1 = AmpData(III)
          MaximoTotalArIndice1 = III

       End If


    Next

      maximoar2 = 0

    For III = 2 * CDbl(Armonica2 * Afinacion.LabelFrecuencialI - (Afinacion.LabelFrecuencialI / 2)) To 2 *
CDbl(Armonica2 * Afinacion.LabelFrecuencialI + (Afinacion.LabelFrecuencialI / 2))

        If AmpData(III) > maximoar2 Then
          maximoar2 = AmpData(III)
          MaximoTotalArIndice2 = III

       End If


    Next


      maximoar3 = 0

    For III = 2 * CDbl(Armonica3 * Afinacion.LabelFrecuencialII - (Afinacion.LabelFrecuencialII / 2)) To 2 *
CDbl(Armonica3 * Afinacion.LabelFrecuencialII + (Afinacion.LabelFrecuencialII / 2))

        If AmpData(III) > maximoar3 Then
          maximoar3 = AmpData(III)
          MaximoTotalArIndice3 = III

       End If


    Next


      maximoar4 = 0

    For III = 2 * CDbl(Armonica4 * Afinacion.LabelFrecuencialIII - (Afinacion.LabelFrecuencialIII / 2)) To 2 *
CDbl(Armonica4 * Afinacion.LabelFrecuencialIII + (Afinacion.LabelFrecuencialIII / 2))

        If AmpData(III) > maximoar4 Then
          maximoar4 = AmpData(III)
          MaximoTotalArIndice4 = III

       End If
```

```
Next


Label5.Caption = "" & IndiceFrecuencia(maximoTotalindice)
Armo1.Caption = "" & IndiceFrecuencia(MaximoTotalArIndice1)
Armo2.Caption = "" & IndiceFrecuencia(MaximoTotalArIndice2)
Armo3.Caption = "" & IndiceFrecuencia(MaximoTotalArIndice3)
Armo4.Caption = "" & IndiceFrecuencia(MaximoTotalArIndice4)


If (IndiceFrecuencia(maximoindice) > 62.95907221) Then

   If (IndiceFrecuencia(maximoindice) < 66.70281346) Then

        Label1.Caption = "C"
        Label2.Caption = ""

        If IndiceFrecuencia(maximoindice) - 65.40639133 < 0 Then
          Label3.Caption = "" & Chr$(60)
          Label4.Caption = "" & Chr$(60)
        Else
          Label3.Caption = "" & Chr$(62)
          Label4.Caption = "" & Chr$(62)

        End If

   End If
   End If

If (IndiceFrecuencia(maximoindice) > 66.70281346) Then

   If (IndiceFrecuencia(maximoindice) < 70.66916916) Then

        Label1.Caption = "C"
        Label2.Caption = "#"

        If IndiceFrecuencia(maximoindice) - 69.29565774 < 0 Then
          Label3.Caption = "" & Chr$(60)
          Label4.Caption = "" & Chr$(60)

        Else
          Label3.Caption = "" & Chr$(62)
          Label4.Caption = "" & Chr$(62)

        End If


   End If
   End If

If (IndiceFrecuencia(maximoindice) > 70.66916916) Then

   If (IndiceFrecuencia(maximoindice) < 74.87137663) Then

        Label1.Caption = "D"
        Label2.Caption = ""

        If IndiceFrecuencia(maximoindice) - 73.41619198 < 0 Then
          Label3.Caption = "" & Chr$(60)
          Label4.Caption = "" & Chr$(60)

        Else
          Label3.Caption = "" & Chr$(62)
          Label4.Caption = "" & Chr$(62)

        End If

   End If
   End If


If (IndiceFrecuencia(maximoindice) > 74.87137663) Then
```

```
If (IndiceFrecuencia(maximoindice) < 79.32346036) Then

    Label1.Caption = "D"
    Label2.Caption = "#"

    If IndiceFrecuencia(maximoindice) - 77.7860593 < 0 Then
      Label3.Caption = "" & Chr$(60)
      Label4.Caption = "" & Chr$(60)

    Else
      Label3.Caption = "" & Chr$(62)
      Label4.Caption = "" & Chr$(62)

    End If

  End If
  End If

If (IndiceFrecuencia(maximoindice) > 79.32346036) Then

  If (IndiceFrecuencia(maximoindice) < 84.04027877) Then

    Label1.Caption = "E"
    Label2.Caption = ""

    If IndiceFrecuencia(maximoindice) - 82.40688923 < 0 Then
      Label3.Caption = "" & Chr$(60)
      Label4.Caption = "" & Chr$(60)

    Else
      Label3.Caption = "" & Chr$(62)
      Label4.Caption = "" & Chr$(62)

    End If

  End If
  End If


If (IndiceFrccuencia(maximoindice) > 84.04027877) Then

  If (IndiceFrecuencia(maximoindice) < 89.0375738) Then

    Label1.Caption = "F"
    Label2.Caption = ""


    If IndiceFrecuencia(maximoindice) - 87.30705786 < 0 Then
      Label3.Caption = "" & Chr$(60)
      Label4.Caption = "" & Chr$(60)

    Else
      Label3.Caption = "" & Chr$(62)
      Label4.Caption = "" & Chr$(62)

    End If

  End If
  End If


If (IndiceFrecuencia(maximoindice) > 89.0375738) Then

  If (IndiceFrecuencia(maximoindice) < 94.33202345) Then

    Label1.Caption = "F"
    Label2.Caption = "#"

    If IndiceFrecuencia(maximoindice) - 92.49860568 < 0 Then
      Label3.Caption = "" & Chr$(60)
      Label4.Caption = "" & Chr$(60)

    Else
      Label3.Caption = "" & Chr$(62)
```

```
        Label4.Caption = "" & Chr$(62)

      End If

  End If
  End If


If (IndiceFrecuencia(maximoindice) > 94.33202345) Then

  If (IndiceFrecuencia(maximoindice) < 99.94129746) Then

      Label1.Caption = "G"
      Label2.Caption = ""

      If IndiceFrecuencia(maximoindice) - 97.998859 < 0 Then
        Label3.Caption = "" & Chr$(60)
        Label4.Caption = "" & Chr$(60)

      Else
        Label3.Caption = "" & Chr$(62)
        Label4.Caption = "" & Chr$(62)

      End If

  End If
  End If


If (IndiceFrecuencia(maximoindice) > 99.94129746) Then

  If (IndiceFrecuencia(maximoindice) < 105.8841163) Then

      Label1.Caption = "G"
      Label2.Caption = "#"

      If IndiceFrecuencia(maximoindice) - 103.8261744 < 0 Then
        Label3.Caption = "" & Chr$(60)
        Label4.Caption = "" & Chr$(60)

      Else
        Label3.Caption = "" & Chr$(62)
        Label4.Caption = "" & Chr$(62)

      End If

  End If
  End If


If (IndiceFrecuencia(maximoindice) > 105.8841163) Then

  If (IndiceFrecuencia(maximoindice) < 112.1803135) Then

      Label1.Caption = "A"
      Label2.Caption = ""

      If IndiceFrecuencia(maximoindice) - 110 < 0 Then
        Label3.Caption = "" & Chr$(60)
        Label4.Caption = "" & Chr$(60)

      Else
        Label3.Caption = "" & Chr$(62)
        Label4.Caption = "" & Chr$(62)

      End If

  End If
  End If


If (IndiceFrecuencia(maximoindice) > 112.1803135) Then

  If (IndiceFrecuencia(maximoindice) < 118.850902) Then
```

```
            Label1.Caption = "A"
            Label2.Caption = "#"

            If IndiceFrecuencia(maximoindice) - 116.5409404 < 0 Then
               Label3.Caption = "" & Chr$(60)
               Label4.Caption = "" & Chr$(60)

            Else
               Label3.Caption = "" & Chr$(62)
               Label4.Caption = "" & Chr$(62)

            End If

         End If
         End If


      If (IndiceFrecuencia(maximoindice) > 118.850902) Then

         If (IndiceFrecuencia(maximoindice) < 125.9181444) Then

            Label1.Caption = "B"
            Label2.Caption = ""

            If IndiceFrecuencia(maximoindice) - 123.4708253 < 0 Then
               Label3.Caption = "" & Chr$(60)
               Label4.Caption = "" & Chr$(60)

            Else
               Label3.Caption = "" & Chr$(62)
               Label4.Caption = "" & Chr$(62)
            End If

         End If
         End If

      ScopeBuff.Cls
      SeopeBuff.CurrentX = 0
      ScopeBuff.CurrentY = ScopeHeight
      BARRAU.Cls
      BARRAU.CurrentX = 0
      BARRAU.CurrentY = ScopeHeight

      BARRAL.Cls
      BARRAL.CurrentX = 0
      BARRAL.CurrentY = ScopeHeight


      For X = 0 To 260
      ScopeBuff.CurrentY = SeopeHeight
      ScopeBuff.CurrentX = X
      BARRAL.CurrentY = ScopeHeight
      BARRAL.CurrentX = X
      BARRAU.CurrentY = ScopeHeight
      BARRAU.CurrentX = X

      XX = Int(X / 2) + 130


      ScopeBuff.Line Step(0, 0)-(X, ScopeHeight - (AmpData(XX) * 90)), &HC0FFFF
      BARRAL.Line Step(0, 0)-(X, ScopeHeight - (nota(XX) * 200)), &HFF&
      BARRAU.Line Step(0, 0)-(X, ScopeHeight - (nota(XX) * 200)), &HFF&


      Next

      Scope.Picture = ScopeBuff.Image
      DoEvents
      If Afinando = True Then
      Exit Do
      End If
   Loop While DevHandle <> 0
```

```
    Visualizing = False

End Sub
```

Modulo 1: Declaracion de la librería FFT.dll

```
Option Explicit

Public frecuenciaBuscadaI As Double
Public frecuenciaBuscadaII As Double
Public frecuenciaBuscadaIII As Double
Public frecuenciaBuscadaIIII As Double

Public Afinando As Boolean


Declare Sub FFTDouble Lib "FFT.dll" Alias "fft_double" _
(ByVal NumSamples As Long, ByVal InverseTransform As Boolean, _
RealIn As Double, _
ImagIn As Double, _
RealOut As Double, _
ImagOut As Double)

Declare Sub FFTSingle Lib "FFT.dll" Alias "fft_float" _
(ByVal NumSamples As Long, ByVal InverseTransform As Boolean, _
RealIn As Single, _
ImagIn As Single, _
RealOut As Single, _
ImagOut As Single)

Declare Function IndexToFrequency Lib "FFT.dll" _
Alias "Index_to_frequency" _
(ByVal NumSamples As Long, _
ByVal Index As Long) As Double
```

Forma 2: Controlador de servo-motores por medio del puerto serial.

Este módulo se encarga de la selección y accionamiento de los servo-motores mediante el protocolo utilizado por la tarjeta servo-controladora. También realiza un monitoreo al puerto serial y maneja un temporizador mediante la lectura periódica del reloj del sistema.

Código fuente:

```
Option Explicit

Dim T As New cTimer
Public funcionando As Boolean
Public TIMERStop As Boolean
Dim ValorRecividoPic As String
Dim ValorRecividoPic1 As String
Dim CONTA As Integer
Dim cccc As Integer


Private Sub apagar_Click()

Call Servo_TurnOffServo(servoencuestion)
funcionando = False


End Sub

Private Sub Derecha_full_Click()

Dim tiempo As Variant

funcionando = True

Call Servo_TurnOnServoAndSetPosition(servoencuestion, 255)

'Call Servo_TurnOffServo(ServoEnCuestion)
'Call Servo_TurnOffServo(ServoEnCuestion)

funcionando = False

End Sub

Private Sub Form_Terminate()

If Forma1.PuertoSerialCOM.PortOpen = True Then

Forma1.PuertoSerialCOM.PortOpen = False

End If

End Sub


Private Sub Form_Unload(Cancel As Integer)

If Forma1.PuertoSerialCOM.PortOpen = True Then

Forma1.PuertoSerialCOM.PortOpen = False

End If

End Sub

Private Sub Izquierda_full_Click()

Dim tiempo As Variant

funcionando = True

Call Servo_TurnOnServoAndSetPosition(servoencuestion, 1)

funcionando = False

End Sub

Private Sub Form_Load()

funcionando = False
```

```
servoSelect_Click (0)

InicializarPuertoSerial

InicializarServoValores

End Sub

Private Sub PuertoSerialCOM_OnComm()

CONTA = CONTA + 1
If Forma1.PuertoSerialCOM.CommEvent = comEvReceive Then

ValorRecividoPic1 = Forma1.PuertoSerialCOM.Input
ValorRecividoPic = Asc(ValorRecividoPic1)
cccc = ValorRecividoPic
Forma1.PuertoSerialCOM.Output = "" & Chr$(cccc)
Debug.Print Chr$(ValorRecividoPic)

'If CONTA = 6 Then
'cccc = ValorRecividoPic
'Debug.Print Chr$(cccc)

'End If

End If

End Sub

Private Sub servoSelect_Click(Index As Integer)

Select Case Index

    Case 0
      servoencuestion = "1"
      servoSelect(0) = 1
    Case 1
      servoencuestion = "2"
      servoSelect(1) = 1
    Case 2
      servoencuestion = "3"
      servoSelect(2) = 1
    Case 3
      servoencuestion = "4"
      servoSelect(3) = 1
    Case 4
      servoencuestion = "5"
      servoSelect(4) = 1
    Case 5
      servoencuestion = "6"
      servoSelect(5) = 1
    Case 6
      servoencuestion = "7"
      servoSelect(6) = 1
    Case 7
      servoencuestion = "8"
      servoSelect(7) = 1


End Select

End Sub

Private Sub Timer1_Timer()
Izquierda_full_Click
End Sub

Public Sub TimerPuertoSerial_Timer()

  If ContadorParaTimeOutPuertoSerial > 0 Then
     ContadorParaTimeOutPuertoSerial = ContadorParaTimeOutPuertoSerial - 1
  Else:
```

```
    Forma1.TimerPuertoSerial.Enabled = False
    TimedOut_PuertoSerial = True
  End If

End Sub

Public Sub Unload()

  If PuertoSerialCOM.PortOpen Then PuertoSerialCOM.PortOpen = False
   Set T = Nothing

End Sub

Public Sub Terminate()

  If PuertoSerialCOM.PortOpen Then PuertoSerialCOM.PortOpen = False
   Set T = Nothing

End Sub

'//////////////////////////////////////////////////////////////////////////////////////////////


Public Sub Motor_Pulso_Derecha(motor As Integer)

Dim tiempo As Variant

Dim Comandoderecha As Long
Dim ComandoParar As Long


If motor = 1 Then

   servoencuestion = "5"
   Comandoderecha = DerechaMotor1
   ComandoParar = PararMotor1


End If
If motor = 2 Then

   servoencuestion = "6"
   Comandoderecha = DerechaMotor2
   ComandoParar = PararMotor2


End If
If motor = 3 Then

   servoencuestion = "7"
   Comandoderecha = DerechaMotor3
   ComandoParar = PararMotor3


End If
If motor = 4 Then

   servoencuestion = "8"
   Comandoderecha = DerechaMotor4
   ComandoParar = PararMotor4


End If



funcionando = True


Call Servo_TurnOnServoAndSetPosition(servoencuestion, ComandoParar)
Call Servo_TurnOnServoAndSetPosition(servoencuestion, ComandoParar)

TIMERStop = False
T.StartTimer
```

```
Do
  DoEvents
 Label9.Caption = Format(T.CheckTimer / 1000, "0.####")

  If T.CheckTimer >= 60 Then Exit Do
Loop

Call Servo_TurnOnServoAndSetPosition(servoencuestion, Comandoderecha)

TIMERStop = False
T.StartTimer

Do
  DoEvents

 Label9.Caption = Format(T.CheckTimer / 1000, "0.####")

  If T.CheckTimer >= 60 Then Exit Do
Loop

Call Servo_TurnOnServoAndSetPosition(servoencuestion, ComandoParar)
Call Servo_TurnOnServoAndSetPosition(servoencuestion, ComandoParar)

TIMERStop = False
T.StartTimer
Do
  DoEvents
 Label9.Caption = Format(T.CheckTimer / 1000, "0.####")

  If T.CheckTimer >= 100 Then Exit Do
Loop

funcionando = False


End Sub


Public Sub Motor_Pulso_Izquierda(motor As Integer)

Dim tiempo As Variant

Dim ComandoIzquierda As Long
Dim ComandoParar As Long


If motor = 1 Then

   servoencuestion = "5"
   ComandoIzquierda = IzquierdaMotor1
   ComandoParar = PararMotor1


End If
If motor = 2 Then

   servoencuestion = "6"
   ComandoIzquierda = IzquierdaMotor2
   ComandoParar = PararMotor2


End If
If motor = 3 Then

   servoencuestion = "7"
   ComandoIzquierda = IzquierdaMotor3
   ComandoParar = PararMotor3


End If
If motor = 4 Then

   servoencuestion = "8"
```

```
  ComandoIzquierda = IzquierdaMotor4
  ComandoParar = PararMotor4

End If

funcionando = True

Call Servo_TurnOnServoAndSetPosition(servoencuestion, ComandoParar)
Call Servo_TurnOnServoAndSetPosition(servoencuestion, ComandoParar)

TIMERStop = False
T.StartTimer
Do
 DoEvents
 Label9.Caption = Format(T.CheckTimer / 1000, "0.####")
  If T.CheckTimer >= 60 Then Exit Do
Loop

Call Servo_TurnOnServoAndSetPosition(servoencuestion, ComandoIzquierda)

TIMERStop = False
T.StartTimer

Do
  DoEvents

 Label9.Caption = Format(T.CheckTimer / 1000, "0.####")

  If T.CheckTimer >= 60 Then Exit Do
Loop

Call Servo_TurnOnServoAndSetPosition(servoencuestion, ComandoParar)
Call Servo_TurnOnServoAndSetPosition(servoencuestion, ComandoParar)

TIMERStop = False
T.StartTimer
Do
 DoEvents

 Label9.Caption = Format(T.CheckTimer / 1000, "0.####")

  If T.CheckTimer >= 100 Then Exit Do
Loop

funcionando = False

End Sub
```

Módulo 2: Manejo de servomotores y puerto serial

```
Option Explicit

Public PuertoNumero As Integer
Public ModuloServoControl As String
Public ComandoDelTorqueBoard As String
Public servoencuestion As String
Public ContadorParaTimeOutPuertoSerial As Integer
Public TimedOut_PuertoSerial As Boolean
Public BaudRate As Integer
Public startUp As Integer
Public PortBusy_PuertoSerial As Integer

Public PararMotor1 As Long
```

```
Public PararMotor2 As Long
Public PararMotor3 As Long
Public PararMotor4 As Long

Public DerechaMotor1 As Long
Public DerechaMotor2 As Long
Public DerechaMotor3 As Long
Public DerechaMotor4 As Long

Public IzquierdaMotor1 As Long
Public IzquierdaMotor2 As Long
Public IzquierdaMotor3 As Long
Public IzquierdaMotor4 As Long


Dim ping As Integer
Dim Data As Variant


Public Sub InicializarServoValores()

PararMotor1 = 125
PararMotor2 = 125
PararMotor3 = 128
PararMotor4 = 125

DerechaMotor1 = 143    'Derecha=Apretar
DerechaMotor2 = 140
DerechaMotor3 = 160
DerechaMotor4 = 129

IzquierdaMotor1 = 114  'Izquierda=Aflojar
IzquierdaMotor2 = 112
IzquierdaMotor3 = 108
IzquierdaMotor4 = 109

End Sub

Public Sub InicializarPuertoSerial()


  ping = 1
  Forma1.TimerPuertoSerial.Enabled = False

  startUp = 1


  PortBusy_PuertoSerial = 0
  BaudRate = 19200
  PuertoNumero = 1
  ModuloServoControl = "1"
  servoencuestion = "1"
  ComandoDelTorqueBoard = ">"


  startUp = 0

  OpenPort


End Sub


Public Sub OpenPort()



Dim Data As Byte
Dim InBuffer() As Byte
Dim OutBuffer As String
Dim SError As Integer
```

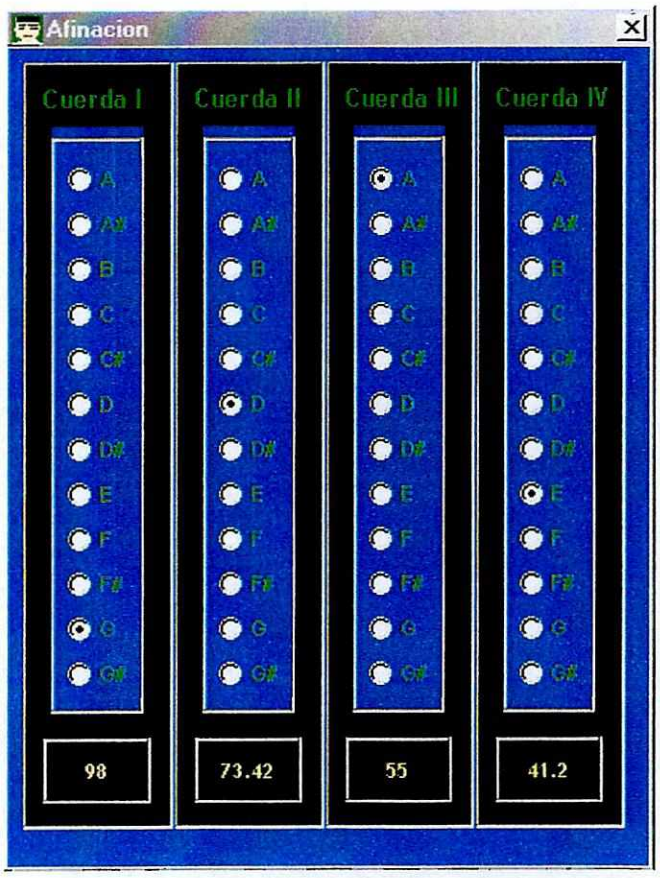
```
    If startUp Then Exit Sub

    Forma1.imgnotconnected.ZOrder

    If Forma1.PuertoSerialCOM.PortOpen Then

        Forma1.PuertoSerialCOM.PortOpen = False

    Else

        Forma1.PuertoSerialCOM.CommPort = PuertoNumero
        Forma1.PuertoSerialCOM.Handshaking = comNone              'Sin Handshake
        Forma1.PuertoSerialCOM.RThreshold = 1              'Cada Vez que recive 1, genera un evento
        Forma1.PuertoSerialCOM.SThreshold = 0
        Forma1.PuertoSerialCOM.InputLen = 1
        Forma1.PuertoSerialCOM.Settings = Str(BaudRate) + ",N,8,1"    'settings del Puerto serial
        Forma1.PuertoSerialCOM.InputMode = comInputModeText       'Recive Texto en una Variant
        Forma1.PuertoSerialCOM.NullDiscard = False
        Forma1.PuertoSerialCOM.DTREnable = False                  'sin data terminal ready
        Forma1.PuertoSerialCOM.RTSEnable = False                  'sin ready to send

        Forma1.PuertoSerialCOM.PortOpen = True

        Forma1.imgconnected.ZOrder


    End If


End Sub

Public Sub Servo_GetPosition(servoencuestion As String)

Dim OutBuffer As String

If Forma1.PuertoSerialCOM.PortOpen And PortBusy_PuertoSerial = 0 Then


    PortBusy_PuertoSerial = 1

        OutBuffer = ComandoDelTorqueBoard & ModuloServoControl & servoencuestion & "g"
        Forma1.PuertoSerialCOM.Output = OutBuffer

    PortBusy_PuertoSerial = 0
End If


End Sub
Public Sub Servo_GoToHomePosition(servoencuestion As String)

Dim OutBuffer As String

If Forma1.PuertoSerialCOM.PortOpen And PortBusy_PuertoSerial = 0 Then


    PortBusy_PuertoSerial = 1

        OutBuffer = ComandoDelTorqueBoard & ModuloServoControl & servoencuestion & "h"
        Forma1.PuertoSerialCOM.Output = OutBuffer

    PortBusy_PuertoSerial = 0

End If
```

```
End Sub

Public Sub Servo_SetCurrentPositionAsHome(servoencuestion As String)


Dim OutBuffer As String

If Forma1.PuertoSerialCOM.PortOpen And PortBusy_PuertoSerial = 0 Then


    PortBusy_PuertoSerial = 1

        OutBuffer = ComandoDelTorqueBoard & ModuloServoControl & servoencuestion & "c"
        Forma1.PuertoSerialCOM.Output = OutBuffer

    PortBusy_PuertoSerial = 0

End If


End Sub

Public Sub Servo_SetWidthResolution(servoencuestion As String, ServoResolucion As Long)


Dim OutBuffer As String

If Forma1.PuertoSerialCOM.PortOpen And PortBusy_PuertoSerial = 0 Then


    PortBusy_PuertoSerial = 1

        OutBuffer = ComandoDelTorqueBoard & ModuloServoControl & servoencuestion & "w" & Chr$(ServoResolucion)
        Forma1.PuertoSerialCOM.Output = OutBuffer

    PortBusy_PuertoSerial = 0

End If


End Sub

Public Sub Servo_SetBaudRate(servoencuestion As String, Baudios As Long)


Dim OutBuffer As String

If Forma1.PuertoSerialCOM.PortOpen And PortBusy_PuertoSerial = 0 Then


    PortBusy_PuertoSerial = 1

        OutBuffer = ComandoDelTorqueBoard & ModuloServoControl & servoencuestion & "b" & Chr$(Baudios)
        Forma1.PuertoSerialCOM.Output = OutBuffer

    PortBusy_PuertoSerial = 0

End If
```

```
End Sub

Public Sub Servo_TurnOffServo(servoencuestion As String)



Dim OutBuffer As String

Dim ApagarServo As Long
ApagarServo = 0

If Forma1.PuertoSerialCOM.PortOpen And PortBusy_PuertoSerial = 0 Then


    PortBusy_PuertoSerial = 1

        OutBuffer = ComandoDelTorqueBoard & ModuloServoControl & servoencuestion & "a" & Chr$(ApagarServo)
        Forma1.PuertoSerialCOM.Output = OutBuffer

    PortBusy_PuertoSerial = 0

End If



End Sub

Public Sub Servo_TurnOnServoAndSetPosition(servoencuestion As String, ServoPosicion As Long)



Dim OutBuffer As String

Dim ApagarServo As Byte
ApagarServo = 0

If Forma1.PuertoSerialCOM.PortOpen And PortBusy_PuertoSerial = 0 Then


    PortBusy_PuertoSerial = 1

        OutBuffer = ComandoDelTorqueBoard & ModuloServoControl & servoencuestion & "a" & Chr$(ServoPosicion)
        Forma1.PuertoSerialCOM.Output = OutBuffer

    PortBusy_PuertoSerial = 0

End If



End Sub
```

Forma 3: Afinación

Esta forma permite al usuario la colocación de la afinación deseada para cada cuerda, indicando la frecuencia principal de oscilación, en Hertz, que corresponde a cada una.



Código fuente:

```
Option Explicit


Private Sub Form_Load()

C_1(10) = 1    'G
C_2(5) = 1     'D
C_3(0) = 1     'A
C_4(7) = 1     'E

End Sub


Private Sub C_4_Click(Index As Integer)

Select Case Index

        Case 0
```

```
    frecuenciaBuscadaIIII = 55#    'A
    C_4(0) = 1
  Case 1
    frecuenciaBuscadaIIII = 58.27  'A#
    C_4(1) = 1
  Case 2
    frecuenciaBuscadaIIII = 30.87  'B
    C_4(2) = 1
  Case 3
    frecuenciaBuscadaIIII = 32.7   'C
    C_4(3) = 1
  Case 4
    frecuenciaBuscadaIIII = 34.65  'C#
    C_4(4) = 1
  Case 5
    frecuenciaBuscadaIIII = 36.71  'D
    C_4(5) = 1
  Case 6
    frecuenciaBuscadaIIII = 38.89  'D#
    C_4(6) = 1
  Case 7
    frecuenciaBuscadaIIII = 41.2   'E
    C_4(7) = 1
  Case 8
    frecuenciaBuscadaIIII = 43.65  'F
    C_4(8) = 1
  Case 9
    frecuenciaBuscadaIIII = 46.25  'F#
    C_4(9) = 1
  Case 10
    frecuenciaBuscadaIIII = 49#    'G
    C_4(10) = 1
  Case 11
    frecuenciaBuscadaIIII = 51.91  'G#
    C_4(11) = 1

End Select

LabelFrecuenciaIIII.Caption = "" & frecuenciaBuscadaIIII

End Sub


Private Sub C_3_Click(Index As Integer)

Select Case Index

  Case 0
    frecuenciaBuscadaIII = 55#    'A
    C_3(0) = 1
  Case 1
    frecuenciaBuscadaIII = 58.27  'A#
    C_3(1) = 1
  Case 2
    frecuenciaBuscadaIII = 61.74  'B
    C_3(2) = 1
  Case 3
    frecuenciaBuscadaIII = 65.41  'C
    C_3(3) = 1
  Case 4
    frecuenciaBuscadaIII = 69.3  'C#
    C_3(4) = 1
  Case 5
    frecuenciaBuscadaIII = 73.42  'D
    C_3(5) = 1
  Case 6
    frecuenciaBuscadaIII = 77.78  'D#
    C_3(6) = 1
  Case 7
    frecuenciaBuscadaIII = 41.2  'E
    C_3(7) = 1
```

```
        Case 8
           frecuenciaBuscadaIII = 43.65  'F
           C_3(8) = 1
        Case 9
           frecuenciaBuscadaIII = 46.25  'F#
           C_3(9) = 1
        Case 10
           frecuenciaBuscadaIII = 49#    'G
           C_3(10) = 1
        Case 11
           frecuenciaBuscadaIII = 51.91  'G#
           C_3(11) = 1

End Select

LabelFrecuenciaIII.Caption = "" & frecuenciaBuscadaIII

End Sub

Private Sub C_2_Click(Index As Integer)

Select Case Index

        Case 0
           frecuenciaBuscadaII = 55#     'A
           C_2(0) = 1
        Case 1
           frecuenciaBuscadaII = 58.27   'A#
           C_2(1) = 1
        Case 2
           frecuenciaBuscadaII = 61.74   'B
           C_2(2) = 1
        Case 3
           frecuenciaBuscadaII = 65.41   'C
           C_2(3) = 1
        Case 4
           frecuenciaBuscadaII = 69.3    'C#
           C_2(4) = 1
        Case 5
           frecuenciaBuscadaII = 73.42   'D
           C_2(5) = 1
        Case 6
           frecuenciaBuscadaII = 77.78   'D#
           C_2(6) = 1
        Case 7
           frecuenciaBuscadaII = 82.41   'E
           C_2(7) = 1
        Case 8
           frecuenciaBuscadaII = 87.31   'F
           C_2(8) = 1
        Case 9
           frecuenciaBuscadaII = 92.5    'F#
           C_2(9) = 1
        Case 10
           frecuenciaBuscadaII = 98#     'G
           C_2(10) = 1
        Case 11
           frecuenciaBuscadaII = 103.83  'G#
           C_2(11) = 1

End Select

LabelFrecuenciaII.Caption = "" & frecuenciaBuscadaII

End Sub

Private Sub C_1_Click(Index As Integer)
```

```
Select Case Index

    Case 0
      frecuenciaBuscadaI = 110#    'A
      C_1(0) = 1
    Case 1
      frecuenciaBuscadaI = 116.4   'A#
      C_1(1) = 1
    Case 2
      frecuenciaBuscadaI = 123.47  'B
      C_1(2) = 1
    Case 3
      frecuenciaBuscadaI = 130.81  'C
      C_1(3) = 1
    Case 4
      frecuenciaBuscadaI = 138.59  'C#
      C_1(4) = 1
    Case 5
      frecuencıaBuscadaI = 73.42   'D
      C_1(5) = 1
    Case 6
      frecuenciaBuscadaI = 77.78   'D#
      C_1(6) = 1
    Case 7
      frecuenciaBuscadaI = 82.41   'E
      C_1(7) = 1
    Case 8
      frecuenciaBuscadaI = 87.31   'F
      C_1(8) = 1
    Case 9
      frecuenciaBuscadaI = 92.5    'F#
      C_1(9) = 1
    Case 10
      frecuencıaBuscadaI = 98#     'G
      C_1(10) = 1
    Case 11
      frecuenciaBuscadaI = 103.83  'G#
      C_1(11) = 1

End Select

LabelFrecuencial.Caption = "" & frecuenciaBuscadaI

End Sub

Private Sub Label5_Click()

End Sub
```

Forma 4: Control general del proceso

Esta forma controla el proceso de afinación. Indica cual es la parte del proceso que se está llevando a cabo en el momento, además indica el error entre el mando estereotípico y la frecuencia principal actual.

Código fuente:

```
Option Explicit

Dim luna1 As Integer
Dim luna2 As Integer
Dim luna3 As Integer
Dim luna4 As Integer
Dim EstadoLuna1 As Integer

Dim ToleranciaCuerda1 As Double
Dim ToleranciaCuerda2 As Double
Dim ToleranciaCuerda3 As Double
Dim ToleranciaCuerda4 As Double

Dim DiferenciaDeFrecuencia As Double
Dim LlamadasAMotor As Integer
Dim ContadorMotor As Integer



Private Sub BotonAfinar_Click()

Afinando = True
AfinandoAhora.ZOrder
   ANV.ZOrder
```

```
Call Base.StartButton_Click
TiempoEntrePulsosMotor = True
Do
DoEvents
Loop Until (TiempoEntrePulsosMotor = False)
Call Servo_TurnOffServo("8")
TiempoEntrePulsosMotor = True
Do
DoEvents
Loop Until (TiempoEntrePulsosMotor = False)
Call Servo_TurnOffServo("7")
TiempoEntrePulsosMotor = True
Do
DoEvents
Loop Until (TiempoEntrePulsosMotor = False)
Call Servo_TurnOffServo("6")
TiempoEntrePulsosMotor = True
Do
DoEvents
Loop Until (TiempoEntrePulsosMotor = False)
Call Servo_TurnOffServo("5")
TiempoEntrePulsosMotor = True
Do
DoEvents
Loop Until (TiempoEntrePulsosMotor = False)

'*************************************************************************************************
'*************************************************************************************************
'*************************************************************************************************
'-------------------------------------Cuerda # 1-----------------------------------------
'*************************************************************************************************
'*************************************************************************************************
'*************************************************************************************************


Cuerda1:

AAR.ZOrder

LabelFrecuenciaMando.Caption = Afinacion.LabelFrecuencial.Caption
LabelCuerdaNumero.Caption = "1"
'-------------------------------------------
Timer2 = True
EstadoLuna1 = 0
servoencuestion = "2"
Call Servo_TurnOnServoAndSetPosition(servoencuestion, 255)
'-------------------------------------------
EmpezarFFT = True
Do
DoEvents
Loop Until (EmpezarFFT = False)
Timer3 = True
Call Base.Proceso
Timer4 = False
ListoProcesar.ZOrder

LabelFrecuenciaActual.Caption = Format(Base.Label5.Caption, "0.##")


If (Base.Label5.Caption > 139) And (Base.Label5.Caption < 278) Then
LabelFrecuenciaActual.Caption = Format(Base.Label5.Caption / 2, "0.##")
End If

If (Base.Label5.Caption > 278) And (Base.Label5.Caption > 524) Then
LabelFrecuenciaActual.Caption = Format(Base.Label5.Caption / 3, "0.##")
End If


LabelFrecuenciaError = Format(Abs(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption), "0.##")
'-------------------------------------------
```

```
DiferenciaDeFrecuencia = CDbl(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption)

If Abs(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption) > ToleranciaCuerda1 Then
  If Abs(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption) > 33 Then
    TimerError = True
    Do
    DoEvents
    Loop Until (TimerError = False)
    GoTo Cuerda1
  End If
  Timer5 = True
  If (LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption) > 0 Then

    LlamadasAMotor = CInt(8 * (DiferenciaDeFrecuencia - ToleranciaCuerda1))
    If LlamadasAMotor = 0 Then
      LlamadasAMotor = 1
    End If

      For ContadorMotor = 1 To LlamadasAMotor
      Call Forma1.Motor_Pulso_Derecha(1)
      TiempoEntrePulsosMotor = True

      Do
      DoEvents
      Loop Until (TiempoEntrePulsosMotor = False)
      Next
      Timer5 = False
      ListoModificar.ZOrder
      TimerModificarTension = True
      Do
      DoEvents
      Loop Until (TimerModificarTension = True)
  Else
  DiferenciaDeFrecuencia = -DiferenciaDeFrecuencia
    LlamadasAMotor = CInt(11.5 * (DiferenciaDeFrecuencia - ToleranciaCuerda1))
    If LlamadasAMotor = 0 Then
      LlamadasAMotor = 1
    End If

      For ContadorMotor = 1 To LlamadasAMotor
      Call Forma1.Motor_Pulso_Izquierda(1)
      TiempoEntrePulsosMotor = True

      Do
      DoEvents
      Loop Until (TiempoEntrePulsosMotor = False)
      Next
      Timer5 = False
      ListoModificar.ZOrder
      TimerModificarTension = True
      Do
      DoEvents
      Loop Until (TimerModificarTension = True)
  End If


Else
  TimerEntreCuerdas = True
  Do
  DoEvents
  Loop Until (TimerEntreCuerdas = False)
  Call Servo_TurnOffServo(servoencuestion)
  TimerEntreCuerdas = True
  Do
  DoEvents
  Loop Until (TimerEntreCuerdas = False)
  GoTo Cuerda1AR

End If

  GoTo Cuerda1:

'*************************************************************************************************
'*************************************************************************************************
```

```
Cuerda1AR:

AAV.ZOrder


LabelFrecuenciaMando.Caption = Afinacion.LabelFrecuencia1.Caption * 2
LabelCuerdaNumero.Caption = "1"
'-------------------------------------------
Timer2 = True
EstadoLuna1 = 0
servoencuestion = "2"
Call Servo_TurnOnServoAndSetPosition(servoencuestion, 255)
'-------------------------------------------
EmpezarFFT = True
Do
DoEvents
Loop Until (EmpezarFFT = False)
Timer3 = True
Call Base.Proceso
Timer4 = False
ListoProcesar.ZOrder

LabelFrecuenciaActual.Caption = Format(Base.Armo1.Caption, "0.##")

LabelFrecuenciaError = Format(Abs(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption), "0.##")
'-------------------------------------------

DiferenciaDeFrecuencia = CDbl(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption)

If Abs(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption) > ToleranciaCuerda1 Then
  If Abs(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption) > 33 Then
    TimerError = True
    Do
    DoEvents
    Loop Until (TimerError = False)
    GoTo Cuerda1AR
  End If
  Timer5 = True
  If (LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption) > 0 Then

    LlamadasAMotor = CInt(8 * (DiferenciaDeFrecuencia - ToleranciaCuerda1))
    If LlamadasAMotor = 0 Then
      LlamadasAMotor = 1
    End If

      For ContadorMotor = 1 To LlamadasAMotor
      Call Forma1.Motor_Pulso_Derecha(1)
      TiempoEntrePulsosMotor = True

      Do
      DoEvents
      Loop Until (TiempoEntrePulsosMotor = False)
      Next
      Timer5 = False
      ListoModificar.ZOrder
      TimerModificarTension = True
      Do
      DoEvents
      Loop Until (TimerModificarTension = True)
  Else
  DiferenciaDeFrecuencia = -DiferenciaDeFrecuencia
    LlamadasAMotor = CInt(11.5 * (DiferenciaDeFrecuencia - ToleranciaCuerda1))
    If LlamadasAMotor = 0 Then
      LlamadasAMotor = 1
    End If

      For ContadorMotor = 1 To LlamadasAMotor
      Call Forma1.Motor_Pulso_Izquierda(1)
      TiempoEntrePulsosMotor = True

      Do
      DoEvents
      Loop Until (TiempoEntrePulsosMotor = False)
      Next
```

```
        Timer5 = False
        ListoModificar.ZOrder
        TimerModificarTension = True
        Do
        DoEvents
        Loop Until (TimerModificarTension = True)
    End If


Else
    TimerEntreCuerdas = True
    Do
    DoEvents
    Loop Until (TimerEntreCuerdas = False)
    Call Servo_TurnOffServo(servoencuestion)
    TimerEntreCuerdas = True
    Do
    DoEvents
    Loop Until (TimerEntreCuerdas = False)
    GoTo Cuerda2

End If

    GoTo Cuerda1AR:

'*************************************************************************************************
'*************************************************************************************************
'*************************************************************************************************
'-------------------------------------Cuerda # 2-------------------------------------
'*************************************************************************************************
'*************************************************************************************************
'*************************************************************************************************


Cuerda2:

AAR.ZOrder

LabelFrecuenciaMando.Caption = Afinacion.LabelFrecuenciaII.Caption
LabelCuerdaNumero.Caption = "2"
'-----------------------------------------
Timer2 = True
EstadoLuna1 = 0
servoencuestion = "1"
Call Servo_TurnOnServoAndSetPosition(servoencuestion, 255)
'-----------------------------------------
EmpezarFFT = True
Do
DoEvents
Loop Until (EmpezarFFT = False)
Timer3 = True
Call Base.Proceso
Timer4 = False
ListoProcesar.ZOrder

If Base.Label5.Caption > 105 Then
LabelFrecuenciaActual.Caption = Format(Base.Label5.Caption / 2, "0.##")
Else
LabelFrecuenciaActual.Caption = Format(Base.Label5.Caption, "0.##")
End If
LabelFrecuenciaError = Format(Abs(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption), "0.##")
'-----------------------------------------

DiferenciaDeFrecuencia = CDbl(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption)

If Abs(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption) > ToleranciaCuerda2 Then
    If Abs(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption) > 24 Then
        TimerError = True
        Do
        DoEvents
        Loop Until (TimerError = False)
        GoTo Cuerda2
    End If
```

```
Timer5 = True
If (LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption) > 0 Then

    LlamadasAMotor = CInt(6.5 * (DiferenciaDeFrecuencia - ToleranciaCuerda1))
    If LlamadasAMotor = 0 Then
      LlamadasAMotor = 1
    End If

      For ContadorMotor = 1 To LlamadasAMotor
      Call Forma1.Motor_Pulso_Derecha(2)
      TiempoEntrePulsosMotor = True

      Do
      DoEvents
      Loop Until (TiempoEntrePulsosMotor = False)
      Next
      TimerModificarTension = True
      Do
      DoEvents
      Loop Until (TimerModificarTension = True)
      Timer5 = False
      ListoModificar.ZOrder
  Else
  DiferenciaDeFrecuencia = -DiferenciaDeFrecuencia
    LlamadasAMotor = CInt(7 * (DiferenciaDeFrecuencia - ToleranciaCuerda1))
    If LlamadasAMotor = 0 Then
      LlamadasAMotor = 1
    End If

      For ContadorMotor = 1 To LlamadasAMotor
      Call Forma1.Motor_Pulso_Izquierda(2)
      TiempoEntrePulsosMotor = True
      Do
      DoEvents
      Loop Until (TiempoEntrePulsosMotor = False)
      Next
      TimerModificarTension = True
      Do
      DoEvents
      Loop Until (TimerModificarTension = True)
      Timer5 = False
      ListoModificar.ZOrder
  End If


Else

  TimerEntreCuerdas = True
  Do
  DoEvents
  Loop Until (TimerEntreCuerdas = False)
  Call Servo_TurnOffServo(servoencuestion)
  TimerEntreCuerdas = True
  Do
  DoEvents
  Loop Until (TimerEntreCuerdas = False)

  GoTo Cuerda2AR

End If

  GoTo Cuerda2:

'*****************************************************************************************************
'*****************************************************************************************************
'*****************************************************************************************************


Cuerda2AR:

AAV.ZOrder

LabelFrecuenciaMando.Caption = Afinacion.LabelFrecuenciaII.Caption * 3
LabelCuerdaNumero.Caption = "2"
```

```
'-------------------------------------------------
Timer2 = True
EstadoLuna1 = 0
servoencuestion = "1"
Call Servo_TurnOnServoAndSetPosition(servoencuestion, 255)
'-------------------------------------------------
EmpezarFFT = True
Do
DoEvents
Loop Until (EmpezarFFT = False)
Timer3 = True
Call Base.Proceso
Timer4 = False
ListoProcesar.ZOrder

LabelFrecuenciaActual.Caption = Format(Base.Armo2.Caption, "0.##")

LabelFrecuenciaError = Format(Abs(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption), "0.##")
'-------------------------------------------------

DiferenciaDeFrecuencia = CDbl(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption)

If Abs(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption) > ToleranciaCuerda2 Then
  If Abs(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption) > 24 Then
    TimerError = True
    Do
    DoEvents
    Loop Until (TimerError = False)
    GoTo Cuerda2AR
  End If
  Timer5 = True
  If (LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption) > 0 Then

    LlamadasAMotor = CInt(6.5 * (DiferenciaDeFrecuencia - ToleranciaCuerda1))
    If LlamadasAMotor = 0 Then
      LlamadasAMotor = 1
    End If

      For ContadorMotor = 1 To LlamadasAMotor
      Call Forma1.Motor_Pulso_Derecha(2)
      TiempoEntrePulsosMotor = True

      Do
      DoEvents
      Loop Until (TiempoEntrePulsosMotor = False)
      Next
      TimerModificarTension = True
      Do
      DoEvents
      Loop Until (TimerModificarTension = True)
      Timer5 = False
      ListoModificar.ZOrder
  Else
  DiferenciaDeFrecuencia = -DiferenciaDeFrecuencia
    LlamadasAMotor = CInt(7 * (DiferenciaDeFrecuencia - ToleranciaCuerda1))
    If LlamadasAMotor = 0 Then
      LlamadasAMotor = 1
    End If

      For ContadorMotor = 1 To LlamadasAMotor
      Call Forma1.Motor_Pulso_Izquierda(2)
      TiempoEntrePulsosMotor = True
      Do
      DoEvents
      Loop Until (TiempoEntrePulsosMotor = False)
      Next
      TimerModificarTension = True
      Do
      DoEvents
      Loop Until (TimerModificarTension = True)
      Timer5 = False
      ListoModificar.ZOrder
  End If
```

```
Else

   TimerEntreCuerdas = True
   Do
   DoEvents
   Loop Until (TimerEntreCuerdas = False)
   Call Servo_TurnOffServo(servoencuestion)
   TimerEntreCuerdas = True
   Do
   DoEvents
   Loop Until (TimerEntreCuerdas = False)

   'GoTo Cuerda3
   GoTo MotoresAfinados
End If

   GoTo Cuerda2AR:


'*************************************************************************************************
'*************************************************************************************************
'*************************************************************************************************
'-----------------------------------------Cuerda # 3-----------------------------------------
'*************************************************************************************************
'*************************************************************************************************
'*************************************************************************************************


Cuerda3:

AAR.ZOrder

LabelFrecuenciaMando.Caption = Afinacion.LabelFrecuenciaIII.Caption
LabelCuerdaNumero.Caption = "3"
'-------------------------------------------
Timer2 = True
EstadoLuna1 = 0
servoencuestion = "4"
Call Servo_TurnOnServoAndSetPosition(servoencuestion, 255)
'-------------------------------------------
EmpezarFFT = True
Do
DoEvents
Loop Until (EmpezarFFT = False)
Timer3 = True
Call Base.Proceso
Timer4 = False
ListoProcesar.ZOrder
If Base.Label5.Caption > 79 Then
LabelFrecuenciaActual.Caption = Format(Base.Label5.Caption / 3, "0.##")
Else
LabelFrecuenciaActual.Caption = Format(Base.Label5.Caption, "0.##")
End If

LabelFrecuenciaError = Format(Abs(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption), "0.##")
'-------------------------------------------

DiferenciaDeFrecuencia = CDbl(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption)

If Abs(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption) > ToleranciaCuerda3 Then
   If Abs(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption) > 17 Then
      TimerError = True
      Do
      DoEvents
      Loop Until (TimerError = False)
      GoTo Cuerda3
   End If
   Timer5 = True
   If (LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption) > 0 Then

      LlamadasAMotor = CInt(5 * (DiferenciaDeFrecuencia - ToleranciaCuerda1))
      If LlamadasAMotor = 0 Then
```

```
      LlamadasAMotor = 1
    End If

      For ContadorMotor = 1 To LlamadasAMotor
      Call Forma1.Motor_Pulso_Derecha(3)
      TiempoEntrePulsosMotor = True

      Do
      DoEvents
      Loop Until (TiempoEntrePulsosMotor = False)
      Next
      TimerModificarTension = True
      Do
      DoEvents
      Loop Until (TimerModificarTension = True)
      Timer5 = False
      ListoModificar.ZOrder
  Else
  DiferenciaDeFrecuencia = -DiferenciaDeFrecuencia
    LlamadasAMotor = CInt(5 * (DiferenciaDeFrecuencia - ToleranciaCuerda1))
    If LlamadasAMotor = 0 Then
      LlamadasAMotor = 1
    End If

      For ContadorMotor = 1 To LlamadasAMotor
      Call Forma1.Motor_Pulso_Izquierda(3)
      TiempoEntrePulsosMotor = True

      Do
      DoEvents
      Loop Until (TiempoEntrePulsosMotor = False)
      Next
      TimerModificarTension = True
      Do
      DoEvents
      Loop Until (TimerModificarTension = True)
      Timer5 = False
      ListoModificar.ZOrder
  End If


Else
  TimerEntreCuerdas = True
  Do
  DoEvents
  Loop Until (TimerEntreCuerdas = False)
  Call Servo_TurnOffServo(servoencuestion)
  TimerEntreCuerdas = True
  Do
  DoEvents
  Loop Until (TimerEntreCuerdas = False)
  GoTo Cuerda3AR

End If

  GoTo Cuerda3

'*****************************************************************************************************
'*****************************************************************************************************
'*****************************************************************************************************

Cuerda3AR:

AAV.ZOrder

LabelFrecuenciaMando.Caption = Afinacion.LabelFrecuenciaIII.Caption * 3
LabelCuerdaNumero.Caption = "3"
'-----------------------------------------
Timer2 = True
EstadoLuna1 = 0
servoencuestion = "4"
Call Servo_TurnOnServoAndSetPosition(servoencuestion, 255)
'-----------------------------------------
EmpezarFFT = True
```

```
Do
DoEvents
Loop Until (EmpezarFFT = False)
Timer3 = True
Call Base.Proceso
Timer4 = False
ListoProcesar.ZOrder

LabelFrecuenciaActual.Caption = Format(Base.Armo3.Caption, "0.##")

LabelFrecuenciaError = Format(Abs(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption), "0.##")
'-----------------------------------------

DiferenciaDeFrecuencia = CDbl(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption)

If Abs(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption) > ToleranciaCuerda3 Then
  If Abs(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption) > 17 Then
    TimerError = True
    Do
    DoEvents
    Loop Until (TimerError = False)
    GoTo Cuerda3AR
  End If
  Timer5 = True
  If (LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption) > 0 Then

    LlamadasAMotor = CInt(5 * (DiferenciaDeFrecuencia - ToleranciaCuerda1))
    If LlamadasAMotor = 0 Then
      LlamadasAMotor = 1
    End If

      For ContadorMotor = 1 To LlamadasAMotor
      Call Forma1.Motor_Pulso_Derecha(3)
      TiempoEntrePulsosMotor = True

      Do
      DoEvents
      Loop Until (TiempoEntrePulsosMotor = False)
      Next
      TimerModificarTension = True
      Do
      DoEvents
      Loop Until (TimerModificarTension = True)
      Timer5 = False
      ListoModificar.ZOrder
  Else
  DiferenciaDeFrecuencia = -DiferenciaDeFrecuencia
    LlamadasAMotor = CInt(5 * (DiferenciaDeFrecuencia - ToleranciaCuerda1))
    If LlamadasAMotor = 0 Then
      LlamadasAMotor = 1
    End If

      For ContadorMotor = 1 To LlamadasAMotor
      Call Forma1.Motor_Pulso_Izquierda(3)
      TiempoEntrePulsosMotor = True

      Do
      DoEvents
      Loop Until (TiempoEntrePulsosMotor = False)
      Next
      TimerModificarTension = True
      Do
      DoEvents
      Loop Until (TimerModificarTension = True)
      Timer5 = False
      ListoModificar.ZOrder
  End If


Else
  TimerEntreCuerdas = True
  Do
  DoEvents
  Loop Until (TimerEntreCuerdas = False)
```

```
  Call Servo_TurnOffServo(servoencuestion)
  TimerEntreCuerdas = True
  Do
  DoEvents
  Loop Until (TimerEntreCuerdas = False)
  GoTo Cuerda4

End If

  GoTo Cuerda3AR


'*************************************************************************************************
'*************************************************************************************************
'*************************************************************************************************
'-----------------------------------------Cuerda # 4-----------------------------------------
'*************************************************************************************************
'*************************************************************************************************
'*************************************************************************************************


Cuerda4:


LabelFrecuenciaMando.Caption = Afinacion.LabelFrecuenciaIII.Caption
LabelCuerdaNumero.Caption = "4"
'-------------------------------------------
Timer2 = True
EstadoLuna1 = 0
servoencuestion = "3"
Call Servo_TurnOnServoAndSetPosition(servoencuestion, 255)
'-------------------------------------------
EmpezarFFT = True
Do
DoEvents
Loop Until (EmpezarFFT = False)
Timer3 = True
Call Base.Proceso
Timer4 = False
ListoProcesar.ZOrder
If Base.Label5.Caption > 60 Then
LabelFrecuenciaActual.Caption = Format(Base.Label5.Caption / 3, "0.##")
Else
LabelFrecuenciaActual.Caption = Format(Base.Label5.Caption, "0.##")
End If

LabelFrecuenciaError = Format(Abs(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption), "0.##")
'-------------------------------------------

DiferenciaDeFrecuencia = CDbl(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption)

If Abs(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption) > ToleranciaCuerda4 Then
  If Abs(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption) > 10 Then
    TimerError = True
    Do
    DoEvents
    Loop Until (TimerError = False)
    GoTo Cuerda4
  End If
  Timer5 = True
  If (LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption) > 0 Then

    LlamadasAMotor = CInt(5 * (DiferenciaDeFrecuencia - ToleranciaCuerda1))
    If LlamadasAMotor = 0 Then
      LlamadasAMotor = 1
    End If

      For ContadorMotor = 1 To LlamadasAMotor
      Call Forma1.Motor_Pulso_Derecha(4)
      TiempoEntrePulsosMotor = True

      Do
      DoEvents
```

```
        Loop Until (TiempoEntrePulsosMotor = False)
        Next
        TimerModificarTension = True
        Do
        DoEvents
        Loop Until (TimerModificarTension = True)
        Timer5 = False
        ListoModificar.ZOrder
    Else
    DiferenciaDeFrecuencia = -DiferenciaDeFrecuencia
      LlamadasAMotor = CInt(3 * (DiferenciaDeFrecuencia - ToleranciaCuerda1))
       If LlamadasAMotor = 0 Then
         LlamadasAMotor = 1
       End If

        For ContadorMotor = 1 To LlamadasAMotor
        Call Forma1.Motor_Pulso_Izquierda(4)
        TiempoEntrePulsosMotor = True

        Do
        DoEvents
        Loop Until (TiempoEntrePulsosMotor = False)
        Next
        TimerModificarTension = True
        Do
        DoEvents
        Loop Until (TimerModificarTension = True)
        Timer5 = False
        ListoModificar.ZOrder
    End If


Else
    TimerEntreCuerdas = True
    Do
    DoEvents
    Loop Until (TimerEntreCuerdas = False)
    Call Servo_TurnOffServo(servoencuestion)
    TimerEntreCuerdas = True
    Do
    DoEvents
    Loop Until (TimerEntreCuerdas = False)
    GoTo Cuerda4AR

End If

    GoTo Cuerda4

'*****************************************************************************************************
'*****************************************************************************************************
'*****************************************************************************************************


Cuerda4AR:

AAV.ZOrder

LabelFrecuenciaMando.Caption = Afinacion.LabelFrecuenciaIII.Caption * 3
LabelCuerdaNumero.Caption = "4"
'-------------------------------------------
Timer2 = True
EstadoLuna1 = 0
servoencuestion = "3"
Call Servo_TurnOnServoAndSetPosition(servoencuestion, 255)
'-------------------------------------------
EmpezarFFT = True
Do
DoEvents
Loop Until (EmpezarFFT = False)
Timer3 = True
N Call Base.Proceso
Timer4 = False
ListoProcesar.Zorder
```

```
LabelFrecuenciaActual.Caption = Format(Base.Armo4.Caption, "0.##")

LabelFrecuenciaError = Format(Abs(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption), "0.##")
'-------------------------------------------------

DiferenciaDeFrecuencia = CDbl(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption)

If Abs(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption) > ToleranciaCuerda4 Then
  If Abs(LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption) > 10 Then
    TimerError = True
    Do
    DoEvents
    Loop Until (TimerError = False)
    GoTo Cuerda4AR
  End If
  Timer5 = True
  If (LabelFrecuenciaMando.Caption - LabelFrecuenciaActual.Caption) > 0 Then

    LlamadasAMotor = CInt(5 * (DiferenciaDeFrecuencia - ToleranciaCuerda1))
    If LlamadasAMotor = 0 Then
      LlamadasAMotor = 1
    End If

      For ContadorMotor = 1 To LlamadasAMotor
      Call Forma1.Motor_Pulso_Derecha(4)
      TiempoEntrePulsosMotor = True

      Do
      DoEvents
      Loop Until (TiempoEntrePulsosMotor = False)
      Next
      TimerModificarTension = True
      Do
      DoEvents
      Loop Until (TimerModificarTension = True)
      Timer5 = False
      ListoModificar.ZOrder
  Else
  DiferenciaDeFrecuencia = -DiferenciaDeFrecuencia
    LlamadasAMotor = CInt(3 * (DiferenciaDeFrecuencia - ToleranciaCuerda1))
    If LlamadasAMotor = 0 Then
      LlamadasAMotor = 1
    End If

      For ContadorMotor = 1 To LlamadasAMotor
      Call Forma1.Motor_Pulso_Izquierda(4)
      TiempoEntrePulsosMotor = True

      Do
      DoEvents
      Loop Until (TiempoEntrePulsosMotor = False)
      Next
      TimerModificarTension = True
      Do
      DoEvents
      Loop Until (TimerModificarTension = True)
      Timer5 = False
      ListoModificar.ZOrder
  End If


Else
  TimerEntreCuerdas = True
  Do
  DoEvents
  Loop Until (TimerEntreCuerdas = False)
  Call Servo_TurnOffServo(servoencuestion)
  TimerEntreCuerdas = True
  Do
  DoEvents
  Loop Until (TimerEntreCuerdas = False)
  GoTo MotoresAfinados
```

```
End If

  GoTo Cuerda4AR

'*****************************************************************************************************
'*****************************************************************************************************
'*****************************************************************************************************

MotoresAfinados:

Call Base.StartButton_Click
TiempoEntrePulsosMotor = True
Do
DoEvents
Loop Until (TiempoEntrePulsosMotor = False)
Call Servo_TurnOffServo("8")
TiempoEntrePulsosMotor = True
Do
DoEvents
Loop Until (TiempoEntrePulsosMotor = False)
Call Servo_TurnOffServo("7")
TiempoEntrePulsosMotor = True
Do
DoEvents
Loop Until (TiempoEntrePulsosMotor = False)
Call Servo_TurnOffServo("6")
TiempoEntrePulsosMotor = True
Do
DoEvents
Loop Until (TiempoEntrePulsosMotor = False)
Call Servo_TurnOffServo("5")
TiempoEntrePulsosMotor = True
Do
DoEvents
Loop Until (TiempoEntrePulsosMotor = False)
Call Base.StopButton_Click


AfinacionLista.ZOrder
Afinando = False




End Sub

Private Sub EmpezarFFT_Timer()

EmpezarFFT = False

End Sub

Private Sub Form_Load()

Afinacion.Show
Base.Show
Forma1.Show

Image14.ZOrder
Image27.ZOrder
Image35.ZOrder
Image38.ZOrder

AfinandoAhora.ZOrder


Afinando = False


ToleranciaCuerda1 = 0.3
ToleranciaCuerda2 = 0.3
ToleranciaCuerda3 = 0.3
ToleranciaCuerda4 = 0.3
```

```
End Sub


Private Sub Image2_Click()

End Sub

Private Sub Image5_Click()

End Sub

Private Sub TiempoEntrePulsosMotor_Timer()
TiempoEntrePulsosMotor = False

End Sub

Private Sub Timer2_Timer()
luna1 = luna1 + 1


Select Case luna1

        Case 0
           Image13.ZOrder
        Case 1
           Image14.ZOrder
        Case 2
           Image15.ZOrder
        Case 3
           Image16.ZOrder
        Case 4
           Image17.ZOrder
        Case 5
           Image18.ZOrder
        Case 6
           Image19.ZOrder
        Case 7
           Image20.ZOrder
        luna1 = 0
        EstadoLuna1 = EstadoLuna1 + 1

End Select
If EstadoLuna1 >= 2 Then
Timer2 = False
ListoTocar.ZOrder
End If

End Sub

Private Sub Timer3_Timer()
luna2 = luna2 + 1


Select Case luna2

        Case 0
           Image28.ZOrder
        Case 1
           Image27.ZOrder
        Case 2
           Image26.ZOrder
        Case 3
           Image25.ZOrder
        Case 4
           Image24.ZOrder
        Case 5
           Image23.Zorder
```

```
        Case 6
           Image22.ZOrder
        Case 7
           Image21.ZOrder
        luna2 = 0

End Select

End Sub

Private Sub Timer4_Timer()
luna3 = luna3 + 1


Select Case luna3

        Case 0
           Image36.ZOrder
        Case 1
           Image35.ZOrder
        Case 2
           Image34.ZOrder
        Case 3
           Image33.ZOrder
        Case 4
           Image32.ZOrder
        Case 5
           Image31.ZOrder
        Case 6
           Image30.ZOrder
        Case 7
           Image29.ZOrder
        luna3 = 0

End Select

End Sub

Private Sub Timer5_Timer()
luna4 = luna4 + 1


Select Case luna4

        Case 0
           Image37.ZOrder
        Case 1
           Image38.ZOrder
        Case 2
           Image39.ZOrder
        Case 3
           Image40.ZOrder
        Case 4
           Image41.ZOrder
        Case 5
           Image42.ZOrder
        Case 6
           Image43.ZOrder
        Case 7
           Image44.ZOrder
        luna4 = 0

End Select

End Sub
```

```
Private Sub TimerEntreCuerdas_Timer()
TimerEntreCuerdas = False

End Sub

Private Sub TimerError_Timer()
TimerError = False
End Sub

Private Sub TimerModificarTension_Timer()
TimerModificarTension = False

End Sub
```

Clase 1: Timer

Utilizada para controlar confiablemente el tiempo del proceso. Se realiza por medio de lecturas al reloj interno del sistema.

```
Option Explicit
Private Declare Function timeGetTime Lib "winmm.dll" () As Long

Public TiempoInicial As Long

Public Sub StartTimer()
  TiempoInicial = timeGetTime
End Sub

Public Function StopTimer() As Long
  StopTimer = timeGetTime - TiempoInicial
End Function

Public Function CheckTimer()
  CheckTimer = timeGetTime - TiempoInicial
End Function
```

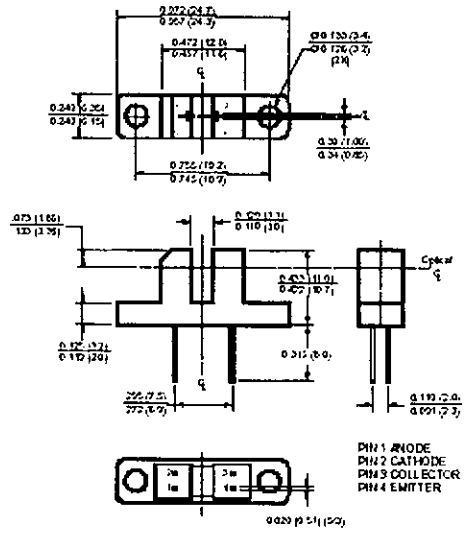
ANEXO E

Hoja de datos del H21A

FAIRCHILD
SEMICONDUCTOR®

H21A1 / H21A2 / H21A3
PHOTOTRANSISTOR
OPTICAL INTERRUPTER SWITCH

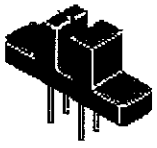
PACKAGE DIMENSIONS



NOTES:

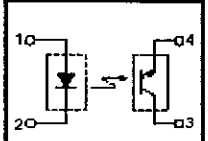
1. Dimensions for all drawings are in inches (mm).
2. Tolerance of ± .010 (.25) on all non-nominal dimensions unless otherwise specified.

DESCRIPTION



The H21A1, H21A2 and H21A3 consist of a gallium arsenide infrared emitting diode coupled with a silicon phototransistor in a plastic housing. The packaging system is designed to optimize the mechanical resolution, coupling efficiency, ambient light rejection, cost and reliability. The gap in the housing provides a means of interrupting the signal with an opaque material, switching the output from an "ON" to an "OFF" state.

SCHEMATIC



FEATURES

- Opaque housing
- Low cost
- .035" apertures
- High $I_{C(Off)}$

1. Derate power dissipation linearly 1.33 mW/°C above 25°C.
2. RMA flux is recommended.
3. Methanol or isopropyl alcohols are recommended as cleaning agents.
4. Soldering iron tip 1/16" (1.6mm) minimum from housing.

ABSOLUTE MAXIMUM RATINGS (T_A = 25°C unless otherwise specified)

Parameter	Symbol	Rating	Unit
Operating Temperature	T_{OPR}	-55 to +100	°C
Storage Temperature	T_{STG}	-55 to +100	°C
Soldering Temperature (Iron)[2,3 and 4]	T_{SOL-I}	240 for 5 sec	°C
Soldering Temperature (Flow)[2 and 3]	T_{SOL-F}	260 for 10 sec	°C
INPUT (EMITTER) Continuous Forward Current	I_F	50	mA
Reverse Voltage	V_R	6	V
Power Dissipation [1]	P_D	100	mW
OUTPUT (SENSOR) Collector to Emitter Voltage	V_{CEO}	30	V
Emitter to Collector Voltage	V_{ECO}	4.5	V
Collector Current	I_C	20	mA
Power Dissipation (T_C = 25°C)[1]	P_D	150	mW

FAIRCHILD
SEMICONDUCTOR®

H21A1 / H21A2 / H21A3
PHOTOTRANSISTOR
OPTICAL INTERRUPTER SWITCH

ELECTRICAL / OPTICAL CHARACTERISTICS (T_A =25°C)(All measurements made under pulse condition)

PARAMETER	TEST CONDITIONS	SYMBOL	DEVICES	MIN	TYP	MAX	UNITS
INPUT (EMITTER) Forward Voltage	I_F = 60 mA	V_F	All	—	—	1.7	V
Reverse Breakdown Voltage	I_R = 10 µA	V_R	All	6.0	—	—	V
Reverse Leakage Current	V_R = 3 V	I_R	All	—	—	1.0	µA
OUTPUT (SENSOR) Emitter to Collector Breakdown	I_F = 100 µA, Ee = 0	BV_{ECO}	All	6.0	—	—	V
Collector to Emitter Breakdown	I_C = 1 mA, Ee = 0	BV_{CEO}	All	30	—	—	V
Collector to Emitter Leakage	V_{CE} = 25 V, Ee = 0	I_{CEO}	All	—	—	100	nA
COUPLED On-State Collector Current	I_F = 5 mA, V_{CE} = 5 V	$I_{C(ON)}$	H21A1	0.15	—	—	mA
			H21A2	0.30	—	—	
			H21A3	0.60	—	—	
	I_F = 20 mA, V_{CE} = 5 V		H21A1	1.0	—	—	
			H21A2	2.0	—	—	
			H21A3	4.0	—	—	
	I_F = 30 mA, V_{CE} = 5 V		H21A1	1.9	—	—	
			H21A2	3.0	—	—	
			H21A3	5.5	—	—	
Saturation Voltage	I_F = 20 mA, I_C = 1.8 mA	$V_{CE(SAT)}$	H21A2/3	—	—	0.40	V
	I_F = 30 mA, I_C = 1.8 mA		H21A1	—	—	0.40	V
Turn-On Time	I_F = 30 mA, V_{CC} = 5 V, R_L = 2.5 KΩ	t_{on}	All	—	8	—	µs
Turn-Off Time	I_F = 30 mA, V_{CC} = 5 V, R_L = 2.5 KΩ	t_{off}	All	—	50	—	µs

FAIRCHILD
SEMICONDUCTOR®

H21A1 / H21A2 / H21A3
PHOTOTRANSISTOR
OPTICAL INTERRUPTER SWITCH

Figure 1. Output Current vs. Input Current

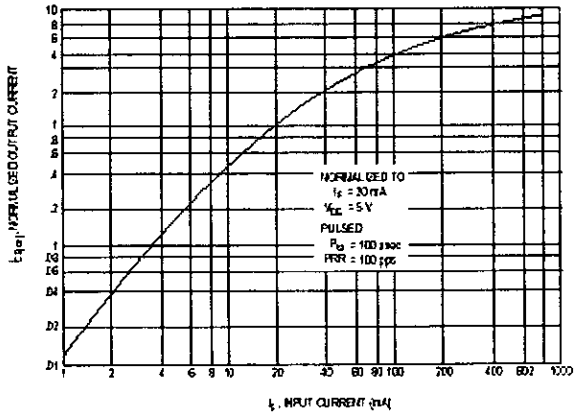


Figure 2. Output Current vs. Temperature

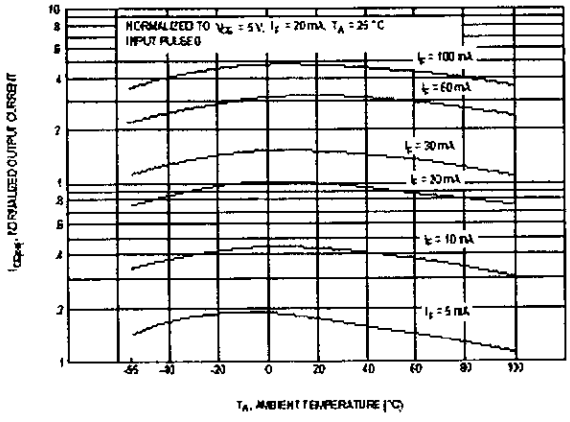
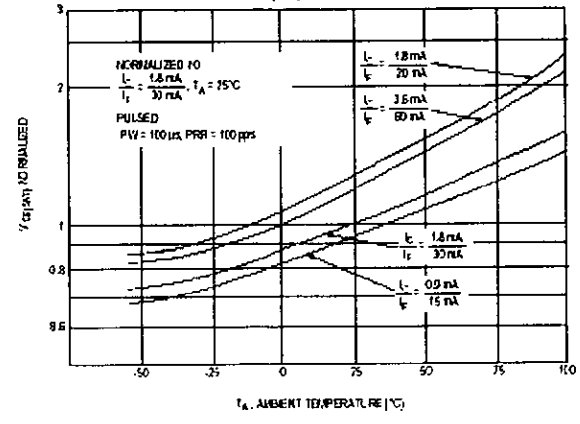


Figure 3. $V_{CE(SAT)}$ vs. Temperature



FAIRCHILD
SEMICONDUCTOR®

H21A1 / H21A2 / H21A3
PHOTOTRANSISTOR
OPTICAL INTERRUPTER SWITCH

Figure 4. Leakage Current vs. Temperature

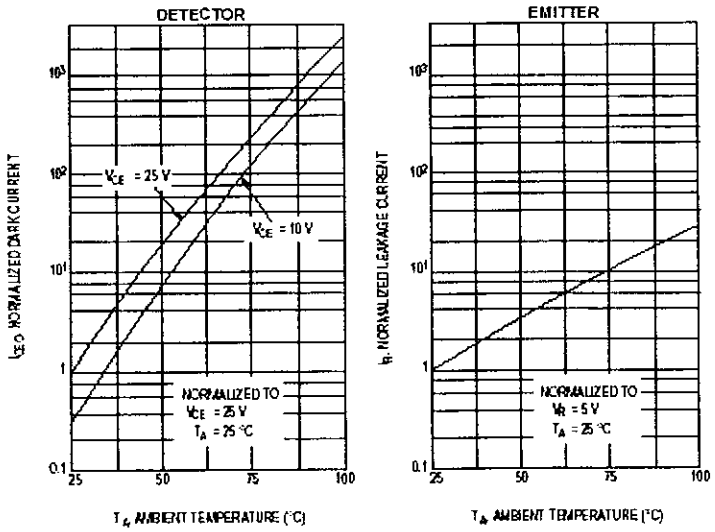


Figure 5. Switching Speed vs. RL

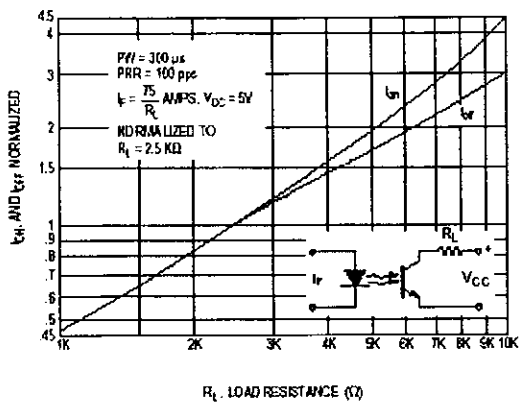
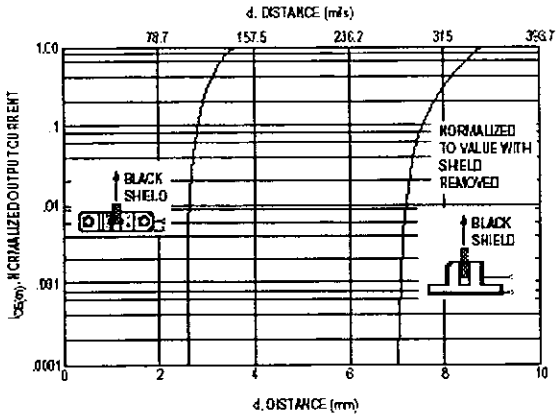


Figure 6. Output Current vs. Distance



FAIRCHILD
SEMICONDUCTOR

H21A1 / H21A2 / H21A3
PHOTOTRANSISTOR OPTICAL INTERRUPTER SWITCH

DISCLAIMER
FAIRCHILD SEMICONDUCTOR RESERVES THE RIGHT TO MAKE CHANGES WITHOUT FURTHER NOTICE TO ANY PRODUCTS HEREIN TO IMPROVE RELIABILITY, FUNCTION OR DESIGN. FAIRCHILD DOES NOT ASSUME ANY LIABILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT OR CIRCUIT DESCRIBED HEREIN; NEITHER DOES IT CONVEY ANY LICENSE UNDER ITS PATENT RIGHTS, NOR THE RIGHTS OF OTHERS.

LIFE SUPPORT POLICY
FAIRCHILD'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF FAIRCHILD SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body,or (b) support or sustain life, and (c) whose failure to perform when properly used in accordance with instructions for use provided in labeling, can be reasonably expected to result in a significant injury of the user.
2. A critical component in any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

ANEXO F

Hoja de datos del Servo 8 torque board

Servo 8 Torque Board Doc V 1.2

RS-232 hobby servo controller with torque feedback
No servo modifications required

Features:

- Eight independent 8-bit servo control outputs allow 254 positions for each servo.
- Standard 90 degree or extended 180 degree rotation range.
- Servos can be turned on or off.
- Eight independent 16-bit servo torque input values using standard hobby servos.
- Eight independent servo home positions.
- Daisy chain connector allows independent control of up to 8 boards from a single RS-232 serial line.
- ServoGUI included. Free source code.
- CPU power range is 5.5 to 9 VDC.
- Servo power range is 5 to 6 VDC.
- CPU and servos can share power, or they can use two separate power inputs for increased isolation. Shared power must be 6 VDC if used.

Method of operation

The torque board measures torque by analyzing the servo. Here's how it works -- when you send a command to a servo to move to a certain position, the servo needs to figure out how far it should move. It does this by taking the difference between its current position and commanded position. This difference is called the error.

The servo's job in life is to keep the error as close to zero as possible. It does this by applying power to its internal motor in such a way as to move the output arm in the direction that minimizes the error.

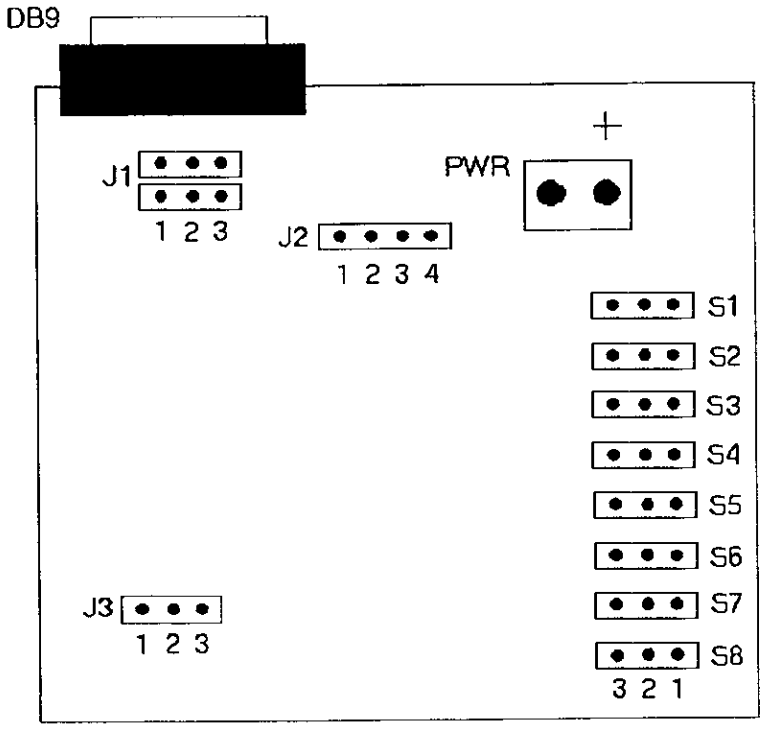
Consider the case where the servo is holding a constant position, and an external torque is applied to the output arm. If you assume an ideal servo with a frictionless gear train, the torque will cause the output arm to move slightly, and the absolute value of the error will increase to a nonzero value. In attempting to drive the error to zero, the servo will apply power to the motor to oppose the torque. The CPU on the board analyzes this condition.

In a real servo, the gear train will have a certain amount of friction. The friction needs to be taken into consideration, since the torque being measured is actually at the motor shaft rather than the output arm.

The presence of friction can sometimes be used to advantage, since it can cause hysteresis when the servo movement changes direction. The servo board actually reports the absolute value of torque, not the torque itself, which means it is possible to infer the torque sign by comparing the torque readings when approaching a position from two different directions.

Servos generally need to be calibrated if you want to be able to quantitatively relate the actual torque to the numerical torque values reported by the board. For a typical calibration, you would apply a known torque to the servo, as measured in units of Newton-meters or ounce-inches, then read the value reported by the servo board. Calibration curves will generally vary, depending on the design and size of each servo.

Servo 8 Torque Board Configuration



Connectors and Jumpers:

DB-9: 9-pin female RS-232 connector.

PWR: Servo power, +5 to 6.5 VDC. Also supplies CPU power if shared power option is used.

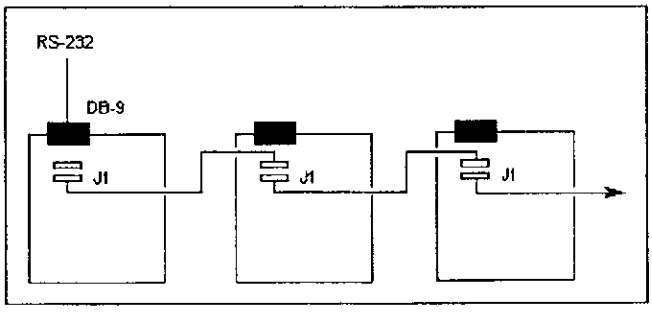
J1: RS-232 daisy chain connector. Default is open.

J2: Configures CPU power.

J3: Allows changes to module address and baud rate. Default is open.

S1-S8: Servo connectors. Pin 1 is signal, pin 2 is positive power, pin 3 is ground.

Jumper	Position	Purpose
J2	1 to 2	CPU shares power with servos (default)
	Open	Pin 2 is +5 to 9 VDC CPU power, pin 3 is ground
J3	Open	Write-protect module address and baud rate (default)
	1 to 2	Restore factory settings for module address & baud rate
	2 to 3	Allow changes to module address and baud rate



DAISY CHAIN CONFIGURATION

A daisy chain allows you to use a single serial line to control up to 8 modules. You can chain modules together by connecting three in-row pins on jumper J1 to the corresponding pins on adjacent modules. J1 is internally connected to the DB-9 as follows:

J1 Pin	DB-9 Pin	Function
1	5	Ground
3	3	Transmit data
2	2	Receive data

Note: The daisy chain connector (J1) connects directly to the DB-9 and is not TTL level.

COMMAND FORMAT

Each command is a sequence of bytes sent to the servo. The length of the sequence is four or five bytes, depending on the command.

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Header	Module Address	Servo Address	Command	Data (optional)

All commands:

Header: ">" (ASCII 62)

Module Address: "1" .. "8" (ASCII 49 .. 56)

Servo Address: "1" .. "8" (ASCII 49 .. 56)

Send multiple position commands:

Command: "m" (ASCII 109)

Data: 8 separate bytes representing servo position 1 - 8 (binary)

Turn on servo and set position:

Command: "a" (ASCII 97)

Data: 1 .. 255 (binary) position. Turns on pulse train to servo.

Turn off servo:

Command: "a" (ASCII 97)

Data: 0 (binary). Turns off pulse train to servo.

Set current position as home:

Command: "c" (ASCII 99)

Data: (not used)

Go to home position:

Command: "h" (ASCII 104)

Data: (not used)

Set width resolution

Command: "w" (ASCII 119)

Data: "0" (ASCII 48) = standard
"1" (ASCII 49) = extended

Get position

Command: "g" (ASCII 103)

Data: (not used)

Get torque

Command: "t" (ASCII 116)

Data: (not used)

Set module address

Command: "s" (ASCII 115)

Data: "1" .. "8" (ASCII 49..56) new address

Remarks: Jumper J3 pins 2 and 3 must be connected for this command.

Set baud rate

Command: "b" (ASCII 98)

Data: "0" (ASCII 48) = 19200 baud
"1" (ASCII 49) = 9600 baud
"2" (ASCII 50) = 4800 baud
"3" (ASCII 51) = 2400 baud

Remarks: Jumper J3 pins 2 and 3 must be connected for this command.

Example command:

Byte #	1	2	3	4	5
ASCII	">"	"1"	"2"	"a"	--
Binary	62	49	50	97	255

This command goes to module #1, turns on servo #2 and sets the servo position to 255, which is one endpoint of the servo travel range.

RESPONSES TO COMMANDS

With two exceptions, the servo board responds to all commands with a carriage return character (Dec 13). The exceptions are as follows:

(1) The response to the "get position" command is a 1-byte binary position, followed by carriage return.

(2) The response to the "get torque" command is a 2-byte binary torque value, followed by carriage return. The first and second data bytes form the lower and upper byte of a 16-bit unsigned integer torque value. The precise relationship between measured torque and actual torque depends on many things, such as the type of servo.

Note: In processors with buffered serial ports such as the BasicX, you should periodically clear the serial input buffer to prevent the accumulation of the servo board's (Dec 13) responses from overflowing the serial port buffer.

CHANGING BOARD ADDRESS AND BAUD RATE

Procedure for changing the board address or baud rate:

1. Turn on power.
2. On jumper J3, connect pins 2 and 3.
3. Send the "set module address" or "set baud rate" software command.
4. Turn off power.
5. Move jumper J3 to the center-off position (no pins connected).
6. Turn on power.

The board is factory-set to board address 1 and 19200 baud rate. If you want to restore the factory settings, use the following procedure:

1. On jumper J3, connect pins 1 and 2. Power can be on or off.
2. Turn on power if it's not already on.
3. Turn off power.
4. Move jumper J3 to the center-off position (no pins connected).

SERVO CONTROL SIGNAL BASICS

Each servo is controlled by sending a train of pulses over the signal wire, which is pin 2 of the servo connectors. The pulse rate varies somewhat, but is a minimum of 50 Hz.

The servo position is controlled by the pulse width. In standard mode, the pulse width varies between 1 ms and 2 ms, with the center at 1.5 ms. In extended mode, the pulse width varies between 0.48 ms and 2.52 ms, with a center again at 1.5 ms.

When you send a command to turn off the servo, the pulse train is halted, although power continues to be applied to the internal servo electronics. Absence of a pulse train usually means that the servo motor is left unpowered, and the servo will not try to maintain any position – in other words, the servo is free to move.

Servo 8 Torque Board FAQ

Frequently Asked Questions

1. What size power supply do I need?
You will need 1 Amp of current for each connected servo.

2. How do I verify that the board is working without a computer attached?
Turn off power to the board. Plug a servo into connector S1, turn the servo arm away from 90 degrees and then apply power. The servo should center to 90 degrees, which indicates the board is controlling the servo.

3. What are the default settings?
The factory default settings are board address 1, servo home position set to 127 (center position) and the baud rate is 19200,N,8,1.

4. How do I reset the board to its default settings?
On jumper J3, connect pins 1 and 2. Turn power on if it's not already on. Turn off power. Move jumper J3 to the center-off position (no pins connected).

5. How do I change the board baud rate?
Turn on power. On jumper J3, connect pins 2 and 3. Send the "set baud rate" software command (you can use the ServoGUI for this). Turn off power. Move jumper J3 to the center-off position (no pins connected). Turn on power. The board is now set to your selected baud rate.

6. How do I change the board address?
Turn on power. On jumper J3, connect pins 2 and 3. Send the "set module address" software command (you can use the ServoGUI for this). Turn off power. Move jumper J3 to the center-off position (no pins connected). Turn on power. The board is now set to your selected baud rate.

7. Why does the LED image change to red when using the ServoGUI?
When using shared power it is possible that the servos can cause a voltage drop that will power down the board, which needs a solid 5.2-VDC level. This is why we suggest 6 VDC @ 1Amp minimum when using the shared power jumper.

8. How do I use separate power?
Remove the jumper on J2 pins 1 and 2, which makes it possible to put up to 9 VDC on J2 pin 2 and Gnd on Pin 3, The servo PWR connector should never exceed 6.5 VDC. In this configuration it's possible to use a 9 VDC battery on J2 and four D-cell type batteries on the servo PWR connector.

9. How do I disable a servo?
When you send the "turn off servo" command (which is equivalent to sending a servo position command of zero) this turns the pulse train off for the selected servo. This minimizes drain on the batteries as well.

NetMedia, Inc.
10940 N. Stallard Place
Tucson, Arizona 85737

Tel: 520-544-4567
Fax: 520-544-0800

E-mail: Sales@Netmedia.com

ANEXO G

Codigo utilizado en el módulo de exitacion

```
        LIST    P=16F877
        INCLUDE <P16F877.INC>

;__CONFIG _CP_OFF & _DEBUG_OFF & _WRT_ENABLE_ON & _CPD_OFF & _LVP_OFF &
_BODEN_ON & _PWRTE_ON & _WDT_OFF & _HS_OSC

;
;*****VARIABLES*****
****

        W_TEMP          EQU     H'0020'
        STATUS_TEMP     EQU     H'0021'
        BANDERA         EQU H'0022'
        PDel0   EQU H'0023'
PDel1   EQU H'0024'
        CONTADOR001     EQU     H'0025'
        PTOBTEMP        EQU     H'0026'

;*****
***



        org     0

                GOTO    MAIN

        org     4

INTERRUPT

        GUARDA_W_S_P

        MOVWF   W_TEMP
        SWAPF   STATUS,W
        CLRF    STATUS
        MOVWF   STATUS_TEMP


        RESTAURA_W_S_P

        SWAPF   STATUS_TEMP,W
        MOVWF   STATUS
        SWAPF   W_TEMP,F
        SWAPF   W_TEMP,W

RETFIE



;-----------------------------------------------
; Generado con PDEL ver SP  r 1.0  el 23/10/02 Hs 12:52:19 p.m.
; Descripcion: Delay 500000 ciclos
;-----------------------------------------------
DEMORA100
                                BCF     STATUS,RP0
                                BCF     STATUS,RP1

                                movlw   229     ; 1 set numero de repeticion  (B)
        movwf   PDel0   ; 1 |
PLoop10 movlw   232     ; 1 set numero de repeticion  (A)
        movwf   PDel1   ; 1 |
PLoop20 clrwdt          ; 1 clear watchdog
PDelL10 goto PDelL20    ; 2 ciclos delay
PDelL20 goto PDelL30    ; 2 ciclos delay
PDelL30 clrwdt          ; 1 ciclo delay
        decfsz  PDel1, 1  ; 1 + (1) es el tiempo 0  ?(A)
        goto    PLoop20 ; 2 no, loop
        decfsz  PDel0,  1 ; 1 + (1) es el tiempo 0  ?(B)
        goto    PLoop10 ; 2 no, loop
PDelL40 goto PDelL50    ; 2 ciclos delay
PDelL50 goto PDelL60    ; 2 ciclos delay
PDelL60 goto PDelL70    ; 2 ciclos delay
PDelL70 clrwdt          ; 1 ciclo delay
        return          ; 2+2 Fin.
;-----------------------------------------------

;-----------------------------------------------
; Generado con PDEL ver SP  r 1.0  el 24/10/03 Hs 07:31:49 a.m.
; Descripcion: Delay 50000 ciclos
;-----------------------------------------------
DEMORA10
                                BCF     STATUS,RP0
                                BCF     STATUS,RP1


                                movlw   55      ; 1 set numero de repeticion  (B)
        movwf   PDel0   ; 1 |
PLoop11 movlw   181     ; 1 set numero de repeticion  (A)
        movwf   PDel1   ; 1 |
PLoop21 clrwdt          ; 1 clear watchdog
        clrwdt          ; 1 ciclo delay
        decfsz  PDel1, 1  ; 1 + (1) es el tiempo 0  ?(A)
        goto    PLoop21 ; 2 no, loop
        decfsz  PDel0,  1 ; 1 + (1) es el tiempo 0  ?(B)
        goto    PLoop11 ; 2 no, loop
        return          ; 2+2 Fin.
;-----------------------------------------------

;-----------------------------------------------
; Generado con PDEL ver SP  r 1.0  el 24/10/03 Hs 07:35:48 a.m.
; Descripcion: Delay 5000 ciclos
;-----------------------------------------------
DEMORA1
                                BCF     STATUS,RP0
                                BCF     STATUS,RP1


                                movlw   6       ; 1 set numero de repeticion  (B)
        movwf   PDel0   ; 1 |
PLoop12 movlw   207     ; 1 set numero de repeticion  (A)
        movwf   PDel1   ; 1 |
PLoop22 clrwdt          ; 1 clear watchdog
        decfsz  PDel1, 1  ; 1 + (1) es el tiempo 0  ?(A)
        goto    PLoop22 ; 2 no, loop
        decfsz  PDel0,  1 ; 1 + (1) es el tiempo 0  ?(B)
        goto    PLoop12 ; 2 no, loop
PDelL12 goto            PDelL22 ; 2 ciclos delay
PDelL22 clrwdt          ; 1 ciclo delay
        return          ; 2+2 Fin.
;-----------------------------------------------



        org     70
MAIN:

        INICIALIZACION_DE_VARIABLES

        BCF     STATUS,RP0
        BCF     STATUS,RP1

        CLRF    W_TEMP
        CLRF    STATUS_TEMP


        CONFIGURACION_GENERAL


;***********BANK 0**********************

        BCF     STATUS,RP0
        BCF     STATUS,RP1

        BCF     INTCON,GIE
        BCF     INTCON,PEIE
        BCF     INTCON,RBIE
        BCF     INTCON,INTE
        BCF     INTCON,T0IE


        MOVLW   B'00000000'
        MOVWF   T1CON
        MOVWF   T2CON
        MOVWF   SSPCON
        MOVWF   ADCON0

        BSF     RCSTA,SPEN      ;SERIAL PORT ENABLED
        BCF     RCSTA,RX9       ;8 BIT RX
        BCF     RCSTA,ADDEN     ;ADDRES DETECT DISABLED
        BCF     RCSTA,FERR      ;FRAMING ERROR DISABLED
        BCF     RCSTA,OERR      ;OVEERRUN ERRDR DISABLED


;***********BANK 1**********************

        BSF     STATUS,RP0
        BCF     STATUS,RP1


        BCF     PIE1,PSPIE
        BCF     PIE1,ADIE
        BCF     PIE1,RCIE  ;RECEIVE INTERRUPT DISABLED
        BCF     PIE1,TXIE
        BCF     PIE1,SSPIE
        BCF     PIE1,CCP1IE
        BCF     PIE1,TMR2IE
        BCF     PIE1,TMR1IE
        BCF     PIE2,EEIE
        BCF     PIE2,BCLIE
        BCF     PIE2,CCP2IE


        MOVLW   B'00000000'
        MOVWF   TRISA
        MOVWF   TRISE

        MOVLW   B'11111111';ENTRADA INT
        MOVWF   TRISB

        MOVLW   B'00000000';SALIDA DATA
        MOVWF   TRISD

        MOVWF   SSPCON2

        MOVLW   B'10000000';SALIDA TX ENABLE
        MOVWF   TRISC

        BCF     TXSTA,TX9       ;8 BIT TX
        BCF     TXSTA,SYNC      ;ASINCRONOUS ENABLE
        BSF     TXSTA,BRGH      ;HIGH BAUD RATE
        MOVLW   64              ;BAUD RATE 19.2 KBAUD
        MOVWF   SPBRG
```

```
        BSF     TXSTA,BRGH      ;HIGH BAUD RATE
        MOVLW   .129            ;BAUD RATE 9.6 KBAUD
        MOVWF   SPBRG


        BCF     STATUS,RP0
        BCF     STATUS,RP1
;*****************************************************************


INICIALIZARSERVOS

        BSF             PORTD,2
        BCF             PORTD,3

        CLRF    PTOBTEMP
        CALL    ApagarServosTodos
        CALL    DEMORA100

MOTOR1PREVIO

        CALL    MOVERSERVO1
        BCF             PTOBTEMP,4
        CALL    DEMORA100
        CALL    DEMORA100
        CALL    DEMORA100

MOTOR1

        BTFSC   PORTB,4
        CALL    PARARSERVO1
        BTFSS   PTOBTEMP,4
        GOTO    MOTOR1
        BTFSS   PORTB,4
        GOTO    MOTOR1PREVIO


MOTOR2PREVIO

        CALL    MOVERSERVO2
        BCF             PTOBTEMP,5
        CALL    DEMORA100
        CALL    DEMORA100
        CALL    DEMORA100

MOTOR2

        BTFSC   PORTB,5
        CALL    PARARSERVO2
        BTFSS   PTOBTEMP,5
        GOTO    MOTOR2
        BTFSS   PORTB,5
        GOTO    MOTOR2PREVIO

MOTOR3PREVIO

        CALL    MOVERSERVO3
        BCF             PTOBTEMP,6
        CALL    DEMORA100
        CALL    DEMORA100
        CALL    DEMORA100

MOTOR3

        BTFSC   PORTB,6
        CALL    PARARSERVO3
        BTFSS   PTOBTEMP,6
        GOTO    MOTOR3
        BTFSS   PORTB,6
        GOTO    MOTOR3PREVIO

MOTOR4PREVIO

        CALL    MOVERSERVO4
        BCF             PTOBTEMP,7
        CALL    DEMORA100
        CALL    DEMORA100
        CALL    DEMORA100

MOTOR4

        BTFSC   PORTB,7
        CALL    PARARSERVO4
        BTFSS   PTOBTEMP,7
        GOTO    MOTOR4
        BTFSS   PORTB,7
        GOTO    MOTOR4PREVIO


        MOTORES_EN_POSICION

        BTFSS   PORTB,4
        GOTO    MOTOR1PREVIO
        BTFSS   PORTB,5
        GOTO    MOTOR2PREVIO
        BTFSS   PORTB,6
        GOTO    MOTOR3PREVIO
        BTFSS   PORTB,7
        GOTO    MOTOR4PREVIO


        BCF             PORTD,2
        BSF             PORTD,3

        ESTADO_ESPERA_DE_CAMBIO

        BTFSS   PORTB,4
        GOTO    MOVIMIENTO_MOTOR_1_DETECTADO
        BTFSS   PORTB,5
        GOTO    MOVIMIENTO_MOTOR_2_DETECTADO
        BTFSS   PORTB,6
        GOTO    MOVIMIENTO_MOTOR_3_DETECTADO
        BTFSS   PORTB,7
        GOTO    MOVIMIENTO_MOTOR_4_DETECTADO
        GOTO    ESTADO_ESPERA_DE_CAMBIO

MOVIMIENTO_MOTOR_1_DETECTADO
        CALL    DEMORA100
        CALL    DEMORA100
MOVIMIENTO_MOTOR_1_DETECTADOA

        NOP
        NOP
        NOP
        NOP

        BTFSS   PORTB,4
        GOTO    MOVIMIENTO_MOTOR_1_DETECTADOA
        MOTOR1PREVIOA

        CALL    PARARSERVO1
        BTFSS   PORTB,4
        GOTO    MOTOR1PREVIOA
        GOTO    ESTADO_ESPERA_DE_CAMBIO


MOVIMIENTO_MOTOR_2_DETECTADO
        CALL    DEMORA100
        CALL    DEMORA100
MOVIMIENTO_MOTOR_2_DETECTADOA

        NOP
        NOP
        NOP
        NOP

        BTFSS   PORTB,5
        GOTO    MOVIMIENTO_MOTOR_2_DETECTADOA
        MOTOR2PREVIOA

        CALL    PARARSERVO2
        BTFSS   PORTB,5
        GOTO    MOTOR2PREVIOA
        GOTO    ESTADO_ESPERA_DE_CAMBIO


MOVIMIENTO_MOTOR_3_DETECTADO
        CALL    DEMORA100
        CALL    DEMORA100
MOVIMIENTO_MOTOR_3_DETECTADOA

        NOP
        NOP
        NOP
        NOP

        BTFSS   PORTB,6
        GOTO    MOVIMIENTO_MOTOR_3_DETECTADOA
        MOTOR3PREVIOA

        CALL    PARARSERVO3
        BTFSS   PORTB,6
        GOTO    MOTOR3PREVIOA
        GOTO    ESTADO_ESPERA_DE_CAMBIO


MOVIMIENTO_MOTOR_4_DETECTADO
        CALL    DEMORA100
        CALL    DEMORA100
MOVIMIENTO_MOTOR_4_DETECTADOA

        NOP
        NOP
        NOP
        NOP

        BTFSS   PORTB,7
        GOTO    MOVIMIENTO_MOTOR_4_DETECTADOA
        MOTOR4PREVIOA

        CALL    PARARSERVO4
        BTFSS   PORTB,7
        GOTO    MOTOR4PREVIOA
        GOTO    ESTADO_ESPERA_DE_CAMBIO




ORG .512


;*****************************************************************
;*****************************************************************
ApagarServosTodos

        BCF     STATUS,RP0
        BCF     STATUS,RP1

        MOVLW   B'00001000'
        MOVWF   CONTADOR001


LOOP001

        BSF     STATUS,RP0
        BCF     STATUS,RP1

        BSF     TXSTA,TXEN      ;ENABLE TX

        BCF     STATUS,RP0
        BCF     STATUS,RP1
```

```
MOVLW   A'>'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE001

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE001

BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   A'1'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE002

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE002


BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

DECF    CONTADOR001
MOVFW   CONTADOR001
ADDLW   .49

MOVWF   TXREG

INCF    CONTADOR001

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE003

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE003


BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   A'a'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE004

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE004


BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   B'00000000'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE005

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO    ASE005

CALL DEMORA100


BCF     STATUS,RP0
BCF     STATUS,RP1

DECFSZ  CONTADOR001

GOTO LOOP001

RETURN

;*****************************************************
;*****************************************************

MOVERSERVO1

BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   A'>'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE006

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE006

BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   A'1'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE007

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE007


BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   '1'
MOVWF   TXREG

INCF    CONTADOR001

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE008

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE008


BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   A'a'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE009

NOP
NOP
NOP
```

```
        NOP

        BTFSS   TXSTA,TRMT
        GOTO ASE009

        BSF     STATUS,RP0
        BCF     STATUS,RP1

        BSF     TXSTA,TXEN      ;ENABLE TX

        BCF     STATUS,RP0
        BCF     STATUS,RP1

        MOVLW   B'11111111'
        MOVWF   TXREG

        BSF     STATUS,RP0
        BCF     STATUS,RP1

        ASE010

        NOP
        NOP
        NOP
        NOP

        BTFSS   TXSTA,TRMT
        GOTO    ASE010

        BCF     STATUS,RP0
        BCF     STATUS,RP1

        CALL DEMORA1
        CALL DEMORA1
        CALL DEMORA1

RETURN

;*****************************************************
;*****************************************************

        PARARSERVO1

        CALL DEMORA10
        CALL DEMORA10
        CALL DEMORA10
        CALL DEMORA10
        CALL DEMORA10
        CALL DEMORA10

        BCF     STATUS,RP0
        BCF     STATUS,RP1

        BSF             PTOBTEMP,4


        BSF     STATUS,RP0
        BCF     STATUS,RP1

        BSF     TXSTA,TXEN      ;ENABLE TX

        BCF     STATUS,RP0
        BCF     STATUS,RP1

        MOVLW   A'>'
        MOVWF   TXREG

        BSF     STATUS,RP0
        BCF     STATUS,RP1

        ASE011

        NOP
        NOP
        NOP
        NOP

        BTFSS   TXSTA,TRMT
        GOTO ASE011

        BSF     STATUS,RP0
        BCF     STATUS,RP1

        BSF     TXSTA,TXEN      ;ENABLE TX

        BCF     STATUS,RP0
        BCF     STATUS,RP1

        MOVLW   A'1'
        MOVWF   TXREG

        BSF     STATUS,RP0
        BCF     STATUS,RP1

        ASE012

        NOP
        NOP
        NOP
        NOP

        BTFSS   TXSTA,TRMT
        GOTO ASE012


        BSF     STATUS,RP0
        BCF     STATUS,RP1

        BSF     TXSTA,TXEN      ;ENABLE TX

        BCF     STATUS,RP0
        BCF     STATUS,RP1

        MOVLW   '1'
        MOVWF   TXREG

        INCF    CONTADOR001

        BSF     STATUS,RP1
        BCF     STATUS,RP1

        ASE013

        NOP
        NOP
        NOP
        NOP

        BTFSS   TXSTA,TRMT
        GOTO ASE013


        BSF     STATUS,RP0
        BCF     STATUS,RP1

        BSF     TXSTA,TXEN      ;ENABLE TX

        BCF     STATUS,RP0
        BCF     STATUS,RP1

        MOVLW   A'a'
        MOVWF   TXREG

        BSF     STATUS,RP1
        BCF     STATUS,RP1

        ASE014

        NOP
        NOP
        NOP
        NOP

        BTFSS   TXSTA,TRMT
        GOTO ASE014


        BSF     STATUS,RP0
        BCF     STATUS,RP1

        BSF     TXSTA,TXEN      ;ENABLE TX

        BCF     STATUS,RP0
        BCF     STATUS,RP1

        MOVLW   100
        MOVWF   TXREG

        BSF     STATUS,RP0
        BCF     STATUS,RP1

        ASE015

        NOP
        NOP
        NOP
        NOP

        BTFSS   TXSTA,TRMT
        GOTO    ASE015

        BCF     STATUS,RP0
        BCF     STATUS,RP1

        CALL DEMORA100


        BSF     STATUS,RP0
        BCF     STATUS,RP1

        BSF     TXSTA,TXEN      ;ENABLE TX

        BCF     STATUS,RP0
        BCF     STATUS,RP1

        MOVLW   A'>'
        MOVWF   TXREG

        BSF     STATUS,RP0
        BCF     STATUS,RP1

        ASE016

        NOP
        NOP
        NOP
        NOP

        BTFSS   TXSTA,TRMT
        GOTO ASE016

        BSF     STATUS,RP0
        BCF     STATUS,RP1

        BSF     TXSTA,TXEN      ;ENABLE TX

        BCF     STATUS,RP0
        BCF     STATUS,RP1

        MOVLW   A'1'
        MOVWF   TXREG

        BSF     STATUS,RP0
        BCF     STATUS,RP1

        ASE017

        NOP
        NOP
        NOP
        NOP

        BTFSS   TXSTA,TRMT
```

```
GOTO ASE017

BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   '1'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE018

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE018


BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   'a'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE019

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE019


BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   B'00000000'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE020

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO    ASE020

BCF     STATUS,RP0
BCF     STATUS,RP1

CALL DEMORA100
CALL DEMORA100
CALL DEMORA100
CALL DEMORA100

RETURN
;*****************************************************
;*****************************************************

MOVERSERVO2

BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   A'>'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE021

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE021


BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   A'1'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE022

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE022


BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   '2'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE023

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE023


BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   A'a'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE024

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE024


BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   B'11111111'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE025

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO    ASE025

BCF     STATUS,RP0
BCF     STATUS,RP1

CALL DEMORA1
CALL DEMORA1
CALL DEMORA1

RETURN
;*****************************************************
;*****************************************************

PARARSERVO2

CALL DEMORA10
CALL DEMORA10
CALL DEMORA10
CALL DEMORA10
CALL DEMORA10
```

```
CALL DEMORA10

BCF     STATUS,RP0
BCF     STATUS,RP1

BSF             PTOBTEMP,5

BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   A'>'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE026

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE026

BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   A'1'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE027

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE027


BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   '2'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE028

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE028



BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   A'a'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE029

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE029


BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   .85
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE030

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO    ASE030

BCF     STATUS,RP0
BCF     STATUS,RP1

CALL DEMORA100


BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   A'>'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE031

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE031

BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   A'1'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE032

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE032


BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   '2'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE033

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE833


BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   A'a'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE034

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE034
```

```
        BSF     STATUS,RP0
        BCF     STATUS,RP1

        BSF     TXSTA,TXEN      ;ENABLE TX

        BCF     STATUS,RP0
        BCF     STATUS,RP1

        MOVLW   B'00000000'
        MOVWF   TXREG

        BSF     STATUS,RP0
        BCF     STATUS,RP1

        ASE035

        NOP
        NOP
        NOP
        NOP

        BTFSS   TXSTA,TRMT
        GOTO    ASE035

        BCF     STATUS,RP0
        BCF     STATUS,RP1

        CALL DEMORA100
        CALL DEMORA100
        CALL DEMORA100
        CALL DEMORA100

        RETURN

;*****************************************************
;*****************************************************

        MOVERSERVO3

        BSF     STATUS,RP0
        BCF     STATUS,RP1

        BSF     TXSTA,TXEN      ;ENABLE TX

        BCF     STATUS,RP0
        BCF     STATUS,RP1

        MOVLW   A'>'
        MOVWF   TXREG

        BSF     STATUS,RP0
        BCF     STATUS,RP1

        ASE036

        NOP
        NOP
        NOP
        NOP

        BTFSS   TXSTA,TRMT
        GOTO ASE036

        BSF     STATUS,RP0
        BCF     STATUS,RP1

        BSF     TXSTA,TXEN      ;ENABLE TX

        BCF     STATUS,RP0
        BCF     STATUS,RP1

        MOVLW   A'1'
        MOVWF   TXREG

        BSF     STATUS,RP0
        BCF     STATUS,RP1

        ASE037

        NOP
        NOP
        NOP
        NOP

        BTFSS   TXSTA,TRMT
        GOTO ASE037

        BSF     STATUS,RP0
        BCF     STATUS,RP1

        BSF     TXSTA,TXEN      ;ENABLE TX

        BCF     STATUS,RP0
        BCF     STATUS,RP1

        MOVLW   '3'
        MOVWF   TXREG

        BSF     STATUS,RP0
        BCF     STATUS,RP1

        ASE030

        NOP
        NOP
        NOP
        NOP

        BTFSS   TXSTA,TRMT
        GOTO ASE038


        BSF     STATUS,RP0
        BCF     STATUS,RP1

        BSF     TXSTA,TXEN      ;ENABLE TX

        BCF     STATUS,RP0
        BCF     STATUS,RP1

        MOVLW   A'a'
        MOVWF   TXREG

        BSF     STATUS,RP0
        BCF     STATUS,RP1

        ASE039

        NOP
        NOP
        NOP
        NOP

        BTFSS   TXSTA,TRMT
        GOTO ASE039


        BSF     STATUS,RP0
        BCF     STATUS,RP1

        BSF     TXSTA,TXEN      ;ENABLE TX

        BCF     STATUS,RP0
        BCF     STATUS,RP1

        MOVLW   B'11111111'
        MOVWF   TXREG

        BSF     STATUS,RP0
        BCF     STATUS,RP1

        ASE040

        NOP
        NOP
        NOP
        NOP

        BTFSS   TXSTA,TRMT
        GOTO    ASE040

        BCF     STATUS,RP0
        BCF     STATUS,RP1

        CALL DEMORA1
        CALL DEMORA1
        CALL DEMORA1

RETURN

;*****************************************************
;*****************************************************

        PARARSERVO3

        CALL DEMORA10
        CALL DEMORA10
        CALL DEMORA10
        CALL DEMORA10
        CALL DEMORA10
        CALL DEMORA10

        BCF     STATUS,RP0
        BCF     STATUS,RP1

        BSF             PTOBTEMP,6

        BSF     STATUS,RP0
        BCF     STATUS,RP1

        BSF     TXSTA,TXEN      ;ENABLE TX

        BCF     STATUS,RP0
        BCF     STATUS,RP1

        MOVLW   A'>'
        MOVWF   TXREG

        BSF     STATUS,RP0
        BCF     STATUS,RP1

        ASE041

        NOP
        NOP
        NOP
        NOP

        BTFSS   TXSTA,TRMT
        GOTO ASE041

        BSF     STATUS,RP0
        BCF     STATUS,RP1

        BSF     TXSTA,TXEN      ;ENABLE TX

        BCF     STATUS,RP0
        BCF     STATUS,RP1

        MOVLW   A'1'
        MOVWF   TXREG

        BSF     STATUS,RP0
        BCF     STATUS,RP1

        ASE042

        NOP
        NOP
        NOP
        NOP
```

```
BTFSS   TXSTA,TRMT
GOTO ASE042

BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   '3'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE043

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE043


BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   A'a'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE044

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE044


BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   .112
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE045

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO    ASE045

BCF     STATUS,RP0
BCF     STATUS,RP1

CALL DEMORA100
CALL DEMORA10
BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   A'>'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE046

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE046

BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   A'1'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE047

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE047

BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   '3'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE048

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE048


BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   A'a'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE049

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE049


BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   B'00000000'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE050

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO    ASE050

BCF     STATUS,RP0
BCF     STATUS,RP1

CALL DEMORA100
CALL DEMORA100
CALL DEMORA100
CALL DEMORA100


RETURN

;*****************************************************************
;*****************************************************************

MOVERSERVO4

BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   A'>'
MOVWF   TXREG

BSF     STATUS,RP0
```

```
BCF     STATUS,RP1

ASE051

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE051

BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   A'1'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE052

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE052


BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   '4'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE053

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE053


BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   A'a'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE054

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE054


BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   B'11111111'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE055

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO    ASE055

BCF     STATUS,RP0
BCF     STATUS,RP1

CALL DEMORA1
CALL DEMORA1
CALL DEMORA1

RETURN

;*****************************************************************
;*****************************************************************

PARARSERVO4

CALL DEMORA10
CALL DEMORA10
CALL DEMORA10
CALL DEMORA10
CALL DEMORA10
CALL DEMORA10

BCF     STATUS,RP0
BCF     STATUS,RP1

BSF             PIOBTEMP,7

BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   A'>'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE056

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE056

BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   A'1'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE057

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE057


BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   '4'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE058

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE058


BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   A'a'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE059

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE059
```

```
BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   .110
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE060

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO    ASE060

BCF     STATUS,RP0
BCF     STATUS,RP1

CALL DEMORA100

BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   A'>'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE061

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE061

BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   A'1'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE062

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE062


BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   '4'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE063

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE063


BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   A'a'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE064

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO ASE064


BSF     STATUS,RP0
BCF     STATUS,RP1

BSF     TXSTA,TXEN      ;ENABLE TX

BCF     STATUS,RP0
BCF     STATUS,RP1

MOVLW   B'00000000'
MOVWF   TXREG

BSF     STATUS,RP0
BCF     STATUS,RP1

ASE065

NOP
NOP
NOP
NOP

BTFSS   TXSTA,TRMT
GOTO    ASE065

BCF     STATUS,RP0
BCF     STATUS,RP1

CALL DEMORA100
CALL DEMORA100
CALL DEMORA100
CALL DEMORA100


        RETURN

;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

END
```