UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA
ELECTRICA – PROCESAMIENTO DIGITAL DE SEÑALES

DISEÑO DEL CONTROL ELECTRÓNICO PARA EMISIÓN Y ADQUISICIÓN DE SEÑALES DE UN TOMÓGRAFO ULTRASÓNICO DE DOS ELEMENTOS

TESIS

QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN INGENIERÍA

PRESENTA:
ISAIAS ALCUDIA CALLEJA

TUTOR PRINCIPAL:

DRA. LUCIA MEDINA GOMEZ, FACULTAD DE CIENCIAS

MÉXICO, D. F. OCTUBRE 2014

JURADO ASIGNADO:

Presidente:	Dr. Perez Alcázar Pablo Roberto
Secretario:	M.I. Escobar Salguero Larry
Vocal:	Dra. Medina Gómez Lucia
1 er. Suplente:	Dr. Solís Nájera Sergio Enrique
2 d o. Suplente:	Dr. Psenicka Bohumil

Lugar o lugares donde se realizó la tesis: FACULTAD DE CIENCIAS

TUTOR DE TESIS:

Dra. Lucia Medina Gómez

_____

FIRMA

Dedicatoria

- A Selene, por todo su cariño y motivación en los momentos más complicados de mi vida.

- A mis padres que siempre me apoyaron en mi trayectoria escolar.

- A Kanji que siempre se desvelaba junto a mí.

Agradecimientos

- A la UNAM por brindarme los conocimientos y permitirme estudiar un posgrado.

- A la Dra. Lucía Medina Gómez, por su increíble paciencia, tiempo y apoyo en el desarrollo de este proyecto de tesis.

- A la DGAPA por la beca otorgada para la elaboración del trabajo de tesis, proyecto DGAPA-PAPIIT IT118811-3: "Diseño de un sistema básico de tomografía ultrasónica en transmisión para materiales altamente heterogéneos".

- A la SEP y CONACyT, por el proyecto SEP-CONACyT 131937: "Análisis de la velocidad de fase y atenuación acústica en materiales heterogéneos y dispersivos".

- Al CONACyT por la beca otorgada en el periodo 2012-2013 durante el posgrado de ingeniería.

Índice general

Índice de figuras

Índice de tablas

Introducción

Tomografía se refiere a la adquisición de imágenes desde diferentes ángulos de iluminación, generando rebanadas que contienen información sobre la estructura interna del medio en estudio, utilizando fuentes de energía capaces de penetrar el medio.

Los sistemas más comunes tanto para aplicaciones médicas como industriales son los basados en rayos X y ultrasonido.

La tomografía axial computarizada (CAT) originalmente desarrollada en la década de los 1970s para aplicaciones médicas, se basa en la capacidad de las ondas electromagnéticas de alta frecuencia para atravesar el medio de propagación. Los elementos básicos que conforman el sistema son una fuente, un detector de rayos X (transmisor y receptor) y un sistema mecánico para obtener información desde diferentes posiciones o arreglo de sensores. El objeto bajo estudio se sitúa entre la fuente y el detector y se envía una descarga de rayos X desde la fuente hacia el objeto, midiendo la intensidad de la radiación que llega al detector, la cual depende tanto de la densidad mineral (número atómico) como la microestructura del objeto.

La tomografía ultrasónica está conformada por dispositivos que generan pulsos mecánicos de alta frecuencia que permiten explorar la microestructura del material en estudio, mediante los mapas de reflectividad o atenuación de la onda ultrasónica.

Existen dos modalidades de tomografía ultrasónica:

i) Pulso-eco, donde el(los) transductor(es) actúan como emisores y receptores, comúnmente utilizada cuando no se tiene acceso a todas las superficies del material.

ii) Modo en transmisión, donde los emisores y receptores son independientes tanto en funcionamiento como para el movimiento mismo de los elementos.

Estas técnicas son muy seguras, ya que tanto el medio como el que opera el equipo de ultrasonido no son expuestos a radiación ionizante y no sufren daño; su costo es bajo si se le compara con otros métodos y el procedimiento es rápido de realizar.

Objetivo de la tesis

El objetivo de este proyecto es desarrollar un sistema electrónico de tomografía ultrasónica de dos elementos: un receptor y un emisor.

Este sistema tiene la capacidad de controlar motores a pasos para ajustar una posición para la emisión y recepción de pulsos ultrasónicos, con la posición de los motores y almacenar los datos recibidos para formar los sinogramas. Además el sistema tiene una interfaz gráfica que permita al usuario excitar al transductor emisor con diversos pulsos, configurar el movimiento de los motores a pasos, la frecuencia de muestreo y el número de datos que se requieren para aplicaciones específicas.

Estructura del trabajo de la tesis

El proyecto de tesis está definido por cuatro capítulos, los cuales muestran los principios de la tomografía por ultrasonido y su actualidad, además de plantear la necesidad de desarrollar un módulo de ultrasonido para el estudio posterior de temas en procesamiento de señales y generación de imágenes por ultrasonido.

- El primer capítulo describe los principios y modalidades para la adquisición del ultrasonido, dándose una breve introducción a la descripción del tomógrafo de ultrasonido básico desarrollado en este trabajo de tesis.

- El segundo capítulo explica y describe el desarrollo, la electrónica desarrollada que controla el movimiento y posición a los elementos del tomógrafo y los diversos componentes que integran el sistema de emisión y recepción de ultrasonido de dos elementos y la interfaz gráfica de usuario desarrollada.

- El tercer y cuarto capítulos hacen referencia a los resultados de cada etapa y del sistema integrado.

- Por último, las conclusiones y el trabajo futuro son descritas en el capítulo cinco.

Capítulo 1. Sistemas de Ultrasonido y Despliegue Gráfico de Señales de Ultrasonido

En diversos sistemas de ultrasonido se tienen distintos arreglos de transductores que, dependiendo de la aplicación, pueden trabajar a rangos diferentes de frecuencia. En este capítulo se mostrará como se puede inspeccionar un objeto con ultrasonido y las formas de desplegar gráficamente las señales de ultrasonido adquiridas.

El ultrasonido es energía mecánica que se propaga a frecuencias mayores al rango audible humano (ver figura 1-1), que, dependiendo de las características del medio, pueden dar lugar a diferentes tipos de ondas: longitudinales, transversales o de corte, superficiales u ondas de Lamb.

Las ondas longitudinales son ondas mecánicas en las que su movimiento de oscilación de las partículas del medio es paralelo a la dirección donde se propaga la onda, consisten en cambios de presión de forma alternante; compresiones (alta presión) y expansiones (baja presión) y se generan en cualquier tipo de medio. En las ondas transversales, el desplazamiento del medio es perpendicular a la dirección propagación de la onda, y sólo se pueden generan en fluidos viscosos o sólidos debido a que existen esfuerzos de corte entre las partículas oscilantes.

Los fenómenos que se presentan al interactuar con el ultrasonido y el medio o medios en estudio son: transmisión, reflexión, dispersión, refracción y scattering, que dependerán de las propiedades acústicas, dimensiones y microestructura e inhomogeneidades del medio.

Para caracterizar acústicamente medios heterogéneos y generar la imagen de su estructura es importante considerar la atenuación de la onda transmitida, así como, la capacidad de los medios para permitir el paso de la onda acústica, conocida como impedancia acústica. La atenuación es función de la frecuencia y de la distancia viajada por la onda, y la impedancia acústica es una propiedad de los materiales que depende de la velocidad de propagación acústica en el medio y su densidad.

Figura 1-1. Espectro acústico y sus aplicaciones.

1.1 Método para la inspección ultrasónica

Las pruebas ultrasónicas son inspecciones versátiles, siendo las más comúnes las técnicas de pulso-eco y transmisión directa, las cuales pueden realizarse en contacto o en inmersión, dependiendo de las facilidades de acoplamiento entre los transductores y el material inspeccionado.

La técnica de pulso eco consiste en enviar un pulso ultrasónico y detectar la energía reflejada por el medio usando el mismo transductor, como se muestra en la figura 1-2. Los ecos detectados son la representación de la energía reflejada debido a la presencia de discontinuidades, microestructura y superficies del medio de propagación. Esta modalidad es de gran utilidad para la localización de fracturas o inhomogeneidades mediante el cálculo de tiempo de vuelo y atenuación.

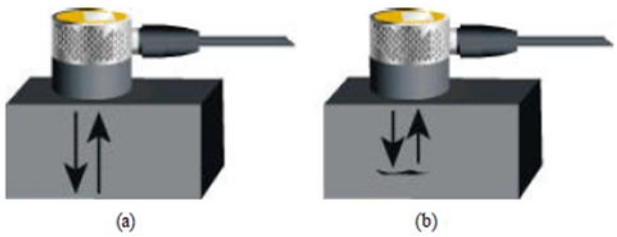


Figura 1-2. Ultrasonido pulso-eco [1].

La técnica de transmisión directa (ver figura 1-3) consiste en dos transductores ubicados en los lados opuestos del espécimen, de los cuales uno actúa como emisor y el otro como receptor. En esta modalidad, la reflexión no es detectada, únicamente la energía transmitida a través del medio en estudio. En presencia de discontinuidades parciales o totales (fracturas), la trayectoria, amplitud y forma de la onda ultrasónica es modificada, e inclusive puede que dicha onda no se detecte.

Las aplicaciones más comunes son la detección de discontinuidades de tamaño comparable o menor a la longitud de onda de la energía emitida y la caracterización de materiales heterogéneos y altamente dispersivos. Cabe mencionar que para el caso de no detección de energía transmitida la información es valiosa ya que se puede determinar la presencia de fracturas de tamaño mayor a la longitud de onda.

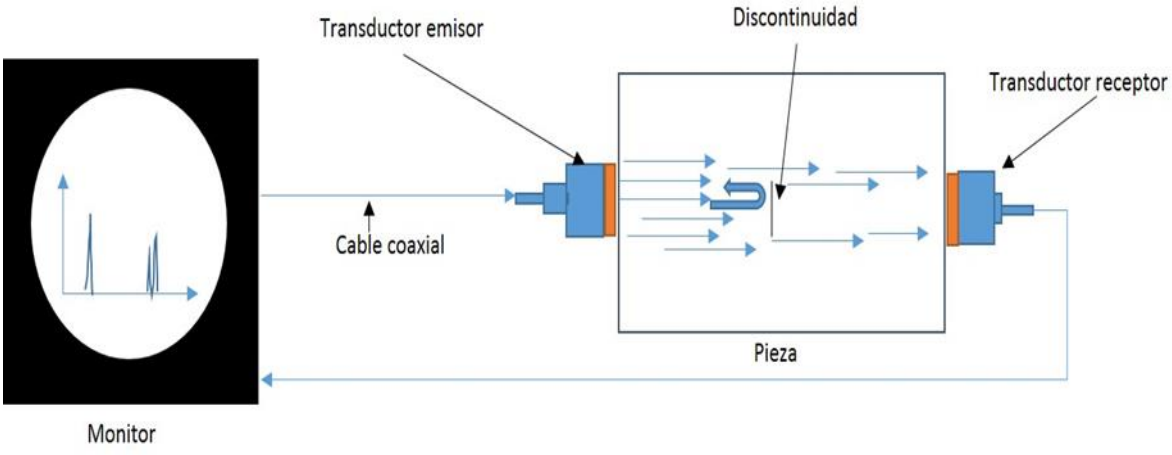


Figura 1-3. Método de transmisión directa.

1.2 Modalidades de despliegue gráfico

Las señales adquiridas en las inspecciones ultrasónicas, utilizando cualquiera de las dos modalidades descritas en la sección 1.1, se pueden representar como la variación del voltaje respecto al tiempo en un osciloscopio, o bien como intensidad o absorción de la energía transmitida o reflejada en un plano espacial o inclusive en 3D ó 4D, con el uso de arreglos de transductores que permiten la detección de la energía en diversos puntos del espacio.

Los tres formatos más comunes para desplegar la información adquirida son:

- Método en modo A.
- Método en modo B.
- Método en modo C.

Cada forma de presentación ofrece una manera diferente de observar y evaluar la región de material que se está inspeccionando.

1.2.1 Modo A

Este método de representación de ultrasonido es el más sencillo, ya que muestra el cambio del voltaje o amplitud respecto al tiempo. La figura 1-4 muestra los ecos producidos al utilizar la modalidad de pulso-eco, donde los ecos de mayor intensidad se refieren a reflexiones de la pared del objeto inspeccionado y los ecos intermedios se refieren a una fractura en el interior del material. La figura 1-5 se refiere a la modalidad de transmisión directa, donde el pulso de mayor amplitud se refiere al tiempo que la onda viaja del emisor al receptor y t_1 es el tiempo que viaja del receptor a la fractura y regresa al receptor.

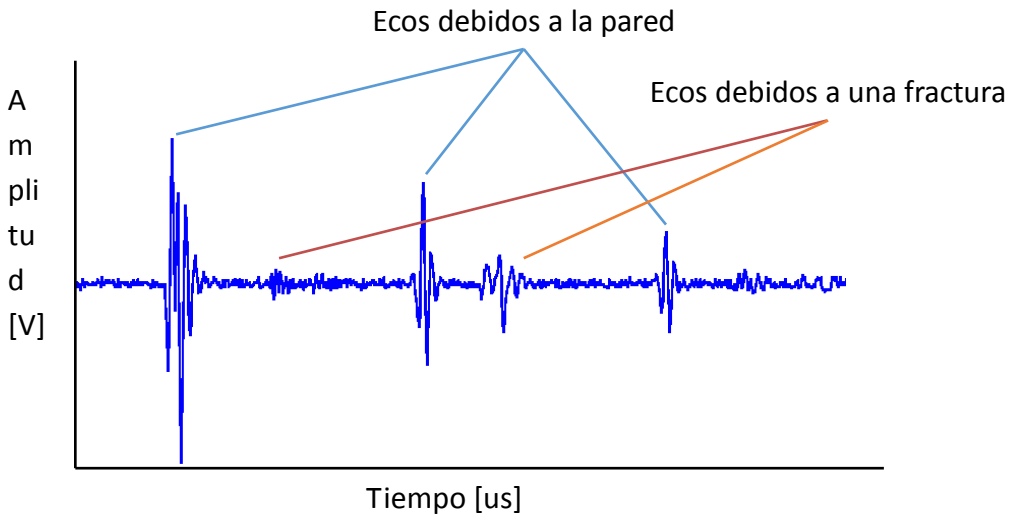


Figura 1-4. Representación general de una señal pulso-eco en modo A.

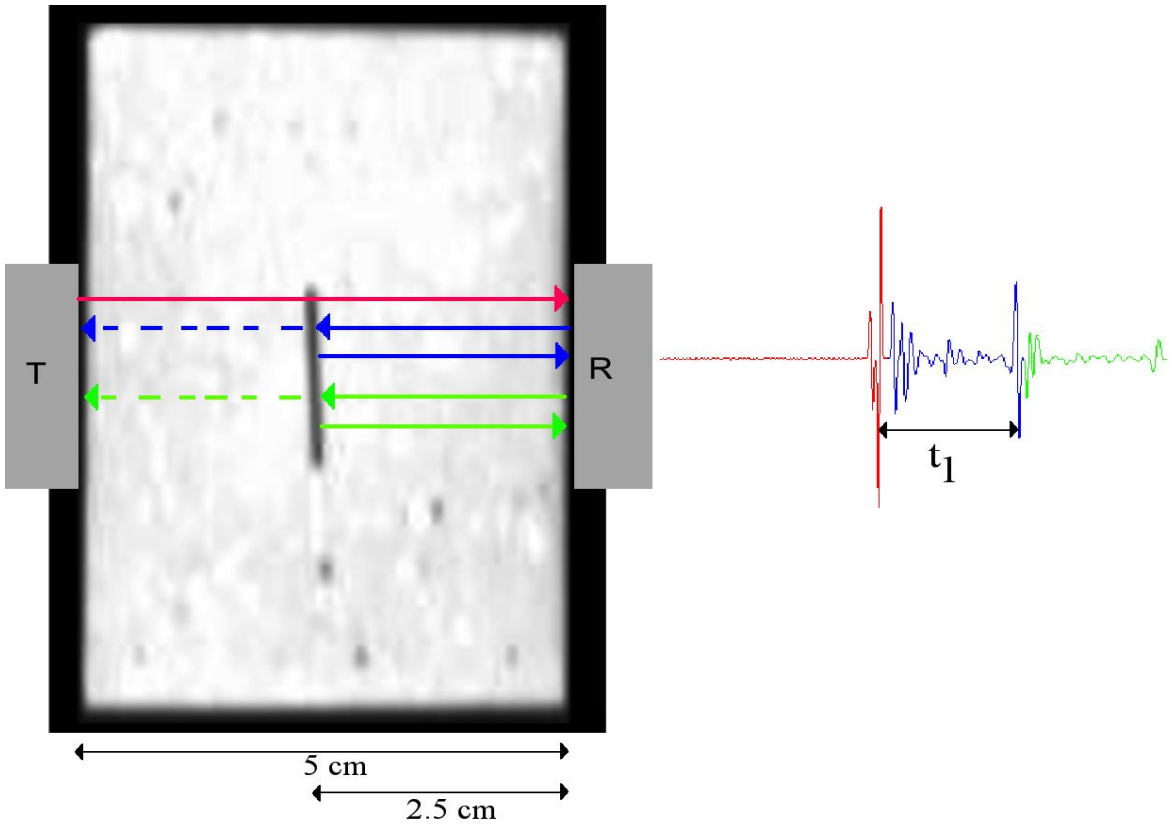


Figura 1-5. Señales adquiridas en transmisión directa.

1.2.2 Modo B

En el escaneo del objeto con el "modo B", la imagen que se forma generalmente corresponde a una sección transversal del objeto (un sector del área del objeto), y la imagen se obtiene a partir de procesar los ecos recibidos por el sistema de ultrasonido. Existen varias representaciones de modo B, la más sencilla es generar la imagen de intensidades basado en un conjunto de señales en modo A, adquiridas de distintas regiones iluminadas.

Otra posibilidad, donde se obtiene mayor información sobre la estructura interna del medio, es mediante la reflectancia, que requiere de considerar dos parámetros importantes:

1. La distancia del objeto a los transductores, llamada *rango*.

2. La dirección de la parte activa del transductor (la posición del transductor emisor), es decir, la orientación de la onda ultrasónica.

Para obtener el rango o la distancia (d_n) al objeto, se utiliza el tiempo de vuelo (t_n) que le toma a la onda viajar del transductor al objeto reflector y la velocidad de propagación (c), mediante

$$d_n = \frac{t_n c}{2} \qquad (1\text{-}1)$$

Donde t_n se calcula como el tiempo transcurrido entre la emisión del pulso y la detección del eco como se muestra en la figura 1-6.

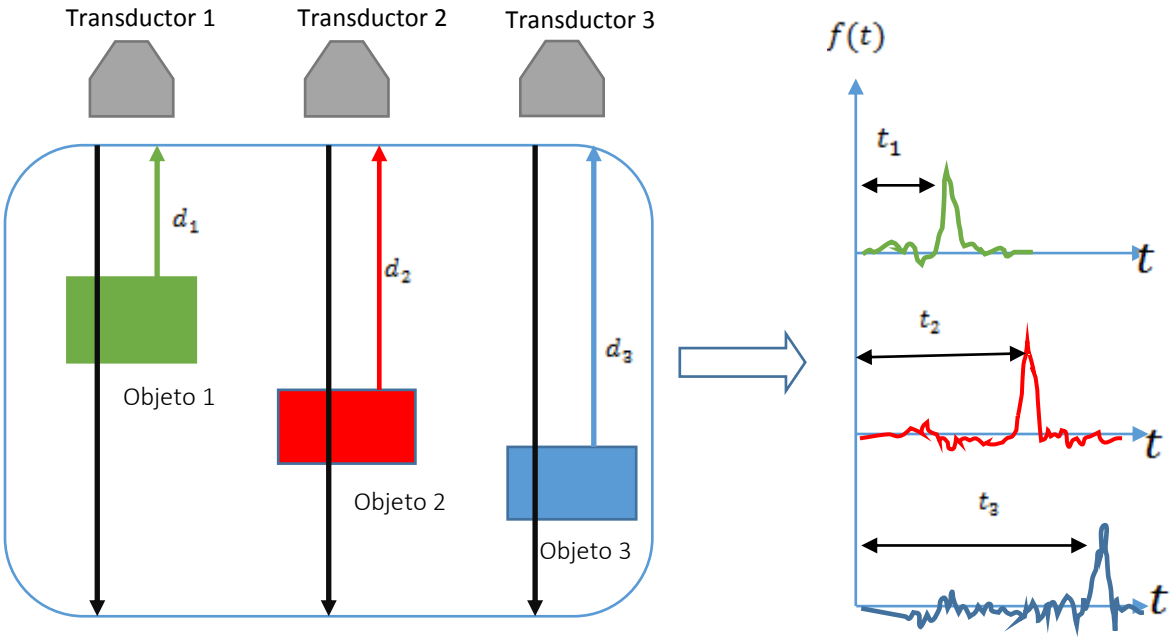


Figura 1-6. Tiempo de vuelo para una escaneo lineal en pulso eco.

1.3 Señales tomográficas: sinogramas.

En general, un sinograma es un conjunto de señales adquiridas (proyecciones) durante la inspección de un material o un objeto cualquiera y están arregladas en

una forma especial, con el conjunto de todas estas señales y un algoritmo de reconstrucción de imágenes adecuado para ultrasonido se puede obtener la imagen de una sección del objeto inspeccionado.

El sinograma depende de la forma en cómo se adquieren dichas señales de ultrasonido y del número de proyecciones que se tomen cuando se realiza una inspección del material.

Los sinogramas se adquieren de diversas formas dependiendo de la trayectoria que realicen los transductores mediante el movimiento de control automático. Una de las más comunes es el movimiento circular permitiendo escanear el material en estudio desde diversos ángulos de visión. En cada posición emisor-receptor la señal es adquirida. Un ejemplo de forma de rastreo circular se muestra en la figura 1-7, donde el emisor y el receptor se encuentran a la misma distancia uno frente al otro de forma radial, durante el movimiento angular, adquiriendo información de una región transversal del objeto analizado. En la figura 1-8, el arreglo emisor-receptor consiste de un emisor fijo en una posición y el receptor se mueve angularmente cubriendo 360°; una vez cubierta la trayectoria, el emisor se mueve angularmente cierta cantidad y el receptor inicia su trayectoria circular.

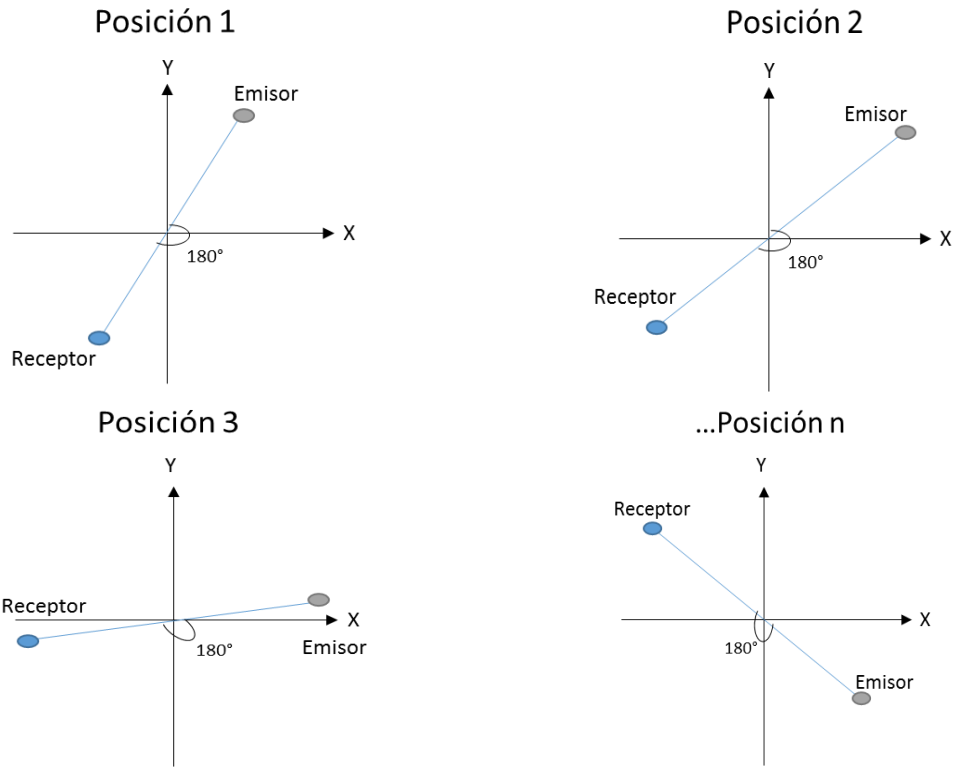


Figura 1-7. Muestra las diferentes posiciones de los transductores cuando se adquiere en forma radial.

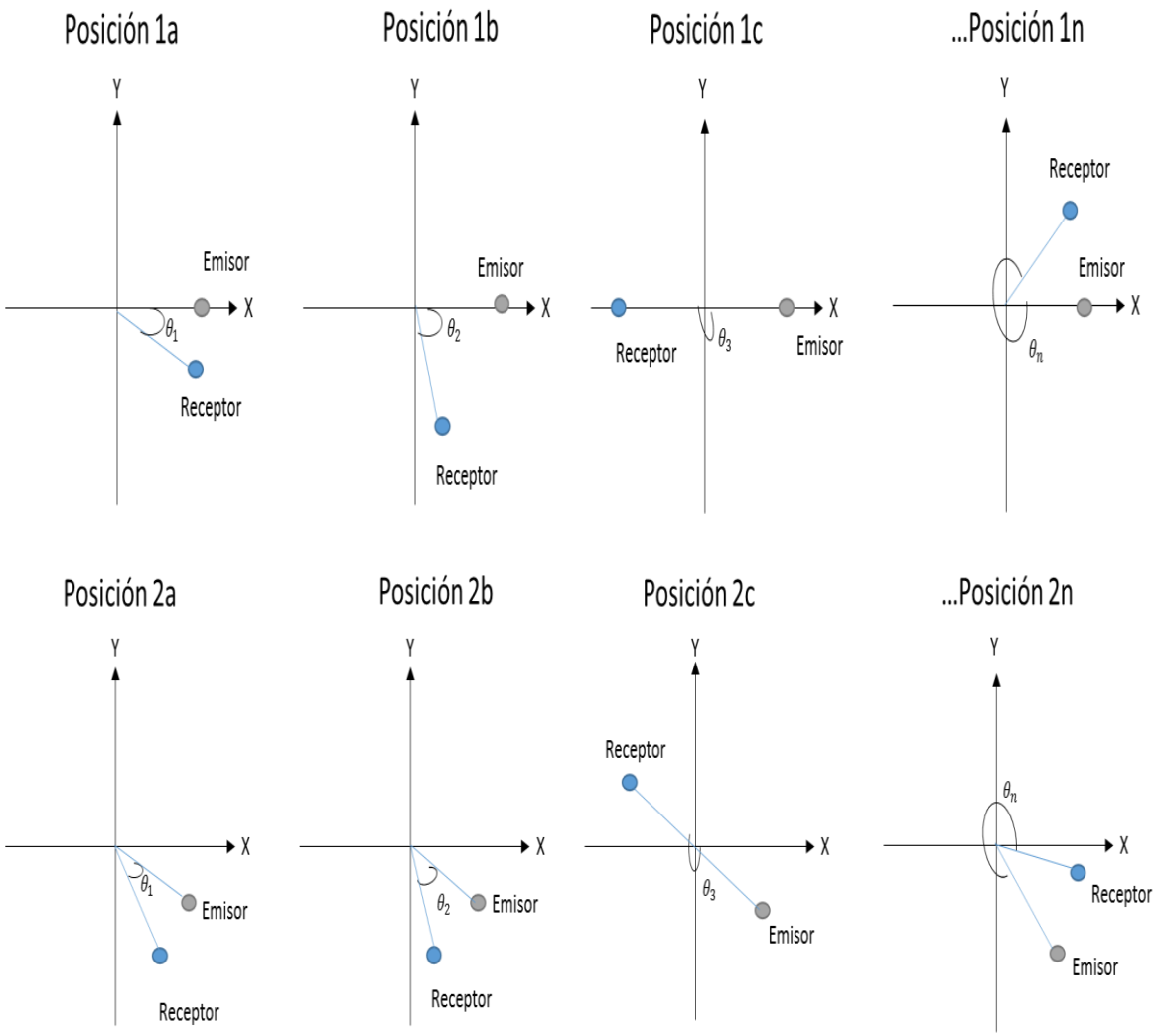


Figura 1.8. Muestra solo dos posiciones del emisor empleando el formato en abanico.

Para obtener una imagen de la sección del objeto con los sinogramas, es necesario desarrollar un algoritmo de reconstrucción de imágenes tomograficas por ultrasonido que trabaje con los parámetros de adquisición de las señales de ultrasonido y modalidades de despliegue de las señales que se están obteniendo en el proceso de tomografía.

Capítulo 2. Descripción de un Sistema Tomográfico de Ultrasonido

El propósito de este capítulo consiste en describir el diseño y desarrollo de un sistema de emisión y adquisición de ultrasonido, incluyendo la electrónica asociada y el despliegue gráfico para que el usuario pueda controlar la emisión y adquisición de las señales ultrasónicas.

La tomografía es una técnica de generación de imágenes que permite visualizar la estructura interna del objeto en estudio, mediante múltiples proyecciones para reconstruir las imágenes de secciones transversales del medio. Para realizar la adquisición generalmente se utilizan dos modalidades: pulso eco y transmisión, ambos permiten la reconstrucción de la información basados en las propiedades elásticas tales como: la velocidad de propagación, el índice de refracción o el coeficiente de atenuación.

Un sistema de tomografía ultrasónico consiste en cuatro partes (módulos) importantes:

1. Sistema mecánico.
2. Sistema de control electrónico de motores a paso.
3. Sistema de emisión y recepción de datos.
4. Software de usuario, la interfaz.

Como se muestra en la figura 2-1.

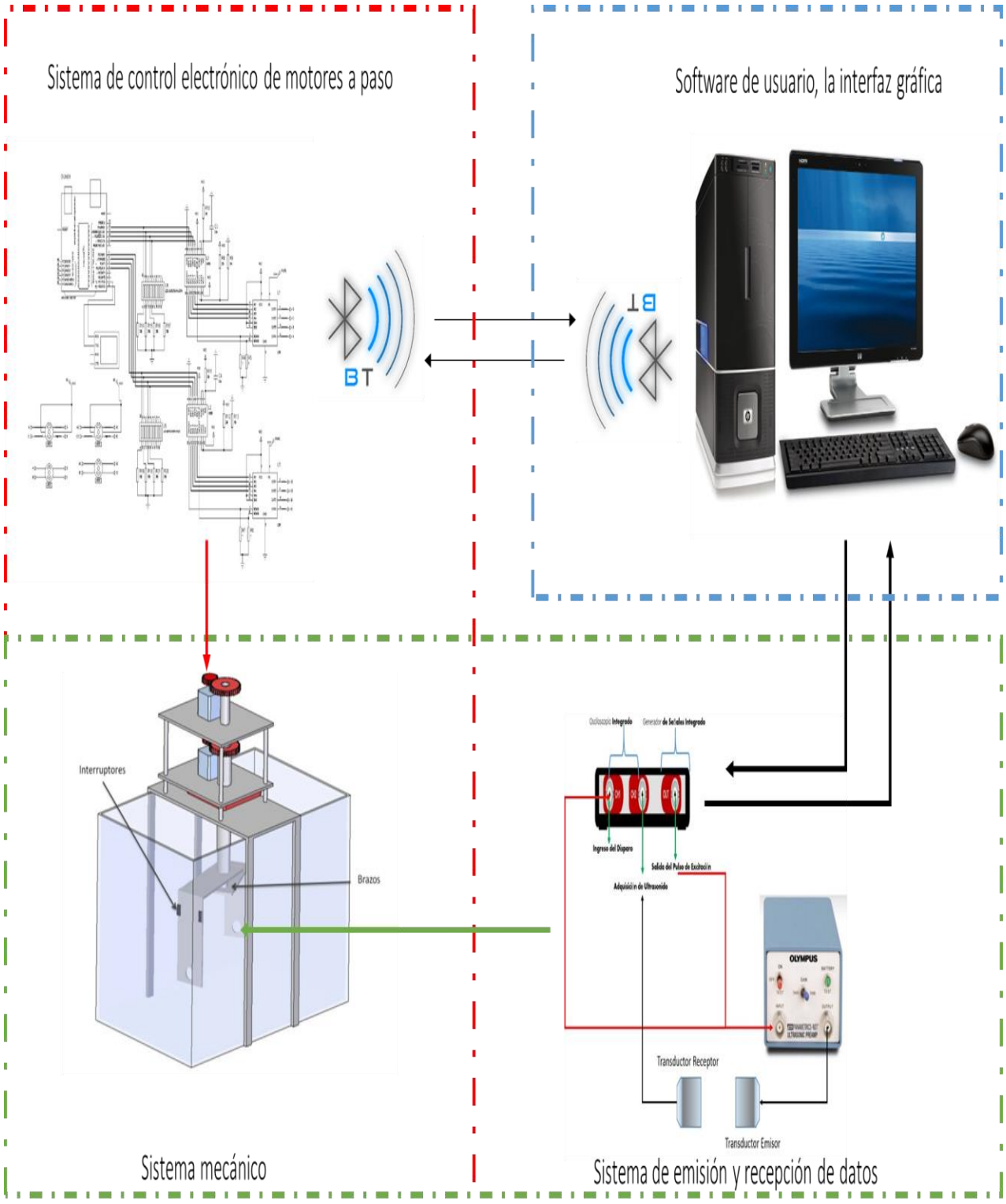


Figura 2-1. Desarrollo general del sistema de tomografía ultrasónica de dos elementos.

2.1 Sistema mecánico

El sistema mecánico consta de dos juegos de engranes (dos trenes de engranes) distribuidos en un soporte metálico (ver figura 2-2). En el diseño de los engranes se consideró el diámetro primitivo y el número de dientes [2]. Estos parámetros están relacionados por la expresión:

$$\frac{N_1}{N_2} = \frac{D_1}{D_2} \qquad (2\text{-}1)$$

donde N_1, N_2 corresponde al número de dientes de cada engrane y D_1, D_2 son los diámetros respectivos de cada engrane.

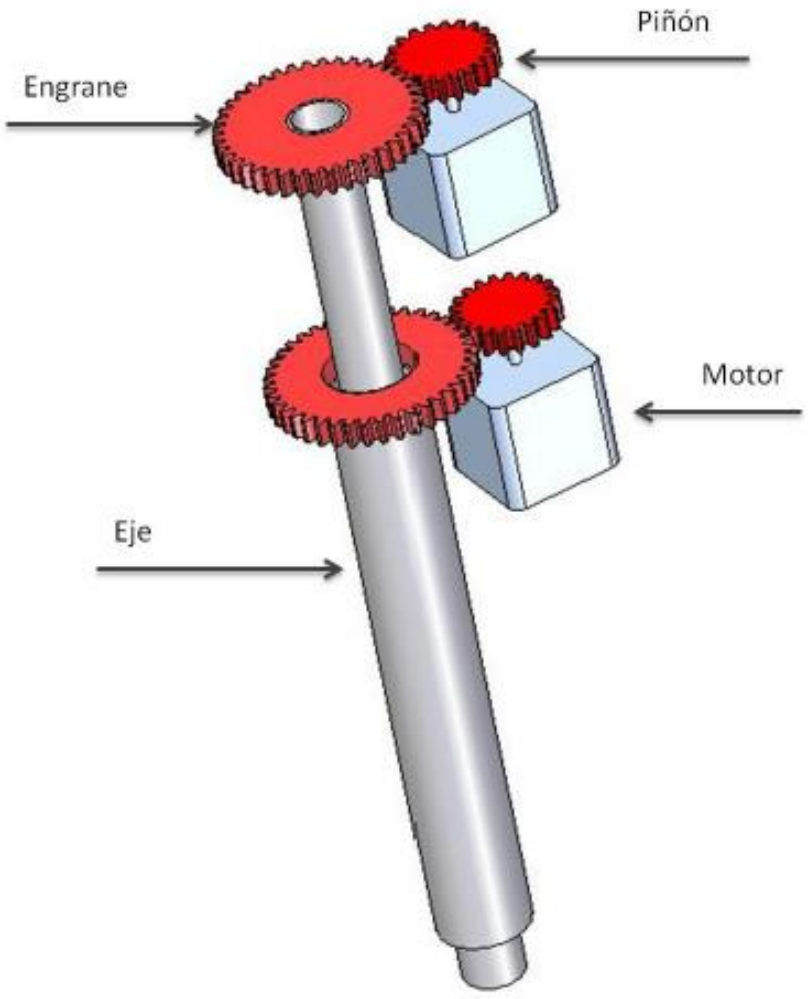


Figura 2-2. Acoplamiento de engrane y piñón [3].

El tren de engranes y los ejes se montan sobre los soportes y las chumaceras que se encuentran atornilladas sobre una base metálica, lo que le permite tener sujeción y hacer girar a los ejes (ver figura 2-3).

Este tipo de unión permite que se inicie el movimiento entre los engranes, y que los motores se mantengan fijos evitando que exista un funcionamiento incorrecto entre los trenes de engranes. El soporte principal está unido por medio de cuatro tornillos con sus respectivas tuercas, de manera que también se puede dar la altura a los transductores y llevar a cabo diversas pruebas de análisis [3].

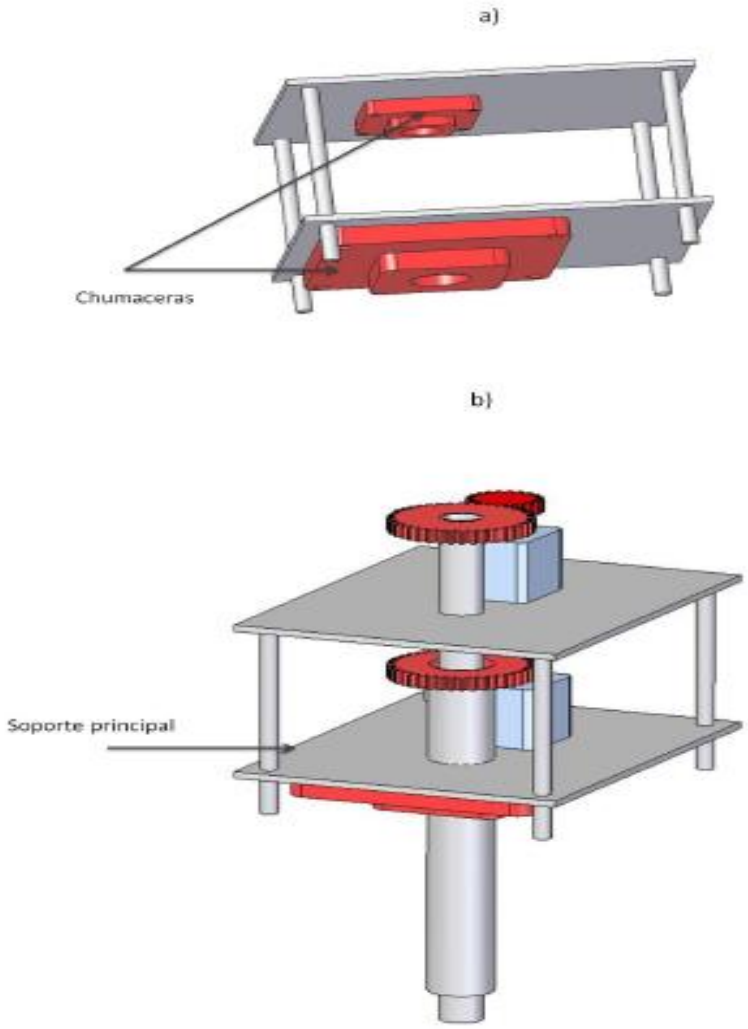


Figura 2-3. Soportes implementados en el tomógrafo ultrasónico a) Chumaceras b) Montaje del tren de engranes en el soporte [3].

Las características físicas y de diseño del sistema engranes-piñón se muestran en la tabla 2-1.

Característica	***Engrane***	***Piñón***
Diámetro [mm]	80	40
Numero de dientes	40	20
Paso diametral [mm]	0.5	

Tabla 2-1. Parámetros básicos de diseño de los trenes de engranes del sistema mecánico [2].

El movimiento de los transductores ultrasónicos se realiza por medio de dos ejes concéntricos acoplados a dos brazos que los sujetan. En la figura 2-4 se muestra la estructura rotacional del diseño propuesto.

El sistema mecánico tiene las especificaciones de posicionamiento siguientes:

a) El movimiento de los brazos es concéntrico.

b) La distancia entre los brazos es de 30 [cm] cuenta con la posibilidad de disminuir la distancia entre ellos.

c) La posición de los brazos es controlado, por cada motor a pasos de manera independiente.

Los brazos están alineados para garantizar que las caras de los transductores no formen ángulo respecto a su eje principal, en todo momento [3].

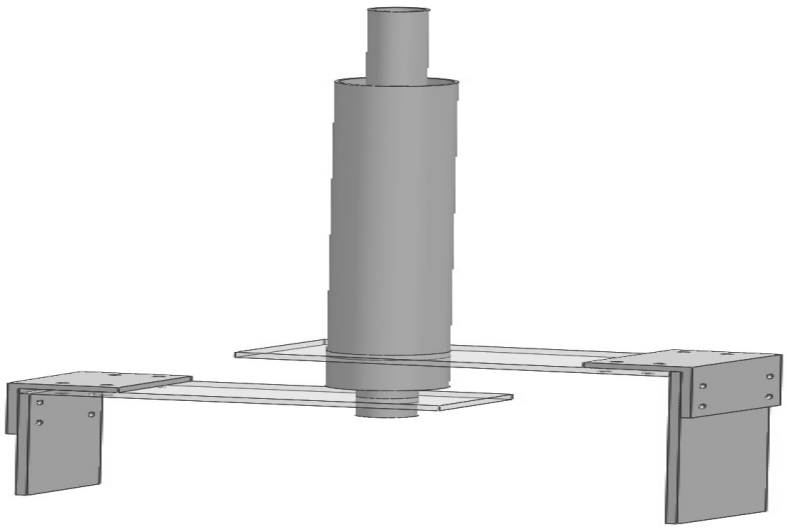


Figura 2-4.Diseño de los ejes concéntricos usados en el tomógrafo ultrasónico [3].

El sistema mecánico integrado se muestra en la figura 2-5.

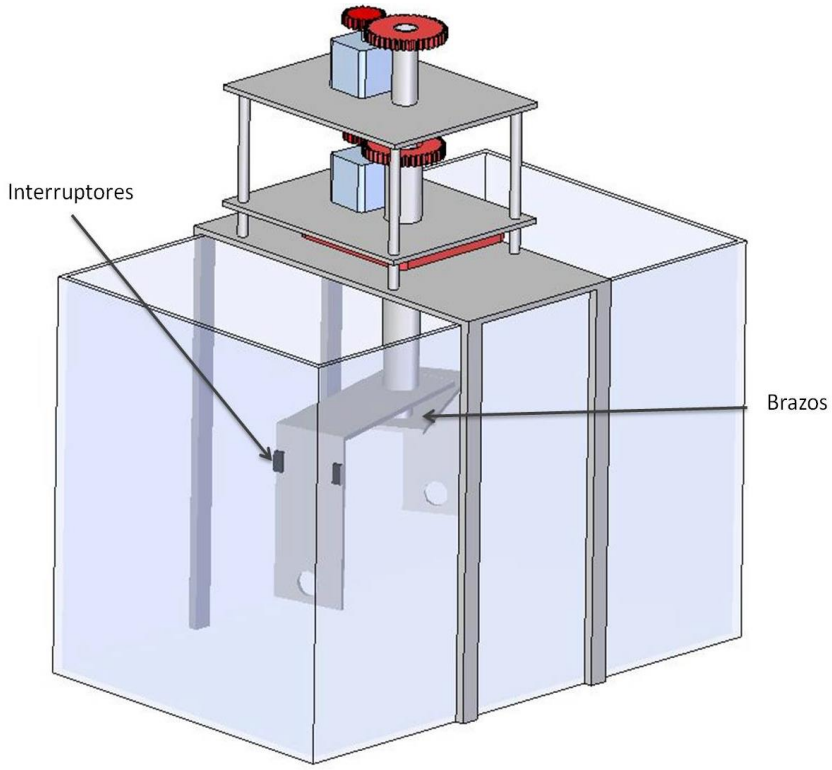


Figura 2-5. Sistema mecánico con dos motores a paso.

Los motores a pasos (ver figura 2-6) en el sistema mecánico tiene las siguientes características:

- 10V de alimentación de corriente directa.
- 2.5A de corriente directa máxima por cada bobina.
- 1.8° por paso.

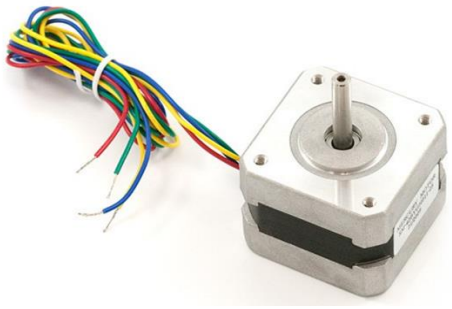


Figura 2-6. Motores a pasos Step Syn® con la capacidad de 1.8° por paso, el color del cable indica una fase o una bobina.

2.2 Sistema de control electrónico de motores a paso

El sistema de control electrónico tiene la función de excitar las bobinas de los motores a pasos para posicionar los brazos en las coordenadas requeridas para el análisis solicitado. Este sistema se integra por los componentes mostrados en la figura 2-7:

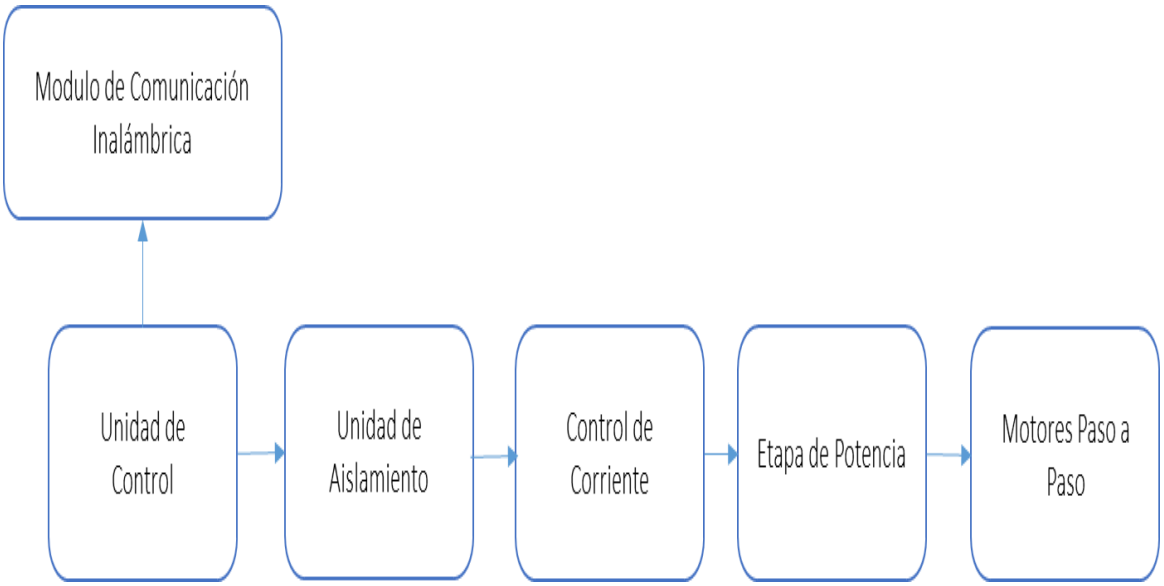


Figura 2-7. Diagrama de bloques del control electrónico para motores a paso.

2.2.1 Unidad de control

La función de esta unidad es manipular la electrónica para controlar los motores a paso qué moverán a los brazos mecánicos a la posición deseada.

La unidad de control está conformada y programada por la tarjeta "*arduino UNO",* véase figura 2-8) el componente más importante de todo el sistema. En esta tarjeta se llevan a cabo las operaciones lógicas y la señalización necesaria para que el sistema electrónico de motores a paso funcione adecuadamente.

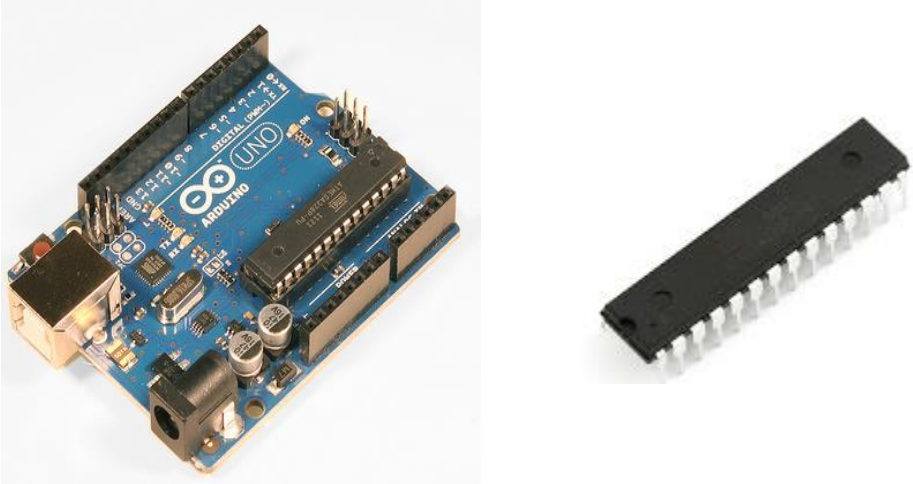


Figura 2-8. En la izquierda la tarjeta arduino UNO, a la derecha el microcontrolador AVR328p

La tarjeta arduino UNO es una plataforma electrónica abierta para la creación de prototipos basada en software y hardware fáciles de usar y de código abierto, con lo que permite crear proyectos electrónicos versátiles, trabajar con otros sistemas electromecánicos escalables y adicionalmente, tiene la capacidad de reconfigurarse.

Está compuesta por 14 entradas/salidas digitales, 6 de las cuales se pueden emplear como salidas de modulación por ancho de pulsos (PWM por sus siglas en inglés), 6 entradas analógicas, un oscilador de 16MHz, una conexión USB, un conector de alimentación, un conector ICSP y un pulsador para el reset.

Las características de este tipo de tarjeta son:

- Microcontrolador ATmega328p.
- Tensión de alimentación (recomendado) 7-12V.
- Integra regulador y estabilización de + 5V.
- 14 líneas de entradas/salidas Digitales (6 de estas se pueden utilizar para salidas PWM).
- 6 Entradas analógicas (que por software pueden ser salidas digitales).
- Máxima corriente continúa para las entradas: 40 mA.
- Salida de tensión de 3.3V y 50 mA.
- Memoria de programa de 32 KB (el bootloader pregrabado usa 0.5 KB).
- Memoria SRAM de 2Kb para las variables de trabajo.
- Memoria EEPROM de 1Kb para variables y datos no volátiles.
- Velocidad del reloj de trabajo de 16MHz.
- Reducidas dimensiones de 70 x 50 mm.

- El software de arduino se ejecuta en sistemas operativos Windows, Macintosh OSX y GNU/Linux.

Una de las diferencias más importantes de la tarjeta arduino UNO respecto a sus predecesoras y a otros microcontroladores, es la sustitución de un microcontrolador Atmega 8U2 programado como un convertidor o puente de USB en serie, por el convertidor USB-serie de la firma FTDI.

Para utilizar la tarjeta sólo es necesario conectarla a la computadora a través de un cable USB, o bien alimentarla con un adaptador de corriente AC/DC, o alimentar con una batería de 9V.

El algoritmo de control de posición está programado, en la IDE, para cubrir un rango de 360° dentro de una trayectoria circular, de modo que para mover cualquiera de los motores a pasos debe cumplir con una secuencia de activación.

La secuencia de activación para el caso de los motores descritos en 2.1 se muestra en la tabla 2-2:

Giro	Secuencia	φ_1	φ_2	φ_3	φ_4
Sentido normal del reloj	1	1	0	1	0
	2	1	0	0	1
Sentido contrario al reloj	3	0	1	0	1
	4	0	1	1	0

Tabla 2-2. Secuencia de pasos para mover el motor en “paso completo”.

donde $\varphi_1, \varphi_2, \varphi_3, \varphi_4$ representan cada uno de los devanados y los valores 1 indica 10 V y 0 indica 0V.

2.2.2 Unidad de aislamiento

Esta unidad está diseñada para proteger las etapas de potencia y la unidad de control del sistema de posicionamiento del tomógrafo. El diseño se basa en el circuito mostrado en la figura 2-9:

- Optoaisladores.

La combinación de un led con un fotodiodo forman un dispositivo conocido como optoaislador [3]. El led convierte la señal eléctrica aplicada al optoaislador en luz, la cual es detectada por el fotodiodo y convertida nuevamente a una señal eléctrica a la salida del optoaislador.

Figura 2-9. Unidad de control de corriente y aislamiento en el sistema de control electrónico.

Los optoaisladores sirven para proteger de altas corrientes a la etapa de control mediante el aislamiento eléctrico.

El uso de éste proporciona un aislamiento eléctrico completo entre el circuito eléctrico conectado a la entrada del aislador y el circuito conectado a su salida y es útil para reducir el efecto de interferencia eléctrica sobre la transmisión de *señales del tipo digital* [4].

En la figura 2-10 se muestra el circuito, del lado derecho (diagrama eléctrico), tiene un fotodiodo y un fototransistor acoplado [5], lo cual hace posible que se utilice al fototransistor como transistor común.

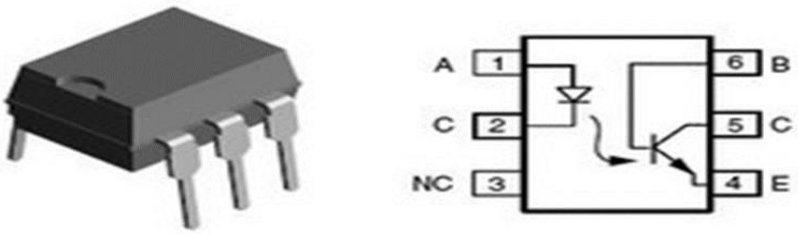


Figura 2-10. Circuito integrado de un optoaislador y su diagrama electrónico.

La corriente tanto el emisor (I_E) como en el colector (I_C) está definida por:

$$I_C = I_E = CTR(I_F) \qquad (2\text{-}2)$$

donde I_F es la corriente que pasa por el fotodiodo y CTR representa una relación de transferencia de corriente del fotodiodo hacia la base del fototransistor, típicamente es una relación entre 10% - 100%.

2.2.3 Unidad de potencia

La unidad de potencia es responsable de alimentar a los motores en cada uno de sus devanados llevando consigo la energía necesaria para moverlos y ejecutar el movimiento deseado. Se compone de dos partes:

1. Control de corriente.
2. Etapa de potencia.

La unidad de potencia está formada por dos circuitos integrados, uno de ellos el CI L6506 y el otro es el CI L298 (ver figura 2-11).

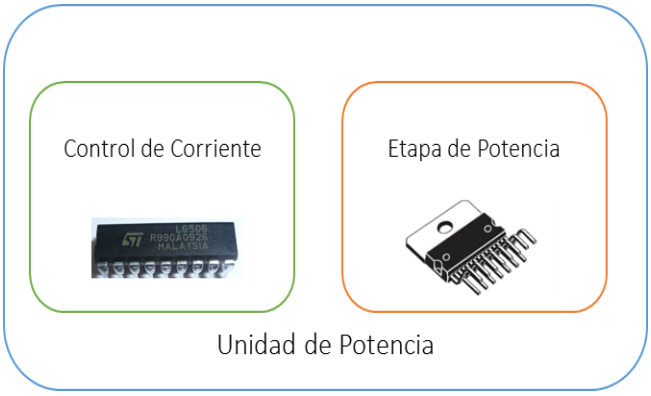


Figura 2-11. Etapa de potencia implementado en el diagrama de bloques del sistema de control electrónico.

El CI L6506 tiene como función regular la corriente que será suministrada en las bobinas del motor a pasos. Un esquema del circuito integrado puede observarse en la figura 2-12:

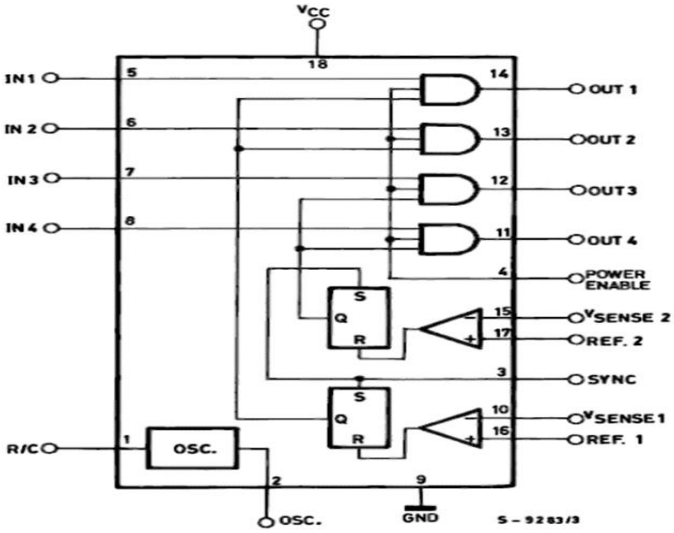


Figura 2-12. CI L6506, a la izquierda se puede ver el chip físicamente y a la derecha su composición electrónica.

El circuito integrado L6506 se compone principalmente por:

- Un oscilador interno.
- Circuitos flip-flops.
- Comparadores de voltaje.

La etapa de potencia utiliza el circuito integrado L298 que tiene las siguientes características:

- Voltaje de operación hasta 46V.
- Corriente total de DC soportado en el chip es de 4A.
- Compatible con lógica TTL (5V), un "0" lógico lo reconoce con un voltaje hasta 1.5V.

El CI L298 es dual, soporta alta corriente y está diseñado para aceptar niveles lógicos TTL estándar y unidades de carga inductivas como son relés, solenoides, y motores paso a paso, el circuito y sus componentes se muestran en la figura 2-13.

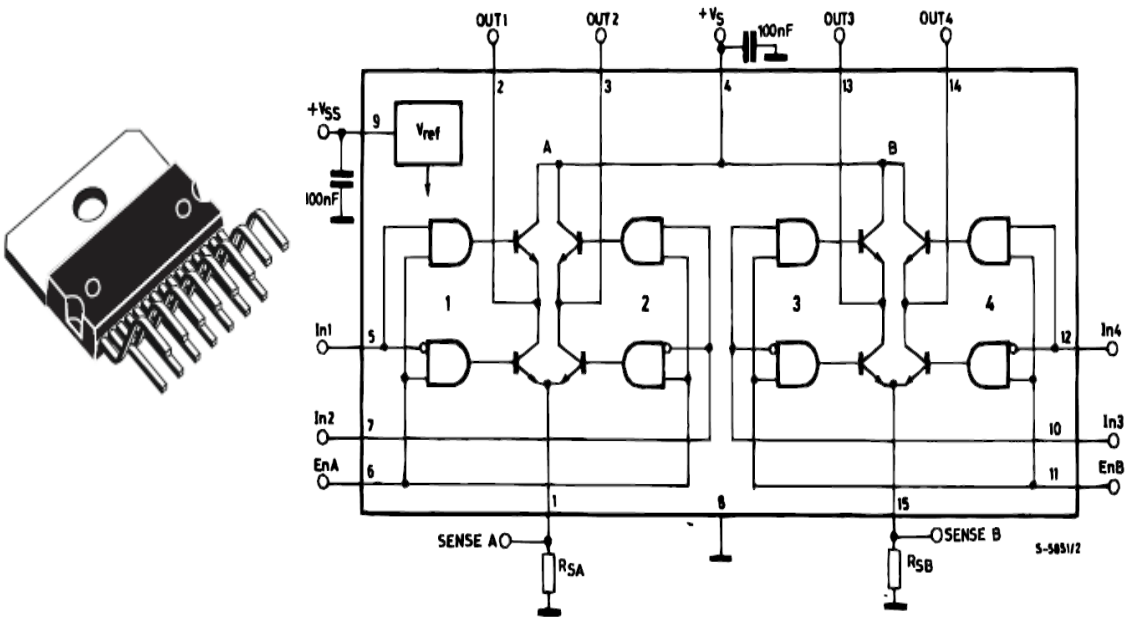


Figura 2-13. A la izquierda se observa el chip físicamente, a la derecha se muestra sus componentes.

Las entradas lógicas del chip L298 (In1, In2, In3 y In4) son conectadas a las salidas del L6506 y, las cuatro etapas de potencia de salida del CI L298 (OUT1, OUT2, OUT3, OUT4) están conectadas a las bobinas de los motores a pasos.

Adicionalmente, el integrado L298 tiene la función de detectar si la corriente que se consume en el motor de pasos es mayor que cierto umbral; para ello tiene dos pines (SENSE A y SENSE B) cuyo voltaje será comparado con un divisor de tensión del chip L6506.

Para poder sensar y controlar la corriente en el motor se conectan al circuito integrado L6506 unas resistencias de sensado R_{SA} y R_{SB}, típicamente de 1Ω, con el propósito de administrar la potencia de todo el circuito de posicionamiento.

2.2.4 Comunicación del sistema electrónico con la interfaz gráfica

La comunicación entre la interfaz y el control electrónico está configurada para trabajar de forma inalámbrica. El hardware que se usa para este propósito corresponde a un módulo electrónico de bluetooth HC-06 (ver figura 2-14).

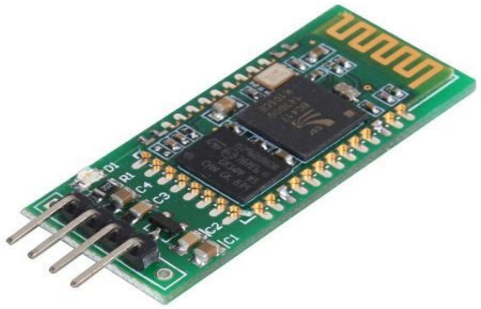


Figura 2-14. Módulo de comunicación por bluetooth, HC-06.

Este módulo usa dos pines para mover información de la interfaz de usuario al micro controlador y viceversa. El esquema se muestra en la figura 2-15:

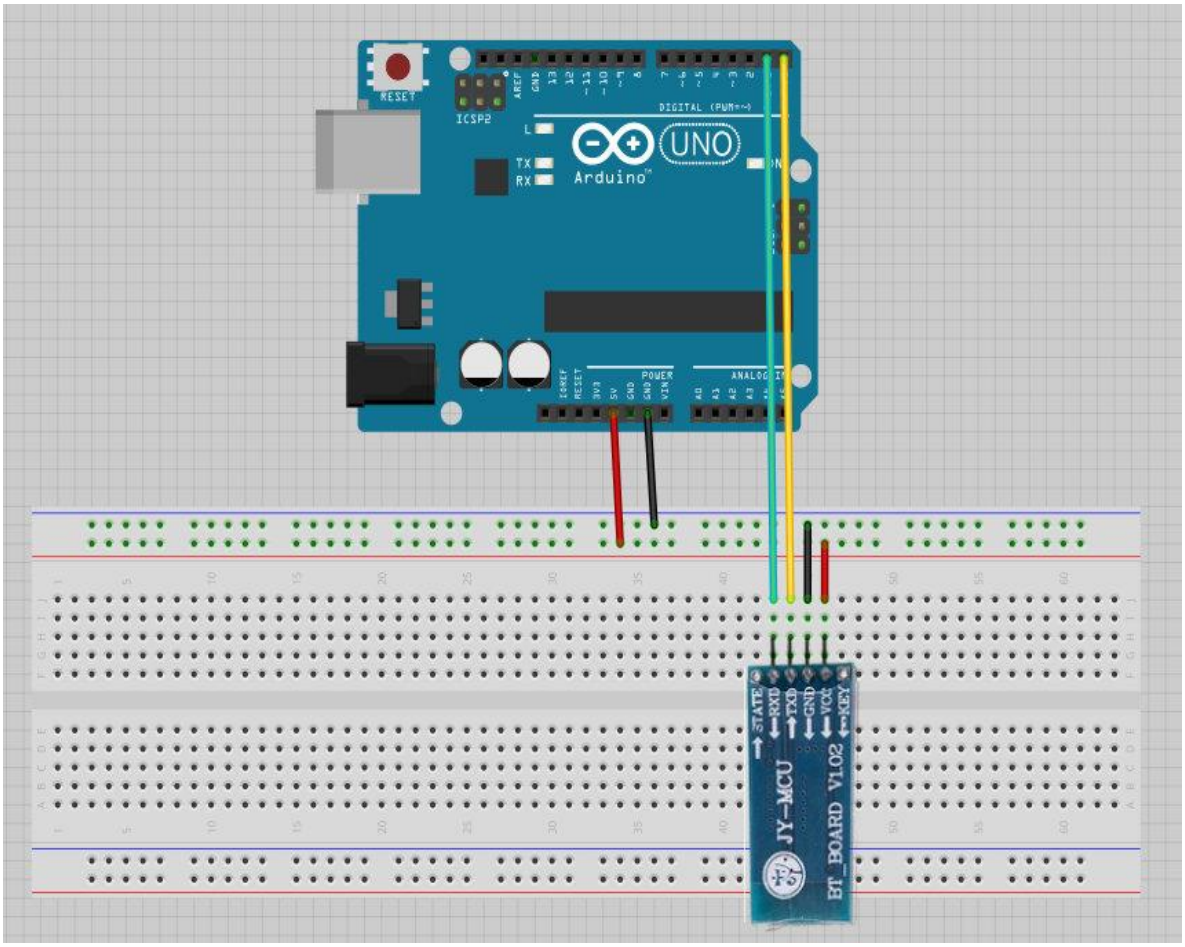


Figura 2-15. Esquema de conexión del módulo HC-06 al micro controlador.

En la tabla 2-3 se muestran las conexiones del módulo bluetooth:

Pines del Módulo HC-06	Pines del arduino UNO R3
Vcc	Arduino +5V (preferentemente a 3.3V pin)
Gnd	Arduino GND
TXD	Arduino RX (pin recepción serial)
RXD	Arduino TX (pin de transmisión serial)

Tabla 2-3. Conexiones del módulo HC-06 al micro controlador de la placa arduino UNO.

Gracias a este módulo de comunicación inalámbrica, la interfaz gráfica transfiere la información necesaria para ejecutar el proceso de tomografía elegida por el usuario.

Para que el módulo opere se requiere que sea configurado mediante el software propietario o en arduino.

2.3 Sistema de emisión y recepción de datos

Una parte importante en el desarrollo del tomógrafo ultrasónico es el sistema de emisión y recepción de la información que proviene de la onda de ultrasonido, este sistema está integrado por:

- Pulser ultrasónicos.
- Recepción de ultrasonido.
- Etapa de amplificación.
- Transductores ultrasónicos.

El diagrama de bloques de la figura 2-16 representa a este sistema de emisión y recepción del ultrasonido.

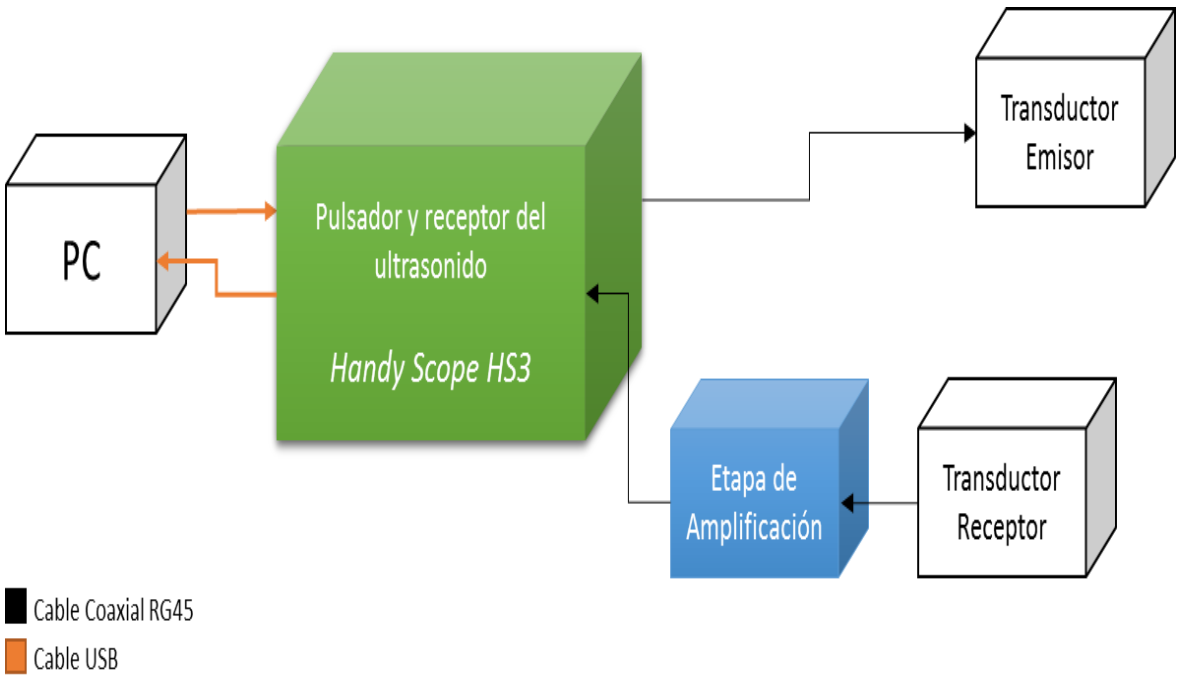


Figura 2-16 Diagrama de bloques del sistema de emisión y recepción del ultrasonido.

2.3.1 Pulser ultrasónico

Las tareas de generación de los pulsos ultrasónicos para energizar a los transductores y la adquisición de la energía transmitida en el medio de estudio son realizadas por un dispositivo digital programable llamado HandyScope HS3® (ver figura 2-17).

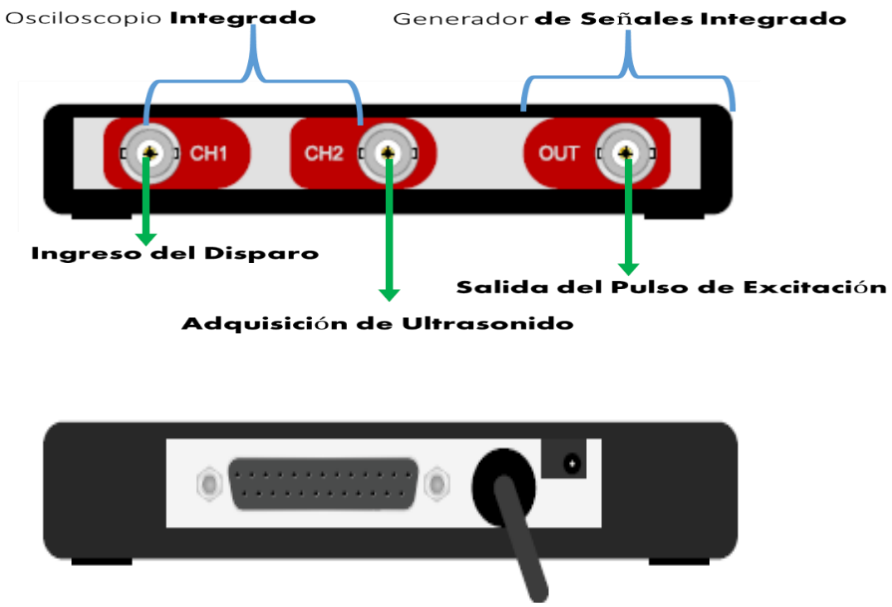


Figura 2-17. Partes que integran la unidad de recepción y emisión del ultrasonido y una vista trasera de este dispositivo.

2.3.1.1 Emisión del ultrasonido

La generación de los pulsos de excitación de los transductores se realiza mediante un generador de funciones contenido en el HandyScope HS3®. Una característica importante de este generador de funciones es la capacidad para implementar señales arbitrarias, ya sean almacenadas en *un archivo o bien utilizando algún software externo*.

La implementación de los pulsos de excitación y la frecuencia asociada a éstos se pueden realizar utilizando dos métodos: *método lineal y el de síntesis digital directa (DDS)*[6].

Los dispositivos que usan DDS son utilizados para generar formas de onda arbitrarias a frecuencias variadas, tomando como base una señal de frecuencia fija.

Las excitación de los transductores se puede realizar generando ondas en modo continuo o empleando pulsos definidos utilizando una opción del Handy Scope® llamada *disparo único* (*single shot*).

El modo continuo consiste en generar ondas continuas a partir de tener grabado en la memoria del Handy Scope un solo ciclo de la señal a generar. El modo de disparo

único consiste en generar la forma de onda almacenada en la memoria del HS3 una sola vez. Observe la figura 2-18 para ver como se genera una señal continua y un pulso con el HS3.

En la figura 2-18, se muestra una señal arbitraria generada en modo continuo y en disparo único, aquí la frecuencia de salida para una señal arbitraria puede quedar definida a partir de la duración de un ciclo de la señal a generar o de N ciclos de señal que existan en la memoria del HS3.

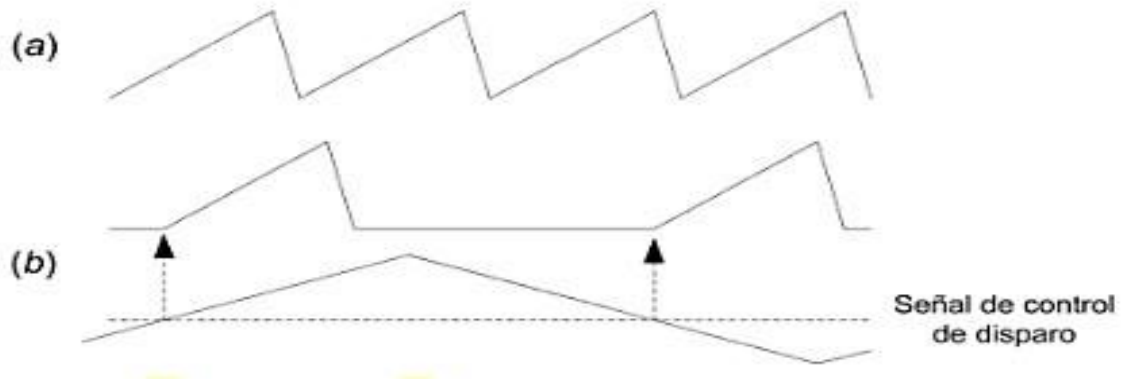


Figura 2-18. DDS en a) modo continuo, b) en modo de disparo único “single shot”.

Para el caso del modo disparo único se requiere determinar la duración del disparo t_d, dada por:

$$t_d = \frac{L}{f_r} \quad (2\text{-}5)$$

donde L es la longitud de la memoria (cantidad de muestras en memoria) y f_r es la frecuencia de reloj de muestreo.

2.3.2 Recepción del ultrasonido

La recepción del ultrasonido depende de que se produzca un evento generado por una señal de disparo que se reciba en el canal 1 del Handy Scope ®.

Cuando este evento de disparo se produzca en el canal uno, la onda de ultrasonido sera capturada por transductor el receptor, conectado al canal dos del osciloscopio digital configurado en el Handy Scope HS3, será transferida de la memoria hacia el software del computador por USB.

El tiempo de adquisición del ultrasonido $t_{captura}$ está dado por:

$$t_{captura} = (N_{muestras})\left(\frac{1}{f_{muestreo}}\right) = (N_{muestras})(T_{muestreo}) \qquad (2\text{-}6)$$

donde:

$N_{muestras}$ → número de muestras
$f_{muestreo}$ → frecuencia de muestreo

Cuando la señal eléctrica del ultrasonido es capturada, se almacena en la memoria del HS3 a distintas "resoluciones verticales", es decir 8 bits, 12 bits y 14 bits.

Las conexiones en el HandyScope HS3® para la emisión y recepción de los pulsos ultrasonido se muestran en la tabla 2-4.

Handy Scope HS3	*Función*	*Amplificador*	*Función*
OUT	Salida de pulsos de alimentación	IN	Amplificación
CH2	Adquisición del ultrasonido	OUT – CH2	-
CH1	Adquisición de la señal de disparo	-	-

Tabla 2-4. Conexión del osciloscopio de mano HS3 al amplificador Olympus.

La recepción de las ondas de ultrasonido y la señal de disparo también están embebidas en el HS3, como se muestra en la figura 2-19.

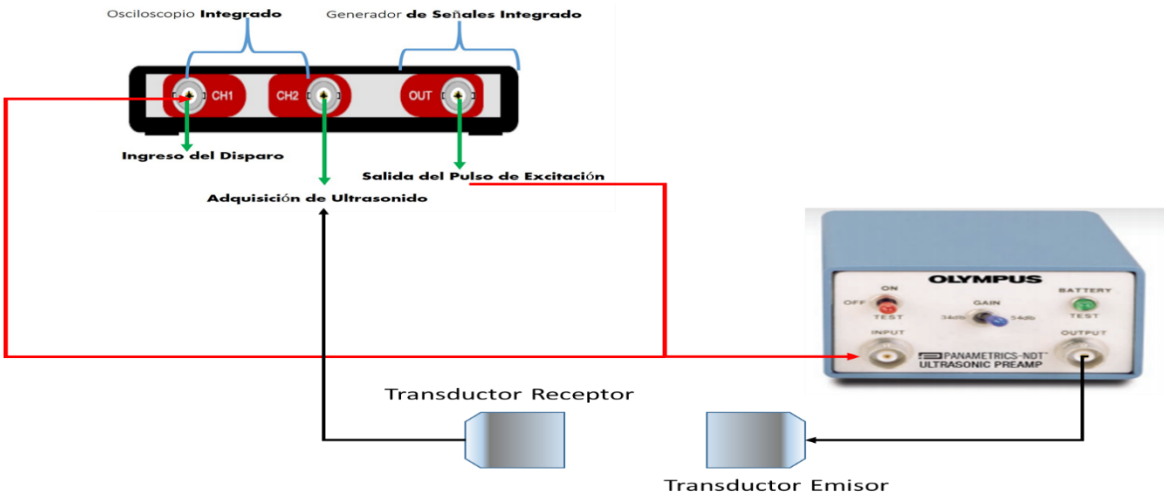


Figura 2-19. Conexión entre el Handy Scope HS3 y el amplificador con transductores ultrasónicos.

2.3.3 Etapa de amplificación

El objetivo principal del amplificador de señales es otorgarle a la señal ultrasónica proveniente del transductor receptor una ganancia, es decir, incrementar su amplitud para que después las señales de ultrasonido pueden ser procesadas por medio de algún algoritmo de procesamiento de señales.

Este preamplificador fue requerido para su instrumentación, debido a que cuando se estudia o se ensaya con materiales heterogéneos o que presenten fallas, estos materiales presentan características atenuantes y por lo tanto es necesario amplificarlas para su posterior estudio. El preamplificador presenta 2 niveles de ganancia constante: una de ellas es 34dB y la otra es 54dB, como se puede observar en la figura 2-20.

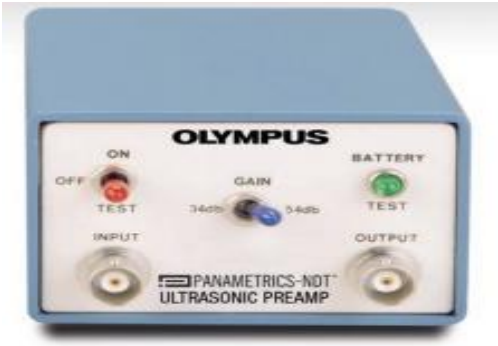


Figura 2-20. Preamplificador instrumentado, con dos opciones de ganancia ajustable.

La etapa de amplificación está definida por la relación

$$\alpha = \left(\frac{V_{salida}}{V_{entrada}}\right) \quad (2\text{-}7)$$

donde:

$\alpha \rightarrow$ ganancia de amplificación.
$V_{salida} \rightarrow$ amplitud en volts de la señal recibida en el transductor receptor.
$V_{entrada} \rightarrow$ amplitud en volts de la señal emitida en el transductor emisor.

2.3.4 Transductores ultrasónicos

Los transductores de ultrasonido tienen en su interior un elemento piezoeléctrico que tiene como función la conversión de las señales del tipo eléctrico a vibraciones mecánicas y de vibraciones mecánicas a señales eléctricas.

Se pueden clasificar de acuerdo a la aplicación con la que se quiera trabajar y son:

- Transductores de contacto.
- Transductores de inmersión.

Los transductores de contacto se usan en inspecciones donde el transductor se encuentra en contacto con la superficie del material a inspeccionar y utilizan un acoplante para que no haya pérdida de la energía entre el material y la cara de los transductores.

Los transductores de inmersión son aquellos elementos que se utilizan inmersos en un acoplante, generalmente agua, cuyas conexiones son herméticas.

2.4 Software de usuario, la interfaz.

El sistema electrónico cuenta con hardware de comunicación electrónica vía "bluetooth". Este tipo de comunicación se realiza entre la PC y la unidad de control del sistema electrónico. Por medio de esta vía de comunicación, el sistema de control electrónico es configurado por el usuario a través de la interfaz gráfica.

El sistema de tomografía tiene una interfaz gráfica que permite controlar, comunicar, configurar a todos los dispositivos integrados en el tomógrafo, escoger las formas de inspeccionar al objeto y guardar las señales de ultrasonido. Adicionalmente contiene un bloque que está disponible para visualizar la imagen reconstruida del medio en estudio.
La interfaz tiene las siguientes características:

1. Permite al usuario configurar los parámetros de escaneo, es decir, se puede ingresar el ángulo de paso, la velocidad de cambio de paso de los motores y transferir estos datos inalámbricamente al control electrónico de posicionamiento.

2. Permite configurar el sistema de recepción y emisión ultrasónica, lo cual implica que puede cambiar la longitud de la señal que se recibe, permite cambiar la frecuencia de trabajo o el ancho del pulso de excitación, es decir significa que se puede cambiar el transductor con que se quiera trabajar.

3. Permite probar de manera independiente todas las partes que integran el sistema de tomografía.

La figura 2-21, muestra la pantalla principal de la interfaz gráfica de usuario.

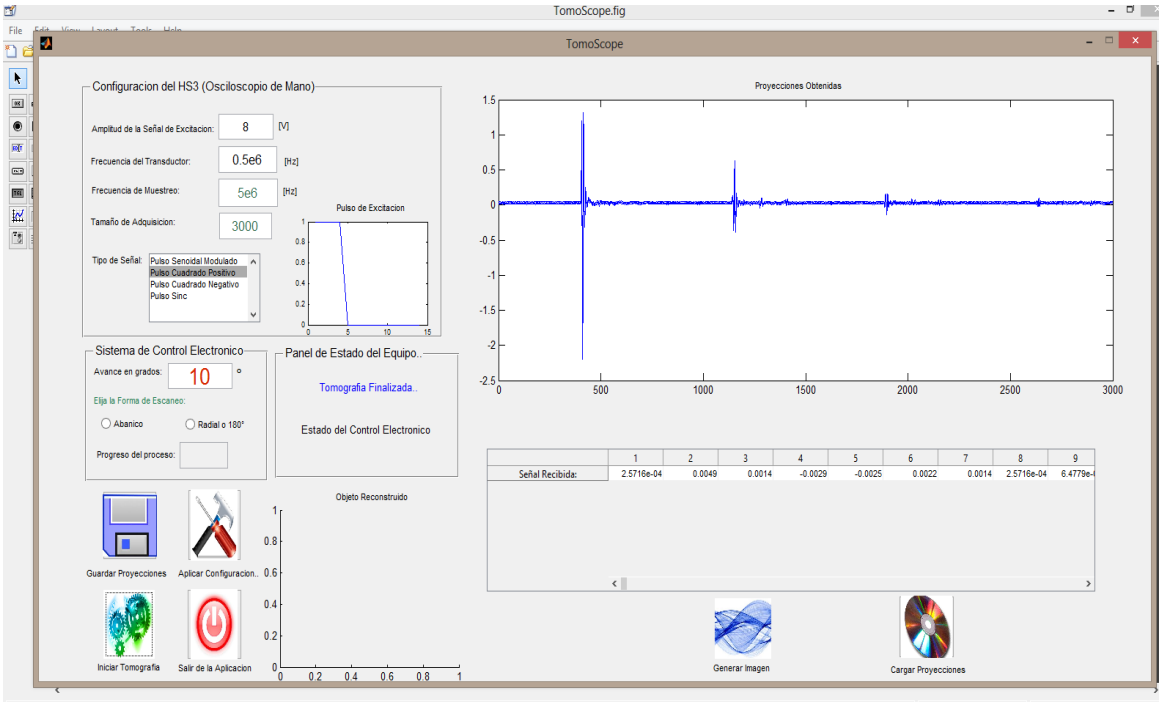


Figura 2-21. Interfaz gráfica principal del tomógrafo.

2.4.1 Configuración de la emisión y recepción del ultrasonido

El software tiene módulos con la función de configurar de manera separada todos los componentes involucrados en el sistema tomográfico. La finalidad de estos módulos es verificar que las operaciones del tomógrafo se efectúen adecuadamente.

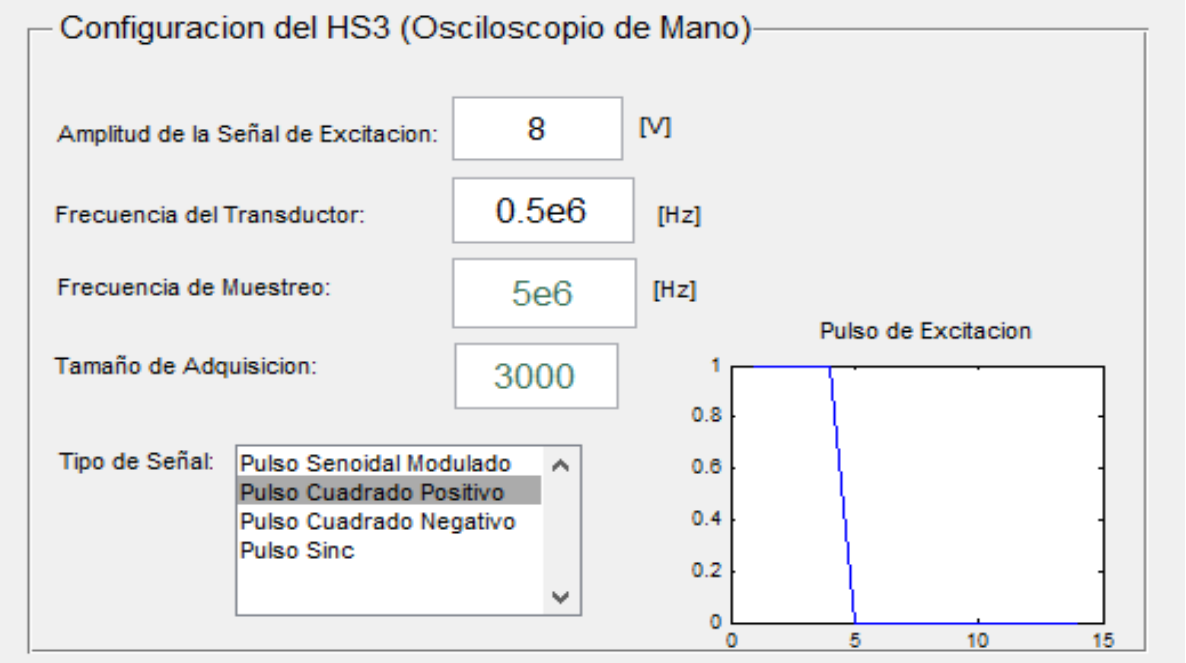


Figura 2-22. Módulo de configuración de la emisión y adquisición del ultrasonido.

La figura 2-22 muestra el módulo de configuración de las características del sistema de tomografía, *referentes a la emisión y recepción del ultrasonido*. Los parámetros configurables son:

- Amplitud del pulso de emisión.
- Frecuencia de trabajo del transductor.
- Frecuencia de muestreo, tanto para el pulso de emisión, como la velocidad de adquisición del ultrasonido.
- Número de muestras a guardar en la memoria de adquisición del ultrasonido.
- Tipo de pulso a emitir.

2.4.2 Configuración del tipo de inspección por medio del control electrónico.

Una sección importante del software de usuario es la sección del control electrónico de configuración de motores, véase figura 2-23. Con esta sección se establece el tipo de escaneo a realizar y el avance en grados de cada transductor de ultrasonido;

en secciones anteriores, se explicó que la formas de escanear al objetivo son dos: escaneo en abanico y escaneo en forma radio o 180°.

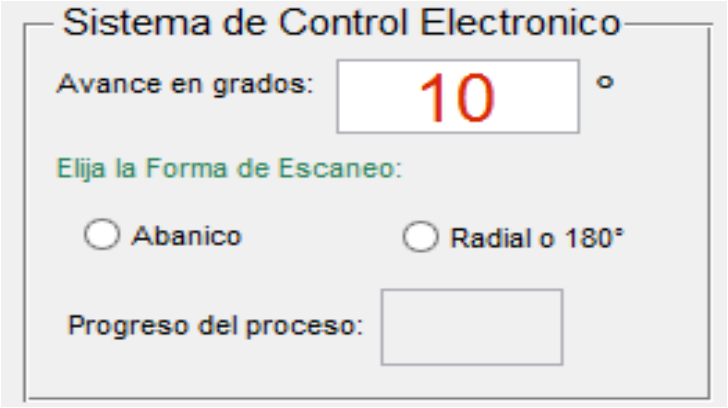


Figura 2-23. Sección de control electrónico de configuración de los motores.

El software de usuario, también permite graficar y colocar los datos de las señales adquiridas en forma de tabla o panel y acomodarlas en forma de arreglo, para después procesarlas y guardarlas si el usuario final así lo desea, ver figura 2-24.

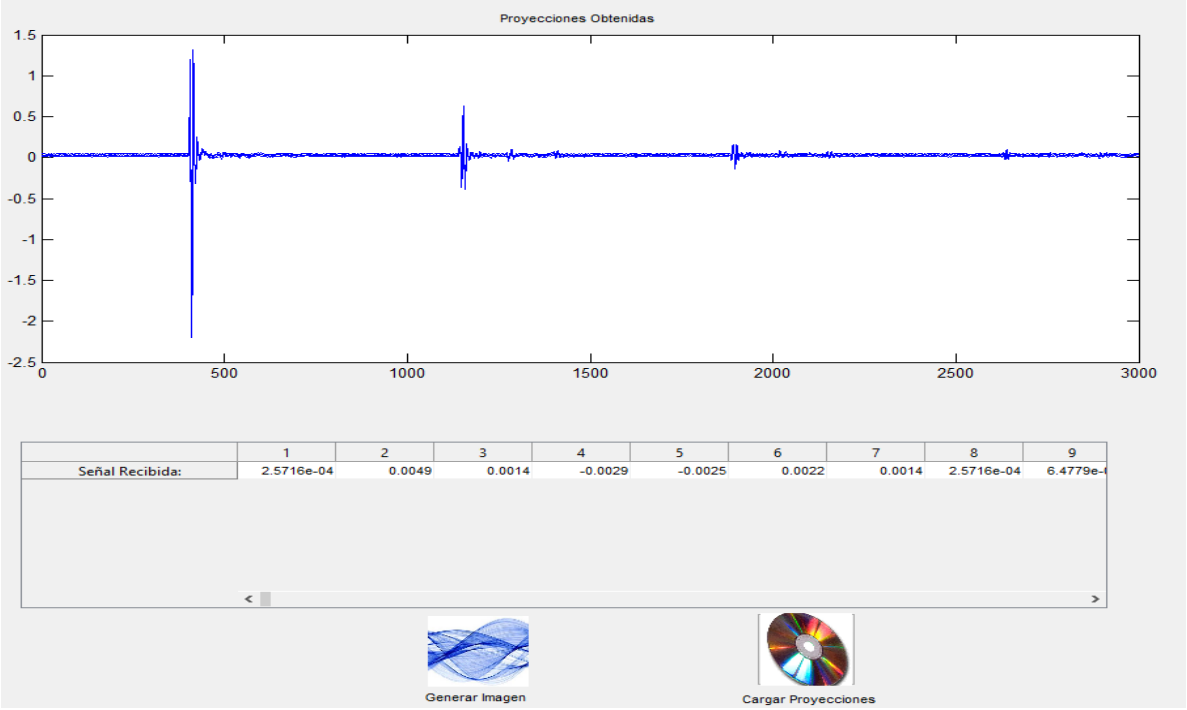


Figura 2-24 Modulo de graficas de señales.

En la figura 2-25 se muestra un esquema de cómo escanear al material en estudio de forma en abanico y el procedimiento seguido por la interfaz para escanear con esta manera:

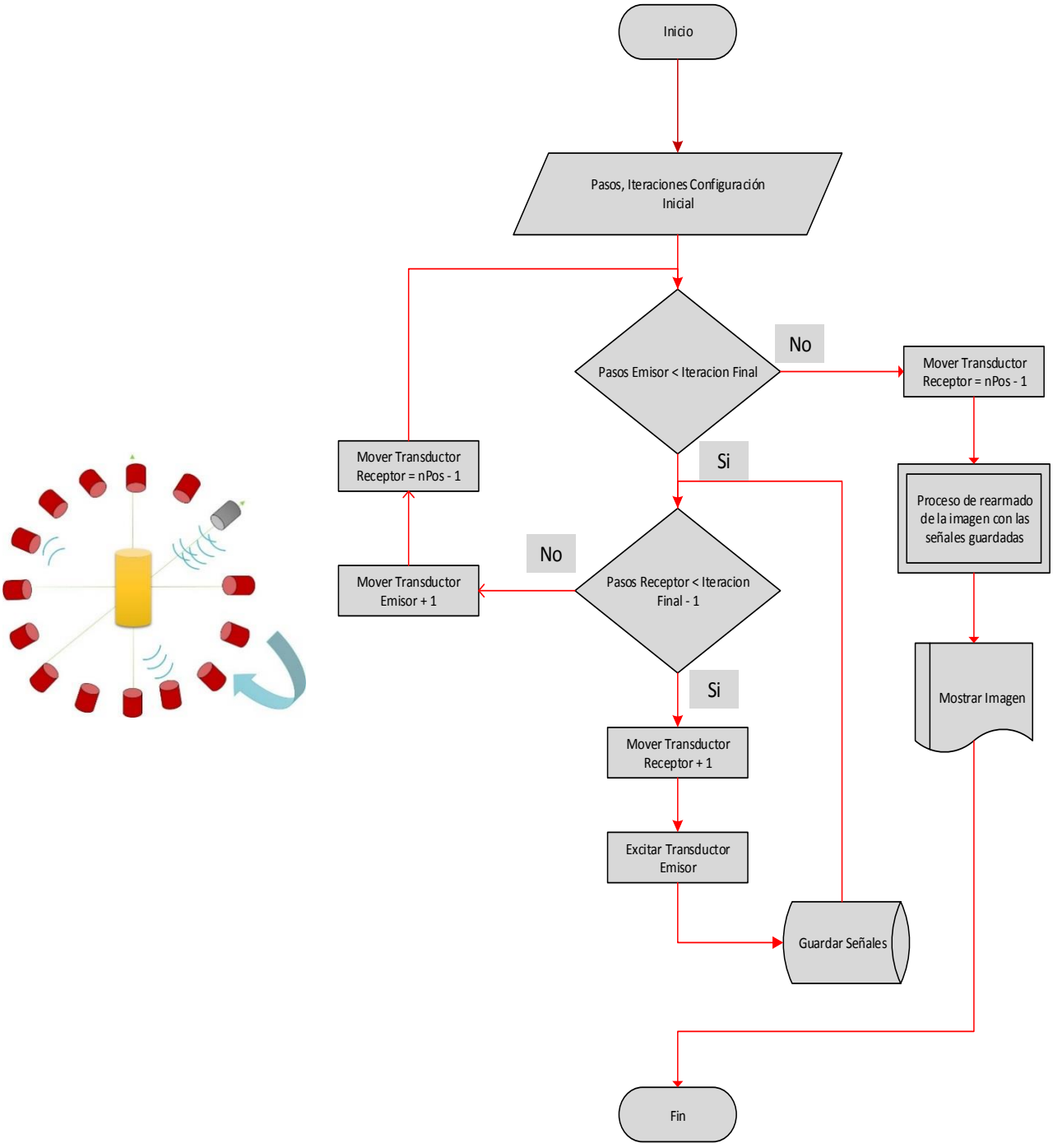


Figura 2-25. Escaneo en abanico y su diagrama del flujo, muestra el procedimiento seguido en la interfaz.

En el diagrama anterior, se genera un mapa de señales de ultrasonido que permitirían obtener una visualización del objeto en computadora, la cantidad de señales o proyecciones que se obtienen del objeto con esta forma de escaneo corresponde a:

$$p(x, \theta_{mn}) = m\ (n-1) \quad (2\text{-}8)$$

En la figura 2-26, se muestra un esquema de cómo escanear al objeto de forma radial o 180° y el procedimiento seguido por la interfaz para escanear por esta otra manera:

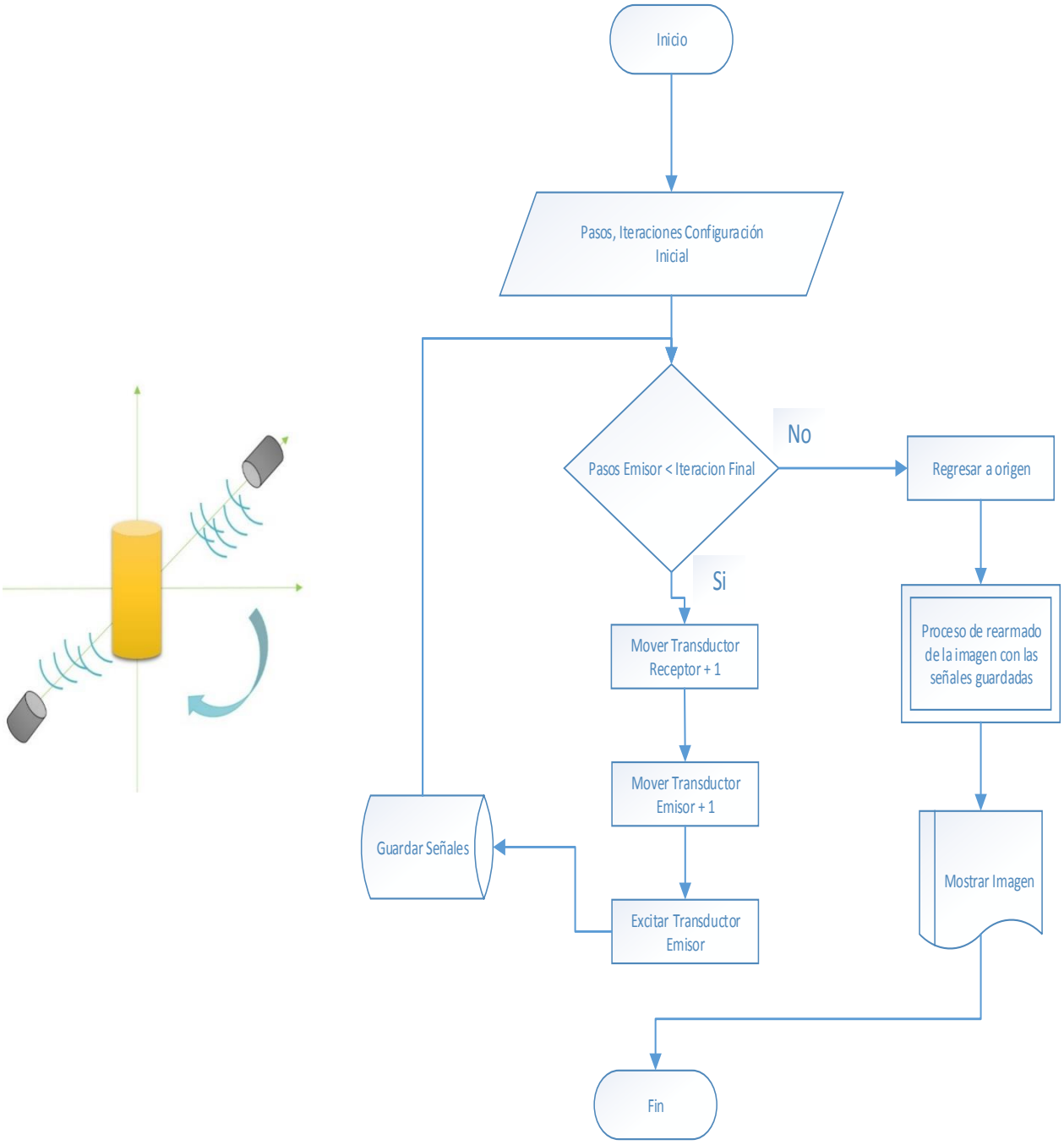


Figura 2-26. Escaneo en 180° o radial y su diagrama de flujo, se muestra otro procedimiento seguido por la interfaz gráfica.

En este último diagrama de flujo, se generan menos señales que en el primer procedimiento descrito anteriormente. La cantidad de señales generadas se determina por:

$$p(x,\theta_n) = n \quad (2\text{-}9)$$

Para cada tipo de configuración de escaneo del objeto que se escoja (ya sea radial o abanico) el parámetro *n* que representa la cantidad de proyecciones a obtener por el tomógrafo, resulta de dividir los 360° entre el número de grados de avance del motor a pasos.

Capítulo 3. Implementación de un Sistema de Emisión y Adquisición de Señales Ultrasónicas de Dos Elementos Transductores

El objetivo de este capítulo es mostrar cómo se implementaron los diferentes componentes de cada sistema.

3.1 Diseño del circuito electrónico de control de motores a pasos

El sistema de control electrónico, véase figura 3-1, de los motores a paso del sistema de adquisición de señales ultrasónicas, se integra por cinco módulos: comunicación vía inalámbrica, unidad de control, unidad de aislamiento, unidad de potencia, y los motores a pasos del tomógrafo.

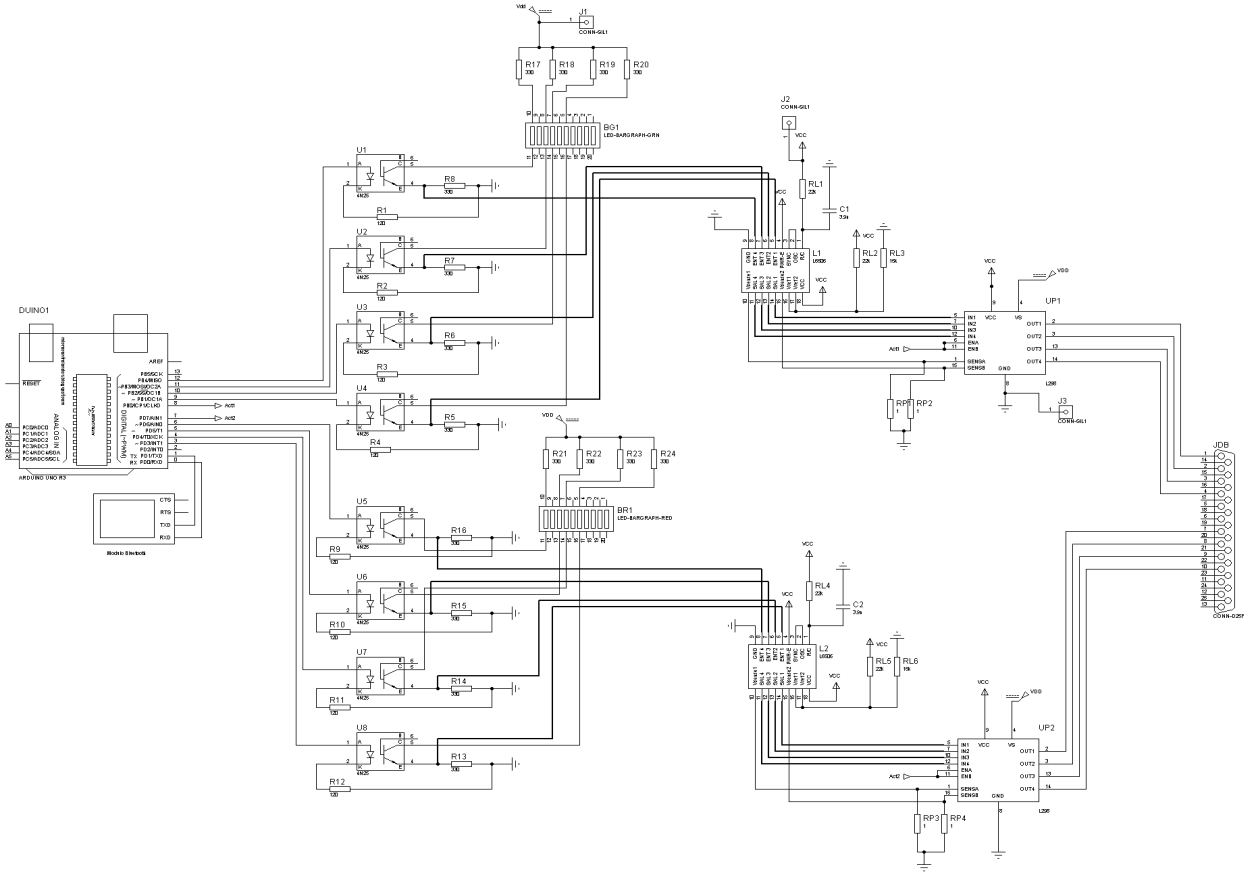


Figura 3-1. Esquema electrónico de control de los motores a paso.

3.2 Implementación de la unidad de aislamiento

Para proteger la etapa de control es necesario diseñar una unidad de aislamiento. Esta etapa es implementada de base se circuitos opto-acopladores genéricos, y su función principal es aislar la etapa digital de la etapa analógica (figura 3-2).

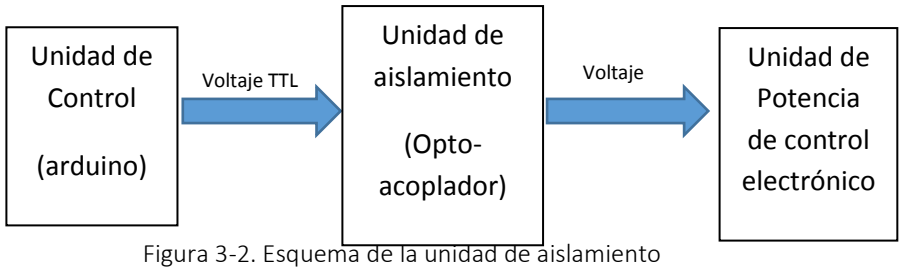


Figura 3-2. Esquema de la unidad de aislamiento

El optoacoplador o también conocido como optoaislador o dispositivo de acoplamiento óptico, son componentes electrónicos en los cuales para su funcionamiento emplean un haz de radiación luminosa para comunicar señales de un circuito a otro sin conexión eléctrica. El diseño de la unidad de aislamiento fue realizado tomando en cuenta circuito mostrado en la figura 3-3.

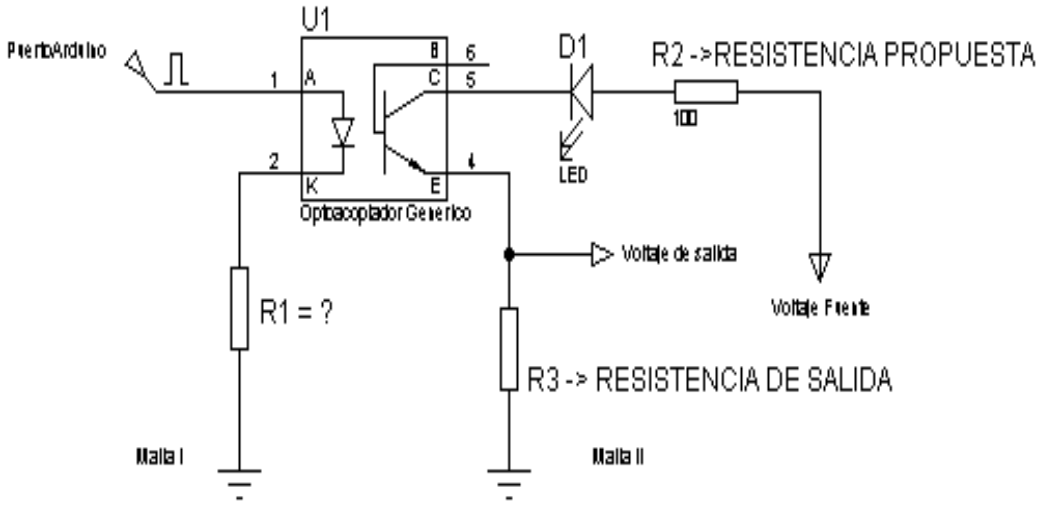


Figura 3-3. Diagrama genérico del opto acoplador para uso digital.

El optocaplador o elemento U1 mostrado en el circuito debe de recibir los voltajes lógicos o TTL de la unidad de control que indicaran a la unidad de potencia cuando realizar un giro en el motor de pasos. Para el diseño del circuito se tiene que tomar en cuenta las amplitudes o niveles de corriente y voltaje que provienen de la unidad

de control. Las señales provenientes de la tarjeta arduino, son señales digitales (niveles TTL) que indican cuando la unidad de potencia debe de proveer el voltaje suficiente a los motores de paso para que realicen el movimiento. El optoacoplador permite que estas señales sean trasmitidas sin establecer una comunicación física entre el control y la potencia, con lo que se protege y aisla la etapa de control. Para el diseño de la unidad de aislamiento se debe considerar los siguientes parámetros eléctricos:

1) Voltaje de los puertos digitales del arduino = 5V
2) Voltaje de la fuente de corriente directa = 10V
3) Voltaje de salida requerido = 5V
4) Corriente máxima de los puertos digitales del arduino = 30mA
5) Resistencia propuesta = 100Ω
6) CTR = 50% (propuesto) o menos.

Con los datos anteriores se realizaron los cálculos necesarios para que en el emisor del fototransistor se obtengan 5V y exista compatibilidad en los niveles de voltaje requeridos por la electrónica utilizada.

Los cálculos se llevaron a cabo empleando la ecuación 2-2 y la ley de voltaje de Kirchhoff aplicada en la malla I y la malla II. Los detalles de los cálculos se muestran a continuación.

Para la malla I, se tiene que:

$$V_{puerto\_arduino} - V_{fotodiodo} - V_{R1} = V_{puerto_{arduino}} - V_{fotodiodo} - I_F R_1 = 0$$

$$R_1 = \frac{V_{puerto_{arduino}} - V_{fotodiodo}}{I_F} = \frac{5V - 1.3V}{30mA} = 123.33\Omega$$

Con la corriente $I_F = 30mA$, se conoce la corriente en el emisor del fototransistor I_E, se obtiene que la esta corriente es:

$$I_C = I_E = CTR\,(I_F) = \frac{50\%}{100\%}(30mA)$$

$$I_E = 15mA$$

Para la malla II, si $I_E = 15mA$ y $V_{salida} = 5V$ que se necesita, se obtiene lo siguiente:

- En el emisor del fototransistor:

$$V_E = V_{salida} = I_E R_3$$

$$R_3 = \frac{V_{salida}}{I_E} = \frac{5V}{15mA} = 333.33\Omega$$

- En el colector del fototransistor, se hace LVK:

$$-V_c - V_{led1} - V_{R2} + V_{fuente} = 0$$

$$V_c = -V_{led1} - V_{R2} + V_{fuente} = -V_{led1} - I_E R_2 + V_{fuente}$$

$$V_c = -1.8V - 15mA(100\Omega) + 10V = 6.7V$$

- El voltaje de colector emisor del fototransistor es:

$$V_{CE} = V_c - V_E = 6.7V - 5V = 1.7V$$

Los valores comerciales de las resistencias R_3 y R_1 qué cumplen con las características del sistema son:

$$R_1 = 120\Omega$$

$$R_3 = 130\Omega$$

Con los cálculos elaborados, se realiza la simulación del circuito utilizando el software de simulación electrónica Multisim 13®. El circuito integrado del optoaislador que puede trabajar con una CTR del 50% y un voltaje de fotodiodo a 1.3V es *el integrado 4N25*.

3.2.1 Simulación de la unidad de aislamiento

Con los cálculos elaborados en la sección 3.1, se realizó el diagrama electrónico para la simulación en Mulisim 13® (ver figura 3-4).

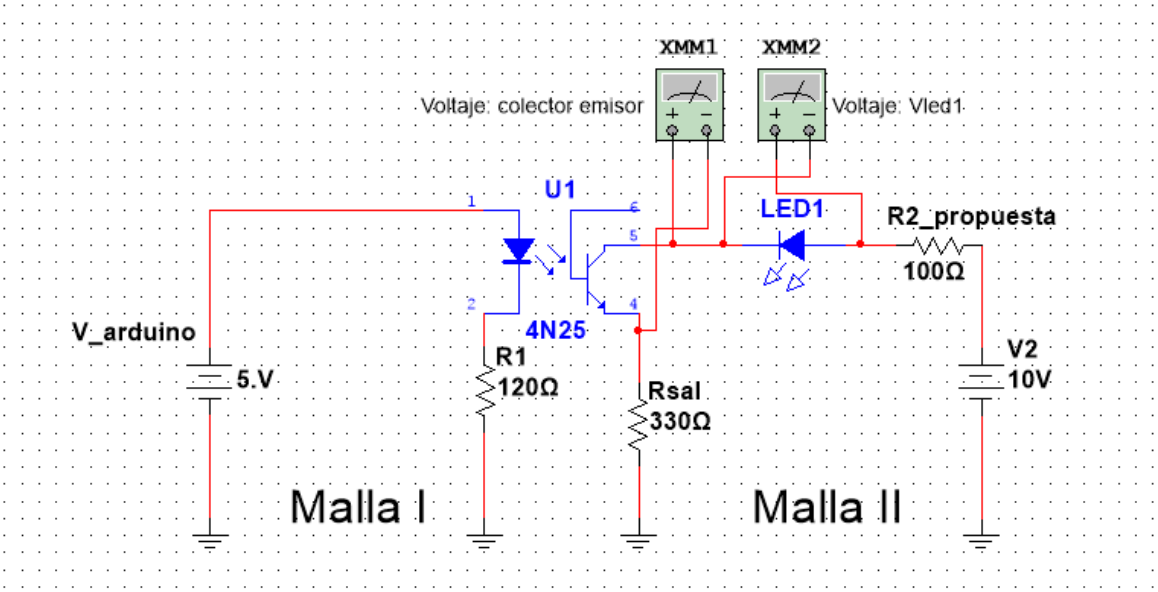


Figura 3-4. Diagrama del circuito optoaislador a simular.

Los resultados de la simulación se muestran en la figura 3-5:

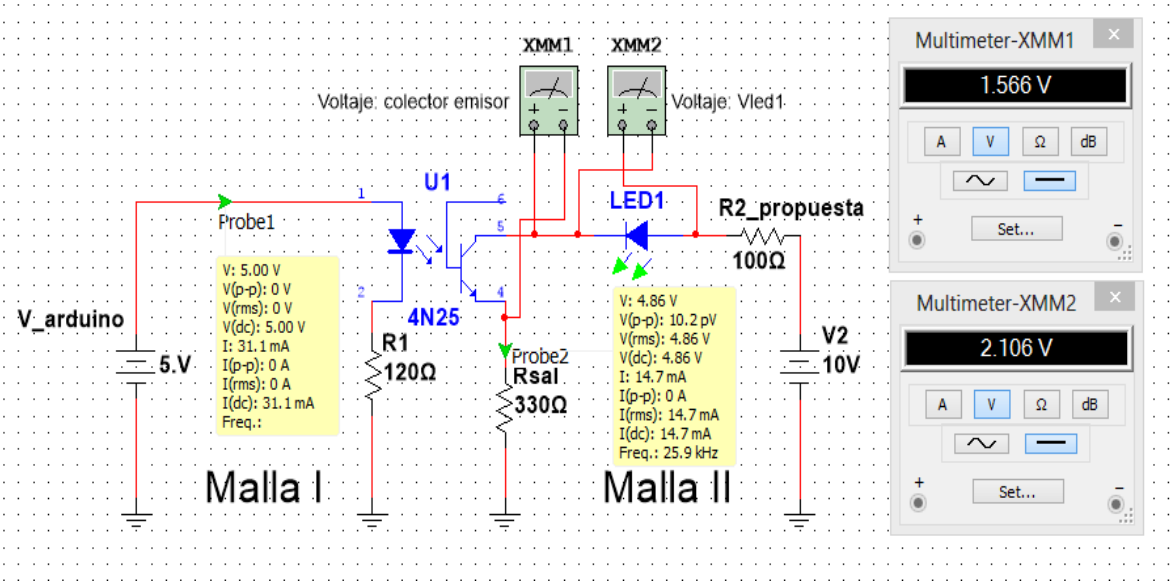


Figura 3-5. Resultados de la simulación del circuito optoaislador.

Los recuadros en color amarillo representan en la simulación “sondas” que recogen información de la corriente y el voltaje en las redes del circuito a simular.

La “sonda 1” muestra que el voltaje del puerto del arduino es de 5V con una corriente de 31.01 mA, mientras que en la “sonda 2” el voltaje de salida es de 4.86 V con una corriente en el emisor del fototransistor de 14.7 mA.

Los recuadros con el nombre de “multímetro XMM1” y “multímetro XMM2” miden el voltaje de colector-emisor del fototransistor y el voltaje del diodo led número uno, respectivamente.

Para la implementación de este circuito fue necesario armarlo ocho veces, *ya que cada motor ocupa 4 pines digitales* y estos pines digitales se tienen que aislar de la etapa de potencia. El optoaislador se combina con un circuito integrado que limita la corriente a usar por los motores a pasos, el chip *limitador de corriente es el L6506, cada salida del optoaislador se conecta a las entradas (IN1, IN2, IN3, IN4) del chip L6506*.

3.3 Diseño e implementación de la unidad de potencia

La unidad de potencia está desarrollada con los circuitos integrados L298 (unidad de potencia) y L6506 (control de corriente).

El circuito L6506 se encarga de controlador la corriente en cada uno de las bobinas de carga en los motores a paso que mueven a los brazos mecánicos del sistema de tomografía. En la figura 3-6 se observa la composición interna de este circuito integrado.

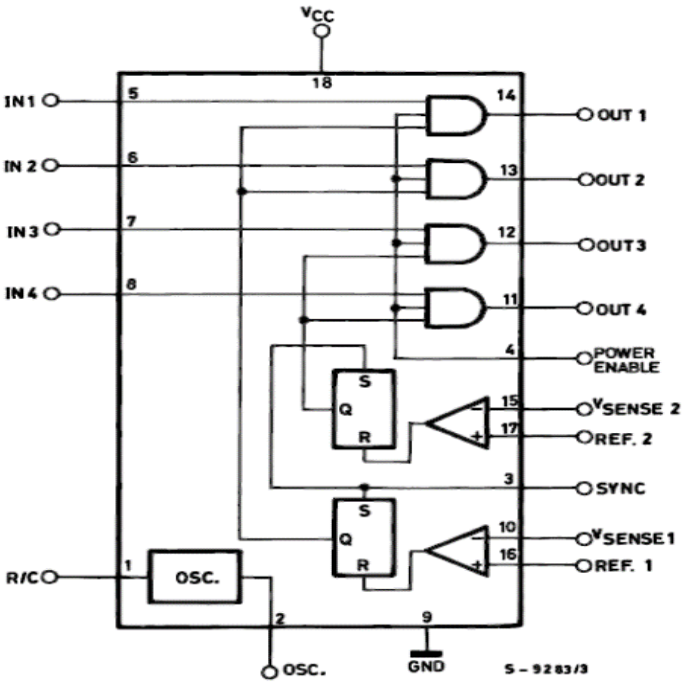


Figura 3-6. Componentes internos del integrado L6506.

El CI L6506 tiene un oscilador interno que acciona un interruptor doble y establece la frecuencia de operación de una onda modulada por ancho de pulso. La frecuencia del oscilador se puede configurar por medio de un circuito RC (resistencia y capacitor) y conectarlo a pin #1 del chip L6506. La frecuencia de oscilación se determina por:

$$f_{osc} = \frac{1}{0.69(RC)} \quad (3\text{-}1)$$

El oscilador proporciona los pulsos PWM a dos flip-flops, que funcionan como interruptores, y son los responsables de activar las salidas de la etapa de potencia.

Para interrumpir la corriente en los devanados del motor a pasos, este circuito utiliza dos comparadores conectados a los flip-flops que sensan el voltaje que llega de dos resistencias llamadas R_{SA} y R_{SB} que proviene del circuito L298 en la etapa final de potencia y que son conectados en paralelo a los pines V_{sense1} y V_{sense2} del chip L6506 como se observa en la figura 3-7. Cuando el voltaje de los pines V_{sense1} y V_{sense2} supera o alcanza el voltaje de referencia V_{ref} el circuito deja pasar corriente a los motores a pasos y significa que el motor *ha llegado a consumir una corriente pico* (que por diseño no tiene que superar a los 2A).

En el proyecto, el integrado L6506 está implementado para controlar la corriente y limitarla a efecto de controlar el consumo de corriente por los motores a paso, de modo que la corriente pico en la carga del motor no exceda la capacidad de corriente de la etapa de potencia, ni tampoco exceda la corriente en las bobinas de los motores, la ecuación que ilustra cómo es controlada la corriente en la parte comparadora del chip L6506 cuando va acompañado de los puentes H L298 es:

$$I_{pico} = \frac{V_{Ref}}{R_{Sensado}} \quad (3\text{-}2)$$

El L6506 es útil cuando se trabaja con otros circuitos integrados donde es necesario tener compatibilidad a nivel TTL (5V), como el puente H doble L298 y microcontroladores.

Para poder implementarlo es necesario calcular la frecuencia del oscilador y la corriente pico puede fijarse en los motores paso a paso, como se explicó en la sección 2.2.3.

El circuito propuesto para implementar la unidad de potencia en el proyecto se muestra en la figura 3-7. Este circuito opera conjuntamente los integrados L298 y L6506, además ayuda a realizar los cálculos necesarios para saber a qué frecuencia trabajará el oscilador y el corriente pico máximo que puede consumir los motores a pasos.

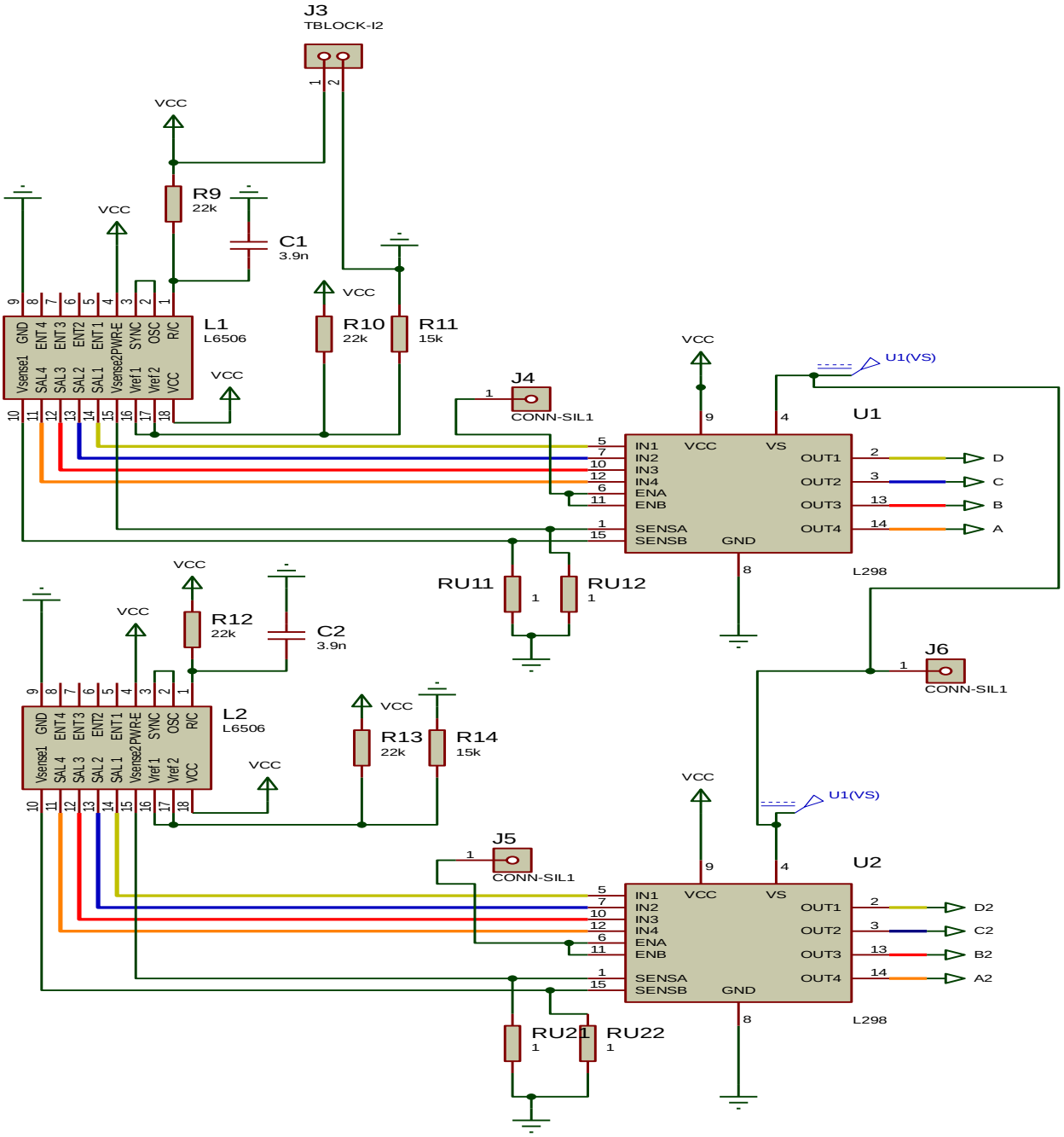


Figura 3-7. Circuito electrónico que comprende la etapa de control de corriente y la etapa de potencia.

La *frecuencia del oscilador* está determinada por la ecuación 2-3 y utiliza los elementos de la red RC conectados al chip L6506 en su pin número 1, esta frecuencia se calcula como:

$$f_{osc} = \frac{1}{0.69(RC)} = \frac{1}{0.69(R_9 C_1)} = \frac{1}{0.69(22K\Omega\ x\ 3.9nF)} = 16.8KHz$$

El voltaje de referencia, en los pines de los comparadores, se determina por un circuito divisor de voltaje. El divisor de voltaje está conformado por las resistencias R10, R11, R13 y R14 conectadas a los pines 16 y 17 del chip L6506, como se muestra en el esquema de la figura 3-6. En base a ellas el voltaje necesario para conocer la corriente pico de consumo en los motores a pasos, se calcula como:

$$V_{ref1} = \frac{R_{11}}{(R_{10} + R_{11})} V_{cc} = \frac{15K\Omega}{(22K\Omega + 15K\Omega)} (5V) = 2.97V$$

$$V_{ref2} = \frac{R_{13}}{(R_{13} + R_{14})} V_{cc} = \frac{15K\Omega}{(22K\Omega + 15K\Omega)} (5V) = 2.97V$$

Por la ecuación 3-2 y una resistencia de sensado de 1Ω, la corriente pico es

$$I_{pico} = \frac{V_{Ref}}{R_{Sensado}} = \frac{2.97V}{1\Omega} = 2.97A$$

3.3.1 Simulación del divisor de voltaje en el control de corriente

Para verificar el voltaje de referencia y la corriente pico calculada, se realizó una simulación del circuito divisor de corriente. El circuito implementado en el software Multisim 13 se muestra en la figura 3-8.

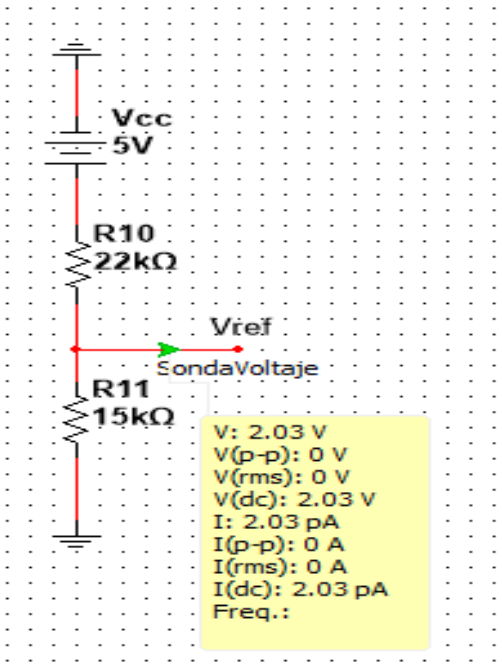


Figura 3-8. Divisor de voltaje implementado en el pin 16 y pin 17 del chip L6506,

En el resultado de la simulación obtenido en el recuadro amarillo se mide voltaje de referencia con una sonda, este voltaje es igual a 2.03V. Este valor es menor a la corriente máxima soportada por CI L298.

3.4 Diseño de pulsos para la emisión de ultrasonido

Se generaron cuatro pulsos en la plataforma Matlab 2013a®, que son registrados por el generador de funciones del *Handy Scope HS3. Para las aplicaciones consideradas en el proyecto los pulsos incluidos son:*

1. Pulsos cuadrados negativos o positivos.
2. Pulsos Sa (función sinc).
3. Pulsos exponenciales decrecientes.
4. Pulsos gaussianos.

Los pulsos cuadrados, los más comunes en los sistemas comerciales, son de utilidad para la calibración del sistema, sin embargo tienen la desventaja que no permite excitar al emisor en su ancho de banda. Los pulsos de excitación están definidos en la tabla 3-1, están programados y guardados en la memoria del *Handy Scope HS3* e incluidos en la interfaz de usuario.

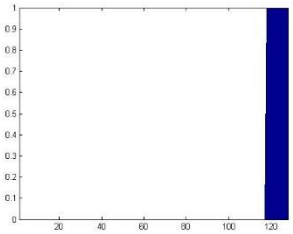
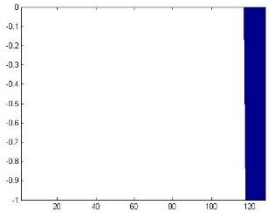
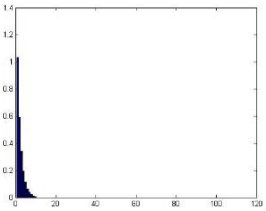
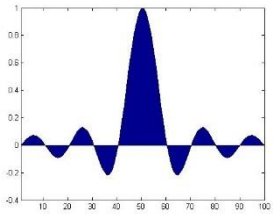
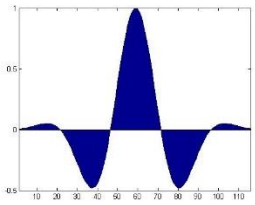
Nombre del perfil	*Definición*	*Forma del perfil*
Pulso cuadrado	$f(n)=\begin{cases}1 & \forall n = 0 \dots nT_0 \\ 0 & \forall n \neq 0 \dots nT_0\end{cases}$	
Pulso cuadrado negativo	$f(n)=\begin{cases}-1 & \forall n = 0 \dots nT_0 \\ 0 & \forall n \neq 0 \dots nT_0\end{cases}$	
Pulso exponencial	$f(n)=\begin{cases}Ae^{-an} & \forall n = 0 \dots nT_0 \\ 0 & \forall n \neq 0 \dots nT_0\end{cases}$	
Pulso sinc	$f(n) = Sa(\omega_0 n) = \frac{\sin(\omega_0 n)}{\omega_0 n}$	
Pulso senoidal modulado con una envolvente gaussiana	$f(n) = (1-\cos(\omega_0 n))\cos(\omega_0 n)$	

Tabla 3-1. Representaciones de los perfiles y su definición, estos pulsos están programados en HS3.

3.5 Diseño e implementación de la interfaz gráfica de usuario

La interfaz gráfica de usuario fue elaborada con el software Matlab2013a®, con esta interfaz se puede controlar y configurar el tipo de inspección a realizar por sistema de emisión y adquisición de señales ultrasónicas, el software del sistema de adquisición y emisión de señales ultrasónicas se implementó y se dividió en tres interfaces diferentes, una de ellas corresponde a una interfaz principal (figura 3-9), y

otras dos interfaces secundarias, que cumplen con el objetivo de revisar el funcionamiento general del sistema tomografía ultrasónica (figuras 3-11 y 3-12).

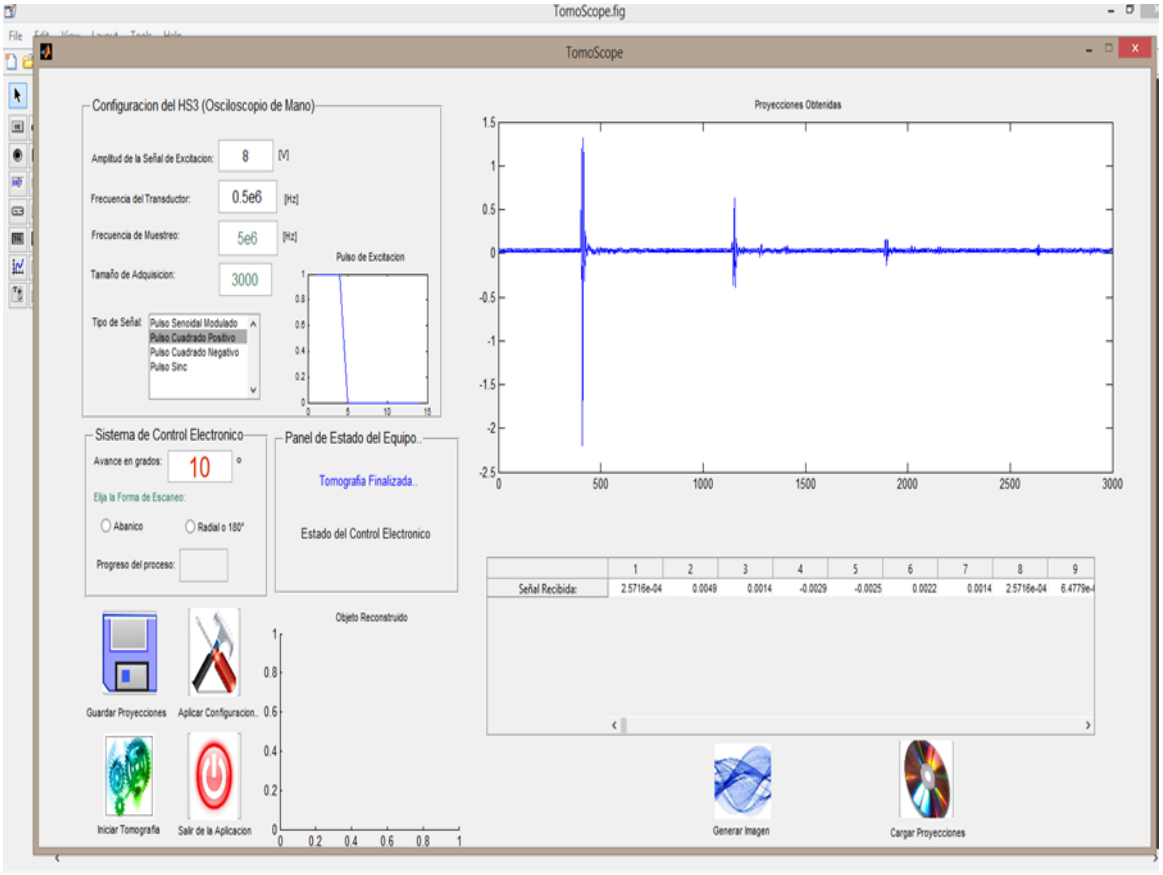


Figura 3-9. Interfaz de usuario principal

El funcionamiento de la interfaz principal, se basa en el diagrama de flujo mostrado en la figura 3-10. Permite configurar y arrancar el sistema de adquisición ultrasónica para realizar la inspección del objeto, en esta interfaz se puede observar tanto el control electrónico de motores a paso y el módulo de adquisición y emisión de ultrasonido; si están conectados y configurados antes de iniciar cualquier proceso de tomografía.

En el diseño de la interfaz principal se implementaron ejes para la visualización tanto de las señales que se adquieren como la del objeto inspeccionado. Adicionalmente tiene la posibilidad de graficar archivos de señales ultrasónicas almacenadas en disco, guardar el conjunto de señales obtenidas junto con la imagen de la sección del objeto inspeccionado.

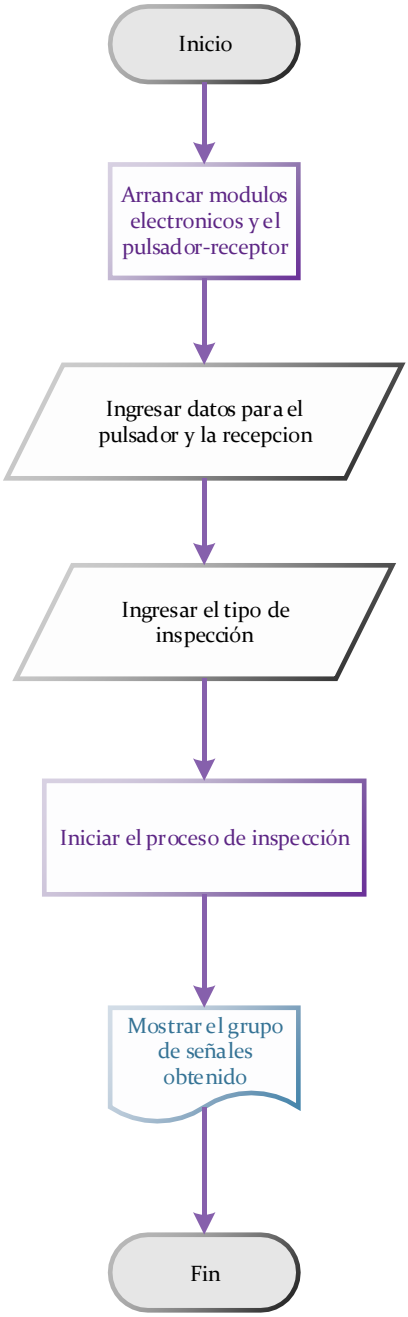


Figura 3-10. Diagrama de flujo del proceso de arranque y de inspección general del tomógrafo.

La interfaz secundaria, se diseñó con el objetivo de comprobar el funcionamiento adecuado del control electrónico de motores a paso.

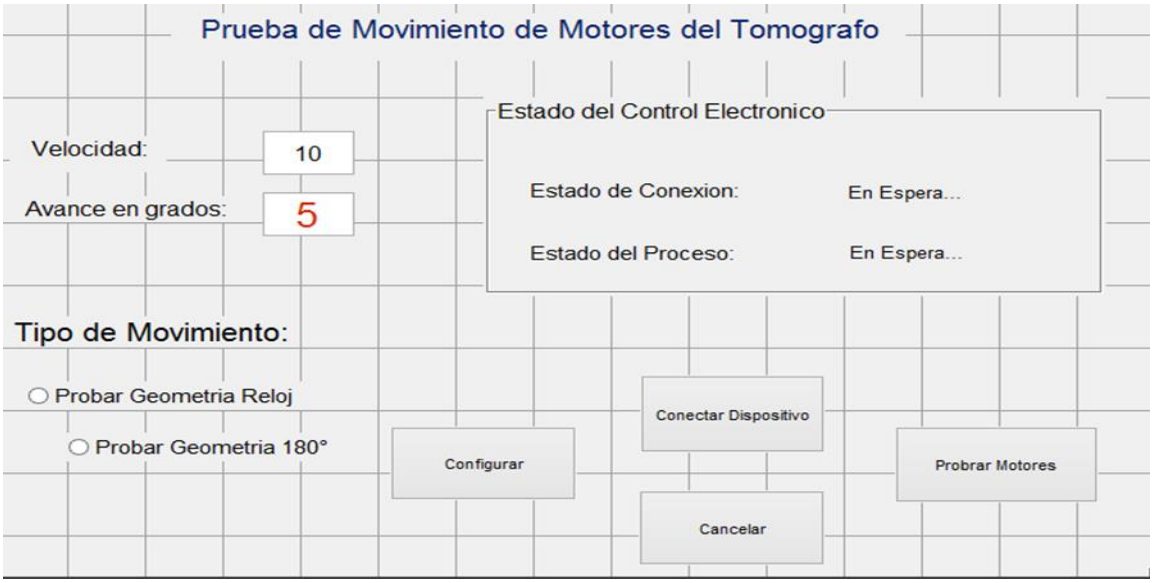


Figura 3-11. Interfaz secundaria para probar el control electrónico de motores a paso.

La interfaz secundaria, tiene la finalidad de comprobar el funcionamiento del pulsador y la adquisición del ultrasonido del tomógrafo ultrasónico, en esta interfaz se implementaron los pulsos de emisión para el transductor que emite el ultrasonido.

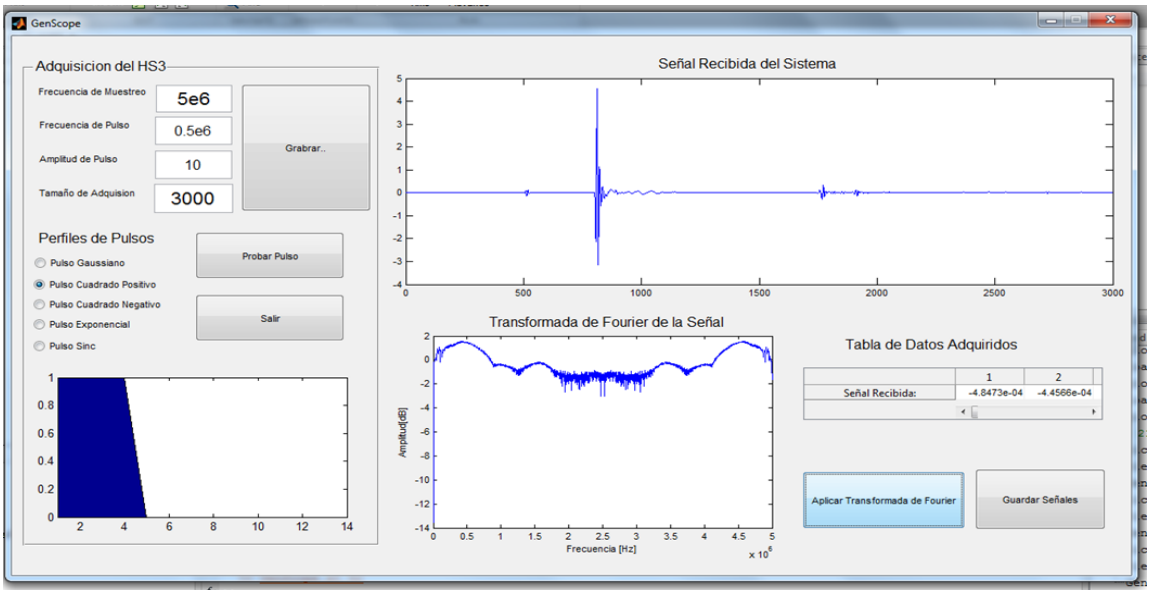


Figura 3-12. Interfaz gráfica secundaria, módulo de prueba del pulsador y de la adquisición de ultrasonido.

Esta interfaz cuenta con módulos de visualización de la señal que se adquiere en el dominio del tiempo y en el dominio de la frecuencia, la interfaz secundaria está habilitada para guardar las señales que se están adquiriendo, el programa del software de todas las interfaces puede observarse en el apéndice A2.

3.6 Integración del sistema de control electrónico de los motores a pasos

Para controlar el paso de los motores y sincronizarlo con el envío de los pulsos ultrasónicos en cada posición de los motores, fue necesario construir una tarjeta de circuito impreso. El circuito final, desarrollado en Proteus 8® (ver Figura 3-13), se basó en las características físicas del tomógrafo y lo descrito de la electrónica en la sección 2.2.

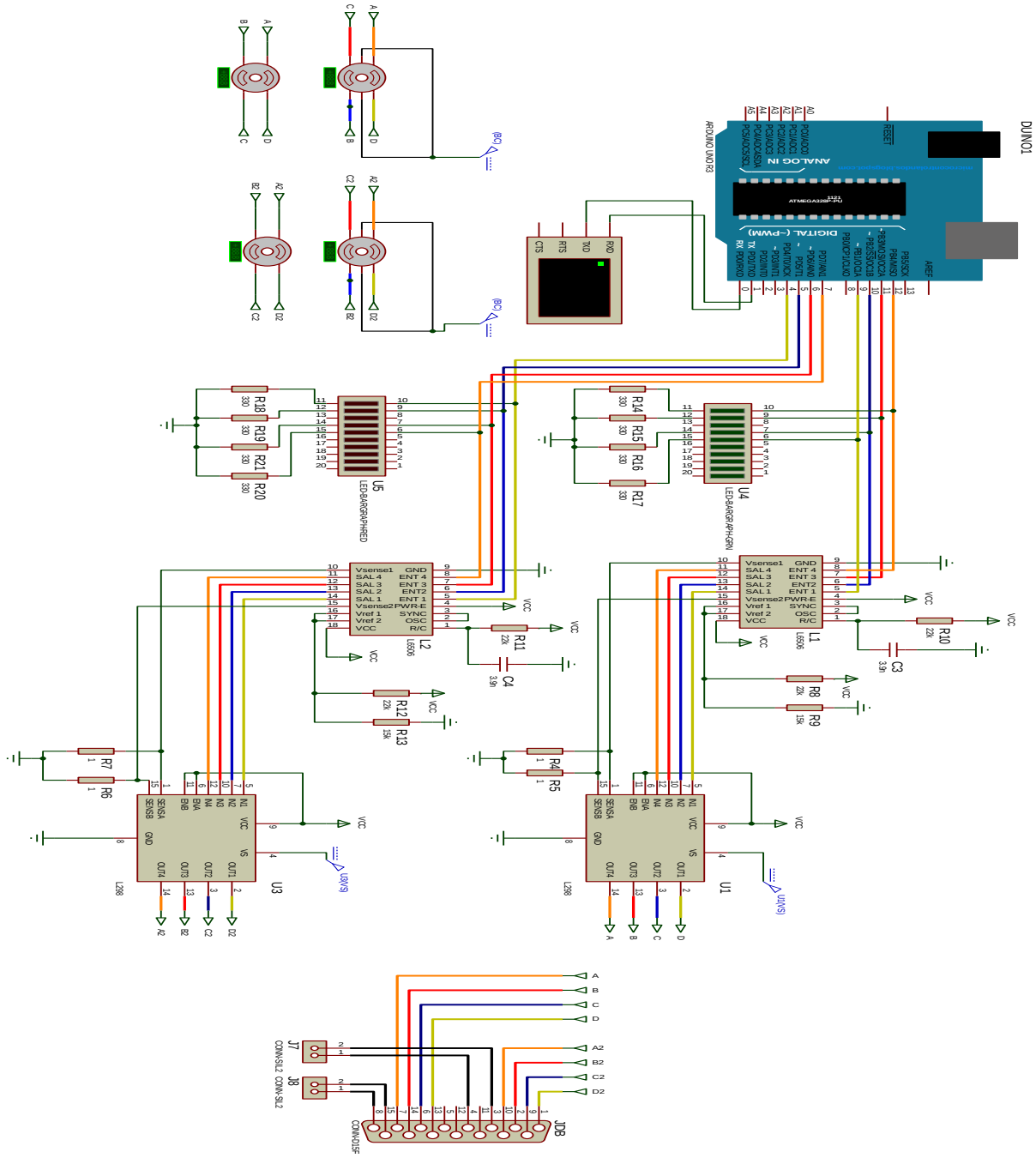


Figura 3-13. Esquema final del circuito electrónico de control de motores a paso.

Para su integración se requirió elaborar un tarjeta de circuito impreso para su montaje, esta tarjeta de circuito impreso se desarrolló con ayuda del software de Ares® de Proteus 8 (el circuito impreso puede verse en el apéndice A5).

En el diseño de esta tarjeta de circuito impreso, no se incluyó la tarjeta Arduino UNO®, ya que viene diseñada en su propia PCB y solo tiene que montarse al diseño final.

Una vista en 3D del diseño de la tarjeta del control electrónico de los motores a paso se puede observar en la figura 3-14 esta vista da una idea del aspecto final del PCB para su montaje.

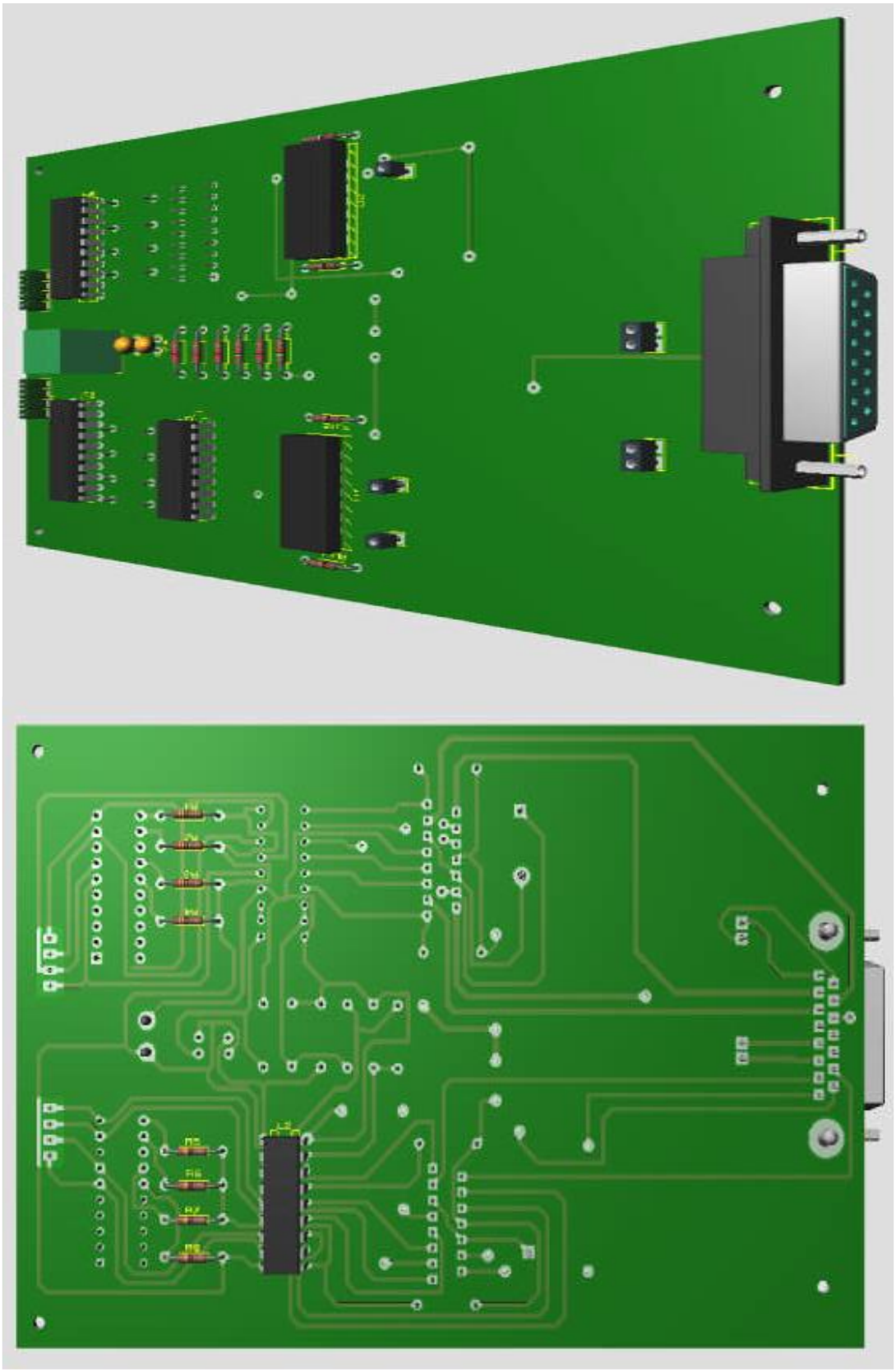


Figura 3-14. Vista 3D del control electrónico de motores a paso.

El control electrónico de los motores interactúa con el sistema mecánico y el software de usuario por medio de una tabla de comandos, parte del programa principal del control electrónico, en el programa principal se distinguen dos tablas: una para configurar la adquisición radialmente (ver tabla 3-2) y otra para adquirir señales de ultrasonido en modo abanico (ver tabla 3-3).

Comando	*Función que se ejecuta*
"1"	Mueve el transductor receptor θ grados.
"2"	Mueve el transductor emisor θ grados.
"5"	Retorna al transductor receptor a la posición de origen.
"R"	Prepara al sistema de adquisición de señales para moverse en forma radial.

Tabla 3-2. Funciones y comandos para adquirir señales en modo radial.

Comando	*Función que se ejecuta*
"3"	Mueve el transductor receptor θ grados.
"4"	Mueve el transductor emisor θ grados.
"6"	Coloca al transductor receptor (n-1) posiciones del transductor emisor.
"A"	Prepara al sistema de adquisición de señales para moverse en forma abanico.

Tabla 3-3. Funciones y comandos para adquirir señales en modo abanico.

Otra característica de este circuito es que la corriente que consume el circuito de control cuando mueve los motores a pasos es de 1.52A, esto permite obtener un tiempo de vida de la unidad de potencia más largo.

La comunicación entre el modulo bluetooth (HC-06) del circuito de control de motores y la computadora se lleva cabo emulando un puerto serie en la computadora (COM), la velocidad de transmisión de datos está configurada a 9600 baudios.

Capítulo 4. Integración Final del Sistema de Tomografía de Dos Elementos Transductores

El objetivo de este capítulo es mostrar los resultados de los componentes que integran el sistema de tomografía ultrasónica en transmisión.

El sistema de tomografía ultrasónica desarrollado en este proyecto se muestra en la figura 4-1. Se compone del sistema mecánico, del sistema de posicionamiento electrónico, de un pulser ultrasónico, preamplificador y un par de transductores. Adicionalmente se desarrolló el despliegue gráfico que permite al usuario tener control del movimiento de motores en una trayectoria circular, el tipo de pulso de excitación y el despliegue gráfico de las señales adquiridas.

Figura 4-1. Sistema de adquisición de señales de ultrasónico integrado por completo.

4.1 Pulsos de excitación y respuesta obtenida en el transductor receptor

Las respuestas de los transductores ultrasónicos al ser excitados por los pulsos de emisión, descritos en la sección 3.4, se midieron en base a lo siguiente: se utilizaron dos transductores Olympus para inmersión en agua de 500 KHz de frecuencia de operación; uno de ellos es el receptor y otro el emisor. La distancia entre las caras paralelas de los transductores es de 25 cm y están inmersos en agua. La frecuencia de adquisición de las señales recibidas es de 10 MHz. La tabla 4-1 muestra las respuestas del transductor- receptor almacenada en un osciloscopio de la marca Rigol® y en el despliegue gráfico desarrollado durante el proyecto de tesis, a través del HandyScope HS3®.

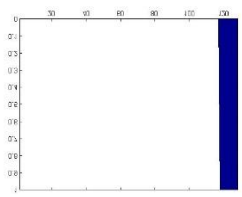
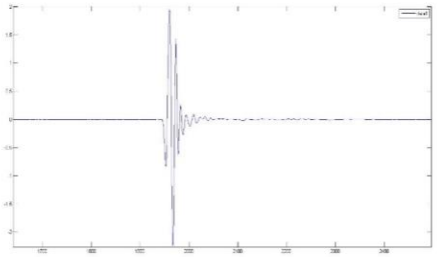
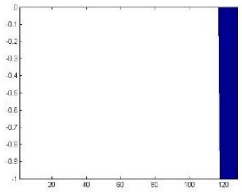
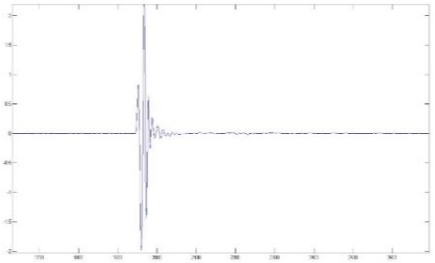
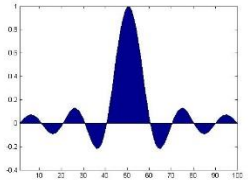
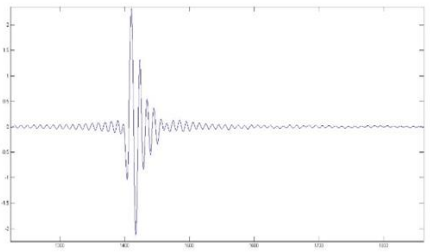
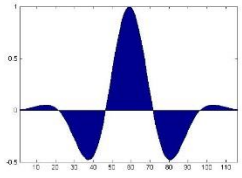
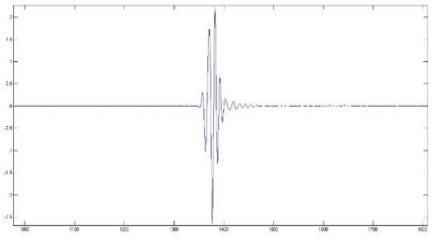
Pulso de excitación	Respuesta recibida en otro osciloscopio	Sistema de adquisición de señales ultrasónicas
 Pulso cuadrado positivo		
 Pulso cuadrado negativo		
 Pulso sinc		
 Pulso gaussiano		

Tabla 4-1. Respuesta del transductor receptor a distintos pulsos de excitación.

4.2 Sinogramas generados por el sistema de adquisición de señales ultrasónicas

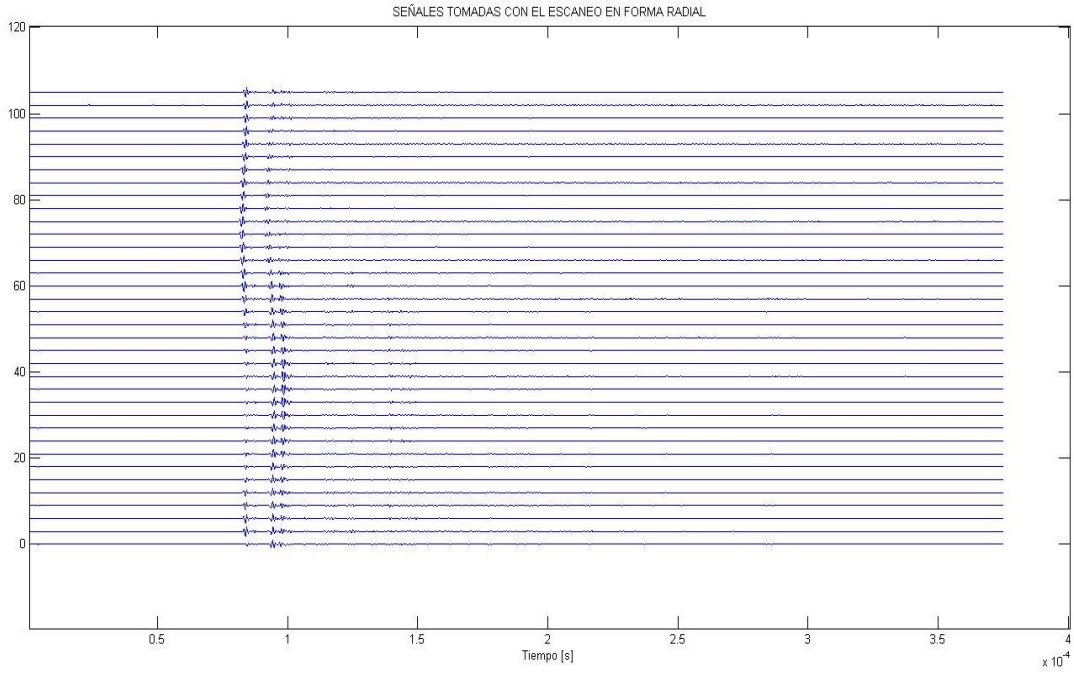
El funcionamiento del sistema tomográfico ultrasónico de dos elementos consiste en el movimiento de los transductores en forma circular en un radio de 15 cm. Éste tiene la capacidad de mover los elementos con pasos a partir de 5°, donde el emisor permanece estático hasta que el receptor se movió casi 360°. Para cada una de las posiciones emisor/receptor se envía un pulso de excitación y se almacena el eco en recepción, formando los sinogramas. Adicionalmente el despliegue gráfico permite seleccionar la frecuencia de muestreo, el número de muestras, el pulso de excitación, la frecuencia de operación, etc., permitiendo que el usuario tenga el mayor control de los parámetros experimentales.

Se programaron dos tipos de inspección: radial y en abanico. Para ambos casos se utilizaron unos cilindros de aluminio y bronce al 90% de pureza con radio 4.8cm y una altura de 14cm, una probeta de concreto y un cubo de aluminio, transductores Olympus de inmersión en agua que operan a 1 MHz y a 500KHz, la frecuencia de muestreo es de 10 MHz y los pulsos de excitación elegidos fueron: pulso cuadrado positivo y un pulso gaussiano modulado.

4.2.1 Inspección radial

Las inspecciones radiales (uniaxiales) consisten en que ambos transductores estén alineados en una recta, es decir a una posición de 180° de diferencia. Este arreglo permaneció fijo para cada movimiento de motores. Para el caso de la probeta de aluminio (ver figura 4-2), se adquirieron las señales mostradas en las Figuras 4-3 y 4-4, donde los sinogramas adquiridos para el movimiento del arreglo cada 10° y 20°, respectivamente, cubriendo una región total de 360°. De estas señales y sus acercamientos se puede observar que la probeta no estaba totalmente centrada, sin embargo, la diferencia mínima en los retrasos de los ecos permite concluir que es un material homogéneo con una geometría regular, ya que no varía la forma de los pulsos recibidos respecto al ángulo de visión. Por otro lado tenemos más información sobre el material de prueba al inspeccionarlo con un paso de motores menores, como se muestra en la Figura 4-4.

Figura 4-2. Probeta de aluminio.



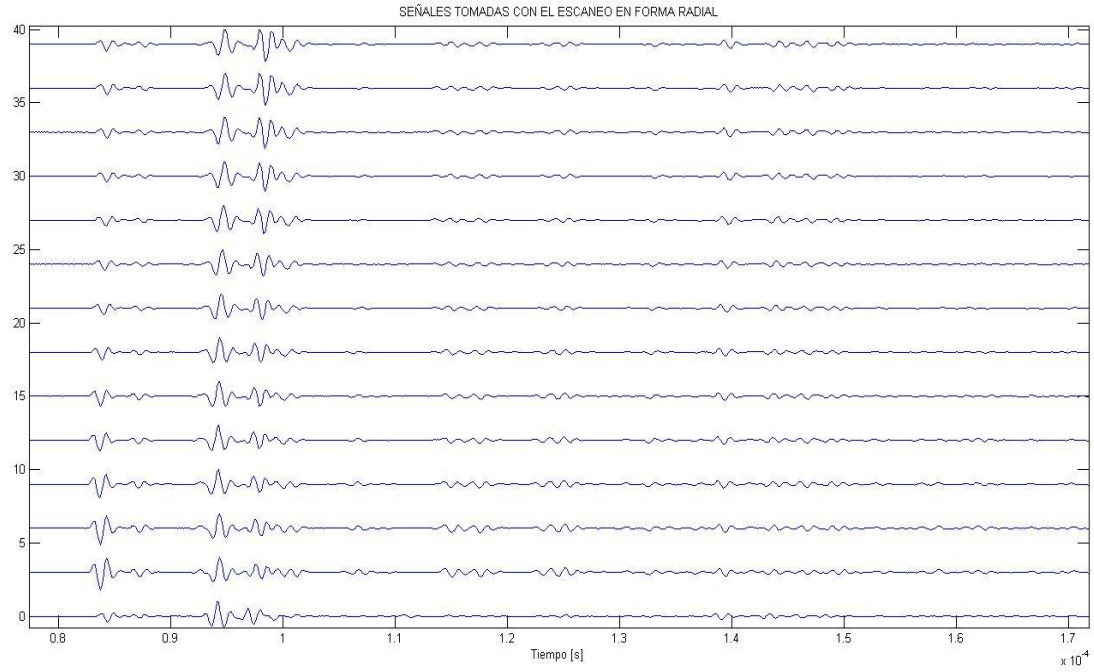
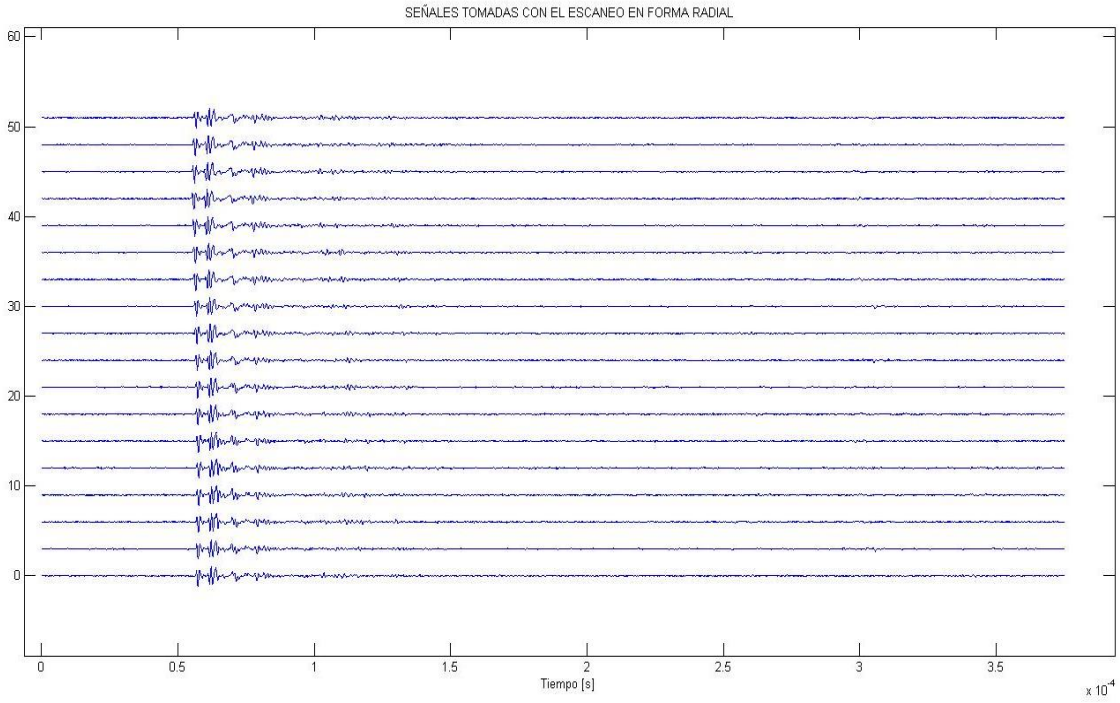


Figura 4-3. Sinograma a 10° y un acercamiento a las señales adquiridas (abajo).



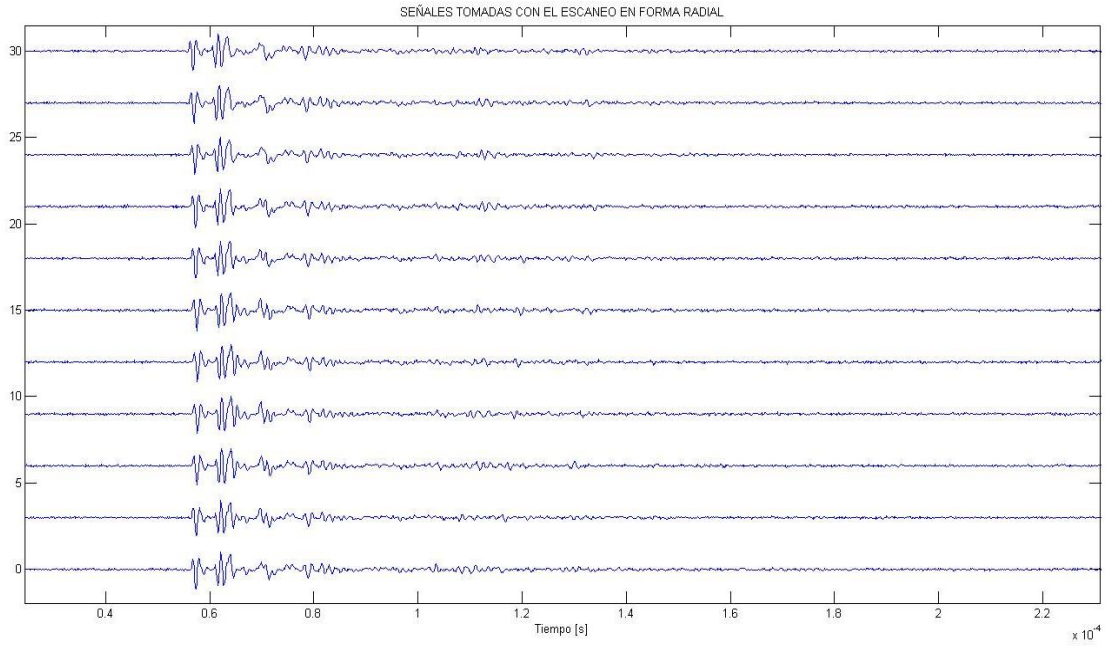


Figura 4-4. Señales a 20° y un acercamiento a las señales (abajo).

Los sinogramas adquiridos al examinar la probeta de bronce (Figura 4-5), excitando al transductor con un pulso gaussiano modulado y pasos de 20°, se muestran en la figura 4-6.

Figura 4-5. Probeta de bronce usada en las inspecciones.

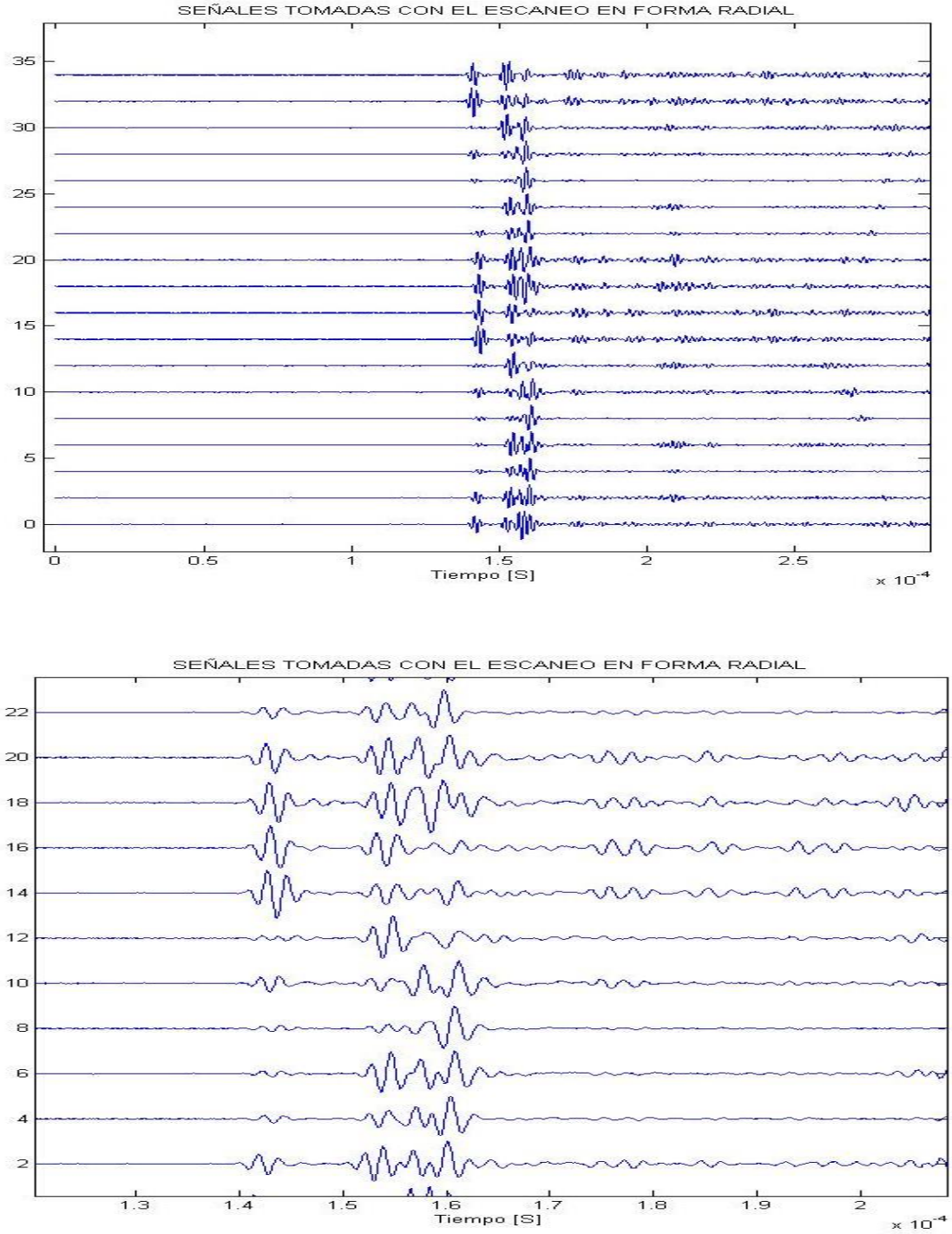
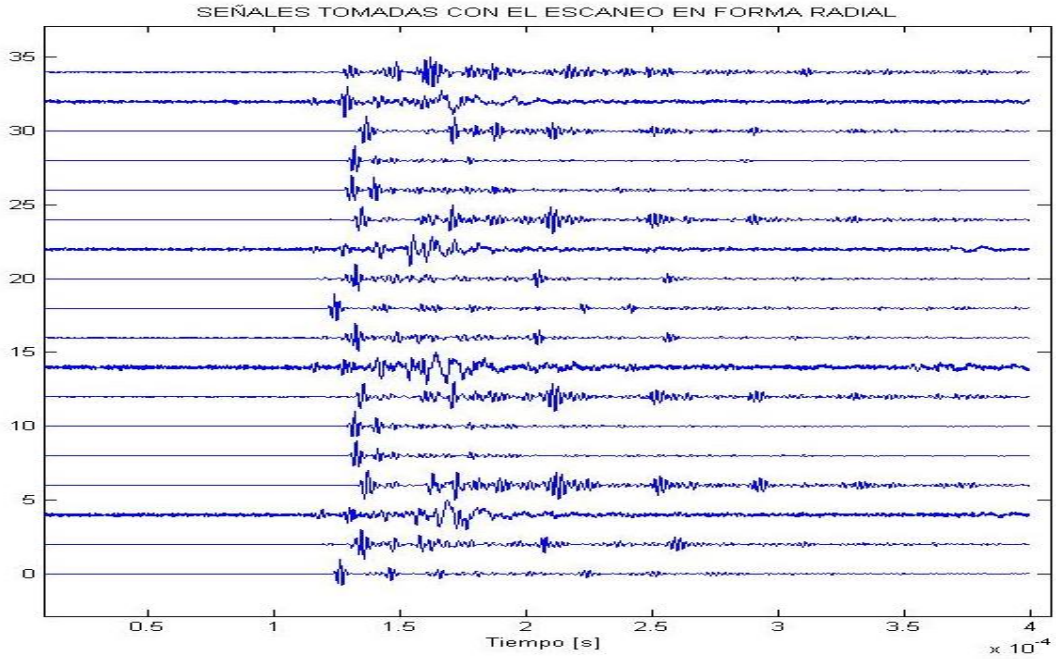


Figura 4-6. Sinograma a 20° excitando con un pulso gaussiano y un acercamiento a las señales.

Por último se experimentó con un cubo de aluminio (Figura 4-7) de 8cm x 8cm x 5cm, y la excitación del transductor ultrasónico se realizó con un pulso gaussiano. Los sinogramas generados se pueden observar en la Figura 4-8 junto con un acercamiento a estas señales, la adquisición fue tomada cada 20°.

Figura 4-7. Cubo de aluminio utilizado en las inspecciones radial y abanico.



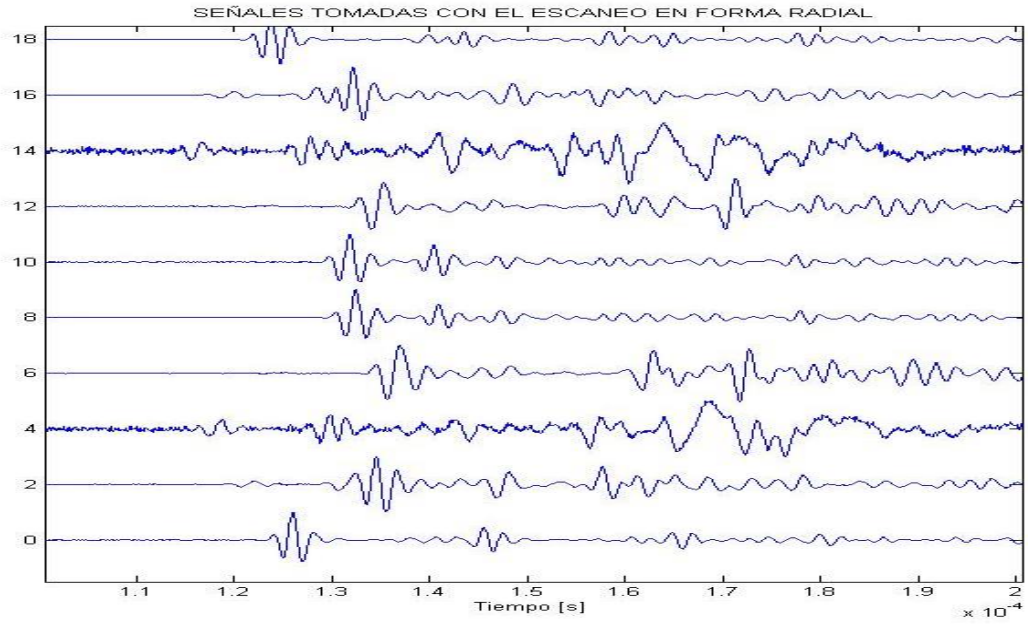


Figura 4-8. Sinograma radial tomado a 20° por paso y un acercamiento a las señales.

4.2.2 Inspección en abanico

La inspección en abanico consiste en que para cada posición del emisor, el receptor cubre 360°. Para cada posición Emisor/Receptor se adquiere y almacena la señal detectada.

Los sinogramas al inspeccionar la probeta de aluminio, a distintos pasos de motor y con el pulso de excitación cuadrado positivo se muestran en las figuras 4-9 y 4-10. Con este tipo de inspección podemos inferir que la probeta es de forma regular donde los retrasos se incrementan hasta cierto límite y pasando ese límite se disminuyen. El límite, que corresponde al punto de inflexión del tiempo de vuelo, sucede cuando los transductores se encuentran en la posición más lejana, es decir a 30 cm y están uniaxiales.

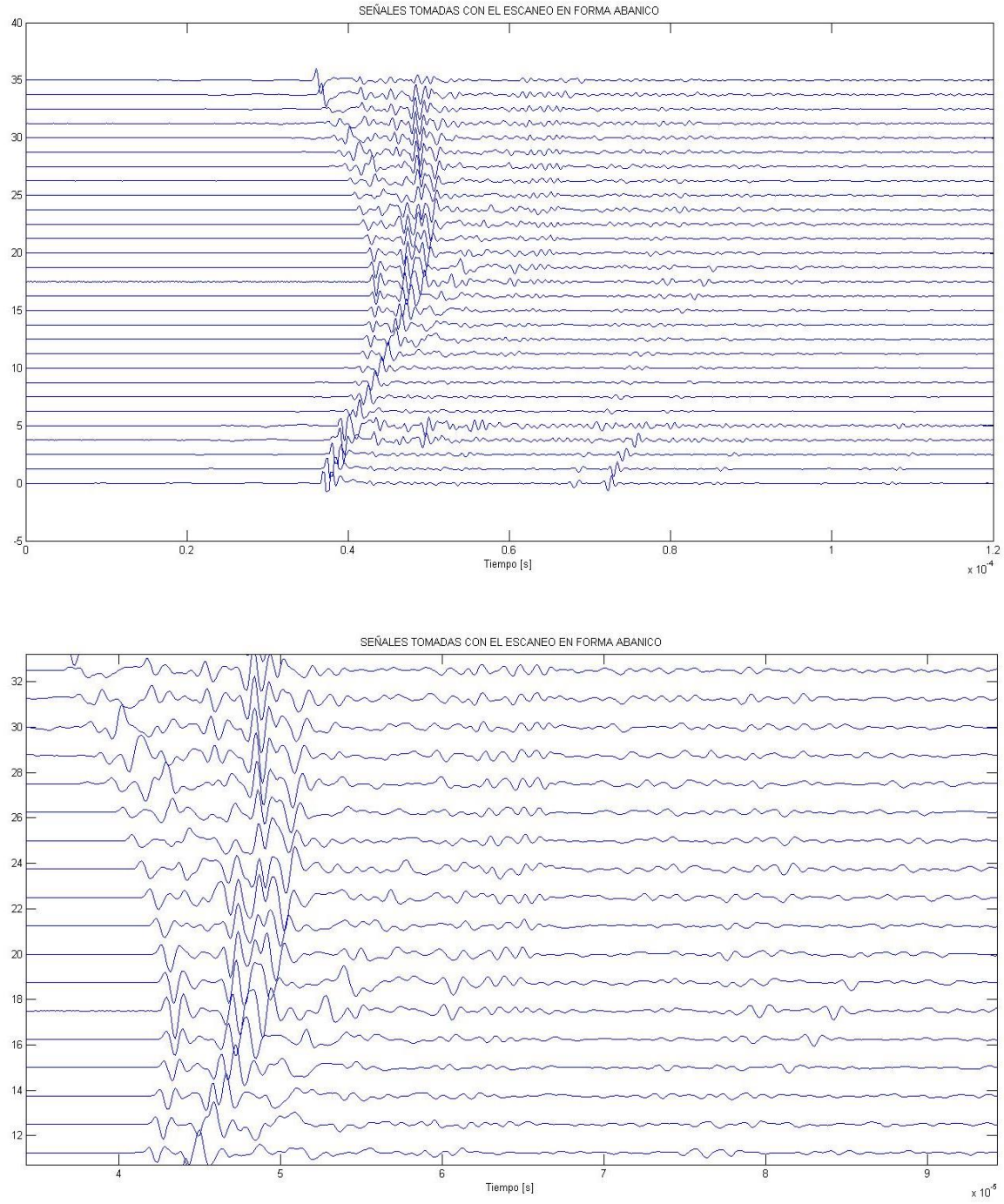


Figura 4-9. Sinograma a 10° y un acercamiento a las señales adquiridas.

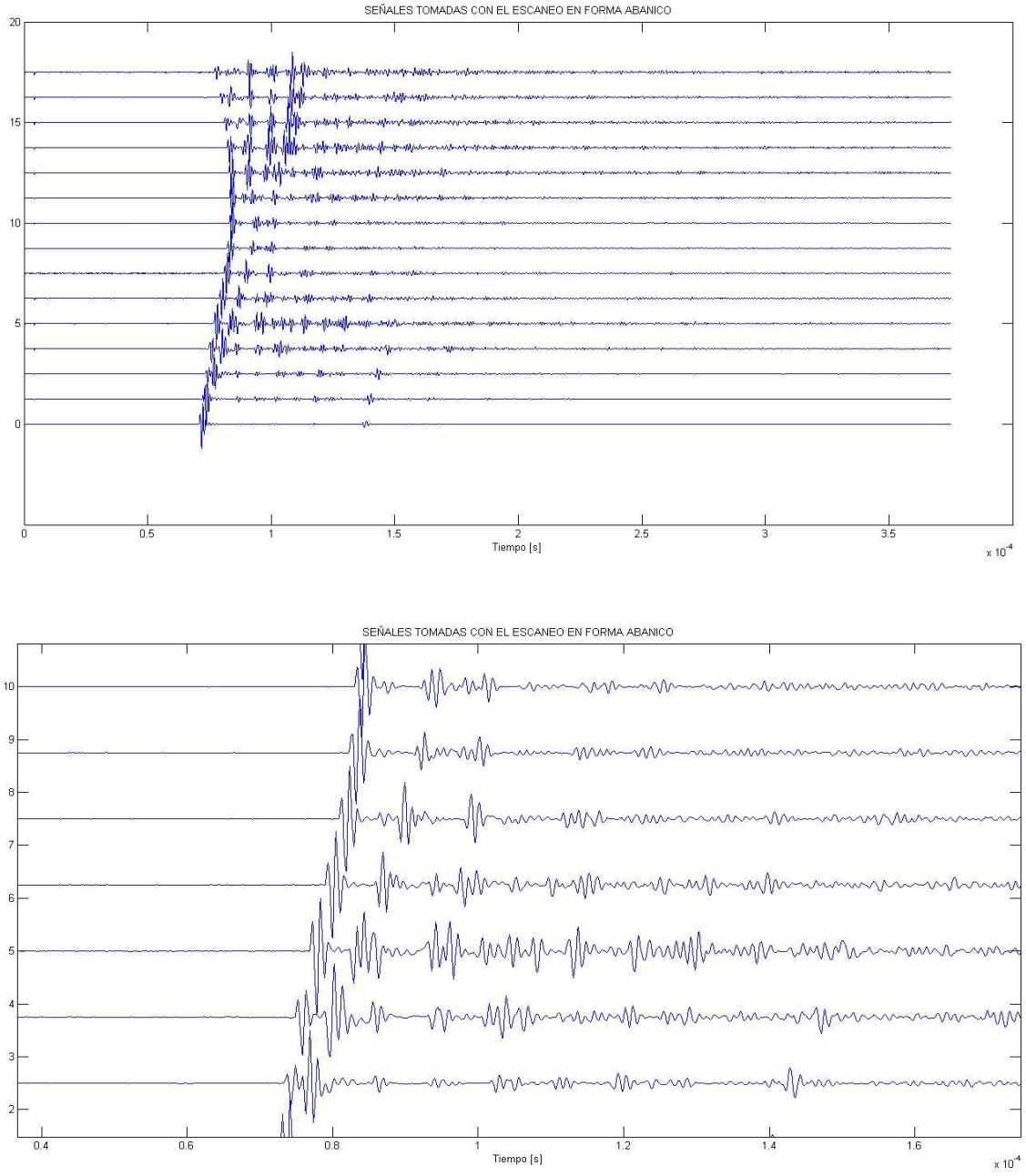


Figura 4-10. Sinograma a 20° y un acercamiento a las señales adquiridas.

Las Figuras 4-11 y 4-12 grafica los sinogramas al inspeccionar la probeta de bronce y excitar al emisor con un pulso gaussiano y pasos de motor de 5° y 10°, respectivamente.

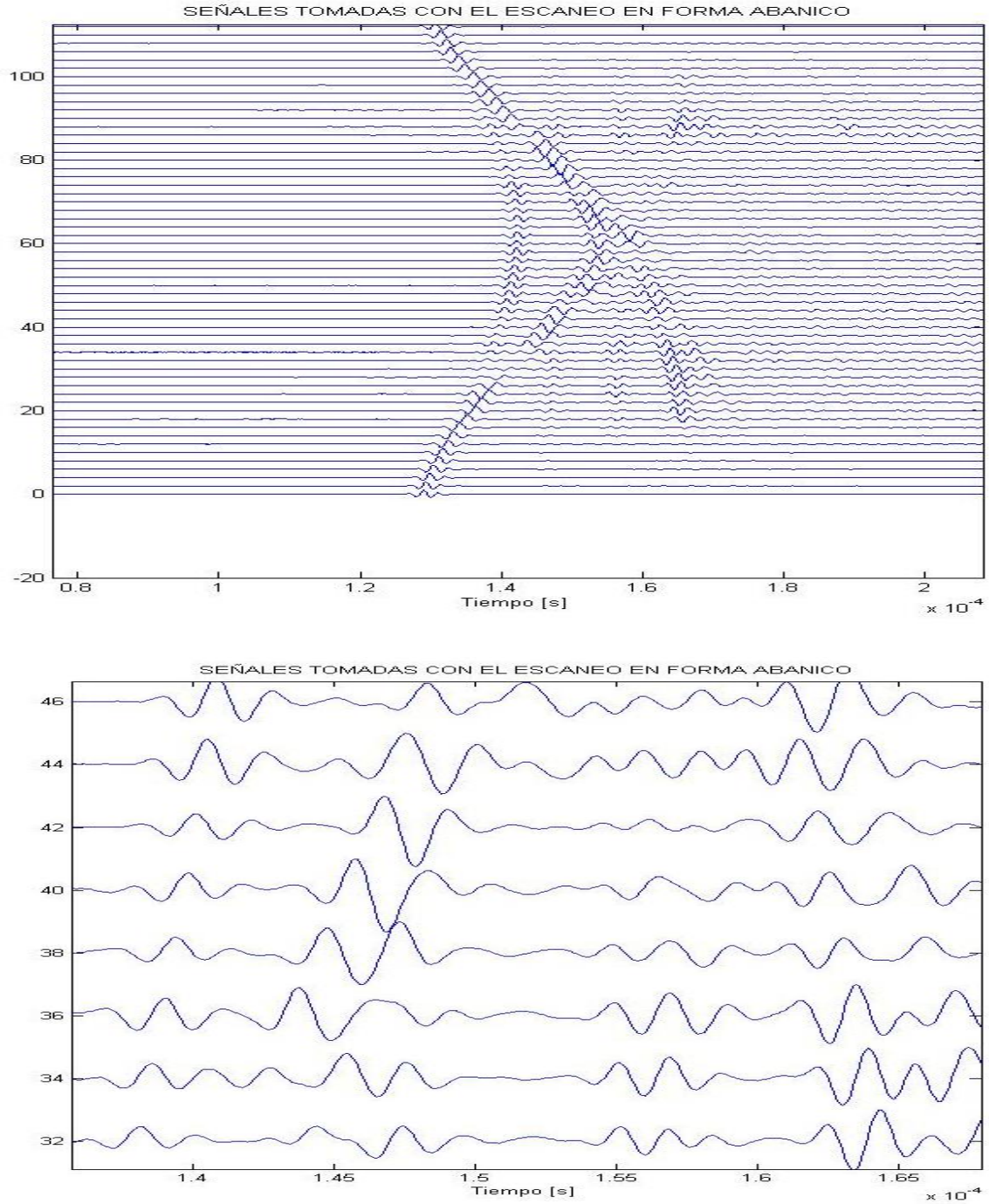


Figura 4-11. Sinograma abanico tomado a 5° y su acercamiento a las señales.

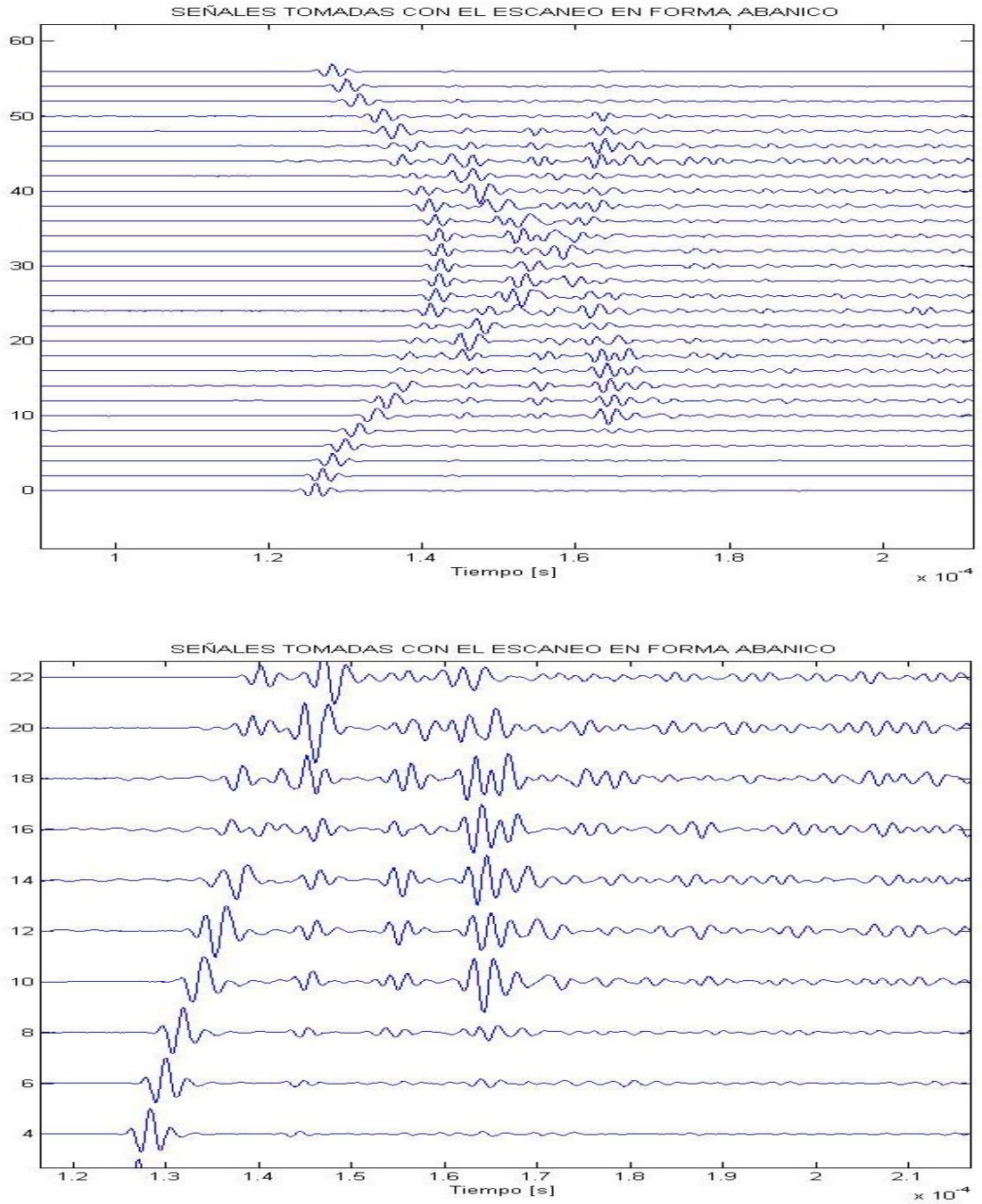


Figura 4-12. Sinograma adquirido a 10° y un acercamiento a estas señales.

Por último se trabajó con un material heterogéneo, un paralelepidedo de 8cm x 8cm x 5cm de aluminio (ver figura 4-7). Los sinogramas muestran gran diferencia con respecto al comportamiento de las señales adquiridas en un material no homogéneo, como se puede verificar en las figuras 4-13 y 4-14.

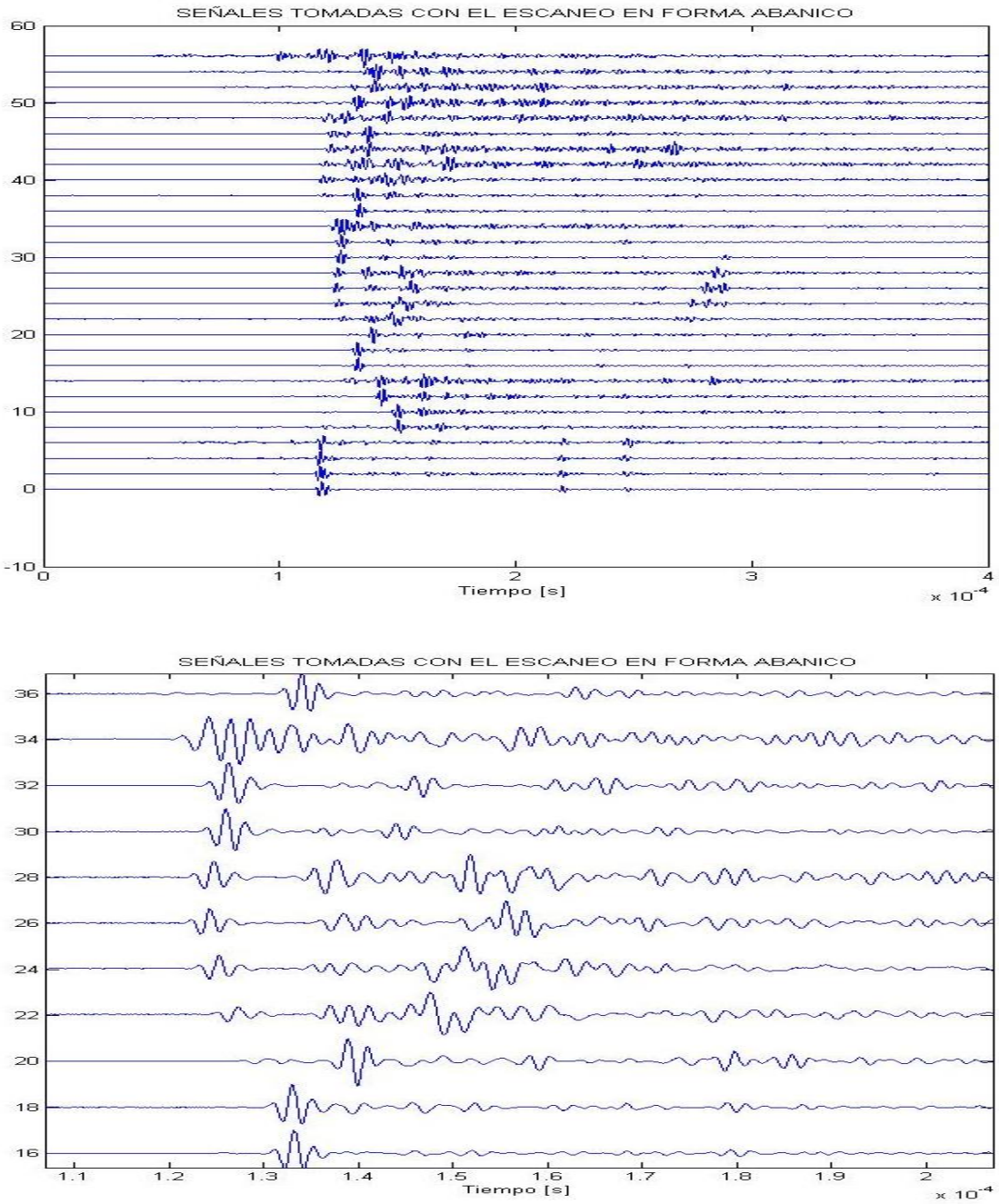


Figura 4-13. Sinograma en abanico adquirido a 10° por paso y un acercamiento.

En la Figura 4-15 se muestra el sinograma obtenido de examinar una probeta de concreto de 5.5cm de diámetro y 10.5cm de alto (Figura 4-14). El pulso de alimentación con el cual se excito al transductor emisor fue un pulso gaussiano, el sinograma fue adquirido a 5° por posición del transductor receptor.

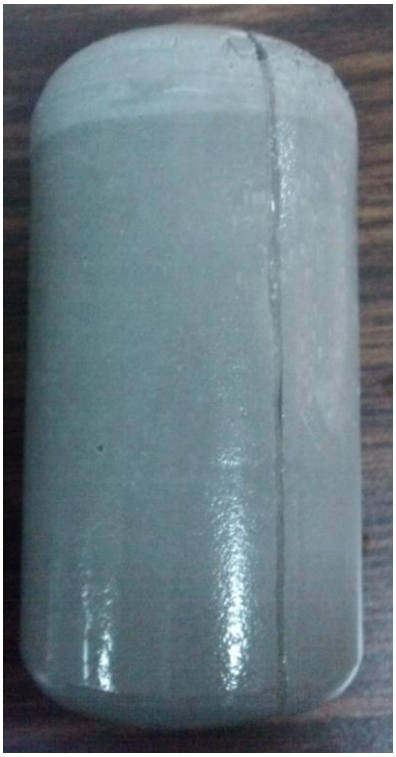
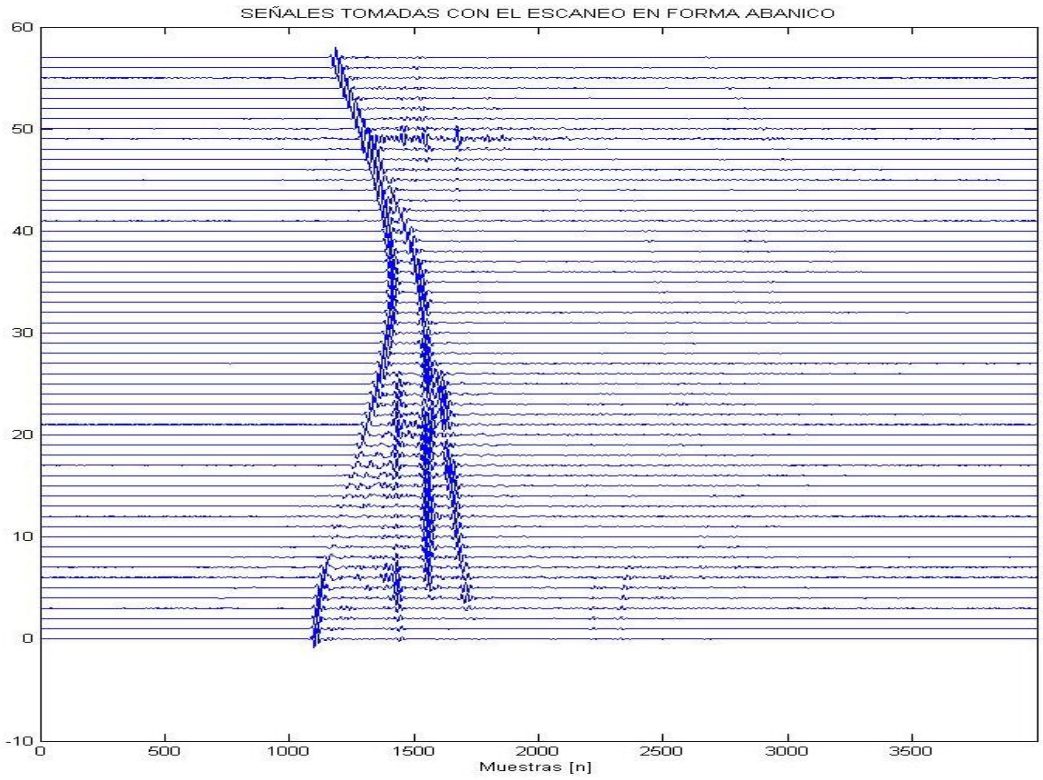


Figura 4-14. Probeta de concreto utilizada en las inspecciones en abanico.



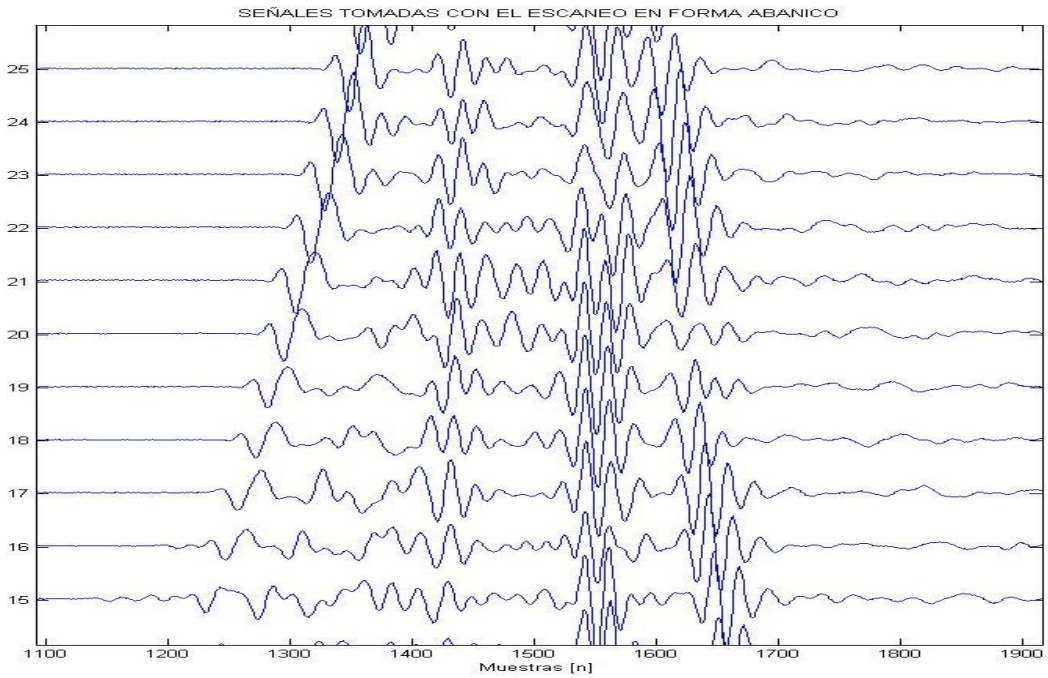


Figura 4-15. Sinograma en abanico adquirido a 5° por paso y un acercamiento.

A partir de los sinogramas y la información de los parámetros de adquisición y velocidad de propagación en el medio en estudio se puede reconstruir la imagen del material inspeccionado.

Capítulo 5. Conclusión y trabajo a futuro

El circuito de control electrónico de motores a pasos fue diseñado para poder manejar motores de 200 a 400 pasos, dependiendo del motor y de sus características eléctricas. Adicionalmente, puede cambiar la corriente que entrega a los motores de pasos con solo variar las resistencias del divisor de tensión en la unidad de control de corriente, lo cual ayuda a optimizar y hacer más eficiente el uso de la energía eléctrica; es decir, reduce el consumo de la energía que utilizan los motores a paso. Otra forma de controlar la energía que se entrega en los motores es haciendo que se desactiven los motores cuando estos no están en uso durante una inspección ultrasónica.

El control electrónico puede ser utilizado en cualquier otra aplicación donde se requiera mover un motor a pasos. Este circuito puede trabajar con motores de seis y cuatro hilos conductores y con secuencias de activación de paso completo y medio paso.

La forma de comunicación en el sistema electrónico es inalámbrica, esto es de gran ayuda, porque se elimina un cable que está conectado a la computadora, reduce el espacio utilizado por el sistema y libera puertos de la computadora.

El módulo de comunicación inalámbrico HC06 que se implementó, trabaja con la tecnología bluetooth, opera desde 9600baud hasta 11000bauds, lo cual es suficiente para enviar comandos para iniciar la inspección tomográfica. El módulo de comunicación por bluetooth solo usa cuatro pines, dos de ellos son tierra (GND) y alimentación (VCC) y para enviar y recibir información solo utiliza el puerto serie de la unidad de control conformada por la tarjeta arduino uno. Adicionalmente puede ser implementado con otras tecnologías como PICs, FPGA y DSPs.

La unidad de control fue desarrollada e implementada con la tarjeta arduino uno, lo cual tarjeta se tiene la facilidad de trabajar y utilizar diversos módulos como el de bluetooth y posee librerías para usar motores a pasos.

En el sistema de emisión y recepción de ultrasonido se requirió implementar un pulsador para emitir ultrasonido y un osciloscopio digital programable, para poder recibir las ondas de ultrasonido.

El ultrasonido se genera a partir de la emisión de pulsos de duración finita, que se implementan en el sistema de emisión y recepción de ultrasonido y corresponden a pulsos cuadrados positivos, negativos, gaussianos y sinc.

Los parámetros importantes para la formación de los pulsos de excitación son: la frecuencia de trabajo del transductor emisor y la velocidad de muestreo (frecuencia de muestreo del sistema), ya que estos parámetros son utilizados por el sistema de adquisición de señales para formar el perfil del pulso a enviar.

La recepción de las ondas ultrasónicas estuvo a cargo del dispositivo handyscope HS3®. Sin embargo, antes de ser detectadas por el osciloscopio digital las señales tienen que ser amplificadas, ya que las ondas de ultrasonido son atenuadas al propagarse dentro del objeto en estudio.

Las ondas de ultrasonido capturadas y el pulso de excitación son digitalizados a la misma tasa de muestreo. Para capturar la señal de ultrasonido, el sistema de recepción utiliza una señal de disparo (de control) que indica al osciloscopio cuando tiene que iniciar la captura del ultrasonido.

El sistema de emisión y adquisición ultrasónica emplea transductores para inmersión en agua para realizar las pruebas tomográficas, estos traductores tienen una frecuencia de trabajo de 500KHz y de 1MHz, esto no es una limitación, ya que el sistema de emisión y recepción está habilitado para trabajar con diversos tipos de transductores, tanto en frecuencia de operación como de inmersión o contacto.

Con la finalidad de manejar y configurar cada módulo que integra el sistema tomográfico, se diseñó una interfaz de usuario. Las funciones de la interfaz de usuario es configurar y comunicar el control electrónico de motores a pasos y el sistema de emisión-recepción de datos. Adicionalmente realizar tareas básicas como el almacenamiento, visualización y almacenamiento de las señales de ultrasonido. Respecto al módulo de adquisición, en la interfaz se puede configurar el número de muestras, que permite observar partes de las señales que se están adquiriendo o contener en la memoria del sistema de adquisición la señal completa. Además con el software de usuario es posible seleccionar entre dos tipos de inspección de ultrasonido; radial o abanico.

El arreglo de señales de ultrasonido (sinogramas) es resultado de realizar una inspección por ultrasonido, ya sea en formato radial o abanico. El sinograma que se forma cuando se escanea un objeto cilindrico tiene la característica de que las señales están en fase o alineadas, otra característica es que cada señal representa una posición del transductor emisor y receptor.

El número de señales que se producen por la tomografía con formato radial son resultado de distribuir los 360° de trayectoria circular de escaneo entre el número de grados en la inspección que fue configurado en el software de usuario.

Cuando se realiza una inspección con formato en abanico, las señales que se obtienen corresponden a una posición del transductor receptor con respecto al transductor emisor. Estas señales como conjunto tiene la característica de no estar en fase, lo cual cuando se escanea con este formato a un objeto circular o cilíndrico las señales se verán desplazadas (recorridas) en el eje temporal y pueden formar una curva cuando son vistas como arreglo, otra característica es que se puede tener más información acerca de la estructura del objeto que se esté escaneando con este formato y puede tener información redundante, además la forma de los sinogramas está relacionado con la estructura y la forma del cuerpo que se está inspeccionando.

El tomógrafo desarrollado en este trabajo de tesis está diseñado para cubrir un rango de 360° de exploración (siguiendo una trayectoria circular), a velocidad de posicionamiento constante y ángulo de paso constante configurado por software. El control de posicionamiento electrónico hace del tomógrafo muy fácil de configurar para realizar cualquier tipo de inspección que se desee ejecutar.

Con el software de usuario se pueden configurar los pulsos de excitación para generar ultrasonido, desde esta interfaz se puede trabajar con cuatro pulsos, con la posibilidad de agregar nuevos pulsos fácilmente. También es muy fácil cambiar la velocidad de muestreo y la frecuencia de operación de los transductores.

Como trabajo a futuro se puede mejorar el diseño mecánico aumentando un eje o una dimensión para que se pueda capturar señales de ultrasonido, realizar inspecciones y reconstruir imágenes volumétricas. De manera adicional si se desea aumentar el número de transductores se tendrá que aumentar la carga en los brazos y cambiar el diseño de estos brazos mecánicos que soportan a los transductores.

Referido al diseño electrónico al aumentar una dimensión más, se tiene que rediseñar o agregar un módulo a la unidad de potencia para trabajar con un tercer motor a pasos. La unidad de control tendrá que ser migrada a una nueva tarjeta de desarrollo que tenga más recursos o puertos.

No olvidando la conectividad y la multimedia que tiene el sistema electrónico de posicionamiento, también es posible y por mucho mejorarla añadiendo aplicaciones para realizar inspecciones sobre una tableta electrónica o arrancar una inspección vía internet.

Para mejorar el sistema de adquisición y emisión ultrasónica, también se puede incrementar el número de canales de recepción de modo que permita conectar un mayor cantidad de transductores y formar un arreglo de transductores para trabajar en transmisión de ultrasonido, sumado a lo anterior cabe la posibilidad de conmutar los canales emisión y recepción para que trabajen con ultrasonido con eco-pulsado.

En cuanto al software diseñado puede enriquecerse agregando nuevos pulsos de excitación para, mejorar aspecto visual, la interactividad con el usuario y además es necesario que se programen algoritmos de reconstrucción de imágenes que permitan la recuperación de la sección de un objeto.

Bibliografía

[1] Olympus, *Detección de defectos por ultrasonido,* http://www.olympus-ims.com/es/applications-and-solutions/introductory-ultrasonics/introduction-flaw-detection/, Octubre 2014.

[2] R. Budynas, J. Nissbett, *Diseño en ingeniería mecánica,* Mc Graw Hill, México, 2012.

[3] V. Solórzano, *Diseño y construcción de sistema básico de tomografía ultrasónica*, Tesis, UNAM, 2012.

[4] A. Sedra, K. Smith, *Circuitos Microelectrónicos*, Mc Graw Hill, México, 2006.

[5] T. Floyd, *Dispositivos Electrónicos*, Prentice Hall, México, 2008.

[6] Tie Pie, *Manual de programación DLL, Tie Pie Engineering, Netherlands, 2012.*

[7] A.Calzado, J.Gelejins, *Tomografía computarizada, evolución y principios técnicos, aplicaciones*, Madrid, 2010.

[8] L. Medina, E. Moreno, *Journal of Computational Acoustics – Ultrasonic Imaging Using a 2D Envelope Beamforming Technique: In-Homogeneities Location,* Volume 12, Number 4 *World* Scientific – Diciembre 2014.

[9] K. Ain, D.Kurniadi, A.Trisnobudi, *Summation Convolution Filtered Back Projection and Algebraic Reconstruction Technique for Ring Array Ultrasound Tomography Based on Time of Flight.* Indonesia, Noviembre 2011.

[10] C. Pintavirooj, A.Romputtal, *Ultrasonic refracttive index tomography,* Japon, 2003.

[11] P. Lasaygues, R. Guillermin, *Ultrasonic computed tomography,* Francia, 2010.

[12] I. Peterlik, R. Jink, *Algebraic reconstrucción technique for ultrasonic transmisión tomography,* Czech, 2008.

[13] G. Beylkin, *Discrete Radon Transform, NY*, 1987.

[14] D. Romero, O. Martinez, *Paralelización de los procesos de formación de haz para imagen ultrasónica con técnicas GPGPU,* Madrid, 2012.

[15] S. Weng, P. Chi, *Ultrasonic computed tomography reconstruction of the attenuation coefficient using a linear array,* Taiwan, 2005.

[16] E. Hecth, *Optica*, Pearson, Madrid, 2000 .

[17] A. French, *Vibraciones y Ondas,* MIT Press, Barcelona, 2006.

[18] P. Hoskins, K. Martin, *Diagnostic Ultrasonic, Physic and Equipment,* Cambrige Medicine, Edinburgh, 2010.

[19] J.David, N.Cheeke, *Fundamentals And Applications of Ultrasonics Waves,* CRC Press, Boca Raton, 2012.

[20] B. Carlin, *Ultrasonics*, Mc Graw Hill, 1982.

[21] D. Johnson, D, Dudgeon, *Array Signal Processing: Concepts and Techniques,* Prentice Hall, NJ, 1993.

[22] S. Umbaugh, *Digital Image Processing and Analisys "Human and Computer Vision Applications with CVIPtools",* Boca Raton, 2010.

[23] D. Ensminger, L. Bond, *Ultrasonic, Fundamentals, Technologies and Applications,* CRC Press, Boca Raton, 2011.

[24] A. Jain, *Fundamentals of Digital Image Processing,* Prentice Hall, 1988.

[25] M. Postema, *Fundamentals of Medical Ultrasonics,* Spon Press, London, 2011.

[26] A. Kak, M. Slaney, *Principles of Computed Tomographic Imaging,* IEEE Press, NY, 1999.

[27] F. Natterer, *The Mathematics of Computerized Tomography,* SIAM Press, Philadelphia, 2001.

[28] S. Helgason, *Integral Geometry and Radon Transforms,* Springer, Cambridge, 2010.

[29] L. Debnath, D. Bhatta, *Integral Transforms and Their Applications,* Champman and Hall/CRC, London, 2006.

[30] H. Moore, *MATLAB para Ingenieros*¸ Pearson Prentice Hall, México, 2007.

[31] P. Marchand, T. Holland, *Graphics and GUIs with MATLAB, Chapman and Hall/CRC,* Washington, 1999.

[32] E. Gómez, *Ultrasonidos Nivel II END ensayos no destructivos*, FC Editorial, Madrid, 2006.

[33] P. Martin, *Formación de la imagen en tomografía computarizada vía la transformada de Radón,* España, 2008.

[34] L. Brancheriau**,** *Application of ultrasonic tomography to characterize the mechanical state of standing trees*, Francia, Julio 2012.

Apéndice

A1. Programa del control electrónico de posición de los motores.

A continuación se muestra software que se programó y que está dentro de la tarjeta arduino:

```
/*Nota: Este programa, configura al control electronico para que trabaje haciendo que los motores se muevan segun el tipo de escaneo a realizar (ya sea en reloj o 180°).*/

/*Nota 2:

    a) Usar la definicion de "Motor 2" y "Motor 3" para moverse de manera plana a 180°.

    b) Usar la definicion de "Motor 1" y "Motor 2" para moverse en forma de reloj.*/

#include <Stepper.h>

char TipoMovimiento;
int ActB = 3;     //Activacion del motor RECEPTOR.
int ActA = 8;     //Activacion del motor EMISOR.
int Pasos = 50;
int Pasos1 = 0;
int Pasos2 = 0;
int Retro1 = 0;
int Flagpasos = 1;
int Opcion;


const int PasoFinal1 = 200;   //200 pasos completa 1 revolucion del motor, configurar segun el motor.  USADO EN GEOMETRIA RELOJ

const int PasoFinal2 = 200;  //200 pasos completa 1 revolucion del motor, configurar segun el motor.   USADO EN GEOMETRIA RELOJ

const int PasoFinal3 = 200;  //200 pasos completa 1 revolucion de los motores, configurar segun el motor. USADO EN GEOMETRIA 180

const int PasoFinal4 = 200;  //200 pasos completa 1 revolucion de los motores, configurar segun el motor. USADO EN GEOMETRIA 180

        //MOTORES USADOS EN GEOMETRIA RELOJ

Stepper Motor1(PasoFinal1,9,11,10,12);     //Definicion del motor unipolar en el Arduino, barra verde, caso A.  (ESTE MOTOR ES DEL TRANSDUCTOR: RECEPTOR).

Stepper Motor2(PasoFinal2,4,6,5,7);         //Definicion del motor unipolar en el Arduino, barra roja, caso B.   (ESTE MOTOR ES DEL TRANSDUCTOR: EMISOR).

        //MOTORES USADOS EN GEOMETRIA 180

Stepper Motor3(PasoFinal3,9,11,10,12);     //Definicion del motor unipolar en el Arduino, barra verde, caso A.  (ESTE MOTOR ES DEL TRANSDUCTOR: RECEPTOR).

Stepper Motor4(PasoFinal4,4,6,5,7);         //Definicion del motor unipolar en el Arduino, barra roja, caso B.   (ESTE MOTOR ES DEL TRANSDUCTOR: EMISOR).
```

```
void setup() {

 // Configurando puerto para el bluetooth, y activacion de los motores.

 Serial.begin(9600);

 pinMode(ActB,OUTPUT);  //Activacion del L298.

 pinMode(ActA,OUTPUT);  //Activacion del L298.

 //Configurando velocidades de rotacion de cada motor.

 Motor1.setSpeed(10);   //Receptor
 Motor2.setSpeed(10);   //Emisor
 Motor3.setSpeed(10);   //Receptor
 Motor4.setSpeed(10);   //Emisor
 delay(1000);

          }

void loop() {

 //Codigo principal a repetirse cada vez, esperando entrada de la interfaz.

  if (Serial.available()>0)

    Serial.flush();

    Opcion = Serial.read();

     switch(Opcion)    {

                              //LOS CASOS 1 Y 2 SON PARA MOVER EN 180°

       case '1':

         digitalWrite(ActA,HIGH);

         delay(100);

         Motor3.step(Pasos);

         digitalWrite(ActA,LOW);

         //Serial.write(1);   //Respondemos que ya terminamos de mover el motor A, RECEPTOR

         break;

         case '2':

         digitalWrite(ActB,HIGH);

         delay(100);

         Motor4.step(Pasos);

         digitalWrite(ActB,LOW);

         //Serial.write(2);   //Respondemos que ya terminamos de mover el motor B, EMISOR

         break;
```

```
case '5': //Retornar al motor 1 a la posicion del transductor RECEPTOR por defecto.
 digitalWrite(ActA,HIGH);
 Motor3.step(-172);
 digitalWrite(ActA,LOW);
 break;
 case 'R':          //Configurar en modo radial
 Pasos = 0;
 Flagpasos = 1;
 delay(500);
 Serial.flush();
       //Esperado datos: "Numero de Pasos"
 while(Flagpasos != 0)
  {
    if(Serial.available()>0)
    {
      Pasos = Serial.read();
      Pasos = 2*Pasos;   //Recordar que tiene multiplicar las rotaciones por 2, 2*Pasos
        if(Pasos >= 0)
        {
         Flagpasos = 0;
        }
     }

  }
 digitalWrite(ActA,HIGH);
 delay(100);
 Motor3.step(172);  //172 pasos Colocando en la posicion frente a frente a los transductores.
 digitalWrite(ActA,LOW);
 Serial.flush();
 Serial.write('X'); //Escribe en el puerto serie al PC que ya termino de recibir datos y configurar
 break;
```

```
//-------------------------------------------------------------------------------------------------------------------------------

                    //LOS CASOS 3 Y 4 SON PARA MOVERSE COMO RELOJ

case '3':

  digitalWrite(ActA,HIGH);

  delay(100);

  Motor1.step(Pasos);          //Movemos el RECEPTOR

  Pasos1++;

  digitalWrite(ActA,LOW);

  break;


case '4':

 digitalWrite(ActB,HIGH);

 delay(100);

 Motor2.step(Pasos);          //Movemos el EMISOR

 Pasos2++;

 digitalWrite(ActB,LOW);

 break;


case '6': //Retornar al motor 1 (n-1) posiciones. Este motor controla la posicion del transductor RECEPTOR.

 digitalWrite(ActA,HIGH);

 Retro1 = (Pasos1 - 1) * Pasos;

 Motor1.step(-Retro1-2);

 Pasos1 = 0;

 digitalWrite(ActA,LOW);

 break;
```

```
case 'A':          //Configuarar en modo abanico o reloj
  Pasos = 0;
  Flagpasos = 1;
  delay(500);
  Serial.flush();
        //Esperado datos: "Numero de Pasos"
 while(Flagpasos != 0)
   {
     if(Serial.available()>0)
     {
       Pasos = Serial.read();
       Pasos = 2*Pasos;   //Recordar que tiene multiplicar las rotaciones por 2, 2*Pasos
         if(Pasos >= 0)
         {
          Flagpasos = 0;
         }
      }
    }
  Serial.flush();
  Serial.write('Y'); //Escribe en el puerto serie al PC que ya termino de recibir datos y configurar
  break;
           } //Cierre del switch
}   //Cierre del loop
```

A2. Programa en Matlab, Interfaz Gráfica del Tomógrafo

A2.1 Prueba de Control del Motores

En esta sección se muestra el script y la imagen de la interfaz secundaria que fue desarrollada para probar el movimiento de los motores vía bluetooth o cable USB:

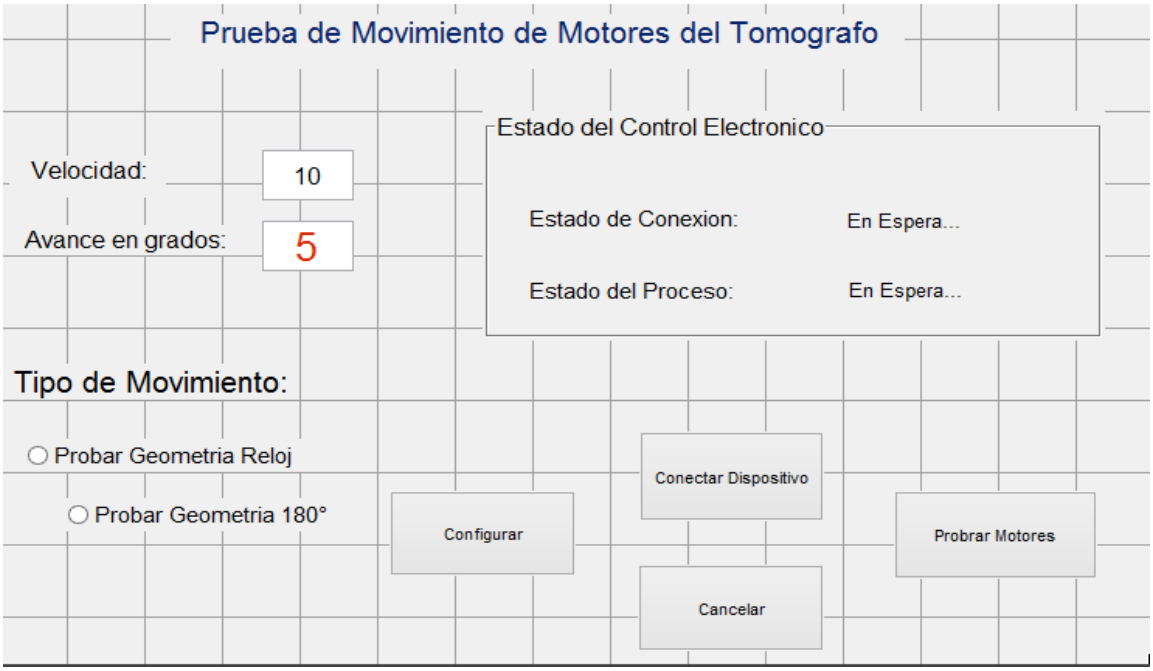


Figura. Interfaz secundaria que sirve para probar los motores, creada en Matlab.

El script para esta interfaz es el siguiente:

```
function varargout = Motores(varargin)
% MOTORES MATLAB code for Motores.fig
%      MOTORES, by itself, creates a new MOTORES or raises the existing
%      singleton*.
%
%      H = MOTORES returns the handle to a new MOTORES or the handle to
%      the existing singleton*.
%
%      MOTORES('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MOTORES.M with the given input arguments.
%
%      MOTORES('Property','Value',...) creates a new MOTORES or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Motores_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Motores_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
```

```
% Edit the above text to modify the response to help Motores

% Last Modified by GUIDE v2.5 27-Mar-2014 15:53:11

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Motores_OpeningFcn, ...
                   'gui_OutputFcn',  @Motores_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Motores is made visible.
function Motores_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Motores (see VARARGIN)

% Choose default command line output for Motores
handles.output = hObject;
global TablaIteracion;
global TablaGrados;
TablaGrados = 5:5:30;
TablaIteracion = [58, 29, 21, 16, 12, 10];
  set(handles.PopGrados,'String',TablaGrados);

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Motores wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Motores_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure

varargout{1} = handles.output;
```

```
% --- Executes on button press in RbGeoReloj.
function RbGeoReloj_Callback(hObject, eventdata, handles)
% hObject    handle to RbGeoReloj (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of RbGeoReloj
global OpcionGeometrica;
Estado = get(hObject,'Value');
 if (Estado == 1)
     set(handles.RbGeo180,'Value',0);
     set(handles.PopGrados,'Enable','on');
     set(handles.EdGrados,'Enable','off');
     OpcionGeometrica = 'Abanico';

 end

% --- Executes on button press in RbGeo180.
function RbGeo180_Callback(hObject, eventdata, handles)
% hObject    handle to RbGeo180 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of RbGeo180
global OpcionGeometrica;
Estado = get(hObject,'Value');

 if (Estado == 1)
     set(handles.RbGeoReloj,'Value',0);
     set(handles.PopGrados,'Enable','off');
     set(handles.EdGrados,'Enable','on');
     OpcionGeometrica = 'Radial';
 end


function EdVel_Callback(hObject, eventdata, handles)
% hObject    handle to EdVel (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of EdVel as text
%        str2double(get(hObject,'String')) returns contents of EdVel as a double


% --- Executes during object creation, after setting all properties.
function EdVel_CreateFcn(hObject, eventdata, handles)
% hObject    handle to EdVel (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function EdGrados_Callback(hObject, eventdata, handles)
% hObject    handle to EdGrados (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of EdGrados as text
%        str2double(get(hObject,'String')) returns contents of EdGrados as a
double


% --- Executes during object creation, after setting all properties.
function EdGrados_CreateFcn(hObject, eventdata, handles)
% hObject    handle to EdGrados (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in PbProbar.
function PbProbar_Callback(hObject, eventdata, handles)
% hObject    handle to PbProbar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Puerto;
global Solucion;
   global Iteracion;

Radial = 1;
Comando = ['1','2','3','4','5','6','A','R'];
GradosX = str2double(get(handles.EdGrados,'String'));
CantidadProyecciones = round(360/GradosX); %Numero de proyecciones en las dos
formas, segun emisor

switch Solucion
                  %%TOMOGRAFIA CON FORMA RADIAL

  case 'X'    %Se indica que se quiere explorar en forma "radial"
      while (Radial < CantidadProyecciones)
          pause(2);
          fwrite(Puerto,Comando(1,1),'char');  %Mover del RECEPTOR
          pause(3);
          fwrite(Puerto,Comando(1,2),'char');  %Mover del EMISOR
          pause(3);
          Radial = Radial + 1;
       end
      fwrite(Puerto,Comando(1,5),'char');  %Mover RECEPTOR A POSICION POR DEFECTO
      hold off;
      set(handles.TxtProceso,'String','Escaneo
Terminado..','ForegroundColor','b');
```

```
        %%TOMOGRAFIA CON FORMA ABANICO O RELOJ

  case 'Y'    %Se indica que se quiere explorar en forma "abanico"
    set(handles.TxtProceso,'String','Moviendo..','ForegroundColor','b');
    for Proy = 1:1 %CantidadProyecciones

      for Receptor = 1 : Iteracion
         pause(3);
         fwrite(Puerto,Comando(1,3),'char');  %Mover el RECEPTOR
         set(handles.EdProc,'String',Receptor);
         pause(5);
      end
      fwrite(Puerto,Comando(1,4),'char');  %Mover el EMISOR
      pause(3);
      fwrite(Puerto,Comando(1,6),'char');  %Mover el RECEPTOR, Npos - 1
      pause(10);
    end
   hold off;
   set(handles.TxtProceso,'String','Escaneo Terminado..','ForegroundColor','b');
end

% --- Executes on button press in PbConfigurar.
function PbConfigurar_Callback(hObject, eventdata, handles)
% hObject    handle to PbConfigurar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global OpcionGeometrica;
global Puerto;
global Pasos;
global Solucion;

switch(OpcionGeometrica)
 case 'Abanico'
  Solucion = ModoControl(Puerto,Pasos,OpcionGeometrica);
    if (Solucion == 'Y')
      set(handles.TxtProceso,'String','Configurado como
Abanido','ForegroundColor','b');
    end
 case 'Radial'
  GradosX = str2double(get(handles.EdGrados,'String'));
  PasosX = round(GradosX*200/360);
  Solucion = ModoControl(Puerto,PasosX,OpcionGeometrica);
    if (Solucion == 'X')
     set(handles.TxtProceso,'String','Configurado como
Radial','ForegroundColor','b');
    end
end
```

```
% --- Executes on button press in PbCancelar.
function PbCancelar_Callback(hObject, eventdata, handles)
% hObject    handle to PbCancelar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Puerto;
global Pabierto;
if(Pabierto == 1)
 fclose(Puerto);
 delete(Puerto);
 clear all;
 clc;
 close;
end
 close;

% --- Executes on button press in PbConectar.
function PbConectar_Callback(hObject, eventdata, handles)
% hObject    handle to PbConectar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Puerto;
global Pabierto;
Puerto = Bluetooth('HC-06',1);                  %Se crea el objeto bluetooth
fopen(Puerto);
Pabierto = strcmp(Puerto.Status,'open');        %Compara cadenas de caracteres.
  if(Pabierto == 1)
   set(handles.TxtConexion,'String','Equipo Conectado','ForegroundColor','b');
  else
   set(handles.TxtConexion,'String','Equipo No Conectado','ForegroundColor','r');
  end


% --- Executes during object creation, after setting all properties.
function RbGeoReloj_CreateFcn(hObject, eventdata, handles)
% hObject    handle to RbGeoReloj (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- Executes on selection change in PopGrados.
function PopGrados_Callback(hObject, eventdata, handles)
% hObject    handle to PopGrados (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns PopGrados contents as
cell array
%        contents{get(hObject,'Value')} returns selected item from PopGrados
global Iteracion;
global Pasos;
global TablaIteracion;
global TablaGrados;
  ContenidoPop = get(hObject,'Value');
  Grados = TablaGrados(1,ContenidoPop);
  Iteracion = TablaIteracion(ContenidoPop);
  Pasos = round(Grados*200/360);  %Obtencion de pasos por cada avance
```

```
% --- Executes during object creation, after setting all properties.
function PopGrados_CreateFcn(hObject, eventdata, handles)
% hObject    handle to PopGrados (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function EdProc_Callback(hObject, eventdata, handles)
% hObject    handle to EdProc (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of EdProc as text
%        str2double(get(hObject,'String')) returns contents of EdProc as a double


% --- Executes during object creation, after setting all properties.
function EdProc_CreateFcn(hObject, eventdata, handles)
% hObject    handle to EdProc (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

A2.2 Prueba del HS3-Sistema Digital de Adquisición del Ultrasonido

A continuación se mostrara una imagen de la interfaz secundaria que sirve para probar el sistema de adquisición de señales del tomógrafo y después el script que se desarrollo.

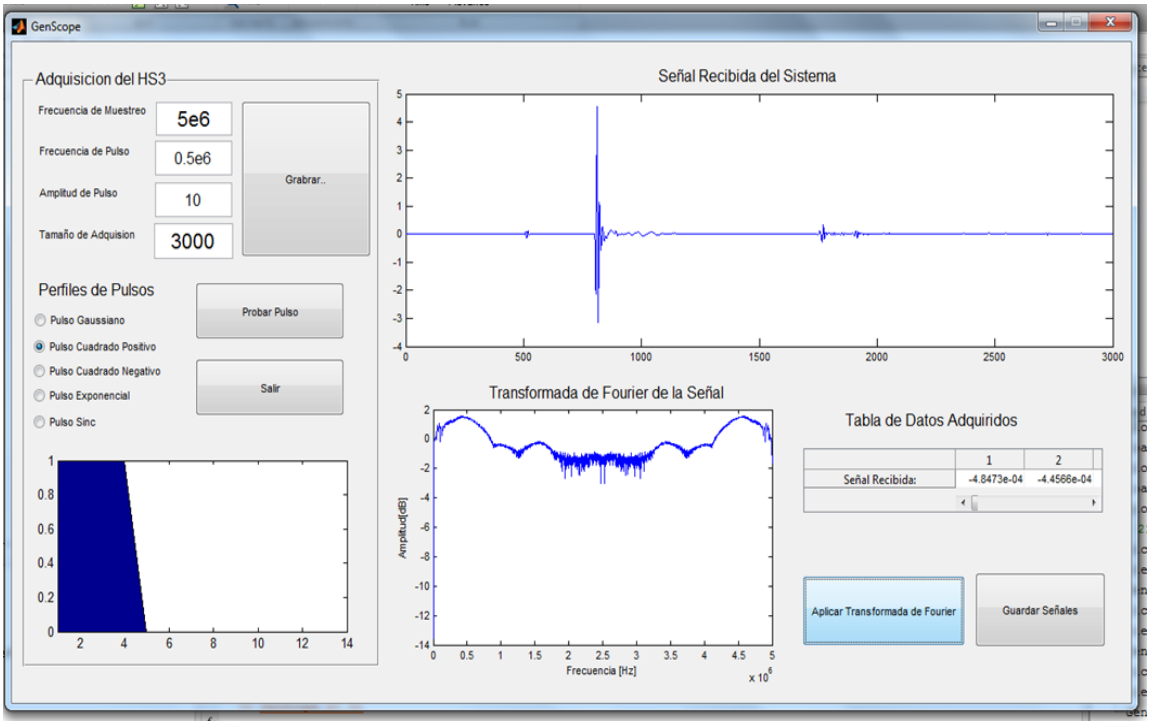


Figura. Módulo de la interfaz gráfica de control, permite probar el sistema de adquisición de señales ultrasónicas.

El script es el siguiente:

```
function varargout = GenScope(varargin)
% GENSCOPE MATLAB code for GenScope.fig
%      GENSCOPE, by itself, creates a new GENSCOPE or raises the existing
%      singleton*.
%
%      H = GENSCOPE returns the handle to a new GENSCOPE or the handle to
%      the existing singleton*.
%
%      GENSCOPE('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GENSCOPE.M with the given input arguments.
%
%      GENSCOPE('Property','Value',...) creates a new GENSCOPE or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before GenScope_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to GenScope_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
```

```
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GenScope

% Last Modified by GUIDE v2.5 03-Mar-2014 22:30:49

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GenScope_OpeningFcn, ...
                   'gui_OutputFcn',  @GenScope_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before GenScope is made visible.
function GenScope_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to GenScope (see VARARGIN)

% Choose default command line output for GenScope
handles.output = hObject;
%-----------------------Configuracion del objeto--------------------------
    global Osciloscopio    %Estructura OSCILOSCOPIO "Setup"
    global Generador;      %Estructura GENERADOR "SetGen"
    global Lib;
        Lib = 'hs3';
    Numdisparos = 100;          %Numero de disparos de la señal.
    Fm = str2double(get(handles.Edfm,'String'));    %Frecuencia de muestra
    Fsig = str2double(get(handles.Edfsig,'String'));%Frecuencia señal
    Amp = str2double(get(handles.Edamp,'String'));  %Amplitud de la señal
    Ladq = str2double(get(handles.Edtam,'String')); %Tamaño de adquisicion

    %Parametros por defecto del osciloscopio.
    Osciloscopio.TriggerSource = 0;
    Osciloscopio.Sensitivity   = 12;
    Osciloscopio.Samples       = Ladq;
    Osciloscopio.PostSamples   = 32;
    Osciloscopio.Fm            = Fm;
    Osciloscopio.Nmean         = Numdisparos;
    Osciloscopio.Resolution = 12;
```

```
 %Parametros por defecto del generador.
   Generador.Signal     = 5;        %Tipo de señal: 5, indica arbitrario
   Generador.Amplitude  = Amp;
   Generador.Fm         = Osciloscopio.Fm;
   Generador.Fsig       = Fsig;
   Generador.Trigger    = 9;
   %-------------------------------------------------
    ConstHS3 = InitHS(Lib); %Arranque del osciloscopio y generador.

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes GenScope wait for user response (see UIRESUME)
% uiwait(handles.GenScope);


% --- Outputs from this function are returned to the command line.
function varargout = GenScope_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function Edfm_Callback(hObject, eventdata, handles)
% hObject    handle to Edfm (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Edfm as text
%        str2double(get(hObject,'String')) returns contents of Edfm as a double


% --- Executes during object creation, after setting all properties.
function Edfm_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Edfm (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


function Edfsig_Callback(hObject, eventdata, handles)
% hObject    handle to Edfsig (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Edfsig as text
%        str2double(get(hObject,'String')) returns contents of Edfsig as a double
```

```
% --- Executes during object creation, after setting all properties.
function Edfsig_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Edfsig (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Edamp_Callback(hObject, eventdata, handles)
% hObject    handle to Edamp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Edamp as text
%        str2double(get(hObject,'String')) returns contents of Edamp as a double


% --- Executes during object creation, after setting all properties.
function Edamp_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Edamp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Edtam_Callback(hObject, eventdata, handles)
% hObject    handle to Edtam (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Edtam as text
%        str2double(get(hObject,'String')) returns contents of Edtam as a double


% --- Executes during object creation, after setting all properties.
function Edtam_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Edtam (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in Rbcuadp.
function Rbcuadp_Callback(hObject, eventdata, handles)
% hObject    handle to Rbcuadp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 Valor = get(hObject,'Value');
 if (Valor == 1)
    set(handles.Rbgauss,'Value',0);
    set(handles.Rbcuadn,'Value',0);
    set(handles.Rbexp,'Value',0);
    set(handles.Rbsinc,'Value',0);
   Fm = str2double(get(handles.Edfm,'String'));    %Frecuencia de muestra
   Fsig = str2double(get(handles.Edfsig,'String'));%Frecuencia señal
   n = 2000;               %Pre-longitud de la señal(solo para mostrar)

    %-----Generacion de la señal y muestra en pantalla
   Signal = pulsocuad(Fsig,Fm,n);
   area(handles.Axperfil,Signal);
 end
% Hint: get(hObject,'Value') returns toggle state of Rbcuadp


% --- Executes on button press in Pbgrabar.
function Pbgrabar_Callback(hObject, eventdata, handles)
% hObject    handle to Pbgrabar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Osciloscopio;
global Generador;
global Lib;

%---------------------Proceso de grabacion en memoria del HS3-------------
   Fm = str2double(get(handles.Edfm,'String'));    %Frecuencia de muestra
   Fsig = str2double(get(handles.Edfsig,'String'));%Frecuencia señal
   Amp = str2double(get(handles.Edamp,'String'));  %Amplitud de la señal
   Ladq = str2double(get(handles.Edtam,'String')); %Tamaño de adquisicion
   Osciloscopio.Fm = Fm;
   Osciloscopio.Samples = Ladq;
   Generador.Fsig = Fsig;
   Generador.Amplitude = Amp;

  if(Osciloscopio.Fm <= 50e6)
    Osciloscopio.Resolution = 12;
  else
    Osciloscopio.Resolution = 14;
  end

  ConfigHS(Lib,Osciloscopio); %Se configura el HS3 la parte del osciloscopio.
%-----------------Se graban en memoria las funciones------------------
  if (get(handles.Rbgauss,'Value') == 1)
     Op = 1;                        %Pulso gaussiano
     ConfigGeneratorHS3(Lib,Generador,Op);
  end

  if (get(handles.Rbcuadp,'Value') == 1)
     Op = 2;                         %Pulso cuadrado positivo
     ConfigGeneratorHS3(Lib,Generador,Op);
  end
```

```
 if (get(handles.Rbcuadn,'Value') == 1)
    Op = 3;                     %Pulso cuadrado negativo
    ConfigGeneratorHS3(Lib,Generador,Op);
 end

 if (get(handles.Rbexp,'Value') == 1)
    Op = 4;                     %Pulso exponencial
    ConfigGeneratorHS3(Lib,Generador,Op);
 end

 if (get(handles.Rbsinc,'Value') == 1)
    Op = 5;                     %Pulso sinc
    ConfigGeneratorHS3(Lib,Generador,Op);
 end

% --- Executes on button press in Rbexp.
function Rbexp_Callback(hObject, eventdata, handles)
% hObject    handle to Rbexp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Valor = get(hObject,'Value');
 if (Valor == 1)
    set(handles.Rbgauss,'Value',0);
    set(handles.Rbcuadn,'Value',0);
    set(handles.Rbcuadp,'Value',0);
    set(handles.Rbsinc,'Value',0);
    Fm = str2double(get(handles.Edfm,'String'));    %Frecuencia de muestra
    Fsig = str2double(get(handles.Edfsig,'String'));%Frecuencia señal
    n = 2000;               %Pre-longitud de la señal(solo para mostrar)

    %-----Generacion de la señal y muestra en pantalla
   Signal = pulsoexpo(Fsig,Fm,n);
   area(handles.Axperfil,Signal);
 end
% Hint: get(hObject,'Value') returns toggle state of Rbexp


% --- Executes on button press in Rbsinc.
function Rbsinc_Callback(hObject, eventdata, handles)
% hObject    handle to Rbsinc (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Valor = get(hObject,'Value');
 if (Valor == 1)
    set(handles.Rbgauss,'Value',0);
    set(handles.Rbcuadn,'Value',0);
    set(handles.Rbcuadp,'Value',0);
    set(handles.Rbexp,'Value',0);
   Fm = str2double(get(handles.Edfm,'String'));    %Frecuencia de muestra
   Fsig = str2double(get(handles.Edfsig,'String'));%Frecuencia señal
   n = 2000;               %Pre-longitud de la señal(solo para mostrar)

    %-----Generacion de la señal y muestra en pantalla
   Signal = pulsosinc(Fsig,Fm,n);
   area(handles.Axperfil,Signal);
 end
% Hint: get(hObject,'Value') returns toggle state of Rbsinc
```

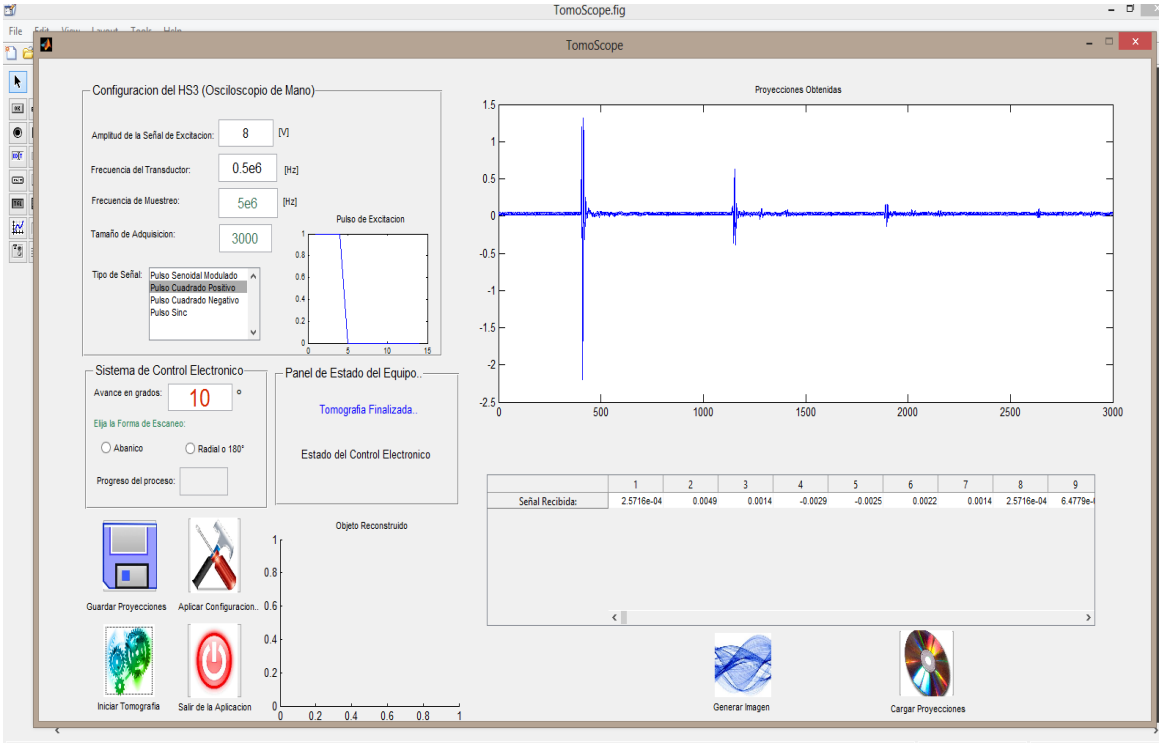
```
% --- Executes on button press in Rbgauss.
function Rbgauss_Callback(hObject, eventdata, handles)
% hObject    handle to Rbgauss (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 Valor = get(hObject,'Value');
 if (Valor == 1)
    set(handles.Rbcuadp,'Value',0);
    set(handles.Rbcuadn,'Value',0);
    set(handles.Rbexp,'Value',0);
    set(handles.Rbsinc,'Value',0);
    %-------
   Fm = str2double(get(handles.Edfm,'String'));    %Frecuencia de muestra
   Fsig = str2double(get(handles.Edfsig,'String'));%Frecuencia señal
   n = 2000;               %Pre-longitud de la señal(solo para mostrar)

    %-----Generacion de la señal y muestra en pantalla
   Signal = pulsosin(Fsig,Fm,n);
   area(handles.Axperfil,Signal);
 end
% Hint: get(hObject,'Value') returns toggle state of Rbgauss


% --- Executes on button press in Rbcuadn.
function Rbcuadn_Callback(hObject, eventdata, handles)
% hObject    handle to Rbcuadn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Valor = get(hObject,'Value');
 if (Valor == 1)
    set(handles.Rbgauss,'Value',0);
    set(handles.Rbcuadp,'Value',0);
    set(handles.Rbexp,'Value',0);
    set(handles.Rbsinc,'Value',0);
    Fm = str2double(get(handles.Edfm,'String'));    %Frecuencia de muestra
    Fsig = str2double(get(handles.Edfsig,'String'));%Frecuencia señal
    n = 2000;               %Pre-longitud de la señal(solo para mostrar)

    %-----Generacion de la señal y muestra en pantalla
   Signal = pulsocuadn(Fsig,Fm,n);
   area(handles.Axperfil,Signal);
 end
% Hint: get(hObject,'Value') returns toggle state of Rbcuadn


% --- Executes on button press in Pbguardar.
function Pbguardar_Callback(hObject, eventdata, handles)
% hObject    handle to Pbguardar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%%--------------Se prepara para guardar señales adquiridas-----------------
global DSenal;
[ArchivoSal,~,Filtro] = uiputfile({'*.mat'},'Guardar Puntos de la Señal Como:');
if (Filtro == 1)
   save(ArchivoSal,'DSenal','-mat');
end
```

```
% --- Executes on button press in Pbpulso.
function Pbpulso_Callback(hObject, eventdata, handles)
% hObject    handle to Pbpulso (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Lib;
   Lib = 'hs3';
global Osciloscopio;
global DSenal;
%%%%---------Preconfigurando la adquisicion del ultrasonido----------
  data1  = double(1:Osciloscopio.Samples);
  data2  = double(1:Osciloscopio.Samples );

  pData1 = libpointer( 'doublePtr' , data1 );
  pData2 = libpointer( 'doublePtr' , data2 );

  dat1  = double( 1:Osciloscopio.Samples );
  dat2  = double( 1:Osciloscopio.Samples );
  indice = 0;
%----------------Adquisicion del ultrasonido----------------------------
 for Capturas = 1:1
       dat1(:)=0;
       dat2(:)=0;
       calllib(Lib, 'SetFuncGenEnable',1);
       pause(0.01);
       for i = 1: Osciloscopio.Nmean  %Indica el numero de disparos
        calllib(Lib, 'SetFuncGenOutputOn',1);
        [data1,data2] = AdquirirHS(Lib,pData1,pData2);
        pause(0.01);
        calllib(Lib, 'SetFuncGenOutputOn',0);
        dat1 = dat1 + data1;
        dat2 = dat2 + data2;
       end
       calllib(Lib, 'SetFuncGenEnable',0);
       dat1 = dat1/Osciloscopio.Nmean;
       dat1 = dat1-mean(dat1);
       dat2 = dat2/Osciloscopio.Nmean;
       dat2 = dat2-mean(dat2);
       DSenal.dat1=dat1;
       DSenal.dat2=dat2;
       indice = indice +0.015;
       plot(handles.Axsignal,(DSenal.dat2)+indice); hold on;
 end
      hold off;
      %---------Mostrando datos de la señal en tablas----------------
      arreglo_elementos = cell(1,length(DSenal.dat2));
      arreglo_elementos(1,:) = num2cell(DSenal.dat2);
      set(handles.Tabla,'Data',arreglo_elementos);
      set(handles.Tabla,'RowName',{'Señal Recibida:'});
```

```
% --- Executes on button press in Pbsalir.
function Pbsalir_Callback(hObject, eventdata, handles)
% hObject    handle to Pbsalir (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Lib;
     Lib = 'hs3';
if ~libisloaded(Lib)
   error([ 'No se cargo la libreria: ' , Lib ]);
 else
   calllib(Lib,'ExitInstrument');
   unloadlibrary 'hs3';
   disp('Libreria e Instrumento Liberado..');
end
close;


% --- Executes on button press in Pbfft.
function Pbfft_Callback(hObject, eventdata, handles)
% hObject    handle to Pbfft (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Osciloscopio;
global DSenal;
N = length(DSenal.dat2);
n = 1:N;
n(1:end) = n(1:end)*(Osciloscopio.Fm/N);
Xfft_Magnitud = log10(abs(fft(DSenal.dat2)));
plot(handles.Axfft,n,Xfft_Magnitud);
ylabel('Amplitud[dB]'); xlabel('Frecuencia [Hz]');
```

A2.3 Sistema de Tomografía Completo

El sistema de tomografía completo se muestra a continuación:



El script que precede a la interfaz se muestra a continuación:

```
function varargout = GenScope(varargin)
% GENSCOPE MATLAB code for GenScope.fig
%      GENSCOPE, by itself, creates a new GENSCOPE or raises the existing
%      singleton*.
%
%      H = GENSCOPE returns the handle to a new GENSCOPE or the handle to
%      the existing singleton*.
%
%      GENSCOPE('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GENSCOPE.M with the given input arguments.
%
%      GENSCOPE('Property','Value',...) creates a new GENSCOPE or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before GenScope_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to GenScope_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GenScope

% Last Modified by GUIDE v2.5 03-Mar-2014 22:30:49
```

```
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',      mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GenScope_OpeningFcn, ...
                   'gui_OutputFcn',  @GenScope_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

 % --- Executes just before GenScope is made visible.
function GenScope_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to GenScope (see VARARGIN)

% Choose default command line output for GenScope
handles.output = hObject;
%-----------------------Configuracion del objeto--------------------------
   global Osciloscopio    %Estructura OSCILOSCOPIO "Setup"
   global Generador;      %Estructura GENERADOR "SetGen"
   global Lib;
        Lib = 'hs3';
   Numdisparos = 100;          %Numero de disparos de la señal.
   Fm = str2double(get(handles.Edfm,'String'));    %Frecuencia de muestra
   Fsig = str2double(get(handles.Edfsig,'String'));%Frecuencia señal
   Amp = str2double(get(handles.Edamp,'String'));  %Amplitud de la señal
   Ladq = str2double(get(handles.Edtam,'String')); %Tamaño de adquisicion

   %Parametros por defecto del osciloscopio.
   Osciloscopio.TriggerSource = 0;
   Osciloscopio.Sensitivity   = 12;
   Osciloscopio.Samples       = Ladq;
   Osciloscopio.PostSamples   = 32;
   Osciloscopio.Fm            = Fm;
   Osciloscopio.Nmean         = Numdisparos;
   Osciloscopio.Resolution = 12;

   %Parametros por defecto del generador.
   Generador.Signal     = 5;        %Tipo de señal: 5, indica arbitrario
   Generador.Amplitude  = Amp;
   Generador.Fm         = Osciloscopio.Fm;
   Generador.Fsig       = Fsig;
   Generador.Trigger    = 9;
   %-------------------------------------------------
    ConstHS3 = InitHS(Lib); %Arranque del osciloscopio y generador.

% Update handles structure
guidata(hObject, handles);
```

```
% UIWAIT makes GenScope wait for user response (see UIRESUME)
% uiwait(handles.GenScope);


% --- Outputs from this function are returned to the command line.
function varargout = GenScope_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


 function Edfm_Callback(hObject, eventdata, handles)
% hObject    handle to Edfm (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Edfm as text
%        str2double(get(hObject,'String')) returns contents of Edfm as a double


% --- Executes during object creation, after setting all properties.
function Edfm_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Edfm (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Edfsig_Callback(hObject, eventdata, handles)
% hObject    handle to Edfsig (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Edfsig as text
%        str2double(get(hObject,'String')) returns contents of Edfsig as a double



% --- Executes during object creation, after setting all properties.
function Edfsig_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Edfsig (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function Edamp_Callback(hObject, eventdata, handles)
% hObject    handle to Edamp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Edamp as text
%        str2double(get(hObject,'String')) returns contents of Edamp as a double


% --- Executes during object creation, after setting all properties.
function Edamp_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Edamp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Edtam_Callback(hObject, eventdata, handles)
% hObject    handle to Edtam (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Edtam as text
%        str2double(get(hObject,'String')) returns contents of Edtam as a double


% --- Executes during object creation, after setting all properties.
function Edtam_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Edtam (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in Rbcuadp.
function Rbcuadp_Callback(hObject, eventdata, handles)
% hObject    handle to Rbcuadp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 Valor = get(hObject,'Value');
 if (Valor == 1)
    set(handles.Rbgauss,'Value',0);
    set(handles.Rbcuadn,'Value',0);
    set(handles.Rbexp,'Value',0);
    set(handles.Rbsinc,'Value',0);
   Fm = str2double(get(handles.Edfm,'String'));    %Frecuencia de muestra
```

```
    Fsig = str2double(get(handles.Edfsig,'String'));%Frecuencia señal
    n = 2000;               %Pre-longitud de la señal(solo para mostrar)

     %-----Generacion de la señal y muestra en pantalla
    Signal = pulsocuad(Fsig,Fm,n);
    area(handles.Axperfil,Signal);
 end
% Hint: get(hObject,'Value') returns toggle state of Rbcuadp

 % --- Executes on button press in Pbgrabar.
function Pbgrabar_Callback(hObject, eventdata, handles)
% hObject    handle to Pbgrabar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Osciloscopio;
global Generador;
global Lib;

%----------------------Proceso de grabacion en memoria del HS3-------------
    Fm = str2double(get(handles.Edfm,'String'));    %Frecuencia de muestra
    Fsig = str2double(get(handles.Edfsig,'String'));%Frecuencia señal
    Amp = str2double(get(handles.Edamp,'String'));  %Amplitud de la señal
    Ladq = str2double(get(handles.Edtam,'String')); %Tamaño de adquisicion
    Osciloscopio.Fm = Fm;
    Osciloscopio.Samples = Ladq;
    Generador.Fsig = Fsig;
    Generador.Amplitude = Amp;

   if(Osciloscopio.Fm <= 50e6)
     Osciloscopio.Resolution = 12;
   else
     Osciloscopio.Resolution = 14;
   end

   ConfigHS(Lib,Osciloscopio); %Se configura el HS3 la parte del osciloscopio.
%-----------------Se graban en memoria las funciones-------------------
   if (get(handles.Rbgauss,'Value') == 1)
      Op = 1;                       %Pulso gaussiano
      ConfigGeneratorHS3(Lib,Generador,Op);
   end

   if (get(handles.Rbcuadp,'Value') == 1)
      Op = 2;                       %Pulso cuadrado positivo
      ConfigGeneratorHS3(Lib,Generador,Op);
   end

  if (get(handles.Rbcuadn,'Value') == 1)
      Op = 3;                       %Pulso cuadrado negativo
      ConfigGeneratorHS3(Lib,Generador,Op);
  end

  if (get(handles.Rbexp,'Value') == 1)
      Op = 4;                       %Pulso exponencial
      ConfigGeneratorHS3(Lib,Generador,Op);
  end

  if (get(handles.Rbsinc,'Value') == 1)
      Op = 5;                       %Pulso sinc
      ConfigGeneratorHS3(Lib,Generador,Op);
  end
```

```
% --- Executes on button press in Rbexp.
function Rbexp_Callback(hObject, eventdata, handles)
% hObject    handle to Rbexp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Valor = get(hObject,'Value');
 if (Valor == 1)
    set(handles.Rbgauss,'Value',0);
    set(handles.Rbcuadn,'Value',0);
    set(handles.Rbcuadp,'Value',0);
    set(handles.Rbsinc,'Value',0);
    Fm = str2double(get(handles.Edfm,'String'));    %Frecuencia de muestra
    Fsig = str2double(get(handles.Edfsig,'String'));%Frecuencia señal
    n = 2000;               %Pre-longitud de la señal(solo para mostrar)

    %-----Generacion de la señal y muestra en pantalla
   Signal = pulsoexpo(Fsig,Fm,n);
   area(handles.Axperfil,Signal);
 end
% Hint: get(hObject,'Value') returns toggle state of Rbexp

% --- Executes on button press in Rbsinc.
function Rbsinc_Callback(hObject, eventdata, handles)
% hObject    handle to Rbsinc (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Valor = get(hObject,'Value');
 if (Valor == 1)
    set(handles.Rbgauss,'Value',0);
    set(handles.Rbcuadn,'Value',0);
    set(handles.Rbcuadp,'Value',0);
    set(handles.Rbexp,'Value',0);
   Fm = str2double(get(handles.Edfm,'String'));    %Frecuencia de muestra
   Fsig = str2double(get(handles.Edfsig,'String'));%Frecuencia señal
   n = 2000;               %Pre-longitud de la señal(solo para mostrar)

    %-----Generacion de la señal y muestra en pantalla
   Signal = pulsosinc(Fsig,Fm,n);
   area(handles.Axperfil,Signal);
 end
% Hint: get(hObject,'Value') returns toggle state of Rbsinc

% --- Executes on button press in Rbgauss.
function Rbgauss_Callback(hObject, eventdata, handles)
% hObject    handle to Rbgauss (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 Valor = get(hObject,'Value');
 if (Valor == 1)
    set(handles.Rbcuadp,'Value',0);
    set(handles.Rbcuadn,'Value',0);
    set(handles.Rbexp,'Value',0);
    set(handles.Rbsinc,'Value',0);
    %-------
   Fm = str2double(get(handles.Edfm,'String'));    %Frecuencia de muestra
   Fsig = str2double(get(handles.Edfsig,'String'));%Frecuencia señal
   n = 2000;               %Pre-longitud de la señal(solo para mostrar)
```

```
    %-----Generacion de la señal y muestra en pantalla
   Signal = pulsosin(Fsig,Fm,n);
   area(handles.Axperfil,Signal);
 end
% Hint: get(hObject,'Value') returns toggle state of Rbgauss


% --- Executes on button press in Rbcuadn.
function Rbcuadn_Callback(hObject, eventdata, handles)
% hObject    handle to Rbcuadn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Valor = get(hObject,'Value');
 if (Valor == 1)
    set(handles.Rbgauss,'Value',0);
    set(handles.Rbcuadp,'Value',0);
    set(handles.Rbexp,'Value',0);
    set(handles.Rbsinc,'Value',0);
    Fm = str2double(get(handles.Edfm,'String'));    %Frecuencia de muestra
    Fsig = str2double(get(handles.Edfsig,'String'));%Frecuencia señal
    n = 2000;               %Pre-longitud de la señal(solo para mostrar)

    %-----Generacion de la señal y muestra en pantalla
   Signal = pulsocuadn(Fsig,Fm,n);
   area(handles.Axperfil,Signal);
 end
% Hint: get(hObject,'Value') returns toggle state of Rbcuadn


% --- Executes on button press in Pbguardar.
function Pbguardar_Callback(hObject, eventdata, handles)
% hObject    handle to Pbguardar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%%--------------Se prepara para guardar señales adquiridas-----------------
global DSenal;
[ArchivoSal,~,Filtro] = uiputfile({'*.mat'},'Guardar Puntos de la Señal Como:');
if (Filtro == 1)
   save(ArchivoSal,'DSenal','-mat');
end
% --- Executes on button press in Pbpulso.
function Pbpulso_Callback(hObject, eventdata, handles)
% hObject    handle to Pbpulso (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Lib;
   Lib = 'hs3';
global Osciloscopio;
global DSenal;
%%%%---------Preconfigurando la adquisicion del ultrasonido----------
  data1  = double(1:Osciloscopio.Samples);
  data2  = double(1:Osciloscopio.Samples );

  pData1 = libpointer( 'doublePtr' , data1 );
  pData2 = libpointer( 'doublePtr' , data2 );

  dat1  = double( 1:Osciloscopio.Samples );
  dat2  = double( 1:Osciloscopio.Samples );
  indice = 0;
```

```
%----------------Adquisicion del ultrasonido----------------------------
 for Capturas = 1:1
       dat1(:)=0;
       dat2(:)=0;
       calllib(Lib, 'SetFuncGenEnable',1);
       pause(0.01);
       for i = 1: Osciloscopio.Nmean  %Indica el numero de disparos
        calllib(Lib, 'SetFuncGenOutputOn',1);
        [data1,data2] = AdquirirHS(Lib,pData1,pData2);
        pause(0.01);
        calllib(Lib, 'SetFuncGenOutputOn',0);
        dat1 = dat1 + data1;
        dat2 = dat2 + data2;
       end
       calllib(Lib, 'SetFuncGenEnable',0);
       dat1 = dat1/Osciloscopio.Nmean;
       dat1 = dat1-mean(dat1);
       dat2 = dat2/Osciloscopio.Nmean;
       dat2 = dat2-mean(dat2);
       DSenal.dat1=dat1;
       DSenal.dat2=dat2;
       indice = indice +0.015;
       plot(handles.Axsignal,(DSenal.dat2)+indice); hold on;
 end
      hold off;
      %---------Mostrando datos de la señal en tablas-----------------
      arreglo_elementos = cell(1,length(DSenal.dat2));
      arreglo_elementos(1,:) = num2cell(DSenal.dat2);
      set(handles.Tabla,'Data',arreglo_elementos);
      set(handles.Tabla,'RowName',{'Señal Recibida:'});

% --- Executes on button press in Pbsalir.
function Pbsalir_Callback(hObject, eventdata, handles)
% hObject    handle to Pbsalir (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Lib;
     Lib = 'hs3';
if ~libisloaded(Lib)
   error([ 'No se cargo la libreria: ' , Lib ]);
 else
   calllib(Lib,'ExitInstrument');
   unloadlibrary 'hs3';
   disp('Libreria e Instrumento Liberado..');
end
close;

% --- Executes on button press in Pbfft.
function Pbfft_Callback(hObject, eventdata, handles)
% hObject    handle to Pbfft (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Osciloscopio;
global DSenal;
N = length(DSenal.dat2);
n = 1:N;
n(1:end) = n(1:end)*(Osciloscopio.Fm/N);
Xfft_Magnitud = log10(abs(fft(DSenal.dat2)));
plot(handles.Axfft,n,Xfft_Magnitud);
ylabel('Amplitud[dB]'); xlabel('Frecuencia [Hz]');
```

A3. Tabla de Propiedades Acústicas de Algunos Materiales.

Material	Velocidad Longitudinal		Velocidad Superficial		Impedancia Acústica
	(in/µs)	*(m/s)*	*(in/µs)*	*(m/s)*	*(Kg/m^{2}s x 10^6)*
Resina Acrílico	0.107	2,730	0.056	1,430	3.22
Aluminio	0.249	6,320	0.123	3,130	17.06
Cadmio	0.109	2,780	0.059	1,500	24.02
Cobre	0.183	4,660	0.089	2,260	41.61
Oro	0.128	3,240	0.047	1,200	62.60
Acero	0.232	5,900	0.127	3,230	45.43
Mercurio	0.057	1,450	-	-	19.66
Níquel Puro	0.222	5,630	0.117	2,960	49.99
Plata	0.142	3,600	0.063	1,590	37.76
Agua a 20°C	0.058	1,480	-	-	1.48
Aceite SAE 20	0.069	1,740	-	-	1.51

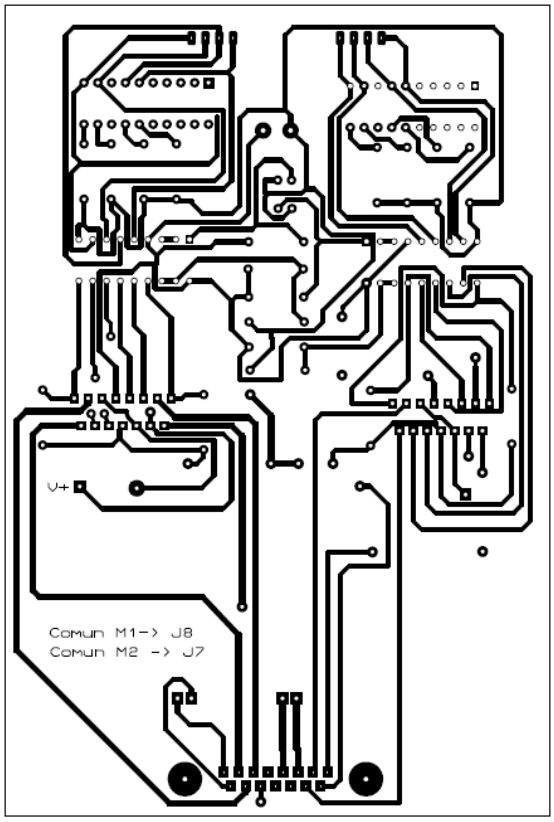
Fuente: Notas técnicas de transductores de ultrasonido de Olympus.

A4. Campo Cercano de Transductores Planos para Inmersión en Agua de Olympus.

Frecuencia	Diámetro del Transductor	Campo Cercano "N"	Distancia Focal PTF	
MHz	*in*	*in*	*Min in*	*Max in*
0.5	1.50	4.757	2.15	3.80
	1.125	2.661	1.50	2.10
	1.00	2.095	1.25	1.65
	0.75	1.164	0.78	0.93
1.0	1.50	9.559	2.50	7.65
	1.125	5.366	1.90	4.30
	1.00	4.235	1.625	3.38
	0.75	2.37	1.00	1.90
	0.50	1.043	0.60	0.80

Fuente: Notas técnicas de transductores de ultrasonido de Olympus

A5. Circuito Impreso del Control Electrónico de Motores a Paso a Paso



Circuito impreso del sistema de control de motores a paso.