FECHA	JUNIO DE 2011
NÚMERO RAE	
Autor/es:	HIDALGO SANCHEZ, Paola – SOLARTE TOVAR, María Fernanda
Título:	DISEÑO Y CONSTRUCCIÓN DE UN CONTROLADOR MIDI DE UN ARPA LASER QUE REPRODUZCA AUDIO Y VIDEO
PALABRAS CLAVES	Analógico *Code Warrior* Digital Edición de audio Fotorresistencia. *FreeScale(Tower)* Laser Patrón Polar Protocolo MIDI Técnicas de grabación Tiempo de reverberación RS232 Sampler *Visual C# Express 2008*
Descripción:	En el presente proyecto se desarrolló el diseño y construcción de un controlador MIDI con que reproduzca audio y video, la cual estaba constituida con un banco de sonidos de un arpa real y un banco de imágenes realizadas por las autoras, de igual forma se puede programar para ser útil en cualquier campo que implemente el protocolo MIDI, brindando así más opciones para controlar sonidos, luces entre otros.
FUENTES BIBLIOGRÁFICAS FUENTES BIBLIOGRÁFICAS	- HERMANN,M., NORTON,M. Basilik subscale laser experiments, 1992-1993. - PUIG, Sergi Jordà, Audio Digital y MIDI capitulo 7 y 8, Guías Monográficas Anaya Multimedia, Madrid 1997. - SANTAMARIA,Tomas Pollan, , Electrónica Digital . Universidad de Zaragoza. - ZICARELLI. David, Basic Javascript programming.pdf 2000-2006. - Institut de Recherche et Coördination Acoustique /Musique, Max/Msp Fundamentals.pdf, 2000-2006 JORDÁ PUIG, Sergi, Audio Digital y MIDI capitulo 7 y 8, guias

	monográficas Anaya Multimedia, Madrid 1997 -ProgramData/Processor%20Expert/CW08_PE3_07/DOCs/PEh4CA2.html -REJANO DE LA ROSA, Manuel. Ruido Industrial y Urbano, Pág. 63 http://laserharp.manuel-schulz.com/readarticle.php?article_id=3 http://arauca.net/p4.html http://www.telefonica.net/web2/blasinski/microfonos/modelos.htm
CONTENIDOS:	**OBJETIVOS** Objetivo General Diseñar y Construir un controlador MIDI con características de arpa utilizando laser para la reproducción de audio y video. Objetivos Específicos 1. Realizar el banco de sonidos reales del instrumento. 2. Realizar un banco de imágenes por medio de fotos y videos. 3. Diseñar y construir electrónicamente el Arpa laser. 4. Implementar electrónicamente la Pedalera como octavador. 5. Programar el SAMPLER para reproducir las muestras obtenidas anteriormente. 6. Ensamblar la parte electrónica con la estructura del arpa. 7. Probar el desempeño del arpa y su funcionamiento.
CONTENIDOS:	**MARCO TEÓRICO** Fundamento teórico del MIDI Funcionamiento para el protocolo MIDI Fundamentación de los controladores y varias unidades Tipos de Micrófonos. Técnicas de Grabación Teoría de los diferentes Elementos Electrónicos y Software **MARCO CONCEPTUAL** Que es MIDI?

.

	Definición de SAMPLER Definición de Muestrear Definición de Audio Definición Tiempo de Reverberación Definición Edición de Audio Láser Patrón Polar Fotorresistencia.
CONTENIDOS: CONTENIDOS:	**DESARROLLO INGENIERIL** * DISEÑO Y CONSTRUCCIÓN ELECTRÓNICA DEL ARPA LASER Diseño de los circuitos y pruebas del Controlador Circuito dirigido al laser Circuito dirigido a la fotorresistencia * IMPLEMENTACIÓN ELECTRÓNICA DE LA PEDALERA COMO OCTAVADOR Desarrollo de SAMPLER por medio de visual C# 2008 Express Edition. Desarrollo del SAMPLER por medio de MAX/MSP * BANCO DE SONIDOS REALES DEL INSTRUMENTO Selección y Tipos de Micrófonos. Técnica de grabación Corrección de errores del banco de sonidos * BANCO DE IMÁGENES POR MEDIO DE FOTOS Y VIDEOS Desarrollo de SAMPLER por medio de visual C# 2008 Express Edition Desarrollo del SAMPLER por medio de MAX/MSP. * PROGRAMACIÓN DEL SAMPLER PARA LA REPRODUCCIÓN DE LAS MUESTRAS Implementación del SAMPLER desarrollado para reproducir audio e imágenes, empleando el programa MAX/MSP Elementos para implementación del SAMPLER Implementación del SAMPLER desarrollado para reproducir audio y

.

	video, empleando el programa visual C# 2008 Express Edition * PROGRAMACIÓN DEL ARPA LASER uint16_t value1 = 0; uint8_t sucCuerda1_ON rs232_SendBlock Códigos de activación Y desactivación de cuerda Código de activación y desactivación de octavador Muestreo de cada cuerda Programación del arpa laser * ENSAMBLAJE DE LA PARTE ELECTRÓNICA EN LA ESTRUCTURA DEL ARPA Diámetro del laser con las perforaciones del arpa. Asignación y conexión de cada cable de datos del circuito de la fotorresistencia a un puerto análogo de la TOWER Conexión y ruteo del controlador MIDI del arpa laser con su respectivo SAMPLER Conexión y ruteo por el puerto Serial. Conexión y ruteo por el puerto MIDI * PROBAR EL DESEMPEÑO DEL ARPA Y SU FUNCIONAMIENTO Funcionamiento a través del puerto serial Funcionamiento a través del puerto MIDI
METODOLOGÍA	1. **ENFOQUE DE LA INVESTIGACIÓN** El enfoque de la investigación es de tipo empírico analítico ya que el objetivo de este proyecto se basa en el análisis detallado del diseño y la construcción de un producto utilizando los conocimientos de la electrónica, sistemas y sonido; desarrollando una herramienta nueva e innovadora. **2. LÍNEA INSTITUCIONAL DE USB / SUB- LÍNEA DE FACULTAD / CAMPO TEMÁTICO DEL PROGRAMA** **2.1 Línea Institucional** El trabajo se desarrolla en la de línea de *TECNOLOGÍAS ACTUALES Y*

.

METODOLOGIA	*SOCIEDAD,* ya que el proyecto presenta una nueva herramienta para la sociedad, que ayuden a mejorar las diferentes presentaciones en vivo, tanto en la interpretación cono en la audiencia **2.2 Sub- Línea de Facultad** Este proyecto se basa en el *PROCESAMIENTO DE SEÑALES* ya que integra varios métodos para la implementación e intervención de señales. **2.3 Campo Temático del Programa** Este trabajo se enfoca en el *DISEÑO DE SISTEMAS DE SONIDO* ya que fundamentalmente es la creación y construcción de un nuevo instrumento que emula sonidos reales y reproduce audio y video simultáneamente. **3. HIPÓTESIS**
METODOLOGIA	La reproducción de audio y video se ha apoderado de la gran mayoría de eventos a lo largo del mundo entero; lo cual hace que sea un requisito obligado el innovar y crear nuevas herramientas que faciliten la interpretación musical para los diferentes shows presentados ante la sociedad; lo cual ha requerido del aporte ingenieril importante para su progreso. Este proyecto pretende brindar nuevas alternativas en el campo de audio y video, mejorando los espectáculos teniendo en cuenta la innovación no solo en el ámbito musical, sino también en el visual. 4 **VARIABLES** **4.1 Variables Independientes** Efectos sonoros (En la captura banco de sonidos). Efectos en el banco de imágenes. **4.2 Variables Dependientes** Dispositivos electrónicos utilizados en el proceso como los son: ➢ Láser ➢ Fotorresistencias

.

	➢ Resistencias ➢ Transductores Sampler
CONCLUSIONES	El Controlador MIDI de un ARPA LASER es un dispositivo electrónico capaz de reproducir con cada nota musical imágenes y videos aleatorios o programados y los botones de reproducción de los mismos. De igual forma se puede programar para ser útil en cualquier campo que implemente el protocolo MIDI, brindando así más opciones para controlar sonidos, luces entre otros. La utilización de las fotorresistencias para detectar el haz de los laser los cuales funcionan como las cuerdas del arpa laser, requiere de un previo análisis ya que son muy sensibles a la luz; esto hace que se vea afectado por las condiciones de iluminación en los recintos. Por lo cual se podría probar otros elementos que cumplan con la misma función como sensores y fotodiodos El controlador MIDI del arpa laser no solo permite reproducir muestras del arpa, si no también muestras de cualquier instrumento que se tengan pregrabadas. La implementación del protocolo MIDI y el puerto serial RS232 en el dispositivo permite que este sea compatible con la mayoría o su totalidad de software y samplers que existen en el mercado y proporciona una herramienta de control enfocada a cualquier instrumento musical. El diseño del controlador MIDI del arpa laser en ningún momento pretende remplazar el instrumento real, más bien, el prototipo fue diseñado para brindar nuevos espacios para la construcción de proyectos sobre el tema de controladores digitales capaces de emular instrumentos musicales reales que sean llamativos, novedosos y a su vez didácticos. El costo del KIT del microcontrolador utilizado en el desarrollo del prototipo es económico teniendo en cuenta que brinda una variedad de opciones al ser reconfigurable y que se puede programar más de un dispositivo.

PAOLA HIDALGO SÁNCHEZ
MARÍA FERNANDA SOLARTE

UNIVERSIDAD DE SAN BUENAVENTURA
INGENIERIA DE SONIDO
BOGOTA
2011

DISEÑO Y CONSTRUCCIÓN DE UN CONTROLADOR MIDI DE UN ARPA LASER QUE REPRODUZCA AUDIO Y VIDEO

.

PAOLA HIDALGO SÁNCHEZ
MARÍA FERNANDA SOLARTE

UNIVERSIDAD DE SAN BUENAVENTURA
INGENIERIA DE SONIDO
BOGOTA
2011

Nota de aceptación:

.

_____
_____
_____
_____
_____

_____
Firma de Jurado

_____
Firma de Jurado

Bogotá, 2 de Junio de 2011

TABLA DE CONTENIDO

.

INTRODUCCIÓN

La música años atrás era utilizada por el hombre como medio para comunicarse y expresar sus sentimientos, a través de sonidos emitidos por elementos básicos y su propia voz; dependiendo hacia quien eran emitidos los sonidos variaban las intensidades de los golpes o de la voz, diferenciaba si eran hacia animales o hacia otros hombres.

A partir de las diversas experiencias algunas civilizaciones conceptualizaron la música, dependiendo del momento histórico que vivía cada uno de los países, los chinos por ejemplo formaron la “escala musical” la cual los europeos llamaron "cromática", planteada por el maestro de música de Hoan-Ti (antiguo emperador)

La música egipcia es un misterio, se admite influencia de la cultura musical griega y la existencia de indicios sobre instrumentos conservados, como también el hallazgo de bajorrelieves en templos religiosos, hacen deducir que los egipcios poseían instrumentos de cuerda, viento y percusión, ya que se encontraron en algunas tumbas faraónicas figuras de flautas, etc.

La música era interpretada desde una doble posibilidad; por un lado la capacidad de motivar en el hombre una sensación (de gozo, alegría) y por el otro, al de crear sensaciones de naturaleza mística y mágica.

La Iglesia Católica ejerció un impacto alto en el avance musical de la humanidad, ya que al ser un movimiento cultural de tan alto impacto, impuso 4 dialectos musicales el Milanés, el Galiciano, el Mozárabe y el Romano, el que más se dio a conocer por su fuerza cultural fue el Romano, si bien el Canto Ambrosiano ejerció una poderosa influencia A fines del Siglo IV, el Canto Gregoriano se difundió a toda la Cristiandad

.

alrededor de dos siglos después ;y marco un trascendente camino en el desarrollo de la humanidad.

La melodía del canto gregoriano asimila 3 estilos diferentes; el "Silábico" (cada nota representada por una sílaba), el "Neumasico" (una misma sílaba corresponden 2, 3 ó 4 sonidos diferentes), y las "Secuencias" (intercalación de un texto en las notas del aleluya).

El desarrollo de las nuevas tecnologías se ha visto atraída de un tiempo hacia acá por el diseño y construcción de innovadoras herramientas para la interpretación musical. En la actualidad la música y sus presentaciones en vivo han adquirido importancia, lo cual ha llevado al hombre a experimentar con interdisciplinariedad entre varias especialidades y brindar amplias opciones para el desarrollo de estas.

La interpretación musical digital adquirió una evolución llevando a lo convencional a otro plano. En Colombia el diseño de estas nuevas herramientas no tiene mucho registro por falta de conocimiento. Pero el auge que ha tenido la música con el pasar del tiempo ha llevado al país y al mundo entero a ampliar las técnicas de creación de instrumentos que le faciliten la ejecución al músico y mejore la parte visual y auditiva al espectador.

Por ello surge la idea de realizar un proyecto que se basa en diseñar y construir un controlador MIDI de un arpa laser para la reproducción simultanea de audio y video, en el cual a partir de la electrónica, sistemas y el sonido implementen y faciliten la interpretación de cualquier instrumento.

.

1. PLANTEAMIENTO DEL PROBLEMA

1.1 ANTECEDENTES

En épocas anteriores la construcción de instrumentos musicales no era catalogado como una profesión, esta actividad se la conocía como violeros o guitarreros, luego se le designó el nombre de luteros asociado al instrumento luth o Laúd siendo estos instrumentos los más populares.

Michael Praetorius (1571-1621) hace referencia que en el siglo XVII es el inicio de la fabricación de instrumentos; creándose el monopolio[1] para la construcción de instrumentos de cuerda frotada y pinzada; el trabajo de construcción consistía en tres actividades importantes para su calidad como son; la construcción, la afinación del instrumento y la restauración y dictamen.

La fabricación de los instrumentos de cuerda depende de: el tipo de instrumento, de las distintas categorías de éste y de la utilización de diferentes clases de materiales para su elaboración ya sean sintéticos, de acero, de tripas de animal o de materias vegetales, siendo determinante para el sonido la clase de material que se utilice en su construcción.

Los instrumentos de cuerda son categorizados como cordófonos, caracterizados por poseer una caja de resonancia o armónica, que permiten que el sonido se amplifique en su volumen y duración; teniendo claro este concepto se determina el tipo de instrumento que se necesite construir, por cuanto mayor sea su volumen mayor es la prolongación del sonido.

El arpa pertenece a la anterior categoría, en particular éste es uno de los instrumentos musicales más antiguos, encontrándose relatos históricos hasta en apartados bíblicos,

1 PRAETORIUS, Michael. Historia de los instrumentos.

.

se lo ha construido de diferentes formas y en diferentes partes de mundo, como en Israel, Egipto luego en Grecia, ya en el siglo XVIII se retoma pero con la incorporación de pedales.

Este instrumento de cuerda pulsada está compuesto por un marco resonante y una serie de cuerdas tensionadas por sus extremos. Las cuerdas pueden ser pulsadas con los dedos o con una púa o plectro. Existen diversos tipos de arpas como son: el arpa clásica, el arpa celta y el arpa paraguaya. Ver figura 1,2y 3.

Figura1. Arpa celta Figura2.Arpa Paraguaya Figura3.Arpa Clásica

La diferencia entre estos tipos de arpa, esta dado por el número de cuerdas que la compone, el material de las cuerdas, tamaño del instrumento, su afinación y sus diferentes usos, dependiendo del género musical .

Pasando a otro tema importante para esta investigación es el Sampler, que en inglés “sample” significa muestra, su origen data del 1950, en esta época eran llamados fonógenos, compuesto por una cinta magnética circular montada sobre un tambor de cabezas y cuya velocidad de reproducción era conectado a un circuito generando tonos sobre un sonido ya grabado. Ya en los años 60 se crea el melotrón, un precursor análogo a los modernos samplers, éste controlaba un sistema de cinta a través de su teclado el cual no era cerrado y poseía varias pistas a diferencia de los fonógenos. El

.

melotrón fue el primer instrumento de esta clase en usarse en música pop The Beatles and The Rolling Stones.

A finales de los años 70 con el avance de la electrónica digital aparece el primer sampler digital, y su difusión se hace a principios de los años 80 siendo Depeche Mode y Kraftwerk algunos abanderados.

Por lo tanto, un *sampler* es un instrumento que permite muestrear (grabar) digitalmente secuencias sonoras, para ser reproducidas posteriormente tal cual fueron grabadas, o transformadas mediante efectos. También permite recuperar estas secuencias en un soporte de almacenamiento secundario como: discos duros, unidades ZIP, disquetes, entre otros. Estas muestras de sonido son luego reproducidas como sonido asignado a una o varias notas de un teclado musical, o a un pad formado por varios botones asignables o disparadas desde un secuenciador. Teniendo en cuenta lo anterior el *sampler* es utilizado como herramienta en muchos géneros musicales, entre los que destaca la música electrónica.[2]

Haciendo un revisión histórica de temas relacionados con el diseño y construcción de un controlador de un arpa laser que reproduce audio y video, a nivel internacional se encontró un estudio de un Arpa Láser, en la Universidad Pontificia del Perú en el año de 2002, encontrándose un video dando a conocer su funcionalidad. Sin embargo no se encuentra un documento científico que presente un fundamento teórico de su construcción.

A nivel nacional en la universidad de San Buenaventura de Bogotá se han realizado varios trabajos de grado basados en protocolo MIDI; un estudio cercano al presente proyecto es el elaborado por Moreno Ricardo en el año de 2007, relacionado con el diseño y construcción de una flauta controladora MIDI, la cual emula las características de una flauta y que es aplicable a un hardware y software a través del protocolo MIDI

2 M.Hermann,M.Norton, BASILIK SUBSCALELASER EXPERIMENTS,1992-1993

.

generado por un algoritmo. Este trabajo muestra la similitud del instrumento musical real y el prototipo creado.
Las similitudes entre estos dos proyectos es que son instrumentos capaces de emular algunos parámetros del instrumento real, implementan protocolo MIDI y son una herramienta de control a diferentes aplicaciones. Y sus diferencias es que la flauta controladora MIDI es la variedad de conexiones que no solo se centra por puerto MIDI si no también por un puerto serial (RS232), además de su reproducción simultanea de imágenes y video por medio de software y la manera de ejecutar el prototipo en este caso por la interrupción de laser.

Otro proyecto es el Diseño y construcción de un dispositivo USB - HID que controla parámetros de síntesis de audio en tiempo real, mediante el programa MAX/MSP. Realizado por Valenzuela, Gómez, David y Larrota en el año 2007. Este proyecto se centra en programar un software permitiendo controlar los parámetros de un algoritmo o de un patch. En particular se trabajó en la misma plataforma MAX/MSP, un software encargado para controlar algunos parámetros de síntesis de audio en tiempo real al igual que el arpa laser, pero con la diferencia que el controlador MIDI de un arpa laser controla parámetros de video simultáneamente.

En el 2007, Escobar y Gracia, construyen un Theremin y diseñan la implementación de salida MIDI para control de altura "Hybrid" generando una señal de audio análoga, proveniente del circuito que hace parte el instrumento y una señal digital dentro del protocolo MIDI, que poseen sincronismo en el tiempo de ejecución a través de mensajes MIDI para controlar su altura tonal. Este proyecto no tiene mayor semejanza con el presente, excepto por la utilización del protocolo MIDI en el envió de mensajes provenientes del circuito análogo por parte del instrumento.
En el 2007 Ripe, Andrés, crea un Dispositivo conversor de nota musical a nota MIDI para bajo eléctrico y su programación en un procesador de señal digital para ser reproducido por un sintetizador por medio de un software o un hardware. Este se

.

asemeja con el presente proyecto en la reproducción por medio de un software y hardware y su diferencia es el instrumento y puede reproducirse con la mayoría de software.

1.2 DESCRIPCIÓN Y FORMULACIÓN DEL PROBLEMA

En los últimos años la reproducción de audio y video se desarrolla con mayor auge en Europa y Asia, esto los convierte en pioneros en este campo; sin embargo, Colombia a pesar de caracterizarse de ser una cultura con muchas raíces musicales, cuenta con muy poca experiencia en el campo de diseño y construcción; por lo tanto, con el avance de la tecnología sin perder el legado cultural, es necesario crear nuevas herramientas que permitan el reconocimiento de la riqueza de instrumentos del contexto cultural en que vivimos y que la interpretación de estos instrumentos reales sean prácticos en su utilización, digitales y novedosos.

Con lo anterior, este trabajo parte del siguiente interrogante: ¿Es posible diseñar y construir un controlador MIDI de un arpa laser, que emule algunos parámetros del instrumento real y simultáneamente reproduzca video?

1.3 JUSTIFICACIÓN

.

Teniendo en cuenta lo anterior, se puede decir que los avances tecnológicos a nivel mundial, han tomado mucha fuerza en el campo musical, lo cual ha llevado a crear diferentes formas de expresión artística por medio de la electrónica, integrando varias ramas ingenieriles que generan nuevas e innovadoras alternativas que brindan al público experiencias no solo auditivas sino visuales.

Por lo tanto, el diseño de un controlador MIDI de un arpa laser propone una nueva forma de integrar los saberes del profesional en sonido, ya que permite demostrar que los conocimientos adquiridos en MIDI, programación y electrónica, son aprendizajes que generan competencias de aplicabilidad en diseño y construir de elementos para apreciar la música sin perder su estructura musical.

Este proyecto se posesiona como algo novedoso, ya que Colombia cada día exige de escenarios artísticos con alta calidad en su producción, siendo la tecnología la que posibilite nuevas alternativas de ejecutar instrumentos de manera práctica, llamativa.

Otro beneficio es que este proyecto, genera un impacto social y cultural, ya que la tecnología no pretende remplazar el instrumento real acústico, por el contrario, el objetivo fundamental con su divulgación, es crear familiaridad de la riqueza cultural del contexto en que se vive rescatando el folclor de una región.

Así mismo se convierte en algo práctico ya que las personas de diferentes edades pueden aprender de manera fácil, convirtiéndose en un medio didáctico para la enseñanza. Además es llamativo, ya mantiene el foco de atención facilitando el aprendizaje musical por medio de imágenes y sonidos.

Por último, este proyecto es una forma de proyectar al profesional de Ingeniero de Sonido ya presenta una de las alternativas de su quehacer profesional en el medio laboral y social con una perspectiva ética.

.

1.4 OBJETIVOS DE LA INVESTIGACIÓN

1.4.1 Objetivo General

.

Diseñar y Construir un controlador MIDI con características de arpa utilizando laser para la reproducción de audio y video.

1.4.2 Objetivos Específicos

8. Realizar el banco de sonidos reales del instrumento.
9. Realizar un banco de imágenes por medio de fotos y videos.
10. Diseñar y construir electrónicamente el Arpa laser.
11. Implementar electrónicamente la Pedalera como octavador.
12. Programar el SAMPLER para reproducir las muestras obtenidas anteriormente.
13. Ensamblar la parte electrónica con la estructura del arpa.
14. Probar el desempeño del arpa y su funcionamiento.

1.5 ALCANCES Y LIMITACIONES DEL PROYECTO

1.5.1 Alcances

.

- Facilitar la interpretación de un instrumento musical tanto para músicos como para personas que carecen de conocimientos sobre instrumentos.
- El documento abre puertas a nuevos instrumentos, ya que este puede ser un punto de partida importante para la creación de herramientas tanto visuales como sonoras.

1.5.2 Limitaciones

- Disponibilidad de los dispositivos electrónicos en el mercado local.
  - Láser
  - Fotorresistencias
  - Resistencias
  - Transductores
- El instrumento no permite implementar control de velocity al ejecutarlo.

2. MARCO DE REFERENCIA

2.1 MARCO TEÓRICO

.

2.1.1 Fundamento teórico del MIDI. El MIDI[3] son las siglas en ingles de una Interfaz Digital de los Instrumentos Musicales. Fue inicialmente propuesto en un documento dirigido a la Audio Engineering Society por Dave Smith, presidente de la compañía Sequential Circuits en 1981. La primera especificación MIDI se publicó en agosto de 1983.

Y se trata de un protocolo industrial estándar que permite que los computadores, sintetizadores, secuenciadores, controladores y otros dispositivos musicales electrónicos sepan comunicar y compartir información para la generación de sonidos.

Esta información define diversos tipos de datos como números que pueden corresponder a notas particulares, números de patches de sintetizadores o valores de controladores. Gracias a esta simplicidad, los datos pueden ser interpretados de diversas maneras y utilizados con fines diferentes a la música.

El protocolo MIDI incluye especificaciones complementarias de hardware y software, que permite entre otros, reproducir y componer música. Este se caracteriza por la ligereza de los archivos, pudiendo almacenar multitud de melodías complejas, como las de música clásica tocadas con varios instrumentos y en muy poca memoria.

Cabe aclarar que MIDI no transmite señales de audio, sino datos de eventos y mensajes controladores que se pueden interpretar de manera arbitraria, de acuerdo con la programación del dispositivo que los recibe. Es decir, MIDI es una especie de "partitura" que contiene las instrucciones en valores numéricos (0-127) sobre cuándo

3 Sergi Jordà Puig, Audio Digital y MIDI capitulo 7 y 8, Guías Monográficas Anaya Multimedia, Madrid 1997.

.

generar cada nota de sonido y las características que debe tener; el aparato al que se envíe dicha partitura la transformará en música completamente audible.

En la actualidad la gran mayoría de los creadores musicales utilizan el lenguaje MIDI a fin de llevar a cabo la edición de partituras y la instrumentación previa a la grabación con instrumentos reales. Sin embargo, la perfección adquirida por los sintetizadores en la actualidad lleva a la utilización de forma directa en las grabaciones de los sonidos resultantes del envío de la partitura electrónica a dichos sintetizadores de última generación.

2.1.2 Funcionamiento para el protocolo MIDI[4]

2.1.2.1 Hardware. Buena parte de los dispositivos MIDI son capaces de enviar y recibir información, pero desempeñan un papel diferente dependiendo de si están recibiendo o enviando información, también depende de la configuración del programa o programas que puede usar dicho dispositivo. El que envía los mensajes de activación se denomina Maestro (del inglés máster, o 'amo') y el que responde a esa información Esclavo (slave).

2.1.2.2 Dispositivos. Los dispositivos MIDI se pueden clasificar en tres grandes categorías:

• Controladores: Generan los mensajes MIDI (activación o desactivación de una nota, variaciones de tono, etc.). El controlador más familiar a los músicos tiene forma de teclado de piano, al ser este instrumento el más utilizado a la hora de componer e interpretar las obras orquestales; sin embargo, hoy día se han construido todo tipo de instrumentos con capacidad de transmisión vía interfaz MIDI: guitarras, parches de percusión, clarinetes electrónicos, incluso gaitas MIDI.

4 Ibid. Capítulo 7 y8.

• Unidades generadoras de sonido: También conocidas como módulos de sonido, reciben los mensajes MIDI y los transforman en señales sonoras (recordemos que MIDI no transmite audio, sino paquetes de órdenes en formato numérico).
• Secuenciadores: No son más que aparatos destinados a grabar, reproducir o editar mensajes MIDI. Pueden desarrollarse bien en formato de hardware, bien como software de computadora, o bien incorporados en un sintetizador.

Éstos son los tres grandes tipos de aparatos MIDI. Aun así, podemos encontrar en el mercado aparatos que reúnen dos o tres de las funciones descritas. Por ejemplo, los órganos electrónicos disponen de un controlador (el propio teclado) y una unidad generadora de sonido; algunos modelos también incluyen un secuenciador.

2.1.2.3. Cables y conectores. Un cable MIDI utiliza un conector del tipo DIN de 5 pines o contactos. La transmisión de datos sólo usa uno de éstos, el número 5. Los números 1 y 3 se reservaron para añadir funciones en un futuro. Los restantes (2 y 4) se utilizan -respectivamente- como blindaje y para transmitir una tensión de +5 voltios, para asegurarse que la electricidad fluya en la dirección deseada. La finalidad del cable MIDI es la de permitir la transmisión de los datos entre dos dispositivos o instrumentos electrónicos. En la actualidad, existen fabricantes de equipos económicos y por ello, muy populares, tales como Casio, Korg y Roland han previsto la sustitución de los cables y conectores MIDI estándar, por los del tipo USB que son más fáciles de hallar en el comercio y que permiten una fácil conexión a las computadoras personales.

2.1.2.4. Conexiones. El sistema de funcionamiento MIDI es de tipo simplex, es decir, sólo puede transmitir señales en un sentido. La dirección que toman las señales es siempre desde un dispositivo 'maestro' hacia un dispositivo 'esclavo'. El primero genera la información y el segundo la recibe. Para entender bien el sistema de conexión, debemos saber que en un aparato MIDI puede haber hasta tres conectores:

.

• MIDI OUT: Conector del cual salen los mensajes generados por el dispositivo maestro.
• MIDI IN: Sirve para introducir mensajes al dispositivo esclavo.
• MIDI THRU: También es un conector de salida, pero en este caso se envía una copia exacta de los mensajes que entran por MIDI IN.
El formato más simple de conexión es el formado por un dispositivo maestro (por ejemplo, un controlador) y un esclavo (como un sintetizador). En este caso, el maestro dispondrá de un conector MIDI OUT, de donde saldrán los mensajes MIDI generados, el cual deberemos unir al conector MIDI IN en el esclavo.

MIDI admite la conexión de un solo maestro a varios dispositivos esclavos en cascada. Para esos casos se utilizará MIDI THRU, uniendo el maestro con una de las unidades del modo descrito anteriormente. En el conector MIDI THRU de esa unidad se obtiene una copia de los mensajes MIDI que se introducen a través de MIDI IN, por lo que ese MIDI THRU se conectará con MIDI IN de otra de las unidades, a ésto se le llama Daisy Chain.
Supongamos que uno de los esclavos también incluye un controlador (como un sintetizador con teclado). Éste dispondrá de conector MIDI OUT. En ese caso, obtendremos los mensajes generados desde controlador en MIDI OUT, mientras que los mensajes correspondientes al controlador situado al inicio de la cadena aparecerán en MIDI THRU.

Por último, si se dispone de un aparato secuenciador (capaz de almacenar y reproducir información MIDI recibida), se conectará entre el controlador y la primera unidad generadora de sonido. En ese caso, el secuenciador dispondrá de conectores MIDI OUT y MIDI IN.

Aunque existe la posibilidad de la conexión en cascada de varios aparatos MIDI, es cierto que existe una limitación. Las características eléctricas de los conectores MIDI

.

hacen la señal proclive a la degradación, por lo que son pocos los aparatos que se pueden conectar en cascada antes de notar pérdidas apreciables de información.

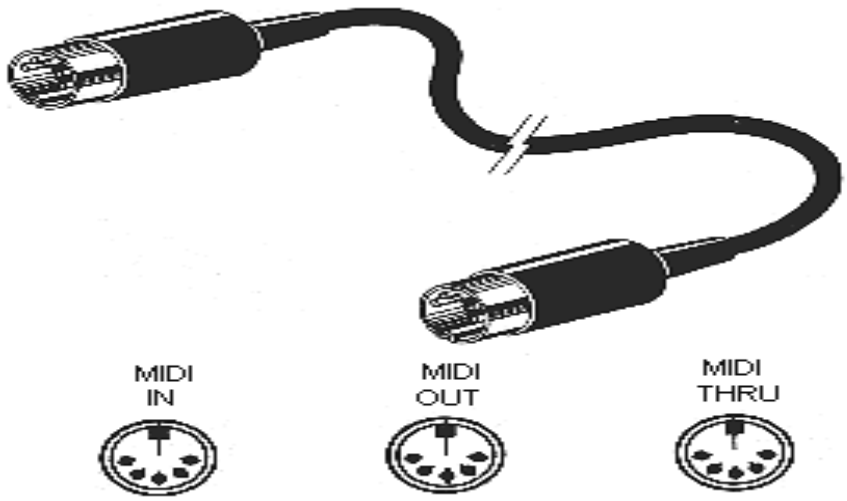


Figura 4. Cable y puertos MIDI

2.1.2.5. Software. La especificación MIDI incluye un aspecto de software que parte de la misma organización de los bytes.

2.1.2.6. Bytes MIDI. El byte MIDI, a diferencia de los bytes estándar de 8 bits de las computadoras, está compuesto por diez bits que se envían/reciben a una velocidad de 31250 bits/segundo con una tolerancia de +/- 1% según el estándar. El primero es el bit de inicio (start bit, que siempre es 0) y el último el bit de terminación (stop bit que siempre es 1). Esto con el fin de que los dispositivos MIDI puedan llevar la cuenta de cuantos bytes se han enviado o recibido. Los ocho bits restantes contienen los mensajes MIDI.

Existen dos tipos de bytes: De estado -status byte- y de información -data byte-. Se diferencian por el primer bit: si es un 1, tenemos un byte de estado, y si es un 0, es un

.

byte de datos. Al generar un mensaje MIDI, por norma general, siempre enviamos un byte de estado, que puede estar seguido de cierta cantidad de bytes de datos.

En la siguiente tabla tenemos una lista con todos los mensajes disponibles.

Byte estado	Descripción
1000cccc	Desactivación de nota
1001cccc	Activación de nota
1010cccc	Pos pulsación polifónica
1011cccc	Cambio de control
1100cccc	Cambio de programa
1101cccc	Post-pulsación monofónica de canal
1110cccc	Pitch
11110000	Mensaje exclusivo del fabricante
11110001	Mensaje de trama temporal
11110010	Puntero posición de canción
11110011	Selección de canción
11110100	Indefinido
11110101	Indefinido
11110110	Requerimiento de entonación
11110111	Fin de mensaje exclusivo
11111000	Reloj de temporización
11111001	Indefinido
11111010	Inicio
11111011	Continuación
11111100	Parada
11111101	Indefinido
11111110	Espera activa
11111111	Reseteo del sistema

Tabla 1. Mensajes MIDI

Los primeros bytes, cuyos últimos cuatro bits están marcados como "cccc", se refieren a mensajes de canal; el resto de bytes son mensajes de sistema.

.

Antes de explicar más detalladamente las características de algunos de los mensajes, conviene conocer dos importantes características de MIDI: los canales y los modos.

2.1.2.*7. Canales MIDI.* Como se comentó con anterioridad, MIDI está pensado para comunicar un único controlador con varias unidades generadoras de sonido (cada una de las cuales puede tener uno o varios instrumentos sintetizados que deseemos utilizar), todo por un mismo medio de transmisión. Es decir, todos los aparatos conectados a la cadena MIDI reciben todos los mensajes generados desde el controlador. Ello hace necesario un método para diferenciar cada uno de los instrumentos. Este método es el denominado canal.
MIDI puede direccionar hasta 16 canales (también llamados voces, o instrumentos); por ello, al instalar el sistema MIDI será necesario asignar un número de canal para cada dispositivo.
Instrumentos MIDI
Estos son los 128 instrumentos de la especificación estándar de MIDI, también conocidos como GM o "General Midi"

2.1.2.8. Modos MIDI. Dentro del sistema MIDI, se decidió crear una serie de diferentes modos de funcionamiento, cada uno con ciertas características. Antes de verlo, debemos diferenciar entre los siguientes conceptos:

• Monofónico: un instrumento monofónico sólo puede reproducir una nota simultáneamente. Es decir, para reproducir una nueva nota debe primero dejar de sonar la anterior. Por ejemplo, los instrumentos de viento son monofónicos, ya que sólo reproducen un único sonido cada vez.
• Polifónico: un instrumento polifónico puede reproducir varias notas simultáneamente. Un ejemplo es un piano, que puede formar acordes por medio de hacer sonar dos o más notas a la vez.

.

Se puede resumir los modos MIDI en la siguiente tabla:

Número	Nombre	Descripción
1	Omni on / poly	Funcionamiento polifónico sin información de canal
2	Omni on / mono	Funcionamiento monofónico sin información de canal
3	Omni off / poly	Funcionamiento polifónico con múltiples canales
4	Omni off / mono	Funcionamiento monofónico con múltiples canales

Tabla 2. Modos de funcionamiento MIDI

Los dos primeros modos se denominan "Omni on". Esto se debe a que en esos modos la información de canal está desactivada. Esas configuraciones se reservan para configuraciones donde sólo utilicemos un instrumento. Los otros dos modos, "Omni off", sí admiten la información de canal.

Mensajes de canal

Existen tipos de mensajes channel: - Note on - Note off - Pitch-Bend - Program change - Control change

Controlador y unidad generadora de sonido

Tanto en el sentido de generar el/los sonido/s se auto-complementa en el sentido de grabación - difusión - al mismo tiempo con consolas preparadas y dispuestas para dicho sistema. Ejemplo: Sea una o varias voces humanas o generada por instrumental se compaginan cambiando información ó datos, tarea que es realizada en el sistema Midi.

2.1.3 Fundamentación de los controladores y varias unidades

.

2.1.3.1. Secuenciador. Un secuenciador es un dispositivo que permite realizar grabaciones de datos MIDI paso a paso donde quedan almacenados la altura MIDI (0-127) duración la nota, la velocidad (análoga a la intensidad con valores de 0 a 127) el tipo de instrumentos (patch) y efectos. Todo esto se combina para formar el corpus de datos a emitir. Estos datos pueden ser utilizados para piezas de música, así como para el control de consolas de luces, consolas de audio o cualquier equipamiento que interprete el protocolo MIDI y pueda usar éste para fines particulares.

MIDI son las siglas del (Interfaz Digital de Instrumentos Musicales). Se trata de un protocolo industrial estándar que permite a los computadores, sintetizadores, secuenciadores, controladores y otros dispositivos musicales electrónicos comunicarse y compartir información para la generación de sonidos.

Generalidades

Esta información define diversos tipos de datos como números que pueden corresponder a notas particulares, números de patches de sintetizadores o valores de controladores. Gracias a esta simplicidad, los datos pueden ser interpretados de diversas maneras y utilizados con fines diferentes a la música. El protocolo incluye especificaciones complementarias de hardware y software.

Permite por ejemplo reproducir y componer música en este formato. Se caracteriza por la ligereza de los archivos, pudiendo almacenarse multitud de melodías complejas, como las de música clásica tocadas con varios instrumentos, en muy poca memoria.

2.1.3.2. Sampler. El sampler es una interfaz gráfica que permite muestrear digitalmente secuencias sonoras o samples para ser reproducidas posteriormente, o transformadas mediante efectos. También permite recuperar y almacenar, este es utilizado como herramienta en muchos géneros musicales, entre los que destaca la música electrónica.

.

I.

II. ***Características***

Suelen contar con conectores MIDI para ser manipulados desde otros dispositivos electrónicos, como ordenadores o secuenciadores. Permiten transformar las muestras con efectos (eco, reverberación, flanger FGD y otros), cambiar el tono, el volumen, estructura, color la intensidad, etc., y posteriormente almacenar estas muestras en disquetes, discos duros, u otros dispositivos similares.

- *Origen y evolución*

El origen del sampler se remonta a los años 1950, a partir de los denominados dispositivos fonógenos,[1] piezas casi exclusivamente de laboratorio consistentes en una cinta magnética circular montada sobre un tambor de cabezales y cuya velocidad de reproducción era controlada por un circuito conectado a un teclado semejante al de un pequeño órgano, permitiendo generar todos los tonos sobre cualquier sonido previamente grabado. Karlheinz Stockhausen, Pierre Boulez, Pierre Schaeffer o Iannis Xenakis fueron algunos de los músicos y compositores que comenzaron a explorar las posibilidades de este tipo de instrumentos en campos experimentales de la época como la música estocástica, la música concreta y la tape music.

En los años 60 aparece el órgano melotrón, considerado el verdadero precursor analógico de los modernos samplers. De forma análoga a los viejos fonógenos, el melotrón controlaba un sistema de cinta a través de su teclado. Una de las diferencias es que el sistema de cinta no era cerrado y además poseía varias pistas. La cinta, de pocos segundos de duración, era rebobinada automáticamente al llegar al final. Una de las principales desventajas (la misma que los fonógenos) es que el tiempo de ataque era lento, es decir, el motor que controlaba la cinta requería cierto tiempo para pasar de tonos muy graves a muy agudos o viceversa, en función del intervalo que se ejecutara. Otro de los motivos por el que es notable el melotrón es que fue el primer instrumentó de esta clase en usarse en música pop (Ej. The Beatles).

.

A finales de los años 1970, y gracias al avance de la electrónica, aparece el Fairlight CMI, el primer sampler digital. Podemos destacar a Trevor Horn, productor de The Buggles y Frankie Goes To Hollywood entre otros, como uno de los pioneros en su uso.
Finalmente la propagación y difusión del uso del sampler se hace realmente patente cuando la compañía E-Mu saca al mercado su famosa serie Emulator a principios de los años 1980, siendo Depeche Mode y Kraftwerk algunos de sus grandes abanderados.

2.1.4 Tipos de Micrófonos[5]:

Los micrófonos son unos transductores encargados de transformar la energía acústica en energía eléctrica, permitiendo así el registro, almacenamiento, procesamiento y transmisión de las señales de audio.
Los micrófonos se pueden clasificar dependiendo de la forma en cómo se transforma la señal acústica en eléctrica.

2.1.5 Técnicas de Grabación[6]

Técnicas de Microfónia Estéreo

5 Martínez, Javier, LA MUSICA Y EL ARTE DE LA GRABACIÖN,ED.Disonex,

6 Martinez, Javier,LA MUSICA Y EL ARTE DE LA GRABACIÖN, Ed.PROSONEX

.

Las principales características que se deben tener en cuenta para la grabación estéreo son:

Profundidad de la instrumentación
Perspectiva de campo del oyente
Entorno acústico
Preserva mejor el timbre global de la composición y su equilibrio dinámico.

Las principales técnicas utilizadas en grabación estéreo son:

- Pares coincidentes
- Pares espaciados
- Pares casi-coincidentes
- Cabeza artificial.

2.1.5.1. Pares coincidentes [7]

Las principales particularidades de esta técnica son:

· Micrófonos direccionales angulados, Diafragmas superpuestos.
· Codifica el ST por diferencias de nivel entre canales.
· Figura sonora escarpada.
· ST estrecho.
· El ángulo correcto depende del diagrama polar.
· Micrófonos cardiodes: Amplitud más estrecha del ST.
· Método Blumlein: Micrófonos bidireccionales a 90º.
· Método MS (Mid-Side=medio lado)

7 Ibid.

.

Figura 13.- Técnica de Pares Coincidentes

2.1.5.2. Par espaciado. Las principales particularidades de esta técnica son:

· Dos micrófonos iguales separados. Apuntando frontalmente.
· Los micrófonos deben ser omnidireccionales.
· A mayor espacio entre micrófonos mayor ST.

Figura 14.- Técnica de Par espaciado

2.1.5.3. ORTF. Los principales rasgos de esta técnica son:
· Micrófonos direccionales angulados. A mayor ángulo mejor ST.
· Combina la diferencia de niveles entre canales y de tiempos. Efecto ST total.
· Cardiodes espaciados 17 cm y angulados a 110º.
· Localización precisa. Mayor sensación de volumen y profundidad.

.

Figura 15. Técnica de ORTF

2.1.6 Elementos Electrónicos

2.1.6.1. Fotorresistencia. Una fotorresistencia es un componente electrónico cuya resistencia disminuye con el aumento de intensidad de luz incidente. También puede ser llamado fotoresistor, fotoconductor, célula fotoeléctrica o resistor dependiente de la luz, cuyas siglas (LDR) se originan de su nombre en inglés light-dependent resistor.

Un fotoresistor está hecho de un semiconductor de alta resistencia. Si la luz que incide en el dispositivo es de alta frecuencia, los fotones son absorbidos por la elasticidad del semiconductor dando a los electrones la suficiente energía para saltar la banda de conducción. El electrón libre que resulta (y su hueco asociado) conduce electricidad, de tal modo que disminuye la resistencia.

Un dispositivo fotoeléctrico puede ser intrínseco o extrínseco. En dispositivos intrínsecos, los únicos electrones disponibles están en la banda de la valencia, por lo tanto el fotón debe tener bastante energía para excitar el electrón a través de toda la banda prohibida. Los dispositivos extrínsecos tienen impurezas agregadas, que tienen energía de estado a tierra más cercano a la banda de conducción puesto que los .electrones no tienen que saltar lejos, los fotones más bajos de energía (es decir, de mayor longitud de onda y frecuencia más baja) son suficientes para accionar el dispositivo.

.

Se fabrican de diversos tipos. Se pueden encontrar células baratas de sulfuro del cadmio en muchos artículos de consumo, por ejemplo cámara fotográfica, medidores de luz, relojes con radio, alarmas de seguridad y sistemas de encendido y apagado del alumbrado de calles en función de la luz ambiente. En el otro extremo de la escala, los fotoconductores de Ge: Cu son los sensores que funcionan dentro de la gama más baja "radiación infrarroja".

Figura 16. Fotorresistencia

III. 2.1.6.2Láser[8]Un láser (*Light Amplification by Stimulated Emission of Radiation, Amplificación de Luz por Emisión Estimulada de Radiación*) es un dispositivo que utiliza un efecto de la mecánica cuántica, la emisión inducida o estimulada, para generar un haz de luz coherente de un medio adecuado y con el tamaño, la forma y la pureza controlados.

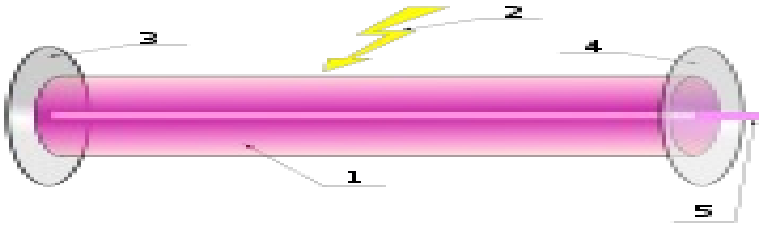
En 1916, Albert Einstein estableció los fundamentos para el desarrollo de los lásers y de sus predecesores, los máseres (que emiten microondas), utilizando la ley de radiación de Max Planck basada en los conceptos de emisión espontánea e inducida de radiación.

Ya en el siglo XXI, científicos de la *Universidad de St. Andrews* crean un láser que puede manipular objetos muy pequeños. Al mismo tiempo, científicos japoneses crean

8 M. Hermann, M. Norton , Basilik subscale laser experiments, 1992-1993.

.

objetos del tamaño de un glóbulo rojo utilizando el láser. En 2002, científicos australianos "teletransportan" con éxito un haz de luz láser de un lugar a otro. Dos años después el escáner láser permite al Museo Británico efectuar exhibiciones virtuales.[] En 2006, científicos de la compañía Intel descubren la forma de trabajar con un chip láser hecho con silicio abriendo las puertas para el desarrollo de redes de comunicación mucho más rápidas y eficientes.



Componentes principales:

1. Medio activo para la formación del láser
2. Energía bombeada para el láser
3. Espejo reflectante al 100%
4. Espejo reflectante al 99%
5. Emisión del rayo láser

Los láseres constan de un medio activo capaz de generar el láser. Hay cuatro procesos básicos que se producen en la generación del láser, denominados bombeo, emisión espontánea de radiación, emisión estimulada de radiación y absorción.

IV. Aplicaciones

El tamaño de los láser varía ampliamente, desde diodos láser microscópicos (arriba) con numerosas aplicaciones, al láser de cristales de neodimio con un tamaño similar al de un campo de fútbol, (abajo) usado para la fusión de confinamiento inercial, investigación sobre armas nucleares de destrucción masiva u otros experimentos físicos en los que se presenten altas densidades de energía

.

En bastantes aplicaciones, los beneficios del láser se deben a sus propiedades físicas como la coherencia, la alta mono-cromaticidad y la capacidad de alcanzar potencias extremadamente altas. A modo de ejemplo, un haz láser altamente coherente puede ser enfocado por debajo de su límite de difracción que, a longitudes de onda visibles, corresponde solamente a unos pocos nanómetros. Cuando se enfoca un haz de láser potente sobre un punto, éste recibe una enorme densidad de energía.[5] Esta propiedad permite al láser grabar gigabytes de información en las microscópicas cavidades de un DVD o CD. También permite a un láser de media o baja potencia alcanzar intensidades muy altas y usarlo para cortar, quemar o incluso sublimar materiales.

A continuación se expone de forma casi íntegra la nueva clasificación publicada en la norma UNE EN 60825-1/A2, y en la tabla 3 un resumen simplificado de la misma.

Clase 1: Productos láser que son seguros en todas las condiciones de utilización razonablemente previsibles, incluyendo el uso de instrumentos ópticos en visión directa.
Clase 1M: Láser que emitiendo en el intervalo de longitudes de onda entre 302,5 y 4000 nm son seguros en condiciones de utilización razonablemente previsibles, pero que pueden ser peligrosos si se emplean instrumentos ópticos para visión directa.
Clase 2: Láser que emiten radiación visible en el intervalo de longitudes de onda comprendido entre 400 y 700 nm. La protección ocular se consigue normalmente por las respuestas de aversión. Esta reacción puede proporcionar la adecuada protección aunque se usen instrumentos ópticos.
Clase 2M: Láser que emiten radiación visible (400 y 700 nm). La protección ocular se consigue normalmente por las respuestas de

.

aversión, pero la visión del haz puede ser peligrosa si se usan instrumentos ópticos.
Clase 3R: Láser que emiten entre 302,5 y 106 nm, cuya visión directa del haz es potencialmente peligrosa pero su riesgo es menor que para los láser de Clase 3B. Necesitan menos requisitos de fabricación y medidas de control del usuario que los aplicables a láser de Clase 3B. El límite de emisión accesible es menor que 5 veces el LEA de la Clase 2 en el rango 400-700 nm, y menor de 5 veces el LEA de la Clase 1 para otras longitudes de onda.
Clase 3B: Láser cuya visión directa del haz es siempre peligrosa (por ej. dentro de la Distancia Nominal de Riesgo Ocular)
Clase 4: Láser que también pueden producir reflexiones difusas peligrosas. Pueden causar daños sobre la piel y pueden también constituir un peligro de incendio. Su utilización precisa extrema precaución.

Tabla 3. Clasificación de láser según UNE EN 60825-1 /A2-2002

V.

VI. 2.1.6.3. Micro-controlador[9]. Un micro-controlador es un circuito integrado o chip que incluye en su interior las tres unidades funcionales de una computadora: unidad central de procesamiento, memoria y unidades de E/S (entrada/salida).

VII. Características

Son diseñados para aumentar el costo económico y el consumo de energía de un sistema en particular. Por eso el tamaño de la unidad central de procesamiento, la cantidad de memoria y los periféricos incluidos dependerán de la aplicación. El control

9 Tomas Pollan, Santamaría, Electrónica Digital. Universidad de Zaragoza.

.

de un electrodoméstico sencillo como una batidora, utilizará un procesador muy pequeño (4 u 8 bit) por que sustituirá a un autómata finito. En cambio un reproductor de música y/o vídeo digital (mp3 o mp4) requerirá de un procesador de 32 bit o de 64 bit y de uno o más Códec de señal digital (audio y/o vídeo). El control de un sistema de frenos ABS (Antilock Brake System) se basa normalmente en un micro-controlador de 16 bit, al igual que el sistema de control electrónico del motor en un automóvil.

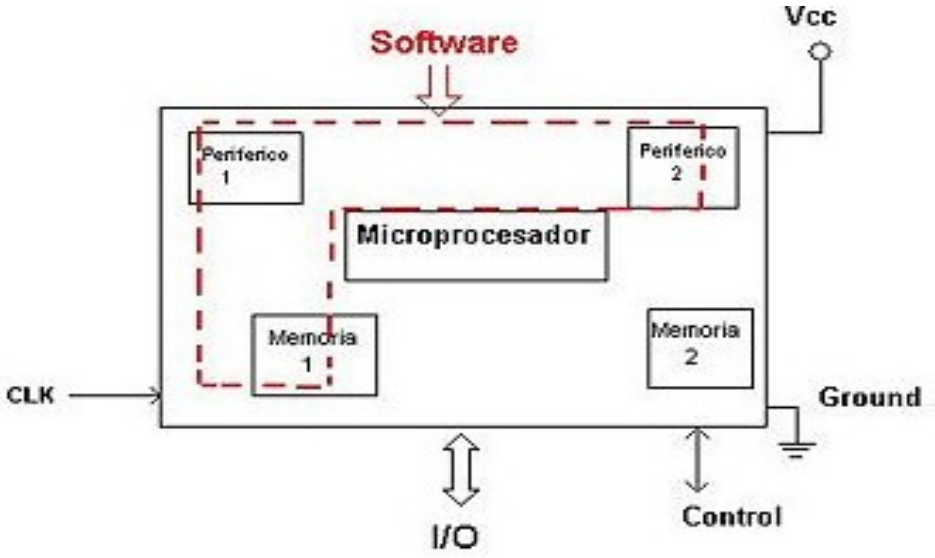


Figura 23. Esquema de un micro-controlador.

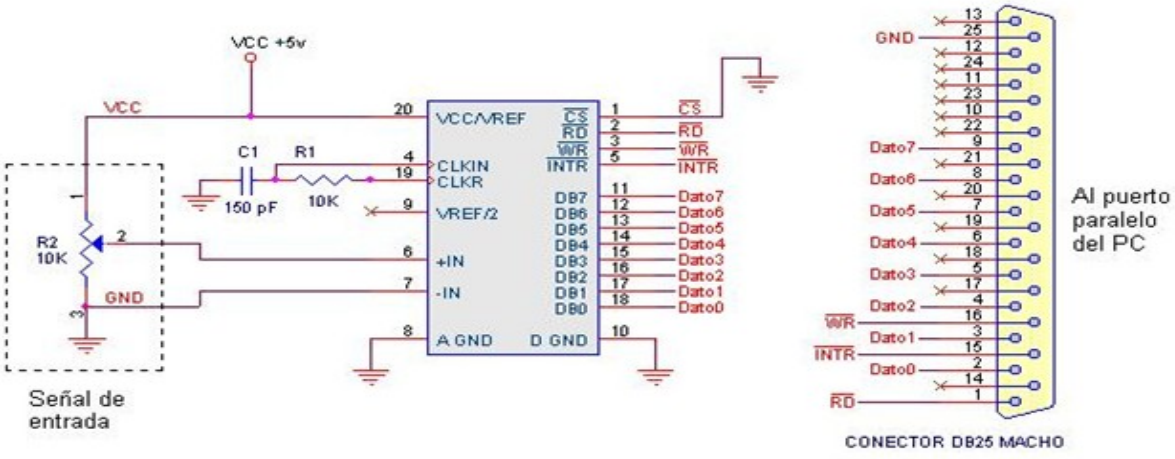


Figura 18. Diagrama de un micro-controlador

.

Los micro-controladores representan la inmensa mayoría de los chips de computadoras vendidos, sobre un 50% son controladores "simples" y el restante corresponde a DSPs más especializados. Mientras se pueden tener uno o dos microprocesadores de propósito general en casa (Vd. está usando uno para esto), usted tiene distribuidos seguramente entre los electrodomésticos de su hogar una o dos docenas de micro-controladores. Pueden encontrarse en casi cualquier dispositivo electrónico como automóviles, lavadoras, hornos microondas, teléfonos, etc.

Un micro-controlador difiere de una CPU normal, debido a que es más fácil convertirla en una computadora en funcionamiento, con un mínimo de chips externos de apoyo. La idea es que el chip se coloque en el dispositivo, enganchado a la fuente de energía y de información que necesite, y eso es todo. Un microprocesador tradicional no le permitirá hacer esto, ya que espera que todas estas tareas sean manejadas por otros chips. Hay que agregarle los módulos de entrada/salida (puertos) y la memoria para almacenamiento de información.

Por ejemplo, un micro-controlador típico tendrá un generador de reloj integrado y una pequeña cantidad de memoria RAM y ROM/EPROM/EEPROM/FLASH, significando que para hacerlo funcionar, todo lo que se necesita son unos pocos programas de control y un cristal de sincronización. Los micro-controladores disponen generalmente también de una gran variedad de dispositivos de entrada/salida, como convertidores de analógico a digital, temporizadores, UARTs y buses de interfaz serie especializados, como I^2C y CAN. Frecuentemente, estos dispositivos integrados pueden ser controlados por instrucciones de procesadores especializados. Los modernos micro-controladores frecuentemente incluyen un lenguaje de programación integrado, como el BASIC que se utiliza bastante con este propósito.

Los micro-controladores distribuyen la velocidad y la flexibilidad para facilitar su uso. Debido a que se utiliza bastante capacidad en el chip para incluir funcionalidad, como los dispositivos de entrada/salida o la memoria que incluye el micro-controlador, se ha de prescindir de cualquier otra circuitería.

.

VIII. Estructura básica de un micro-controlador

En esta figura, se observa un micro-controlador metido dentro de un encapsulado de circuito integrado, con su procesador (CPU), buses, memoria, periféricos y puertos de entrada salida. Fuera del encapsulado se ubican otros circuitos para completar periféricos internos y dispositivos que pueden conectarse a los pines de entrada/salida. También se conectarán a los pines del encapsulado la alimentación, masa, circuito de completamiento del oscilador y otros circuitos necesarios para que el micro-controlador pueda trabajar.

IX. Núcleo de un micro-controlador

Aún cuando el micro-controlador es una computadora embebida dentro de un circuito integrado, se compone de un núcleo y un conjunto de circuitos adicionales. Dentro del núcleo se encuentran el procesador y la memoria, todo ello estructurado de forma tal que conforme una arquitectura de computadora.

Aunque la importancia de los registros parezca trivial, no lo es en absoluto. De hecho una parte de los registros, la destinada a los datos, es la que determina uno de los parámetros más importantes de cualquier microprocesador. Cuando escuchamos que un procesador es de 4, 8, 16, 32 ó 64 bits, nos estamos refiriendo a procesadores que realizan sus operaciones con registros de datos de ese tamaño, y por supuesto, esto determina muchas de las potencialidades de estas máquinas.

Mientras mayor sea el número de bits de los registros de datos del procesador, mayores serán sus prestaciones, en cuanto a poder de cómputo y velocidad de ejecución, ya que este parámetro determina la potencia que se puede incorporar al resto de los componentes del sistema, por ejemplo, no tiene sentido tener una ALU de 16 bits en un procesador de 8 bits.

Por otro lado un procesador de 16 bits, puede que haga una suma de 16 bits en un solo ciclo de máquina, mientras que uno de 8 bits deberá ejecutar varias instrucciones antes de tener el resultado, aún cuando ambos procesadores tengan la misma velocidad de ejecución para sus instrucciones. El procesador de 16 bits será más rápido porque puede hacer el mismo tipo de tareas que uno de 8 bits, en menos tiempo.

.

x. **Conjunto de instrucciones**

Aunque no aparezca en el esquema, no podíamos dejar al conjunto o repertorio de instrucciones fuera de esta fiesta, porque este elemento determina lo que puede hacer el procesador.

Define las operaciones básicas que puede realizar el procesador, que conjugadas y organizadas forman lo que conocemos como software. El conjunto de instrucciones vienen siendo como las letras del alfabeto, el elemento básico del lenguaje, que organizadas adecuadamente permiten escribir palabras, oraciones y cuanto programa se le ocurra.

Familias de micro-controladores

Los micro-controladores más comunes en uso son:

Empresa	8 bits	12 bits	14 bits	16 bits	32 bits	64 bits	Observ.
Atmel	ATmega8,89Sxxxx			ATmega16			
AVR	familia similar 8051						
Freescale	68HC05, 68HC08,	X	X	68HC12,68HCS12	683xx, PowerPC	MCF51C	x
(antes Motorola)	68HC11, HCS08			68HCSX12,68HC16	Architecture, ColdFire	N128	
Hitachi, Ltd	H8	X	X	X	X	x	x
Holtek	HT8						
Intel	MCS-48 (familia 8048)	X	X	MCS96, MXS296	X	x	x
	MCS51 (familia 8051)						
	8xC251						
National Semiconductor	COP8	X	X	X	X	x	x
Microchip	Familia 10f2xx Familia 12Cxx Familia 12Fxx, 16Cxx y 16Fxx 18Cxx y 18Fxx			PIC24F, PIC24H y dsPIC30FXX,dsPIC 33F con motor dsp integrado	PIC32	x	x
NEC Corporation	78K						
STMicroelectronics	ST 62,ST 7						

1. **Online/OEM products**

[

.

xi. Conversor analógico/digital

Como es muy frecuente el trabajo con señales analógicas, éstas deben ser convertidas a digital y por ello muchos de los micro-controladores incorporan un conversor A/D, el cual se utiliza para tomar datos de varias entradas diferentes que se seleccionan mediante un multiplexor.

Las resoluciones más frecuentes son 8 y 10bits, aunque hay micro-controladores con conversores de 11 y 12 bits, para resoluciones mayores es preciso utilizar conversores A/D externos. Los conversores A/D son uno de los periféricos más codiciados en el mundo de los micro-controladores y es por ello que muchísimos PIC los incorporan, siendo esta una de las características más destacables de los dispositivos que fabrica Microchip.

xii. Puerto serie

Este periférico está presente en casi cualquier micro-controlador, normalmente en forma de UART (Universal Asynchronous Receiver Transmitter) o USART (Universal Synchronous Asynchronous Receiver Transmitter) dependiendo de si permiten o no el modo sincrónico de comunicación.

El destino común de este periférico es la comunicación con otro micro-controlador o con una PC y en la mayoría de los casos hay que agregar circuitos externos para completar la interfaz de comunicación. La forma más común de completar el puerto serie es para comunicarlo con una PC mediante la interfaz EIA-232 (más conocida como RS-232), es por ello que muchas personas se refieren a la UART o USART como puerto serie RS-232, pero esto constituye un error, puesto que este periférico se puede utilizar para interconectar dispositivos mediante otros estándares de comunicación.

xiii. Puerto serie sincrónico

Este tipo de periférico se utiliza para comunicar al micro-controlador con otros micro-controladores o con periféricos externos conectados a él, mediante las interfaces SPI (Serial Peripheral Interface) o I2C (Inter-Integrated Circuit).

A pesar de que es también un tipo de puerto serie, es costumbre tratarlo de forma diferenciada respecto a la UART/USART porque las interfaces SPI e I2C aparecieron

.

mucho después que la UART/USART, su carácter es únicamente sincrónico y no están diseñadas para interconectar al sistema con otros dispositivos independientes como una PC, sino para conectar al micro-controlador dispositivos tales como memorias, pantallas LCD, conversores A/D o D/A.

xiv. Otros puertos de comunicación

En un mundo cada vez más orientado a la interconexión de dispositivos, han aparecido muchas interfaces de comunicación y los micro-controladores no se han quedado atrás para incorporarlas, es por ello que podemos encontrar algunos modelos con puertos USB (Universal Serial Bus), CAN (Controller Area Network), Ethernet, puerto paralelo entre otros.

xv. Comparadores

Son circuitos analógicos basados en amplificadores operacionales que tienen la característica de comparar dos señales analógicas y dar como salida los niveles lógicos '0' o '1' en dependencia del resultado de la comparación. Es un periférico muy útil para detectar cambios en señales de entrada de las que solamente nos interesa conocer cuando está en un rango determinado de webetas

- FREESCALE TWR MCF51CN128

Este kit de desarrollo es estructura modular exclusiva en freescale basada en un hardware reconfigurable y con capacidad de permitir el intercambio de micro controladores y tarjetas de periféricos de acuerdo a las necesidades de los ingenieros.

Esta cualidad abástese de un rango extra de posibilidad a los usuarios para estandarizar si es aplicable a sus proyectos, sirviéndose de otros elementos como el hardware de contenido abierto para la creación de de herramientas de fácil conexión para las necesidades de de los diseños.

De igual manera el tower system incluye el modulo elevador (TWR-Ele) que suministra la regulación de potencia e integridad estructural de la misma plataforma modular y

.

términos de compatibilidad su más cercano complemento es el micro controlador Ethrnet MCF51CN128 COLDFIRE v1 que así interpola el software exclusivo MQX, así como un modulo periférico (TWR-Serv) para conexiones de protocolo USB Ethrnet, CAN y RS323 /232 /435

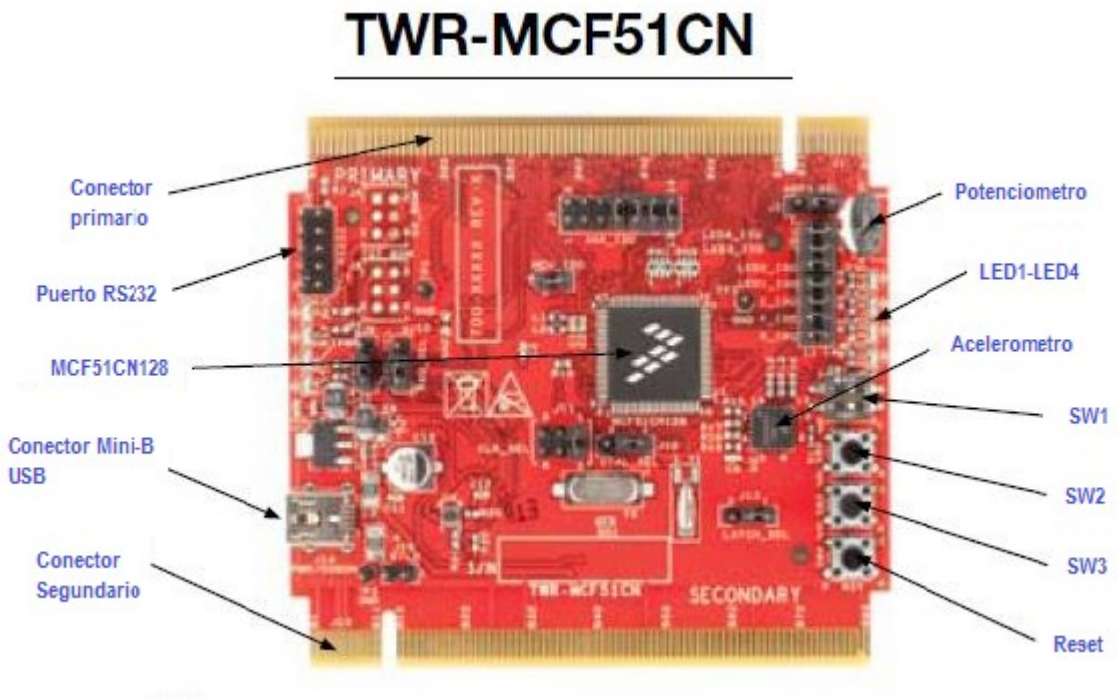


Figura 19. TWR MCF51CN128

2.1.7. SOFTWARE

2.1.7.1*Visual C# express 2008.* Visual Studio express permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión net 2002, se incorpora las versiones Framework 3.5 y Framework 4.0 para las ediciones 2005 ,2008, 2010). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles. Cabe destacar que estas ediciones son iguales al entorno de desarrollo comercial de visual estudio professional pero sin características avanzadas. Las ediciones que hay dentro de cada suite son:

.

- **Visual Basic** Express Edition.
- **Visual C#** Express Edition.
- **Visual C++** Express Edition.
- **Visual J#** Express Edition (Desaparecido en Visual Studio express 2008).
- Visual Web Developer Express Edition:

Para programación con lenguaje ASP.NET. Está orientado a la programación y diseño web, incluyendo un editor visual WYSIWYG y otro HTML con autocompletado de código (IntelliSense), coloración de sintaxis y validación. Aparte de ASP. NET, también soporta Visual Basic .NET y C Sharp (C#).También tiene un servidor web local para realizar pruebas en ASP.NET, un depurador para ubicar errores en el código fuente y una herramienta de publicación en línea de sitios creados.

- Adicionalmente, Microsoft ha puesto gratuitamente a disposición de todo el mundo una versión reducida de MS SQL Server llamada **SQL Server Express** Edition cuyas principales limitaciones son que no soporta bases de datos superiores a 4 GB de tamaño, únicamente utiliza un procesador y un Gb de Ram, y no cuenta con el Agente de SQL Server.

En el pasado se incluyeron los siguientes productos, actualmente desaparecidos en versiones como Visual Studio express 2005 y 2008:

- Visual InterDev.
- Visual J++.
- Visual FoxPro.
- Visual SourceSafe.

2.1.7.2 CODE WARRIOR. CodeWarrior fue desarrollado originalmente por Metrowerks basa en un compilador de C y el medio ambiente de 68k, desarrollado por Andreas Hommel y con licencia para Metrowerks. Las primeras versiones de CodeWarrior dirigido el PowerPC Macintosh, con gran parte del desarrollo realizado por un grupo de la original THINK C del equipo. Al igual que C, que era conocido por su rápido tiempo

.

de compilación, CodeWarrior fue más rápido que el Macintosh Programmer's Workshop (MOP), las herramientas de desarrollo por escrito de Apple.

CodeWarrior es un factor clave en el éxito de la transición de Apple de su arquitectura de la máquina de Motorola 68K procesadores PowerPC , ya que proporciona un sólido PowerPC compilador completa cuando la competencia (Ministerio de Obras Públicas de las herramientas de Apple y Symantec C + +) fue en su mayoría incompletas. Metrowerks también hizo fácil para generar los binarios de grasa que incluía tanto 68K y el código PowerPC.

Sin embargo, después de Metrowerks fue adquirida por Motorola en 1999, la compañía se concentró en aplicaciones embebidas, dedicando una fracción más pequeña de sus esfuerzos a los compiladores para las computadoras de escritorio. El 29 de julio de 2005, se anunció que CodeWarrior para Mac se interrumpieron después de la próxima versión, CodeWarrior Pro 10. Aunque Metrowerks no detalló sus razones, la demanda de CodeWarrior había caído, presumiblemente después de que Apple comenzó a distribuir el desarrollo de herramientas gratuitas con Mac OS X. Además, de cambio de Apple a Intel chips izquierda Metrowerks sin un producto evidentes, ya que había vendido su tecnología de los compiladores Intel Nokia a principios de 2005.

Las versiones anteriores de CodeWarrior todavía están en uso por retro informática entusiastas en el Mac OS clásico, entre otros proyectos, Classilla actualmente construida con ella.

Durante su apogeo, el producto era conocido por su ciclo de liberación rápida, con múltiples revisiones cada año, y por su peculiar campaña publicitaria.

2.2 MARCO CONCEPTUAL

.

2.2.1 Que es MIDI.MIDI es el lenguaje para el envió y la recepción de información para que los dispositivos como computadores, sintetizadores y software se puedan comunicar.

2.2.2 Definición de SAMPLER.[10] El sampler es una interfaz gráfica que permite muestrear digitalmente secuencias sonoras o samples para ser reproducidas posteriormente, o transformadas mediante efectos. También permite recuperar y almacenar, este es utilizado como herramienta en muchos géneros musicales, entre los que destaca la música electrónica.

2.2.3 Definición de Muestrear.[11] Dicho de una forma muy simple, muestrear es hacer fotos de un sonido a mucha velocidad. La comparación más fácil es el cine: el ojo humano necesita unas 24 imágenes o "fotos" por segundo para crear la ilusión de movimiento continuo.

2.2.4 Definición de Audio: Con este nombre se identifica la señal sonora una vez transformada en señal eléctrica.

2.2.5 Definición Tiempo de Reverberación: Tiempo transcurrido entre una vez que cesa la emisión hasta que el valor de la presión sonora sea la millonésima parte de su valor primitivo.

2.2.6 Definición Edición de Audio: Proceso mediante el cual se le cambia la apariencia de una señal sonora, bien sea cortándola, fragmentarla, doblándola, entre

---

10**Breve Historia y Teoría del Sampler**

[historiasampler.pdf]

11 Alberto Gutiérrez, Septiembre 2000.Muestreo, Sampler, Historia y tecnología.

.

otras, en diferentes señales eléctricas, que a su vez, van a ser entregadas en forma distinta a la señal original.

2.2.7 láser. Haz de luz generado para trazar las cuerdas del prototipo.

2.2.8 Patrón Polar. Campo y dirección de captura que posee un micrófono.

2.2.9 Fotorresistencia. Transductor que recibe los valores de voltaje generados por el haz del laser.

2.3 MARCO LEGAL

2.3.1. *CodeWarrior* Metrowerks y Motorola han dejado disponible al público una edición especial del entorno de desarrollo para micro-controladores HC08: CodeWarrior v4.0, la cual es libre y puede compilar hasta 4Kbytes de código C. Una de las ventajas importantes de esta nueva versión es que adiciona el simulador de P&E Microcomputer Systems Inc., el cual cuenta con una máquina virtual que permite simular la CPU, periféricos e interrupciones de todos los micro-controladores HC08 actuales, lo que facilita el proceso de depuración de las aplicaciones desarrolladas en lenguaje C. Los instaladores se pueden descargar fácilmente de la página de Freescale

2.3.2. Microsoft Visual Studio Express Edition. Es un programa de desarrollo en entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows desarrollado y distribuido por Microsoft Corporation. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros. Es de carácter gratuito y es proporcionado por la compañía Microsoft Corporation orientado el producto a principiantes, estudiantes y entusiastas en la programación web y de aplicaciones, ofreciéndose dicha aplicación a partir de la versión 2005 de Microsoft visual studio.

.

2.3.3. WaveLab es un editor de audio digital de Steinberg dirigida a los profesionales, así como la semi-profesional del mercado. Soporta archivos multi-canal, DirectX plugin, plugin VST y DVD-Audio creación. WaveLab fue durante mucho tiempo el principal competidor de Sound Forge. Wavelab se inició en 1995 y es principalmente el trabajo de un programador, Philippe Goultier. Hacia abajo versiones Corte de WaveLab WaveLab Studio incluyen, esenciales WaveLab WaveLab y LE. El programa es un favorito entre el dominio de los ingenieros que trabajan en el ámbito digital.

Steingberg ofrece al público un trial libre para la realización de pruebas, totalmente libre.

3. METODOLOGÍA

3.1 ENFOQUE DE LA INVESTIGACIÓN

.

El enfoque de la investigación es de tipo empírico analítico ya que el objetivo de este proyecto se basa en el análisis detallado del diseño y la construcción de un producto utilizando los conocimientos de la electrónica, sistemas y sonido; desarrollando una herramienta nueva e innovadora.

3.2 LÍNEA INSTITUCIONAL DE USB / SUB- LÍNEA DE FACULTAD / CAMPO TEMÁTICO DEL PROGRAMA

Figura 20 .Diagrama Líneas USB

3.2.1 Línea Institucional

El trabajo se desarrolla en la de línea de *TECNOLOGÍAS ACTUALES Y SOCIEDAD,* ya que el proyecto presenta una nueva herramienta para la sociedad, que ayuden a mejorar las diferentes presentaciones en vivo, tanto en la interpretación cono en la audiencia

3.2.2 Sub- Línea de Facultad

Este proyecto se basa en el *PROCESAMIENTO DE SEÑALES* ya que integra varios métodos para la implementación e intervención de señales.

3.2.3 Campo Temático del Programa

Este trabajo se enfoca en el *DISEÑO DE SISTEMAS DE SONIDO* ya que fundamentalmente es la creación y construcción de un nuevo instrumento que emula sonidos reales y reproduce audio y video simultáneamente.

.

3.3 TÉCNICAS DE RECOLECCIÓN DE INFORMACIÓN

Se realizaron varias encuestas dividida en dos etapas; la primera fue hecha a personas del común con la intensión de conocer que tanto conoce la población sobre algunos temas que son importantes para este proyecto se pregunto sobre MIDI, sobre que es un sampler y por instrumentos que implementen laser, ya que es fundamental en el desarrollo de esta tesis por que el desempeño de este producto se implementara en el área del sonido en vivo.

La segunda Etapa se dirigió a personas que están en más inmersas al contexto en el que se desarrolla el ARPA LASER, este nos permitió conocer la opinión de las personas cercanas conociendo su opinión sobre los sonidos, y el funcionamiento del Arpa Láser

3.4 POBLACIÓN Y MUESTRA

La población en la primera etapa fue la común la que tiene contacto con eventos en vivo, pero no tienen conocimientos específicos en sonido, personas que tienen alta vida social, por que se necesitaba ver y conocer lo que se conoce en general para lograr presentar una nueva opción de ante los espectadores.

El grupo de la segunda etapa fue de personas con experiencia en algunos campos de sonido en vivo y en electrónica, así mismo personas con contactos y conocimientos musicales.

Ver anexo A. Encuesta.

3.5 HIPÓTESIS

La reproducción de audio y video se ha apoderado de la gran mayoría de eventos a lo largo del mundo entero; lo cual hace que sea un requisito obligado el innovar y crear nuevas herramientas que faciliten la interpretación musical para los diferentes shows

.

presentados ante la sociedad; lo cual ha requerido del aporte ingenieril importante para su progreso.

Este proyecto pretende brindar nuevas alternativas en el campo de audio y video, mejorando los espectáculos teniendo en cuenta la innovación no solo en el ámbito musical, sino también en el visual.

3.6 VARIABLES

3.6.1 Variables Independientes

- Efectos sonoros (En la captura banco de sonidos).
- Efectos en el banco de imágenes.

3.6.2 Variables Dependientes

- Dispositivos electrónicos utilizados en el proceso como los son:
  - Láser
  - Fotorresistencias
  - Resistencias
  - Transductores
- Sampler

4. DESARROLLO INGENIERIL

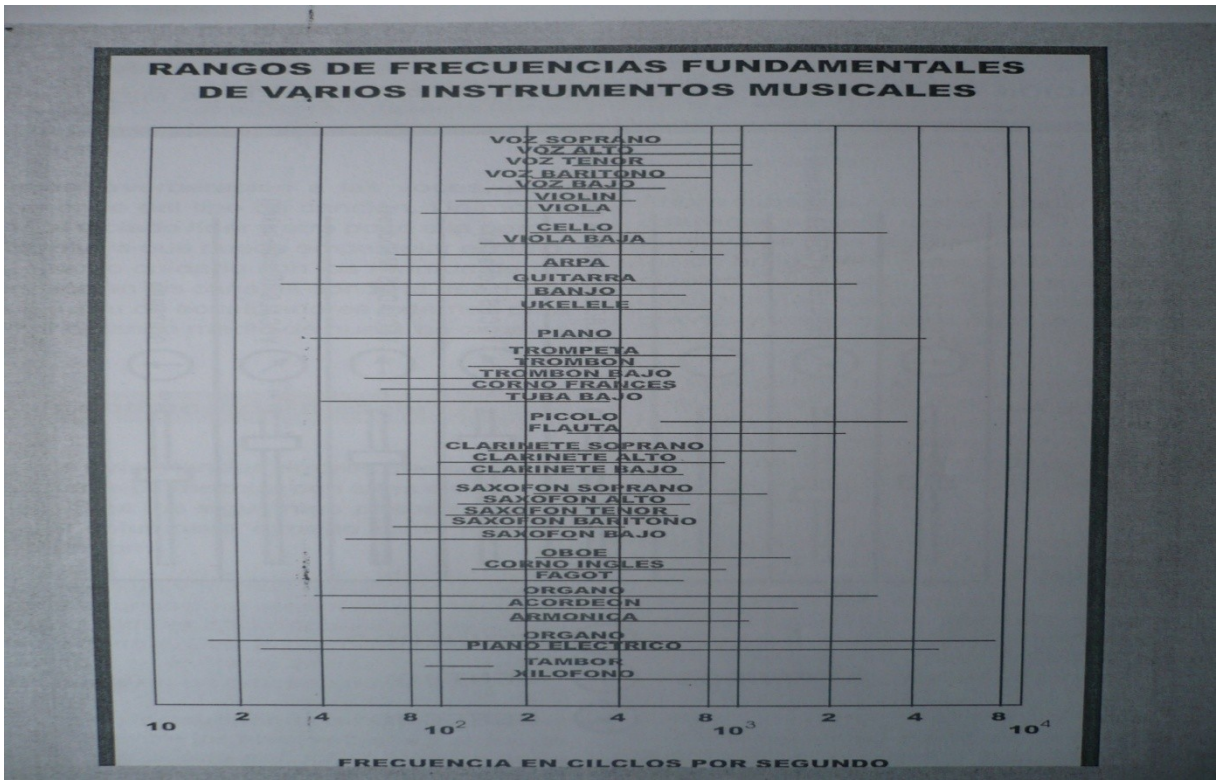
Para el diseño y construcción del **Controlador MIDI de un Arpa Laser** se hizo necesario desarrollar las diferentes metas planteadas en los objetivos específicos, que se irán detallando a continuación:

.

4.1 BANCO DE SONIDOS REALES DEL INSTRUMENTO

Para la creación del banco de sonidos se hizo necesario realizar una investigación teórica a cerca de las técnicas de grabación, así como también, de los tipos de micrófonos para instrumentos de cuerda, y lograr la grabación obteniendo las muestras reales:

4.1.1 Selección y Tipos de Micrófonos. Teniendo en cuenta el rango de frecuencia fundamental del Arpa que se encuentra 40Hz a 4KHz que observa a continuación, se tomaron en cuenta los micrófonos dinámicos SHURE SM 57 ya que presentan un rango de captura similar al Arpa, además que es un micrófono especial para la captura de instrumentos de cuerda. Manteniendo el sonido claro y brillante característico del instrumento.



.

Figura 21. Rangos de Frecuencias fundamentales de varios instrumentos Musicales[12].

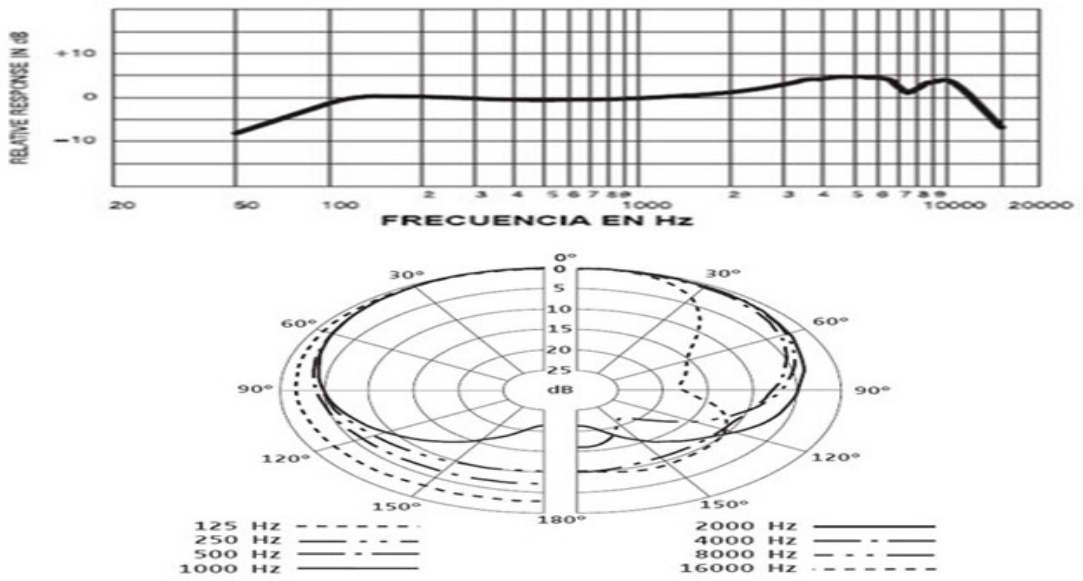


Figura 21.1 Patrón polar y Respuesta en frecuencia Micrófono SHURE SM 57

Además se selecciono un micrófono omnidireccional AKG 451, con el fin de capturar el ambiente ya que presenta una respuesta plana en frecuencia, precisa para instrumentos en altas frecuencias y respuesta en bajos sin pérdida.

12 Martinez, Javier,LA MUSICA Y EL ARTE DE LA GRABACIÖN, Ed.PROSONEX, pag. 93

.

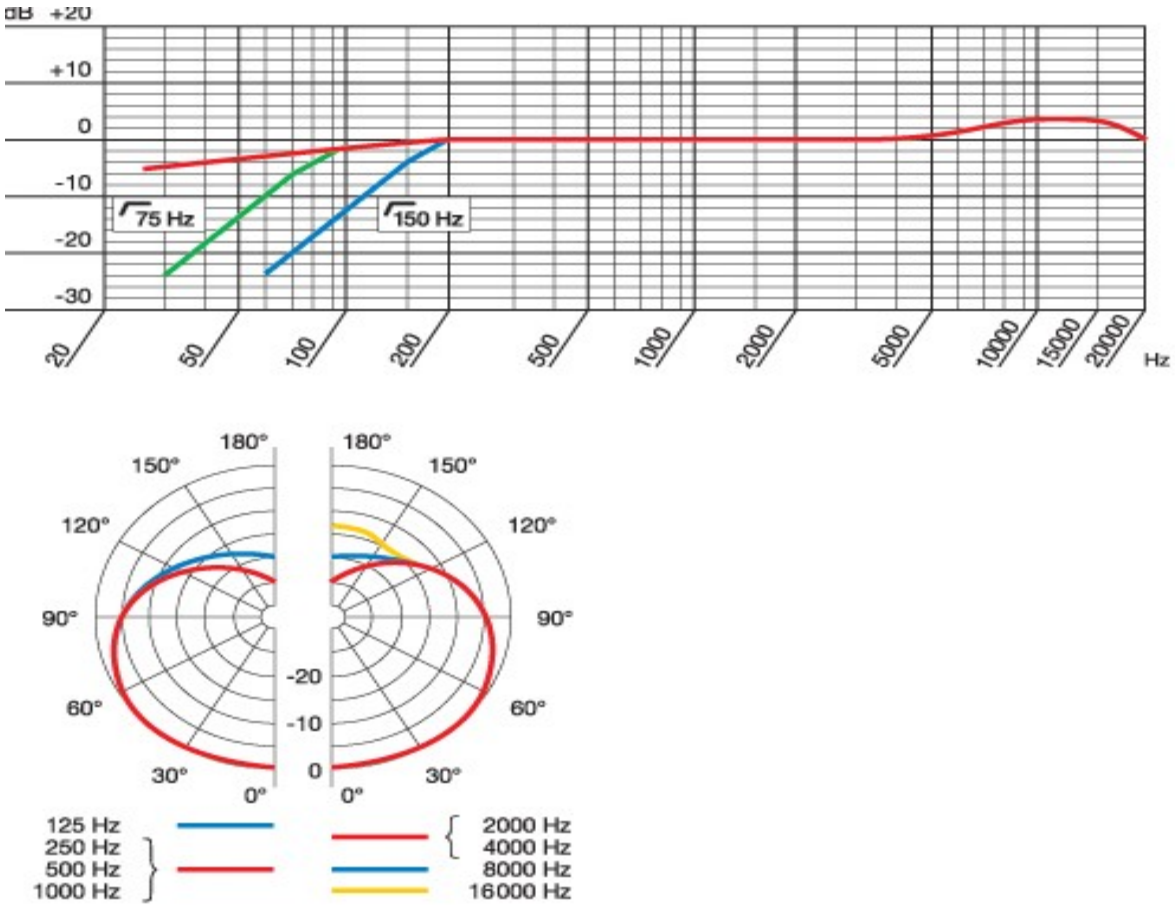


Figura 22. Patrón polar y Respuesta en frecuencia Micrófono AKG 451

4.1.2 técnica de grabación. Para la grabación de las muestras para el SAMPLER se utilizaron los dos micrófonos SM 57, los cuales facilitaban la ejecución de la técnica ORTF, para obtener la captura precisa, la sensación de volumen y profundidad de cada una de las muestras, la cual estaba dispuesta hacia las bocas del arpa, para poder capturar las señales emitidas por estas. Para capturar cada una de las pulsaciones del arpa se utilizó un tercer micrófono SM 57 creando la posibilidad de darle más realismo a las muestras, acercando el micrófono a roce de la uña con la cuerda. Y para la captura del ambiente se utilizó un micrófono omnidireccional como es el AKG 451, ya que por sus características de patrón permite tener una captura de todo el entorno del músico y de la sala. Ver la siguiente figura.

.

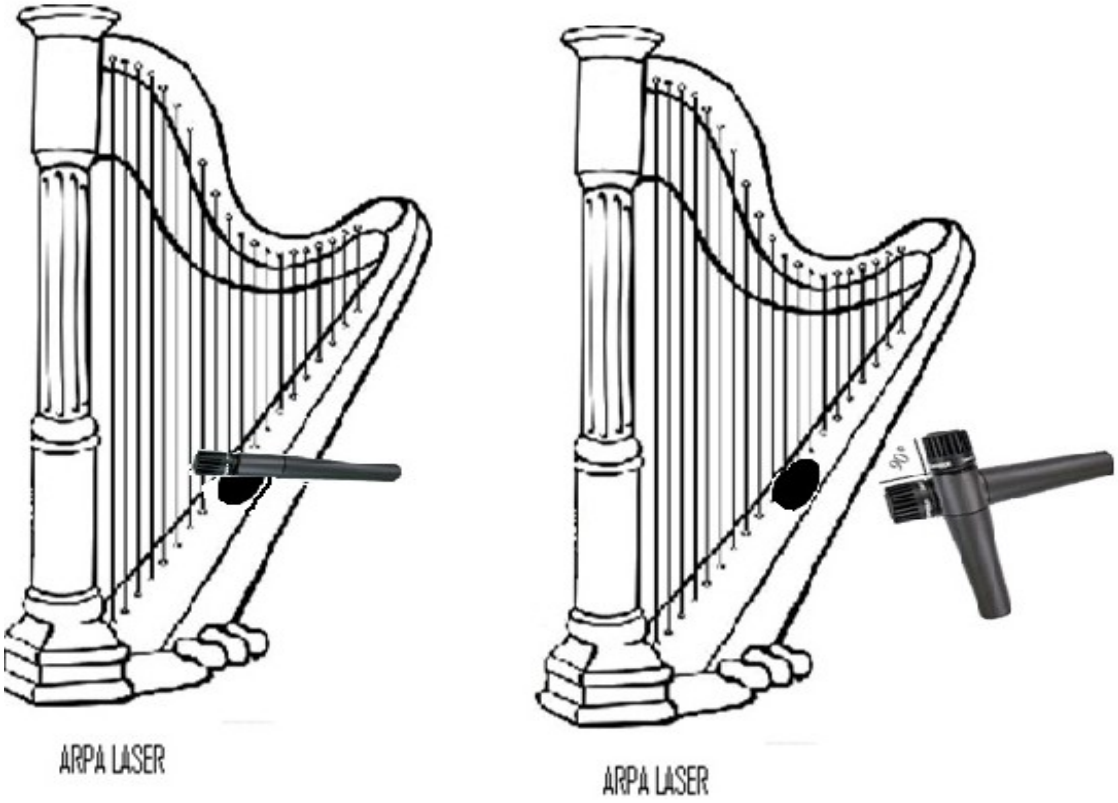


Figura 23. Ubicación de micrófonos

4.1.3 corrección de errores del banco de sonidos. Teniendo en cuenta que hubo dificultades técnicas al realizar el bounce de las muestras obtenidas en el desarrollo del banco de sonidos, se hizo corrección de estos errores y maximización de las señales mediante el programa free trial de WAVELAB 6.1.1., sin embargo se tuvo en cuenta la ganancia de cada toma que fué realizada y se trató de efectuar un incremento de nivel, utilizando el efecto LEVELER, que permite calcular los decibeles exactos para aumentar o disminuir el nivel de la ganancia de la señal.

Por otra parte se implemento el efecto X-NOISE para realizar una redución leve del ruido generado por el intrumentista al momento de la grabación, este efecto hace parte de un banco de plugins llamado WAVES, el cual nos brinda una amplia gama de

.

opciones de efectos entre los cuales se encuentran ecualizadores, compresores. Ver las siguientes figuras:

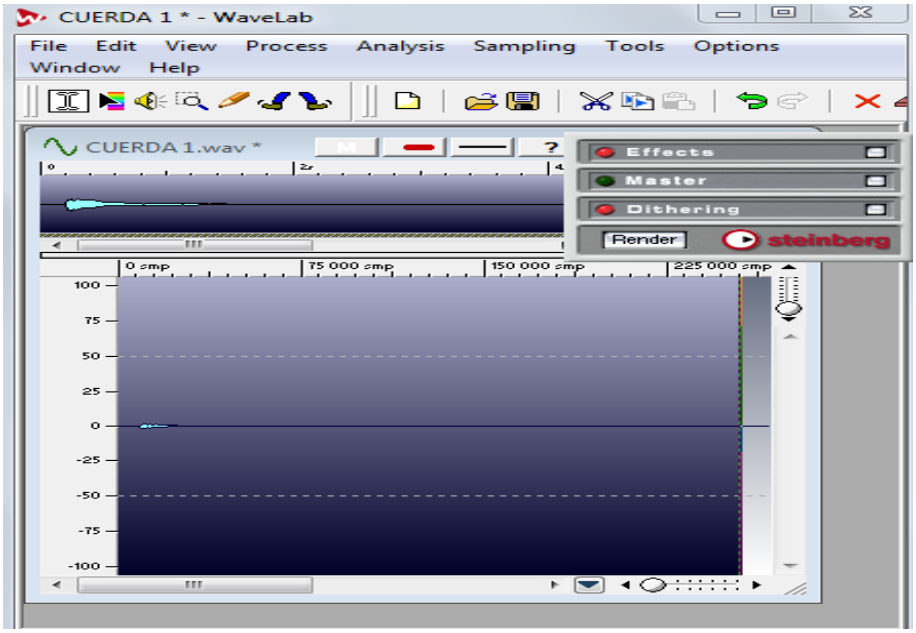


Figura 24, WaveLab introducción archivo wav.

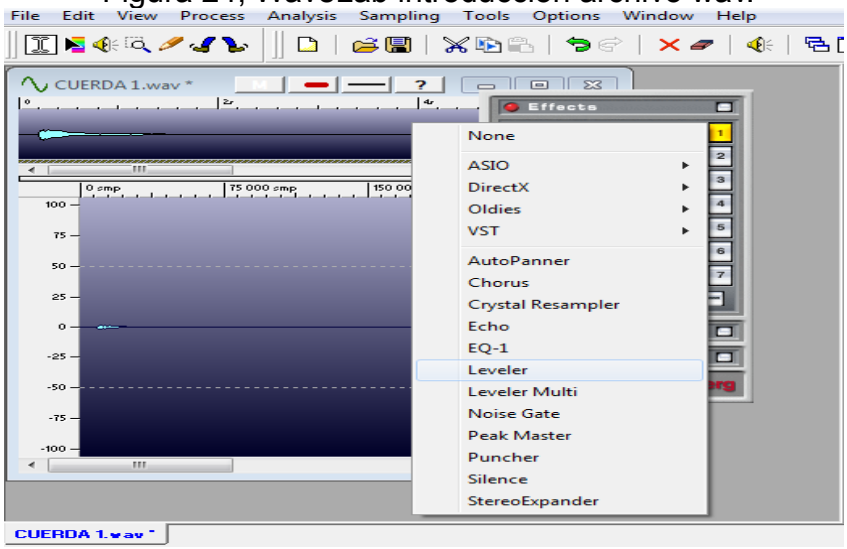


Figura 25. Efecto de LEVELER

.

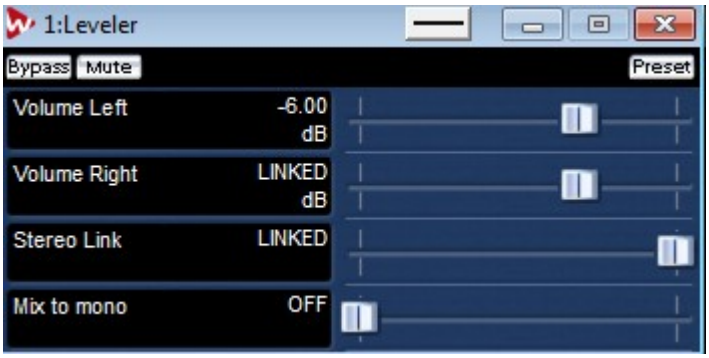


Figura 26. Características del efecto Leveler

En la figura 27 se puede evidenciar las diferentes opciones del efecto, brindando la posibilidad de aumentar la ganancia a la derecha, a la izquierda o a la mezcla como tal.

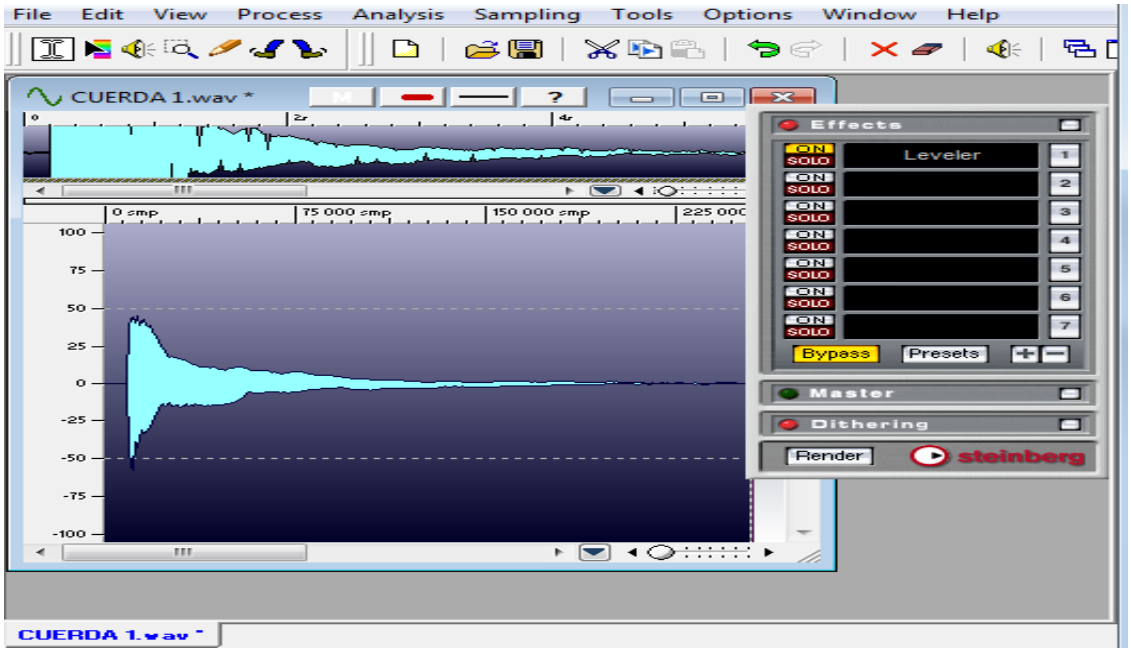


Figura 27. Muestra afectada por el efecto LEVELER.

.

Así en la anterior figura se observa como se ve la muestra después de ser alterada por el efecto LEVELER con un aumento de aproximadamente 24 dB.

Figura 28. Efecto Waves- X-NOISE

Y en la figura 28 se aprecia con el efecto X-Noise, trabaja con una interfaz gráfica la cual permite ver la señal y permite determinar la magnitud del ruido que está afectando la mezcla.

Todo este procedimiento se lo aplicó a cada una de las muestras, teniendo en cuenta la duración de cada una de las mismas para que quedaran igual de tiempo.

4.2 BANCO DE IMÁGENES POR MEDIO DE FOTOS Y VIDEOS

Para el cumplimento de este objetivo se optó por un banco de imágenes y de un video, que se realizó por medio de la captura de imágenes de diferentes lugares de Colombia,

.

en diferentes situaciones cotidianas de cualquier persona, como por ejemplo; un parque en Tabio, un zoológico del parque Jaime Duque entre otras que se muestran aleatoriamente en el SAMPLER y el video que se tomo en cuenta es el que está dado por MAX/MSP.

Ver Anexo F Banco de imágenes.

4.2.1 Desarrollo de SAMPLER por medio de visual C# 2008 Express Edition. Esto se irá reproduciendo aleatoriamente de manera simultánea con la muestra de audio, del banco Ver la siguiente Figura.

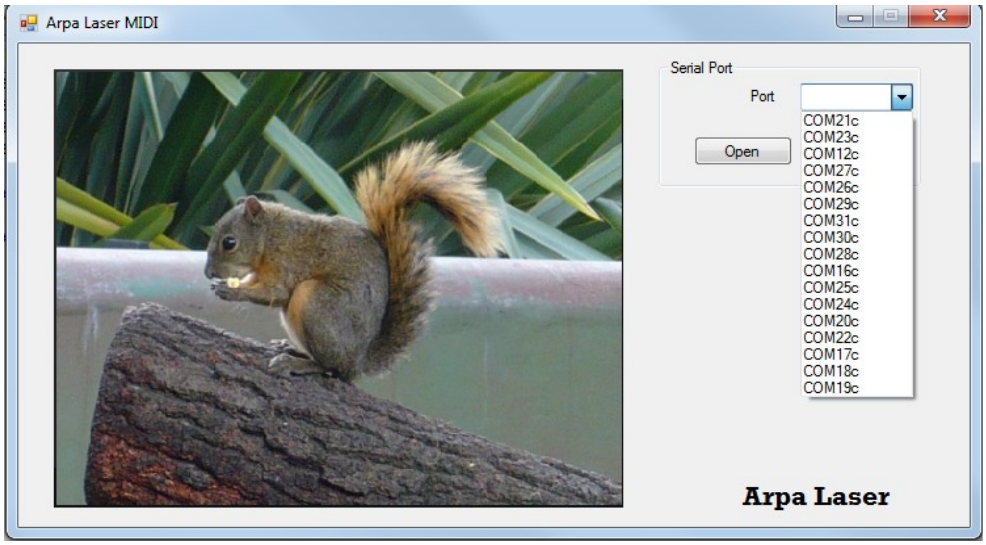


Figura 29. Una de las imágenes del banco

4.2.2 Desarrollo del SAMPLER por medio de MAX/MSP. En esta plataforma se muestra la reproducción del video simultáneamente con el audio, una de las ventajas es él control de algunos paramentos del video como binalizar, parar el video, entre otras opciones. Ver Figura 30.

.

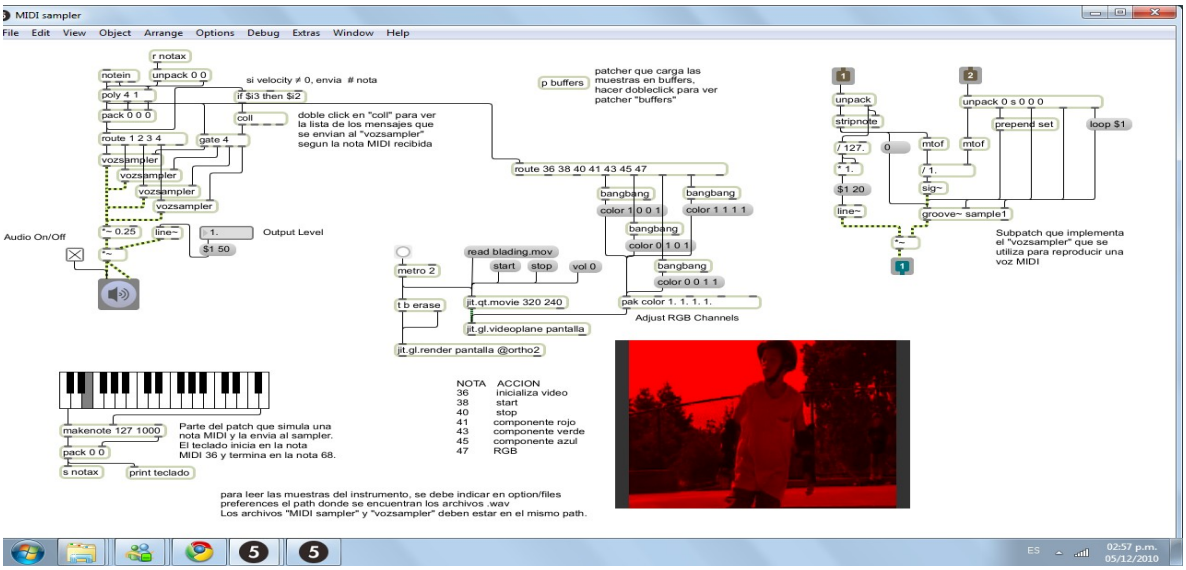


Figura 30. Plataforma Visual por medio MAX/MSP.

Este proceso se detalla con mayor precisión en el siguiente aparte en la programación del SAMPLER.

4.3 DISEÑO Y CONSTRUCCIÓN ELECTRÓNICA DEL ARPA LASER

Para el desarrollo de este objetivo encaminado al diseño y construcción electrónica del Arpa Laser, se tuvo en cuenta las pruebas ensayo y error para determinar los diferentes componentes de los circuitos

4.3.1 Diseño de los circuitos y pruebas del Controlador. Para la construcción del controlador MIDI de un arpa láser se requirió del diseño de un circuito electrónico que permitió el funcionamiento del los componentes y el dispositivo. En este caso se crearon dos tipos de circuitos el primero para los lásers y el segundo para las fotorresistencias.

.

4.3.1.1 circuito dirigido al laser

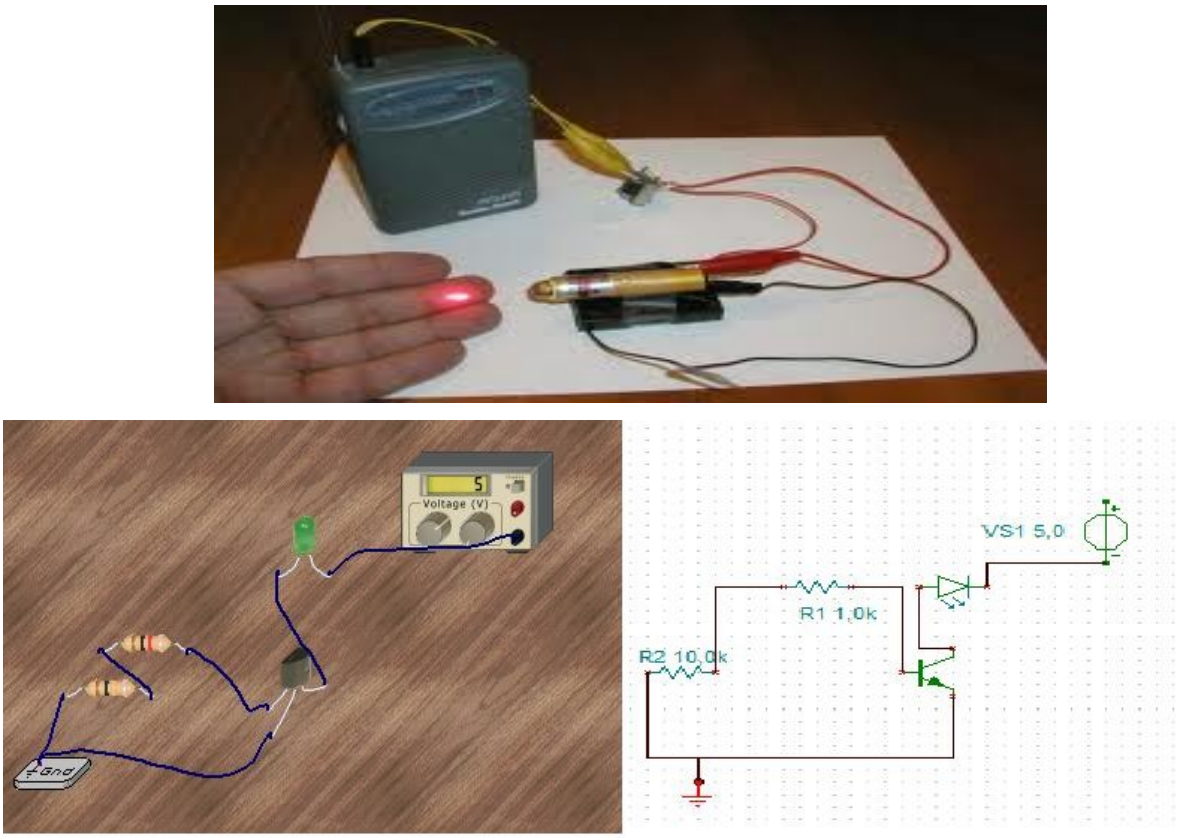


Figura 31.Circuito Laser

En la Figura 31 se muestra el circuito dirigido al laser que está diseñado para enviar datos cada vez que el laser se interrumpe por la mano o un objeto. Este consta de cuatro elementos electrónicos detallándose a continuación:

1. Transistor 2N2222A

2. Resistencia 10k

3. Resistencia de 1k

*4. Laser de 5mv (representado como un led., E*n la Figura 28 se muestra la plantilla que permitió quemar el circuito en la váquela.

.

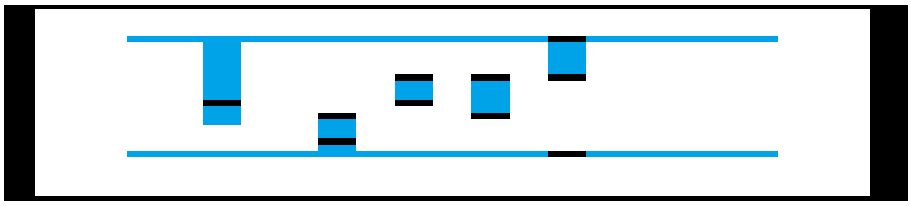


Figura 32 Plantilla de quemado circuito laser

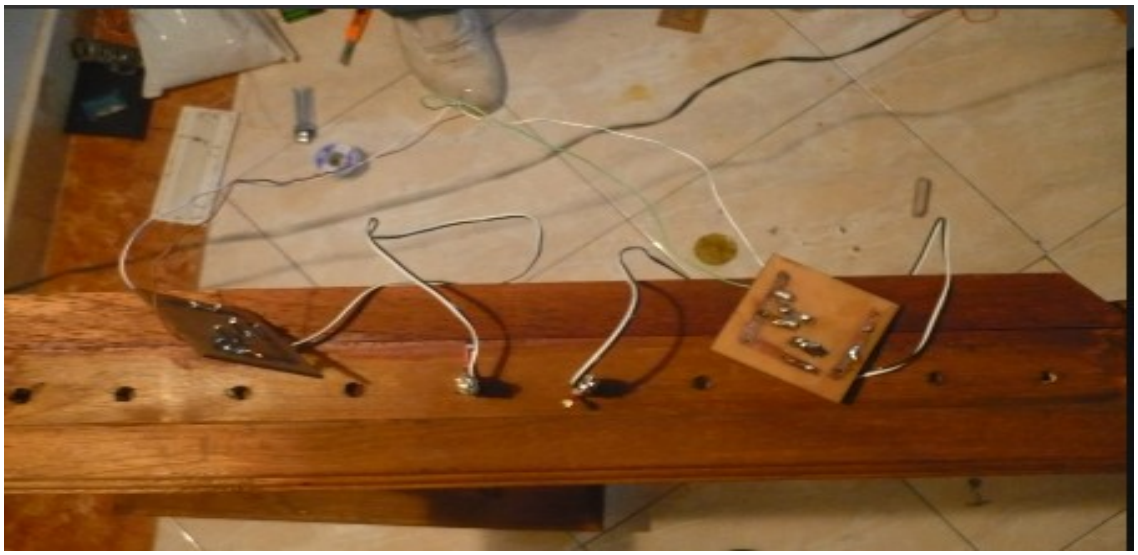


Figura 33 Circuitos Arpa

.

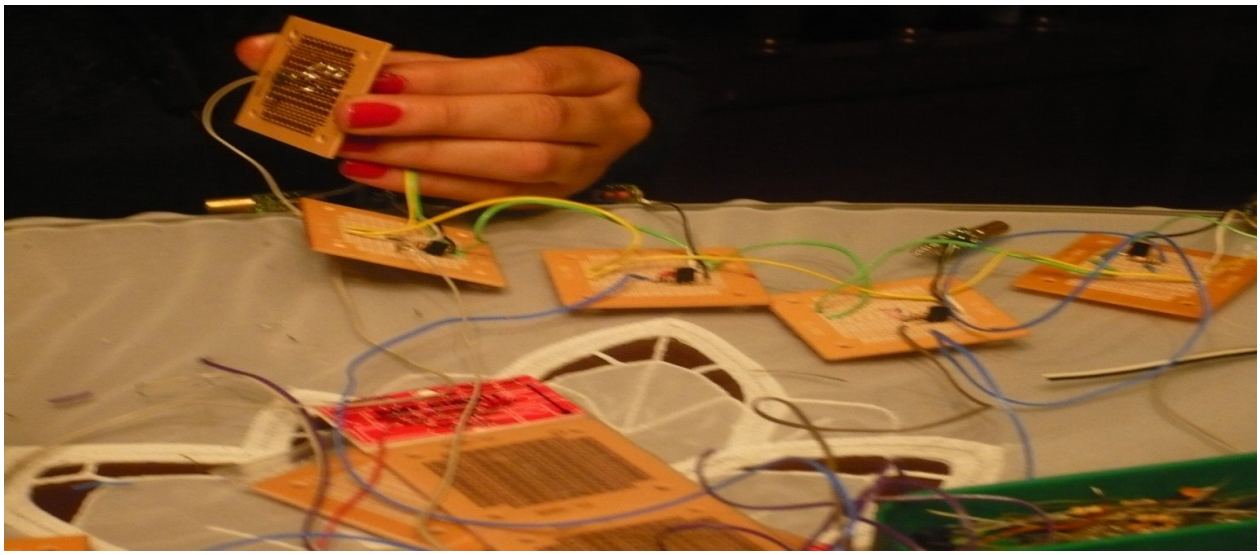


Figura 34. Construcción de circuitos para cada laser2

4.3.1.2 circuito dirigido a la fotorresistencia

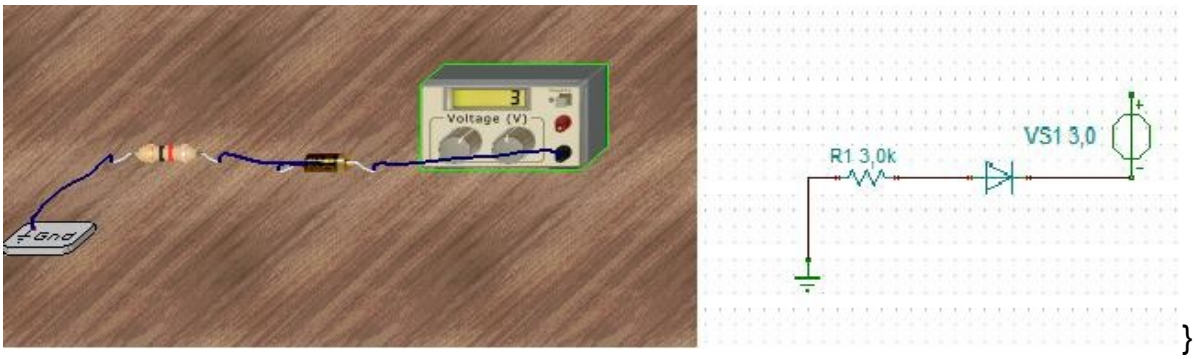


Figura.35. Circuito Fotorresistencia

Este circuito de la fotorresistencia es el encargado de recibir los mensajes de activación o desactivación significando si la señal del laser se encuentra interrumpida, este consta de dos componentes:

1. Resistencia de 3K
2. Fotorresistencia (representada como un foto diodo). Dispuesto como se presenta en la figura 35.

Para lograr encontrar el componente adecuado fue necesario realizar las siguientes pruebas:

a. La primera prueba consistió en colocar un voltaje de entrada y encontrar el valor de la resistencia exacta que deje pasar el 75% del voltaje de entrada pasando por la fotorresistencia.

b. La siguiente prueba fue con el objetivo de probar el anterior circuito, con la diferencia de que el haz de luz apuntara a la fotorresistencia y tratar de saturarla aún más con el láser, con el fin de que no existieran problemas de luz y así determinar cuánto voltaje resulta de salida y determinando los valores para que la TOWER reconociera el valor de voltaje 1 y 0, 1 para prendido y o para apagado.

Como resultado al circuito se le coloco un voltaje continuo de 3.3V y al medir el voltaje en la fotorresistencia se obtuvieron los siguientes resultados, que se muestran en la siguiente tabla.

Resistencia	Voltaje Resistencia	Voltaje foto diodo	Láser Resistencia	Láser fotodiodo
33 K	3.304 V	0.070 V	3260 V	0.057 V
180 K	3.219 V	0.093 V	3230 V	0.072 V
5 K	2.804 V	0.409 V	3030 V	0.332 V
2.2 K	2.283 V	0.797 V	2561 V	0.690 V
3 K	2.4 V	0.695 V	3.2 V	0.135 V

Tabla 5. Datos de pruebas

En la tabla número 6 según los resultados, se obtuvieron las variaciones que con laser y sin el laser en la fotorresistencia la relación de voltaje era de 0.9 V y 0.2 V para que la TOWER sea reconocida para su activación.

4.4 IMPLEMENTACIÓN ELECTRÓNICA DE LA PEDALERA COMO OCTAVADOR

.

Para la implementación de la pedalera como octavador, se hizo necesario agregar un switch adaptado en circuito con la función del que al ser presionado este subiera la escala musical del arpa una octava arriba. Ver figura las siguientes figuras.

Figura 36.Switch Implementado

Figura 37. Pedal incorporado como octavador.

.

4.5 PROGRAMACIÓN DEL SAMPLER PARA LA REPRODUCCIÓN DE LAS MUESTRAS

A continuación se explica el desarrollo del SAMPLER, que consta de dos fases, la primera permite detallar la interfaz gráfica que se desarrollo por medio de MAX/MSP y la segunda fase que posibilita dar cuenta de la interfaz grafica por medio del programa de visual C# 2008 Express Edition.

4.5.1 Implementación del SAMPLER desarrollado para reproducir audio e imágenes, empleando el programa MAX/MSP. El diseño contempla algunas de las características principales de un sampler comercial básico; como asignar muestras a regiones de cuerdas, ubicar de manera específica una función en el instrumento virtual para cada muestra disponible, reproducir muestras con la transposición adecuada dependiendo de qué cuerda es ejecutada y hacer asignaciones polifónicas de varias voces, entre otros.

4.5.1.1 Elementos para implementación del SAMPLER. En esta sección se muestra el diseño de un patch, usando el software de programación grafica Max/MSP, para reproducir las muestras del arpa previamente grabadas de acuerdo a la información MIDI proveniente del dispositivo electrónico.

1. En el patch se uso el objeto groove~ para reproducir muestras del arpa a diferentes velocidades, con la posibilidad que en algunos casos sean reproducidas en modo loop. En este caso se usaron cuatro copias de un subpatch llamado samplervoice~ para soportar una polifonía de cuatro voces, también se utilizó el objeto poly para asignar un número de voz a cada nota MIDI, que permite abrir el outlet de manera correcta el objeto gate y llegue al subpatch asignado. Y el objeto route para enviar información de nota al correspondiente subpatch samplervoice~. Ver figura 38.

.

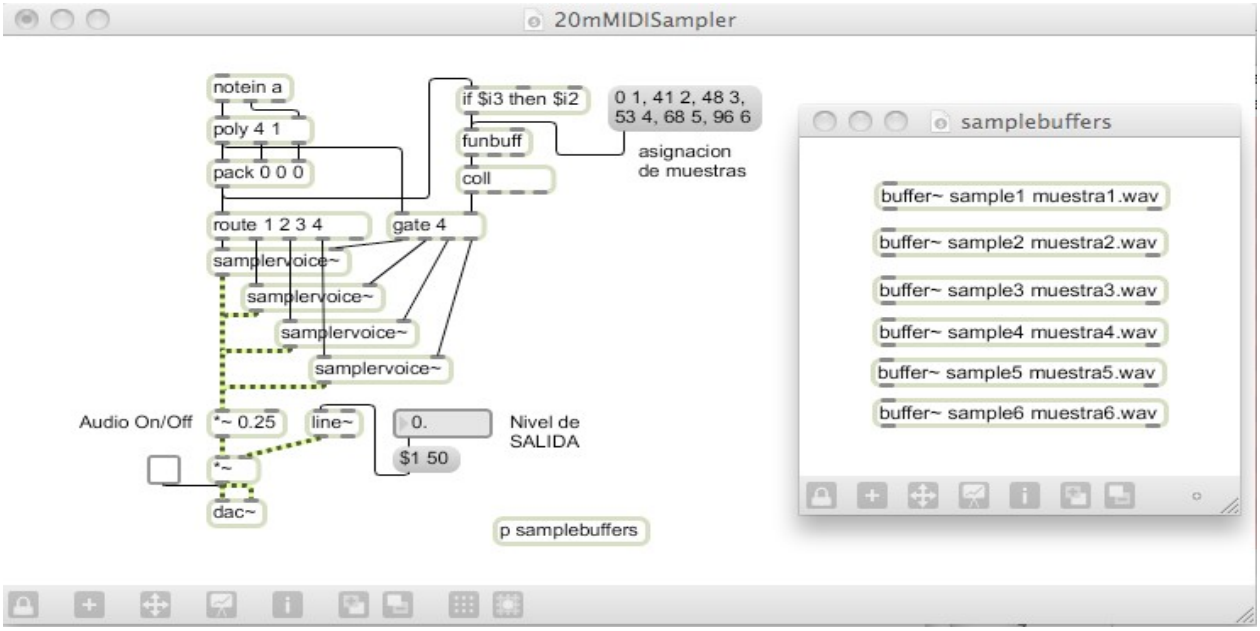


Figura 38. Figura del patch

2. para la reproducción de las muestras grabadas de manera correcta fue necesario tener en cuenta:

- Que las muestras debían ser leídas en memoria con un objeto buffer~, guardando en memoria una lista de sus respectivos nombres.
- Cada muestra debe ser asignada a una región de cuerdas y debe guardarse una lista de dichas asignaciones.
- Cada vez que una nota MIDI es recibida, debe enrutarse a un subpatch samplervoice~ para que el objeto groove~ en este subpatch reproduzca la muestra leyendo el buffer~ donde esta se encuentra. Ver figura 39.

.

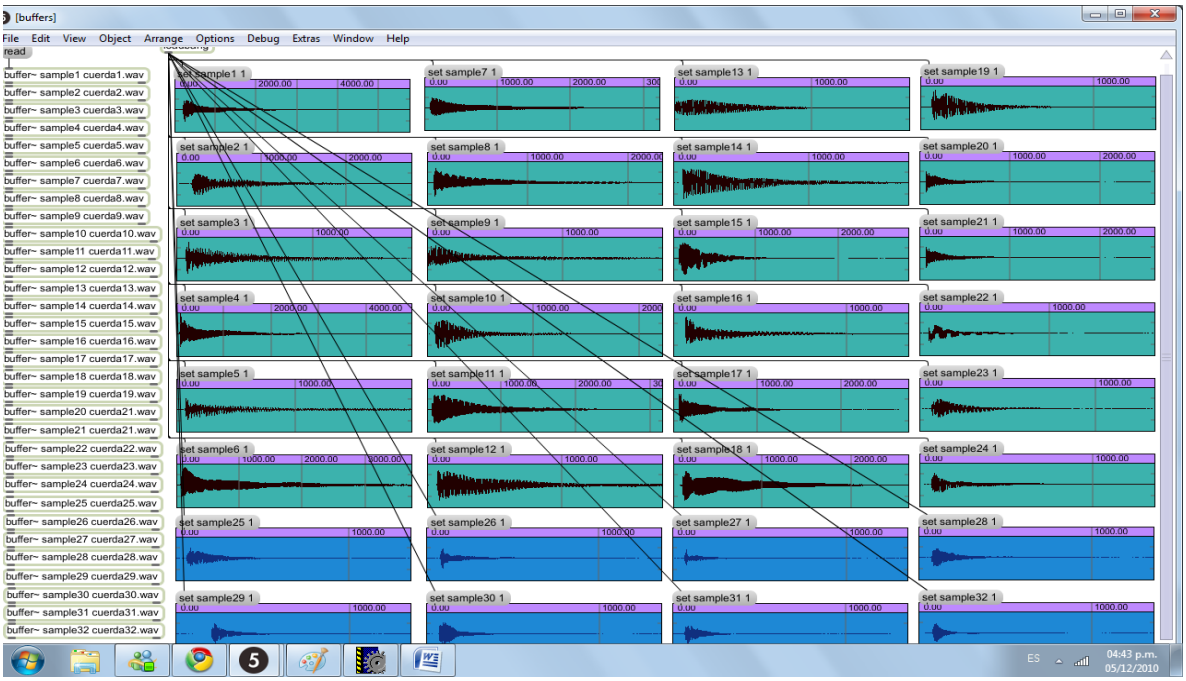
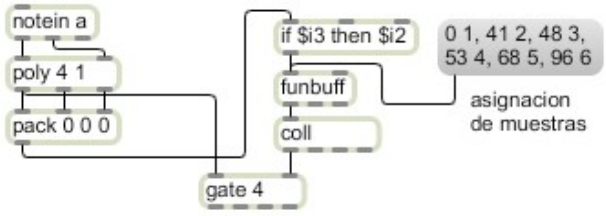


Figura 39. Subpatch samplebuffers con varios objetos tipo buffer~ para cargar cada muestra por separado.

3. Un message con nombre “asignación de muestras” que almacena un conjunto de regiones numeradas en el objeto funbuff. Este mensaje de note-on es recibido y leído por funbuff, que reporta el número de la región de nota para la cuerda interpretada. Dicho número de región es usado para buscar información vital en el objeto coll. Ver figura 46.



.

Figura 40. Número de la nota usado en funbuff para reproducción de la muestra

El número de región se uso para indexar la información en coll. Donde cada región se asigno la nota base, nombre del buffer~, tiempo de inicio del loop, tiempo de fin del loop y flag de loop on/off.

4. Un subpatch samplervoice~ que implementa un ¨vozsampler¨ utilizado para reproducir la voz MID. En la figura 4, se muestra como la información que proviene de coll es desempacada y enviada a los lugares apropiados para preparar el objeto groove~ para la nota que está a punto de ser reproducida. Esto le permite a groove~ leer el buffer~, los tiempos de reproducción de loop que deben ser utilizados, además de constatar si está activado o no para su reproducción.
Cuando la información de la nota llega al inlet izquierdo, la velocidad es usada para enviar un valor de amplitud al objeto *~ y el número de nota ¨note-on¨ es usado para calcular la velocidad apropiada de reproducción y para indicarle a groove~ que comience la reproducción desde el tiempo 0.

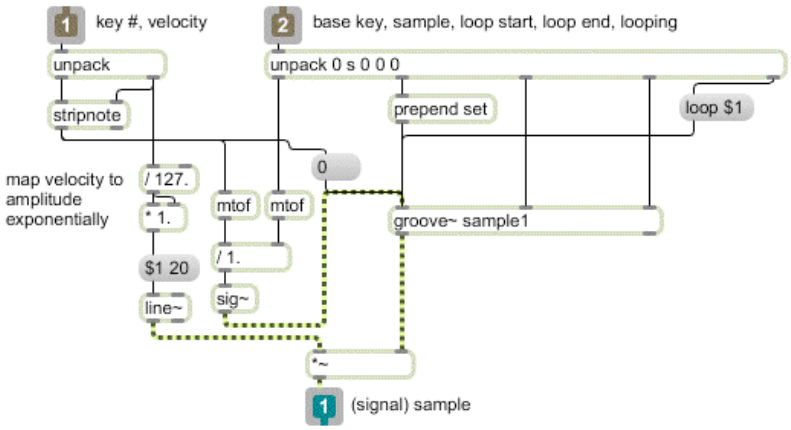


Figura 41. Subpatch samplervoice~.

.

4.5.2 Implementación del SAMPLER desarrollado para reproducir audio y video, empleando el programa visual C# 2008 Express Edition. Como su nombre lo indica es una programación en C, y tiene una plataforma practica para el usuario, en la Figura 42 se muestra la interfaz gráfica, donde se visualiza el puerto que despliega

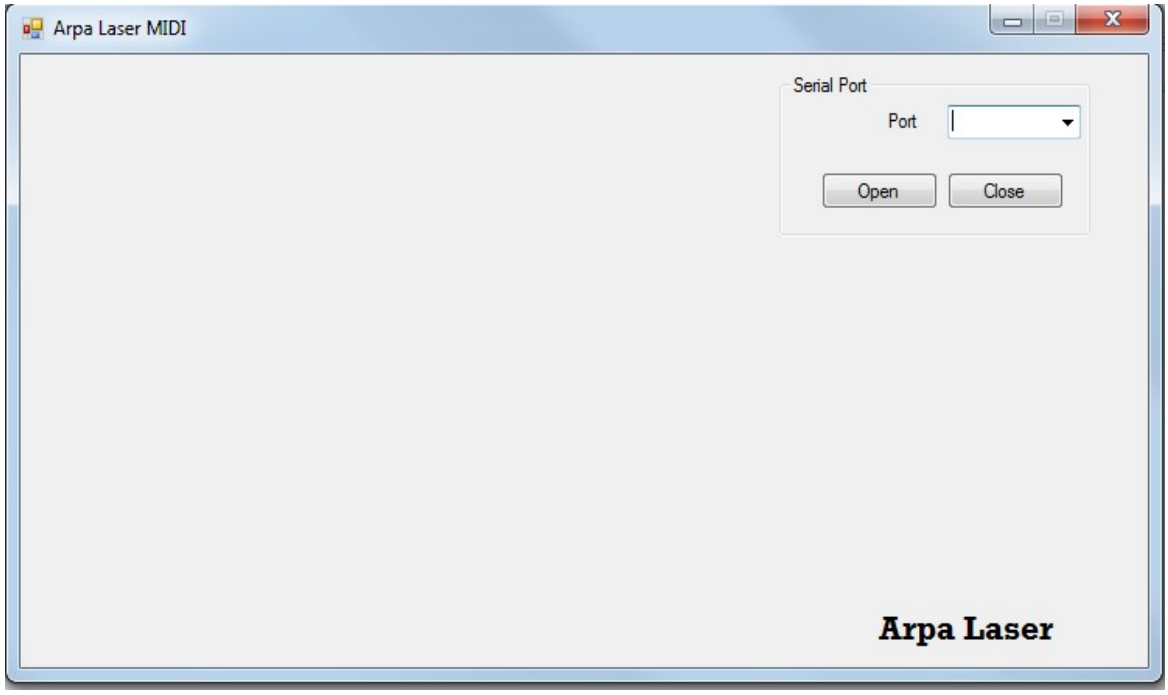


Figura 42.Interfaz Gráfica ARPA LASER OFF

una lista que permite escoger entre varios puertos como: MIDI, RS232, USB; que posee la TOWER, los cuales son punto de envío y recepción de datos, datos que son controladores del prototipo del arpa laser. Además está programado para cargar el banco de muestras e imágenes que tiene la función de reproducirlas cuando se utiliza el puerto serial RS232. Ver Figura 43

.

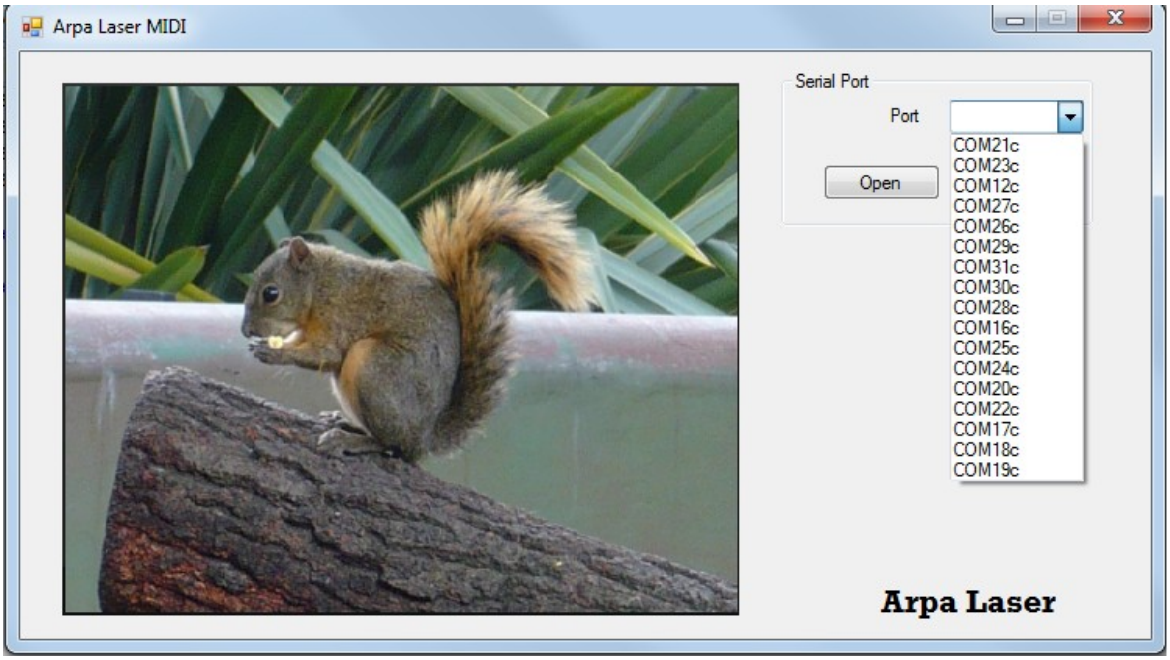


Figura 43. Interfaz Gráfica de ARPA LASER ON

- **Programación en Visual C# express 2008.** A continuación se presenta el código en C, utilizado para la interfaz grafica del arpa laser.

Ingreso de Bibliotecas

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
```


.

```
using System.Threading;
using System.Media;
namespace ArpaLaserMidi
{
 public partial class Main : Form
 {
 const string path = "";
 const string configNameFile = "config.ini";
 ConfigManager configManager;
 SerialPortManager serialPortManager;
 PacketManager packetManager;
 Thread hLecturaPuertos;
 private delegate void lecturasPuertosDelegate();
 public Main()
 {
 InitializeComponent();
 this.configManager = new ConfigManager(path + configNameFile);
 this.hLecturaPuertos = new Thread(new ThreadStart(this.lecturaPuertosHilo));
 this.hLecturaPuertos.Start();
 this.packetManager = new PacketManager(this.pictureBox1);
 }
 private void lecturaPuertos()
 {
 string[] portNames = System.IO.Ports.SerialPort.GetPortNames();
 this.comboBoxSerialPort.Items.Clear();
 for (int i = 0; i < portNames.Length; i++)
 {
 this.comboBoxSerialPort.Items.Add(portNames[i]);
 }
```

```
 {
 while (true)
 {
 Thread.Sleep(1000);
this.comboBoxSerialPort.Invoke(newlecturasPuertosDelegate(lecturaPuertos),new
object[] { });
 }
 }
private void cerrarHilos()
{
if (hLecturaPuertos != null)
if (hLecturaPuertos.IsAlive)
hLecturaPuertos.Abort();
}
private void Main_FormClosing(object sender, FormClosingEventArgs e)
{
this.cerrarHilos();
}
private void buttonOpen_Click(object sender, EventArgs e)
{
this.configManager.ConfigBean.PortName = this.comboBoxSerialPort.Text;
this.serialPortManager = new
SerialPortManager(this.configManager.ConfigBean.PortName);
this.serialPortManager.NewPacket += new
EventHandler(this.packetManager.PacketReceived_Event);
this.serialPortManager.OpenPort();
}
private void buttonClose_Click(object sender, EventArgs e)
{
```


.

```
this.serialPortManager.ClosePort();
}
U
}
}
```

4.5.3 PROGRAMACIÓN DEL ARPA LASER

Para la programación del arpa laser fue necesario usar la programación en C, ya que dada sus características especificas permite desarrollar una gran variedad de proyectos adecuándose a las necesidades particulares del mismo. En esta parte de la programación realizada para el desarrollo del ***ARPA LASER*** se tuvo en cuenta las notas, los bytes de activación y desactivación y las muestras grabadas del instrumento real, mencionándolos a continuación.

4.5.3.1 ***uint16_t value1 = 0; uint8_t sucCuerda1_ON;*** estas funcionan como variables enteras de tipo de Processor Expert generan archivos (PE_Types.h) que contienen marcas utilizadas con un registro de acceso periférico.

4.5.3.2 rs232_SendBlock; Esta orden envía la información obtenida durante todo el programa, a uno de los Puertos RS232, el cual uno de ellos será convertido en cable MIDI.

4.5.3.3 Códigos de activación Y desactivación de cuerda

```
void main(void)
{
uint8_t sucCuerda1_ON [] = { 0x90, 0x00 };
uint8_t sucCuerda1_OFF [] = { 0x91, 0x00 };
```


.

```
uint8_t sucCuerda2_ON [] = { 0x91, 0x00 };
uint8_t sucCuerda2_OFF [] = { 0x91, 0x00 };
```

4.5.3.4 Código de activación y desactivación de octavador

```
// Código cuerda 1 octavada
uint8_t sucCuerda1Oct_ON [] = { 0x90, 0x00 };
uint8_t sucCuerda1Oct_OFF [] = { 0x91, 0x00 };
// Código cuerda 2 octavada
uint8_t sucCuerda2Oct_ON [] = { 0x90, 0x00 };
uint8_t sucCuerda2Oct_OFF [] = { 0x91, 0x00 };
```

4.5.3.5 Muestreo de cada cuerda

```
for(;;)
{
if(upDateCuerda1())
{
rs232_SendBlock(sucCuerda1_ON,sizeof(sucCuerda1_ON),&Snd);
}
else
{
rs232_SendBlock(sucCuerda1_OFF,sizeof(sucCuerda1_OFF),&Snd);
}
if(upDateCuerda2())
{
rs232_SendBlock(sucCuerda2_ON,sizeof(sucCuerda2_ON),&Snd);
}
else
```


.

```
{
rs232_SendBlock(sucCuerda2_OFF,sizeof(sucCuerda2_OFF),&Snd);
}
} }
}
bool upDateCuerda1()
{
uint16_t value1 = 0;
uint16_t value2 = 0;
static uint16_t deltaValues = 0;
 //se realiza lo siguiente para que se active el laser
laserCuerda1_SetVal();
Cpu_Delay100US(100);
//se mide el valor de voltaje a través del conversor A/D
 SensorCuerda1_Measure(TRUE);
 SensorCuerda1_GetValue16(&value1);
   //apagado de laser
  laserCuerda1_ClrVal();
  Cpu_Delay100US(100);
//se mide de nuevo el valor de voltaje a través del conversor A/D
 SensorCuerda1_Measure(TRUE);
 SensorCuerda1_GetValue16(&value2);
//se obtiene el valor absoluto de la diferencia
 if(value2 > value1)
 {
  deltaValues = (uint16_t)(value2 - value1);
  }
  else
  {
```

```
 deltaValues = (uint16_t)(value1 - value2);
 }
 //se filtra a través de un sistema iir
 filter_iir(deltaValues,&deltaValues,4);
// Si la diferencia es menor a un umbral predefinido la cuerda     esta activa
 if(deltaValues > iuThActivation)
 {
// Si se supera el umbral expresa que no está activa la cuerda
return FALSE;
}
else
{
// Si no se supera el umbral quiere decir que se ha activado la cuerda, es decir, que los valores con presencia y ausencia de laser son los mismos por lo tanto el laser fue interrumpido
return TRUE;
}
}
```

Por otra parte fue necesario codificar cada una de las cuerdas grabadas, cada una de las muestras tiene su respectivo código MIDI de activación y desactivación de nota, en la siguiente tabla se muestra el orden y el código de cada una de las notas, así mismo se muestra la frecuencia, la octava en la que se encuentra y la nota musical.

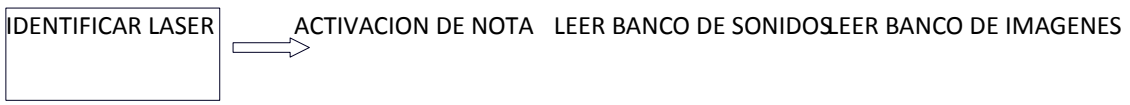
Tabla 6. Notas MIDI y códigos de activación de nota

.

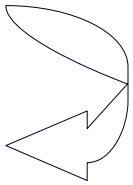
Octava MIDI	Nombre de la Octava	Número Nota MIDI	Nota musical	Cuerdas	Código de activación								Número Nota Piano	Frecuencia (Hz)
3	octava chica	48	C	CUERDA_1	0	0	1	1	0	0	0	0	28	130.813
3	octava chica	49		CUERDA_2	0	0	1	1	0	0	0	1	29	138.591
3	octava chica	50	D	CUERDA_3	0	0	1	1	0	0	1	0	30	146.832
3	octava chica	51		CUERDA_4	0	0	1	1	0	0	1	1	31	155.563
3	octava chica	52	E	CUERDA_5	0	0	1	1	0	1	0	0	32	164.814
3	octava chica	53	F	CUERDA_6	0	0	1	1	0	1	0	1	33	174.614
3	octava chica	54		CUERDA_7	0	0	1	1	0	1	1	0	34	184.997
3	octava chica	55	G	CUERDA_8	0	0	1	1	0	1	1	1	35	195.998
3	octava chica	56		CUERDA_9	0	0	1	1	1	0	0	0	36	207.652
3	octava chica	57	A	CUERDA_10	0	0	1	1	1	0	0	1	37	220
3	octava chica	58		CUERDA_11	0	0	1	1	1	0	1	0	38	233.082
3	octava chica	59	B	CUERDA_12	0	0	1	1	1	0	1	1	39	246.942
4	primera línea	60	C	CUERDA_13	0	0	1	1	1	1	0	0	40	261.626
4	primera línea	61		CUERDA_14	0	0	1	1	1	1	0	1	41	277.183
4	primera línea	62	D	CUERDA_15	0	0	1	1	1	1	1	0	42	293.665
4	primera línea	63		CUERDA_16	0	0	1	1	1	1	1	1	43	311.127
4	primera línea	64	E	CUERDA_17	0	1	0	0	0	0	0	0	44	329.628
4	primera línea	65	F	CUERDA_18	0	1	0	0	0	0	0	1	45	349.228
4	primera línea	66		CUERDA_19	0	1	0	0	0	0	1	0	46	369.994
4	primera línea	67	G	CUERDA_20	0	1	0	0	0	0	1	1	47	391.995
4	primera línea	68		CUERDA_21	0	1	0	0	0	1	0	0	48	415.305
4	primera línea	69	A	CUERDA_22	0	1	0	0	0	1	0	1	49	440
4	primera línea	70		CUERDA_23	0	1	0	0	0	1	1	0	50	466.164
4	primera línea	71	B	CUERDA_24	0	1	0	0	0	1	1	1	51	493.883
5	segunda línea	72	C	CUERDA_25	0	1	0	0	1	0	0	0	52	523.251
5	segunda línea	73		CUERDA_26	0	1	0	0	1	0	0	1	53	554.365
5	segunda línea	74	D	CUERDA_27	0	1	0	0	1	0	1	0	54	587.330
5	segunda línea	75		CUERDA_28	0	1	0	0	1	0	1	1	55	622.254
5	segunda línea	76	E	CUERDA_29	0	1	0	0	1	1	0	0	56	659.255
5	segunda línea	77	F	CUERDA_30	0	1	0	0	1	1	0	1	57	698.456
5	segunda línea	78		CUERDA_31	0	1	0	0	1	1	1	0	58	739.989
5	segunda línea	79	G	CUERDA_32	0	1	0	0	1	1	1	1	59	783.991

4.5.3.6 Programación del arpa laser

Este aparte es la base de desarrollo de todo el proyecto, para cumplir el objetivo de explicar y sustentar el proceso es necesario dividir en seis etapas graficadas en el siguiente diagrama



.



LASER VUELVE AL ESTADO NATURAL REPRODUCIR AUDIO E IMAGEN

Figura 44. Diagrama de bloques de la Programación ARPA LASER

1. Identificación del Laser. Consiste en el proceso mediante el cual la fotorresistencia detecta el laser, leyendo así las entradas analógicas que suplen las cuerdas del arpa tradicional. La importancia de este paso radica en el control que se ejerce sobre los datos; en este caso las muestras de audio o imágenes no se reproducirán al azar, sino, conforme a las órdenes que el prototipo reciba.

2. Activación de nota. Cuando la fotorresistencia ha identificado el laser, a través del puerto serial rs232 o puerto MIDI se envían códigos los cuales intervendrán en la variación del voltaje respecto al mismo mientras el laser es interrumpido; ésta variación del voltaje está asociada al lenguaje MIDI por lo que se envía información al SAMPLER para la posterior lectura de las muestras.

3. Lectura del banco de sonidos. Cuando la nota es activada, la información que está asociada a cada laser en el SAMPLER, se hace la lectura correspondiente a la muestra asignada al laser identificado.

.

4. Lectura del banco de imágenes. Cuando la nota es activada, paralelo a la lectura del banco de sonidos se desarrolla un proceso que lee de forma aleatoria las imágenes como respuesta a la programación asignada a cada laser, las cuales están programadas a cada uno –laser- en su totalidad.

5. Reproducción de audio e imágenes. Posterior a la lectura de imágenes y de sonidos y dada la decodificación de la información contenida en el SAMPLER que está asignada al laser activado, se ejecuta el sonido correspondiente y una de las imágenes programadas.

6. El laser vuelve al estado natural. Al terminar la acción que identifica el laser, que es la interrupción del mismo, la fotorresistencia recibe constantemente el haz de luz lo cual produce la desactivación de nota. El proceso descrito anteriormente se repetirá con cada manipulación del prototipo.

4.6 ENSAMBLAJE DE LA PARTE ELECTRÓNICA EN LA ESTRUCTURA DEL ARPA

Para el ensamblaje de los elementos electrónicos en la estructura del arpa se tuvo en cuenta el diámetro del laser con las perforaciones del arpa, la asignación y conexión de cada cable de datos del circuito de la fotorresistencia a un puerto análogo de la TOWER y la conexión y ruteo del controlador MIDI del arpa laser con su respectivo SAMPLER.

4.6.1 Diámetro del laser con las perforaciones del arpa. Se considera que es importante el diámetro del laser y las perforaciones ya que era necesario que coincidan estos y estén alineados para la recepción del laser en cada fotorresistencia**.** Ver Figura 45 y 46

.

Figura 45 Prueba perforaciones Arpa laser

Figura 46. Prueba Arpa laser orificios coincidentes

4.6.2 Asignación y conexión de cada cable de datos del circuito de la fotorresistencia a un puerto análogo de la TOWER. Para asignar los puertos de la

.

TOWER se observo el DATA SHIP, con el fin de conocer la disposición de los puertos análogos disponibles y asignarlos dependiendo de la escala en la que se encuentra afinado el instrumento real. La conexión de cada puerto análogo y asignación de cuerda se muestra en la tabla siguiente.

Tabla 7 : Asignación de cada cuerda según la afinacion y su octava.

CUERDAS	NOTA	PUERTOS	NOTA OCTAVADA	CHANNEL 0	FOTORESISTENCIAS	PUERTOS	CHANNEL
CUERDA 1	D3	PTH6/B40/PWN4	BB	0	FOTORESISTENCIA 1	PTC4/B46/APD11/J3(2)	69
CUERDA 2	BC	PTF7/A67	A4	1	FOTORESISTENCIA 2	PTE1/B10/ADP1	68
CUERDA 3	BE	PTH7/B39/PWM5	A6	2	FOTORESISTENCIA 3	PTE0/B11/ADP2	67
CUERDA 4	C0	PTF6/A66	A8	3	FOTORESISTENCIA 4	PTD7/B7/ADP3	66
CUERDA 5	C1	PTH3/A37/PWM3	A9	4	FOTORESISTENCIA 5	PCT6/B44/ADP9	52
CUERDA 6	C3	PTE5/A38/PWM2	AB	5	FOTORESISTENCIA 6	PTC7/B48/ADP8	51
CUERDA 7	C5	PTH4	AD	6	FOTORESISTENCIA 7	PTD1/A41/ADP6	50
CUERDA 8	C7	PTE3/A40/PWMO	AF	7	FOTORESISTENCIA 8	PTD0/A42/ADP7	49
CUERDA 9	B0	PTH5	80	8	FOTORESISTENCIA 9	PTE2/B9/ADP0	45
CUERDA 10	B2	PTG0	82	9	FOTORESISTENCIA 10	PTC5/B45/APD10	46

FOTORESISTENCIA 1	PTC4/B46/APD11/J3(2)	T W R M CF 51 CN 12 8C LK	PTH6/B40/PWN4	CUERDA 1
FOTORESISTENCIA 2	PTE1/B10/ADP1		PTF7/A67	CUERDA 2
FOTORESISTENCIA 3	PTE0/B11/ADP2		PTH7/B39/PWM5	CUERDA 3
FOTORESISTENCIA 4	PTD7/B7/ADP3		PTF6/A66	CUERDA 4
FOTORESISTENCIA 5	PCT6/B44/ADP9		PTH3/A37/PWM3	CUERDA 5
FOTORESISTENCIA 6	PTC7/B48/ADP8		PTE5/A38/PWM2	CUERDA 6
FOTORESISTENCIA 7	PTD1/A41/ADP6		PTH4	CUERDA 7
FOTORESISTENCIA 8	PTD0/A42/ADP7		PTE3/A40/PWMO	CUERDA 8
FOTORESISTENCIA 9	PTE2/B9/ADP0		PTH5	CUERDA 9
FOTORESISTENCIA 10	PTC5/B45/APD10		PTG0	CUERDA 10
			PH5/PWU7	PEDAL
GND				VCC

4.6.3 Conexión y ruteo del controlador MIDI del arpa laser con su respectivo SAMPLER. Para este ensamblaje se dispone de dos opciones para la conexión y ruteo del controlador MIDI del arpa laser.

.

4.6.3.1 Conexión y ruteo por el puerto Serial. Esta primera opción se posibilita conectando el controlador MIDI, al puerto serial del computador, el cual envía datos al SAMPLER. Ver figura 47.

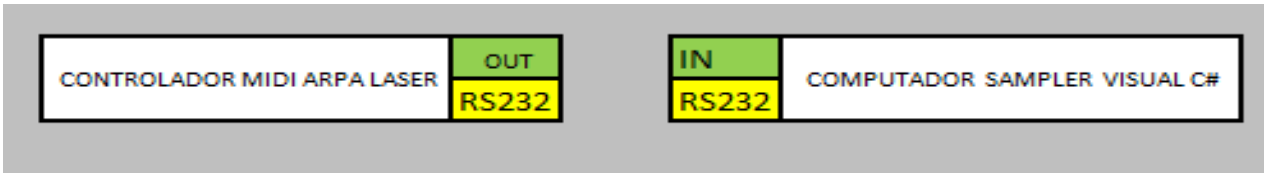


Figura 47. Ruteo controlador arpa laser por puerto serial.

4.6.3.2 Conexión y ruteo por el puerto MIDI. Esta segunda opción se conecta de la salida del puesto MIDI a la interface y pasa al computador a través de USB, enviando la información al SAMPLER asignado. Ver figura 48.

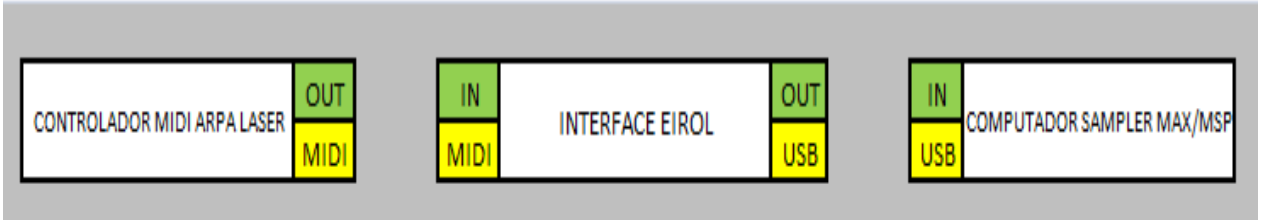


Figura 48. Ruteo controlador arpa laser por puerto MIDI.

4.7 **PROBAR EL DESEMPEÑO DEL ARPA Y SU FUNCIONAMIENTO**

Teniendo en cuenta que no existe un instrumento musical o similar para realizar una comparación y poder describir el desempeño. El funcionamiento del controlador MIDI del arpa laser se comprobó por medio del cumplimiento de cada objetivo planteado en un comienzo.

El controlador MIDI de un arpa laser funciona de dos formas:

.

4.7.1 Funcionamiento a través del puerto serial. Teniendo en cuanta el controlador MIDI del arpa a continuación se muestra los pasos a seguir para el montaje y funcionamiento por medio del puerto serial.

1. Se conecta la fuente de poder del controlador MIDI del arpa laser a la corriente de 110 voltios AC.
2. Se activa el puente (Jumper) en la TOWER para elegir el puerto serial.
3. Se conecta el cable del controlador MIDI al puerto serial del computador.
4. Se ejecuta el programa y se abre la interfaz gráfica (el SAMPLER) Arpa laser MIDI, donde se despliega todos los puertos seriales, de los cuales se debe seleccionar el COM 10C, que es el puerto activo.
5. Una vez seleccionado el puerto, se activa haciendo click en el botón **Open.**
6. Al interrumpir uno de los lásers del controlador MIDI del arpa laser, Este reproducirá la nota correspondiente al laser asignado y en la interfaz gráfica se visualizara una imagen que se reproduce simultáneamente con la muestra. Teniendo presente que existe la pedalera como octavadora y funciona en cualquier instante que se esté ejecutando el controlador.
7. Cada uno de los 11 láser tiene asignado una nota y la imagen es seleccionada de la carpeta de imágenes cargadas en el SAMPLER.

La prueba de funcionamiento está grabada en el video 1, donde se muestra el funcionamiento del arpa láser, y donde al cambiar la nota musical, también cambia el gráfico mostrado en la pantalla del computador.

4.7.2 Funcionamiento a través del puerto MIDI. Existe otra forma de conexión y funcionalidad del controlador MIDI del arpa laser, a través del puerto MIDI, a continuación se describe los pasos para el montaje y funcionamiento:

.

1. Se conecta la fuente de poder del controlador MIDI del arpa laser a la corriente de 110 voltios AC.
2. Asignar el jumper en la TOWER para elegir el puerto MIDI.
3. Conectar el puerto MIDI a la interface EIROL.
4. Conectar desde la EIROL al computador por medio del puerto USB.
5. Abrir el SAMPLER realizado por medio de **MAX/MSP**.
6. Al interrumpir uno de los lásers de controlador MIDI del arpa laser, Este reproducirá la nota correspondiente al laser asignado y en el SAMPLER se reproduce un video con una serie de opciones para manipular el video como pararlo, mostrar el video solo en componentes rojo, verde, azul o escala de gris y también la pedalera funciona como octavadora.

La prueba de funcionamiento está grabada en el video 2, donde se muestra el funcionamiento del arpa láser, y donde al cambiar la nota musical, este maneja algunos parámetros en el video que se reproduce en el SAMPLER MAX/MSP mostrado en la pantalla del computador.

5. PRESENTACIÓN Y ANÁLISIS DE RESULTADOS

.

5.1 ANÁLISIS DE LAS CLASES DE NIVELES DE VOLTAJE UTILIZADOS.

Los niveles de voltaje que se presentaron en desarrollo del prototipo fueron:

5.1.1 TTL. Su tensión de alimentación característica se halla comprendida entre los 4,75v y los 7,25V , los niveles lógicos vienen definidos por el rango de tensión comprendida entre 0,2V y 0,8V para el estado L (bajo) y los 2,4V y Vcc para el estado H (alto).

La velocidad de transmisión entre los estados lógicos es su mejor base, si bien esta característica le hace aumentar su consumo siendo su mayor enemigo. Motivo por el cual han aparecido diferentes versiones de TTL como FAST, LS, S, etc y últimamente los CMOS: HC, HCT y HCTLS. En algunos casos puede alcanzar poco más de los 250 MHz, las señales de salida TTL se degradan rápidamente si no se transmiten a través de circuitos adicionales de transmisión (no pueden viajar más de 2 m por cable sin graves pérdidas).

5.1.2 RS232. La oscilación que se presento en este caso fue de 0 a 5 V lo que certifica que al no haber intervalos de voltaje muy grandes, la posibilidad del que el ruido se filtre o ingrese a la señal es menor y las distancias que se pueden utilizar son mayores.

Analizando estos resultados obtenidos se puede concluir que el puerto serial se manejan niveles de voltajes más bajos pero el funcionamiento por cada uno de sus puertos es similar, aunque para el puerto serial se puede conectar un cable de mayor longitud sin que se filtre el ruido.

5.2 ANÁLISIS DEL MANEJO ANALÓGICO Y DIGITAL

.

En primera instancia se tomo digital para las 10 cuerdas pero al obtener como resultado que las fotorresistencias se saturaran se decide tomar los puertos análogos de la TOWER para así crear un umbral y que las fotorresistencias no se saturen o no se vean afectadas por cada cambio de intensidad de la luz.

5.3 ANÁLISIS ENTRE LOS DIFERENTES PUERTOS UTILIZADOS.

En el controlador MIDI del arpa laser existen dos puertos a utilizar, el primero es el puerto RS232 o serial y el segundo el puerto MIDI, a través de los cuales se puede realizar la conexión, pero para el puerto MIDI se debe conectar una interface que convierte el puerto MIDI a USB para poder realizar la conexión al computador.

5.4 ANÁLISIS DE PRUEBAS DE LA FOTORRESISTENCIA UBICADOS EN LA TABLA 6.

Una vez realizadas estas pruebas con distintos valores de resistencias (33K, 180K, 5K, 2,2K y 3K), se puede concluir que a mayor resistencia, menor es el paso de voltaje. Siendo la resistencia más baja 3K con la que se obtuvo mayor aproximación al voltaje requerido del 75%.

5.5 ANÁLISIS ENTRE LAS DISTANCIAS DE LOS DIFERENTES LASER.

Al realizar las primeras pruebas del controlador se tomo una distancia de 1 centímetro entre láser y láser, pero al tener como resultado que las fotorresistencias se veían afectadas por los haces de luz de los otros láser, se tomó la decisión de separar los láser cada 2 centímetros para que las fotorresistencias fueran saturadas únicamente por el láser asignado.

5.6 ANÁLISIS CON RESPECTO A UNA LIMITACIÓN PLANTEADA (VELOCITY).

.

Al tener la limitante de que el controlador MIDI del arpa no maneje el parámetro de velocity, el arpa mantendrá la reproducción de las muestras en el nivel que fue grabado solamente.

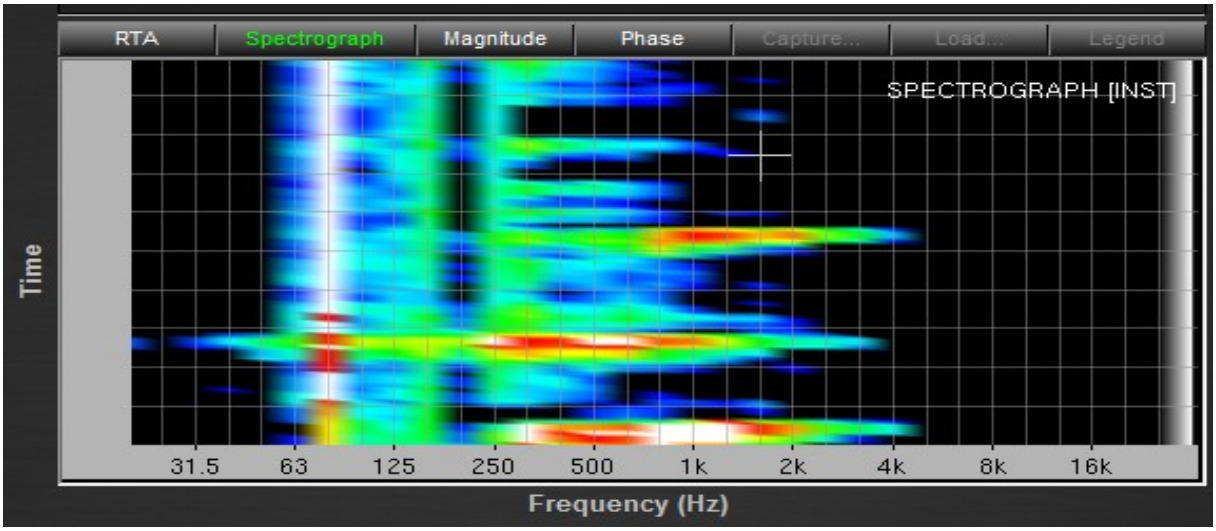
5.7 ANÁLISIS CON RESPECTO AL BANCO DE SONIDOS (GRABACIÓN Y MICROFONOS).

Las muestras recogidas para el banco de sonidos fue realizada por un Arpista profesional, el cual interpreto cada una de las notas musicales del arpa que fueron capturadas una a una y grabadas posteriormente en el banco de sonidos. Para determinar qué tipo de micrófonos se utilizó en la grabación del banco de sonidos se realizó un análisis espectral de las muestras tomadas con cada micrófono para determinar cual demostraba mejor captura con respecto a sus características que se muestran en las siguientes imágenes.

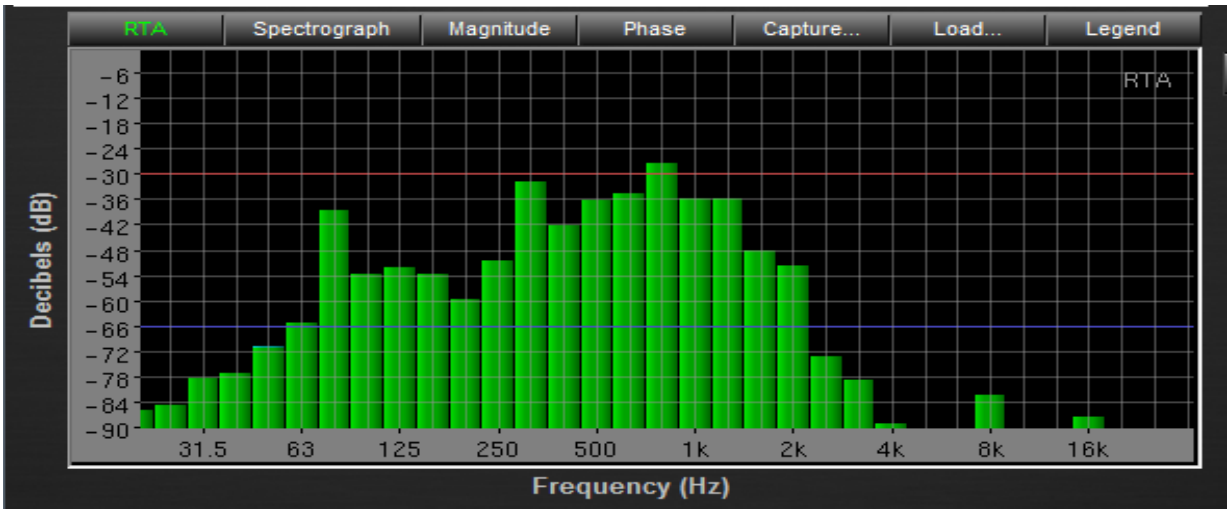
CUERDA 1 MICROFONO BETA 58

.

Imagen Espectro en frecuencia de una de las muestras tomadas con el BETA 58

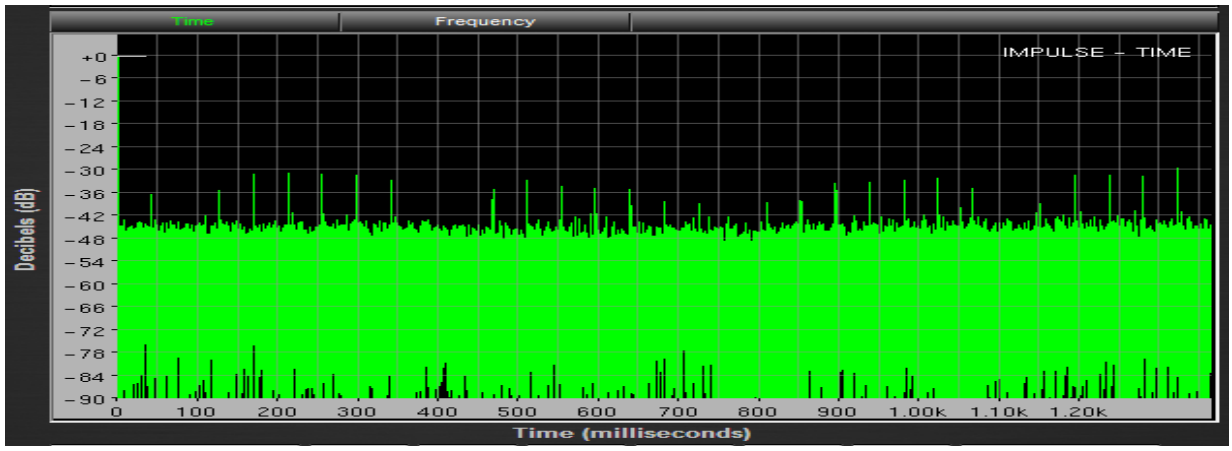


RTA de una de la muestra tomada con el BETA 58.



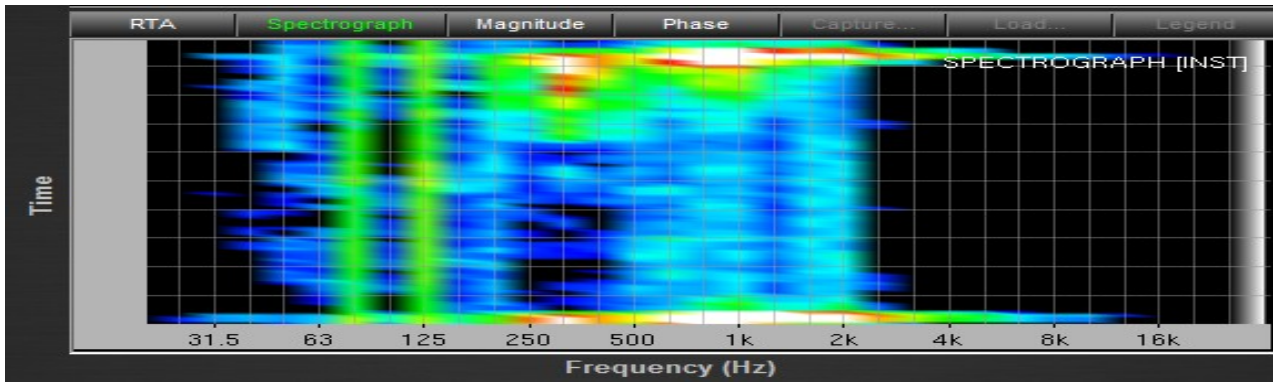
IMPULSE TIME de la muestra tomada con el BETA 58

.

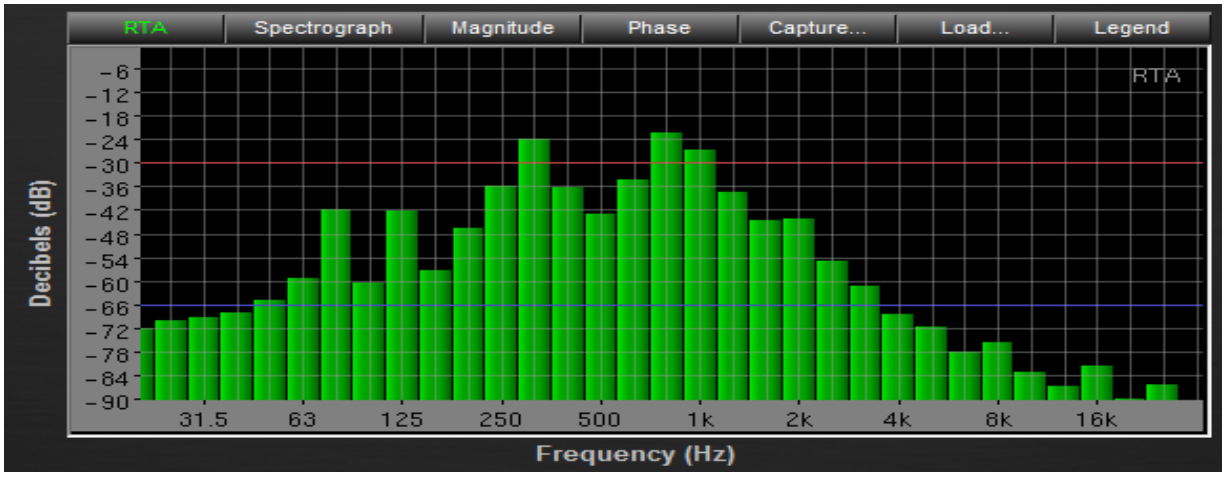


CUERDA 1 MICROFONO SHURE SM57

Imagen Espectro en frecuencia de una de las muestras tomadas con el SHURE SM 57.

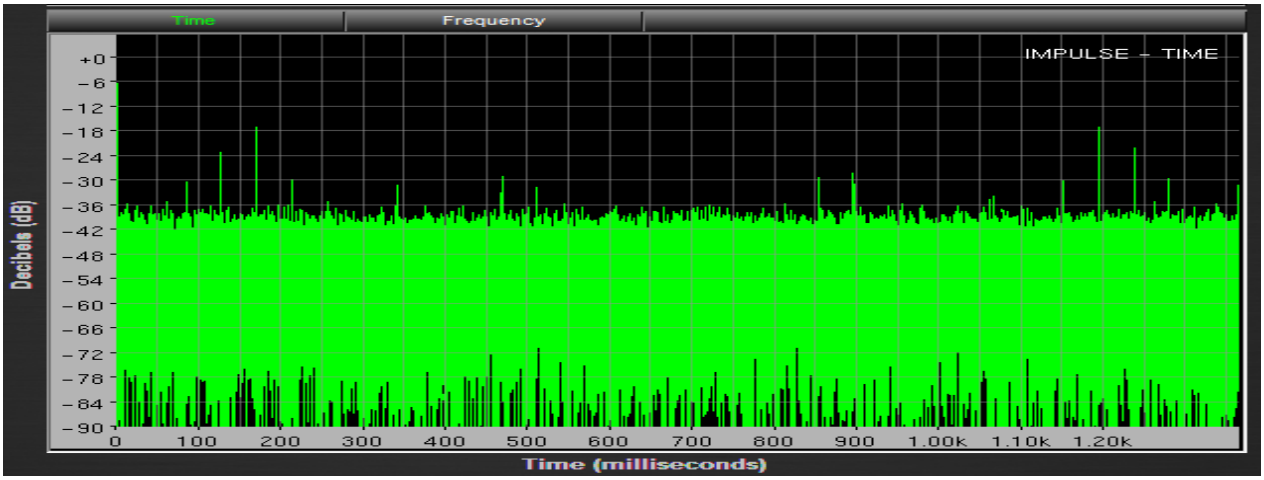


RTA de una de la muestra tomada con el SHURE SM 57.



.

IMPULSE TIME de la muestra tomada con el SHURE SM 57.



Entonces, este banco de sonidos puede ser más amplio cargando mas notas musicales de diferentes octavas ya que solamente se grabo de una sola para realizar las pruebas.

5.8 ANÁLISIS CON RESPECTO A LAS TÉCNICAS DE GRABACIÓN.

Luego de realizar las pruebas de grabación con la aplicación de diferentes técnicas tales como par espaciado, Pares coincidentes, pares casi-coincidentes y ORTF.

Se decide tomar la grabación con la técnica ORTF de las muestras por la sensación de volumen y profundidad que se obtienen en las notas más graves.

5.9 ANÁLISIS CON RESPECTO A LOS MICRÓFONOS.

Una vez realizada las pruebas sobre los diferentes tipos de micrófonos dinámicos, condensador, piezoeléctricos utilizados para la grabación de las notas musicales, se escogió los dinámicos por el rango de frecuencias y la dirección de captura que poseen.

.

.

5.10 ANÁLISIS DE LA ENCUESTA A.

Los datos que se muestran en las siguientes gráficas son los conocimientos que tienen las personas sobre nuevas tecnologías.

Los interrogantes que se han planteado son los siguientes:

¿SABE UD QUE ES MIDI?

Gráfica1.MIDI
Los datos que se exponen en la anterior gráfica muestran un 60% de afirmación, lo cual demuestra que más de la mitad tiene conocimientos sobre este tema.
¿SABE USTED QUE ES UN SAMPLER?

Gráfica 2. SAMPLER
El grafico anterior manifiesta que el 77% de las personas encuestadas no tienen conocimiento alguno sobre lo que es o como funciona un Sampler y que el porcentaje de las personas que conocen es muy bajo.

¿SABE USTED DE INSTRUMENTOS QUE UTILICEN LASER?

Gráfica 3.Instrumentos

La gráfica 3 muestra un porcentaje del 5%, lo cual significa que existen muy pocas personas que conozcan instrumentos o muy pocos instrumentos desarrollados con elementos electrónicos como láser. Por el contrario el 20% no tiene conocimiento alguno sobre el tema o no es de su interés.

5.11 ANÁLISIS DE LA ENCUESTA DE FUNCIONAMIENTO DEL CONTROLADOR MIDI DEL ARPA LÁSER QUE REPRODUCE AUDIO Y VIDEO..

Gráfica1.
El 80% de los encuestados al realizar el acercamiento a la interfaz grafica, pudieron notar que cumple con varias características de fácil aceptación y de fácil interacción.

Gráfica2

.

La gran mayoría de personas encuestadas, pudieron notar que las imágenes que se muestran en el programa son originales y no copias de bancos de gráficos.

Gráfica 3.

En este gráfico se muestra que el 80% de las personas a las cuales se les realizó la encuesta, le resulto muy práctico ver las diferentes alternativas de funcionamiento del controlador MIDI del arpa láser como lo son as dos conexiones MIDI y RS232.

Gráfica 4.

En esta parte de la encuesta la gente que respondió las preguntas, pudo certificar que el octavador o pedalera, realiza el cambio de octava de una manera y práctica.

Gráfica 5.

El 10% que se muestra en la gráfica significa que muy pocas personas erraron en este punto ya que la pedalera les brindo seguridad, y mostro su funcionamiento en el ensayo del arpa.

Gráfica 6.

Como se puede observar en este gráfico el 70% de las personas encuestadas supo identificar el número (20) de notas pregrabadas del controlador.

Gráfica 7.Fue bastante clara la aceptación y escucha de as muestras mediante el acercamiento al instrumento.

Gráfica 8.En esta gráfica se muestra el porcentaje de cada una de las opciones que brinda el controlador dando como 40%, a la opción como una herramienta para aprendizaje, a comparación de las demás opciones planteadas.

.

Gráfica 9. Es claro que el CONTROLADOR MIDI es un instrumento que vale la pena desarrollar creando y abriéndoles puertas a nuevos inventos para el aprendizaje a todas las personas no solo niños.

6. CONCLUSIONES

El Controlador MIDI de un ARPA LASER es un dispositivo electrónico capaz de reproducir con cada nota musical imágenes y videos aleatorios o programados y los botones de reproducción de los mismos. De igual forma se puede programar para ser útil en cualquier campo que implemente el protocolo MIDI, brindando así más opciones para controlar sonidos, luces entre otros.

La utilización de las fotorresistencias para detectar el haz de los laser los cuales funcionan como las cuerdas del arpa laser, requiere de un previo análisis ya que son muy sensibles a la luz; esto hace que se vea afectado por las condiciones de iluminación en los recintos. Por lo cual se podría probar otros elementos que cumplan con la misma función como sensores y fotodiodos

El controlador MIDI del arpa laser no solo permite reproducir muestras del arpa, si no también muestras de cualquier instrumento que se tengan pregrabadas.

.

La implementación del protocolo MIDI y el puerto serial RS232 en el dispositivo permite que este sea compatible con la mayoría o su totalidad de software y samplers que existen en el mercado y proporciona una herramienta de control enfocada a cualquier instrumento musical.

El diseño del controlador MIDI del arpa laser en ningún momento pretende remplazar el instrumento real, más bien, el prototipo fue diseñado para brindar nuevos espacios para la construcción de proyectos sobre el tema de controladores digitales capaces de emular instrumentos musicales reales que sean llamativos, novedosos y a su vez didácticos.

El costo del KIT del microcontrolador utilizado en el desarrollo del prototipo es económico teniendo en cuenta que brinda una variedad de opciones al ser reconfigurable y que se puede programar más de un dispositivo.

7. RECOMENDACIONES

.

Se recomienda realizar un análisis de los elementos electrónicos como fotorresistencias en la implementación del circuito ya que existen variaciones de voltajes que en algún momento la señal no es reconocida y puede llegar a afectar el funcionamiento del controlador MIDI. Puesto que el voltaje reconocido es cero o estado inactivo.

Para este tipo de proyectos se recomienda la utilización de una fuente de poder de bajo voltaje específicamente de 5 voltios que es la óptima para que se envíen los datos de forma adecuada a la fotorresistencia y a la TOWER. También es importante recordar que una fuente de mayor voltaje podría causar daños irreversibles en la TOWER y puede llegar a quemarla.

Para este proyecto se recomienda la utilización de los laser de 5 mili Voltios y con la misma intensidad para que envíen de forma adecuada los datos a las fotorresistencias ya que estos varían y no podrían no enviar los datos correctos de activación y desactivación de cada uno de los laser.

Es recomendable que se realice un banco de sonidos completo para el controlador MIDI del arpa laser para tener más posibilidades de utilizar como herramienta didáctica para música.

Se recomienda Implementar el parámetro del velocity para que al ejecutar el instrumento este pueda emular parámetros más complejos del instrumento real.

8. BIBLIOGRAFÍA

.

- M. Hermann, M. Norton , Basilik subscale laser experiments, 1992-1993.
- Sergi Jordà Puig, Audio Digital y MIDI capitulo 7 y 8, Guías Monográficas Anaya Multimedia, Madrid 1997.
- Tomas Pollan, Santamaría, Electrónica Digital . Universidad de Zaragoza.
- David Zicarelli,Basic Javascript programming.pdf 2000-2006.
- Institut de Recherche et Coördination Acoustique /Musique, Max/Msp Fundamentals.pdf, 2000-2006
- Sergi, Jordá Puig, Audio Digital y MIDI capitulo 7 y 8, guias monográficas Anaya Multimedia, Madrid 1997.
- http://laserharp.manuel-schulz.com/readarticle.php?article_id=3
- http://arauca.net/p4.html
- ProgramData/Processor%20Expert/CW08_PE3_07/DOCs/PEh4CA2.html
- Manuel Rejano de la Rosa. Ruido Industrial y Urbano, Pág. 63
- http://www.telefonica.net/web2/blasinski/microfonos/modelos.htm
- jit.qt.movie 320 240.MAX 1990-2008 cyclin 74/IRCAM all right reserved MSP 1997-2008 cyclin 74;portions based on Pd 1997-2008 the regents of the University of california.Pd and MSP arebased on Max/FTS, an advanced DSP platform IRCAM Jitter 2000-2008 cyclin 74
- Max 5 was created with the JUCE library 2005-2008 Raw Material software ltd.

ANEXOS

ANEXO A.

UNIVERSIDAD SAN BUENAVENTURA
BOGOTA D.C.

NOMBRE: ----------
FECHA: ----------
OCUPACION: ----------

POR FAVOR CONTESTE LAS SIGUIENTES PREGUNTAS

1. SABE USTED QUE ES MIDI?

A) SI B) NO SI SU RESPUESTA ES AFIRMATIVA PROSIGA LA ENCUESTA.

.

2. SABE QUE ES UN SAMPLER?
   A) SI B) NO
3. QUE TIPO DE SOFTWARE CONOCE Y/O UTILIZA

A. MAX/MSP
B. PURE DATA
C. SYNTHEDITH
D. SYNMAKER
E. REAKTOR
F. REASON

4. SABE DE INSTRUMENTOS MUSICALES QUE TENGAN LASER INCLUIDOS EN ELLOS?
   A) SI B) NO CUALES:

5. CREE USTED QUE LA CREACION DE ESTOS INTRUMENTOS FACILITARIAN LA INTERPRETACION MUSICAL?
   A) SI B) NO PORQUE

6. CREE USTED QUE ESTE INSTRUMENTO SE ASEMEJA AL INTRUMENTO REAL?
   A) SI B) NO PORQUE

7. PIENSA USTEDE QUE EL ARPA LASER ES VISUALMENTE LLAMATIVO?

   A) BUENO B) MUY BUENO C) EXCELENTE

8. CREE QUE LAS FUNCIONES QUE PRESTA EL ARPA LASER SON

APROPIADAS PARA ESTE INTRUMENTO?

A) SI B) NO

9. UTILIZARIA EL INSTRUMENTO EN EVENTOS?
10.

A) SI B) NO

ANEXO B

ENCUESTA DE FUNCIONAMIENTO DEL CONTROLADOR MIDI DEL ARPA LASER QUE REPRODUCE AUDIO Y VIDEO.

1. AL REALIZAR EL MONTAJE DEL CONTROLADOR MIDI DEL ARPA LASER QUE REPRUCE AUDIO Y VIDEO, EL SOFTWARE ACTIVA LOS SIGUIENTES PUERTOS.

    - COM 1C
    - COM 2C
    - COM 21C
    - COM 10
    - COM 31C
    - COM 22C
    - COM 16C
    - COM 24C

.

2. CUANDO ESTA ACTIVO EL PUERTO SE PUEDE INTERPRETAR LAS NOTAS PRE GRABADAS, MOSTRANDO EN LA PANTALLA LAS DISTINTAS IMÁGENES QUE HAN SIDO CARGADAS.

    a) SI
    b) NO

3. EL CONTROLADOR MIDI DEL ARPA LASER FUNCIONA CON LOS PUERTOS:
    - MIDI
    - SERIAL

4. AL UTILIZAR LA PEDALERA LAS NOTAS MUSICALES SUBEN UNA OCTAVA EN LA ESCALA.
    a) SI
    b) NO

5. AL SUBIR UNA OCTAVA MEDIANTE EL USO DE LA PEDALERA, CAMBIA LOS SONIDOS EMITIDOS POR EL CONTROLADOR MIDI DEL ARPA LASR.
    a) SI
    b) NO

6. EL NUMERO DE NOTAS MUSICALES PREGRABADAS EN EL CONTROLADOR MIDI DEL ARPA LASER ES:
    a) 11
    b) 33
    c) 22
    d) 44

7. CADA UNA DE LAS NOTAS PREGRABADAS CORRESPONDEN A UNA NOTA MUSICAL REAL.
    a) SI
    b) NO

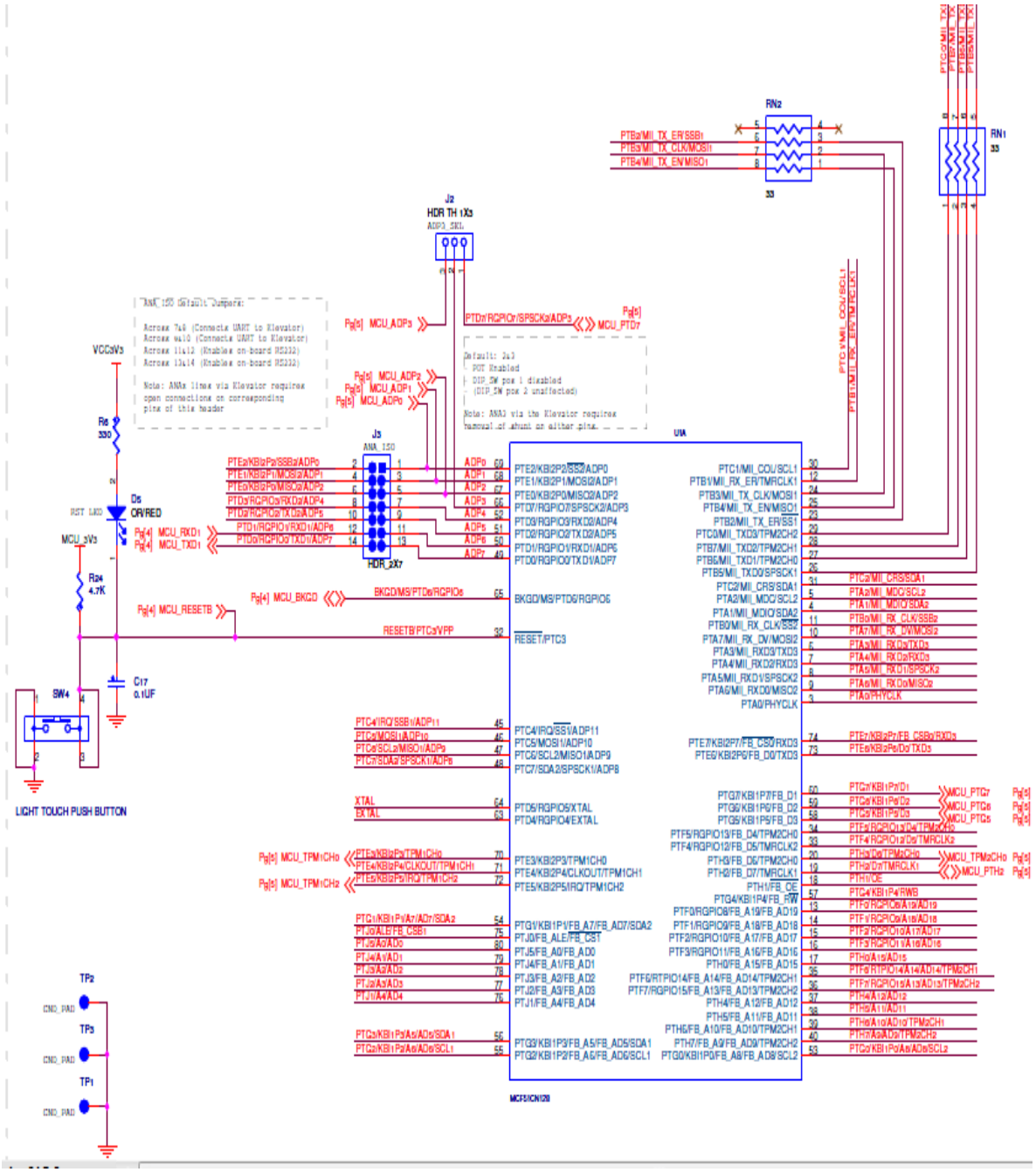
8. CONSIDERA USTED QUE EL CONTROLADOR MIDI DEL ARPA LASER PUEDE SER USADO COMO HERRAMIENTA PARA APRENDER:

    - RECONOCIMIENTO DE NOTAS MUSICALES EN EL CONTROLADOR ARPA LASER.
    - APRENDER LA EJECUCIÓN DEL INSTRUMENTO ARPA LASER.
    - ESCALAS AFINADA AL ARPA LASER.

9. CONSIDERA USTED QUE EL CONTROLADOR MIDI DEL ARPA LASER ES UNA BUENA OPCION PARA LA REPRODUCION DE AUDIO Y VIDEO DE UNA MANERA NOVEDOSA.
    a) SI
    b) NO

ANEXO B.

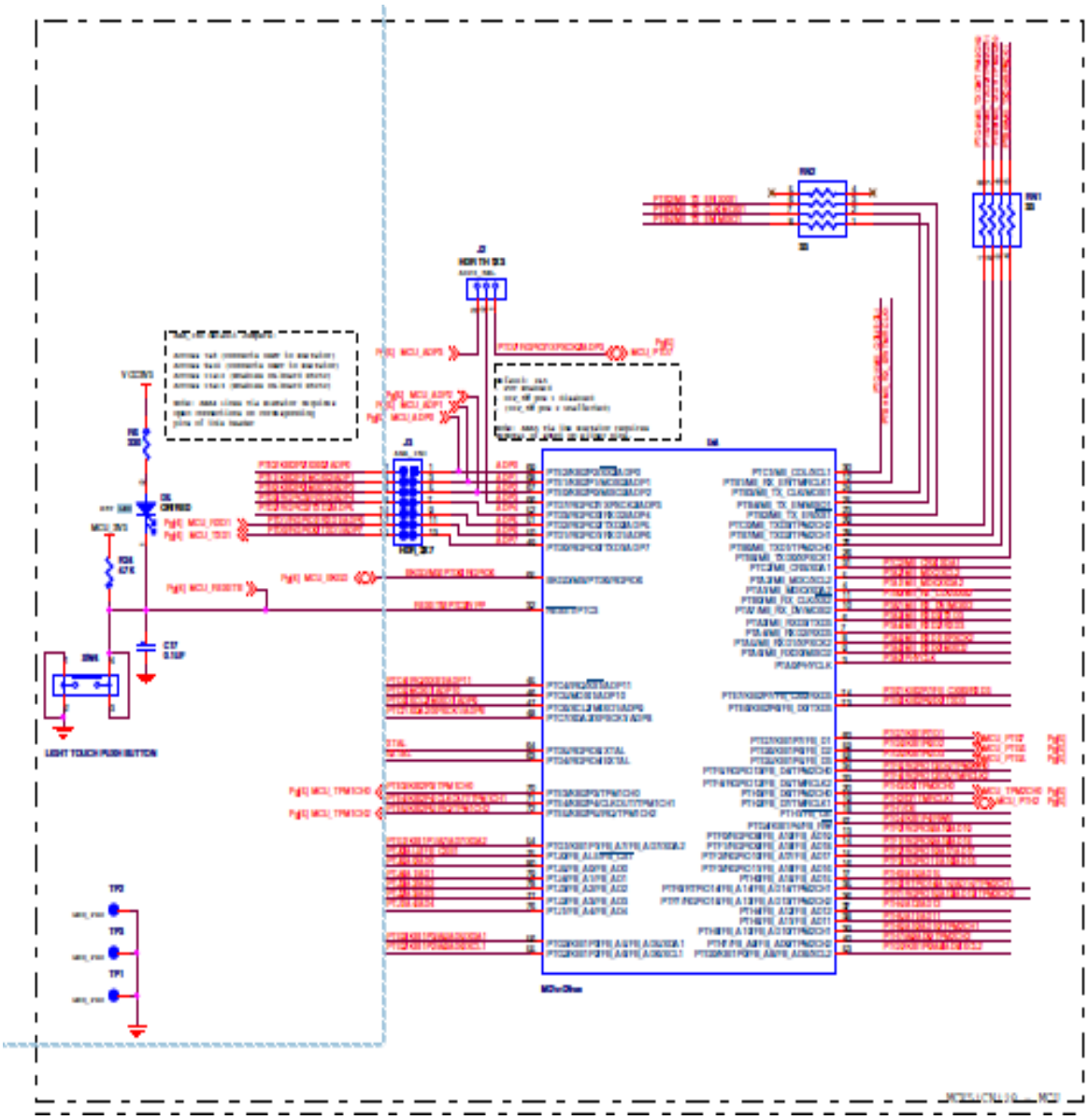
.



TWR MCF51CN128CLK

ANEXO C.

.



TWR MCF51CN128CLK

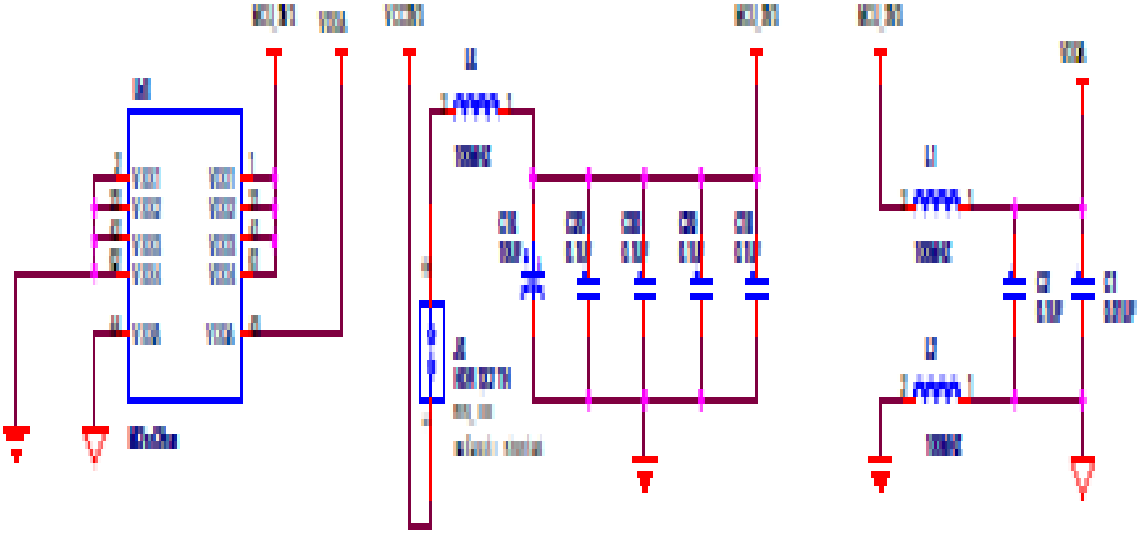
ANEXO D.

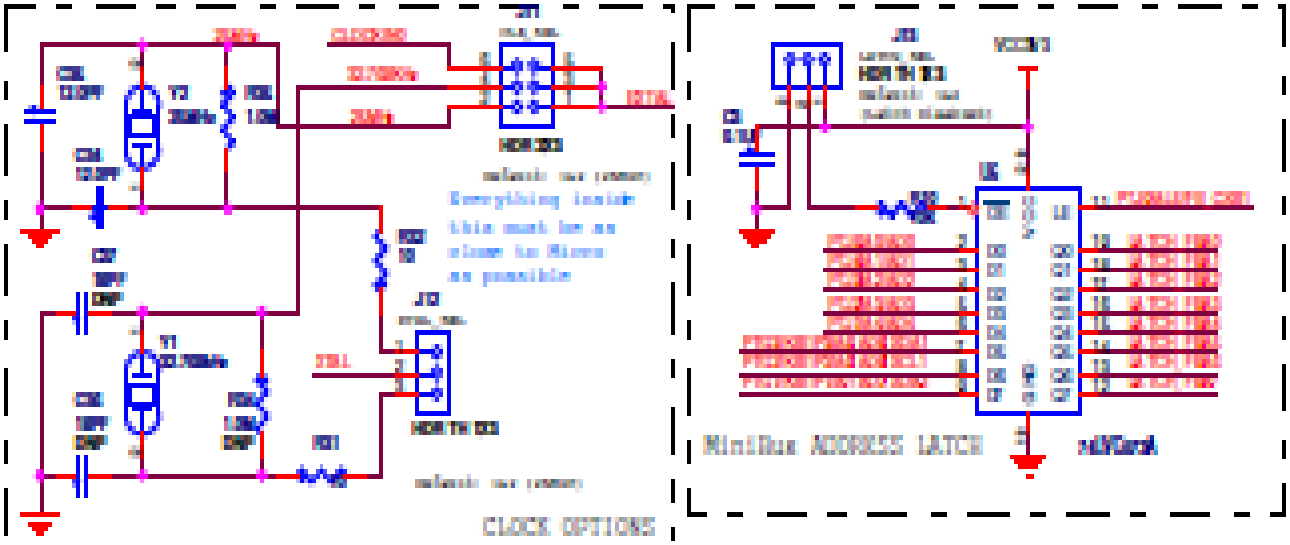
.

TWR MCF51CN128CLK

ANEXO E

.

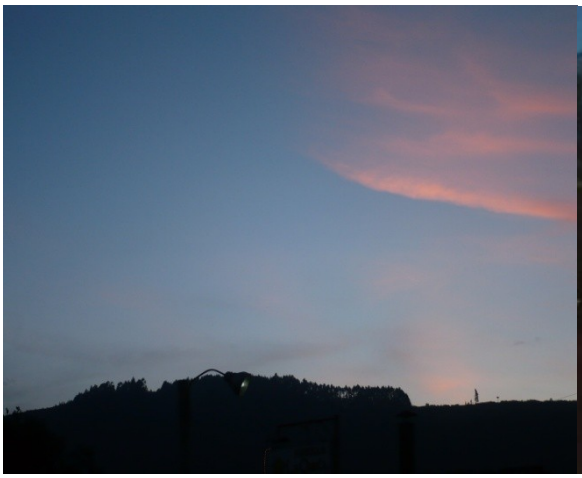




TWR **MCF51CN128CLK**

ANEXO F

BANCO DE IMAGENES



.

.




.

N° Foto	Nombre de Foto
1	ACIDO
2	AMANECER
3	ANOCHECER
4	ANTORCHA
5	ARBOL
6	ARBOLITO
7	ARDILLA
8	ATARDECER
9	BOSQUE
10	CALLE
11	CALMA
12	CIELO
13	CIELOFLECHAS
14	CORAZON
15	FLAMINGOS
16	FLOR
17	FLOR BLANCA
18	FLORECITA
19	FREUD
20	HOMBRE
21	IGLESIA
22	KANGO
23	LIBELULA
24	LOROS
25	LUNA
26	MARICOSITA
27	MARIFLOPITA
28	MARIPOSA
29	MARIPOSITA
30	MICO
31	NIÑO
32	NOCHE
33	OJO
34	PAJARO
35	PAVO
36	PELICANOS
37	TIENDA
38	TIGRE
39	TIGRE_JAULA
40	TRICOLOR
41	TURTLES
42	WOW
43	ZEBRA

ANEXO G

Programación del Controlador MIDI arpa laser

```
/** ###################################################################
**     Filename  : TestArpaLaser.C
**     Project   : TestArpaLaser
**     Processor : MCF51CN128CLK
**     Version   : Driver 01.00
**     Compiler  : CodeWarrior ColdFireV1 C Compiler
**     Date/Time : 27/09/2010, 07:48 p.m.
**     Abstract  :
**         Main module.
**         This module contains user's application code.
**     Settings  :
**     Contents  :
**         No public methods
**
** ###################################################################*/
/* MODULE TestArpaLaser */


/* Including needed modules to compile this module/procedure */
#include "Cpu.h"
#include "Events.h"
#include "LoggerSCI.h"
#include "cuerdasAD.h"
#include "laserCuerda1.h"
#include "laserCuerda2.h"
#include "laserCuerda3.h"
#include "laserCuerda4.h"
```


.

```
#include "laserCuerda5.h"
#include "laserCuerda6.h"
#include "laserCuerda7.h"
#include "laserCuerda8.h"
#include "laserCuerda9.h"
#include "laserCuerda10.h"
#include "TI1.h"
#include "pedal.h"
#include "cuerda.h"
/* Include shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"
#include "filter.h"

/* User includes (#include below this line is not maintained by Processor Expert) */

#define TH_TICKS_ACTIVATE_STRING 40

uint8_t sucPrimeraON[] = { 0x90, 0x3E, 0x50  };
uint8_t sucSegundaON[] = { 0x90,0x3C, 0x50  };
uint8_t sucTerceraON[] = { 0x90,0x3B, 0x50  };
uint8_t sucCuartaON[] = { 0x90,0x39, 0x50  };
uint8_t sucQuintaON[] = { 0x90,0x37, 0x50  };
uint8_t sucSextaON[] = { 0x90,0x35, 0x50  };
uint8_t sucSeptimaON[] = { 0x90,0x34, 0x50  };
uint8_t sucOctavaON[] = { 0x90,0x32, 0x50  };
uint8_t sucNovenaON[] = { 0x90,0x30, 0x50  };
```


.

```
uint8_t sucDecimaON[] = { 0x90,0x2F, 0x50  };

uint8_t sucPrimeraON_oct[] = { 0x90,0x4A, 0x50  };
uint8_t sucSegundaON_oct[] = { 0x90,0x48, 0x50  };
uint8_t sucTerceraON_oct[] = { 0x90,0x47, 0x50  };
uint8_t sucCuartaON_oct[] = { 0x90,0x45, 0x50  };
uint8_t sucQuintaON_oct[] = { 0x90,0x43, 0x50  };
uint8_t sucSextaON_oct[] = { 0x90,0x41, 0x50  };
uint8_t sucSeptimaON_oct[] = { 0x90,0x40, 0x50  };
uint8_t sucOctavaON_oct[] = { 0x90,0x3E, 0x50  };
uint8_t sucNovenaON_oct[] = { 0x90,0x3C, 0x50  };
uint8_t sucDecimaON_oct[] = { 0x90,0x3B, 0x50  };

uint8_t sucPrimeraOFF[] = {0x80, 0xDF };
uint8_t sucSegundaOFF[] = { 0x80,0xC8 };
uint8_t sucTerceraOFF[] ={ 0x80,0xCA };
uint8_t sucCuartaOFF[] ={ 0x80,0xCC };
uint8_t sucQuintaOFF[] ={ 0x80,0xCD };
uint8_t sucSextaOFF[] ={ 0x80,0xCF};
uint8_t sucSeptimaOFF[] ={ 0x80,0xD1 };
uint8_t sucOctavaOFF[] ={ 0x80,0xD3 };
uint8_t sucNovenaOFF[] ={ 0x80,0xBC };
uint8_t sucDecimaOFF[] ={ 0x80,0xBE };



uint8_t ticksActivateString1 = 0;
uint8_t ticksActivateString2 = 0;
uint8_t ticksActivateString3 = 0;
uint8_t ticksActivateString4 = 0;
```


.

```
uint8_t ticksActivateString5 = 0;
uint8_t ticksActivateString6 = 0;
uint8_t ticksActivateString7 = 0;
uint8_t ticksActivateString8 = 0;
uint8_t ticksActivateString9 = 0;
uint8_t ticksActivateString10 = 0;


typedef enum notes
{
  DO,
  RE
}notes_t;


void main(void)
{
  /* Write your local variable definition here */
  volatile uint16_t stringState = 0;
  uint16_t Snd = 0;

  bool bActivatedString1 = FALSE;
  bool bActivatedString2 = FALSE;
  bool bActivatedString3 = FALSE;
  bool bActivatedString4 = FALSE;
  bool bActivatedString5 = FALSE;
  bool bActivatedString6 = FALSE;
  bool bActivatedString7 = FALSE;
  bool bActivatedString8 = FALSE;
```


.

```
bool bActivatedString9  = FALSE;
bool bActivatedString10 = FALSE;



/*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
PE_low_level_init();
/*** End of Processor Expert internal initialization.                    ***/

/* Write your code here */
for(;;)
{

  stringState = (uint16_t)upDateStrings();
  if(((stringState & STRING1) == STRING1) && (!bActivatedString1))
  {
    LoggerSCI_ClearTxBuf();
    bActivatedString1 = TRUE;
    ticksActivateString1 = 0;

    if(pedal_GetVal())
      LoggerSCI_SendBlock(sucPrimeraON,sizeof(sucPrimeraON),&Snd);
    else
      LoggerSCI_SendBlock(sucPrimeraON_oct,sizeof(sucPrimeraON_oct),&Snd);
  }

  if((stringState & STRING2) == STRING2 && (!bActivatedString2))
  {
    LoggerSCI_ClearTxBuf();
```

```
  bActivatedString2 = TRUE;
  ticksActivateString2 = 0;

  if(pedal_GetVal())
    LoggerSCI_SendBlock(sucSegundaON,sizeof(sucSegundaON),&Snd);
  else
    LoggerSCI_SendBlock(sucSegundaON_oct,sizeof(sucSegundaON_oct),&Snd);
}

if((stringState & STRING3) == STRING3 && (!bActivatedString3))
{
  LoggerSCI_ClearTxBuf();

  bActivatedString3 = TRUE;
  ticksActivateString3 = 0;

  if(pedal_GetVal())
    LoggerSCI_SendBlock(sucTerceraON,sizeof(sucTerceraON),&Snd);
  else
    LoggerSCI_SendBlock(sucTerceraON_oct,sizeof(sucTerceraON_oct),&Snd);
}

if((stringState & STRING4) == STRING4 && (!bActivatedString4))
{
  LoggerSCI_ClearTxBuf();

  bActivatedString4 = TRUE;
  ticksActivateString4 = 0;
```

```
  if(pedal_GetVal())
    LoggerSCI_SendBlock(sucCuartaON,sizeof(sucCuartaON),&Snd);
  else
    LoggerSCI_SendBlock(sucCuartaON_oct,sizeof(sucCuartaON_oct),&Snd);
}

if((stringState & STRING5) == STRING5 && (!bActivatedString5))
{
  LoggerSCI_ClearTxBuf();

  bActivatedString5 = TRUE;
  ticksActivateString5 = 0;

  if(pedal_GetVal())
    LoggerSCI_SendBlock(sucQuintaON,sizeof(sucQuintaON),&Snd);
  else
    LoggerSCI_SendBlock(sucQuintaON_oct,sizeof(sucQuintaON_oct),&Snd);
}

if((stringState & STRING6) == STRING6 && (!bActivatedString6))
{
  LoggerSCI_ClearTxBuf();

  bActivatedString6 = TRUE;
  ticksActivateString6 = 0;

  if(pedal_GetVal())
    LoggerSCI_SendBlock(sucSextaON,sizeof(sucSextaON),&Snd);
```

```
  else
    LoggerSCI_SendBlock(sucSextaON_oct,sizeof(sucSextaON_oct),&Snd);
}

if((stringState & STRING7) == STRING7 && (!bActivatedString7))
{
  LoggerSCI_ClearTxBuf();

  bActivatedString7 = TRUE;
  ticksActivateString7 = 0;

  if(pedal_GetVal())
    LoggerSCI_SendBlock(sucSeptimaON,sizeof(sucSeptimaON),&Snd);
  else
    LoggerSCI_SendBlock(sucSeptimaON_oct,sizeof(sucSeptimaON_oct),&Snd);
}

if((stringState & STRING8) == STRING8 && (!bActivatedString8))
{
  LoggerSCI_ClearTxBuf();

  bActivatedString8 = TRUE;
  ticksActivateString8 = 0;

  if(pedal_GetVal())
    LoggerSCI_SendBlock(sucOctavaON,sizeof(sucOctavaON),&Snd);
  else
    LoggerSCI_SendBlock(sucOctavaON_oct,sizeof(sucOctavaON_oct),&Snd);
}
```

```
if((stringState & STRING9) == STRING9 && (!bActivatedString9))
{
  LoggerSCI_ClearTxBuf();

  bActivatedString9 = TRUE;
  ticksActivateString9 = 0;

  if(pedal_GetVal())
    LoggerSCI_SendBlock(sucNovenaON,sizeof(sucNovenaON),&Snd);
  else
    LoggerSCI_SendBlock(sucNovenaON_oct,sizeof(sucNovenaON_oct),&Snd);
}

if((stringState & STRING10) == STRING10 && (!bActivatedString10))
{
  LoggerSCI_ClearTxBuf();

  bActivatedString10 = TRUE;
  ticksActivateString10 = 0;

  if(pedal_GetVal())
    LoggerSCI_SendBlock(sucDecimaON,sizeof(sucDecimaON),&Snd);
  else
    LoggerSCI_SendBlock(sucDecimaON_oct,sizeof(sucDecimaON_oct),&Snd);
}

// Temporización de reactivación de la cuerda
if((bActivatedString1) && (ticksActivateString1 > TH_TICKS_ACTIVATE_STRING))
```

```
{
  bActivatedString1 = FALSE;
}

// Temporización de reactivación de la cuerda
if((bActivatedString2) && (ticksActivateString2 > TH_TICKS_ACTIVATE_STRING))
{
  bActivatedString2 = FALSE;
}

if((bActivatedString3) && (ticksActivateString3 > TH_TICKS_ACTIVATE_STRING))
{
  bActivatedString3 = FALSE;
}

if((bActivatedString4) && (ticksActivateString4 > TH_TICKS_ACTIVATE_STRING))
{
  bActivatedString4 = FALSE;
}

if((bActivatedString5) && (ticksActivateString5 > TH_TICKS_ACTIVATE_STRING))
{
  bActivatedString5 = FALSE;
}

if((bActivatedString6) && (ticksActivateString6 > TH_TICKS_ACTIVATE_STRING))
{
  bActivatedString6 = FALSE;
}
```

```
    if((bActivatedString7) && (ticksActivateString7 > TH_TICKS_ACTIVATE_STRING))
    {
      bActivatedString7 = FALSE;
    }

    if((bActivatedString8) && (ticksActivateString8 > TH_TICKS_ACTIVATE_STRING))
    {
      bActivatedString8= FALSE;
    }

    if((bActivatedString9) && (ticksActivateString9 > TH_TICKS_ACTIVATE_STRING))
    {
      bActivatedString9= FALSE;
    }

    if((bActivatedString10) && (ticksActivateString10 > TH_TICKS_ACTIVATE_STRING))
    {
      bActivatedString10= FALSE;
    }
  }

  /*** Don't write any code pass this line, or it will be deleted during code generation. ***/
  /*** Processor Expert end of main routine. DON'T MODIFY THIS CODE!!! ***/
  for(;;){}
  /*** Processor Expert end of main routine. DON'T WRITE CODE BELOW!!! ***/
} /*** End of main routine. DO NOT MODIFY THIS TEXT!!! ***/

/* END TestArpaLaser */
```


.

```
/*
** ###################################################################
**
**     This file was created by Processor Expert 3.07 [04.34]
**     for the Freescale ColdFireV1 series of microcontrollers.
**
** ###################################################################
*/
CUERDA.C
/** ###################################################################
**     Filename  : cuerda.C
**     Project   : TestArpaLaser
**     Processor : MCF51CN128CLK
**     Compiler  : CodeWarrior ColdFireV1 C Compiler
**     Date/Time : 27/11/2010, 19:31
**     Contents  :
**         User source code
**
** ###################################################################*/

/* MODULE cuerda */
#include "cuerda.h"

uint16_t upDateStrings()
{
  volatile uint16_t estadoCuerdas = 0;

  uint16_t cuerda1Value1 = 0;
  uint16_t cuerda1Value2 = 0;
```


.

```
uint16_t cuerda2Value1 = 0;
uint16_t cuerda2Value2 = 0;

uint16_t cuerda3Value1 = 0;
uint16_t cuerda3Value2 = 0;

uint16_t cuerda4Value1 = 0;
uint16_t cuerda4Value2 = 0;

uint16_t cuerda5Value1 = 0;
uint16_t cuerda5Value2 = 0;

uint16_t cuerda6Value1 = 0;
uint16_t cuerda6Value2 = 0;

uint16_t cuerda7Value1 = 0;
uint16_t cuerda7Value2 = 0;

uint16_t cuerda8Value1 = 0;
uint16_t cuerda8Value2 = 0;

uint16_t cuerda9Value1 = 0;
uint16_t cuerda9Value2 = 0;

uint16_t cuerda10Value1 = 0;
uint16_t cuerda10Value2 = 0;

static uint16_t cuerda1deltaValues = 0xFFFF;
```

```
static uint16_t cuerda2deltaValues = 0xFFFF;
static uint16_t cuerda3deltaValues = 0xFFFF;
static uint16_t cuerda4deltaValues = 0xFFFF;
static uint16_t cuerda5deltaValues = 0xFFFF;
static uint16_t cuerda6deltaValues = 0xFFFF;
static uint16_t cuerda7deltaValues = 0xFFFF;
static uint16_t cuerda8deltaValues = 0xFFFF;
static uint16_t cuerda9deltaValues = 0xFFFF;
static uint16_t cuerda10deltaValues = 0xFFFF;

// Se encienden los laser
laserCuerda1_SetVal();
laserCuerda2_SetVal();
laserCuerda3_SetVal();
laserCuerda4_SetVal();
laserCuerda5_SetVal();
laserCuerda6_SetVal();
laserCuerda7_SetVal();
laserCuerda8_SetVal();
laserCuerda9_SetVal();
laserCuerda10_SetVal();
Cpu_Delay100US(50);

// Se adquiere el valor analógico con los laser encendidos
cuerdasAD_Measure(TRUE);
cuerdasAD_GetChanValue16(0,&cuerda1Value1);
cuerdasAD_GetChanValue16(1,&cuerda2Value1);
cuerdasAD_GetChanValue16(2,&cuerda3Value1);
cuerdasAD_GetChanValue16(3,&cuerda4Value1);
```

```
cuerdasAD_GetChanValue16(4,&cuerda5Value1);
cuerdasAD_GetChanValue16(5,&cuerda6Value1);
cuerdasAD_GetChanValue16(6,&cuerda7Value1);
cuerdasAD_GetChanValue16(7,&cuerda8Value1);
cuerdasAD_GetChanValue16(8,&cuerda9Value1);
cuerdasAD_GetChanValue16(9,&cuerda10Value1);


// Se apagan los laser
laserCuerda1_ClrVal();
laserCuerda2_ClrVal();
laserCuerda3_ClrVal();
laserCuerda4_ClrVal();
laserCuerda5_ClrVal();
laserCuerda6_ClrVal();
laserCuerda7_ClrVal();
laserCuerda8_ClrVal();
laserCuerda9_ClrVal();
laserCuerda10_ClrVal();
Cpu_Delay100US(50);

// Se adquiere los valores analógicos con los laser apagados
cuerdasAD_Measure(TRUE);
cuerdasAD_GetChanValue16(0,&cuerda1Value2);
cuerdasAD_GetChanValue16(1,&cuerda2Value2);
cuerdasAD_GetChanValue16(2,&cuerda3Value2);
cuerdasAD_GetChanValue16(3,&cuerda4Value2);
cuerdasAD_GetChanValue16(4,&cuerda5Value2);
cuerdasAD_GetChanValue16(5,&cuerda6Value2);
```

```
cuerdasAD_GetChanValue16(6,&cuerda7Value2);
cuerdasAD_GetChanValue16(7,&cuerda8Value2);
cuerdasAD_GetChanValue16(8,&cuerda9Value2);
cuerdasAD_GetChanValue16(9,&cuerda10Value2);

// se calcula el valor absoluto entre las muestras analógicas
// se pasa dicho valor a través de un filto de respuesta infinita al impulso
filter_iir(absV2_V1(cuerda1Value1, cuerda1Value2),&cuerda1deltaValues,2);
filter_iir(absV2_V1(cuerda2Value1, cuerda2Value2),&cuerda2deltaValues,2);
filter_iir(absV2_V1(cuerda3Value1, cuerda3Value2),&cuerda3deltaValues,2);
filter_iir(absV2_V1(cuerda4Value1, cuerda4Value2),&cuerda4deltaValues,2);
filter_iir(absV2_V1(cuerda5Value1, cuerda5Value2),&cuerda5deltaValues,2);
filter_iir(absV2_V1(cuerda6Value1, cuerda6Value2),&cuerda6deltaValues,2);
filter_iir(absV2_V1(cuerda7Value1, cuerda7Value2),&cuerda7deltaValues,2);
filter_iir(absV2_V1(cuerda8Value1, cuerda8Value2),&cuerda8deltaValues,2);
filter_iir(absV2_V1(cuerda9Value1, cuerda9Value2),&cuerda9deltaValues,2);
filter_iir(absV2_V1(cuerda10Value1, cuerda10Value2),&cuerda10deltaValues,2);

estadoCuerdas += (cuerda1deltaValues < TH_ACTIVATION) ? STRING1  : 0x00;
estadoCuerdas += (cuerda2deltaValues < TH_ACTIVATION) ? STRING2  : 0x00;
estadoCuerdas += (cuerda3deltaValues < TH_ACTIVATION) ? STRING3  : 0x00;
estadoCuerdas += (cuerda4deltaValues < TH_ACTIVATION) ? STRING4  : 0x00;
estadoCuerdas += (cuerda5deltaValues < TH_ACTIVATION) ? STRING5  : 0x00;
estadoCuerdas += (cuerda6deltaValues < TH_ACTIVATION) ? STRING6  : 0x00;
estadoCuerdas += (cuerda7deltaValues < TH_ACTIVATION) ? STRING7  : 0x00;
estadoCuerdas += (cuerda8deltaValues < TH_ACTIVATION) ? STRING8  : 0x00;
estadoCuerdas += (cuerda9deltaValues < TH_ACTIVATION) ? STRING9  : 0x00;
estadoCuerdas += (cuerda10deltaValues < TH_ACTIVATION) ? STRING10 : 0x00;
```

```
  return estadoCuerdas;
}


/* END cuerda */
EVENTS
/** ###################################################################
**     Filename  : Events.C
**     Project   : TestArpaLaser
**     Processor : MCF51CN128CLK
**     Component : Events
**     Version   : Driver 01.02
**     Compiler  : CodeWarrior ColdFireV1 C Compiler
**     Date/Time : 27/09/2010, 07:48 p.m.
**     Abstract  :
**         This is user's event module.
**         Put your event handler code here.
**     Settings  :
**     Contents  :
**         No public methods
**
** ###################################################################*/
/* MODULE Events */

#include "Cpu.h"
#include "Events.h"

extern uint8_t ticksActivateString1;
extern uint8_t ticksActivateString2;
```


.

```
extern uint8_t ticksActivateString3;
extern uint8_t ticksActivateString4;
extern uint8_t ticksActivateString5;
extern uint8_t ticksActivateString6;
extern uint8_t ticksActivateString7;
extern uint8_t ticksActivateString8;
extern uint8_t ticksActivateString9;
extern uint8_t ticksActivateString10;


/* User includes (#include below this line is not maintained by Processor Expert) */

/*
**
===================================================================
**     Event      :  LoggerSCI_OnError (module Events)
**
**     Component   :  LoggerSCI [AsynchroSerial]
**     Description :
**         This event is called when a channel error (not the error
**         returned by a given method) occurs. The errors can be
**         read using <GetError> method.
**         The event is available only when the <Interrupt
**         service/event> property is enabled.
**     Parameters  : None
**     Returns     : Nothing
**
===================================================================
*/
```


.

```
void  LoggerSCI_OnError(void)
{
  /* Write your code here ... */
}

/*
**
===================================================================
**     Event       :  LoggerSCI_OnRxChar (module Events)
**
**     Component   :  LoggerSCI [AsynchroSerial]
**     Description :
**         This event is called after a correct character is
**         received.
**         The event is available only when the <Interrupt
**         service/event> property is enabled and either the
**         <Receiver> property is enabled or the <SCI output mode>
**         property (if supported) is set to Single-wire mode.
**     Parameters  : None
**     Returns     : Nothing
**
===================================================================
*/
void  LoggerSCI_OnRxChar(void)
{
  /* Write your code here ... */
}

/*
```


.

```
**
**===========================================================
**     Event       :  LoggerSCI_OnTxChar (module Events)
**
**     Component   :  LoggerSCI [AsynchroSerial]
**     Description :
**         This event is called after a character is transmitted.
**     Parameters  : None
**     Returns     : Nothing
**
**===========================================================
*/
void  LoggerSCI_OnTxChar(void)
{
  /* Write your code here ... */
}

/*
**
**===========================================================
**     Event       :  LoggerSCI_OnFullRxBuf (module Events)
**
**     Component   :  LoggerSCI [AsynchroSerial]
**     Description :
**         This event is called when the input buffer is full;
**         i.e. after reception of the last character
**         that was successfully placed into input buffer.
**     Parameters  : None
**     Returns     : Nothing
```


.

```
**
===================================================================
*/
void  LoggerSCI_OnFullRxBuf(void)
{
  /* Write your code here ... */
}

/*
**
===================================================================
**     Event       :  LoggerSCI_OnFreeTxBuf (module Events)
**
**     Component   :  LoggerSCI [AsynchroSerial]
**     Description :
**         This event is called after the last character in output
**         buffer is transmitted.
**     Parameters  : None
**     Returns     : Nothing
**
===================================================================
*/
void  LoggerSCI_OnFreeTxBuf(void)
{
  /* Write your code here ... */
}

/*
```


.

```
**
**===========================================================
**     Event       :  TI1_OnInterrupt (module Events)
**
**     Component   :  TI1 [TimerInt]
**     Description :
**         When a timer interrupt occurs this event is called (only
**         when the component is enabled - <Enable> and the events are
**         enabled - <EnableEvent>). This event is enabled only if a
**         <interrupt service/event> is enabled.
**     Parameters  : None
**     Returns     : Nothing
**
**===========================================================
*/
void TI1_OnInterrupt(void)
{
  /* Write your code here ... */
   ++ticksActivateString1;
   ++ticksActivateString2;
   ++ticksActivateString3;
   ++ticksActivateString4;
   ++ticksActivateString5;
   ++ticksActivateString6;
   ++ticksActivateString7;
   ++ticksActivateString8;
   ++ticksActivateString9;
   ++ticksActivateString10;
}
```


.

```
/* END Events */

/*
** ###################################################################
**
**     This file was created by Processor Expert 3.07 [04.34]
**     for the Freescale ColdFireV1 series of microcontrollers.
**
** ###################################################################
*/


FILTER
/** ###################################################################
**     Filename  : filter.C
**     Project   : TestArpaLaser
**     Processor : MCF51CN128CLK
**     Compiler  : CodeWarrior ColdFireV1 C Compiler
**     Date/Time : 17/10/2010, 09:32 p.m.
**     Contents  :
**         User source code
**
** ###################################################################*/
#include "PE_Types.h"
/* MODULE filter */
void filter_iir(uint16_t current, uint16_t * pFiltered, uint8_t cte)
{
  uint16_t acumul = *pFiltered;
```


.

```
  acumul = (uint16_t)((current >> cte ) + (acumul - (acumul>>cte)));
  *pFiltered = acumul;
}

/* END filter */

/** ###################################################################
**     Filename  : utilidades.C
**     Project   : TestArpaLaser
**     Processor : MCF51CN128CLK
**     Compiler  : CodeWarrior ColdFireV1 C Compiler
**     Date/Time : 27/11/2010, 20:18
**     Contents  :
**         User source code
**
** ###################################################################*/

/* MODULE utilidades */

#include "utilidades.h"

uint16_t absV2_V1(uint16_t value1, uint16_t value2 )
{
  uint16_t deltaValues = 0;
  if(value2 > value1)
  {
     deltaValues = (uint16_t)(value2 - value1);
  }
  else
```


.

```
  {
     deltaValues = (uint16_t)(value1 - value2);
  }
  return deltaValues;
}

/* END utilidades */
```


.

ANEXO G

VISUALC#2008 EXPRESS EDITION

PARTE DE CARGAR LOS AUDIOS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Reflection;
using System.IO;
using System.Media;

namespace ApraLaserMidi
{

   public sealed class Audio
   {
      public static void SonidoArpaPrimera()
      {
         Assembly a = Assembly.GetExecutingAssembly();
         Stream s = a.GetManifestResourceStream("ApraLaserMidi.Sonidos.CUERDA
primera.wav");
         SoundPlayer auf1 = new SoundPlayer(s);
         auf1.Play();
      }

      public static void SonidoArpaSegunda()
      {
         Assembly a = Assembly.GetExecutingAssembly();
         Stream s = a.GetManifestResourceStream("ApraLaserMidi.Sonidos.CUERDA
segunda.wav");
         SoundPlayer auf1 = new SoundPlayer(s);
         auf1.Play();
      }

      public static void SonidoArpaTercera()
      {
         Assembly a = Assembly.GetExecutingAssembly();
```


.

```
            Stream s = a.GetManifestResourceStream("ApraLaserMidi.Sonidos.CUERDA
tercera.wav");
            SoundPlayer auf1 = new SoundPlayer(s);
            auf1.Play();
        }

        public static void SonidoArpaCuarta()
        {
            Assembly a = Assembly.GetExecutingAssembly();
            Stream s = a.GetManifestResourceStream("ApraLaserMidi.Sonidos.CUERDA
cuarta.wav");
            SoundPlayer auf1 = new SoundPlayer(s);
            auf1.Play();
        }
        public static void SonidoArpaQuinta()
        {
            Assembly a = Assembly.GetExecutingAssembly();
            Stream s = a.GetManifestResourceStream("ApraLaserMidi.Sonidos.CUERDA
quinta.wav");
            SoundPlayer auf1 = new SoundPlayer(s);
            auf1.Play();
        }
        public static void SonidoArpaSexta()
        {
            Assembly a = Assembly.GetExecutingAssembly();
            Stream s = a.GetManifestResourceStream("ApraLaserMidi.Sonidos.CUERDA
sexta.wav");
            SoundPlayer auf1 = new SoundPlayer(s);
            auf1.Play();
        }
        public static void SonidoArpaSeptima()
        {
            Assembly a = Assembly.GetExecutingAssembly();
            Stream s = a.GetManifestResourceStream("ApraLaserMidi.Sonidos.CUERDA
septima.wav");
            SoundPlayer auf1 = new SoundPlayer(s);
            auf1.Play();
        }
        public static void SonidoArpaOctava()
        {
            Assembly a = Assembly.GetExecutingAssembly();
            Stream s = a.GetManifestResourceStream("ApraLaserMidi.Sonidos.CUERDA
octava.wav");
```


.

```
            SoundPlayer auf1 = new SoundPlayer(s);
            auf1.Play();
        }
        public static void SonidoArpaNovena()
        {
            Assembly a = Assembly.GetExecutingAssembly();
            Stream s = a.GetManifestResourceStream("ApraLaserMidi.Sonidos.CUERDA
novena.wav");
            SoundPlayer auf1 = new SoundPlayer(s);
            auf1.Play();
        }
        public static void SonidoArpaDecima()
        {
            Assembly a = Assembly.GetExecutingAssembly();
            Stream s = a.GetManifestResourceStream("ApraLaserMidi.Sonidos.CUERDA
decima.wav");
            SoundPlayer auf1 = new SoundPlayer(s);
            auf1.Play();
        }




        internal static void SonidoArpaPrimera_oct()
        {
            Assembly a = Assembly.GetExecutingAssembly();
            Stream s = a.GetManifestResourceStream("ApraLaserMidi.Sonidos.CUERDA
primera_oct.wav");
            SoundPlayer auf1 = new SoundPlayer(s);
            auf1.Play();
        }

        internal static void SonidoArpaSegunda_oct()
        {
            Assembly a = Assembly.GetExecutingAssembly();
            Stream s = a.GetManifestResourceStream("ApraLaserMidi.Sonidos.CUERDA
segunda_oct.wav");
            SoundPlayer auf1 = new SoundPlayer(s);
            auf1.Play();
        }
```


.

```
        internal static void SonidoArpaTercera_oct()
        {
            Assembly a = Assembly.GetExecutingAssembly();
            Stream s = a.GetManifestResourceStream("ApraLaserMidi.Sonidos.CUERDA
tercera_oct.wav");
            SoundPlayer auf1 = new SoundPlayer(s);
            auf1.Play();
        }

        internal static void SonidoArpaCuarta_oct()
        {
            Assembly a = Assembly.GetExecutingAssembly();
            Stream s = a.GetManifestResourceStream("ApraLaserMidi.Sonidos.CUERDA
cuarta_oct.wav");
            SoundPlayer auf1 = new SoundPlayer(s);
            auf1.Play();
        }

        internal static void SonidoArpaQuinta_oct()
        {
            Assembly a = Assembly.GetExecutingAssembly();
            Stream s = a.GetManifestResourceStream("ApraLaserMidi.Sonidos.CUERDA
quinta_oct.wav");
            SoundPlayer auf1 = new SoundPlayer(s);
            auf1.Play();
        }

        internal static void SonidoArpaSexta_oct()
        {
            Assembly a = Assembly.GetExecutingAssembly();
            Stream s = a.GetManifestResourceStream("ApraLaserMidi.Sonidos.CUERDA
sexta_oct.wav");
            SoundPlayer auf1 = new SoundPlayer(s);
            auf1.Play();
        }

        internal static void SonidoArpaSeptima_oct()
        {
            Assembly a = Assembly.GetExecutingAssembly();
            Stream s = a.GetManifestResourceStream("ApraLaserMidi.Sonidos.CUERDA
septima_oct.wav");
            SoundPlayer auf1 = new SoundPlayer(s);
            auf1.Play();
```


.

```
        }

        internal static void SonidoArpaOctava_oct()
        {
            Assembly a = Assembly.GetExecutingAssembly();
            Stream s = a.GetManifestResourceStream("ApraLaserMidi.Sonidos.CUERDA
octava_oct.wav");
            SoundPlayer auf1 = new SoundPlayer(s);
            auf1.Play();
        }

        internal static void SonidoArpaNovena_oct()
        {
            Assembly a = Assembly.GetExecutingAssembly();
            Stream s = a.GetManifestResourceStream("ApraLaserMidi.Sonidos.CUERDA
novena_oct.wav");
            SoundPlayer auf1 = new SoundPlayer(s);
            auf1.Play();
        }

        internal static void SonidoArpaDecima_oct()
        {
            Assembly a = Assembly.GetExecutingAssembly();
            Stream s = a.GetManifestResourceStream("ApraLaserMidi.Sonidos.CUERDA
decima_oct.wav");
            SoundPlayer auf1 = new SoundPlayer(s);
            auf1.Play();
        }
    }
}
```

PARTE VIDEO

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Drawing;
using System.Reflection;
```


.

```
namespace ApraLaserMidi
{

   public sealed class Visual
   {
      public static void visualCielo(PictureBox pictureBox1)
      {
         pictureBox1.Image = new
Bitmap(Assembly.GetExecutingAssembly().GetManifestResourceStream("ApraLaserMid
i.Imagenes.cielo.JPG"));
      }
      public static void visualTigre(PictureBox pictureBox1)
      {
         pictureBox1.Image = new
Bitmap(Assembly.GetExecutingAssembly().GetManifestResourceStream("ApraLaserMid
i.Imagenes.tigre.JPG"));

      }
      public static void visualArdilla(PictureBox pictureBox1)
      {
         pictureBox1.Image = new
Bitmap(Assembly.GetExecutingAssembly().GetManifestResourceStream("ApraLaserMid
i.Imagenes.ardilla.JPG"));
      }
      public static void visualArbol(PictureBox pictureBox1)
      {
         pictureBox1.Image = new
Bitmap(Assembly.GetExecutingAssembly().GetManifestResourceStream("ApraLaserMid
i.Imagenes.arbol.JPG"));
      }
      public static void visualAtardecer(PictureBox pictureBox1)
      {
         pictureBox1.Image = new
Bitmap(Assembly.GetExecutingAssembly().GetManifestResourceStream("ApraLaserMid
i.Imagenes.atardecer.JPG"));
      }

      public static void visualFlorBlanca(PictureBox pictureBox1)
      {
         pictureBox1.Image = new
Bitmap(Assembly.GetExecutingAssembly().GetManifestResourceStream("ApraLaserMid
i.Imagenes.flor blanca.JPG"));
      }
```


.

```
        public static void visualFlor(PictureBox pictureBox1)
        {
            pictureBox1.Image = new
Bitmap(Assembly.GetExecutingAssembly().GetManifestResourceStream("ApraLaserMid
i.Imagenes.flor.JPG"));
        }
        public static void visualOjo(PictureBox pictureBox1)
        {
            pictureBox1.Image = new
Bitmap(Assembly.GetExecutingAssembly().GetManifestResourceStream("ApraLaserMid
i.Imagenes.OJO.bmp"));
        }
        public static void visualPavo(PictureBox pictureBox1)
        {
            pictureBox1.Image = new
Bitmap(Assembly.GetExecutingAssembly().GetManifestResourceStream("ApraLaserMid
i.Imagenes.PAVO.bmp"));
        }
        public static void visualTurtles(PictureBox pictureBox1)
        {
            pictureBox1.Image = new
Bitmap(Assembly.GetExecutingAssembly().GetManifestResourceStream("ApraLaserMid
i.Imagenes.TURTLES.jpg"));
        }
        internal static void visualFlamingos(PictureBox pictureBox1)
        {
            pictureBox1.Image = new
Bitmap(Assembly.GetExecutingAssembly().GetManifestResourceStream("ApraLaserMid
i.Imagenes.FLAMINGOS.JPG"));
        }
        internal static void visualFlorecita(PictureBox pictureBox1)
        {
            pictureBox1.Image = new
Bitmap(Assembly.GetExecutingAssembly().GetManifestResourceStream("ApraLaserMid
i.Imagenes.FLORECITA.JPG"));
        }
        internal static void visualHombre(PictureBox pictureBox1)
        {
            pictureBox1.Image = new
Bitmap(Assembly.GetExecutingAssembly().GetManifestResourceStream("ApraLaserMid
i.Imagenes.HOMBRE.jpg"));
        }
        internal static void visualLuna(PictureBox pictureBox1)
```


.

```
        {
            pictureBox1.Image = new
Bitmap(Assembly.GetExecutingAssembly().GetManifestResourceStream("ApraLaserMid
i.Imagenes.LUNA.bmp"));
        }
        internal static void visualMariflopita(PictureBox pictureBox1)
        {
            pictureBox1.Image = new
Bitmap(Assembly.GetExecutingAssembly().GetManifestResourceStream("ApraLaserMid
i.Imagenes.MARIFLOPITA.JPG"));
        }
            internal static void visualMariposa(PictureBox pictureBox1)
        {
            pictureBox1.Image = new
Bitmap(Assembly.GetExecutingAssembly().GetManifestResourceStream("ApraLaserMid
i.Imagenes.MARIPOSA.JPG"));
        }
            internal static void visualMariposita(PictureBox pictureBox1)
        {
             pictureBox1.Image = new
Bitmap(Assembly.GetExecutingAssembly().GetManifestResourceStream("ApraLaserMid
i.Imagenes.MARIPOSITA.JPG"));
        }
            internal static void visualNiño(PictureBox pictureBox1)
        {
            pictureBox1.Image = new
Bitmap(Assembly.GetExecutingAssembly().GetManifestResourceStream("ApraLaserMid
i.Imagenes.NIÑO.jpg"));
            }
            internal static void visualPajaro(PictureBox pictureBox1)
            {
                pictureBox1.Image = new
Bitmap(Assembly.GetExecutingAssembly().GetManifestResourceStream("ApraLaserMid
i.Imagenes.PAJARO.jpg"));
            }
            internal static void visualPelicanos(PictureBox pictureBox1)
            {
                 pictureBox1.Image = new
Bitmap(Assembly.GetExecutingAssembly().GetManifestResourceStream("ApraLaserMid
i.Imagenes.PELICANOS.JPG"));
            }

            internal static void visualTigre_jaula(PictureBox pictureBox1)
```


.

```
        {
            pictureBox1.Image = new
Bitmap(Assembly.GetExecutingAssembly().GetManifestResourceStream("ApraLaserMid
i.Imagenes.TIGRE_JAULA.JPG"));
        }

        internal static void visualZebra(PictureBox pictureBox1)
        {
            pictureBox1.Image = new
Bitmap(Assembly.GetExecutingAssembly().GetManifestResourceStream("ApraLaserMid
i.Imagenes.ZEBRA.JPG"));
        }
    }

}
```

PARTE QUE CARGA ARCHIVOS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace ApraLaserMidi
{
    public struct Packet_t
    {
        public byte[] payLoad;
        public int size;
    };

    enum MEF_Sonido
    {
        PRIMERA,
        SEGUNDA,
        TERCERA,
        CUARTA,
        QUINTA,
        SEXTA,
        SEPTIMA,
        OCTAVA,
        NOVENA,
```


.

```
    DECIMA,

    PRIMERA_OCT,
    SEGUNDA_OCT,
    TERCERA_OCT,
    CUARTA_OCT,
    QUINTA_OCT,
    SEXTA_OCT,
    SEPTIMA_OCT,
    OCTAVA_OCT,
    NOVENA_OCT,
    DECIMA_OCT,
    DESCONOCIDO
}

class PacketManager
{
    PictureBox pictureBox1;
    Random fotoRandom;
    public event EventHandler NewMidiCode;

    byte codigoMidi;
    internal byte CodigoMidi
    {
        get { return codigoMidi; }
    }
    public PacketManager(PictureBox pictureBox1)
    {
        this.pictureBox1 = pictureBox1;
        this.fotoRandom = new Random((int)DateTime.Now.Ticks);
    }

    public void PacketReceived_Event(object sender, EventArgs e)
    {
        PacketEventArgs packetEventArgs = (PacketEventArgs)e;
        MEF_Sonido sonido = MEF_Sonido.DESCONOCIDO;
        /*
        string cmd = "";
        cmd = packetEventArgs.cmd;
        cmd = cmd.Replace('\0', ' ');
        cmd = cmd.Replace('\r', ' ');
        cmd = cmd.Trim();
        sonido = ConvertCmdToMEF_Sonido(cmd);
```

```
        */

        for (int i = 0; i < packetEventArgs.size; i++)
        {
            sonido = ConvertCmdToMEF_Sonido(packetEventArgs.cndMidiBuffer[i]);
            if (sonido != MEF_Sonido.DESCONOCIDO)
            {
                CodeMidiEventArgs codeMidi = new
CodeMidiEventArgs(packetEventArgs.cndMidiBuffer[i]);
                this.NewMidiCode(this.codigoMidi, codeMidi);
                break;
            }
            else
                sonido = MEF_Sonido.DESCONOCIDO;
        }
        this.reproducirArpa(sonido);
        this.printImage();
    }

    private MEF_Sonido ConvertCmdToMEF_Sonido(string cmd)
    {
        switch (cmd)
        {
            case "Primera on":
                {
                    return MEF_Sonido.PRIMERA;
                }
            case "Segunda on":
                {
                    return MEF_Sonido.SEGUNDA;
                }
            case "Tercera on":
                {
                    return MEF_Sonido.TERCERA;
                }
            case "Cuarta on":
                {
                    return MEF_Sonido.CUARTA;
                }
            case "Quinta on":
                {
                    return MEF_Sonido.QUINTA;
                }
```


.

```
        case "Sexta on":
           {
              return MEF_Sonido.SEXTA;
           }
        case "Septima on":
           {
              return MEF_Sonido.SEPTIMA;
           }
        case "Octava on":
           {
              return MEF_Sonido.OCTAVA;
           }
        case "Novena on":
           {
              return MEF_Sonido.NOVENA;
           }
        case "Decima on":
           {
              return MEF_Sonido.DECIMA;
           }

        default:
           {
              return MEF_Sonido.DESCONOCIDO;
           }
     }
  }

  private MEF_Sonido ConvertCmdToMEF_Sonido(byte cmd)
  {
     switch (cmd)
     {
        case 0x3E:
           {
              return MEF_Sonido.PRIMERA;
           }
        case 0x3C   :
           {
              return MEF_Sonido.SEGUNDA;
           }
        case 0x3B:
           {
              return MEF_Sonido.TERCERA;
```

```
   }
case 0x39:
   {
      return MEF_Sonido.CUARTA;
   }
case 0x37:
   {
      return MEF_Sonido.QUINTA;
   }
case 0x35:
   {
      return MEF_Sonido.SEXTA;
   }
case 0x34:
   {
      return MEF_Sonido.SEPTIMA;
   }
case 0x32:
   {
      return MEF_Sonido.OCTAVA;
   }
case 0x30:
   {
      return MEF_Sonido.NOVENA;
   }
case 0X2F:
   {
      return MEF_Sonido.DECIMA;
   }


case 0x4A:
   {
      return MEF_Sonido.PRIMERA_OCT;
   }
case 0x48:
   {
      return MEF_Sonido.SEGUNDA_OCT;
   }
case 0x47:
   {
      return MEF_Sonido.TERCERA_OCT;
   }
```

```
        case 0x45:
            {
                return MEF_Sonido.CUARTA_OCT;
            }
        case 0x43:
            {
                return MEF_Sonido.QUINTA_OCT;
            }
        case 0x41:
            {
                return MEF_Sonido.SEXTA_OCT;
            }
        case 0x40:
            {
                return MEF_Sonido.SEPTIMA_OCT;
            }

        default:
            {
                return MEF_Sonido.DESCONOCIDO;
            }
    }
}

private void packetToBean(Packet_t packet)
{

}
private void printImage()
{

    switch (fotoRandom.Next(23))
    {
        case 0:
            {
                Visual.visualArbol(this.pictureBox1);
            } break;
        case 1:
            {
                Visual.visualTigre(this.pictureBox1);
            } break;
        case 2:
            {
```

```
            Visual.visualPavo(this.pictureBox1);
        } break;
    case 3:
        {
            Visual.visualOjo(this.pictureBox1);
        } break;
    case 4:
        {
            Visual.visualFlorBlanca(this.pictureBox1);
        } break;
    case 5:
        {
            Visual.visualFlor(this.pictureBox1);
        } break;
    case 6:
        {
            Visual.visualCielo(this.pictureBox1);
        } break;
    case 7:
        {
            Visual.visualAtardecer(this.pictureBox1);
        } break;
    case 8:
        {
            Visual.visualArdilla(this.pictureBox1);
        } break;
    case 9:
        {
            Visual.visualArbol(this.pictureBox1);
        } break;
    case 10:
        {
            Visual.visualFlamingos(this.pictureBox1);
        } break;
    case 11:
        {
            Visual.visualFlorecita(this.pictureBox1);
        } break;
    case 12:
        {
            Visual.visualHombre(this.pictureBox1);
        } break;
    case 13:
```

```
            {
                Visual.visualLuna(this.pictureBox1);
            } break;
        case 14:
            {
                Visual.visualMariflopita(this.pictureBox1);
            } break;
        case 15:
            {
                Visual.visualMariposa(this.pictureBox1);
            } break;
        case 16:
            {
                Visual.visualMariposita(this.pictureBox1);
            } break;
        case 17:
            {
                Visual.visualNiño(this.pictureBox1);
            } break;
        case 18:
            {
                Visual.visualPajaro(this.pictureBox1);
            } break;
        case 19:
            {
                Visual.visualPelicanos(this.pictureBox1);
            } break;
        case 20:
            {
                Visual.visualTigre_jaula(this.pictureBox1);
            } break;
        case 21:
            {
                Visual.visualZebra(this.pictureBox1);
            } break;
        default:
            {
                Visual.visualTurtles(this.pictureBox1);
            } break;
    }
}
private void reproducirArpa(MEF_Sonido Sonido)
{
```

```
switch (Sonido)
{
   case MEF_Sonido.PRIMERA:
      {
         Audio.SonidoArpaPrimera();


      } break;
   case MEF_Sonido.SEGUNDA:
      {
         Audio.SonidoArpaSegunda();

      } break;
   case MEF_Sonido.TERCERA:
      {
         Audio.SonidoArpaTercera();

      } break;
   case MEF_Sonido.CUARTA:
      {
         Audio.SonidoArpaCuarta();


      } break;
   case MEF_Sonido.QUINTA:
      {
         Audio.SonidoArpaQuinta();


      } break;
   case MEF_Sonido.SEXTA:
      {
         Audio.SonidoArpaSexta();


      } break;
   case MEF_Sonido.SEPTIMA:
      {
         Audio.SonidoArpaSeptima();


      } break;
   case MEF_Sonido.OCTAVA:
```

```
   {
      Audio.SonidoArpaOctava();


   } break;
case MEF_Sonido.NOVENA:
   {
      Audio.SonidoArpaNovena();


   } break;
case MEF_Sonido.DECIMA:
   {
      Audio.SonidoArpaDecima();


   } break;

case MEF_Sonido.PRIMERA_OCT:
   {
      Audio.SonidoArpaPrimera_oct();


   } break;
case MEF_Sonido.SEGUNDA_OCT:
   {
      Audio.SonidoArpaSegunda_oct();

   } break;
case MEF_Sonido.TERCERA_OCT:
   {
      Audio.SonidoArpaTercera_oct();

   } break;
case MEF_Sonido.CUARTA_OCT:
   {
      Audio.SonidoArpaCuarta_oct();


   } break;
case MEF_Sonido.QUINTA_OCT:
   {
      Audio.SonidoArpaQuinta_oct();
```

```
            } break;
        case MEF_Sonido.SEXTA_OCT:
            {
                Audio.SonidoArpaSexta_oct();


            } break;
        case MEF_Sonido.SEPTIMA_OCT:
            {
                Audio.SonidoArpaSeptima_oct();


            } break;
        case MEF_Sonido.OCTAVA_OCT:
            {
                Audio.SonidoArpaOctava_oct();


            } break;
        case MEF_Sonido.NOVENA_OCT:
            {
                Audio.SonidoArpaNovena_oct();


            } break;
        case MEF_Sonido.DECIMA_OCT:
            {
                Audio.SonidoArpaDecima_oct();


            } break;
    }

  }
}

public class CodeMidiEventArgs : EventArgs
{
  public byte midiCode;
  public CodeMidiEventArgs(byte midiCode)
  {
```

```
            this.midiCode = midiCode;
        }
    }

}
```


.

.