Diseño de un sistema de audiometría inalámbrico para niños

UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

Diseño de un sistema de audiometría inalámbrico para niños

Trabajo de graduación presentado por
Sergio Fernando Estrada López
para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala
2013

Vo. Bo. :

(f) _____

MSc. Carlos Esquit

Tribunal Examinador:

(f) _____

MSc. Carlos Esquit

(f) _____

Ing. Julio Vásquez

(f) _____

Dra. Patricia de Muñoz

Fecha de aprobación: Guatemala 3 de Diciembre del 2013

PREFACIO

A los largo del estudio de la carrera de ingeniería electrónica se ha obtenido conocimiento sobre las distintas áreas que la componen. Al final del proceso, llega el momento de aplicar varios conceptos adquiridos durante este tiempo para lograr realizar el diseño de un sistema funcional para el ser humano. Se observó sobre la necesidad de un aparato para realizar audiometrías en escuelas de una forma más rápida. Se planteó diseñar y construir un sistema para realizar audiometrías tonales a un grupo de personas utilizando comunicación inalámbrica.

La idea principal consistió en concluir con un sistema funcional para dos personas, pero tomando en cuenta la capacidad de expansión para un mayor número de usuarios. Debido a que es una aplicación médica se investigó sobre los parámetros básicos que debía de cumplirse y a partir de ese punto se implementó el diseño. Durante el diseño se tuvo la necesidad de recurrir a teorías de circuitos eléctricos, redes de datos y programación en computadora y microcontroladores. Se aplicaron un conjunto técnicas conocidas y algunas que se aprendieron durante este trabajo para hacer posible la finalidad del proyecto.

Este trabajo es el resultado del esfuerzo realizado durante estos años de estudio buscando hacer un objeto funcional para la sociedad. Le dedico este proyecto a mi familia, quienes me han dado su apoyo a lo largo del camino, a mis amigos que me apoyaron y ofrecieron su ayuda para comprender conceptos durante la carrera y a los catedráticos que pusieron su esfuerzo en la orientación de la realización de este trabajo.

ÍNDICE

LISTA DE CUADROS

LISTA DE FIGURAS

RESUMEN

Este trabajo fue desarrollado bajo la modalidad de proyecto de graduación de la Universidad del Valle de Guatemala. Se planteó el objetivo de desarrollar un audiómetro tonal con la capacidad de realizar audiometrías a un conjunto de personas a través del mismo software de control. La primera etapa del proyecto consistió en investigar sobre las audiometrías, reconociendo las formas básicas de su práctica y los tipos diferentes que existen dentro del esquema de audiómetros tonales. Se determinaron las potencias y frecuencias implementadas en las mediciones. Teniendo estos datos se procedió a buscar los componentes que tuviesen la capacidad de funcionar dentro de una misma red y generar los parámetros de medición requeridos.

Se decidió hacer uso de los módulos de comunicación XBee para establecer una red punto a multipunto. A través de un software de computadora programado en C# se generan las señales de las audiometrías hacia el XBee utilizando el puerto serial. Debido a la necesidad de recibir retroalimentación de parte de los usuarios se programó una interfaz que almacene los datos de potencia para cada paciente individual. Los módulos receptores utilizan un microcontrolador PIC18F45K22 para establecer la comunicación con el XBee y generar las señales de salida. Se utilizó la modulación por ancho de pulso, PWM, para generar las frecuencias de las señales. Para aplicar el cambio de potencia se utilizó un potenciómetro digital, el cual es controlado por tres señales del microcontrolador.

I. INTRODUCCIÓN

El oído del ser humano se compone de dos partes principales: el aparato de conducción y el aparato de percepción. El aparato de conducción se encuentra formado por el oído externo y el oído medio. Éste se encarga de transmitir las vibraciones a la parte interna. El oído interno conforma el aparado de percepción, esta parte tiene un carácter sensorial y es donde las ondas son transformadas a impulsos nerviosos. El aparato auditivo tiene un umbral definido por un rango de frecuencias audibles y de intensidad medida en decibeles. Las audiometrías tienen el objetivo de medir la capacidad auditiva de una persona. El método más utilizado para realizar pruebas en escuelas o a nivel general son las del tipo tonal. Las características básicas de estas audiometrías son el envío de tonos pulsantes los cuales se envían por conducción aérea u ósea y se utiliza un número específico de frecuencias. Primero se envía una señal de referencia utilizando la frecuencia de 1kHz a una potencia de 20dB o 30dB, en ambientes clínicos. Luego se cambia la potencia a un nivel más bajo o más alto en dependencia de la retroalimentación percibida por parte del paciente. Al terminar con la frecuencia de 1kHz se realiza el procedimiento para 250Hz hasta llegar a 8kHz. Las frecuencias utilizadas son 250Hz, 500Hz, 1kHz, 2kHz, 3kHz, 4kHz, 6kHz y 8kHz. Los rangos de potencia para los cuales el paciente muestra respuesta indican el grado de pérdida auditiva que tiene.

Este proyecto de graduación consiste en el diseño e implementación de un audiómetro tonal de tipo IV. Las especificaciones del tipo IV determinan que el dispositivo debe de generar pulsos a través de un medio aéreo con dos auriculares y contar con retroalimentación del usuario. Dentro de lo planteado se encuentra la capacidad de implementar un software que pueda controlar a más de un paciente en un momento dado. El objetivo del proyecto radica entonces en un diseño que permita a los evaluadores ejecutar pruebas en escuelas evaluando a grupos de niños. Las pruebas implementadas en escuelas sirven de guía para futuras revisiones, por lo tanto no muestran un resultado final y concluyente para personas que muestren padecimiento de pérdida auditiva.

El diseño contempló una etapa inicial de obtención de información para los aspectos de potencia y frecuencia utilizados. Se realizó una visita a CEDAF para observar la forma en que se realizan las pruebas con los pacientes y conocer sobre las evaluaciones aplicadas en las escuelas del país. Luego de determinar los parámetros pertinentes, se buscó la forma de generar la señalización inalámbrica para el control de múltiples dispositivos. El uso de módulos de comunicación XBee provee varias ventajas para establecer una red inalámbrica. Estos módulos utilizan el protocolo Zigbee y fueron configurados para trabajar en una red punto a multipunto. El nodo central se encarga de enviar la información hacia los esclavos y de recibir información que pertenezca únicamente a los dispositivos anclados a la red. Del lado del módulo maestro se desarrolló un software de computadora en lenguaje C#. Se programaron distintos servicios para poder proveer un control de los usuarios a nivel global e individual. Siguiendo la idea de implementación para una cantidad variada de clientes se estableció el uso de listas para almacenar los datos de cada paciente. De las listas se extrae información para mostrar en pantalla y para almacenamiento en un archivo de hoja de datos. La lista también permite generar una retroalimentación gráfica de los resultados de un usuario seleccionado. El envío de información puede aplicarse a todos los dispositivos esclavos o a uno específico.

Los módulos XBee esclavos se conectan a microcontroladores para poder descifrar los códigos de datos y transformarlos en señales útiles para el circuito externo. Se aprovechó la capacidad del PIC de generar señales PWM para las frecuencias. El módulo PWM utiliza un temporizador interno que depende de la frecuencia de reloj del cristal. Esta característica permite obtener resultados estables en frecuencia. La potencia requirió de circuitería externa, se implementó el integrado AD5220 el cual es un potenciómetro

digital, para variar la amplitud de la señal. Haciendo uso de la teoría obtenida en la primera etapa, se diseño el sistema y las señales de control para obtener las potencias deseadas. El integrado funciona por medio de pulsos para incrementar o disminuir la posición, esto agrega un pequeño retardo en el tiempo de recepción de información hasta la salida final de la evaluación. La necesidad de implementar mediciones diferentes para cada oído, requirió del uso de multiplexores analógicos para enviar las señales al lado esperado.

El producto final desarrollado en este proyecto de graduación logró cumplir con los requerimientos mínimos para un audiómetro tonal del tipo IV. Las mediciones de frecuencia y potencia fueron satisfactorias, donde existió un bajo margen de error. El funcionamiento del sistema completo con dos pacientes demostró resultados positivos. Se diferenció en todo momento las señales propias de cada paciente. El gráfico muestra una representación visual de los resultados de un cliente seleccionado y la capacidad de impresión permite obtener un documento final físico disponible para el paciente. A nivel teórico el sistema cuenta se dejó con la capacidad de establecer mediciones a más de dos usuarios, el número depende de la cantidad de dispositivos XBee posibles en la red.

II. OBJETIVOS

A. OBJETIVO GENERAL

- Realizar un sistema de audiometría tonal inalámbrico de múltiples receptores para niños.

B. OBJETIVOS ESPECÍFICOS

- Diseñar una red inalámbrica consistente de varios dispositivos en modalidad punto a multipunto.
- Diseñar un circuito generador de tonos de múltiples frecuencias y potencias.
- Diseñar un software de computadora para manejo del sistema y obtención de la información de los pacientes.
- Implementar capacidad de almacenamiento de datos para las respuestas de los pacientes.
- Implementar una impresión de los resultados en forma gráfica.

III. JUSTIFICACIÓN

La finalidad de este proyecto es lograr optimizar el proceso de la práctica de audiometrías en las escuelas. La pérdida auditiva en niños puede repercutir en problemas de desarrollo como persona y mostrar un bajo rendimiento escolar. Las audiometrías realizadas en escuelas permiten evaluar la capacidad auditiva de los alumnos. Se implementa una cantidad limitada de frecuencias y potencias para los tonos. En base a los resultados obtenidos se toma la decisión de citar al estudiante para realizar pruebas exhaustivas y determinar con exactitud el grado del problema, además de demostrar si es un tema de pérdida auditiva central o periférica. Actualmente estos exámenes se implementan por medio de aparatos portátiles capaces de generar los tonos adecuados. Este examen se aplica a un estudiante a la vez, los datos obtenidos de las pruebas son ingresados manualmente a una computadora.

En este proyecto se busca implementar las pruebas utilizando un sistema de comunicación inalámbrica. Una computadora contiene el software de control, el cual permite observar los parámetros del usuario ante los tonos de control. De esta manera es posible aplicar el examen a un conjunto de personas en un mismo intervalo de tiempo. Los datos pueden ser almacenados en una hoja de cálculo y de igual manera generar el resultado del examen utilizando gráficas. Aplicando este sistema se puede reducir el tiempo de evaluación total de un grupo de personas y poder determinar a un paciente con problemas auditivos rápidamente.

IV. MARCO TEÓRICO

A. SONIDO

La definición de la palabra sonido depende del contexto sobre el cual se esté utilizando. A nivel físico es el movimiento de partículas en el aire o en medios elásticos. En psicofisiología el sonido se define como una sensación causada por la recepción de la energía a través del sistema auditivo humano.(Quirós, 1981).

El sonido tiene tres propiedades principales que son la frecuencia, la intensidad y la complejidad en sobretonos. La frecuencia es el número de ciclos en un segundo de una señal, medida en Hertz. La frecuencia genera sobre el oído la sensación de tonos agudos o graves, una frecuencia alta generará un sonido agudo mientras que una frecuencia baja provee sonidos graves. La frecuencia se conoce como altura tonal dentro del campo de la psicofisiología. (Quirós, 1981)

La intensidad es la energía que contiene la señal, la cual está dada por la potencia. La potencia de una señal depende de la amplitud de la misma. La intensidad física se refiere a la cantidad de energía que se propaga de manera perpendicular a la dirección de la onda. La intensidad auditiva es la relación entre la intensidad física y la intensidad mínima audible por el oído.

La complejidad de sobretonos se denomina timbre en psicofisiología. El timbre se refiere a los armónicos que se producen junto a la frecuencia central de una señal. Este parámetro da la sensación de escuchar diferentes tonos para una misma frecuencia reproducida por distintos instrumentos. Los armónicos tienen menor intensidad acústica que el tono fundamental.(Quirós, 1981)

B. POTENCIA

Existen diferentes tipos de potencia, ya sea eléctrica, acústica o de sonido. En las mediciones de audiometrías se implementa la potencia del sonido SPL.

1. Potencia SPL. La potencia de señales de audio se mide por el nivel de presión que ejerce y se mide en escala de dB. Esta es una medición logarítmica de la presión del sonido sobre una superficie. La ecuación básica para esta presión se define de la siguiente forma:

$$Lp = 20log\left(\frac{P}{Pref}\right) \quad (1)$$

La presión de referencia se ha establecido como 20uPa. Este valor está definido como el umbral del oído humano. El valor de 0dB implica que la presión medida es igual a la presión de referencia. Por lo tanto la potencia de 0dB no implica la ausencia de sonido, es posible tener potencias negativas, dicho estado se presenta cuando la presión medida es menor a la de referencia. (University of Notre Dame)

El ser humano no tiene un sistema de audición igual para todas las frecuencias. Esta razón es la que impide que las personas escuchen al mismo nivel una frecuencia de 250Hz a una de 200kHz. (University of Notre Dame)

2. Cálculo de la potencia SPL para audífonos.

Para el cálculo de la potencia en escala SPL se debe de considerar varios factores. A continuación se listan los parámetros utilizados.

- V: Voltaje de salida de la señal
- R: Impedancia de los audífonos
- P: Potencia en watts de la señal
- Lv: Sensibilidad de los audífonos a 1 vrms
- L: Sensibilidad de los audífonos a 1 mW.
- Lp: Nivel de presión de sonido

Los datos del audífono pueden conseguirse de las especificaciones dadas por el fabricante. La ecuación especifica la sensibilidad L.

$$L = Lv - 10\log\left(\frac{1000}{R}\right) \tag{2}$$

La potencia en Watts es la relación de voltaje por corriente, al no contar una un valor de corriente se utiliza su forma alterna utilizando la ley de Ohm.

$$P = \frac{V^2}{R} \tag{3}$$

Finalmente se puede obtener la potencia SPL.

$$Lp = L + 10log(1000P) \tag{4}$$

C. FSIOLOGÍA DE LA AUDICIÓN

El órgano del oído se compone de tres partes: el oído externo, el oído medio y el oído interno. El oído externo y el medio recogen las ondas sonoras y las conducen al oído interno, donde excitan a los receptores de origen del nervio auditivo. (Rouviére, 1991)

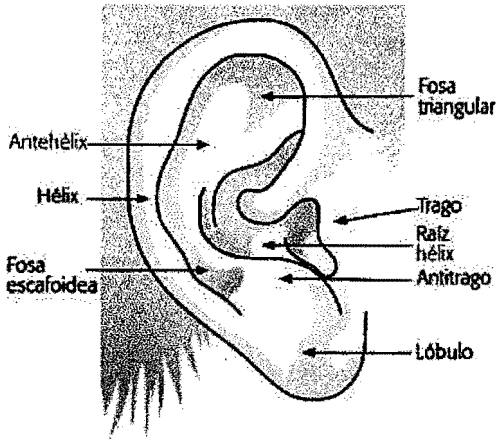
1. Oído externo

a. Pabellón. El pabellón de la oreja es una lámina cartilaginosa plegada sobre si misma en diversos sentidos, de forma oval, con la extremidad mayor hacia arriba y cubierta por la piel. Su forma se encuentra destinada a recoger las ondas sonoras y dirigirlas hacia el conducto auditivo externo.(Rouviére, 1991)

El pabellón de la oreja está situado en las partes laterales de la cabeza, por detrás de la articulación temporomandibular y de la región parotídea, por delante de la región mastoidea y por debajo de la región temporal. (Rouviére, 1991)

Se encuentra unido a la pared lateral de la cabeza por la parte media de su tercio anterior y es libre en el resto de su extensión. Esta parte libre forma con la pared craneal un ángulo cefaloaricular abierto hacia atrás. Cuya abertura, muy variable, mide por término medio 30°. (Rouviére, 1991)

Figura 1. Pabellón auricular(Poch, 2005)

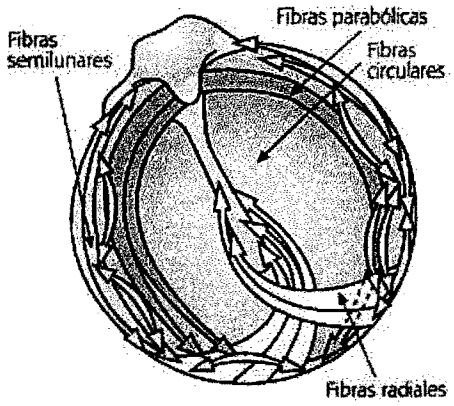


Presenta el pabellón para su estudio dos caras y una circunferencia. En su cara externa el pabellón presenta una serie de repliegues y depresiones, mucho menos marcados en la cara posterior. La hoja fibrocartilaginosa que constituye su esqueleto se incurva en la porción anterior para constituir el repliegue del trago y toma la forma de un cilindro incompleto que contribuye a la formación de la porción cartilaginosa del conducto auditivo externo, CAE. La parte inferior del pabellón está compuesto por el lóbulo de la oreja, constituido solo por piel y grasa. En la cara externa la piel es muy adherente sin panículo adiposo con muy poco pelo, a excepción de la cara interna del trago. La vascularización del pabellón es muy rica a partir de dos pedículos auriculares, anterior y posterior, procedentes de la carótida externa, que explican la hemorragia profusa en las heridas. (Poch, 2005)

b. Conducto auditivo externo. El conducto auditivo externo se extiende desde la excavación de la concha a la membrana del tímpano. La pared de este conducto, fibrosa y cartilaginosa en su tercio externo, está cubierta en toda la extensión de su superficie interna por un revestimiento cutáneo que es continuación de la piel de la cara externa el pabellón. (Poch, 2005)

c. Membrana timpánica. Es una fina membrana transparente, nacarada y brillante de aproximadamente 85mm^2 de superficie, en la que se distingue una región superior o par fláccida, por encina de los ligamentos del martillo y el resto se conoce como pars tensa. La pars fláccida está constituida por una capa epitelial directamente adherida a la capa mucosa, mientras que la pars tensa tiene además una capa intermedia en la que se distinguen fibras circulares, radiales y arciformes, responsables de su particular rigidez y sus capacidades vibratorias. En la pars tensa se aprecia el mango del martillo, firmemente adherido a la capa fibrosa. (Poch, 2005)

Figura 2. Membrana timpánica(Poch, 2005)

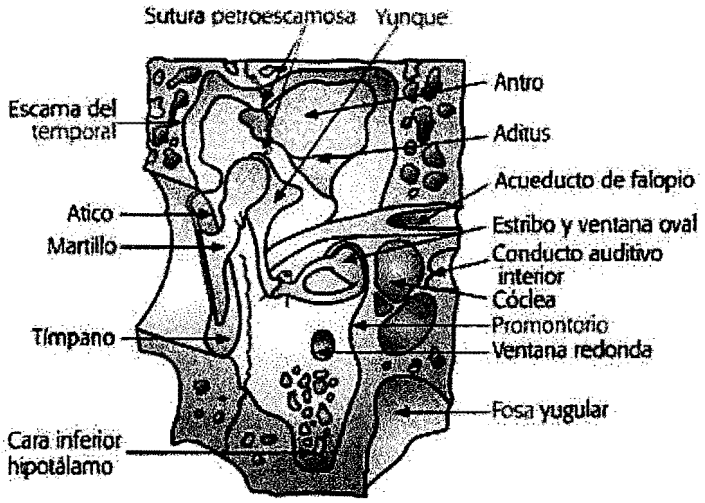


d. Función del oído externo en la audición. El pabellón de la oreja funciona como un receptor de sonidos: su ligera orientación hacia adelante, sus repliegues circulares, su forma en segmento de concha, concentran los sonidos y los dirigen hacia el conducto auditivo. Este último aumenta la concentración de las ondas sonoras, actuando como un resonador. El pabellón permite localizar el origen de los sonidos, o cuando menos, orienta sobre la dirección que han seguido para llegar hasta la oreja.(Rouviére, 1991)

2. Oído medio. El oído medio está formado por un conjunto de cavidades llenas de aire, en la que se consideran tres porciones: la caja del tímpano, la trompa de Eustaquio y las cavidades mastoideas. La caja del tímpano comunica hacia adelante con la rinofaringe y con las vías respiratorias por medio de la trompa de Eustaquio; también comunica hacia atrás con una serie de divertículos, las cavidades mastoideas, desarrolladas en el espesor de la porción mastoidea del temporal. Trompa de Eustaquio, caja del tímpano y cavidades mastoideas siguen una dirección paralela al eje mayor del peñasco. (Rouviére, 1991)

a. Caja del tímpano. Es una cavidad labrada en el centro del peñasco. Tiene la forma de un paralelepípedo irregular en el que se distinguen seis paredes. (Poch, 2005)

Figura 3. Caja de tímpano y cadena timpanosicular(Poch, 2005)



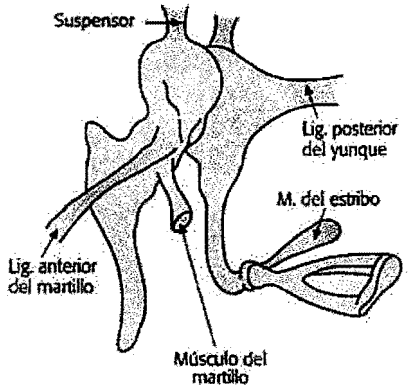
La pared externa se encuentra ocupada fundamentalmente por la membrana timpánica, anclada en el surco timpánico con forma de anillo abierto hacia arriba. Las porciones de la caja por encima del tímpano constituyen el ático, que alberga la articulación del yunque y el martillo; por debajo se encuentra el hipotímpano y la región comprendida entre los límites timpánicos se conoce como mesotímpano. (Poch, 2005)

La pared interna en su porción posterior contiene las ventanas ovales, y una región esferoidal y abombada conocida como promontorio, que corresponde a la primera vuelta de espira del caracol. Toda la cara interna esta recorrida por el segmento horizontal del conducto de Falopio, que forma un relieve a modo de visera sobre la ventana oval. (Poch, 2005)

La pared anterior es la menos desarrollada y está ocupada en su mayor parte por el orificio timpánico de la trompa de Eustaquio, que por debajo se relaciona como la porción vertical del conducto carotídeo. La pared superior está formada por el tegmen timpani, que constituye a este nivel el suelo de la fosa cerebral media y esta recorrido por la sutura petroescamosa. La pared posterior es la pared más alta de la caja del tímpano y la comunica a través del *additus ad antrum* con las celdas de la apófisis mastoides. La pared inferior corresponde a la región del golfo de la yugular. En esa zona se encuentra el orificio del nervio de Jacobson que da inervación vegetativa a la caja. (Poch, 2005)

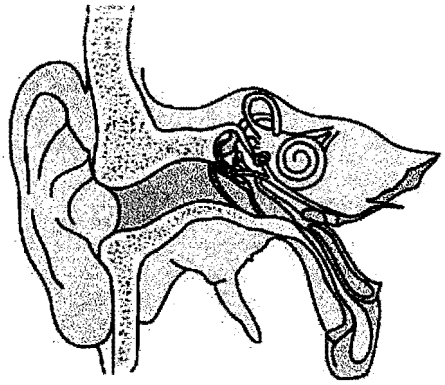
b. Cadena de huesecillos. El martillo, yunque y estribo forman la cadena osicular que trasmite las vibraciones sonoras desde el tímpano al oído interno a través de la ventana oval. Están anclados a las paredes de la caja a través de ligamentos flexibles, su movilidad puede quedar limitada por la acción de los músculos del martillo y por la del estribo. (Poch, 2005)

Figura 4. Articulación de los huesillos, ligamentos y músculos (Poch, 2005)



c. Trompa de Eustaquio. Es el conducto que une la caja del tímpano con la rinofaringe. Su tercio externo es óseo y está constituido por el peñasco y por el hueso timpanal. Sus dos tercios internos o faríngeos son fibrocartilaginosos. En el recién nacido la trompa es más corta y horizontal, lo que tiene mucha importancia en la patología del oído medio a esas edades. En la trompa cartilaginosa se insertan dos músculos, el tensor y el elevador del velo del paladar, cuya acción sinérgica en el momento de la deglución permite la apertura de la trompa que en condiciones de reposo esta ocluida. (Poch, 2005)

Figura 5. Corte sagital del oído(Poch, 2005)

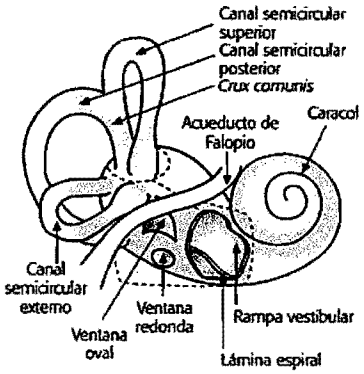


d. Mastoides. Se comunica con la caja del tímpano a través del *additus ad antrum.* Puede ser considerada como una pirámide truncada, dentro de la cual podemos observar múltiples celdillas neumáticas muy irregulares. Su cara superior continúa el techo de la caja y se relaciona con la fosa media, la posterior se relaciona con el seno lateral, la anterior con la pared posterior del conducto y la inferior se corresponde con la punta de la mastoides, donde se insertan por fuera el esternocleidomastoideo y en una ranura con su nombre el músculo digástrico. Todas estas relaciones tienen importancia clínica porque determinan las vías posibles de expansión de los procesos supurados de la mastoides. (Poch, 2005)

3. Oído interno. El oído interno está situado en el espesor del peñasco, por dentro de la caja del tímpano. Comprende: el laberinto óseo, compuesto por distintas cavidades que comunican entre sí y el laberinto membranoso, formado por cavidades de paredes membranosas, contenidas dentro del laberinto óseo. Del laberinto membranoso nacen las vías nerviosas acústicas y vestibulares. Las cavidades del laberinto membranoso están llenas de un líquido llamado endolinfa. Se llama espacio perilinfático a las cavidades del laberinto óseo que no llena completamente el laberinto membranoso. Está lleno de un líquido análogo a la endolinfa que se conoce con el nombre de perilinfa.(Rouviére, 1991)

a. Laberinto óseo. Se distinguen tres regiones: los canales semicirculares, el vestíbulo y el caracol, también llamados laberinto posterior, medio y anterior por sus disposición según el eje antero-posterior.

Figura 6. Laberinto óseo(Poch, 2005)



El vestíbulo tiene la forma de un paralelepípedo muy irregular, en el que asientan dos fositas para el sáculo y el utrículo. A través de su cara externa se relaciona con el oído medio y por la interna se une con el fondo del conducto auditivo interno (CAI), por lo que está perforada para dar salida a los filetes de los nervios sacular y utricular. (Poch, 2005)

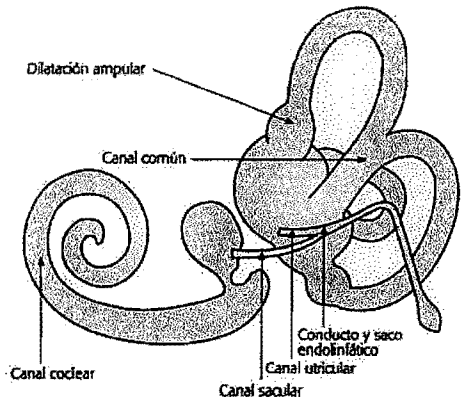
Los canales semicirculares son tres formaciones tubulares que describen algo más de un semicírculo y están situados en los tres planos del espacio. El horizontal o externo forma un ángulo abierto hacia delante de 30° respecto al plano horizontal puro, el vertical superior está situado en un plano parasagital y el vertical posterior en un plano frontal. Las dos extremidades de cada canal se abren en el vestíbulo, presentando una de ellas una dilatación o ampolla, mientras que la otra es de tipo tubula e independiente en el horizontal y están fusionadas en un único canal, *crux comunis*, en los canales verticales. (Poch, 2005)

Es importante comprender que la disposición de un lado frente a la del otro es especular y se debe recordar que los canales verticales forman una figura ortogonal, de tal modo que ambos canales verticales de cada lado forman entre ellos un ángulo de 90°. (Poch, 2005)

La cóclea, también conocida como caracol óseo, es un tubo enrollado en espiral en torno a un eje sólido o modiolo, alrededor del cual se inserta una cresta ósea (lámina espiral) que divide de forma incompleta el tubo coclear. (Poch, 2005)

b. Laberinto membranoso. Al igual que en el laberinto óseo se distingue una porción anterior o auditiva y una central formada por sáculo y utrículo y los canales semicirculares.

Figura 7. Laberinto membranoso(Poch, 2005)



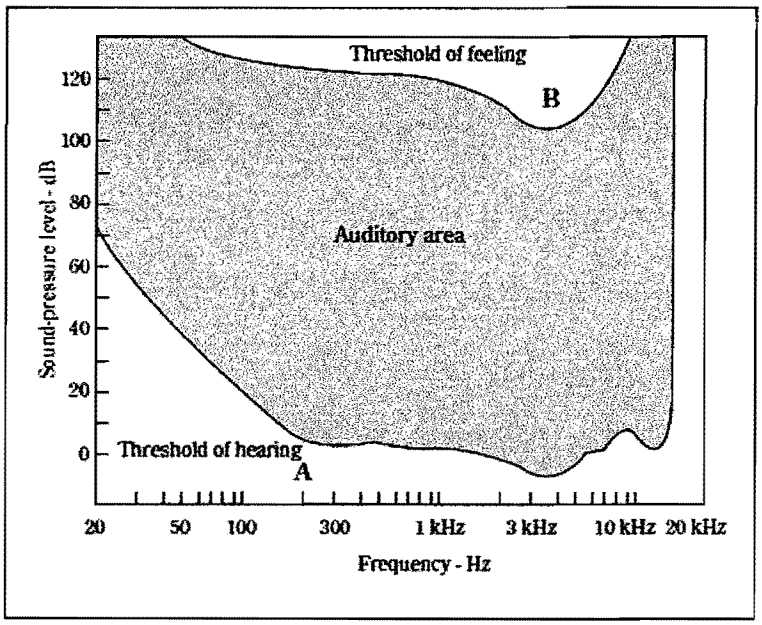
El caracol membranoso se adhiere al óseo, adaptándose a su forma espiral. A la extremidad libre de la lámina espiral se une la membrana basilar (MB) estructura muy compleja de tipo fibroso, mucho más estrecha, delgada y rígida en la base que en la región del ápex, lo que tiene consecuencias funcionales de primer orden. (Poch, 2005)

Los conductos semicirculares tienen la misma configuración que los conductos semicirculares óseos, dentro de los cuales están contenidos. Están bañados por la perilinfa contenida en el espacio perilinfático. Estos conductos desembocan en el utrículo por orificios que tienen la misma configuración, situación y relaciones entre que las de los conductos semicirculares óseos. (Rouviére, 1991)

D. AUDIOMETRÍAS

1. Curva de audición. El umbral auditivo del ser humano se encuentra representado por la Figura 8

Figura 8. Umbral de audición(Everest, 2001)



Como se puede observar se cuentan con dos límites denominados A y B. El límite inferior se refiere al nivel de potencia SPL mínimo audible para la frecuencia dada. Se puede observar que a medida que las frecuencias son más bajas se requiere de mayor potencia. El otro límite es la curva B, el cual especifica el valor máximo para el cual comienza a causar dolor al sistema auditivo. La exposición a sonidos sobre el límite B por largos tiempos llega a ocasionar lesiones auditivas. El oído tiene mayor sensibilidad al sonido en la frecuencia de 3kHz, en la cual la persona promedio puede llegar a escuchar a 0dB. (Everest, 2001)

2. Pérdida auditiva. La pérdida auditiva puede darse por problemas físicos del aparato, accidentes o envejecimiento. Bajo condiciones normales se especifican los siguientes rangos para la pérdida auditiva:

- 0dB – 20dB: Audición normal
- 20dB – 40dB: Hipoacusia leve
- 40dB – 70dB: Hipoacusia moderada
- 70dB – 90dB: Hipoacusia severa
- Mayor a 90dB: Hipoacusia profunda

3. Tipos de audiometrías. Existen diferentes tipos de audiometrías. La forma más común es la audiometría tonal. El objetivo de esta prueba es determinar el umbral para cada frecuencia. Se generan tonos puros que son percibidos por el paciente por conducción aérea o por conducción ósea. (Goycoolea, 2003) En la vía aérea se utilizan las frecuencias de 250Hz, 500Hz, 1kHz, 2kHz, 3kHz, 4kHz, 6kHz y 8kHz. Se inicia con la medición a 1000Hz y se incrementa hasta llegar a 8000Hz. Luego se evalúan las frecuencias bajas de 500Hz y 250Hz. Al finalizar se evalúa

nuevamente 1kHz. En la conducción ósea se realiza de forma ascendente con 250Hz, 1kHz, 2kHz y 4kHz. Se evalúa 1kHz nuevamente para finalizar la prueba. (Mansilla, 2013)

La audiometría tonal evalúa la sensibilidad de recepción cuando una señal se transmite a través del oído externo, medio e interno hasta llegar al cerebro para las frecuencias descritas anteriormente. En la vía aérea pueden utilizarse transductores circumaurales, supra aurales o de inserción. Los auriculares circumaurales tienen contacto con la cabeza, cubriendo todo el oído y reducen el ruido de ambiente. Los supra aurales descansan sobre la oreja y no reducen el ruido de ambiente. Los transductores de inserción tienen la forma de un tubo y son insertados en el oído. Los de inserción reducen el ruido externo y los estímulos auditivos en el oído no evaluado causados por la transmisión del cráneo, para evitar este comportamiento se utiliza el enmascaramiento. La técnica de enmascaramiento consiste en enviar una señal constante al oído que no se está evaluando con el objetivo que únicamente el oído a ser evaluado responda al estímulo. (Kutz, 2013)

La logoaudiometría, también conocida como audiometría verbal, consiste en medir la habilidad de un paciente para reconocer y repetir un listado de palabras que le son dicatadas. El conjunto de palabras tienen características monosílabas, bisílabas y trisílabas. La prueba puede ser evaluada utilizando una grabación de voz o por medio del evaluador dictando las palabras a través de un micrófono hacia los transductores. Debido a que se utiliza la voz, se generan señales en un rango de frecuencias dentro del espectro audible y se tiene una alta correlación con los tonos puros a 500Hz, 1kHz y 2kHz. (Kimball, 2013)

La audiometría a campo libre es un método utilizado para medir la audición en ambos oídos. Consiste de una cámara en la cual se reproducen sonidos sin la necesidad de colocar auriculares al paciente. Es utilizada principalmente con niños y algunas veces implementa reforzamiento visual.(Goycoolea, 2003)

La audiometría de potenciales evocados auditivos utiliza electrodos y una computadora para registrar las ondas eléctricas que viajan de la cóclea a la corteza ante un estímulo sonoro en el oído. Estas pruebas determinan la integridad de la vía nerviosa, no evalúa el nivel de percepción auditiva de la persona. Un resultado positivo en estas pruebas puede considerarse como indicador de audición normal, aunque existen excepciones.(Goycoolea, 2003)

Las emisiones otoacústicas se refieren a los exámenes que miden la función coclear en un rango de frecuencias con una sonda introducida en el oído, ayuda a detectar problemas neurosensoriales. El aparato genera tonos de 55 a 65 dB y captura el eco producido por la cóclea. Este método tiene un carácter automatizado, lo cual permite realizar las pruebas sin la necesidad de la colaboración o retroalimentación del paciente. Esto permite su uso en niños pequeños y suele implementarse en recién nacidos. Es importante resaltar que las emisiones otoacústicas no son una prueba de audición, pero permiten detectar hipocausias. (Denia, 2009)

En una pérdida auditiva conductiva los niveles de conducción ósea se mantienen en sus valores normales, pero se nota una diferencia en la sensibilidad por conducción aérea de al menos 10dB. La pérdida conductiva se debe a problemas en el oído externo y medio. La pérdida neurosensorial es efecto de anormalidades en la cóclea o en los nervios. Las curvas muestran niveles iguales entre evaluaciones ósea y aérea, aunque se muestran por una diferencia de 10dB sobre el nivel normal del paciente. (Kutz, 2013)

4. Tipos de audiómetros tonales según norma ANSI S3.6:1996.

ANSI S3.6:1996 es una norma que clasifica los audiómetros de acuerdo a distintas funciones como lo son el tipo de conducción, frecuencias, ruido, tonos, etc. El Cuadro1 muestra las especificaciones de la norma.

Cuadro 1. Características de audiómetros tonales

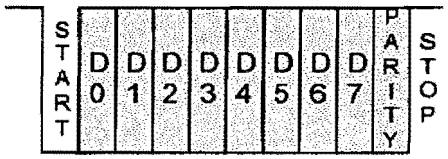
Característica	Tipo I	Tipo II	Tipo III	Tipo IV	Tipo V	Frecuencia extendida
Conducción aérea	si	si	si	si	si	si
Dos auriculares	si	si	si	si		si
Auricular de inserción	si					
Altavoces	si	si				
Conducción ósea	si	si	si			
Ruido de banda estrecha	si	si	si			
Ruido de banda ancha	si					
Auricular contralateral	si	si	si			
Auritular Ipsilateral	si					
Vibrador óseo	si					
Tono pulsado	si	si	si	si	si	
Sistema de respuesta del sujeto	si	si	si	si		
Salida de señal eléctrica auxiliar	si	si				
Entrada para señales externas	si	si				
Comunicación entre operador y sujeto	si					
Supervisión audible	si					

E. UART

UART es una interfaz utilizada para transmitir y recibir información en forma serial de manera asíncrona. El nombre se establece por Universal Asynchronous Receiver Transmitter. El término asíncrono se refiere a la ausencia de una señal de reloj entre los dispositivos. Sin embargo, la estructura del envío de información permite determinar un inicio y un final de la cadena de datos. Es un protocolo de transmisión de datos serial que trabaja a voltajes de 3.3V o 5V. (FTDI Chip)

El protocolo tiene varias características para la transmisión de datos, pero no es necesario implementar todas las propiedades en todas las comunicaciones UART. Las señales TXD y RXD son transmisión y recepción respectivamente. RTS es la señal de petición para enviar datos, por su nombre del inglés Request to Send. Esta señalización la envía el dispositivo hacia otro para conocer si puede enviar el siguiente conjunto de datos. El equipo opuesto responde con un CTS, Clear to Send, indicando que se encuentra libre el enlace y puede recibir nueva información. La señal DTR, Data Terminal Ready, indica que un modem se encuentra conectado en la red y preparado para el intercambio de información. DSR se refiere a Data Set Ready, una señalización de módems indicando que se encuentra preparado para recibir información. DCD son las siglas para Device Carrier Detect, implementado por módems para indicar que la conexión ha sido establecida.(FTDI Chip)

Figura 9. Estructura del paquete UART(FTDI Chip)



El paquete enviado contiene bits de start, seguido de la información, luego bits de paridad y de stop. UART cuenta con soporte para transmisión de 5, 6, 7, 8 o 9 bits, siendo 8 bits la configuración más implementada. El número de bits de stop puede ser 1, 1.5 o 2. El bit de paridad se utiliza para integridad de la información. Se agrega un bit tal que el número de bits con estado 1 en total sea par o impar según la especificación. Uno de los parámetros más importantes es el baud rate. El baud rate especifica la velocidad de transmisión de datos y debe ser idéntico entre los dipositivos. Es importante notar que baud rate no es lo mismo que bits por segundo. Los baudios determinan la velocidad de la transferencia de datos, mide el número de signos por segundo de una transmisión. (FTDI Chip) Utilizando de referencia el ejemplo en el Tech Note TN_111 de FTDI se puede determinar lo siguiente:

$$\frac{bit}{s} = \frac{baud\ rate}{bits\ en\ la\ transmisión} \tag{5}$$

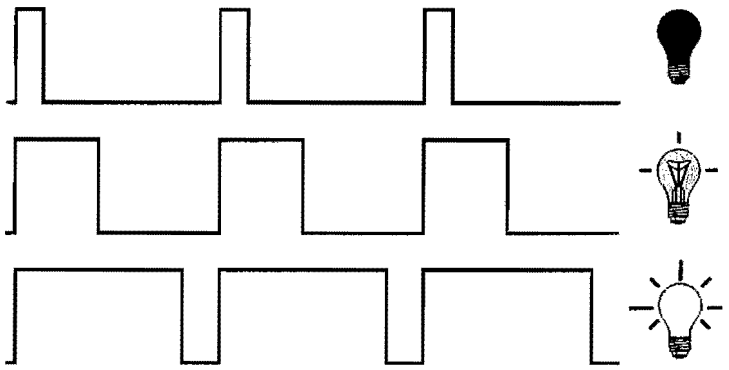
Asumiendo un baud rate de 9600 y un total de 10 bits de transmisión, considerando 1 bit de start, 8 bits de información, no existe paridad y 1 bit de stop. Se tiene el valor en bits por segundo:

$$\frac{bit}{s} = \frac{9600}{10} = \frac{960bits}{s} \tag{6}$$

F. PWM

PWM se refiere a la modulación por ancho de pulso, las siglas provienen del nombre en inglés Pulse Width Modulation. El concepto básico consiste en generar señales periódicas del tipo on/off estables para las que se hace variar el ciclo de trabajo. Esta característica permite hacer uso de las señales PWM en varias aplicaciones.

Figura 10. Variación del ciclo de trabajo en señales PWM.(Verle)



La aplicación más común para la cual se utilizan señales PWM es para el control de servomotores. Estos motores mantienen un estado específico dependiente del ancho de pulso en alto de una señal de control. Debido al control sobre el tiempo en estado alto de la señal, esta modulación puede implementarse para hacer un control analógico de un sistema. A diferencia de un arreglo de resistencias o potenciómetro, no se obtiene la pérdida de energía o sobrecalentamiento de los componentes debido a la potencia. Un incremento en el ancho de pulso en alto genera una salida con mayor potencia, con lo que utilizando un LED éste se vería completamente encendido. Si se disminuye el ciclo de trabajo, el LED no produciría la intensidad de iluminación adecuada. (Verle)

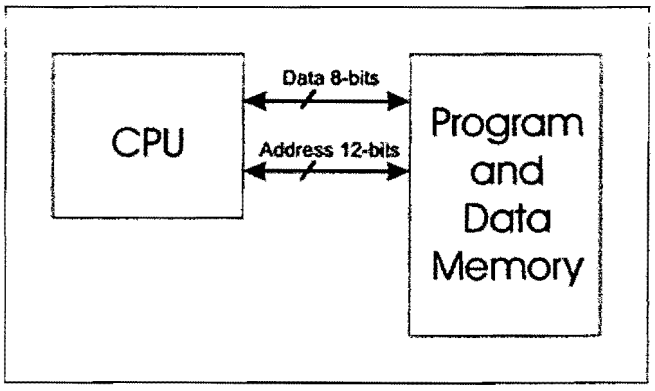
G. MICROCONTROLADOR

Un microcontrolador es un circuito integrado programable, el cual posee todas las partes principales que componen una computadora. Se diferencia de los microprocesadores en la función que cumple como dispositivo. Un microprocesador puede ser programado para múltiples tareas, acoplándose a los módulos necesarios, se conoce como un sistema abierto. Por otro lado un microcontrolador es un sistema cerrado ya que el programa que contiene es propio para una función específica.

1. Arquitectura del microcontrolador. El microcontrolador posee las partes principales de una computadora, estas son: procesador, memoria de programa, memoria de datos y dispositivos de entrada y salida.

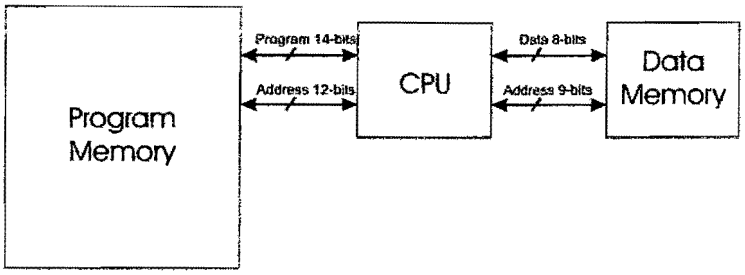
a. Procesador. El procesador es la unidad central que ejecuta las instrucciones programadas. Existen dos estructuras principales conocidas como la arquitectura de von Neumann y la arquitectura Hardvard. La arquitectura von Neumann se caracteriza por tener una memoria única para datos e instrucciones.

Figura 11. Diagrama de arquitectura von Neumann(Hoff)



La arquitectura Hardvard utiliza una memoria dedicada para instrucciones y una memoria dedicada para datos, cada una cuenta con su propio bus de acceso. Esta característica permite la implementación de buses de distinto tamaño para cada memoria

Figura 12. Diagrama de la arquitectura Hardvard(Hoff)



En la actualidad, los microcontroladores implementan la arquitectura Hardvard. A nivel de instrucciones se utiliza la arquitectura RISC, las siglas corresponden a su nombre en inglés: Reduce Instruction Set Computing. La característica importante de RISC consiste en la capacidad de manejar un set de instrucciones definido y pequeño, lo que permite optimizar la cantidad de ciclos de reloj para completar una instrucción. Los microcontroladores tienen la capacidad de aplicar el concepto de pipe-lining. Este método consiste en procesar las instrucciones por etapas, obteniendo la capacidad de procesar múltiples instrucciones al mismo tiempo. Se utilizan 5 pasos para el pipelining que son buscar las instrucciones, decodificar, ejecutar, acceder a memoria y escribir el resultado. Primero se busca la primera instrucción, cuando se encuentra decodificando dicha instrucción, se ejecuta al mismo tiempo la búsqueda de la siguiente instrucción. El proceso continúa de esta forma en cada etapa, con lo cual es posible reducir el tiempo total de ejecución de un programa completo. (Hoff)

b. Memoria de programa. La memoria de programa se encuentra almacenadas todas las instrucciones que constituyen el programa a ejecutar. Esta memoria debe de ser de carácter no volátil, debido a la necesidad del microcontrolador de ejecutar estas instrucciones en todo momento. Existen diferentes tipos de memorias no volátiles que pueden ser implementadas, como se listan a continuación.(Angulo & Angulo, 2005)

- ROM – Read Only Memory: La memoria ROM con máscara es utilizada cuando se aplica el programa durante el proceso de fabricación del controlador. Esta memoria no puede ser modificada posteriormente.

- EPROM – Erasable Programmable Read Only Memory: La memoria EPROM graba los datos por medio de un dispositivo externo a través de una ventana de cristal en la cual se envían rayos ultravioleta para borrarla posteriormente.

- OTP – One Time Programmable Memory: La memoria OTP es un tipo de memoria que puede ser programada una única vez por parte del usuario. Por lo cual es implementada para prototipos finales en donde le programa no volverá a sufrir cambios.

- EEPROM – Electrically Erasable Programmable Read Only Memory: La memoria EEPROM utiliza medios eléctricos para grabar y borrar los datos. Esta memoria puede ser borrada una gran cantidad de veces por lo cual es implementada en los microcontroladores para uso general. Teóricamente puede soportar un máximo de 1,000,000 de ciclos de escritura/borrado.

- FLASH: La memoria FLASH es de bajo consumo y utiliza el mismo medio de escritura y borrado que las memorias EEPROM. El proceso para borrar datos de la memoria es aplicado en bloques completos.

c. Memoria de datos. La memoria de datos es de lectura y escritura, se implementa mayoritariamente la memoria RAM estática o SRAM. En esta memoria se almacena la información de las variables utilizadas a lo largo de un programa. (Angulo & Angulo, 2005)

d. Periféricos de entrada y salida. Son los pines destinados a control de los componentes externos del microcontrolador. Se encuentran agrupados por puertos. Los microcontroladores pueden tener pines dedicados a entrada y salidas digitales como también entrada analógicas. Algunos periféricos se encuentran en la capacidad de comunicarse con características especiales como lo son los módulos de comunicación UART, USB, I2C y PWM entre otros. (Angulo & Angulo, 2005)

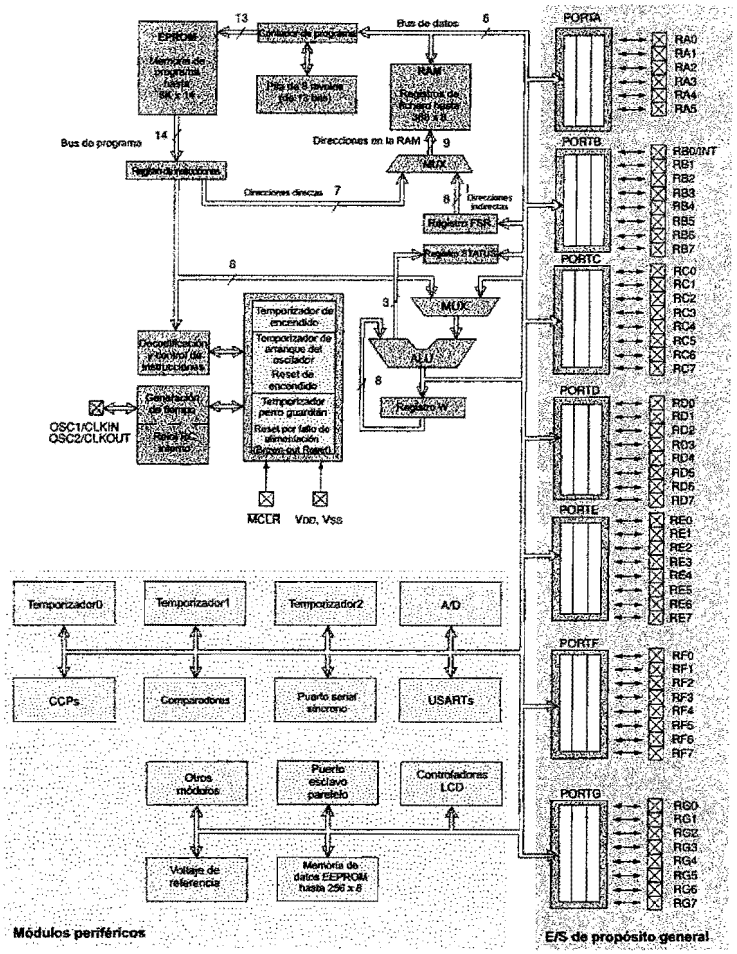
e. Recursos adicionales. Los microcontroladores cuentan con características adicionales que proporciona de mayor flexibilidad para el control de los sistemas. Las interrupciones permiten tener una respuesta inmediata ante un evento específico, como lo puede ser comunicación por algún protocolo con otro dispositivo, cambio de estado de los pines, fin de un contador, conversión analógica, etc. El programa no ejecuta otras instrucciones durante una interrupción. Algunos microcontroladores tienen varios contadores internos, estos timers permiten llevar un tiempo preciso y ser modificados con respecto a la frecuencia del reloj. Se puede contar incluso con sistemas de protección contra fallos eléctricos. (Angulo & Angulo, 2005)

2. Microcontroladores PIC de 8 bits. Microchip Technology desarrolla microcontroladores PIC de 8 bits. Las siglas PIC provienen del nombre original del microcontrolador el cual es Peripheral Interface Controller, también conocido como PICmicro. El primer PIC fue desarrollado por General Instruments, el cual se denominó PIC1650, posteriormente se le agregó una memoria EEPROM. Microchip posee 4 gamas de microcontroladores de 8 bits, de los cuales cada gama tiene características especiales para poder adaptarse a diferentes necesidades de los usuarios.(Verle)

Los PIC16F5X son microcontroladores de la gama baja y utilizan instrucciones de 12 bits. Cuentan con una pila de dos niveles, no poseen capacidad de interrupciones y utilizan voltaje de alimentación de 2.5V. El empaquetado se presenta de 18 o 28 pines. Le siguen los microcontroladores de gama media, denominados por las siglas PIC16FXXX. Estos elementos utilizan instrucciones de 14 bits. Esta gama maneja un amplio repertorio de modelos, los cuales se encuentran de empaquetados de 18 hasta 68 pines. El microcontrolador más utilizado de esta gama es el PIC16F84A. La gama alta se encuentra numerada por PIC17FXXX, tienen instrucciones de 16 bit. Las gamas media y alta tienen la implementación de interrupciones. La gama alta cuenta adicionalmente con periféricos de comunicación serial. (Angulo & Angulo, 2005)

Los PIC18FXXX utilizan instrucciones de 16 bits. Son conocidos como parte de la gama alta. Implementan varias mejoras por parte de Microchip para poder acoplarse a aplicaciones avanzadas como automatización, comunicación y control industrial. El rendimiento fue mejorado, permitiéndole implementar frecuencias de reloj elevadas de hasta 40MHz. Algunos cuentan con capacidad de funcionar de manera efectiva con 3.3V de alimentación.(Angulo & Angulo, 2005)

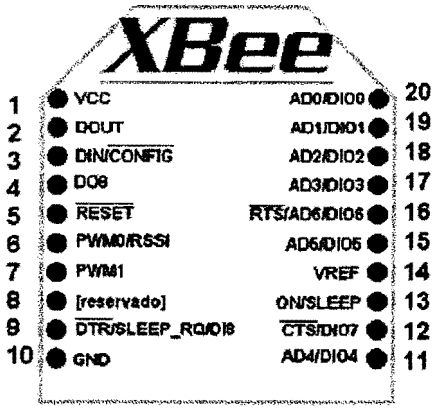
Figura 13. Arquitectura de los PIC de 8 bits.(Verle)



H. MÓDULOS DE COMUNICACIÓN XBEE.

Los módulos de comunicación XBee y XBee PRO son desarrollados por la empresa Digi International Inc. Implementan el protocolo IEEE 802.15.4 en la frecuencia de 2.4GHz. Es una solución de bajo costo y baja potencia para redes inalámbricas. Utiliza el puerto serial para el intercambio de datos con otros dispositivos, como por ejemplo un microcontrolador o una computadora. Cuenta con 7 pines de entrada analógica con capacidad de entrada y salida digital, también incluye dos módulos PWM que pueden ser utilizados para proveer señalización de intensidad de recepción.

Figura 14. Vista superior de los pines del módulo XBee(Oyarce, 2010)



1. Protocolo Zigbee. El grupo de trabajo IEEE 802.15 se dedica a desarrollar los estándares para redes inalámbricas de area personal, WPAN por sus siglas en inglés para Wireless Personal Area Network. Se utilizan subdivisiones basándose en los denominados grupos de trabajo, Task Group como se denomina en inglés. A continuación se presenta una lista de los 7 Grupos de Trabajo:

- 802.15.1: Bluetooth
- 802.15.2: Coexistencia con otros dispositivos inalámbricos.
- 802.15.3: WPAN de alta velocidad.
- 802.15.4: WPAN de baja velocidad.
- 802.15.5: Redes de malla
- 802.15.6: Body Area Networks, implementaciones para dispositivos inalámbricos alocados en el cuerpo.
- 802.15.7: Comunicaciones con luz visible.

El protocolo ZigBee se encuentra clasificado bajo el estándar IEEE 802.15.4 para redes de baja velocidad, desarrollado por la industria de trabajo ZigBee Alliance. La alianza se encuentra definida por un consorcio de más de 150 compañías entre las que se encuentran Motorola, Mitsubishi, Philips, Feescale y Samsung. (Dick) Bajo el objetivo de proveer redes inalámbricas dè bajo costo, consumo y procesamiento, se ha implementado para diferentes aplicaciones como el control industrial, monitoreo médico y ambiental, automatización, entretenimiento, seguridad, emergencias y reportes de desastres. (Callaway)

Cuadro 2. Bandas de comunicación del protocolo Zigbee

Frecuencia	Banda	Cobertura	Ancho de banda	Canales
2.4 GHz	ISM	Mundial	250 kb/s	16
915 MHz	ISM	América	40 kb/s	10
868 MHz		Europa	20 kb/s	1

En capa 2, se utilizan las direcciones MAC para comunicar a los dispositivos. Utiliza dos mecanismos non-beacon y beacon. En configuración non-beacon el receptor se mantiene activo en todo momento. Esta modalidad se implementa cuando los dispositivos se mantienen en constante intercambio de información. En el modo beacon los dispositivos mantienen una señalización para confirmar su presencia en la red. Pero no mantienen la antena activa. Sin embargo requiere de manejo de tiempos para tener comunicación sin errores entre dispositivos. Cada dispositivo cuenta con una dirección MAC de 64 bits única. (Oyarce, 2010)

En la capa de red los módulos implementan dos tipos de algoritmos de ruteo que son en forma de árbol o en malla. Se asignan direcciones de 16 bits, por lo que es posible obtener un tamaño máximo de la red de 65534 dispositivos.(Oyarce, 2010) Las implementaciones de seguridad del protocolo incluye varios niveles en las capas de aplicación, red y enlace de datos. Se hace control de número de secuencia y recibido, aplicación de ID de la red y encriptación de 128 bits AES, conocido como Advanced Encryption Standard.

2. Tipos de dispositivos que componen la red. El protocolo ZigBee diferencia 3 tipos de elementos en la red. Un coordinador, dispositivos routers y dispositivos finales o end devices. La configuración del dispositivo depende de la topología a implementar.

El nodo coordinador es el dispositivo maestro de la red. Tiene como función principal establecer la topología con los demás elementos. Asigna el canal de comunicación a utilizar y el identificador de la red, denominado en XBee como PAN ID. Cumple la función de router para transmitir los paquetes entre los equipos. Un factor importante a considerar es el hecho que este elemento será el encargado de ser el punto de origen y/o destino de toda la información. Por lo tanto puede existir únicamente un coordinador en la red.(Oyarce, 2010)

Los routers son los equipos que distribuyen los paquetes hacia todos los nodos. Inicialmente se encargan de establecer las rutas de mejor acceso para cada destino posible. Estos dispositivos también pueden ser puntos de información según el tipo de red que se implemente.(Oyarce, 2010)

Los end devices son los dispositivos finales de la red. No tienen capacidad para enrutar paquetes, por lo tanto hacen uso de routers o coordinadores para poder trasladar la información hacia el destino pertinente. Al no tener que implementar funciones de ruteo, consumen una menor cantidad de energía. (Oyarce, 2010)

3. Direccionamiento. Los dispositivos XBee soportan direccionamiento para direcciones de 16 y 64 bits. Para el direccionamiento de 16 bits se modifica la variable MY del XBee para definir la dirección propia del dispositivo. El rango de direcciones está compuesto desde 0x0000 hasta 0xFFFE. 0xFFFF es utilizada para habilitar la configuración a 64 bits. Por lo tanto se cuenta con un máximo de 65534 dispositivos dentro de la red. La dirección del módulo al cual se desea transmitir información se aplica al campo DL y se coloca un valor de 0 en el campo DH.

El direccionamiento a 64 bits utiliza 0xFFFF en MY, en este caso la dirección individual de cada módulo es asignada automáticamente utilizando el número de serial impuesto por el fabricante. Los campos DH y DL son de 32 bits cada uno, con lo cual se asigna la dirección del módulo destino. (Oyarce, 2010)

4. Modos de operación

a. Conexión transparente. El modo de conexión transparente consiste en enviar hacia el módulo especificado en la dirección de envío, todos los datos que ingresen a través del puerto RX UART. La información que recibe el módulo la envía por su puerto TX UART hacia el dispositivo que tenga conectado. Se manejan 4 tipos diferentes de configuraciones bajo esta modalidad que son punto a punto, punto a multipunto, broadcast y cable virtual. (Oyarce, 2010)

La configuración punto a punto consiste del intercambio de información entre dos dispositivos únicos en la red. Únicamente se asigna una dirección MY para cada dispositivo y su contraparte en DL. XBee utiliza el Acknowledge ACK para indicar que ha recibido la información, asegurando la transmisión de datos. (Oyarce, 2010)

En la configuración punto a multipunto se utiliza un dispositivo maestro, el cual se comunica hacia uno o más módulos que pertenezcan a la misma red. Para determinar si el dispositivo en cuestión se encuentra bajo la misma red, se modifica el parámetro PAN ID dentro del XBee. Adicionalmente el protocolo ZigBee cuenta con 16 canales para comunicación, todos los dispositivos de la red PMP deben de encontrarse en el mismo canal bajo la frecuencia central de 2.4GHz. La diferenciación de canales permite mantener múltiples equipos que trabajen bajo 2.4GHz realizando operaciones en sus propias redes independientes. (Oyarce, 2010)

El modo broadcast implica una red con la capacidad de enviar información desde un nodo hacia todos los nodos de la red. Con esta configuración no es posible para el transmisor tener conocimiento de la recepción de datos en los otros equipos ya que no se maneja el bit de ACK. Al igual que en protocolos de ruteo, la última dirección disponible es utilizada como el broadcast. Por lo tanto un direccionamiento de 16 bits hace uso de la dirección 0xFFFF en DL para enviar la información. Una capacidad de los XBee es el poder intercomunicar redes individuales entre sí haciendo uso del PAN ID con capacidad de broadcast. El módulo que envíe los datos debe de configurar su identificador con el valor 0xFFFF para hacer broadcast hacia todas las redes PAN. También es posible hacer broadcast hacia un grupo de equipos no necesariamente bajo la misma PAN. Para ello se asigna a DL una dirección arbitraria y el ID con 0xFFFF. El módulo maestro enviará los datos hacia todos los equipos con la dirección especificada bajo DL. (Oyarce, 2010)

El cable virtual permite mantener una comunicación transparente entre diferentes módulos. Los pines se asocian por parejas, por lo tanto la información obtenida en un pin digital de entrada del módulo 1 será visto en el pin par de salida del módulo 2. Para el caso de los pines analógicos, la entrada se aplica hacia un pin analógico y se obtiene su contraparte en los pines PWM. (Oyarce, 2010)

b. Conexión NonBeacon. Los módulos XBee utilizan dos tipos de conexiones bajo las características de nonbeacon, estas son peer-to-peer y con coordinador. En las redes peer-to-peer existe una asociación entre todos los dispositivos, en donde todos pueden ser maestros o esclavos bajo un lapso de tiempo. Se crean conexiones de par en par entre los módulos. Los XBee implementados deben de configurarse como dispositivos terminales, o end device. (Oyarce, 2010)

La configuración nonbeacon con coordinador establece una red del tipo punto a multipunto, en la cual existe un módulo central que administra la red completa. La transmisión puede ser directa o indirecta. La directa consiste en el envío inmediato de la información, la indirecta almacena los datos

por un tiempo especificado previo a su envío. Estos coordinadores permiten establecer la conexión con dispositivos asignándoles el PAN ID y canal a implementar. (Oyarce, 2010)

c. Conexión API. El envío de información utilizando la configuración API encapsula los paquetes de información junto a un frame, otorgando una transmisión y recepción de datos más robusto. El comportamiento se establece a partir del protocolo TCP/IP, en donde el destinatario puede obtener información completa del paquete incluyendo la dirección de origen. La configuración agrega un checksum para verificar la integridad del paquete. Adicionalmente cuenta con propiedades que le permiten obtener información de otro módulo como los datos de asociación y modo de operación. También es posible configurar un dispositivo XBee de forma remota, enviando dentro del paquete los comandos de configuración AT. (Oyarce, 2010)

V. METODOLOGÍA

El diseño del sistema consiste de un software en computadora, el software del microcontrolador, dispositivos de comunicación inalámbrica y el circuito externo. El software de computadora se encarga de enviar las señales correspondientes para cada elemento final para efectuar la audiometría en cada paciente. El microcontrolador es el dispositivo central de procesamiento de cada módulo esclavo. Se encarga de generar los pulsos y las señales de control hacia el circuito externo para obtener las potencias y frecuencias especificadas. Para la comunicación inalámbrica se utilizaron módulos XBee PRO. El circuito externo consiste del conjunto de LEDS de señalización e integrados que permiten al paciente interactuar con el sistema completo.

El sistema se diseñó con el objetivo de soportar N cantidad de dispositivos dentro de la misma red. Por lo tanto a nivel de comunicación inalámbrica entre la computadora y los módulos puede aceptar teóricamente 65534 dispositivos XBee, ya que poseen la capacidad de direccionamiento de 16 bits. El software desarrollado implementa listas y punteros para acceder a los datos de cada usuario, permitiendo el almacenamiento de datos de varios pacientes. Para objetivo de las pruebas se desarrolló el sistema para interactuar con dos módulos receptores.

Las frecuencias utilizadas son 250Hz, 500Hz, 1kHz, 2kHz, 3kHz, 4kHz, 6kHz y 8kHz. El circuito externo permite generar voltajes equivalentes a las potencias de 20dB, 30dB, 40dB, 50dB y 60dB. Este audiómetro está diseñado para efectos de primera evaluación para un grupo de pacientes, y se encuentra dentro del grupo de audiómetros del tipo IV como se observa en el Cuadro1. Por lo tanto los resultados que muestren pérdida auditiva luego de su uso, debe de ser una guía para luego realizar un examen exhaustivo y en ambiente controlado dentro de una clínica.

A. IMPLEMENTACIÓN DE LOS MÓUDLOS XBEE

1. Modalidad de operación

Se implementaron módulos de comunicación inalámbrica XBee. El sistema dispone de un módulo maestro y varios esclavos. El maestro se encuentra conectado hacia la computadora, por lo tanto se encarga de enviar las combinaciones necesarias de datos para que los esclavos puedan reproducir los sonidos a la frecuencia y potencia especificada hacia el oído seleccionado. Es hacia este módulo al cual los esclavos le envían la retroalimentación por parte del usuario. No debe existir comunicación directa entre los esclavos. La configuración punto a multipunto ofrece las características deseadas en donde un elemento central es el punto de inicio y fin de datos. Adicionalmente se cuenta con la posibilidad de escribir sobre el registro DL, siendo posible enviar la información únicamente al módulo especificado en el registro. Esta extensión permite a la aplicación del audiómetro las prestaciones de poder comunicarse únicamente con un paciente, en caso que no se haya recibido registro del mismo ante alguna frecuencia sin la necesidad de que los otros pacientes escuchen nuevamente el tono.

A nivel general de configuración, se estableció el identificador de la red PAN ID con un valor de 585. El direccionamiento es de 16 bits, especificando el registro DH con valor 0 y el DL dependiente de la función del XBee como maestro o esclavo. El reconocimiento de la red en los dispositivos XBee PRO se hace de manera automática, por lo que el maestro asignará la dirección única del esclavo durante la etapa de descubrimiento.

2. Configuración del módulo maestro. El dispositivo central tiene establecida la función de coordinador, específicamente se implementó la opción Coordinator AT, ya que se desea transmisión en modo transparente. El módulo se encarga de obtener la información de la topología de la red y establecer los otros dispositivos que pertenecen a la red. El campo de dirección propia fue establecido con el valor 0x0000. Debido a que este módulo debe de comunicarse con todos los dispositivos de su red tiene establecido el registro DL con 0xFFFF, haciendo referencia a un broadcast.

Para la extensión de comunicación con un único dispositivo, se debe configurar el parámetro del registro DL. Realizar esta acción durante la ejecución regular del programa requiere del uso de los comandos AT. El procedimiento para obtener la dirección de todos los dispositivos consiste del intercambio de datos inicial entre el software de computadora con la rutina de encendido del microcontrolador. La rutina se especifica con detalle en la sección VI.C.5. Flujo del Programa. Este procedimiento es aplicado de la idea del 3 way handshake utilizado en el protocolo TCP, buscando el objetivo de establecer una comunicación inicial entre maestro y esclavo. Durante este proceso el maestro obtiene la dirección del esclavo. Las direcciones son almacenadas por el software de computadora. Se utiliza el comando ATDLxxxx para asignar una nueva dirección en el registro DL, donde xxxx es la dirección de 16 bits del módulo al que se desea enviar la información.

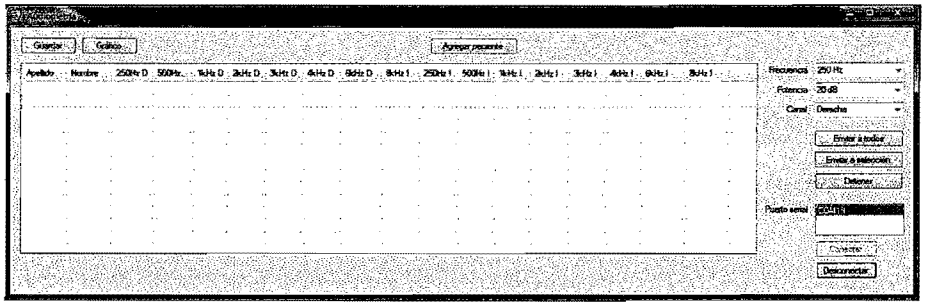
3. Configuración del módulo esclavo. Los módulos XBee configurados para funcionar como esclavos en la red son utilizados para comunicarse con los PIC cumpliendo la función de medio de interacción entre pacientes y examinador. Estos módulos se comunican únicamente con el maestro, por lo tanto todos tienen la misma configuración en su registro DL, donde se asignó el valor 0x0000 en concordancia con el MY del XBee coordinador. La configuración implementada a nivel de tipo de dispositivo, es la de Router AT. Este modo permite mantener la comunicación en modo transparente. Al inicializarse el proceso de descubrimiento de la red, le es asignado un valor para MY automático por parte del coordinador.

B. SOFTWARE DE COMPUTADORA

El software de computadora fue desarrollado en lenguaje de programación C#, utilizando la herramienta Visual Studio 2012. Visual Studio permite crear una interfaz de usuario de manera gráfica, arrastrando los componentes dentro del marco creando automáticamente el código de la GUI.

1. Interfaz de usuario

Figura 15. Interfaz de usuario para envío de señales



La interfaz de usuario posee el elemento listview por medio del cual se muestran los pacientes que son agregados al sistema. Los elementos son almacenados en una lista, utilizando el ID local enumerado de 0 a N, utilizando esta clasificación es posible obtener la información directa de un paciente específico. Del lado derecho se puede observar los campos de selección de frecuencia, voltaje y oído. También se cuenta con las opciones de envío para todos los clientes o para el seleccionado. En la parte inferior se muestra el puerto serial habilitado y las opciones de conectar o desconectar. En la parte superior se dispone de las opciones guardar, gráfico y agregar paciente. La opción guardar, almacena los datos en un formato csv. Gráfico genera una forma externa en la que es posible observar en gráficas el estado de un paciente. La opción de agregar paciente permite agregar un nuevo usuario al sistema, cuando el módulo receptor se encuentra en modo de espera para ser agregado a la red.

2. Comunicación con el módulo XBee.

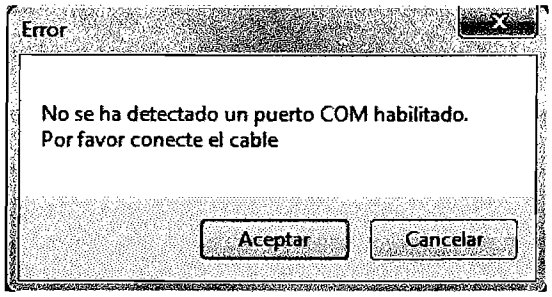
Se implementó la comunicación serial para el intercambio de información con el módulo XBee en modalidad de maestro. Se establecieron los parámetros de paridad, baud rate y bits de fin e inicio dentro del programa, esto implica que no es posible variar dichos valores durante la ejecución del programa. La variable asignada para el control del puerto serial es _serialPort. El módulo XBee maestro cuenta con un adaptador con puerto serial a través de USB. Al conectarlo al ordenador, automáticamente instala el driver necesario para ser reconocido como un puerto COM.

En el inicio del programa, se realiza una búsqueda de todos los puertos COM habilitados en la computadora a través del método BuscaPuerto(). Se hace uso de la propiedad SerialPort de C# para este proceso. Todos los puertos se almacenan en una lista con su respectivo número o identificador. Todos los controles adicionales del programa se encuentran deshabilitados hasta establecer la selección del puerto serial.

```
for (int x = 0; x < (SerialPort.GetPortNames().Length); x++)
   {
PuertoSerial.Items.Add(SerialPort.GetPortNames().ElementAt(x).ToString());
   }
```

En caso que no se encuentre ningún puerto habilitado, se mostrará un mensaje en la computadora por medio la cual el usuario puede determinar si desea hacer la búsqueda nuevamente o cancelar y cerrar el programa.

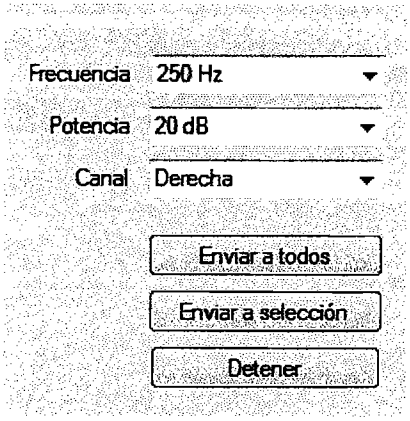
Figura 16. Mensaje de error de puerto serial



Una vez se encuentra establecida la conexión, se debe proceder a obtener la información de los pacientes. Al presionar el botón de agregar paciente, se abre una nueva ventana en la cual se ingresan los siguientes datos de la persona: nombres, apellidos, fecha de nacimiento, colegio y grado cursado. Al presionar aceptar, el programa envía a través del puerto serial el código "{", el cual es reconocido por el PIC como señal de inicialización. Se obtiene la dirección perteneciente al módulo XBee esclavo y posteriormente se asigna un identificador para ser implementado como puntero dentro del programa. Este identificador comienza en cero para el primer usuario y se incrementa con cada usuario adicional. La variable implementada para el índice local se encuentra denominada como índice_id.

Para el envío de cualquier tipo de información se hace uso del siguiente comando _serialPort.Write(DATOS), donde DATOS representa el valor en forma de String que se está transmitiendo. Por esa razón, se envían los números en codificación ASCII que representan cada una de las variables de la evaluación. Se utilizó el componente comboBox para dar las opciones de frecuencia, potencia y oído al usuario.

Figura 17. Conjunto de comboBoxes para opciones de audiometría



Como se especificó anteriormente las frecuencias disponibles son 500Hz, 1kHz, 2kHz, 3kHz, 4kHz, 6kHz y 8kHz. Las potencias utilizadas son 10dB, 20dB, 30dB, 40dB, 50dB y 60dB. El canal toma los valores para oído derecho o izquierdo. Cada vez que el usuario cambia alguno de los parámetros, se llama al método del evento del nuevo valor respectivo. Estos son los tres métodos implementados para cada parámetro:

comboBoxF_SelectionChangeCommitted(object sender, System.EventArgs e)
comboBoxP_SelectionChangeCommitted(object sender, System.EventArgs e)
comboBoxC_SelectionChangeCommitted(object sender, EventArgs e)

Dentro de cada método se hace uso de la comparación switch-case. Se aprovecha el índice de la forma comboBox para implementar las comparaciones. En este punto únicamente se asigna a la variable el nuevo valor según el índice. Debido a que se almacenarán los datos, se utiliza el string del valor real para guardarse en las listas.

```
private void comboBoxP_SelectionChangeCommitted(object sender, System.EventArgs e)
{
      ComboBox senderComboBox = (ComboBox)sender;
      switch (senderComboBox.SelectedIndex)
      {
      case 0:
        p_out = "0";
        potencia = 20;
        break;
      case 1:
        p_out = "1";
        potencia = 30;
        break;
      case 2:
        p_out = "2";
        potencia = 40;
        break;
      case 3:
        p_out = "3";
        potencia = 50;
        break;
      case 4:
        p_out = "4";
        potencia = 60;
        break;
      default:
        p_out = "0";
        break;
      }
}
```

Al presionar el botón para envío de datos se envían las tres variables f_out, p_out y lado utilizando el método del evento BTNtodos_Click(object sender, EventArgs e).

```
_serialPort.Write(f_out);
_serialPort.Write(p_out);
_serialPort.Write(lado);
```

Cuando el usuario ejecuta el botón de enviar selección, se debe configurar el registro DL del XBee utilizando la dirección registrada del módulo esclavo. Se accede al modo de configuración a través del comando +++. Posteriormente se envía ATDLxxxx, donde xxxx especifica la dirección del módulo al cual se desea transmitir, se finaliza con ATCN para salir y guardar cambios. La primera acción antes de reconfigurar la dirección de transmisión consiste de determinar si el módulo al cual se especifica ya se encuentra como destinatario único. Se accede a la lista de información de dispositivos de la red y se compara el nuevo índice con respecto al actual. El siguiente método consigue el valor del ID único del XBee destino.

```
for (int i = 0; i < listView1.Items.Count; i++)
{
  if (listView1.Items[i].Selected == true)
{
     id_temp = listado[i*23];
     hay_seleccion = true;
  }
}
```

La bandera hay_selección verifica que se puede enviar datos y transmite las variables a través del puerto serial. En caso que no se encuentre seleccionado un elemento de la lista en la interfaz del usuario entonces envía mensaje de error.

```
else if ((hay_seleccion == true) && (conf_xbee == true))
{
  _serialPort.Write(f_out);
  _serialPort.Write(p_out);
_serialPort.Write(lado);
  hay_seleccion = false;
}

else
{
  string mensaje = "Seleccione un paciente";
  MessageBox.Show(mensaje);
}
```

La recepción de información se ejecuta a por medio del evento de puerto serial. Esta implementación es el equivalente a una interrupción ya que al momento de recibir la información inmediatamente se dirige a esta línea del código.

```
_serialPort_DataReceived(object sender, SerialDataReceivedEventArgs e)
```

Este método es implementado durante la etapa de reconocimiento de nuevo usuario, en donde la variable handshake con valor true indica nuevo usuario y false implica recepción de retroalimentación de los pacientes. Para la recepción de datos de información sobre la prueba de audiometría se almacena toda la información en un string, La señal de control que indica fin de cadena es el asterisco (*). Todos los datos pasan luego por los métodos de almacenamiento en la lista interna y hacia el almacenamiento en la lista visual. La descripción de la lógica detrás de las listas se encuentra en la sección almacenamiento de datos.

```
if (datos.EndsWith("*"))
{
datos2 = datos.Substring(0, datos.Length - 1); ;
datos = "";
   agregar_respuesta();
  mostrar_respuesta();
}
```

3. Almacenamiento de datos. El almacenamiento de los datos se encuentra dividido en tres partes que son: la lista local, el componente ListView de Visual Studio y una lista local en formato csv. En la lista local principal se almacena la información completa de los pacientes: ID del módulo XBee, ID local, apellidos, nombres, fecha de nacimiento, colegio, grado y potencias a frecuencias de los oídos derecho e izquierdo. En el componente de listados ListView se almacena únicamente el apellido, nombre y las potencias de las frecuencias con el objetivo de tener únicamente la información principal de las pruebas realizadas. La tercera forma utiliza la lista local, pero almacena los datos bajo el formato csv, el cual es un tipo de archivo que separa los elementos por comas, estos datos son utilizados para exportar a un archivo y almacenar los datos.

La implementación de la lista local se encuentra basada en listados de longitud variable del tipo string.

```
List<string> listado = new List<string>();
```

El manejo de toda la información se hace a través de un índice. Este índice es incrementado cada vez que se agrega un nuevo paciente. Adicionalmente se le envía al XBee esclavo para ser almacenado por el microcontrolador y es ese número el primer parámetro de envío hacia la computadora. En la lista se utiliza la siguiente fórmula para determinar la posición en la cual se debe de almacenar datos.

$$Posición = (ID * 23) + 7 + frecuencia + lado \quad (7)$$

ID se refiere al identificador local enumerado desde 0 hasta N. Frecuencia y lado son los datos recibidos por la interacción del paciente y se suma siete para saltar hacia la información general de la persona. Es en esta posición en la cual se almacena el valor de potencia. El método agregar_respuesta() evalúa en primera instancia si la casilla especificada contiene datos o se encuentra vacía. Si se encuentra vacía entonces se inserta el dato. En caso contrario, donde ya existan datos se determina si el dato nuevo es una potencia menor a la actual. De ser afirmativo elimina el valor anterior y coloca el nuevo dato, de lo contrario permanece con la configuración actual. La variable showPot es utilizada para mostrar en forma de texto el valor de potencia.

```
if (listado[indice_temp] == "")
{
  listado.RemoveAt(indice_temp);
  listado.Insert(indice_temp, pot_temp.ToString());
  showPot = pot_temp.ToString();
}

else if (Int16.Parse(listado[indice_temp]) > pot_temp)
{
  listado.RemoveAt(indice_temp);
  listado.Insert(indice_temp, pot_temp.ToString());
  showPot = pot_temp.ToString();
}
else
```

```
{
  showPot = listado[indice_temp];
}
```

Para mostrar datos en la listview, se aplica una ecuación similar a la de listado, pero no considera todos los datos generales del usuario. Esto se debe a que en la interfaz únicamente interesa mostrar el nombre y apellido del paciente junto con los valores de potencia:

$$Posición = 2 + frecuencia + lado \tag{8}$$

El método busca la posición e inserta el valor de potencia utilizando la variable showPot. La variable índice_temp_id identifica el elemento de listview, es decir la fila completa de datos. indice_temp es el índice de posición hacia la derecha del elemento.

```
void mostrar_respuesta()
{
  indice_temp_id = Int16.Parse(datos2.Substring(0,1));
  indice_temp = 2 + Int16.Parse(datos2.Substring(1,1)) + Int16.Parse(datos2.Substring(3,1));

listView1.Items[indice_temp_id].SubItems[indice_temp].Text = showPot;
}
```

El almacenamiento en formato csv se genera cuando se presiona el botón guardar. El método obtiene los datos de listado y les agrega el delimitador de la coma. En la parte superior se incluye un encabezado con los nombres de las columnas. Se utiliza una ventana de diálogo para que el usuario pueda introducir el directorio en la cual quiere guardar la información. Cuando termina la lectura de todas las columnas de una fila, se agrega el delimitador /n, el cual indica que se debe hacer un reinicio de carrete y nueva línea para seguir almacenando los siguientes datos disponibles. El evento del botón guardar, llama al método saveFileDialog1.ShowDialog(), el cual se encarga de todo el procedimiento.

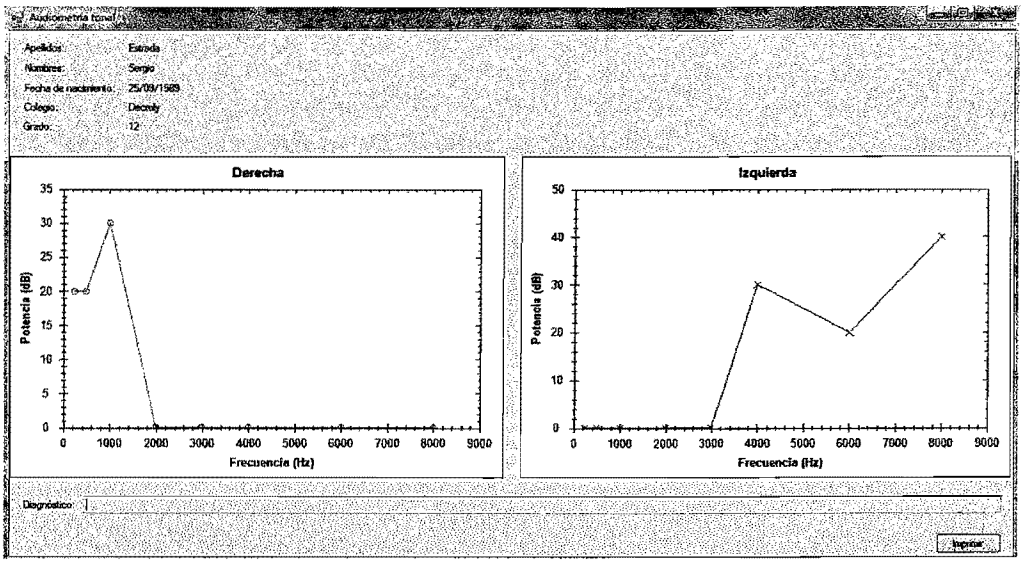
```
st_build.Append(listado[i + num]);
st_build.Append(delimitador);
```

Figura 18. Ejemplo archivo en formato csv para almacenar los datos

A	B	C	D	E	F	G	H	I	J	K
Apellidos	Nombre	Fecha de na	Colegio	Grado	250Hz D	500Hz D	1kHz D	2kHz D	3kHz D	4kHz D
Estrada	Sergio	25/09/1989	Decroly	12	20	20	30	20	20	30
Estrada	Andres	20/10/1989	Decroly	12	20	30	30	40	30	20

4. Gráfico generado. Se agregó una opción para generar de forma gráfica los resultados obtenidos para las pruebas de un paciente seleccionado.

Figura 19. Gráfico generado de la audiometría

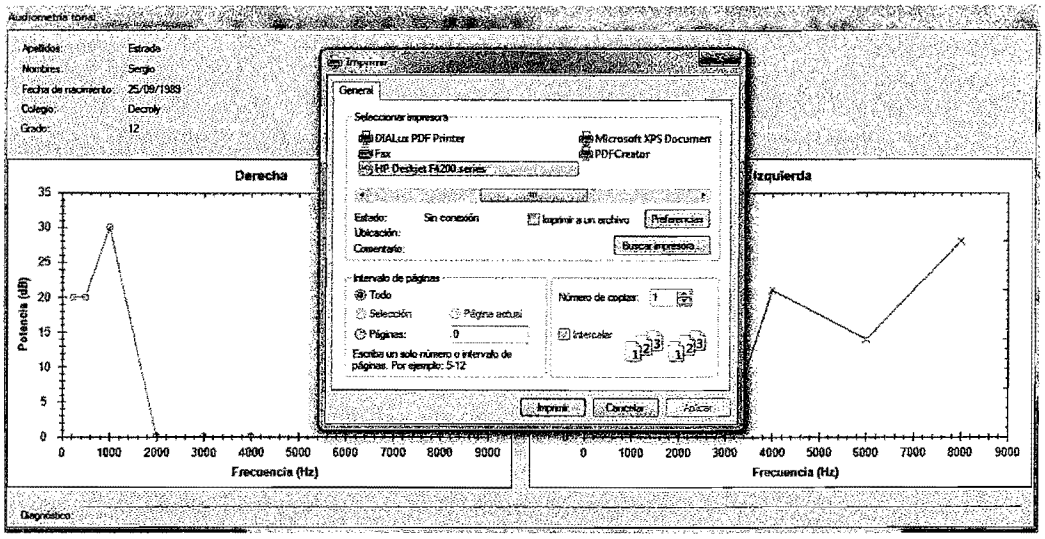


El método button4_Click(object sender, EventArgs e), envía dos listados. La lista listadoGrafica contiene toda la información referente a las potencias y frecuencias a graficar. La lista listadoDatos contiene la información general del paciente. La información para listadoGrafica se obtiene de los componentes de la listView del GUI y listadoDatos obtiene la información de listado. La visualización gráfica permite observar en detalle las potencias que señalizó el usuario para la frecuencia dada. Se utiliza el símbolo O para el oído derecho y X para oído izquierdo, con el objetivo de mantener el estándar de las pruebas de audiometría. Para generar los gráficos se utilizó la librería ZedGraph. Se generó una curva para cada uno, utilizando un método que convierte todos los valores de potencia a su equivalente en número.

```
curva1 = elPanel.AddCurve(null, listaDerecha, Color.Red, SymbolType.Circle);
curva2 = elPanel2.AddCurve(null, listaIzquierda, Color.Blue, SymbolType.XCross);
```

Esta forma incluye un botón para imprimir los resultados. Se genera una imagen de la ventana completa en mapa de bits. Para evitar que salga el botón y los controles de minimizar o cerrar el programa, se genera su bloqueo de vista al inicio. Al finalizar de generar la imagen se apertura un cuadro de diálogo para determinar la impresora a utilizar.

Figura 20. Cuadro de diálogo para impresión



El método de impresión consiste de PrintImage(object o, PrintPageEventArgs e), el cual contiene los recursos para generar la imagen. Cuando el usuario selecciona imprimir en el cuadro de diálogo se procede con el método pd.Print().

```
if (result == DialogResult.OK)
{
  pd.Print();
}
```

C. IMPLEMENTACIÓN DEL PIC18F45K22

Se utilizó el microcontrolador PIC18F45K22, el cual pertenece a los PIC de ocho bits de Microchip. Este microcontrolador cuenta con la característica de operación a 3.3V, por lo que a nivel de circuitería y alimentación no fue necesario implementar reguladores de voltaje para tener las variaciones entre dispositivos. Incluye la capacidad para dos puertos de comunicación UART, generación de pulsos PWM e interrupciones de cambio de estado de puertos y puerto serial. La programación del microcontrolador está desarrollada en el lenguaje MikroC, el cual es un lenguaje de programación basado en C orientado a PICs de Microchip. Es importante la notación del uso de LATn, siendo n el puerto, para encender o apagar un puerto del PIC. Este micrcontrolador utiliza latches en sus puertos, los cuales manipulan el estado físico del puerto en cuestión.

1. Comunicación con módulo XBee.

La comunicación entre el microcontrolador y el XBee se realiza por medio del protocolo UART, como se especificó anteriormente. Los datos que ingresan por la antena del XBee son enviados inmediatamente hacia el microcontrolador. De manera inversa, toda la información serial enviada por el PIC hacia el XBee es luego convertida en el paquete de comunicación ZigBee. Se implementa la interrupción del puerto UART1 de recepción. Para habilitar esta característica, se debe de asignar el valor de uno a los registros INTCON.GIE, INTCON.PEIE y PIE1.RCIE.

GIE se refiere al servicio global de interrupciones, este bit siempre debe ser uno cuando se utiliza cualquier tipo de interrupción. PEIE es la interrupción de periféricos, es decir de cualquier elemento que utilice señalización proveniente de los pines del PIC. RCIE es la interrupción del puerto de recepción de UART. Debido a que se hace uso del UART1 se utiliza el periférico PIE1. La interrupción se habilita luego de la inicialización de reconocimiento del dispositivo dentro de la red. El registro que se lee es PIR1.RC1IF, se lee el contenido de RCREG y se asigna a la variable tempRX, durante este paso se limpia la bandera de interrupción.

```
void interrupt(){
  if (PIR1.RC1IF == 1){
tempRX = RCREG;
```

Se implementa el método para poder obtener los valores de frecuencia, potencia y lado de cada señal enviada por la computadora. La variable bit_control se cambia dependiendo del número de interrupción generada. De esta forma se asegura que el primer dato recibido pertenezca a frecuencia, el segundo a potencia y el tercero al oído. El dato recibido ingresa en formato de código ASCII.

```
if (bit_control == 0x30){
    bit_control=1;
    f_vout = tempRX;
}
```

2. Generación de pulsos utilizando PWM. La frecuencia de los pulsos del dispositivo es generada por el módulo PWM del microcontrolador. Se implementó el ciclo de trabajo del 50%, en MikroC se hace uso de la siguiente ecuación para determinar el parámetro de ciclo de trabajo.

$$Ciclo\ de\ trabajo = \frac{porecentaje*255}{100} \tag{9}$$

Siguiendo la ecuación de MikroC para un ciclo de trabajo del 50%, el resultado es 127.5, en la práctica se implementó el valor 127 dentro del código. El método valor_frecuencia() se encarga de obtener el número decodificado por la interrupción UART y siguiendo un conjunto de if-else, se obtiene la relación directa a utilizar. Se asigna adicionalmente a la variable fout el valor de f_vout para almacenar el nuevo dato implementado. El método PWM1_Init(número) se utiliza para generar la frecuencia.

```
if (f_vout == 0x30){
    PWM1_Init(250);
    fout = f_vout;
  }
```

Finalmente se aplica el método frecuencia() para generar los pulsos durante un tiempo de 250ms, luego se apaga el módulo PWM por 250ms. Este método es llamado hasta que se envíe una nueva señal de la computadora para otra frecuencia, detención del análisis o el usuario indique que ha escuchado el tono.

```
void frecuencia(){
  PWM1_Start();
```

```
  Delay_ms(250);
  PWM1_Stop();
  Delay_ms(250);
}
```

3. Señales de control para generación de potencia. La potencia generada depende del circuito externo, para tener un mejor control del sistema se implementó el potenciómetro digital AD5220. Se requieren de tres señales de control para manipular la dirección que especifica incremento o decremento y habilitación del circuito integrado. Previo a la generación de las señales, se obtiene el valor de potencia proveniente de la computadora. De igual forma que para la frecuencia, se utilizó un conjunto de condiciones if-else para identificar bajo la variable p_pot el valor de la cantidad a incrementar bajo el método valor_potencia().

```
if (p_vout == 0x30){
    pout = p_vout;
    p_pot =3;
  }
```

El método que manipula el circuito externo es ajustaPotDown(int p_pot). Todas las señales son generadas por el puerto D. El pin siete se utiliza para habilitar el integrado, el cual se activa en nivel bajo y desactiva en nivel alto. El pin seis envía las señales de reloj, por lo tanto se coloca en uno lógico durante diez milisegundos y luego se coloca en bajo. Este proceso se realizará el número de veces especificado bajo la variable p_pot. El pin cinco toma valor alto para indicar que la señal es de incremento y en nivel bajo cuando se debe disminuir el valor del potenciómetro. El primer paso para ajustar de manera correcta el potenciómetro, consiste de retornar el potenciómetro a su posición inicial. Se coloca el puerto LATD.F5 en cero y el la variable valor_r se aumenta en cada ciclo hasta superar 130 pulsos.

```
LATD.F5 = 0;
while(valor_r < 130){
    LATD.F6 = 1;
    Delay_ms(10);
    LATD.F6 = 0;
    Delay_ms(10);
    valor_r++;
  }
```

Posteriormente se aplica un método similar pero con la señal de incrementar el potenciómetro y la cantidad de incremento.

```
LATD.F5 = 1;
while(valor_r < p_pot){
    LATD.F6 = 1;
    Delay_ms(10);
    LATD.F6 = 0;
    Delay_ms(10);
    valor_r++;
  }
```

4. **Interacción con el usuario.** Este audiómetro requiere de la interacción con el usuario para conocer el momento en el cual ha escuchado el tono. El PIC18F45K22 cuenta con interrupciones de cambio de estado para los pines del puerto B. Se debe de habilitar la interrupción a través del registro INTCON.RBIE. La bandera de interrupción debe poseer un valor de cero para apagarla. El método recibe el cambio de estado del pin siete. Durante la interrupción se recolecta el valor de fout, vout y d_i para enviar a través del puerto serial los datos de frecuencia, voltaje y lado de la señal actual respectivamente.

```
if (tempRB==1){
  hay_f = 2;
  fout = fout;
  pout = pout;
  d_i = d_i;
  INTCON.RBIE = 0;
}
```

Con el objetivo de evitar interacción por parte del usuario antes del envío de señales por el evaluador, al finalizar el método se desactiva la interrupción y se vuelve a activar luego que ingresa la señal de evaluación. Estableciendo hay_f=2, se genera una condición tal que al salir de la interrupción, el microcontrolador envía todas las señales hacia el puerto serial adicionando un asterisco al final como delimitador de cadena. Al inicio envía su valor de ID2, el cual lo identifica dentro del software de computadora como su índice local.

```
UART1_Write(ID2);
UART1_Write(fout);
UART1_Write(pout);
UART1_Write(d_i);
UART1_Write('*');
```

5. **Flujo del programa.** El programa cuenta con una rutina de inicialización en la cual obtiene en primera instancia la dirección del módulo XBee al cual se encuentra emparejado y reinicia el estado del potenciómetro digital. Para reiniciar el circuito de potencia utiliza el método ajustaPotDown(p_pot), en donde p_pot es la variable de control de incremento de ciclos inicializada en uno.

Se retroalimenta al usuario por medio de dos LEDs. El LED del puerto RA0 se enciende cuando el microcontrolador ha obtenido el MY del XBee, el método utilizado se denomina get_ID(). De esta forma el usuario puede tener conocimiento del momento adecuado para enviar la señal desde la computadora para asignar un nuevo paciente al sistema. La estructura utilizada no implementa en este punto la interrupción del puerto RX de UART. Por lo tanto se utilizaron ciclos para conocer los comandos exactos que ingresan a través del puerto serial provenientes del PIC.

```
UART1_Write_Text("+++");
  while(tempRX != 'O'){
  tempRX = UART1_Read();
}
```

```
while(tempRX != 'K'){
   tempRX = UART1_Read();
}
   while(tempRX != 0x0D){
  tempRX = UART1_Read();
}
```

El primer comando enviado es +++, el cual se utiliza para indicar al XBee que se desea ingresar al modo de configuración AT en lugar de enviar datos por su antena. Se espera que el módulo responda con un OK seguido del retorno de carrete, con el código 0x0D. El siguiente comando a enviar es el de obtención de la dirección MY. El XBee responderá con la salida MYxxxx, donde xxxx representa el código hexadecimal de 16 bits del contenido en MY. Para cerrar la comunicación del modo de comando se utiliza ATCN. El contenido de MY se almacena en el arreglo ID con longitud de cuatro casillas de ocho bits cada una. A continuación se muestra el envío del comando ATMY junto con la lectura del primer nibble.

```
UART1_Write_Text("ATMY");
  UART1_Write(0x0D);
  while(UART1_Data_Ready()==0){

  }
ID[0]=UART1_Read();
```

Un segundo LED fue implementado en el puerto RA1 para indicar que se ha terminado el proceso de intercambio de información inicial entre computadora y microcontrolador. Este LED alternará de estado entre encendido y apagado tres veces al concluir el proceso. El método handshake() se encarga de la manipulación de todas las señales de recibido para avanzar en el proceso de reconocimiento de dirección. El método inicia con la recepción del código {. Este valor indica que el usuario ha enviado la señal desde la computadora para inicializar el reconocimiento de nuevo paciente, al cual le asignará un ID2 de referencia dentro de su lista de contactos. Posteriormente el PIC enviará de retorno el ID de dirección de XBee seguido por la señal de control *. Finalmente el maestro envía el ID2 de referencia.

```
while (x != '{'){
  x = UART1_Read();
  }
UART1_Write_Text(ID);
Delay_ms(1);
ART1_Write('*');
while (UART1_Data_Ready()==0){
}
ID2 = UART1_Read();
```

Al finalizar este proceso, se cambia el valor de la variable denominada existe. Dicha variable permite al PIC identificar que ya ha sido asignado a la red de audiometría, por lo tanto en la interrupción del puerto UART hará omisión de la recepción del comando { para agregar un nuevo usuario. Luego de este punto el PIC ingresa a una modalidad de espera de interrupción UART. Luego de obtener los tres datos que componen la señal de audiometría a evaluar, se habilita la interrupción del puerto B. El programa determina primero la frecuencia del pulso en PWM. Luego se ejecuta el

método de generación de las señales de control para la potencia. Finalmente se evalúa el lado por el cual se debe escuchar. Al terminar estas evaluaciones se generan los pulsos a través del puerto RC2 en PWM.

El programa espera que cualquiera de las siguientes condiciones se cumpla para detener la generación del tono: el usuario pulsa el botón, se recibe una señal de detención por parte de la computadora o se recibe una nueva señal de evaluación.

D. DISEÑO DEL CIRCUITO EXTERNO

El Cuadro3 contiene el listado de materiales para el diseño del circuito impreso utilizado en el módulo receptor.

Cuadro 3. Materiales que componen el circuito externo.

Componente	Valor	Cantidad
PIC18F45K22		1
XBee PRO		1
Adaptador XBee		1
CD4053B		1
AD5220	10kΩ	1
LM358		1
Cristal	4MHz	1
Resistencia de superficie	220Ω	3
Resistencia	10kΩ	1
Potenciómetro de precisión	8024Ω	1
LED de superficie		3
Botón normalmente abierto		1
Capacitor	33pF	2
Capacitor	100nF	1
Socket estéreo	3.5mm	1
Baterías alcalinas	AA	2
Porta baterías AA		1

Los módulos XBee sin adaptador se debe alimentar con 3.3V. En las pruebas realizadas se utilizó el adaptador, con el objetivo de alimentar al circuito con un voltaje de 5V o 3.3V. El diseño del circuito impreso considera el desarrollo del sistema final sin el uso del adaptador, por lo que utiliza una alimentación de 3.3V. Teniendo esta restricción, se buscó la forma de alimentar al circuito con dicho voltaje, eliminando la necesidad de baterías más grandes y reguladores de voltaje. El PIC18F45K22 y los integrados pueden funcionar correctamente en el valor establecido. El circuito es alimentado por 2 baterías AA.

1. **Nivel de potencia con potenciómetro digital.** El integrado AD5220 es un potenciómetro digital de 10kΩ, cuenta con un total de 128 posiciones para variar el valor de la resistencia. Siguiendo la ecuación a continuación se observa que el valor de la resistencia con cada incremento de posición es de 78.125Ω.

$$R = \frac{10000}{128} = 78.125 \quad (8)$$

Se utilizó la ecuación 4 para el cálculo de la potencia en escala SPL. Es importante resaltar que el diseño se encuentra basado en cálculos teóricos. Se resolvió la ecuación 4 para obtener el valor de voltaje requerido en la señal del microcontrolador. Para este voltaje se calculó el valor del divisor de resistencias en el potenciómetro. A continuación se presenta una tabla en la que se relaciona todos los datos. Esta tabla fue implementada para determinar el número teórico de N posiciones necesarias de avanzar desde el 0 para obtener la potencia especificada.

Cuadro 4. Datos necesarios para el cálculo de la potencia

Voltaje de salida	R audífonos	P (W)	Lv db (mW)	L dB (vrms)	E (%)
0.0003	32.0000	3.05E-09	90.0000	75.0515	0.0002
0.0009	32.0000	2.75E-08	90.0000	75.0515	0.0002
0.0031	32.0000	3.05E-07	90.0000	75.0515	0.0002
0.0100	32.0000	3.13E-06	90.0000	75.0515	0.0002
0.0316	32.0000	3.11E-05	90.0000	75.0515	0.0002

Cuadro 5. Cálculo de potencia y valor de potenciómetro

Posición	Potenciómetro	Voltaje base	Lp (dB)	Lp (dB) ideal	%error
1	78.1250	0.0400	19.8970	20.0000	0.5150
3	234.3750	0.0400	29.4394	30.0000	1.8686
10	781.2500	0.0400	39.8970	40.0000	0.2575
32	2500.0000	0.0400	50.0000	50.0000	0.0000
101	7890.6250	0.0400	59.9834	60.0000	0.0276

Se implementó una tabla en Excel para obtener el número de posiciones que se debe incrementar el potenciómetro a partir del cero para lograr la potencia deseada. El voltaje final es el voltaje necesario para generar una potencia específica, este valor depende del divisor de voltaje generado a partir del potenciómetro de precisión conectado hacia el seguidor de voltaje. Este potenciómetro es variado durante calibración inicial por el operador. Se busca un voltaje de salida máximo de 0.04V. A este voltaje se le aplica el potenciómetro digital, con lo que se obtiene la potencia en dB de escala SPL experimental Lp. El valor de posición es el dato programado en el microcontrolador como señal del número de ciclos que se envían hacia el circuito integrado AD5220.

2. **Diseño del PCB.** El diseño del circuito y del PCB para el módulo receptor se desarrolló en Altium Designer 10. El concepto del diseño en placa fue de realizar un circuito pequeño, para ser portable. En los extremos se colocó el botón, el socket para los audífonos y los LEDs de

indicación. Por medio de un portabaterías para 2 baterías AA se alimenta al circuito soldando los cables del dispositivo a sus respectivos agujeros dentro de la placa.

Figura 21. Diseño en PCB del circuito

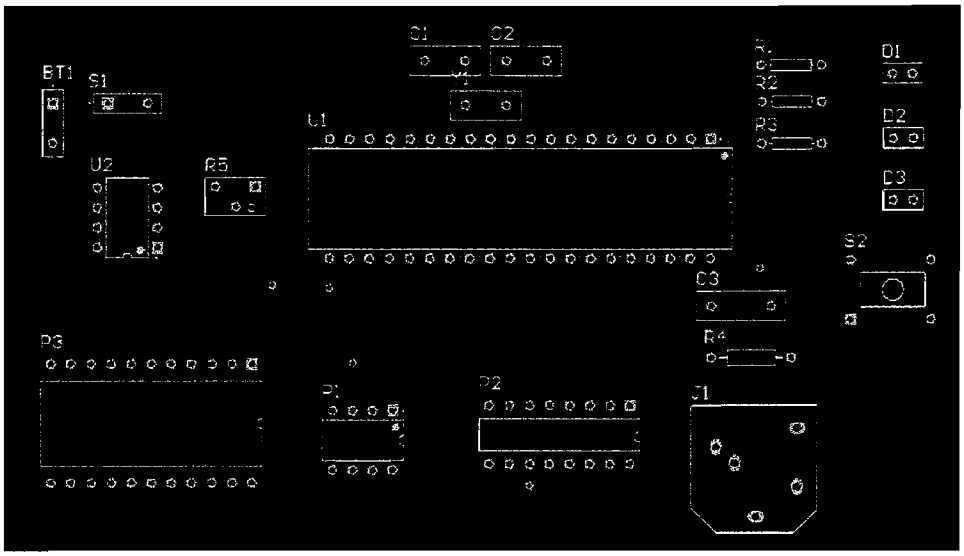
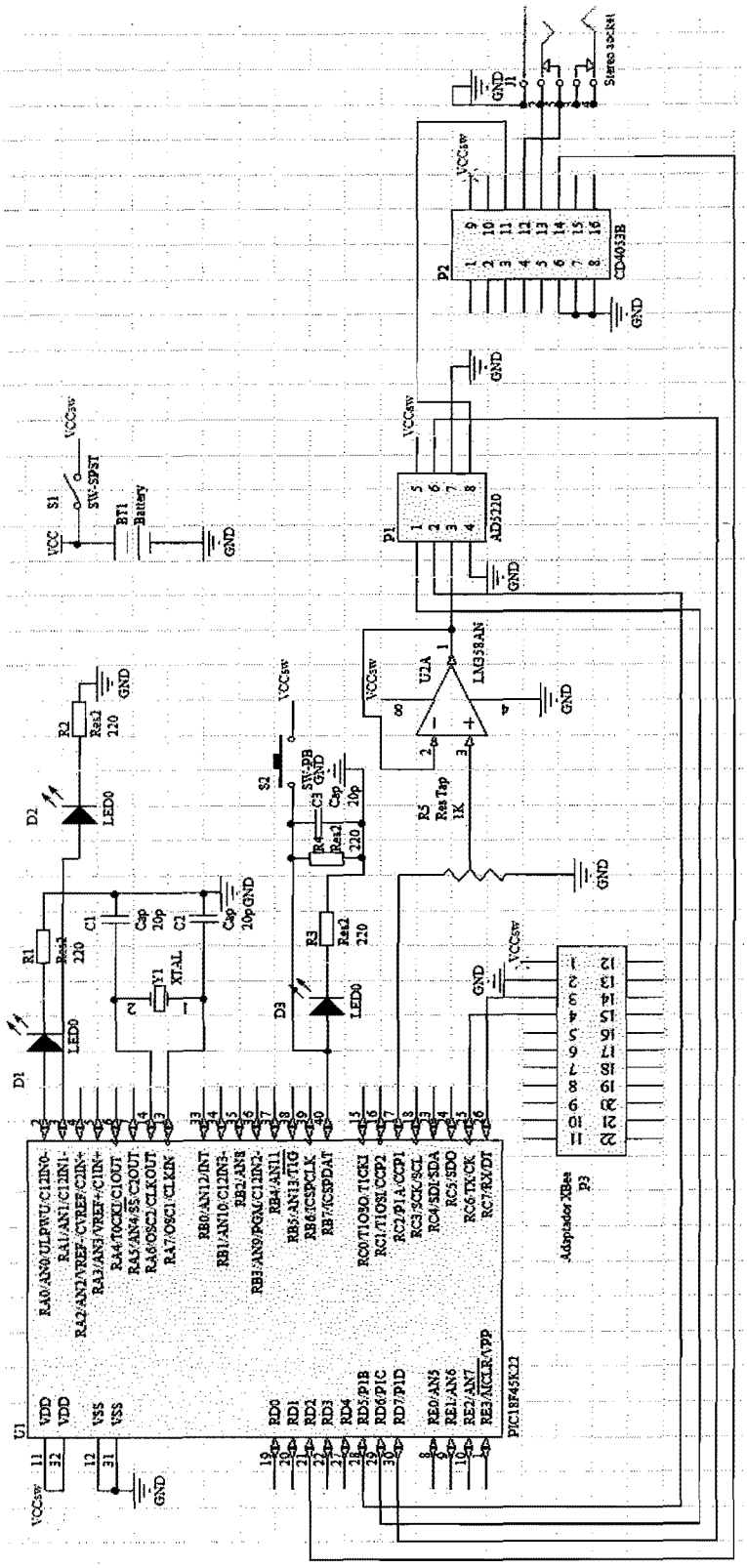


Figura 22. Esquemático del circuito en Altium



3. Potenciómetro de calibración. Se incluyó dentro del diseño del circuito un sistema simple de calibración, el cual consiste de 2 sockets de medición, un botón, un LED en el puerto RA2 y un potenciómetro de precisión. El botón permite ingresar en modo de calibración y el LED de indicación permanecerá encendido. El microcontrolador genera una señal de salida sin frecuencia por el puerto RC2. Esta señal ingresa a un potenciómetro de precisión, el cual funciona como divisor de voltaje. Se estabiliza el voltaje obtenido a través de un amplificador en configuración de seguidor de voltaje. El usuario utiliza un multímetro y coloca las puntas en los sockets para realizar la medición del nivel de salida del seguidor de voltaje. Utilizando un destornillador plano se manipula el potenciómetro de precisión para obtener el nivel esperado. El valor de voltaje que se debe de ajustar es de 40mV, el cual se utiliza como voltaje de base para el potenciómetro digital y generar la amplitud de las señales. Al finalizar con las pruebas, el usuario procede a presionar el botón para retornar a modo de operación normal y el LED se apagará.

VI. RESULTADOS

A. GENERACIÓN DE LOS TONOS

Los tonos fueron generados por medio del módulo PWM del microcontrolador. Las señales transmitidas tienen características cuadradas con ciclo de trabajo del 50%, las cuales fueron generadas por medio del módulo PWM del microcontrolador. Se midieron los resultados en un osciloscopio de las frecuencias de los distintos tonos a la salida del PIC, a continuación se muestran los resultados iniciales de las frecuencias seguidos de la modificación.

Cuadro 6. Comparación de frecuencias originales

Frecuencia deseada (Hz)	Frecuencia medida (Hz)	Porcentaje de error (%)
250	250	0.000
500	500	0.000
1000	1000	0.000
2000	2000	0.000
3000	2976	0.800
4000	4000	0.000
6000	6087	1.450
8000	8068	0.850

Los resultados obtenidos de la primera medición fueron satisfactorios, pero se procedió a variar el valor de la frecuencia especificada para el PWM para las frecuencias de 3kHz, 6kHz y 8kHz, con el objetivo de reducir el error al valor mínimo posible. Se realizaron varias mediciones experimentales y se logró obtener el resultado del Cuadro 7.

Cuadro 7. Comparación de frecuencias medidas con respecto al valor teórico corregido

Frecuencia deseada (Hz)	Frecuencia medida (Hz)	Porcentaje de error (%)
250	250	0.000
500	500	0.000
1000	1000	0.000
2000	2000	0.000
3000	3012	0.400
4000	4000	0.000
6000	6024	0.400
8000	8013	0.163

Observando el Cuadro 7 se puede notar que las mediciones del osciloscopio demuestran que las frecuencias generadas por el microcontrolador se encuentran con un porcentaje de error menor del 1%. Únicamente las frecuencias de 3kHz, 6kHz y 8kHz mostraron valor distinto del 0%.

Figura 23. Tono de 250Hz en la salida del microcontrolador

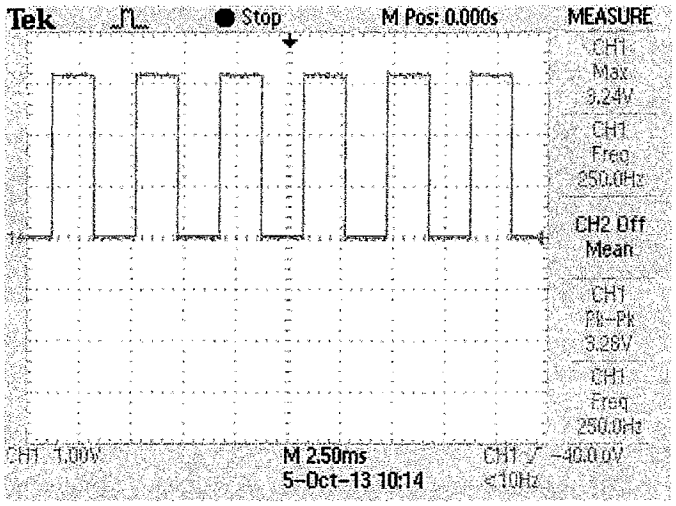


Figura 24. Tono de 500Hz en la salida del microcontrolador

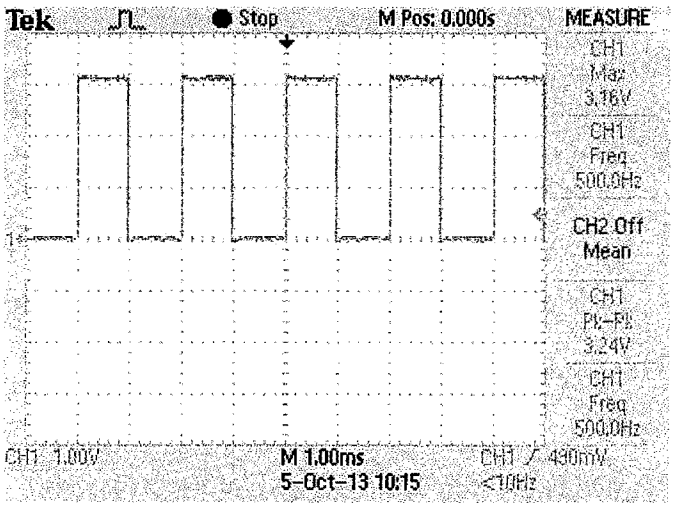


Figura 25. Tono de 1kHz en la salida del microcontrolador

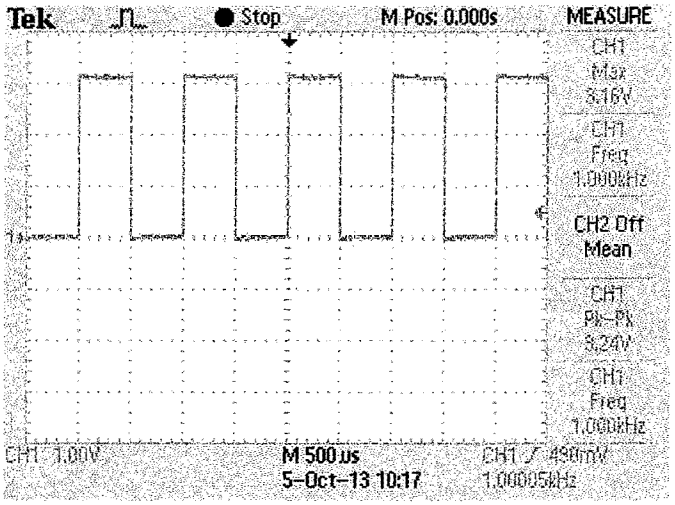


Figura 26. Tono de 2kHz en la salida del microcontrolador

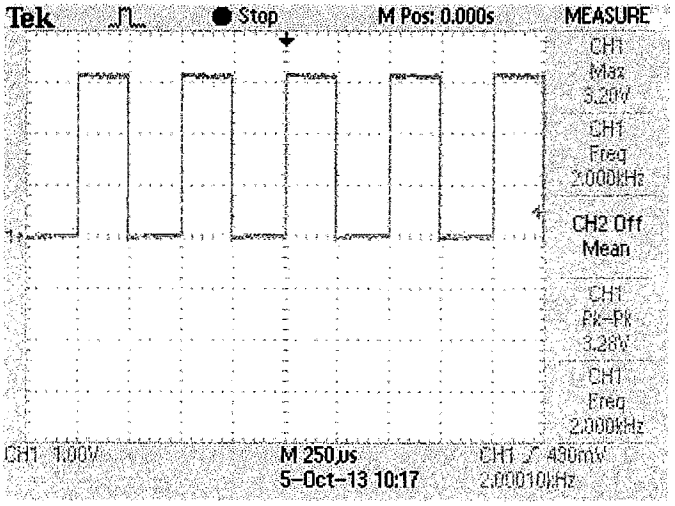


Figura 27. Tono de 3kHz en la salida del microcontrolador

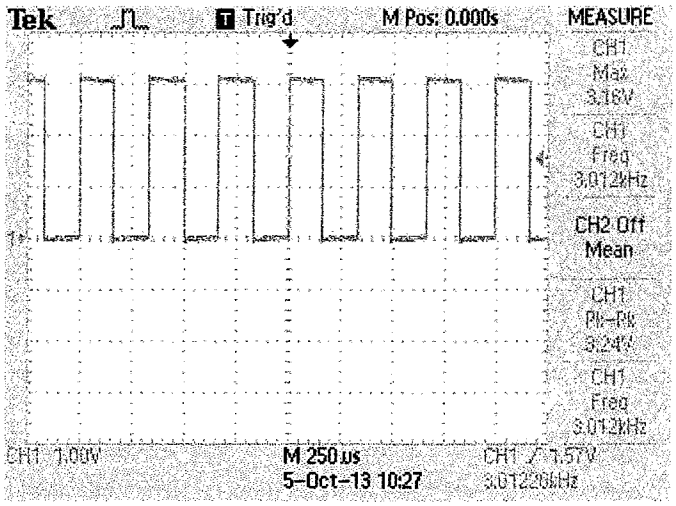


Figura 28. Tono de 4kHz en la salida del microcontrolador

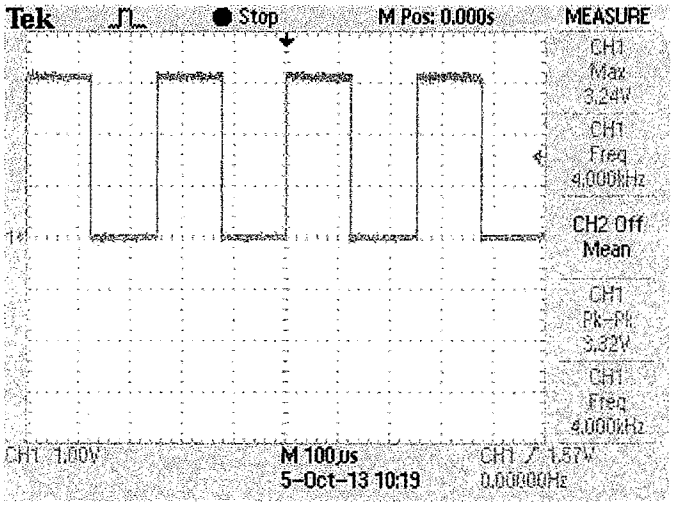


Figura 29. Tono de 6kHz en la salida del microcontrolador

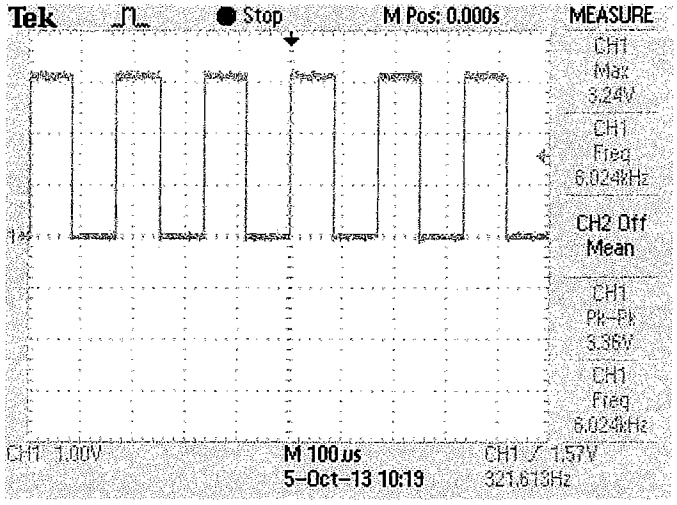
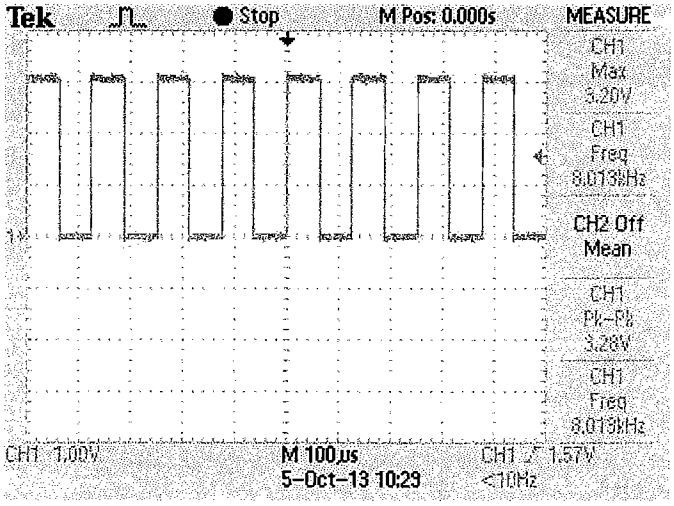


Figura 30. Tono de 8kHz en la salida del microcontrolador



Como se puede observar en las imágenes anteriores, la salida del microcontrolador genera señales con un voltaje máximo de 3.16V al ser alimentado con 2 baterías AA. Este voltaje toma importancia en el cálculo de la generación de potencia, ya que a partir de este punto se realiza la división de voltaje con el potenciómetro.

B. GENERACIÓN DE LA POTENCIA

Se hizo uso de la ecuación de SPL para determinar los valores teóricos de voltaje de salida necesarios para generar los decibeles esperados. Estos datos fueron utilizados para determinar el número de ciclos de incremento necesarios en el potenciómetro digital AD5220 para generar la potencia de salida hacia los auriculares. El Cuadro 9 muestra la amplitud del voltaje de salida del circuito para una frecuencia de 1kHz, medido con un osciloscopio.

Cuadro 8. Potencia SPL para voltajes experimentales

Posición de potenciómetro	Voltaje teórico (mV)	Voltaje experimental (mV)	Porcentaje de error
1	0.313	0.1000	68.051
3	0.938	0.3200	65.88
10	3.125	2.8500	12.30
32	10.000	10.800	8.000
101	31.560	32.300	2.345

Los valores fueron corregidos a través de las señales de control que envía el microcontrolador hacia el potenciómetro digital. Se corrigieron los valores de acuerdo a los niveles de voltaje necesarios calculados. El cuadro 9 muestra las correcciones realizadas

Cuadro 9. Potencia SPL para voltajes experimentales corregidos

Posición de potenciómetro	Voltaje final	Lp (dB)	Lp (dB) ideal	Porcentaje de error
3	0.0003200	20.1030	20.0000	0.5150
5	0.0009700	29.7354	30.0000	0.8819
11	0.0031500	39.9662	40.0000	0.0845
32	0.0108000	50.6685	50.0000	1.3370
101	0.0323000	60.1841	60.0000	0.3068

El control del circuito integrado AD5220 requiere de la recepción de pulsos para incrementar o decrementar el valor de la resistencia. Se implementó un ancho de pulso de 10ms para este propósito. Al implementar anchos de pulsos más pequeños, el sistema no variaba el valor de posición.

La potencia en escala de decibeles fue comparada con un tamizador Phonak. El audiómetro genera las frecuencias de 500Hz, 1kHz, 2kHz y 4kHz a una potencia de 40dB. Se utilizaron 2 aplicaciones de teléfono celular para medir la potencia del tamizador en sus 4 frecuencias. Las aplicaciones fueron UE SPL y Sonómetro. Se observó que el valor medido fue 40dB según lo indicado en la hoja de datos del tamizador. Posteriormente se utilizó la aplicación UE SPL para medirla potencia de las frecuencias del circuito armado. Las potencias evaluadas fueron 40dB, 50dB y 60dB. No fue posible medir a potencias de 20dB y 30dB debido al ruido de ambiente durante las pruebas, el cual mantenía una base de 30dB.

Cuadro 10. SPL medido con aplicación UE SPL

Frecuencia (Hz)	Potencia medida		
	40dB	50dB	60dB
250	39	49.2	59.1
500	39	49.4	59.4
1000	40	50	60
2000	40	50	60
3000	40	50	60
4000	40	50	60
6000	40	50	60
8000	40	51	60.3

C. FUNCIONAMIENTO DEL SISTEMA CON DOS PACIENTES

Los archivos necesarios para ejecutar el software en cualquier computadora son el ejecutable creado a partir de Visual Studio y el archivo dll de ZedGraph para tener la funcionalidad de generación de gráficos. Los requerimientos mínimos de la máquina para implementar este software se encuentran en el Cuadro 11. Los requerimientos se basan en pruebas realizadas con distintas computadoras. Se utilizó el adaptador de XBee para comunicación con la computadora a través del puerto USB. El adaptador incluye el driver necesario para su funcionamiento, éste se instala automáticamente al conectarlo a la computadora.

Cuadro 11. Requerimientos mínimos de la computadora

Característica	Descripción	Comentarios
Sistema operativo	Windows: XP, Vista, 7, 8	
Memoria	60MB	
Espacio en disco	332kB	
Puertos	1 puerto USB	Adaptador XBee incluye drivers

Se realizaron pruebas del sistema para dos pacientes. La Figura 31 muestra el circuito final implementado en protoboard. Como se puede observar en la imagen, la computadora ha ejecutado el comando de agregar nuevo usuario. Se debe agregar cada cliente por separado, el procedimiento consiste en encender el primer XBee receptor y agregar los datos. Luego que haya concluido, se procede a encender el circuito del segundo XBee receptor para incluir los datos del siguiente usuario y de esta forma sucesivamente en caso de implementar un mayor número de pacientes.

Figura 31. Sistema completo con dos pacientes

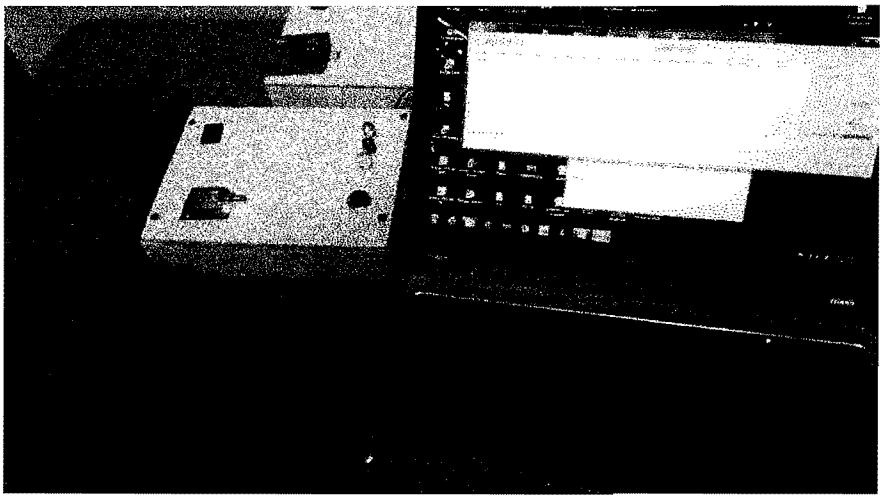
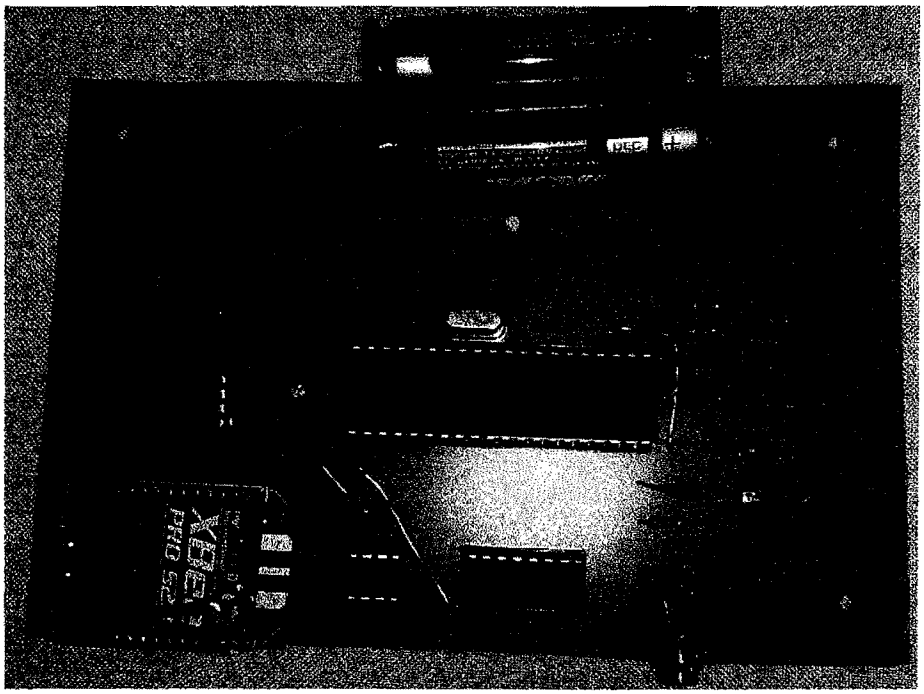
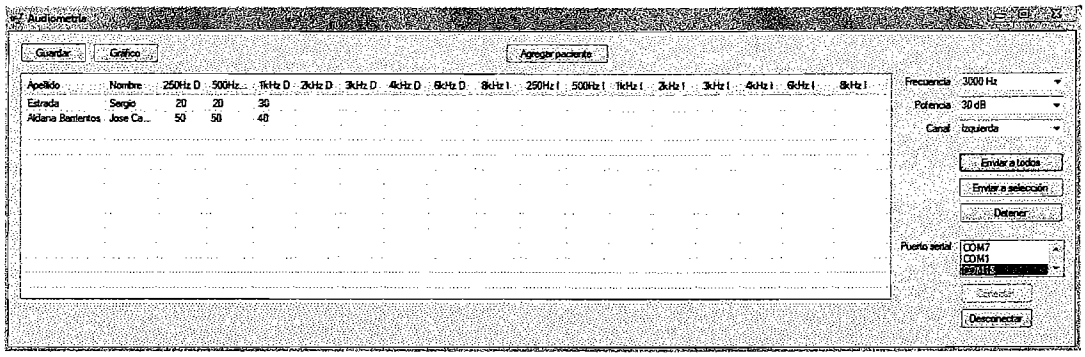


Figura 32. Placa impresa del circuito receptor



Al agregar un usuario se almacena la información en una lista interna y se extraen algunos datos para mostrarlos en la vista de la GUI. La Figura 32muestra el resultado de le interacción del sistema con 2 usuarios simultáneamente. Se ha enviado información hacia ambos usuarios y se recibió retroalimentación de ambos ante las evaluaciones.

Figura 33. Vista de evaluación de dos usuarios simultáneos



La generación del archivo csv con todos los datos de los pacientes permite observar los datos completos ingresados para cada usuario junto con las potencias para cada frecuencia especificada.

Al enviar la información a un único receptor, los demás pacientes no reciben los datos, por lo que su botón de retroalimentación se encuentra bloqueado. Esta funcionalidad forma parte de la programación defensiva implementada.

La generación de los gráficos permite observar los resultados para el paciente seleccionado. Se tiene un campo adicional para escribir el resultado del análisis realizado por el examinador. En la parte inferior el botón para imprimir abre el cuadro de diálogo con las impresoras encontradas en el sistema. Se generó un pdf con la información contenida. La imagen debe ser escalada al formato adecuado por el usuario para imprimir.

Figura 34. Generación de la vista de gráfica.

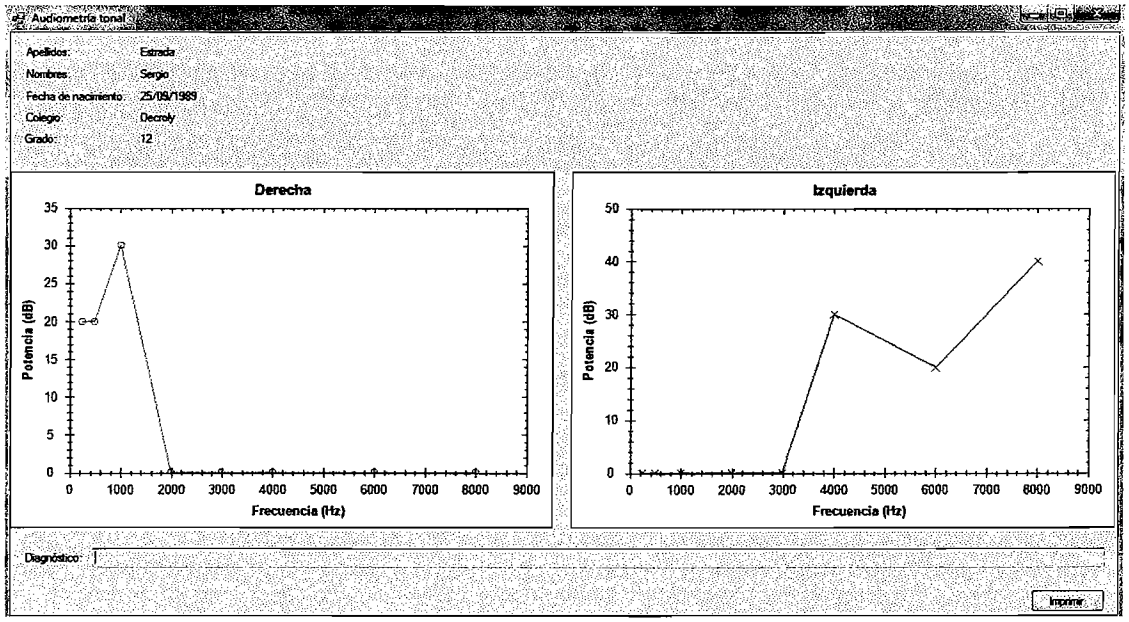
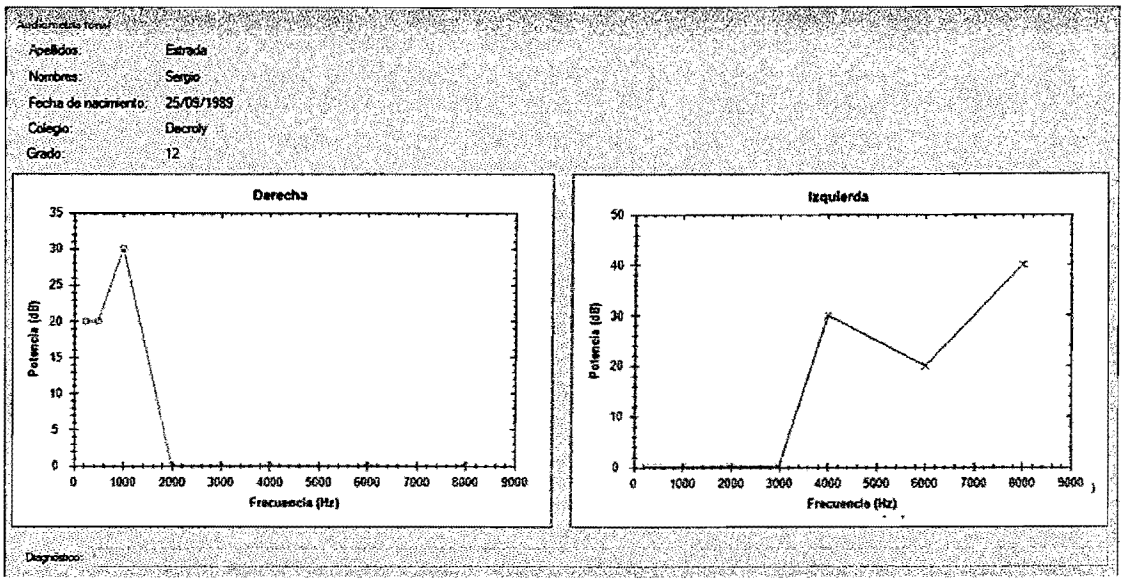


Figura 35. Impresión en PDF de la vista de gráfico



D. SISTEMA DE CALIBRACIÓN DEL CIRCUITO

Se realizaron pruebas del sistema de calibración utilizando el circuito en protoboard. Debe de tomarse en cuenta que al ingresar al modo de calibración se genera un voltaje DC, no existe frecuencia en la señal por lo tanto es recomendable desconectar los auriculares del equipo. Al mover el potenciómetro de precisión se obtiene un leve cambio en el voltaje de salida, el cual es adecuado para poder obtener con mayor precisión el voltaje esperado.

Figura 36. Medición de voltaje del circuito descalibrado

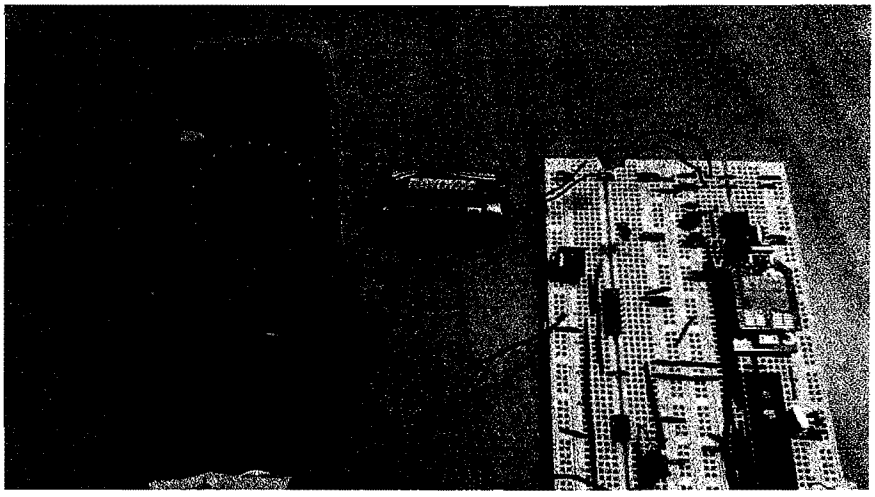
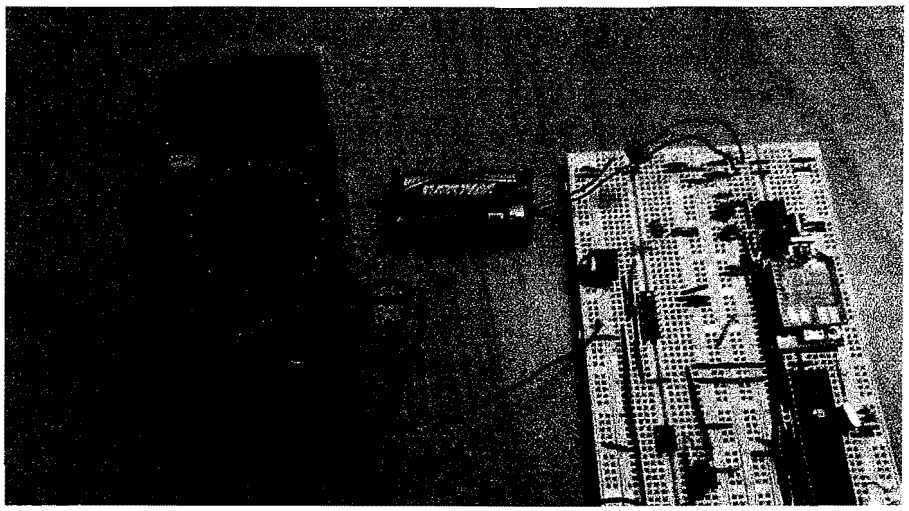


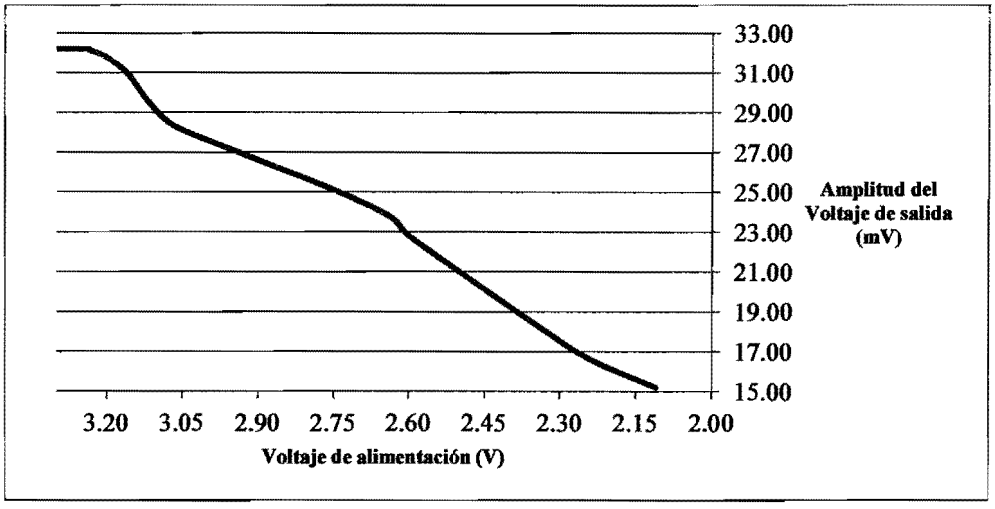
Figura 37. Medición del voltaje luego del ajuste del potenciómetro



E. EFECTO DE LA DESCARGA DE LAS BATERÍAS

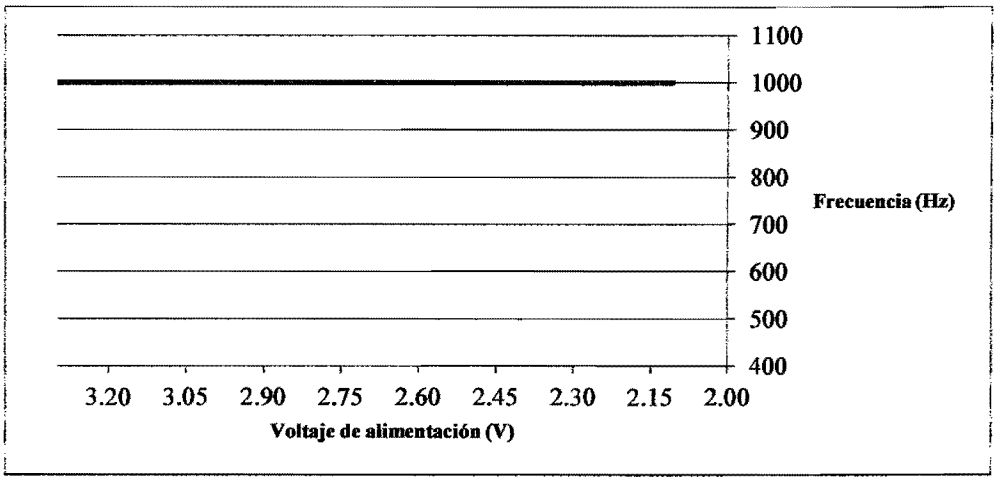
El circuito se encuentra alimentado con 2 baterías AA. En condiciones ideales las baterías y la calibración inicial proveen las señales adecuadas para la generación de las señales de audio. Sin embargo el circuito consume una cantidad de corriente durante la operación, por lo que las baterías se descargan con el tiempo de uso. Este desgaste puede ocasionar diferencias en las características de las señales, por lo que se pueden obtener resultados falsos en las pruebas aplicadas a los pacientes. Se utilizó un osciloscopio para realizar mediciones de voltaje y frecuencia. Las pruebas fueron ejecutadas utilizando una señal de 1kHz con la amplitud de voltaje de 32.3mV, siendo esto el equivalente a una señal de 60dB. Se utilizó el circuito de forma continua y se descargaron las baterías. Las siguientes figuras muestran los resultados obtenidos en voltaje y frecuencia.

Figura 38. Voltaje de salida



Como se puede observar en la Figura 38, la amplitud del voltaje de salida se ve afectada en gran medida conforme el voltaje de alimentación disminuye. Se mantiene un voltaje de 32.2mV en el intervalo de alimentación de 3.28V a 3.24V.

Figura 39. Frecuencia de la señal



En la Figura 39 se observa la frecuencia medida durante las diferentes etapas de descarga de las baterías. Como se puede apreciar en la imagen, la frecuencia permanece en 1kHz durante todas las mediciones. Esto indica que la frecuencia del circuito depende de la capacidad del microcontrolador de generar las señales y generar la oscilación del cristal.

Ambas mediciones tienen su límite en un voltaje de alimentación de 2.11V. Se observó durante las pruebas que a este voltaje el microcontrolador y el módulo XBee lograban entrar en sincronización inicial pero el XBee no era capaz de recibir de forma correcta los comandos del módulo maestro. Esta condición evita que el PIC genere las señales de audio.

F. COMPARACIÓN ENTRE LA SEÑAL CUADRADA DEL CIRCUITO CONTRA UNA SEÑAL SINUSOIDAL

El circuito implementado genera una señal cuadrada con la frecuencia especificada. El espectro de una señal cuadrada está compuesta por armónicos impares de la frecuencia inicial. Esto implica que para una señal de 1kHz, se tendrían también las señales de 3kHz, 5kHz, 7kHz, etc. Estos armónicos poseen una potencia más baja que la frecuencia central, sin embargo podrían causar efectos no esperados en el análisis de pacientes. Se realizaron pruebas en campo con diferentes personas comparando el sonido escuchado por el dispositivo de este proyecto contra el sonido escuchado por un audiómetro en CEDAF. La potencia de las pruebas fue en dependencia del examen previo realizado a los pacientes, por lo tanto no se incluye como parte del análisis. El desarrollador de este proyecto realizó la misma prueba. Los resultados obtenidos se resumen en la siguiente tabla.

Cuadro 12. Resultados experimentales de la diferencia de la señal del dispositivo contra un audiómetro profesional

Frecuencia (Hz)	Percepción
250	Más agudo
500	Más agudo
1000	Iguales
2000	Iguales
3000	Iguales
4000	Iguales
6000	Iguales
8000	Iguales

Los resultados muestran que se percibe una misma frecuencia entre ambos dispositivos para el rango comprendido de 1kHz a 8kHz.. Las frecuencias que aparentan un mayor efecto debido a la sobre posición de los armónicos son las de 250Hz y 500Hz.

VII. DISCUSIÓN DE RESULTADOS

El microcontrolador genera tonos con frecuencias cercanas a los valores teóricos esperados. Se puede observar del cuadro que el porcentaje de error mayor encontrado para alguna frecuencia fue de 0.4% luego de las correcciones establecidas. Posteriormente se determinó la frecuencia obtenida en la salida del demultiplexor, siendo el último punto antes de ingresar a los audífonos. Las frecuencias no tuvieron variaciones significativas como se observa en el cuadro 8. Las frecuencias de 250Hz, 500Hz, 1000Hz y 2000Hz se mantuvieron con un error de 0%. Para las frecuencias más altas, el tiempo de respuesta de los circuitos afectó en la forma de la señal. El valor más alejado de su parámetro teórico fue para 3000Hz, en donde al final del circuito se obtiene una frecuencia de 2941Hz. La frecuencia generada por el módulo PWM del microcontrolador es muy exacta.

La generación de los pulsos PWM dentro del microcontrolador hace uso de algunos registros de timers internos y del puerto CCP con una resolución de 10 bits. El módulo PWM de MikroC al igual que la programación directa en assembler utiliza el TMR2 para generar la frecuencia de la señal. El timer tiene dependencia de la frecuencia de reloj utilizada para el microcontrolador. El cálculo de la frecuencia es realizado por el compilador.

La potencia generada por el circuito se mantuvo con valores cercanos a los teóricos. El cuadro 11 muestra los resultados obtenidos de la implementación de la potencia con la posición calculada a partir del diseño experimental inicial. Como se puede observar, para los voltajes más altos que equivalen a las potencias de 50dB y 60dB se obtuvieron resultados aceptables. El error entre la medición y el valor esperado es muy reducido. Los voltajes más pequeños no fue posible obtener el valor deseado a partir de la configuración inicial. El porcentaje de error supera el 40% en ambas mediciones. La corrección de los datos fue realizada a partir de mediciones experimentales, en las cuales se buscó un valor intermedio entre los datos obtenidos que fuese capaz de retornar el voltaje necesario. Para la corrección del voltaje de 20dB se implementó la posición de potenciómetro especificada inicialmente para 30dB debido a que se encontraba en un voltaje de 0.32mV y el necesario es de 0.313mV. Se hizo variar el potenciómetro en un rango entre 4 y 8 para determinar la mejor posición posible para la generación de 30dB, el valor acertado fue 7. El resultado final para los 40dB fue con una posición del potenciómetro en 13. La variación entre el valor teórico y el experimental se debe a los circuitos implementados. La señal original primero se reduce a un voltaje de 0.4 y a partir de este valor se aplica la división de voltaje en el potenciómetro. Se logró observar que a la salida del circuito existe un ruido base en todas las mediciones de potencia al utilizar el osciloscopio. El ruido tiene una amplitud de 200uV, el cual no representa problemas para las potencias superiores a 20dB.

Al finalizar con las pruebas realizadas con las mediciones de voltaje se realizaron comparaciones entre un tamizador auditivo y el proyecto desarrollado para frecuencias de 500Hz, 1kHz, 2kHz y 4kHz. Las comparaciones fueron por medio de una aplicación para dispositivo móvil, el cual evalúa potencia SPL. El tamizador se encuentra calibrado para una potencia de 40dB, por lo que se realizó un barrido de las frecuencias especificadas a 40dB con el dispositivo desarrollado. El teléfono mostró una potencia de 40dB para las mediciones con el tamizador. Las mediciones con el aparato armado también mostraron niveles de potencia entre 39dB y 40dB. Estas mediciones sirvieron como una prueba adicional a nivel experimental para determinar si las potencias se encontraban en sus valores correctos. Posteriormente se implementó esta aplicación para obtener resultados de la potencias de 40Db, 50dB y 60dB para todas las frecuencias del circuito. Los resultados se observan en el cuadro

10. Como se observa, las mediciones de potencia con este software mostraron que los niveles de voltaje implementados generan la potencia esperada para todo el rango de frecuencias. Para las frecuencias de 250Hz y 500Hz se obtuvo una medición menor al valor esperado, pero la variación máxima fue de 1dB. No fue posible implementar este método para las mediciones de las potencias de 20dB y 30dB. Esto se debe a que la aplicación tuvo como base entre 35dB a 30dB en las mediciones, por lo tanto no fue posible observar el cambio generado por el circuito armado.

El tiempo necesario para el ancho de pulso generado hacia el integrado AD5220 es de 10ms. Este tiempo agrega un delay a la generación de la señal de audiometría final que percibe el paciente desde que se envía la señal. La primera señal consiste de 130 pulsos para resetear la posición a 0 y luego se envían los pulsos de incremento. Para la potencia más grande se utilizan 101 pulsos. Realizando un cálculo para determinar el tiempo de retardo para la potencia de 60dB se observa que el número de pulsos totales es de 231. El total de tiempo de un pulso es 20ms. Multiplicando estos valores se observa que el tiempo total es de 4.62s para la generación de la señal, desde que el microcontrolador envía el primer pulso hacia el integrado.

El funcionamiento del sistema completo para dos personas muestra que el sistema fue capaz de reconocer las retroalimentaciones de cada usuario a su nivel respectivo. Cada paciente fue asignado a un nivel de lista diferente, separados por 23 datos de por medi. Se implementaron pruebas con ambos usuarios presionando de manera simultánea el botón y no hubo conflicto de datos. Como se observa en la figura las gráficas generadas pertenecen al usuario seleccionado y luego de realizar la impresión se eliminan los botones de imprimir y de control de la ventana. Cabe mencionar que el usuario debe de realizar el correcto escalamiento de la imagen previo a su impresión. Los gráficos utilizados muestran señalización con símbolos de X y O. Esta simbología pertenece a la utilizada por los exámenes de audiometrías a nivel internacional, en donde la X representa al oído izquierdo y el O al oído derecho. Se determinó un nivel de cero para mediciones que no presentan datos, este factor debe de considerarse al hacer la lectura del diagrama debido a que no implica una audición del paciente para 0dB.

La generación de un documento en formato csv fue satisfactoria. En la programación se insertó un carácter de coma para separar los elementos de un mismo usuario. Al finalizar con el conteo de la longitud de la fila, se implementó el comando de nueva línea denominado por: /n. El siguiente usuario es entonces asignado a la siguiente fila dentro del archivo.

Las pruebas efectuadas para determinar los efectos que causa la descarga de la batería sobre el circuito, muestran que las baterías utilizadas se encuentran limitadas por un margen de 6V. El voltaje mínimo que puede generar el nivel de salida esperado es de 3.24V. El voltaje de salida a partir de ese punto comienza a descender. La frecuencia de la señal permanece constante, no se ve afectada por el decremento en el voltaje de alimentación. La última medición posible fue con una diferencia de voltaje de alimentación de 2.11V. Por debajo de este valor se observó que el circuito es capaz de lograr la sincronización inicial entre PIC y XBee. El inconveniente se presentó en el lado del XBee, el cual no logra obtener la señal proveniente desde el módulo maestro de forma correcta. Esto evita que el microcontrolador obtenga las señales de control para generar con el circuito la señal de salida. Este resultado hace notar la importancia de implementar un cambio en el diseño inicial del circuito y utilizar una fuente de alimentación de mayor voltaje en conjunto con un regulador de voltaje para alimentar al circuito.

Se efectuaron pruebas experimentales para comparar el sonido percibido por la señal del circuito implementado contra el sonido generado por un audiómetro profesional. Los resultados muestran que

las frecuencias de 1kHz a 8kHz, fueron percibidas como el mismo tipo de tono. Esto permite observar que los armónicos que conforman las señales cuadradas no tienen un efecto mayor en el sonido. Las frecuencias de 250Hz y 500Hz mostraron un cambio en la frecuencia percibida por el oído. Los pacientes, y el desarrollador, indicaron que se escuchaba un tono más agudo en el dispositivo de este proyecto. La presencia de estos armónicos en las frecuencias bajas puede causar un efecto no adecuado en las mediciones con pacientes. Una implicación podría darse en el caso de personas que padezcan de pérdida auditiva con frecuencias bajas, en donde el paciente podría indicar que ha escuchado el tono cuando en realidad únicamente está escuchando uno de los armónicos de la frecuencia central.

El audiómetro realizado en este trabajo no cumple con todos los requerimientos necesarios para ser un audiómetro de tipo IV. El sistema utiliza audífonos, por lo tanto utiliza conducción aérea. Se recibe retroalimentación del paciente a través de un botón y se lleva un control de los resultados. Sin embargo las frecuencias generadas no son tonos puros. Como se observó en la comparación experimental entre señales sinusoidales y las señales cuadradas del proyecto, la falta de tonos puros afectó a los pacientes evaluados únicamente para las frecuencias de 250Hz y 500Hz.

VIII. CONCLUSIONES

1. El microcontrolador generó las frecuencias especificadas para las audiometrías tonales con porcentajes de error menores al 0.4%.
2. Se presenta un ruido sobre las señales generadas de 200uV el cual representa una afectación considerable a la señal de 20dB, ya que ésta se genera a partir de una amplitud de 320uV.
3. La comunicación punto a multipunto de los módulos XBee evita que las señales de los receptores se mezcle entre los mismos.
4. Se implementaron medidas de programación defensiva en el programa del microcontrolador como el bloqueo del botón para evitar que el usuario presione antes de recibir datos.
5. El sistema de programación defensiva en la computadora funcionó correctamente para evitar el envío erróneo de datos, saltos incorrectos en el almacenamiento de datos y reconocimiento inicial del puerto serial habilitado.
6. El sistema con dos receptores funcionó de manera correcta, los datos fueron asignados de manera independiente a cada usuario dentro de la lista y el gráfico.
7. Se desarrolló un sistema que permite imprimir el resultado con gráficas y un campo de análisis.
8. El sistema permite la evaluación de más de dos usuarios. Los módulos XBee proveen un límite teórico de 65,534 dispositivos, por lo que aceptaría 65,533 receptores.
9. El circuito puede operar de forma correcta en un rango de alimentación de las baterías de 3.3V a 3.24V. Por debajo de este valor el voltaje de salida disminuye.
10. El circuito desarrollado no cumple con todos los requerimientos para ser un audiómetro de tipo IV debido a que no utiliza tonos puros.
11. El circuito no genera tonos puros y esto causa una percepción del sonido errónea para las frecuencias de 250Hz y 500Hz, las cuales se escuchan como una frecuencia superior.
12. La generación de señales cuadradas de 1kHz a 8kHz fue percibida igual que una señal sinusoidal por los pacientes evaluados.

IX. RECOMENDACIONES

1. Para agregar un rango mayor de potencias debe de cambiar la configuración del circuito externo para los valores de resistencia posibles.
2. Una de las limitantes del proyecto para la generación de potencias es el circuito externo, el cual afecta en mayor proporción la señal equivalente a 20dB. Por lo que se recomienda la implementación de un circuito que elimine el ruido.
3. Se recomienda implementar un sistema interno del microcontrolador que permita determinar si las potencias se encuentran calibradas dentro de un parámetro establecido.
4. Para agregar un número de pacientes grande, se debe considerar temas de coordinación entre todos los dispositivos ya que aunque las señales no lleguen a ser mezcladas por el maestro, puede existir pérdida de datos.
5. Se recomienda realizar mediciones exhaustivas con un sonómetro bajo condiciones controladas para determinar con mayor precisión la potencia generada.
6. Se recomienda implementar alimentación con una batería de 9V y utilizar un regulador de voltaje de 3.3V para alimentar al circuito.
7. Se recomienda implementar filtros y multiplexores en la salida de la señal generada para obtener una señal final con forma sinusoidal con el objetivo de reducir el efecto de los armónicos que conforman la señal cuadrada.

X. BIBLIOGRAFÍA

1. Angulo, J., & Angulo, I. (2005). *Microcontroladores PIC Diseño práctico de aplicaciones.* Chile: McGrawHill.

2. Callaway, E. (s.f.). *Low Power Consumption Features of the IEEE 802.15.4/ZigBee LR-WPAN Standard.* Recuperado el 28 de Septiembre de 2013, de http://www.cens.ucla.edu/sensys03/sensys03-callaway.pdf

3. Denia, A. L. (2009). *Detección, diagnóstico y tratamiento precoz de la sordera en la infancia.* Recuperado el 13 de Marzo de 2014, de http://sgfm.elcorteingles.es/SGFM/FRA/recursos/doc/Libros/963885904_352010125954.pdf

4. Dick, R. (s.f.). *Wireless Sensor Networks and RFIDs.* Recuperado el 28 de Septiembre de 2013, de http://ziyang.eecs.umich.edu/~dickrp/sensor-nets/lectures/print-lecture4.pdf

5. Everest, F. (2001). *The Master Handbook of Acoustics.* Estados Unidos: McGraw-Hill.

6. FTDI Chip. (s.f.). *Technical Note TN_111 What is UART?* Recuperado el 2 de Octubre de 2013, de http://www.ftdichip.com/Support/Documents/TechnicalNotes/TN_111%20What%20is%20UART.pdf

7. Gerber, S. M. (1987). *Early Diagnosis of Hearing Loss.* Nueva York: Grunc and Stratton.

8. Goycoolea, M. E. (Enero de 2003). *Métodos de Evaluación Auditiva.* Obtenido de http://www.clinicalascondes.com/area_academica/Revista_Medica_Enero_2003/articulo_003.htm

9. Hoff, P. (s.f.). *Microcontroller Hands-On Tutorials.* Recuperado el 4 de Octubre de 2013, de http://www.elec.canterbury.ac.nz/PublicArea/Staff/hof/index.html

10. *IEEE 802.15 Working Group for WPAN.* (s.f.). Recuperado el 28 de Septiembre de 2013, de http://www.ieee802.org/15/

11. Kimball, S. M. (9 de Junio de 2013). *Speech Audiometry.* Recuperado el 20 de Mar de 2014, de http://emedicine.medscape.com/article/1822315-overview#a01

12. Kutz, J. M. (10 de Junio de 2013). *Audiology Pure-Tone Testing.* Recuperado el 20 de Mar de 2014, de http://emedicine.medscape.com/article/1822962-overview#a01

13. Löwe, A. (1981). *Audiometría en el niño: Aplicaciones pedagógicas.* Buenos Aires: Panamericana.

14. Mansilla, C. (2013). *Audiometría.* Recuperado el 14 de Marzo de 2014, de https://www.academia.edu/4398819/2._Audiometria_2013

15. Morse, P. (1948). *Vibration and Sound.* Estados Unidos: McGraw Hill.

16. Oyarce, A. (2010). *Guía del Usuario XBee Series 1.* Santiago, Chile: Olimex.

17. Poch, J. P. (2005). *Otorrinolaringología y patología cervicofacial.* Buenos Aires, Madrid: Médica Panamericana.

18. Quirós, J. D. (1981). *Introducción a la audiometría.* Barcelona, España: Gráficas Instar, S.A.

19. Roeser, R. V.-D. (2007). *Audiology Diagnosis.* Nueva York: Thieme Medical Publishers, Inc.

20. Rouviére, H. D. (1991). *Anatomía humana, descriptiva, topográfica y funcional. Tomo 1. Cabeza y cuello.* Barcelona, España: Masson.

21. University of Notre Dame. (s.f.). *Sound Power and Pressure Measurements*. Obtenido de http://www3.nd.edu/~atassi/Teaching/AME%2060633/Notes/Sound%20Power%20and%20Pressure%20Measurements.pdf

22. Verle, M. (s.f.). *PIC Microcontrollers - Programming in C*. Recuperado el 28 de Septiembre de 2013, de http://www.mikroe.com/products/view/285/book-pic-microcontrollers-programming-in-c/

23. *XBee - XBee PRO RF Modules*. (s.f.). Recuperado el 28 de Septiembre de 2013, de http://ftp1.digi.com/support/documentation/90000982_M.pdf

XI. ANEXOS

A. CÓDIGO DEL PROGRAMA DEL PIC

```
//*******************************************
// Universidad del Valle de Guatemala
// Ingeniería Electrónica
// Trabajo de Graduación// Sergio Estrada 08032
// Código de PIC18F45K22 para módulos esclavos
//*******************************************

//-------------------------------------------
// Variables
//-------------------------------------------
int hay_f, nuevof;
int but_escucha;
int d_i;
int d_i_out;
int fout;
int pout;
int p_pot;
int f_vout;
int p_vout;
int valor;
int tempRX;
int tempRB;
int bit_control;
char ID[4];
int ID2;
int x;
int existe;
int butRespuestaON;
int Calibrar;

//-------------------------------------------
//Interrupciones
//-------------------------------------------
void interrupt(){
     if (PIR1.RC1IF == 1){
        tempRX = RCREG;
        if (tempRX == '{'){
            asm NOP;
            bit_control = 10;
        }
        else if(tempRX == '$'){
            bit_control = 0;
            PWM1_Stop();
            hay_f = 0;
            butRespuestaON = 0;
        }
        else{
            if (bit_control == 0){
                bit_control=1;
                f_vout = tempRX;
```

```
        }
        else if ((bit_control == 1)){
            bit_control=2;
            p_vout = tempRX;
        }
        else if (bit_control == 2){
            d_i_out = tempRX;
            hay_f = 1;
            nuevof = 1;
            bit_control = 0;
            butRespuestaON = 1;
        }
        else if (bit_control == 10){
            bit_control = 0;
        }
        else{
            bit_control = 0;
        }
    }
}

 if (INTCON.RBIF == 1){
    asm MOVF PORTB,1;
    INTCON.RBIF = 0;

    if(PORTB.F7 == 1){
        if (butRespuestaON == 1){
            tempRB++;
            if (tempRB==1){
                hay_f = 2;
                fout = fout;
                pout = pout;
                d_i = d_i;
            }
            else if(tempRB == 2){
                tempRB=0;
                butRespuestaON = 0;
            }
        }
        else{
            asm NOP;
        }
    }
    if(PORTB.F6 == 1){
        tempRB++;
        if (tempRB == 1){
            if(Calibrar == 0){
                PIE1.RCIE = 0;
                hay_f = 3;
                p_vout = 0x34;
                d_i_out = 0x30;
                nuevof = 1;
                PORTA.F2 = 1;
                LATA.F2 = 1;
                Calibrar = 1;
            }
            else if(Calibrar == 1){
```

```
                    PIE1.RCIE = 1;
                    PORTA.F2=0;
                    LATA.F2=0;
                    Calibrar = 0;
                    hay_f = 0;
                }
                else{
                    asm NOP;
                }
            }
            else{
                tempRB = 0;
                asm NOP;
            }
        }
        else{
            asm NOP;
        }
    }
}

//-----------------------------------------------
// Método de emisión de tono
//-----------------------------------------------
void frecuencia(){
    PWM1_Start();
    Delay_ms(250);
    PWM1_Stop();
    Delay_ms(250);
}

//-----------------------------------------------
// Método de ajuste de valor de frecuencia
//-----------------------------------------------
void valor_frecuencia(){
    if (f_vout == 0x30){
        PWM1_Init(250);
        fout = f_vout;
    }
    else if (f_vout == 0x31) {
        PWM1_Init(500);
        fout = f_vout;
    }
    else if (f_vout == 0x32){
        PWM1_Init(1000);
        fout = f_vout;
    }
    else if (f_vout == 0x33){
        PWM1_Init(2000);
        fout = f_vout;
    }
        else if (f_vout == 0x34){
        PWM1_Init(3001);
        fout = f_vout;
    }
    else if (f_vout == 0x35){
        PWM1_Init(4000);
```

```
        fout = f_vout;
    }
    else if (f_vout == 0x36){
        PWM1_Init(6002);
        fout = f_vout;
    }
    else if (f_vout == 0x37) {
        PWM1_Init(7990);
        fout = f_vout;
    }
    else{
        PWM1_Stop();
        nuevof = 0;
        hay_f=0;
    }
}
//-----------------------------------------------
// Método de ajuste de generación de potencia
//-----------------------------------------------
void ajustaPotDown(int p_pot){
     int valor_r;
     LATD.F5 = 0;
     LATD.F7 = 0;
     valor_r = 0;
     Delay_ms(10);

    while(valor_r < 130){
        LATD.F6 = 1;
        Delay_ms(10);
        LATD.F6 = 0;
        Delay_ms(10);
        valor_r++;
    }

    LATD.F7 = 1;
    Delay_ms(50);
    LATD.F5 = 1;
    LATD.F7 = 0;
    valor_r = 0;
    Delay_ms(10);

    while(valor_r < p_pot){
        LATD.F6 = 1;
        Delay_ms(10);
        LATD.F6 = 0;
        Delay_ms(10);
        valor_r++;
    }

    LATD.F7 = 1;
    LATD.F5 = 0;
    Delay_ms(10);
}

//-----------------------------------------------
// Método de selección de valor de potencia
//-----------------------------------------------
```

```
void valor_potencia(){
    if (p_vout == 0x30){
        pout = p_vout;
        p_pot =3;
    }
    else if (p_vout == 0x31){
        pout = p_vout;
        p_pot = 10;
    }
    else if (p_vout == 0x32){
        pout = p_vout;
        p_pot = 31;
    }
    else if (p_vout == 0x33){
        pout = p_vout;
        p_pot = 98;
    }
    else if (p_vout == 0x34){
        pout = p_vout;
        p_pot = 128;
    }
    else{
        p_pot = 0;
    }

    ajustaPotDown(p_pot);
}

//-------------------------------------------
// Selección de lado derecho/izquierdo
//-------------------------------------------
void valor_lado(){
    if (d_i_out == 0x30){
        d_i = 0x30;
        LATD.F2 = 0;
        PORTD.F2 = 0;
    }
    else if (d_i_out == 0x31){
        d_i = 0x38;
        LATD.F2 = 1;
        PORTD.F2 = 1;
    }
    else{
        LATD.F2 = 0x00;
    }
}

//-------------------------------------------
// Método para obtener el ID del xBee
//-------------------------------------------
void get_ID(){
    UART1_Write_Text("+++");
    while(tempRX != 'O'){
        tempRX = UART1_Read();
    }
    while(tempRX != 'K'){
        tempRX = UART1_Read();
```

```
    }
    while(tempRX != 0x0D){
        tempRX = UART1_Read();
    }

    UART1_Write_Text("ATMY");
    UART1_Write(0x0D);
    while(UART1_Data_Ready()==0){
    }
    ID[0]=UART1_Read();
    while(UART1_Data_Ready()==0){

    }
    ID[1]=UART1_Read();
    while(UART1_Data_Ready()==0){

    }
    ID[2]=UART1_Read();
    while(UART1_Data_Ready()==0){

    }
    ID[3]=UART1_Read();

    while(tempRX != 0x0D){
        tempRX = UART1_Read();
    }

    UART1_Write_Text("ATCN");
    UART1_Write(0x0D);
    while(tempRX != 'O'){
        tempRX = UART1_Read();
    }
    while(tempRX != 'K'){
        tempRX = UART1_Read();
    }
    while(tempRX != 0x0D){
        tempRX = UART1_Read();
    }
    tempRX = 0;
    LATA.F0 = 1;
}
//-----------------------------------------------
// Método handshake
//-----------------------------------------------
void handshake(){
    LATA.F1 = 1;
    x=0;
    existe = 0;
    while (x != '{'){
        x = UART1_Read();
    }
    UART1_Write_Text(ID);
    Delay_ms(1);
    UART1_Write('*');
    while (UART1_Data_Ready()==0){
    }
    ID2 = UART1_Read();
```

```
    LATA.F1 = 0;
    Delay_ms(200);
    LATA.F1 = 1;
    Delay_ms(200);
    PORTA.F1 = 0;
    Delay_ms(200);
    PORTA.F1 = 1;
    Delay_ms(200);
    PORTA.F1 = 0;
    existe = 1;
}
//-----------------------------------------------
// Método principal
//-----------------------------------------------
void main() {
    ANSELA = 0;
    ANSELB = 0;
    ANSELC = 0;
    ANSELD = 0;
    LATA = 0;
    LATB = 0;
    LATC = 0;
    LATD = 0b10000000;
    TRISB = 0b11000000;
    TRISA = 0;
    PORTA = 0;
    PORTB = 0;
    TRISC = 0;
    PORTC = 0;
    TRISD = 0;
    PORTD = 0b10000000;
    C1ON_bit=0;
    C2ON_bit=0;
    bit_control=0;
    hay_f=0;
    nuevof=0;
    f_vout=0;
    fout=0;
    d_i=0;
    tempRB=0;
    existe = 0;
    butRespuestaON = 0;
    Calibrar = 0;
    p_pot = 1;

    UART1_Init(9600);
    Delay_ms(1000);

    ajustaPotDown(p_pot);
    Delay_ms(10);

    INTCON.GIE = 1;
    INTCON.RBIE = 1;
    IOCB.F7=1;
    IOCB.F6=1;
    LATC.F2 = 1;
    PORTC.F2 = 1;
```

```
    get_ID();
    handshake();
    LATC.F2=1;
    PORTC.F2=1;

    INTCON = 0b11001000;
    PIE1 = 0b00100000;
    IOCB.F7 = 1;
    IOCB.F6 = 1;

    while(1){
        if(hay_f==0){
            PWM1_Stop();
        }
        else if(hay_f==1){
            if(nuevof==1){
                nuevof = 0;
                valor_frecuencia();
                valor_potencia();
                valor_lado();
            }
            PWM1_Set_Duty(127);
            frecuencia();
        }
        else if(hay_f==2){
            PWM1_Stop();
            hay_f = 0;
            Delay_ms(1);
            UART1_Write(ID2);
            UART1_Write(fout);
            UART1_Write(pout);
            UART1_Write(d_i);
            UART1_Write('*');
        }
        else if (hay_f = 3){
            if(nuevof=1){
                nuevof = 0;
                valor_potencia();
                valor_lado();
                PWM1_Set_Duty(255);
                PWM1_Start();
            }
            asm NOP;
        }
        else{
            hay_f = 0;
            PWM1_Stop();
        }
    }
}
```

B. CÓDIGO DEL PROGRAMA DE COMPUTADORA

```
//*******************************************
// Universidad del Valle de Guatemala
// Ingeniería Electrónica
// Trabajo de Graduación
// Sergio Estrada 08032
// Código de aplicación en computadora
//*******************************************
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;
using System.IO;
using Microsoft.VisualBasic;
using ZedGraph;

namespace Audiometría
{
    public partial class Form1 : Form
    {
        bool conectado = false;
        bool existePuerto = false;
        bool salir = false;
        bool es_handshake = false;
        bool hay_seleccion = false;
        string f_out = "0";
        string p_out = "0";
        string lado = "0";
        string datos = "";
        string nuevo_ID = "";
        int frecuencia;
        int potencia;
        int canal;
        int[] ID_Cliente = new int[3];
        int yaPaso = 0;
        List<string> listado = new List<string>();
        int indice_id = 0;
        int id2=0;
        string eApellido, eNombre, eFecha, eColegio, eGrado, datos2,
showPot;
        int indice_temp, pot_temp, indice_temp_id;
        bool todos = true;
        bool conf_xbee = false;
        string DN_Xbee = "0";
        int answer = 0;
        int EnterKey = 0x0D;

        string[] header = new string[] {"Apellidos","Nombre","Fecha de
nacimiento","Colegio","Grado","250Hz D","500Hz D","1kHz D","2kHz D",
```

```
        "3kHz D","4kHz D","6kHz D","8kHz D","250Hz I","500Hz I","1kHz
I","2kHz I","3kHz I","4kHz I","6kHz I","8kHz I"};

        //****************************************
        // Inicialización de la aplicación
        //****************************************
        public Form1()
        {
            InitializeComponent();
            BTNuno.Hide();
            label5.Hide();
            label6.Hide();

            while (!existePuerto)
            {
                if (salir == true)
                {
                    Application.Exit();
                }
                else
                {
                    BuscaPuerto();
                }

            }
        }

        //****************************************
        // Buscar puerto serial disponible
        //****************************************
        public void BuscaPuerto()
        {
            for (int x = 0; x < (SerialPort.GetPortNames().Length);
x++)
            {

PuertoSerial.Items.Add(SerialPort.GetPortNames().ElementAt(x).ToString(
));
            }
            try
            {
                PuertoSerial.SetSelected(0, true);
                BTNdesconectado.Enabled = false;
                BTNtodos.Enabled = false;
                BTNuno.Enabled = false;
                button1.Enabled = false;
                button2.Enabled = false;
                button3.Enabled = false;
                button4.Enabled = false;
                comboBoxF.Enabled = false;
                comboBoxP.Enabled = false;
                comboBoxC.Enabled = false;
                existePuerto = true;
            }
            catch (Exception e)
            {
```

```
                string mensaje = "No se ha detectado un puerto COM
habilitado." + "\nPor favor conecte el cable";
                string caption = "Error";
                DialogResult respuesta = MessageBox.Show(mensaje,
caption, MessageBoxButtons.OKCancel);

                if (respuesta == DialogResult.Cancel)
                {
                    salir = true;
                }
                else
                {
                    salir = false;
                }

            }
        }

        private void PuertoSerial_SelectedIndexChanged(object sender,
EventArgs e) { }

        //*****************************************
        // Conexión con puerto serial
        //*****************************************
        private void BTNconectado_Click(object sender, EventArgs e)
        {
            try
            {
                _serialPort.PortName =
PuertoSerial.SelectedItem.ToString();
                _serialPort.BaudRate = 9600;
                _serialPort.WriteBufferSize = 10240;
                _serialPort.ReadBufferSize = 10240;
                _serialPort.Open();
                _serialPort.DiscardInBuffer();
                BTNdesconectado.Enabled = true;
                BTNconectado.Enabled = false;
                BTNtodos.Enabled = true;
                BTNuno.Enabled = true;
                button1.Enabled = true;
                button2.Enabled = true;
                button3.Enabled = true;
                button4.Enabled = true;
                comboBoxF.Enabled = true;
                comboBoxP.Enabled = true;
                comboBoxC.Enabled = true;
                conectado = true;
            }
            catch (Exception e1)
            {
                String mensaje = "No existe puerto COM disponible";
                MessageBox.Show(mensaje);
            }
        }

        //*****************************************
        // Desconexión de puerto serial
```

```
        //*****************************************
        private void BTNdesconectado_Click_1(object sender, EventArgs
e)
        {
            conectado = false;
            BTNdesconectado.Enabled = false;
            BTNconectado.Enabled = true;
            BTNtodos.Enabled = false;
            BTNuno.Enabled = false;
            button1.Enabled = false;
            button2.Enabled = false;
            button3.Enabled = false;
            button4.Enabled = false;
            comboBoxF.Enabled = false;
            comboBoxP.Enabled = false;
            comboBoxC.Enabled = false;
            _serialPort.Close();
        }

        //*****************************************
        // Enviar mensaje a todos los receptores
        //*****************************************
        private void BTNtodos_Click(object sender, EventArgs e)
        {
            _serialPort.Write(f_out);
            _serialPort.Write(p_out);
            _serialPort.Write(lado);
            label5.Text = "estamos en todos";
        }

        //*****************************************
        // Evento de selección de frecuencia
        //*****************************************
        private void comboBoxF_SelectionChangeCommitted(object sender,
System.EventArgs e)
        {
            ComboBox senderComboBox = (ComboBox)sender;
            switch (senderComboBox.SelectedIndex)
            {
                case 0:
                    f_out = "0";
                    frecuencia = 250;
                    break;
                case 1:
                    f_out = "1";
                    frecuencia = 1000;
                    break;
                case 2:
                    f_out = "2";
                    frecuencia = 2000;
                    break;
                case 3:
                    f_out = "3";
                    frecuencia = 3000;
                    break;
                case 4:
                    f_out = "4";
```

```
                frecuencia = 4000;
                break;
            case 5:
                f_out = "5";
                frecuencia = 6000;
                break;
            case 6:
                f_out = "6";
                frecuencia = 7000;
                break;
            case 7:
                f_out = "7";
                frecuencia = 8000;
                break;
            default:
                f_out = "0";
                break;
        }
    }

    //*****************************************
    // Evento de selección de potencia
    //*****************************************
    private void comboBoxP_SelectionChangeCommitted(object sender,
System.EventArgs e)
    {
        ComboBox senderComboBox = (ComboBox)sender;
        switch (senderComboBox.SelectedIndex)
        {
            case 0:
                p_out = "0";
                potencia = 20;
                break;
            case 1:
                p_out = "1";
                potencia = 30;
                break;
            case 2:
                p_out = "2";
                potencia = 40;
                break;
            case 3:
                p_out = "3";
                potencia = 50;
                break;
            case 4:
                p_out = "4";
                potencia = 60;
                break;
            default:
                p_out = "0";
                break;
        }
    }

    //*****************************************
    // Evento de selección de lado
```

```
        //*****************************************
        private void comboBoxC_SelectionChangeCommitted(object sender,
EventArgs e)
        {
            ComboBox senderComboBox = (ComboBox)sender;
            switch (senderComboBox.SelectedIndex)
            {
                case 0:
                    lado = "0";
                    break;
                case 1:
                    lado = "1";
                    break;
                default:
                    lado = "0";
                    break;
            }
        }

        //*****************************************
        // Recepción de datos en puerto serial
        //*****************************************
        private void _serialPort_DataReceived(object sender,
SerialDataReceivedEventArgs e)
        {
            string x;
            string y;
            Control.CheckForIllegalCrossThreadCalls = false;
            if (es_handshake == false)
            {
                if (conf_xbee == true)
                {
                    y = _serialPort.ReadExisting();
                    label5.Text += "1" + y;
                    answer++;
                    label6.Text = answer.ToString();
                    if ((answer == 1) && (yaPaso == 0)) {
                        _serialPort.Write("ATDN" + DN_Xbee + "\r");
                        yaPaso = 1;
                    }
                    else if ((answer == 2) && (yaPaso == 1))
                    {
                        yaPaso = 0;
                        answer = 0;
                        y = "";
                        label5.Text = "fin config";
                        conf_xbee = false;
                        _serialPort.Write(f_out);
                        _serialPort.Write(p_out);
                        _serialPort.Write(lado);
                        label5.Text = "fin envio";
                    }
                }
                else
                {
                    try
                    {
```

```
                        x = _serialPort.ReadExisting();
                        datos += "" + x;

                        if (datos.EndsWith("*"))
                        {
                            label5.Text = "entra mensaje";
                            datos2 = datos.Substring(0, datos.Length -
1);
                            datos = "";
                            agregar_respuesta();
                            mostrar_respuesta();
                        }
                        else
                        {

                        }
                    }
                    catch (TimeoutException) { }
                }
            }
            else if (es_handshake == true)
            {
                x = _serialPort.ReadExisting();
                nuevo_ID += "" + x;
                if (nuevo_ID.EndsWith("*"))
                {
                    string nuevo_ID2 = nuevo_ID.Substring(0,
nuevo_ID.Length - 1);
                    listado.Insert(indice_id, nuevo_ID2);
                    listado.Insert(indice_id + 1, id2.ToString());
                    listado.Insert(indice_id + 2, eApellido);
                    listado.Insert(indice_id + 3, eNombre);
                    listado.Insert(indice_id + 4, eFecha);
                    listado.Insert(indice_id + 5, eColegio);
                    listado.Insert(indice_id + 6, eGrado);
                    for (int i = 7; i < 23; i++)
                    {
                        listado.Insert(indice_id + i, "");
                    }
                    _serialPort.Write(id2.ToString());
                    agregar_datos();
                    es_handshake = false;
                    indice_id = indice_id + 23;
                    id2 = id2 + 1;
                }
                else { }
            }
        }

        //*****************************************
        // Evento para agregar paciente
        //*****************************************
        private void button1_Click(object sender, EventArgs e)
        {
            Form2 form2 = new Form2();
            es_handshake = true;
```

```
            DialogResult dialogResult = form2.ShowDialog(this);
            try
            {
                if (dialogResult == DialogResult.OK)
                {
                    eApellido = form2.sApellido;
                    eNombre = form2.sNombre;
                    eFecha = form2.sFecha;
                    eColegio = form2.sColegio;
                    eGrado = form2.sGrado;

                    _serialPort.Write("{");

                    form2.Dispose();
                    form2.Close();
                }
                else
                {
                    form2.Dispose();
                }
            }
            catch
            {
                form2.Dispose();
                form2.Close();
            }
        }

        //*****************************************
        // Agregar datos a la lista de información
        //*****************************************
        private void agregar_datos()
        {
            ListViewItem lvi = new ListViewItem(listado[indice_id +
2]);
            lvi.SubItems.Add(listado[indice_id + 3]);

            for (int i = 0; i < 16; i++)
            {
                lvi.SubItems.Add("");
            }
            listView1.Items.Add(lvi);
        }

        //*****************************************
        // Agregar respuesta de paciente a listado
        //*****************************************
        private void agregar_respuesta()
        {
            dar_potencia();
            try
            {
            indice_temp = Int16.Parse(datos2.Substring(0, 1)) * 23 + 7
+ Int16.Parse(datos2.Substring(1, 1)) + Int16.Parse(datos2.Substring(3,
1));

                if (listado[indice_temp] == "")
```

```
        {
            listado.RemoveAt(indice_temp);
            listado.Insert(indice_temp, pot_temp.ToString());
            showPot = pot_temp.ToString();
        }

        else if (Int16.Parse(listado[indice_temp]) > pot_temp)
        {
            listado.RemoveAt(indice_temp);
            listado.Insert(indice_temp, pot_temp.ToString());
            showPot = pot_temp.ToString();
        }
        else
        {
            showPot = listado[indice_temp];
        }
    }
    catch {}
}

//*****************************************
// Conversión de código de potencia
//*****************************************
private void dar_potencia()
{
    switch (datos2.Substring(2,1))
    {
        case "0":
            pot_temp = 20;
            break;
        case "1":
            pot_temp = 30;
            break;
        case "2":
            pot_temp = 40;
            break;
        case "3":
            pot_temp = 50;
            break;
        case "4":
            pot_temp = 60;
            break;
        default:
            pot_temp = 100;
            break;
    }
}

//*****************************************
// Mostrar respuesta en pantalla
//*****************************************
void mostrar_respuesta()
{
    try
    {
        indice_temp_id = Int16.Parse(datos2.Substring(0, 1));
```

```
                indice_temp = 2 + Int16.Parse(datos2.Substring(1, 1)) +
Int16.Parse(datos2.Substring(3, 1));

listView1.Items[indice_temp_id].SubItems[indice_temp].Text = showPot;
            }
            catch { }
        }

        //*****************************************
        // Evento para guardar resultados
        //*****************************************
        private void button3_Click(object sender, EventArgs e)
        {
            saveFileDialog1.ShowDialog();
        }

        //*****************************************
        // Generación de formato de almacenamiento
        //*****************************************
        private void saveFileDialog1_FileOk(object sender,
CancelEventArgs e)
        {
            string filePath = saveFileDialog1.FileName;
            string delimitador = ",";
            int num = 0;
            int id2g = 0;
            int indice_idg = 0;
            StringBuilder st_build = new StringBuilder();

            try
            {
                for (int j = 0; j < 21; j++ )
                {
                    st_build.Append(header[j]);
                    st_build.Append(delimitador);
                }

                st_build.Append("\n");

                for (int i = 0; i <= 23; i++)
                {
                    if (id2g < id2)
                    {
                        if ((indice_idg == i + num) || (indice_idg + 1
== i + num))
                        {

                        }
                        else if (indice_idg + 23 == i + num)
                        {
                            i = 0;
                            indice_idg = indice_idg + 23;
                            id2g = id2g + 1;
                            num = num + 23;
                            st_build.Append("\n");
                        }
                        else
```

```
                    {
                        st_build.Append(listado[i + num]);
                        st_build.Append(delimitador);
                    }
                }
                else
                {
                    i = 24;
                }

            }

            File.WriteAllText(filePath, st_build.ToString());
            String mensaje = "Archivo guardado";
            MessageBox.Show(mensaje);
        }
        catch
        {
            string mensaje = "Error al guardar el archivo";
            MessageBox.Show(mensaje);
        }
    }

    //*****************************************
    // Enviar código para agregar dispositivo
    //*****************************************
    private void button2_Click(object sender, EventArgs e)
    {
        _serialPort.Write("$");
    }

    //*****************************************
    // Generación de gráfico
    //*****************************************
    private void button4_Click(object sender, EventArgs e)
    {
        List<string> listadoGrafica = new List<string>();
        List<string> listadoDatos = new List<string>();
        bool hay_seleccion = false;
        int id_send = 0;

        for (int i = 0; i < listView1.Items.Count; i++)
        {
            if (listView1.Items[i].Selected == true)
            {
                for (int j = 0; j < 18; j++)
                {

listadoGrafica.Add(listView1.Items[i].SubItems[j].Text);
                    id_send = i;
                }
                hay_seleccion = true;
            }
        }

        if (hay_seleccion == true)
        {
```

```
                for (int j = 2; j < 7; j++ )
                {
                    listadoDatos.Add(listado[id_send*23 + j]);
                }
                Form3 form3 = new Form3(listadoGrafica, listadoDatos);
                form3.Show();
                hay_seleccion = false;
            }

            else
            {
                string mensaje = "Seleccione un paciente";
                MessageBox.Show(mensaje);
            }
        }

        //****************************************
        // Enviar sonido a dispositivo seleccionado
        //****************************************
        private void BTNuno_Click_1(object sender, EventArgs e)
        {
            bool hay_seleccion = false;
            label5.Text = "";
            todos = false;
            conf_xbee = true;
            hay_seleccion = false;

            for (int i = 0; i < listView1.Items.Count; i++)
            {
                if (listView1.Items[i].Selected == true)
                {
                    DN_Xbee = i.ToString();
                    hay_seleccion = true;
                }
            }

            if ((conf_xbee == true) && (hay_seleccion == true))
            {
                _serialPort.Write("+++");
            }

            else if (hay_seleccion == false)
            {
                string mensaje = "Seleccione un paciente";
                MessageBox.Show(mensaje);
                conf_xbee = false;
            }
        }

    }
}

//****************************************
// Universidad del Valle de Guatemala
// Ingeniería Electrónica
// Trabajo de Graduación
// Sergio Estrada 08032
```

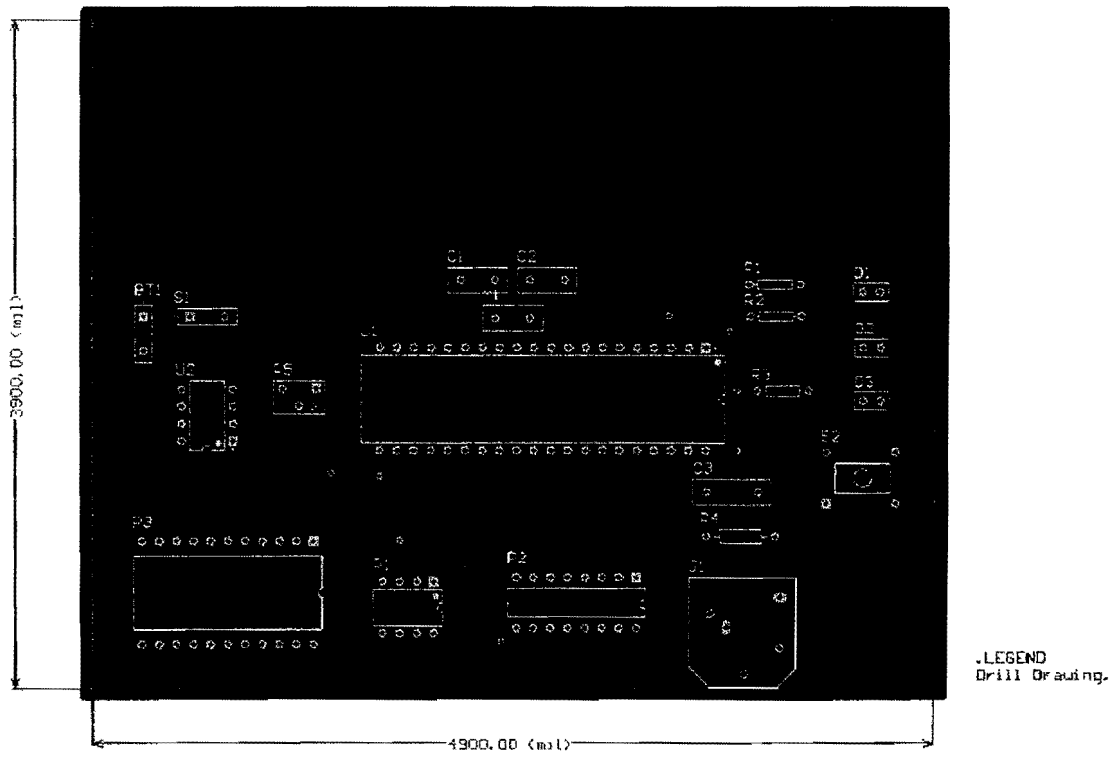
```
// Código de ingreso de datos de paciente
//*****************************************
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Audiometría
{
    public partial class Form2 : Form
    {
        public string sApellido;
        public string sNombre;
        public string sFecha;
        public string sColegio;
        public string sGrado;

        //*****************************************
        // Inicialización del componente
        //*****************************************
        public Form2()
        {
            InitializeComponent();
        }

        //*****************************************
        // Cuadro de texto para ingresar datos
        //*****************************************
        public void button1_Click(object sender, EventArgs e)
        {
            sApellido = textBox1.Text;
            sNombre = textBox2.Text;
            sFecha = textBox3.Text;
            sColegio = textBox4.Text;
            sGrado = textBox5.Text;
        }
    }
}

//*****************************************
// Universidad del Valle de Guatemala
// Ingeniería Electrónica
// Trabajo de Graduación
// Sergio Estrada 08032
// Código de generación e impresión de gráficos
//*****************************************
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Printing;
```

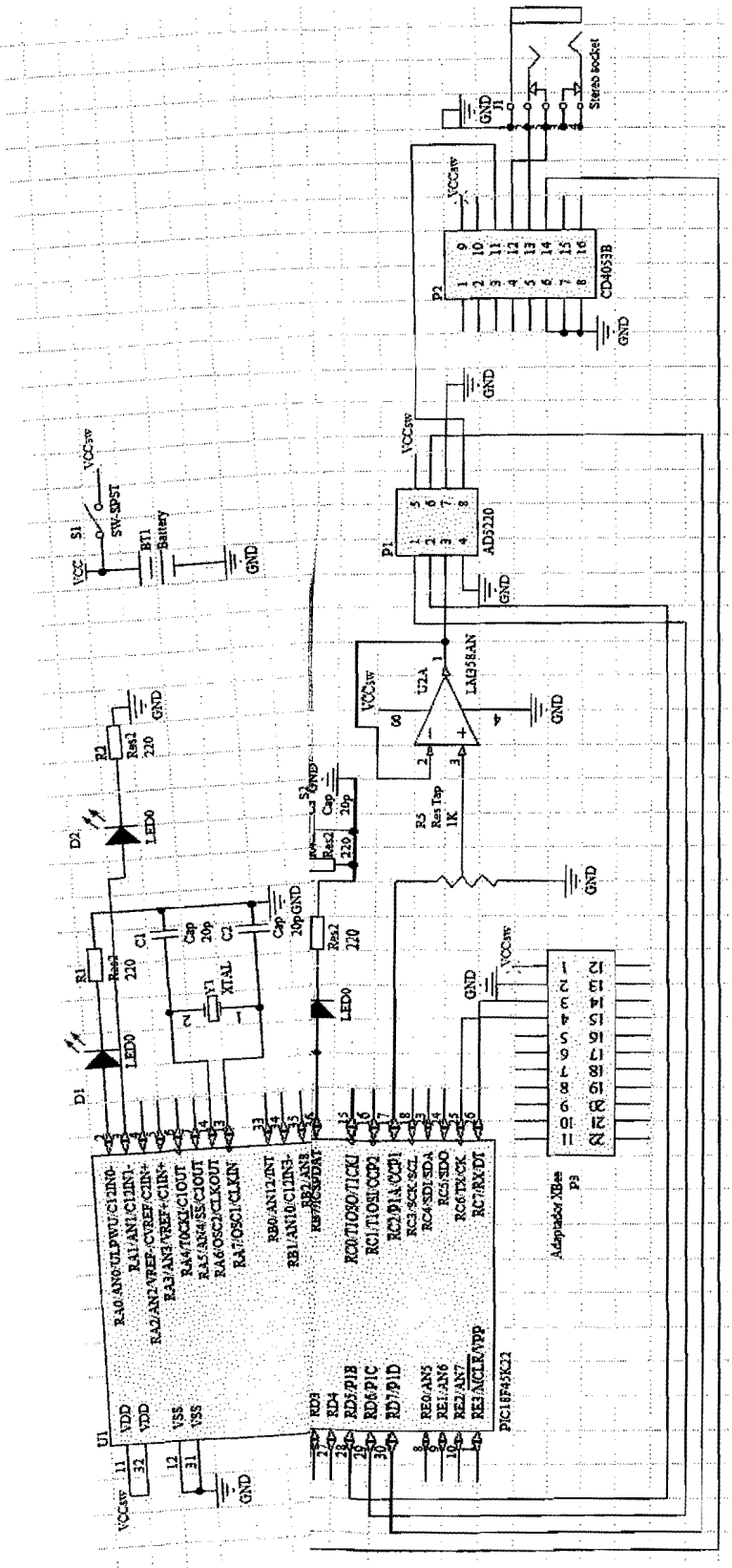
```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using ZedGraph;
using System.Windows;

namespace Audiometría
{
    public partial class Form3 : Form
    {
        GraphPane elPanel = new GraphPane();
        GraphPane elPanel2 = new GraphPane();
        PointPairList listaDerecha = new PointPairList();
        PointPairList listaIzquierda = new PointPairList();

        LineItem curva1;
        LineItem curva2;

        List<string> lista = new List<string>();
        List<string> lista2 = new List<string>();
        int[] ejeX = new int[] { 250, 500, 1000, 2000, 3000, 4000,
6000, 8000 };

        //*****************************************
        // Inicialización de componente gráficos
        //*****************************************
        public Form3(List<string> listadoGrafica, List<string>
listadoDatos)
        {
            InitializeComponent();
            button1.Show();
            lista = listadoGrafica;
            lista2 = listadoDatos;
            for (int i = 2; i < 18; i++)
            {
                if (lista[i] == "")
                {
                    lista.RemoveAt(i);
                    lista.Insert(i, "0");
                }
            }
        }

        //*****************************************
        // Generación de gráficos
        //*****************************************
        private void Form3_Load(object sender, EventArgs e)
        {
            elPanel = zedGraphControl1.GraphPane;
            elPanel2 = zedGraphControl2.GraphPane;
            elPanel.Title.Text = "Derecha";
            elPanel2.Title.Text = "Izquierda";

            elPanel.XAxis.Title.Text = "Frecuencia (Hz)";
            elPanel.YAxis.Title.Text = "Potencia (dB)";
```

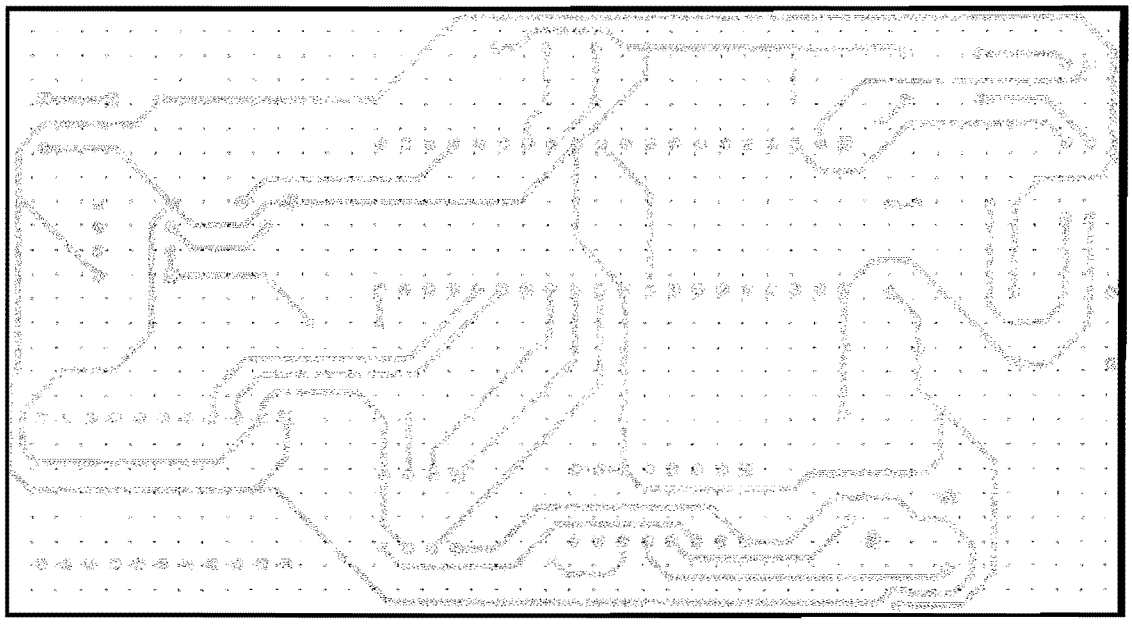
```
            elPanel2.XAxis.Title.Text = "Frecuencia (Hz)";
            elPanel2.YAxis.Title.Text = "Potencia (dB)";

            for (int i = 0; i < 8; i++)
            {
                listaDerecha.Add(ejeX[i], Int16.Parse(lista[i + 2]));
            }

            for (int i = 0; i < 8; i++)
            {
                listaIzquierda.Add(ejeX[i], Int16.Parse(lista[i +
10]));
            }

            agregar_datos();
            curva1 = elPanel.AddCurve(null, listaDerecha, Color.Red,
SymbolType.Circle);
            curva2 = elPanel2.AddCurve(null, listaIzquierda,
Color.Blue, SymbolType.XCross);
            zedGraphControl1.AxisChange();
            zedGraphControl2.AxisChange();
        }

        //*****************************************
        // Agregar datos del paciente
        //*****************************************
        private void agregar_datos()
        {
            label6.Text = lista2[0];
            label7.Text = lista2[1];
            label8.Text = lista2[2];
            label9.Text = lista2[3];
            label10.Text = lista2[4];
        }

        //*****************************************
        // Impresión de gráficos
        //*****************************************
        private void button1_Click(object sender, EventArgs e)
        {
            button1.Hide();
            this.ControlBox = false;
            PrintDocument pd = new PrintDocument();
            pd.PrintPage += new PrintPageEventHandler(PrintImage);
            printDialog1.AllowSomePages = true;
            printDialog1.ShowHelp = true;
            printDialog1.Document = pd;
            DialogResult result = printDialog1.ShowDialog();

            if (result == DialogResult.OK)
            {
                pd.Print();
            }
            button1.Show();
            this.ControlBox = true;
        }
```

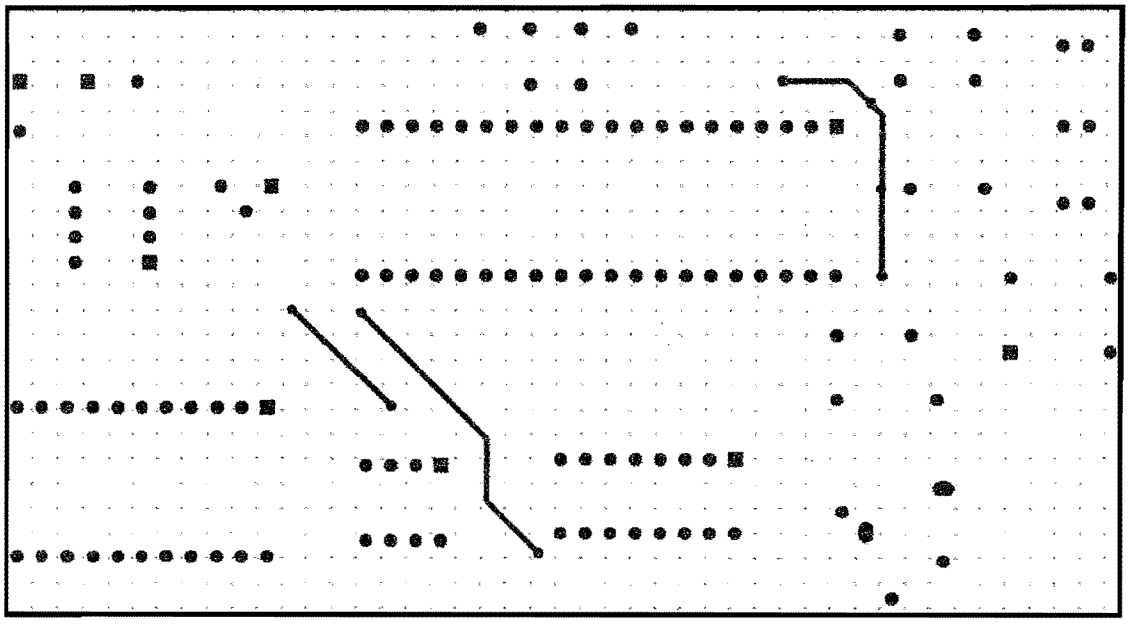
```
        //*****************************************
        // Método de captura de pantalla
        //*****************************************
        void PrintImage(object o, PrintPageEventArgs e)
        {
            int x = SystemInformation.WorkingArea.X;
            int y = SystemInformation.WorkingArea.Y;
            int width = this.Width;
            int height = this.Height;
            int top = this.Top;
            int left = this.Left;
            int right = this.Right;
            int bottom = this.Bottom;

            Rectangle bounds = new Rectangle(x, y, width, height);
            Bitmap img = new Bitmap(width, height);

            this.DrawToBitmap(img, bounds);
            Point p = new Point(5, 20);
            e.Graphics.DrawImage(img, p);
            img.Save(img + ".bmp");
        }
    }
}
```

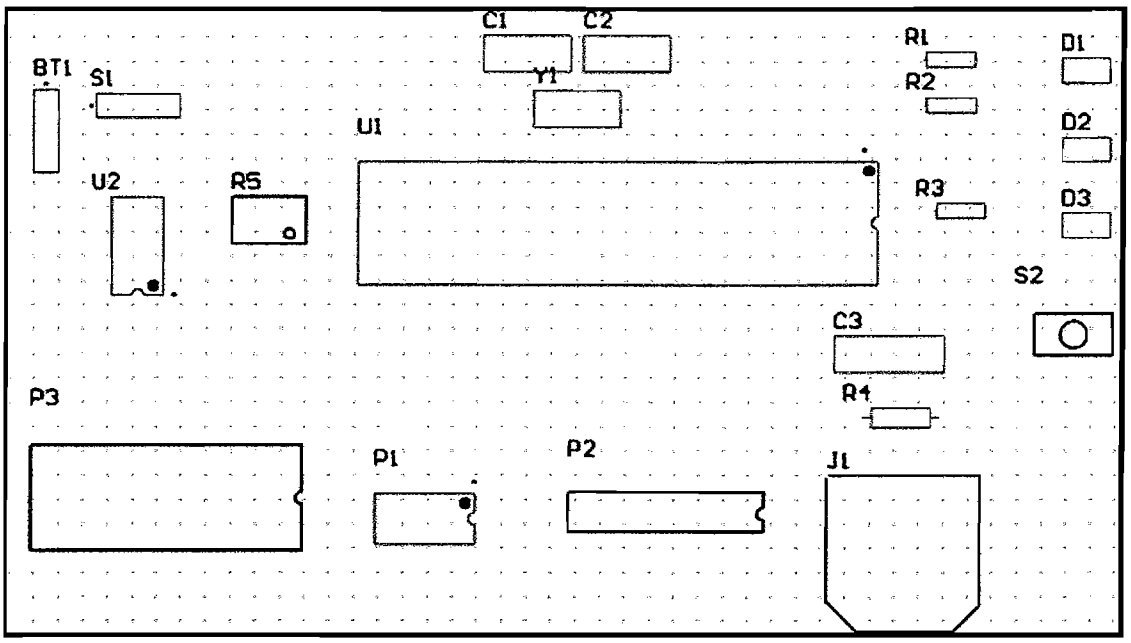
C. ARCHIVOS DEL DISEÑO DE PCB




U1
PIC18F45K22
P1
AD9220
P2
CD4053B
P3
Adaptador XBee
U2A
LM358AN
J1
Stereo socket
S1
SW-SPST
BT1
Battery
Y1
XTAL
VCC
VCCsw
GND
LED0
Res2
220
Cap
20p
Res Trap
1K






C1
C2
R1
D1
BT1
S1
Y1
R2
U1
D2
U2
R5
R3
D3
S2
C3
P3
R4
P1
P2
J1