CARACTERIZACIÓN DEL DESPLAZAMIENTO DE UN DISCO PIEZOELÉCTRICO EN FUNCIÓN DEL VOLTAJE CON CORRECCIÓN DE HISTÉRESIS Y ARRASTRE MEDIANTE UN CONTROL EN LAZO CERRADO, UTILIZANDO INTERFEROMETRÍA.

JOSE EDUARDO RUIZ ROSERO

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
SAN JUAN DE PASTO
2013

CARACTERIZACIÓN DEL DESPLAZAMIENTO DE UN DISCO PIEZOELÉCTRICO EN FUNCIÓN DEL VOLTAJE CON CORRECCIÓN DE HISTÉRESIS Y ARRASTRE MEDIANTE UN CONTROL EN LAZO CERRADO, UTILIZANDO INTERFEROMETRÍA.

JOSE EDUARDO RUIZ ROSERO

Trabajo de grado presentado como requisito parcial para optar al título de Ingeniero Electrónico

DIRECTOR
ING. M.SC. DARÍO FERNANDO FAJARDO

ASESOR
ING. M.SC. JUAN PABLO RUIZ ROSERO

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
SAN JUAN DE PASTO
2013

“Las ideas y conclusiones aportadas en la tesis de grado son responsabilidad exclusiva de los autores"

Artículo 1. Del acuerdo No. 324 del 11 de Octubre de 1966, emanado por el Honorable Consejo Directivo de la Universidad de Nariño.

Nota de aceptación.

_____
_____
_____
_____
_____

Firma del presidente del jurado

Firma del jurado

Firma del jurado

San Juan de Pasto, Febrero 19 de 2013

A Dios por darme la sabiduría, las herramientas necesarias y ser el guía en todos los caminos de mi vida.

A mi madre Mireya Rosero que siempre me ha apoyado en todos mis proyectos de vida.

A todos mis familiares y amigos.

AGRADECIMIENTOS

A mi primo Ing. M.Sc Juan Pablo Ruiz Rosero por ser el asesor y apoyo fundamental de este trabajo de grado, por su dedicatoria, enseñanzas, paciencia e interés en toda mi carrera universitaria.

Al Ing. M.Sc Darío Fernando Fajardo director de este trabajo de grado, director del programa de electrónica y director del grupo de investigación de instrumentación y sistemas inteligentes, por su confianza, enseñanzas, apoyo y orientación.

A la Dr. Alba Ávila por su interés y apoyo para el desarrollo de una etapa importante de este proyecto.

Al Dr. Javier Revelo por su orientación e interés en etapas fundamentales durante el desarrollo de esta investigación.

A la VIPRI (Vicerrectoría de Investigaciones, Postgrados y Relaciones Internacionales) por el financiamiento de este trabajo de grado a través de sus convocatorias.

A la Universidad de los Andes por abrirme las puertas de sus instalaciones y laboratorios para el desarrollo de una etapa importante de mi investigación.

A la Universidad de Nariño por todo lo que he aprendido aquí, lugar donde viví muchos de mis logros y grandes momentos de mi vida.

A todos mis amigos de la carrera que con su compañía, apoyo y amistad hicieron de esta etapa de mi vida una vivencia inolvidable.

CONTENIDO

pág.

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ANEXOS

GLOSARIO

ADC (analog to digital converter) conversión analógica a digital o digitalización

AFM (atomic force microscopes) microscopio de fuerza atómica

ARRASTRE (drift) variación de desplazamiento que sufre el disco piezoeléctrico debido a cambios de temperatura y/o al efecto de arrastre en el material piezoeléctrico

DISCO PIEZOELÉCTRICO o zumbador piezoeléctrico son dispositivos formados por dos placas muy finas de distintos metales, o en algunos casos, de una placa de metal sobre una de cerámica, de tal forma que al recibir una presión emiten una corriente eléctrica o al recibir una señal eléctrica estos alteran su forma

DSPIC tipo de controlador de microchip especialmente diseñado para el tratamiento de señales digitales

HISTÉRESIS tendencia de un material a conservar una de sus propiedades en ausencia del estimulo que la ha generado

INTERFERÓMETRO instrumento que emplea la interferencia de las ondas de luz para medir con gran precisión longitudes de onda de la luz misma o pequeños desplazamientos

LED (light emitting diode) diodo emisor de luz, es un diodo semiconductor que emite luz

MAGNETRÓN dispositivo que transforma la energía eléctrica en energía electromagnética en forma de microonda

PIC familia de microcontroladores fabricados por Microchip

PLETINA placas de metal planas u hojas rectangulares de acero u otro metal

RS-232 es una interfaz que designa una norma para el intercambio serie de datos binarios entre un DTE (Equipo terminal de datos) y un DCE (Data Communication Equipment, Equipo de Comunicación de datos), aunque existen otras situaciones en las que también se utiliza la interfaz RS-232.

USB (universal serial bus), puerto que sirve para conectar periféricos en caliente a una computadora.

RESUMEN

En este proyecto se realiza el control PI en lazo cerrado del desplazamiento de un disco piezoeléctrico para efectuar correcciones de histéresis y arrastre. Se implementa un interferómetro de Michelson que permite realizar mediciones de desplazamiento a micro y nano escalas; y se crea una interfaz de usuario que establece la comunicación entre el dispositivo y el PC para la adquisición de datos y posterior análisis de estos. Con el fin de obtener una medida de la confiabilidad de los datos obtenidos por el interferómetro construido, junto con la eficiencia del controlador PI, se procede a caracterizar el instrumento con el microscopio de fuerza atómica AFM Asylum Ressearch MFP-3D-BIO del laboratorio de microscopía de la Universidad de los Andes en Bogotá Colombia. El sistema final tiene la capacidad de desplazar el disco piezoeléctrico una distancia determinada por el usuario con una resolución de 1nm y un rango de +/- 1950 nm. Se describen en esta investigación los bloques que conforman el diseño final, las pruebas realizadas para la medición de los desplazamientos, los resultados obtenidos y las conclusiones.

ABSTRACT

This project develops closed loop PI control the displacement of a piezoelectric disk hysteresis and drift corrections. It implements a Michelson interferometer that allows measurements of micro and nano displacement scales, and create a user interface that provides communication between the device and the PC for data acquisition and subsequent analysis of these. In order to obtain a measure of the reliability of the data obtained by the interferometer built, along with the efficiency of the PI controller, we proceed to characterize the instrument with the atomic force microscope AFM Ressearch Asylum MFP-3D-BIO Lab microscopy at the University of the Andes in Bogota Colombia. The final system is capable of displacing the piezoelectric disc a distance determined by the user with a resolution of 1 nm and a range of + / - 1950 nm. Are described in this investigation the building blocks of the final design, tests for measurement of displacement, the results and conclusions.

INTRODUCCIÓN

Con el pasar del tiempo la tecnología avanza y sus descubrimientos son muy sorprendentes y los dispositivos electrónicos tienden a ser cada vez más pequeños. También se encuentra el problema de no poder observar imágenes demasiadas pequeñas (menores a 200nm) con los microscopios ópticos; por eso existe la necesidad de encontrar otros medios para poder lograr el objetivo de observar estas diminutas imágenes; además de poder manipular cosas muy pequeñas teniendo una gran exactitud. Pero las herramientas necesarias para trabajar a estas escalas son poco comunes y muy costosas.

Estas herramientas necesitan de una gran exactitud, ya que al trabajar a pequeñas escalas, sus movimientos deben de ser muy precisos y delicados; esto es imposible de lograr con motores o actuadores mecánicos basados en campos magnéticos. Una solución es trabajar pequeñas estructuras que son movidas a escalas nanométricas con discos piezoeléctricos, que se expanden o contraen pequeñas cantidades dependiendo del voltaje que se les aplique; siendo así muy importante caracterizar y controlar su desplazamiento.

JUSTIFICACIÓN

Una herramienta fundamental para el estudio, desarrollo e investigación a nano escalas es el microscopio de fuerza atómica (AFM por sus siglas en inglés *Atomic Force Microscope*)[1]; pero los AFM que no cuentan con un sistema de control para

[1] G. Binnig, C.F. Quate y Ch. Gerber. Atomic Force Microscope. Physical review letters. 3 Marzo 1986. vol. 56, no. 9 [citado en 2012 12 31], pp 930-933

los actuadores piezoeléctricos, generan imágenes deformadas debido a la histéresis y el arrastre[1] como se presenta en la Figura 1.

Figura 1. Imagen tomada sin corrección de linealidad de una rejilla de calibración

Fleming, Andrew J. y Leang, Kam K. Evaluation of Charge Drives for Scanning Probe Microscope Positioning Stages. En: American Control Conference (11-13, Junio, 2008). Westin Seattle Hotel, Seattle, Washington, USA

La implementación de un controlador de posición en lazo cerrado permite corregir estos problemas, pero para realizar esto se debe obtener antes la medición del desplazamiento en función del voltaje del disco piezoeléctrico, esta medición se realiza a través de sensores de desplazamiento láser como procede Wnag, Yu-Chi, CHEN, Li-Kang y UNG, Shao-Kang en *The Design and Characteristic Study of a 3-dimensional Piezoelectric Nano-positioner*, donde se utiliza un sensor de

[1] Fleming, Andrew J. y Leang, Kam K. Evaluation of Charge Drives for Scanning Probe Microscope Positioning Stages. En: American Control Conference (11-13, Junio, 2008). Westin Seattle Hotel, Seattle, Washington, USA

desplazamiento láser KEYENCE LK-H020[1] de elevado costo (aproximadamente U$ 4500.00), actualmente descontinuado pero se encuentran equivalentes en el mercado en la misma compañía. En este proyecto se estudia el desplazamiento del disco piezoeléctrico utilizando un interferómetro, tal como proceden Lloyd y Paetkau[2]; este interferómetro es fabricado para realizar las mediciones de desplazamiento, con un costo mucho más bajo que comprar algún dispositivo comercial para la medición a escalas nanométricas, como los mencionados anteriormente.

En este trabajo se hacen correcciones de histéresis y arrastre con un control en lazo cerrado para la medición del desplazamiento del disco piezoeléctrico en función del voltaje aplicado y utilizando el interferómetro construido, todo esto con un precio bastante asequible para instituciones que no cuenten con un presupuesto muy elevado.

ANTECEDENTES

Ruiz[3] en su documento “Diseño e implementación de un AFM para visualización y fabricación de nanoestructuras” enfatiza en el diseño y la implementación de un microscopio de fuerza atómica, este AFM dispone de un posicionador fino, formado por una estructura que contiene un arreglo de discos piezoeléctricos que permiten efectuar los movimientos a escalas nanométricas. En este documento

---

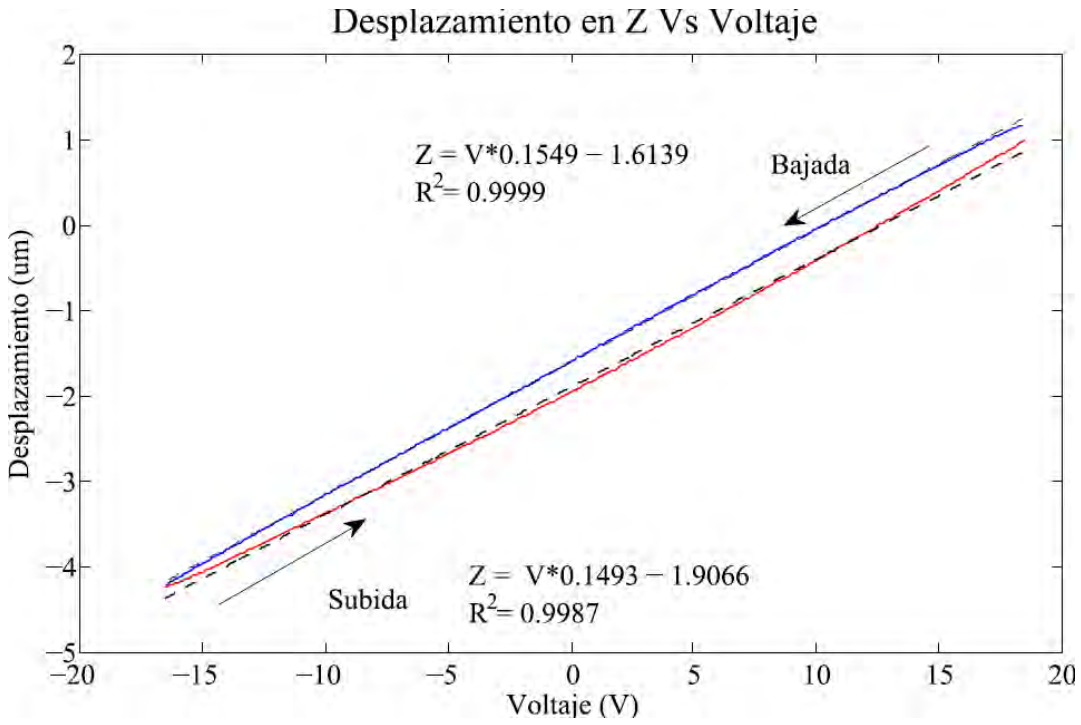
[1] Keyence. Ultra High-Speed/High-Accuracy. Laser Displacement Sensor LK-G5000 series. Catálogo por internet, Disponible en internet: http://www.keyence.com/dwn/lkg5000_ka.pdf

[2] LLOYD, Samantha y PAETKAU, Mark. Characterization of a Piezoelectric Buzzer Using a Michelson Interferometer. Thompson Rivers University, Kamloops, BC, Canadá

[3] RUIZ ROSERO, Juan Pablo. Diseño e implementación de un AFM para visualización y fabricación de nanoestructuras. Tesis de maestría. Bogotá: Universidad de los Andes, 2012

también se realiza una caracterización de los discos piezoeléctricos; se procede a efectuar la medición del desplazamiento del centro de un disco, aplicando una señal de voltaje en rampa desde -17 V a 17 V de forma ascendente y descendente con pasos de 0.1 V. Se obtiene la gráfica de desplazamiento contra voltaje con la ayuda del AFM Asylum Research MFP-3D-BIO de la Universidad de los Andes y se descubre que existe mayor precisión en el desplazamiento cuando se aplica un voltaje descendente en los piezoeléctricos como se muestra en la Figura 2; también se indica que en promedio el disco piezoeléctrico se desplaza aproximadamente 0.1549 µm por cada voltio aplicado.

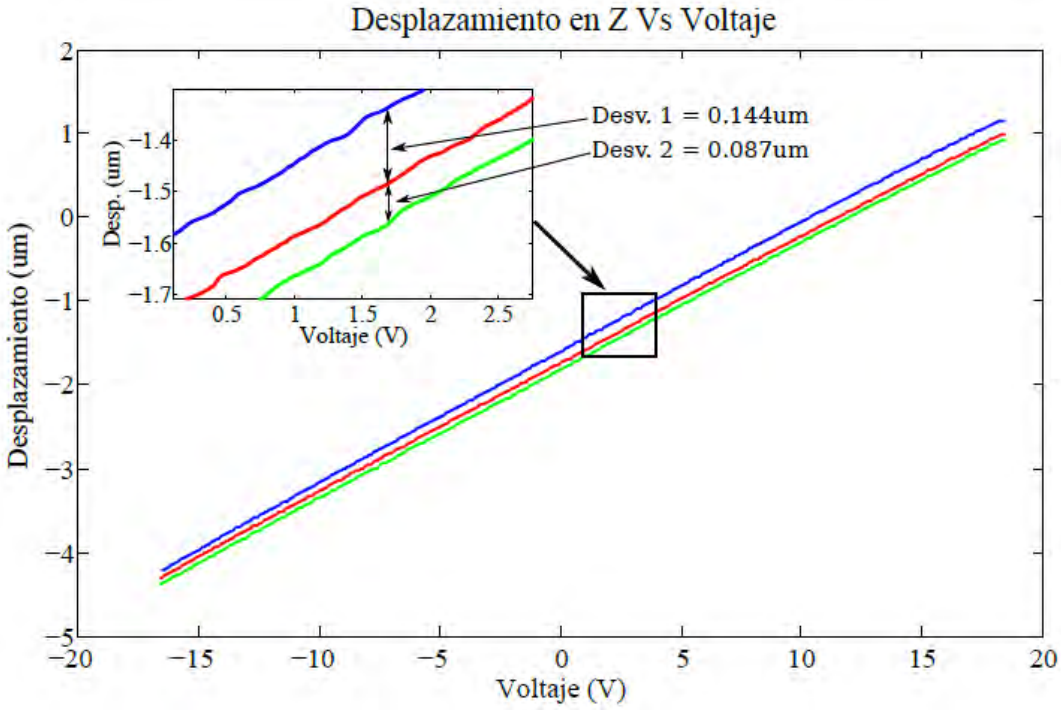
Figura 2. Desplazamiento del piezoeléctrico en función del voltaje. Línea azul: Voltaje descendente, línea roja: Voltaje ascendente, línea punteada: Regresión lineal



RUIZ ROSERO, Juan Pablo. Diseño e implementación de un AFM para visualización y fabricación de nanoestructuras. Tesis de maestría. Bogotá: Universidad de los Andes, 2012

Ruiz[1]; después realiza el estudio para 3 distintos desplazamientos consecutivos con la señal de rampa descendente, para poder encontrar el *drift* o arrastre del disco, que es la variación en el desplazamiento que éste sufre debido al arrastre en el material piezoeléctrico y/o a los cambios de temperatura. Se obtienen los datos de la Figura 3, y se observa que hay un desplazamiento aproximado de 0,144 µm entre tramos descendentes, pero que este desplazamiento tiende a disminuir.

Figura 3. Efecto de *drift* o arrastre en el piezoeléctrico. Línea azul: Primer desplazamiento, Línea roja: Segundo desplazamiento, Línea verde: Tercer desplazamiento, en el detalle se observa la desviación entre cada curva



RUIZ ROSERO, Juan Pablo. Diseño e implementación de un AFM para visualización y fabricación de nanoestructuras. Tesis de maestría. Bogotá: Universidad de los Andes, 2012

[1] RUIZ ROSERO, Juan Pablo. Diseño e implementación de un AFM para visualización y fabricación de nanoestructuras. Tesis de maestría. Bogotá: Universidad de los Andes, 2012

Lloyd y Paetkau[1] en su documento *Characterization of a Piezoelectric Buzzer Using a Michelson Interferometer; r*ealizan la caracterización de un disco piezoeléctrico de tipo zumbador utilizando un interferómetro y un circuito de fotodiodo simple de seguimiento. Para caracterizar la sensibilidad y la histéresis de estos discos se aplican señales de voltaje tipo rampa que van desde los 0V a 17V, obteniendo una taza de desplazamiento de h = 92±3 nm/V. Se observa que se puede realizar el conteo de franjas con un circuito amplificador para un fotodiodo y así evitar el conteo manual de estas; en la práctica siempre existe una intensidad de luz de fondo, pero si esta es constante no afecta el término sinusoidal.

Fleming y Leang[2] en *Evaluation of Charge Drives for Scanning Probe Microscope Positioning Stages;* desarrollan el estudio para mejorar la linealidad de un microscopio de sonda de barrido teniendo en cuenta la histéresis mostrada por los actuadores piezoeléctricos. Además se estudian las ventajas de impulsar los actuadores con carga en lugar de tensión, generando una reducción en el ruido del sensor; se realiza también un estudio detallado de las unidades de carga y se muestra sus ventajas y desventajas; se destacan entre estas: el voltaje final utilizado por el actuador disminuye notablemente para voltajes pequeños, la complejidad de los circuitos y la necesidad de calibración de varios factores.

---

[1] LLOYD, Samantha y PAETKAU, Mark. Characterization of a Piezoelectric Buzzer Using a Michelson Interferometer. Thompson Rivers University, Kamloops, BC, Canadá

[2] Fleming, Andrew J. y Leang, Kam K. Evaluation of Charge Drives for Scanning Probe Microscope Positioning Stages. En: American Control Conference (11-13, Junio, 2008). Westin Seattle Hotel, Seattle, Washington, USA

Wang, Chen y Hung[1] 2010 en *The Design and Characteristic Study of a 3-dimensional Piezoelectric Nano-positioner;* presentan la realización de un estudio para proporcionar una técnica precisa para la medición del desplazamiento de los materiales piezoeléctricos, tratando de reducir los errores de desplazamiento del actuador piezoeléctrico a menos del 5%, para esto se desarrolla la medición con interferometría en actuadores multicapa. Las mediciones se llevaron a cabo con técnicas de interferometría de Michelson y Moiré.

OBJETIVOS

Objetivo general

Controlar los errores de histéresis y arrastre de un disco piezoeléctrico.

Objetivos específicos

- Realizar un análisis bibliográfico sobre interferometría.
- Desarrollar el montaje de un interferómetro para medir desplazamientos a micro y nano escala.
- Medir el desplazamiento en función del voltaje de un disco piezoeléctrico.
- Caracterizar el interferómetro construido con el microscopio de fuerza atómica Asylum Research MFP-3D-BIO de la Universidad de los Andes.
- Implementar un control en lazo cerrado que permita corregir los errores de histéresis y de arrastre del disco piezoeléctrico.

---

1 WNAG, Yu-Chi, CHEN, Li-Kang y HUNG, Shao-Kang. The Design and Characteristic Study of a 3-dimensional Piezoelectric Nano-positioner. En: SICE Annual Conference (18-21, Agosto, 2010) The Grand Hotel, Taipei, Taiwan

- Caracterizar el disco piezoeléctrico con corrección de histéresis y arrastre con el interferómetro caracterizado.

- Elaborar un artículo para publicación internacional con los resultados obtenidos.

ALCANCE

Estudiar la reacción del disco piezoeléctrico frente a diferentes señales eléctricas de voltaje tipo rampa, estas se aplican consecutivamente e intercaladas entre ascendentes y descendentes, se realiza el montaje de un interferómetro, a fin de hacer las mediciones de su desplazamiento, obteniendo diferentes datos de las reacciones del disco frente a estas señales, entre estos se destaca el comprobar la existencia de errores presentados por la histéresis y el arrastre en el disco piezoeléctrico.

El interferómetro permite realizar mediciones de desplazamiento del disco piezoeléctrico al aplicarle diferentes voltajes obteniendo la curva de desplazamiento Vs. voltaje del disco piezoeléctrico, se procede a realizar diferentes pruebas para efectuar mejoras en el sistema de medición y obtener mejores resultados.

Se caracteriza el interferómetro en el microscopio de fuerza atómica Asylum Research MFP-3D-BIO de la Universidad de los Andes y se implementa el control en lazo cerrado que permite corregir los errores de histéresis y de arrastre del disco piezoeléctrico.

Los productos finales de este proyecto son:

- Las curvas de desplazamiento Vs. El voltaje aplicado del disco piezoeléctrico estudiado.

- Un interferómetro con capacidad de medir desplazamiento a escalas nanométricas en una dimensión.
- Un actuador piezoeléctrico controlado de una dimensión con rango de 3 µm y error menor al 1%.

En cuanto al impacto y los resultados esperados se puede decir que: la caracterización del desplazamiento a nano escalas de un disco piezoeléctrico demuestra que la Universidad de Nariño puede realizar investigación en el área de la nanotecnología. Al obtener las características de un disco de este tipo se puede efectuar futuros estudios sobre sus aplicaciones en muchas áreas. El estudio del método de interferometría genera una base en esta área para futuras aplicaciones, permitiendo desarrollar nuevos proyectos y estudios utilizando esta técnica y sus variaciones.

La construcción del interferómetro brinda una herramienta para medir distancias nanométricas; tener un disco piezoeléctrico, caracterizado con un control en lazo cerrado que reduzca los errores de histéresis y arrastre, permite el uso de éste para posteriores aplicaciones en sistemas que necesiten nano posicionadores o desplazamientos nanométricos, incentivando a los estudiantes a seguir su estudio y desarrollar proyectos con bases en este estudio. Se forma un nuevo investigador en el área de control e instrumentación.

1. MARCO CONCEPTUAL

1.1 PIEZOELECTRICIDAD

La piezoelectricidad es una propiedad presentada por ciertos cristales, que generan polarización eléctrica en su masa al ser sometidos a tensiones mecánicas, apareciendo una diferencia de potencial y cargas en su superficie. Esta propiedad también se presenta a la inversa, esto es que estos materiales se deforman bajo la acción de fuerzas internas al ser sometidos a un campo eléctrico. Al dejar de someter los materiales piezoeléctricos a un voltaje o un campo eléctrico estos recuperan su forma[1].

Dentro de los materiales piezoeléctricos pueden distinguirse dos grupos: los que son piezoeléctricos de forma natural, como el cuarzo, y los llamados ferroeléctricos, que poseen propiedades piezoeléctricas tras ser sometidos a una polarización, tales como el tantalio de litio o el nitrato de litio.

Las aplicaciones para estos materiales son muy variadas, entre ellas se puede encontrar: encendedores electrónicos, sensores de vibración, inyectores de combustible de los motores de combustión interna, actuadores, altavoces agudos (Tweeters), pequeños altavoces, reguladores de presión proporcional neumáticos entre otros.

---

[1] CADY, Walter Guyton. Piezoelectricity: An Introduction to the Theory and Applications of Electromechanical Phenomena in Crystals, Dover Press. 1964. 822 p.

Como dicen Zhou, Henson y Bell[1], ignorando el efecto de la histéresis, se tiene la siguiente formula $\frac{\Delta l}{l} = dE$, donde d se conoce como la constante de deformación piezoeléctrica, E es la intensidad del campo eléctrico y $\frac{\Delta l}{l}$ es la deformación, debido a $\frac{V}{l} = E$ entonces $\Delta l = dV$, donde V es el voltaje aplicado.

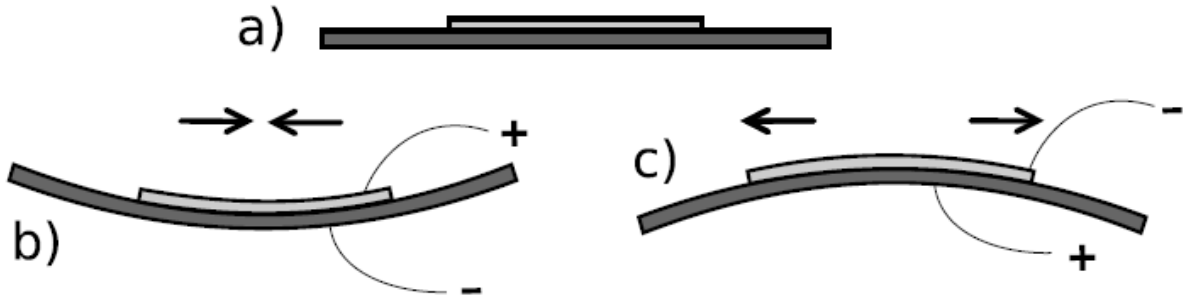
Esta es la forma más sencilla de entender el comportamiento de un piezoeléctrico, ya que la fórmula depende de muchos otros factores y constantes que cambian según el material piezoeléctrico, su composición y las condiciones ambientales, como la temperatura entre otras.

1.1.1 Discos piezoeléctricos Los discos o zumbadores piezoeléctricos están formados por dos placas muy finas de distintos metales, o en algunos casos, de una placa de metal sobre una de cerámica. El comportamiento de ambas capas diferentes es tal que al recibir una presión emiten una corriente eléctrica, la cual es muy pequeña como para notarla con la piel, pero sí lo suficiente como para ser usada de señal electrónica para algunas aplicaciones.

Para esta investigación se selecciona un tipo de piezoeléctrico comercial de fácil adquisición como lo son los parlantes de piezoeléctricos tipo disco, los cuales desplazan su centro hacia afuera o hacia adentro al aplicarles un voltaje DC positivo o negativo según su polarización de fabrica, ver Figura 4.

[1] ZHOU, Huixing, BRIAN Henson y BELL, Andrew. Linear piezo-actuator and its applications. The 5th International Conference on Frontiers of Designand Manufacturing (ICFDM 2002), Dalian, China, Vol. 1, pp 16-20, Julio 2002 [citado en 2013-01-08] Disponible en internet: http://zhouhx.tripod.com/piezopaper.pdf

Figura 4. Deformación del disco piezoeléctrico. a) Piezoeléctrico sin excitación de voltaje, b) Piezoeléctrico con excitación positiva, c) Piezoeléctrico con excitación negativa



*La deformación se muestra de forma exagerada en la grafica para poder percibirla.
RUIZ ROSERO, Juan Pablo. Diseño e implementación de un AFM para visualización y fabricación de nanoestructuras. Tesis de maestría. Bogotá: Universidad de los Andes, 2012

1.2 INTERFEROMETRÍA

Como dicen Bunch y Hellemans[1], la interferometría hace referencia a una familia de técnicas en las que se hacen interferir dos ondas o señales con el fin de obtener información sobre ellas. Este patrón de interferencia puede ser constructivo o destructivo y la imagen en la cual se obtiene se llama interferograma.

1.2.1 Interferómetro El interferómetro es un instrumento que emplea la interferencia de las ondas de luz para obtener con gran precisión diferentes valores de propiedades o variables que se deseen.

[1] BUNCH, Bryan H y HELLEMANS, Alexander. The history of science and technology. Houghton Mifflin Harcourt. Abril 2004. Pp 695

Hay muchos tipos de interferómetros, pero la base del funcionamiento de todos es que utilizan dos haces de luz que recorren dos trayectorias distintas que están determinadas por un sistema de espejos y demás instrumentos ópticos que finalmente logran converger estos haces de luz para formar un patrón de interferencia.

Se puede medir la longitud de onda de un rayo de luz monocromático si se utiliza un interferómetro dispuesto de manera tal que un espejo situado en la trayectoria de uno de los haces de luz se pueda desplazar una pequeña distancia que podamos medir con precisión. Si en lugar de eso lo que se conoce es la longitud de onda del rayo de luz monocromático, entonces se puede medir el desplazamiento pequeño de uno de los espejos[1].

Los índices de refracción de una sustancia también pueden medirse con un interferómetro, y se calculan a partir del desplazamiento en las franjas de interferencia causado por el retraso del haz[2].

1.2.2 Interferómetro de Michelson El interferómetro de Michelson, inventado por Albert Abraham Michelson, es capaz de medir desplazamientos con una precisión muy alta; su funcionamiento se basa en la división de un haz de luz monocromático en dos haces para que recorran caminos diferentes y luego

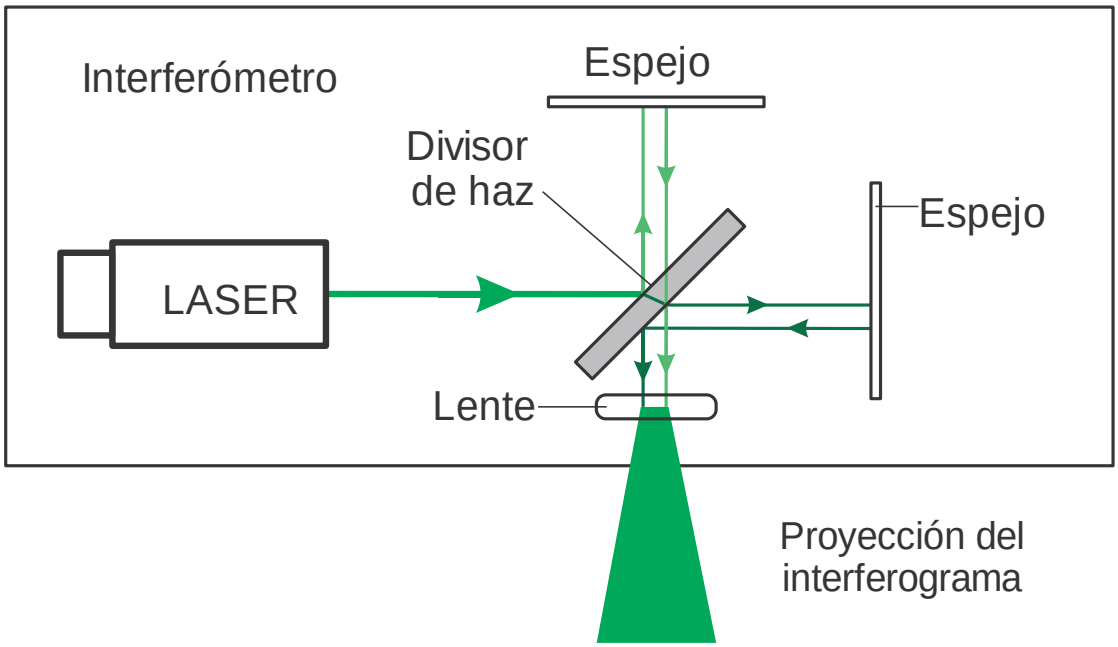
---

[1] HARIHARAN, P. Basics of interferometry. 2 ed. USA: Elsevier science, 2007. 226 p. ISBN 0-12-373589-0

[2] MICHELSON, Albert y Morley, Edward. On the Relative Motion of the Earth and the Luminiferous Ether. *American Journal of Science* 34 (203): 333–345.

converjan nuevamente en un punto creando la figura de interferencia que permite medir desplazamientos en alguno de los espejos como se muestra en la Figura 5.[1]

Figura 5. Modelo del interferómetro de Michelson.



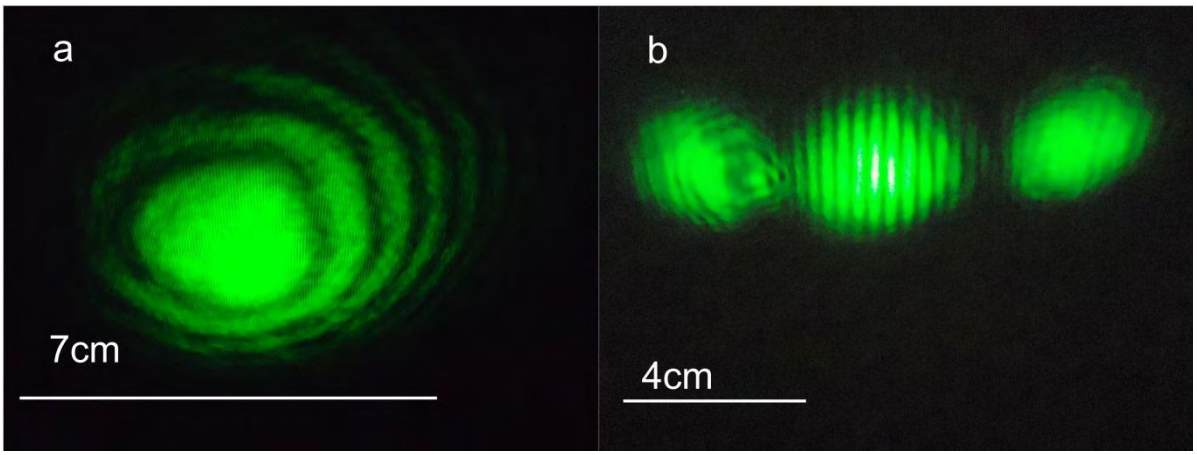
En el interferograma creado según la ubicación de los diferentes elementos se puede observar franjas paralelas o círculos concéntricos correspondientes al patrón de interferencia como se indica en la Figura 6, de este interferograma se pueden realizar las diferentes medidas según su comportamiento.[2]

---

[1] Michelson Interferometer Operation. Block Engineering. [citado en 2013-1-2]. Disponible en internet: http://blockeng.com/technology/ftirtechnology.html

[2] HARIHARAN, P. (2007). Basics of interferometry. 2 ed. USA: Elsevier science, 2007. 226 p. ISBN 0-12-373589-0

Figura 6. Interferogramas obtenidos con el interferómetro construido posteriormente en este proyecto, a) Patrones de interferencia de círculos concéntricos, b) Patrones de interferencia paralelos.



Las franjas se desplazan en proporción al desplazamiento de uno de los espejos, y por cada franja que se desplace en un punto determinado, significará que el espejo se ha desplazado la mitad de la longitud de onda de la fuente de luz monocromática, en este caso un láser. Naturalmente se entendería que debería desplazarse una longitud de onda completa pero hay que tener en cuenta que el recorrido del haz de luz es de ida y regreso al golpear en el espejo por lo cual al mover un espejo una determinada longitud el camino que recorre el haz es el doble de éste.

Según Lloyd y Paetkau[1] la formula correspondiente a esta teoría es:

$$d = \frac{m.\lambda}{2}$$

[1] LLOYD, Samantha y PAETKAU, Mark. Characterization of a Piezoelectric Buzzer Using a Michelson Interferometer. Thompson Rivers University, Kamloops, BC, Canadá

Donde d es la distancia desplazada por uno de los espejos, m el número de franjas que han pasado por un punto y λ es la longitud de onda de la luz monocromática.

1.2.3 Láser Un láser (light amplification by stimulated emission of radiation, amplificación de luz por emisión estimulada de radiación) es un dispositivo que genera un haz de luz coherente, con el tamaño, la forma y la pureza controlados.

En el mercado se encuentran diferentes láser según su aplicación, desde los económicos que usualmente se utilizan para presentaciones o generar diferentes figuras, pasando por los láser utilizados en discotecas para generar varios puntos y figuras con diferentes movimientos, los láser utilizados en industria para cortar algunos tipos de materiales, hasta llegar a los láser utilizados en los laboratorios de física.

También encontramos diferentes tipos de láser como los láser gaseosos, el láser de HeNe, el láser de iones de argón, láser de estado solido, el diodo láser, entre otros, cada uno con sus diferentes propiedades. Para el óptimo desarrollo de este proyecto, es recomendable tener en cuenta algunas recomendaciones para el laser, entre estas: un láser de luz visible y que el error en la longitud de onda o la variación de la luz monocromática no sea muy grande.[1]

[1] Tipos de láser, [citado en 2012-12-24] Disponible en internet: http://www.itlalaguna.edu.mx/academico/carreras/electronica/opteca/OPTOPDF5_archivos/UNIDAD5TEMA4.pdf

1.3 SISTEMAS DE CONTROL

Un sistema de control es una interconexión de componentes que forman una configuración del sistema que proporcionará una respuesta deseada. El fundamento proporcionado por la teoría de los sistemas lineales, es la base para el análisis de un sistema. La relación entrada-salida representa la relación entre causa y efecto del sistema, al mismo tiempo representa un procesamiento de la señal de entrada para proporcionar una señal de salida, que muchas veces contiene una amplificación de potencia[1].

Entre los controladores más utilizados en control de procesos, es el denominado control de tres términos o controlador PID (Proporcional Integral Derivativo). El controlador proporciona un término proporcional, un término integral y un término derivativo; muchos procesos industriales se controlan utilizando controladores PID, se puede atribuir su popularidad a su buen comportamiento en un amplio intervalo de condiciones de operación y en parte a su sencillez funcional, que permite a los ingenieros operar con ellos de manera sencilla y directa[2]. Este controlador tiene una función de transferencia

$$G_s(s) = K_P + \frac{K_I}{s} + K_D s$$

[1] DORF, Richard y BISHOP, Robert. Sistemas de control moderno. Traducido por Sebastián Dormido Canto y Raquel Dormido Canto. Revisión técnica por Sebastian Dormido Bencomo. 10 ed. Madrid: Pearson Prentice Hall. 2005. 928 p. ISBN: 84-205-4401-9

[2] DORF, Richard y BISHOP, Robert. Sistemas de control moderno. Traducido por Sebastián Dormido Canto y Raquel Dormido Canto. Revisión técnica por Sebastian Dormido Bencomo. 10 ed. Madrid: Pearson Prentice Hall. 2005. 928 p. ISBN: 84-205-4401-9

Un control PID es un control por realimentación que calcula la desviación o error entre un valor medido y el valor que se quiere obtener, para aplicar una acción correctora que ajuste el proceso. El algoritmo de cálculo del control PID se da en tres parámetros distintos: el proporcional, el integral y el derivativo.

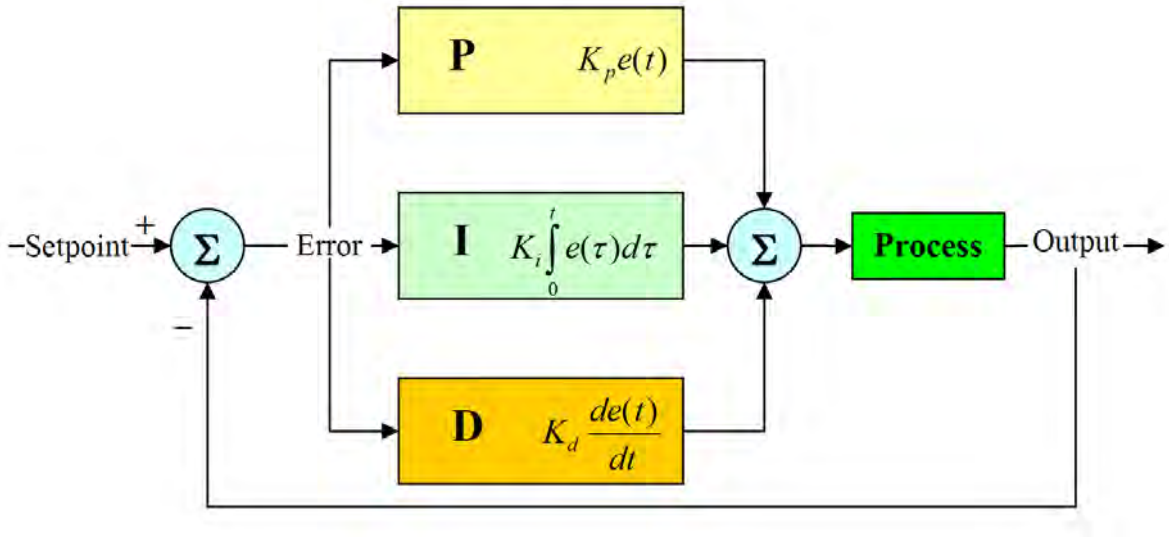
El valor proporcional determina el error actual, el integral genera una corrección proporcional a la integral del error, con esto el error de seguimiento tiende a cero, finalmente el derivativo determina la reacción del tiempo en el que el error se produce. Para ajustar el proceso mediante un control se suman las acciones de estos tres parámetros.

No siempre el uso de un PID garantiza el control óptimo o la estabilidad de un sistema, también algunas aplicaciones no requieren el uso de los tres modos del control PID, el control puede ser PI, PD, P o I, según la ausencia del parámetro.

Los controladores PI son particularmente comunes ya que la acción derivativa es muy sensible al ruido, y la ausencia del proceso integral puede ocasionar un error en estado estacionario.

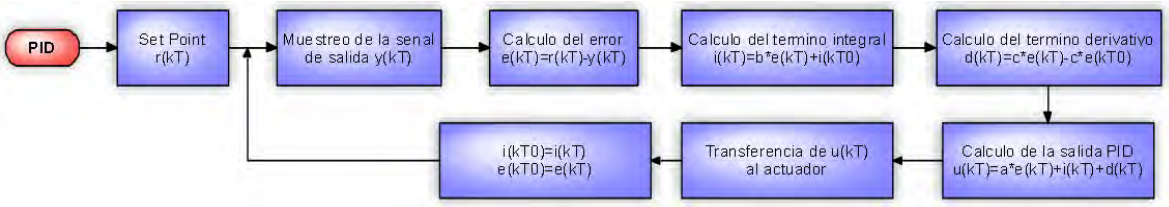
1.3.1 Control PID digital Los controladores PID pueden ser implementados analógica o digitalmente. Comúnmente los controladores PID digitales son implementados en microcontroladores, activando la función de control mediante una interrupción cada determinado tiempo. Su programación es muy fácil teniendo claro el funcionamiento de un controlador PID, en la Figura 7 se observa el diagrama de bloques de un control PID y en la Figura 8 se muestra el algoritmo de programación de un PID digital en un microcontrolador.

Figura 7. Diagrama de bloques de un controlador PID en lazo cerrado



RADHESH. PID Controller Simplified. (en línea). Mayo 11, 2008. [Citado en 2013-12-24]. Disponible en internet: http://radhesh.wordpress.com/2008/05/11/pid-controller-simplified/

Figura 8. Algoritmo de programación de un PID digital en microcontrolador



RUGE, Ilber Adonayt. Método básico para implementar un controlador digital PID en un microcontrolador PIC para desarrollo de aplicaciones a bajo costo.[citado en 2012-12-21] Disponible en internet: http://www.edutecne.utn.edu.ar/microcontrol_congr/industria/MTODOB~1.PDF

El muestreo (T) debe ser mayor que el tiempo de establecimiento del sistema en lazo abierto; efectuando las siguientes operaciones:

$$e(kT) = r(kT) - y(kT)$$

$$i(kT) = b * e(kT) + i(kT0)$$

$$d(kT) = c * e(kT) - c * e(kT0)$$

$$u(kT) = a * e(kT) + i(kT) + d(kT0)$$

Donde: *e* corresponde al error, *r* al SetPoint, *y* al valor de la señal de salida, *i* al termino integral, *b* la constante integral, *d* al termino derivativo, *c* la constante derivativa, *u* la salida del PID o la acción correctora del control, *a* la constante proporcional, *KT* el muestreo actual, *KT0* el muestro anterior al actual.

En seguida efectúa la acción correctora y según el valor de $u(kT)$ se procede a actualizar los valores anteriores del error y el término integral de la siguiente manera:

$$i(kT0) = i(kT)$$
$$e(kT0) = e(kT)$$

Cerrando así el lazo del control.

2. DESARROLLO DEL PROYECTO

Para analizar las soluciones existentes para la medición de desplazamiento en materiales piezoeléctricos se realiza una revisión bibliográfica del tema. Para ello se utiliza diferentes fuentes de información como lo son: internet, la biblioteca de la Universidad de Nariño y las bases de datos a las que está adscrita la Universidad de los Andes. Para desarrollar el montaje del interferómetro se elige la mejor solución según los estudios realizados previamente y se estudia la posibilidad de efectuar mejoras al montaje y así poder medir desplazamientos a micro y nano escala.

Para medir el desplazamiento en función del voltaje de un disco piezoeléctrico se utiliza el interferómetro montado anteriormente y se realiza las medidas aplicándole las señales tipo rampa ascendentes y descendentes, también se realiza el estudio para varios desplazamientos continuos con el fin de encontrar el *drift* o arrastre. Para realizar la caracterización del interferómetro se utiliza el microscopio de fuerza atómica Asylum Research MFP-3D-BIO de la Universidad de los Andes en Bogotá. Finalmente se implementa un control en lazo cerrado que permite corregir los errores de histéresis y de arrastre del disco piezoeléctrico, para esto se utiliza un micro controlador DSPIC30f4013.

2.1 DISEÑO Y ELABORACIÓN DEL INTERFERÓMETRO

Para el éxito del proyecto se requiere de un interferómetro confiable, pero en el mercado los costos de éste son muy elevados para algunas instituciones; entonces se decide diseñar y construir el interferómetro para poder reducir significativamente los costos del proyecto. Al ser un dispositivo que permite medir pequeños desplazamientos, su estructura requiere ser muy firme y se debe reducir

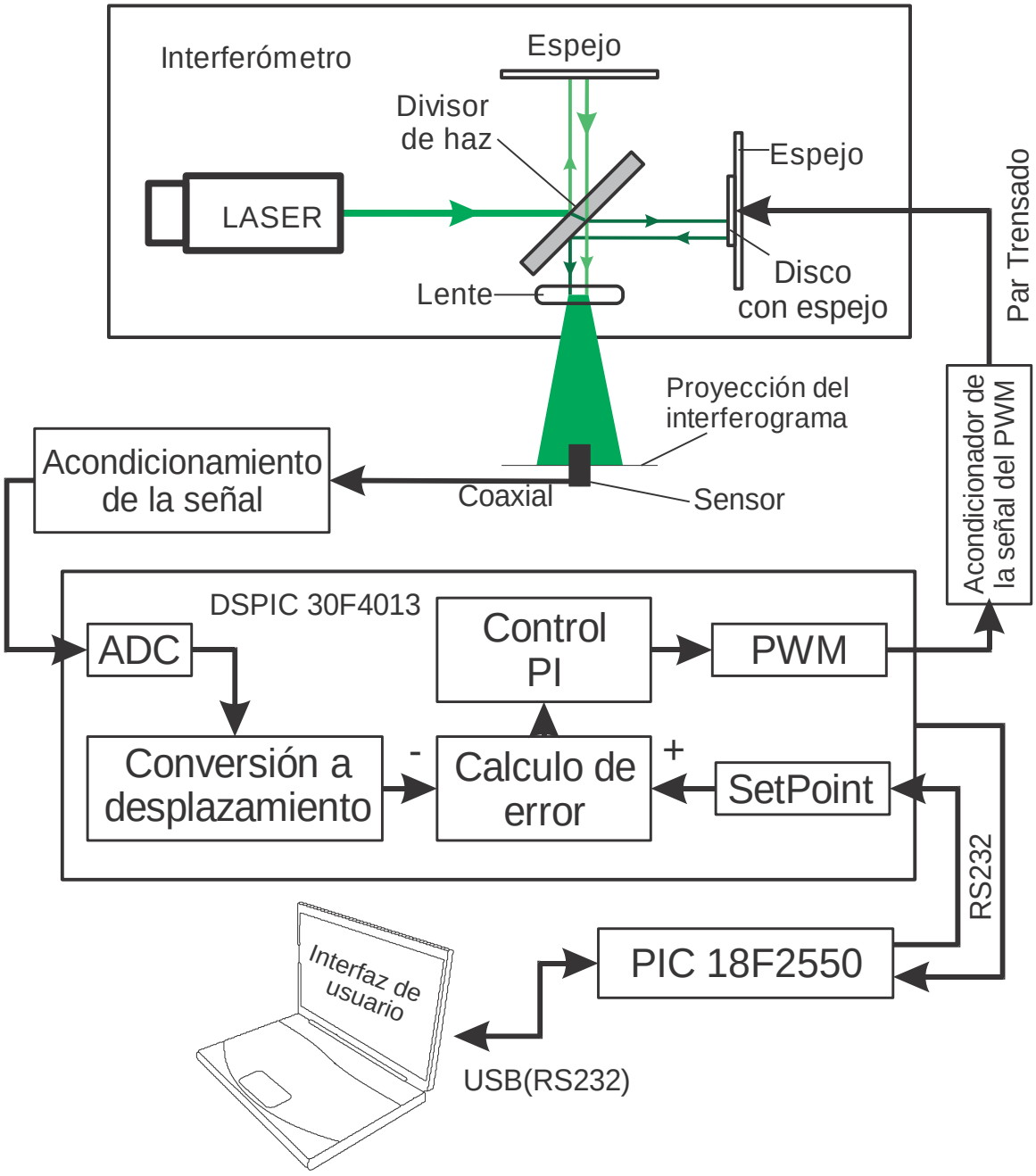
al máximo las alteraciones producidas por vibraciones externas como las pisadas fuertes de las personas, las microvibraciones y el ruido del terreno si es posible.

El esquema del interferómetro propuesto se aprecia en la Figura 9. Se observa la parte del interferómetro compuesta por el láser, que genera un haz de luz monocromática color verde de longitud de onda de 532±1 nm, éste es dividido en dos por medio de un divisor de haz; uno de ellos se dirige hacia el espejo fijo y el otro hacia el espejo móvil, que está situado en el disco piezoeléctrico. Los haces de luz divididos del laser rebotan en los espejos, se recombinan en el divisor de haz y son ampliados por una lupa que visualiza el interferograma.

Por medio del sensor, un fototransistor que se detalla más adelante, se transforma la información del interferograma en una señal eléctrica; esta es acondicionada por medio de filtros pasa bajos análogos, ya que los desplazamientos en el disco piezoeléctrico son lentos, generando una onda sinusoidal de baja frecuencia en el sensor, y es enviada al DSPIC (Digital Signal PIC Microcontroller). En el DSPIC el ADC convierte la señal análoga en una señal digital y mediante cálculos transforma esta señal a desplazamientos del disco piezoeléctrico.

Mediante la interfaz de usuario el computador envía la información de ajuste del control PI, mediante RS232/USB y almacena los datos que necesita. Con la información del desplazamiento del disco piezoeléctrico y el SetPoint, el programa calcula el error y procede a efectuar las operaciones del controlador; según el cálculo del controlador, el DSPIC envía por medio del PWM, la señal adecuada para ser convertida por el acondicionador del PWM, éste genera un voltaje que se aplica al disco piezoeléctrico, que se desplazará según este cambio en la señal, cerrando el laso del control.

Figura 9. Esquema del interferómetro propuesto con sus principales partes: Láser, espejos, sensor, disco piezoeléctrico y microcontroladores; las flechas verdes indican las direcciones de los haces del láser.



2.1.1 Diseño del interferómetro Para reducir las vibraciones y garantizar la estabilidad de las piezas que sostendrán los espejos y demás elementos del interferómetro, se decide que las bases de los espejos, el divisor del haz de luz y la base para el láser, deben tener en la parte inferior un imán para sujetarse a la base del interferómetro; para esto se utiliza imanes extraídos de un magnetrón de un horno microondas ya que tienen forma de argolla y facilita el trabajo a realizar en la fabricación de las bases, además de su reducido costo.

Se decide usar una base pesada y metálica para que se puedan adherir las bases magnéticas de los materiales ópticos; se elige como base una lamina de metal de ½ pulgada de espesor, y con medidas de 40x50 cm, con un marco en madera cedro para aumentar su peso y mejorar su presentación. Las dimensiones y detalles de los elementos del interferómetro se observan posteriormente en este documento.

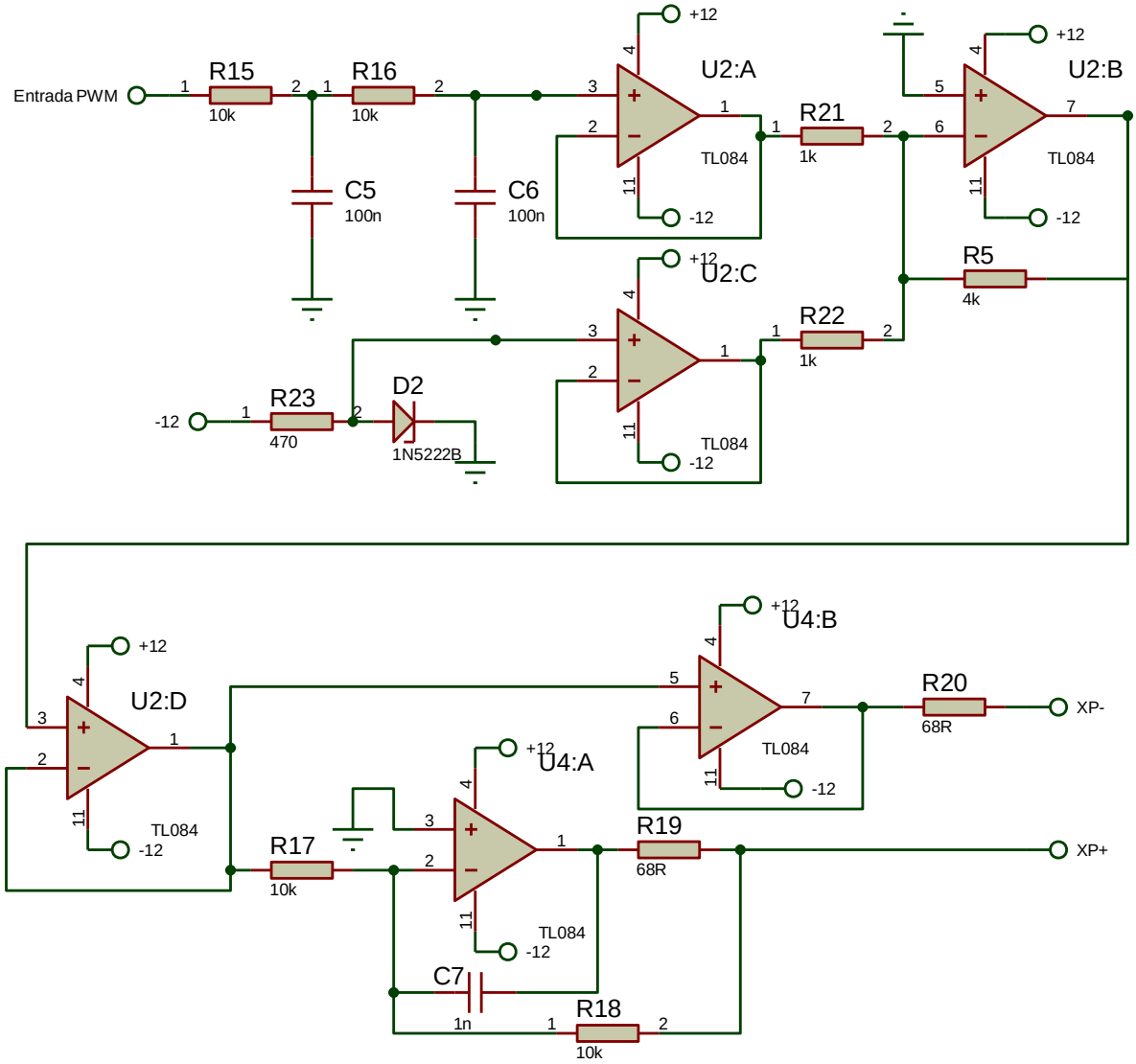
2.1.2 Sistema electrónico del interferómetro Para este proyecto no se puede realizar el conteo del desplazamiento de las franjas del interferograma de forma manual; es necesario contar estas franjas de manera electrónica, para una mayor comodidad, para tener su registro digital en tiempo real y de esta forma poder implementar el controlador.

Uno de los espejos es colocado sobre el disco piezoeléctrico y éste es sujetado a una de las bases del interferómetro; de esta forma, al aplicarle un voltaje al disco, éste mueve el espejo y así se mueven las franjas del interferograma con las cuales se puede determinar su desplazamiento.

Para poder mover el disco se implementa un sistema que permite aplicarle un voltaje desde -17 V a +17 V que es el rango en el cual se hace el estudio; éste es controlado mediante el PWM del DSPIC y se diseña para funcionar en este rango; sin embargo sus limites experimentales corresponde a valores entre -18.5 V y +18

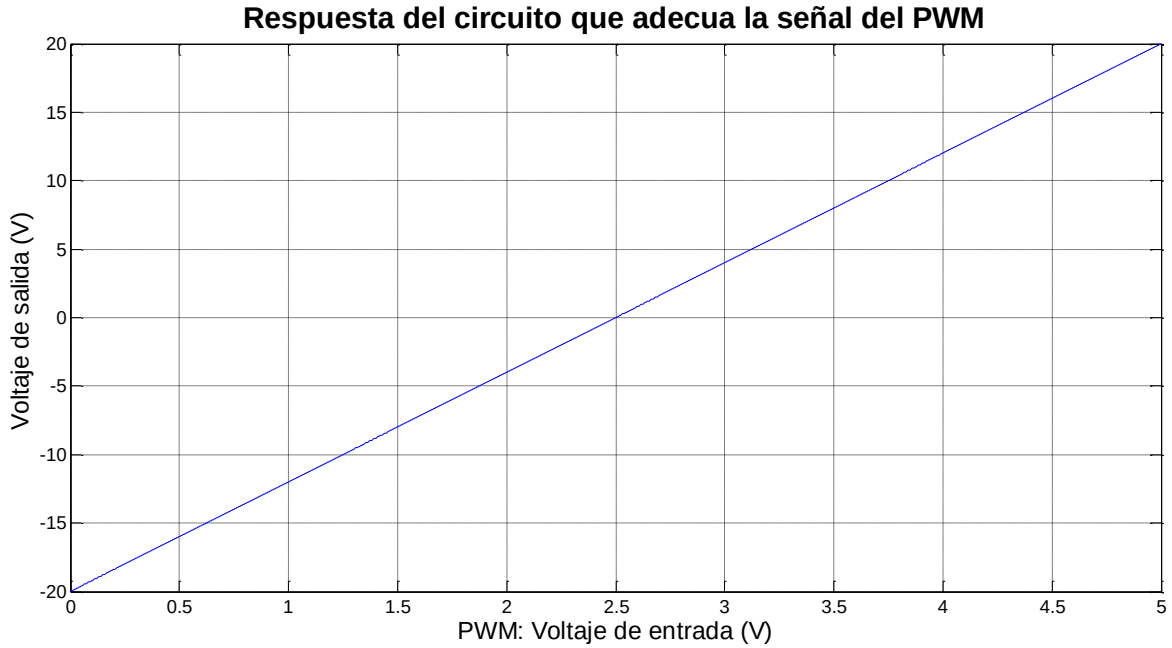
V aproximadamente; estos corresponden al mínimo y al máximo del PWM de 16 bits (0-65535). En la Figura 10 se muestra el circuito que adecua la señal del PWM.

Figura 10. Circuito que adecua la señal del PWM a voltaje para manipular el disco piezoeléctrico



En el anterior circuito se pueden identificar varias etapas: U2:A y los elementos hasta la entrada del PWM, sirven como acoplador de impedancia y para convertir la señal del PWM, en una señal continua; U2:C genera -2,5 V; U2:B es un amplificador sumador inversor, con ganancia igual a 4, de tal manera que a la entrada del PWM le esta restando 2,5 V y a este resultado lo esta multiplicando por -4; U2:D es un acoplador de impedancia que para este caso es un seguidor de voltaje; U4:A y U4:B convierten la señal, en una señal diferencial, se puede observar que la configuración de uno de ellos de un inversor y la del otro es de un no inversor; esto quiere decir que, si la salida de U2:D es igual a +10 V, las salidas de U4:A y U4:B serán iguales a -10 V y +10 V respectivamente; esto hace posible generar la salida diferencial y obtener voltajes entre -20 y +20 V con una alimentación de -12 y +12 V en sus amplificadores. En la Figura 11 se muestra la gráfica de entrada contra salida de este circuito.

Figura 11. Respuesta del circuito que adecua la señal del PWM a voltaje, para manipular el disco piezoeléctrico



Se implementa el controlador en un DSPIC y una interfaz de usuario en computador. Para crear la comunicación entre estos se utiliza un PIC 18F, ya que esta serie cuenta con un modulo de comunicación USB.

La comunicación entre el PC y el DSPIC se realiza mediante RS232/USB; utilizando como puente entre estos dos un PIC18F2550, que únicamente es empleado para recibir la información del PC y enviarla al DSPIC y viceversa.

Antes de empezar a realizar medidas y controlar el disco, el sistema tiene que establecer unas condiciones iniciales. Con el interferómetro se puede conocer el desplazamiento del disco mediante el movimiento de las franjas del interferograma; para realizar los cálculos se deben conocer ciertos valores y así poder realizar los algoritmos en el DSPIC.

La señal que se obtiene es de forma sinusoidal, ésta muestra el desplazamiento de las franjas, que corresponden al desplazamiento del disco piezoeléctrico. Se debe transformar esta onda senoidal a un desplazamiento; el proceso implementado para esta etapa es el siguiente:

- Se realiza el desplazamiento desde el centro del disco hasta el límite inferior para posicionarlo en desplazamiento = 0 nm. Durante este desplazamiento el programa identifica y guarda el valor máximo y mínimo de la onda seno captada por el sensor, detallado más adelante.

- Para convertir la onda seno en un desplazamiento lineal, se debe obtener la inversa del seno y así conseguir una línea recta; esto se realiza si la onda seno tiene como picos en amplitud +1 y -1; se resta a la onda seno una constante calculada por el sistema que es igual a:

$$cte = min + \frac{max - min}{2}$$

Donde min y max corresponden al mínimo y máximo respectivamente, de la onda seno captada por el sensor. Después se divide esta señal entre $(max - min)/2$, obteniendo finalmente una onda seno con amplitud igual a 1 y centrada en el eje Y; realizando el inverso del seno se obtiene una señal triangular, que el programa convierte a una línea recta; se obtiene así un desplazamiento de 4 unidades por cada longitud de onda. Este valor se divide entre 8 y se multiplica por la longitud de onda del láser, en este caso 532 nm. Finalmente se tiene que por cada longitud de onda captada por el sensor, el disco se ha desplazado 266 nm; cumpliendo así la formula de Lloyd y Paetkau[1]:

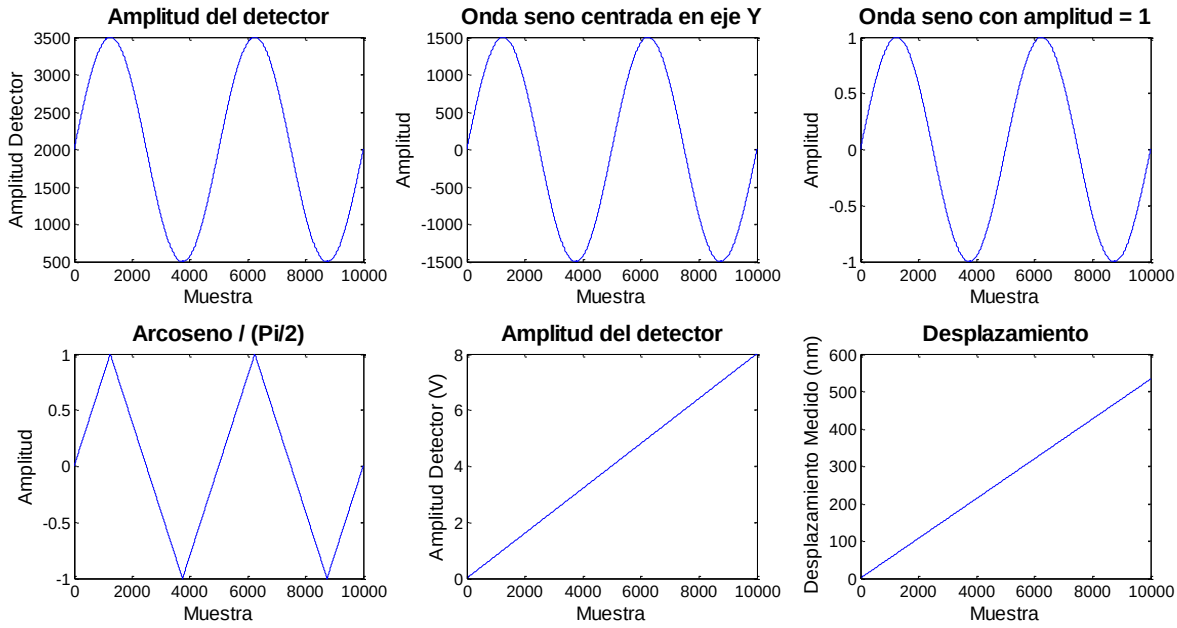
$$d = \frac{m.\lambda}{2}$$

Donde d es la distancia desplazada por uno de los espejos, m el número de franjas que han pasado por un punto y λ es la longitud de onda de la luz monocromática.

Este proceso para el tratamiento de la señal se puede observar mejor en la Figura 12.

[1] LLOYD, Samantha y PAETKAU, Mark. Characterization of a Piezoelectric Buzzer Using a Michelson Interferometer. Thompson Rivers University, Kamloops, BC, Canadá

Figura 12. Proceso de conversión de onda seno a desplazamiento



2.1.2.1 Microcontrolador El microcontrolador seleccionado es un DSPIC de microchip de referencia DSPIC30F4013, utilizados para el tratamiento de señales digitales; además son capaces de realizar operaciones aritméticas a gran velocidad como las operaciones de punto flotante.

De las diferentes propiedades de este DSPIC se menciona algunas de las que motivaron a seleccionarlo:

- Conversor ADC de 12 Bits de resolución, que se utiliza para hacer la adquisición de datos del interferograma mediante el sensor seleccionado.

- PWM de 16 bits de resolución, utilizado como salida para el control del disco piezoeléctrico.

- Se puede configurar para lograr una frecuencia de 80Mhz utilizando un cristal de 20Mhz.

El código del DSPIC se muestra en el ANEXO A; el código fuente para el PIC18F2550 se muestra en el ANEXO B. Se detallan en los anexos cada paso de los programas. Los circuitos correspondientes al DSPIC30F4013 y al PIC18F2550, se indican en la Figura 13 y Figura 14, respectivamente.

Figura 13. Circuito del DSPIC 30F4013

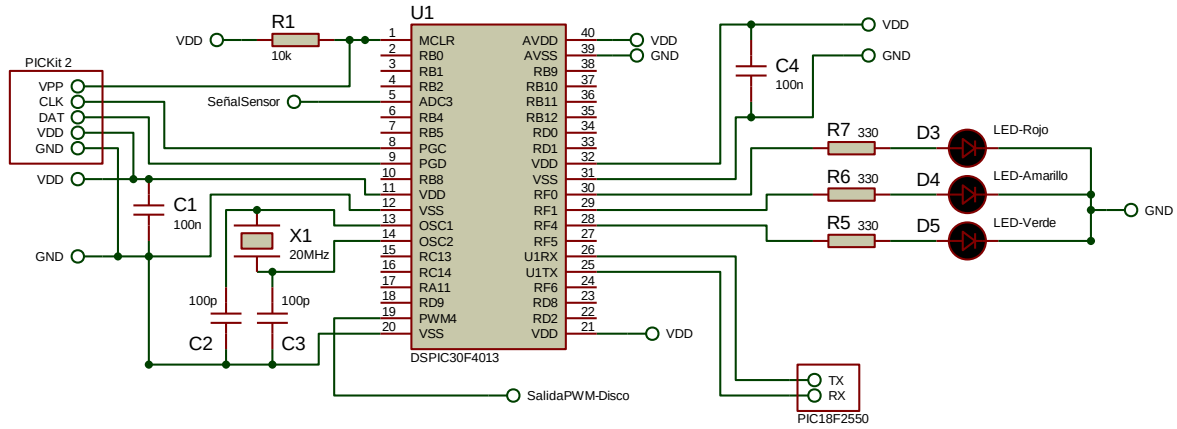
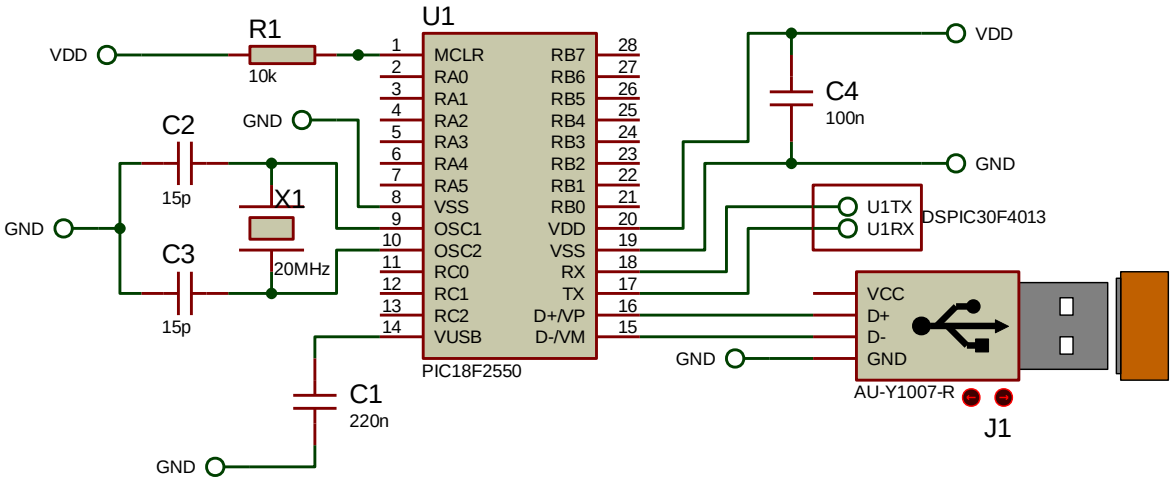
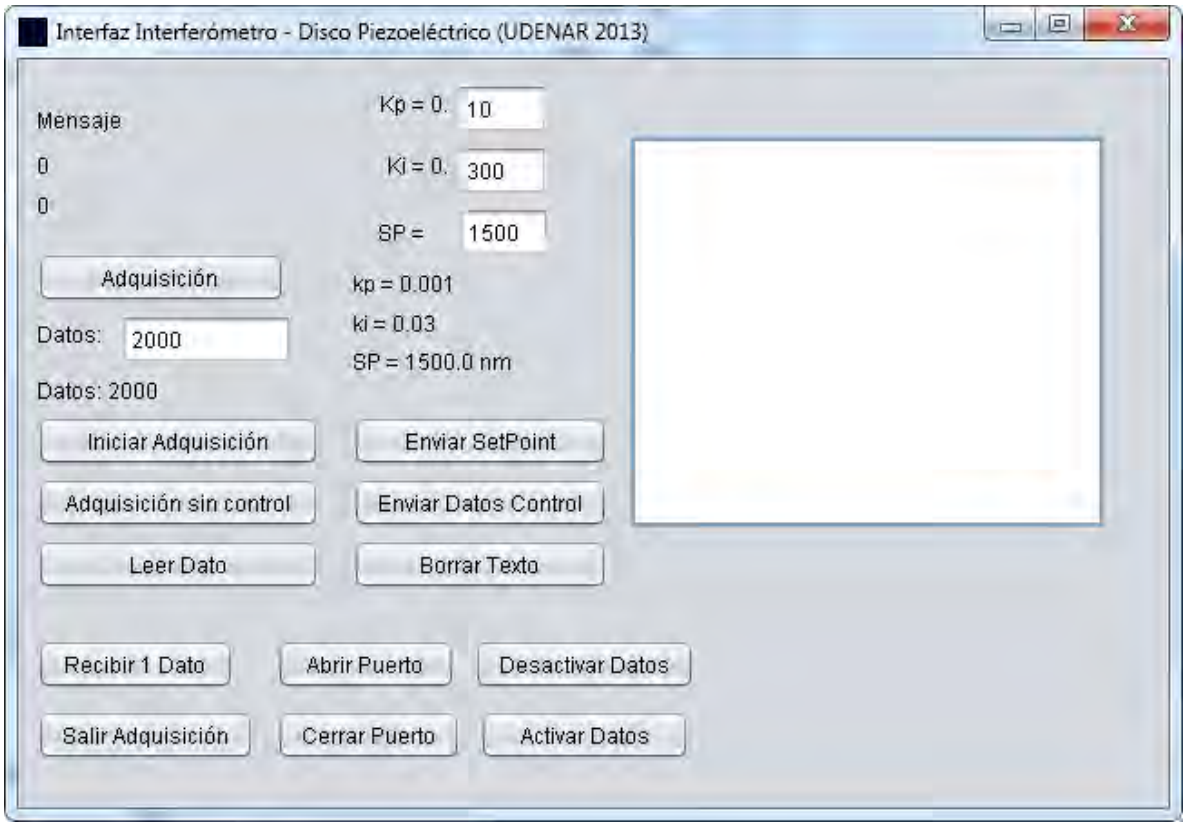


Figura 14. Circuito del PIC 18F2550



2.1.2.2 Interfaz de usuario Para la interfaz de usuario se utiliza un programa en JAVA creado en NetBeans IDE 7.1.2. La interfaz final se muestra en la Figura 15.

Figura 15. Interfaz de usuario en JAVA



En la interfaz encontramos varios botones que permiten realizar toda la manipulación del sistema, al igual que observar sus resultados.

Para las condiciones iniciales el programa tiene el botón “Adquisición”, que envía la orden de realizar movimientos en el disco; de tal forma que el sistema lee esos datos para determinar el máximo y mínimo valor que alcanza el sensor, para poder transformar la señal recibida en el desplazamiento del disco. Además posiciona el

disco en un lugar cercano al extremo para tenerlo como punto de partida y desde éste contar el desplazamiento; este punto será el correspondiente a $d = 0$, siendo d el desplazamiento del disco como se mencionó anteriormente.

Mediante el cuadro de texto “Datos” se digita la cantidad de datos que se desea adquirir; el botón “Adquisición sin control” genera la señal tipo rampa para el disco, efectuando el desplazamiento ascendente, iniciando desde nuestro $d = 0$. El sistema genera la rampa desde -17 V a +17 V con pasos de 11,2 mV; cuando la rampa llega a +17 V, el sistema empieza a generar la rampa descendente de igual manera, lo mismo sucede al llegar nuevamente a -17 V; el sistema continua así hasta generar las muestras indicadas por el programa; cada rampa se genera en un total de 3030 muestras.

El botón “Leer dato” sirve para leer la cantidad de datos que se han enviado desde el DSPIC hasta el PC; esto en el caso que se hubiese enviado un dato que no ha sido leído por el programa, con el fin de no tener datos pendientes por recibir; la cantidad de datos por leer esta determinada por el número que se escriba en el cuadro de dialogo correspondiente a “datos”.

Para realizar la adquisición con el control, primero se debe pulsar el botón de “Adquisición” para establecer los parámetros principales y entrar en una subrutina, que permite continuar con el proceso; esto puede tardar unos pocos segundos. Después se puede pulsar el botón “Iniciar Adquisición”, iniciando así el proceso para alcanzar el SetPoint; utilizando el control con las constantes Ki y Kp que se han enviado al DSPIC. El control se activa cada vez que se recibe un dato y no se ejecuta cada determinado tiempo; éste depende de la velocidad de procesamiento del PC en ese momento.

La adquisición en el caso del botón “Iniciar Adquisición” se realiza de la siguiente forma: el programa envía un dato al DSPIC, que al identificarlo ejecuta los cálculos y conversiones necesarios para establecer el desplazamiento; al estar en el primer

momento en un desplazamiento igual a cero, éste efectúa los cálculos para el controlador, teniendo como base el error actual; ejecuta la corrección a la salida según los cálculos realizados y envía estos datos al PC; el PC al recibir estos datos, los guarda y envía de nuevo una orden al DSPIC, que ejecuta de nuevo los cálculos y correcciones a la salida, según le indique el control; esto se repite tantas veces como se indique en el programa en el cuadro de texto “Datos”.

Lo mismo sucede con el botón ”Adquisición sin control”, pero en lugar de hacer los cálculos para el control, simplemente aumenta el PWM un valor determinado y constante hasta llegar al tope máximo. Después reduce en esa misma cantidad para realizar los desplazamientos ascendentes y descendentes. Al final de la adquisición, el programa muestra 2 gráficas; una con la información del desplazamiento medido y otra con el valor de la lectura del sensor, como se muestran en la Figura 16 y en la Figura 17, respectivamente.

Figura 16. Gráfica del desplazamiento del disco medido

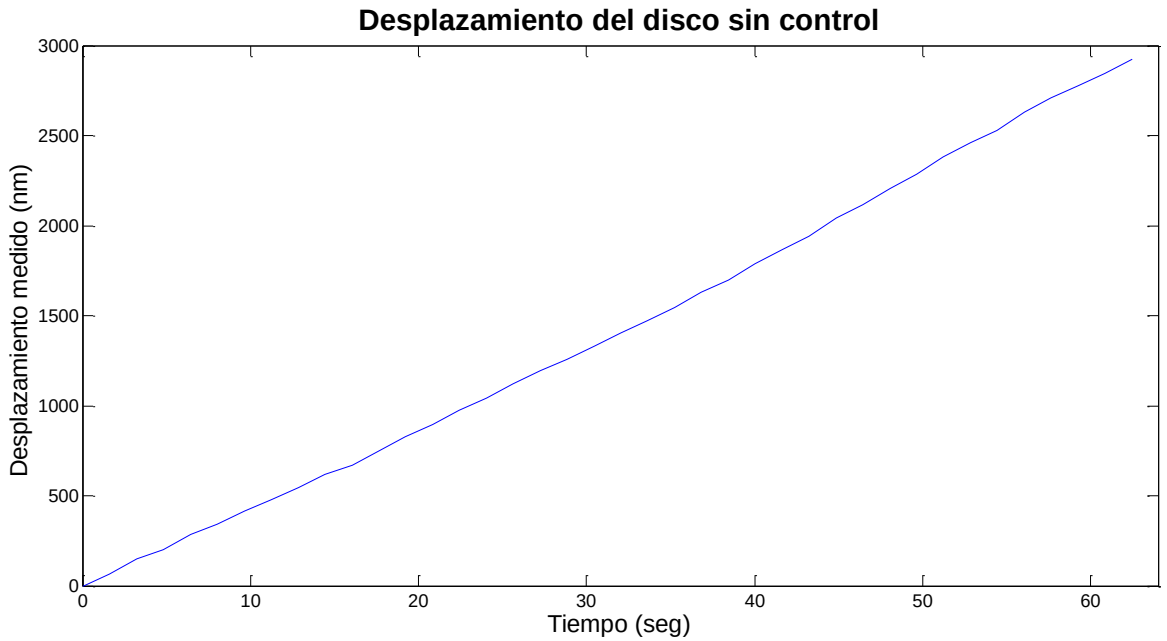
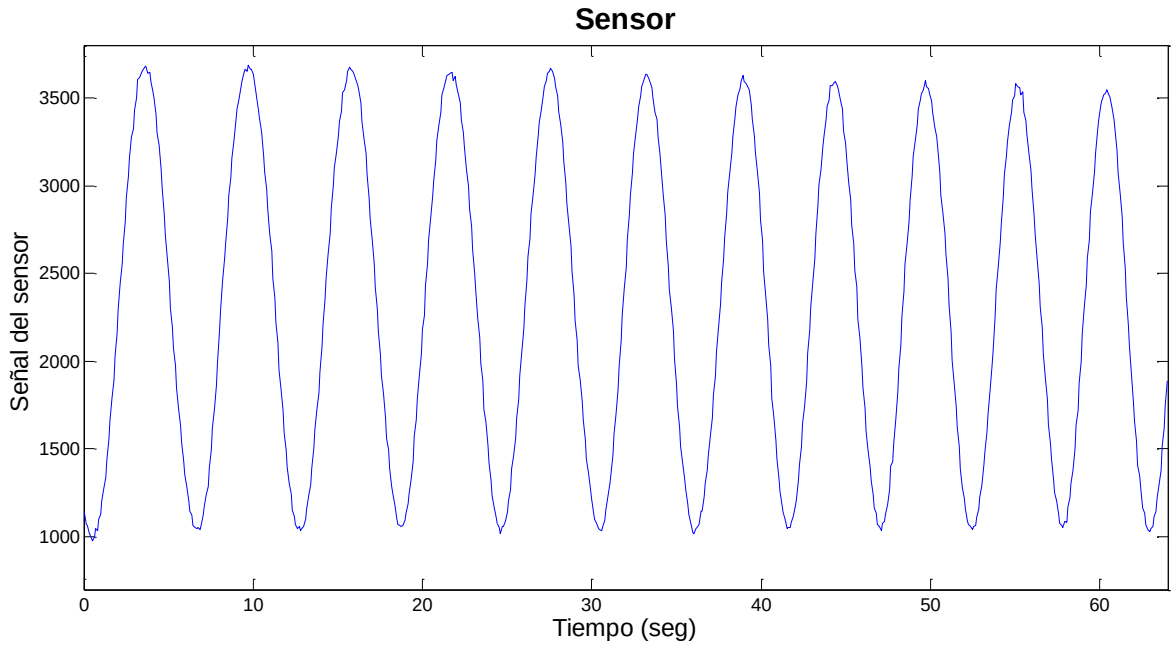


Figura 17. Gráfica del valor del sensor



Si en algún momento se quiere restablecer el disco piezoeléctrico a la posición inicial de 0 nm y establecer de nuevo las constantes para los cálculos de los desplazamientos y algoritmos necesarios, se debe pulsar el botón “Adquisición”. Después de esto el programa efectúa todos los procedimientos y finalmente se sale de esta etapa pulsando el botón “Salir adquisición”; de lo contrario se continúa con cualquiera de las opciones de adquisición, ya sea con o sin control.

En los cuadros de Ki y Kp se escriben los decimales de las constantes correspondientes al control; para el caso de escribir 1 significa que envía como constante 0.001 y si se escribe 9000 se envía como constante el valor 0.9; así funciona para las dos constantes. Para tener una mejor claridad sobre esto, al escribir los valores en los cuadros, aparece el valor que será enviado en la parte inferior del botón “Enviar Datos Control”.

El valor digitado en el cuadro SP, es el valor que se envía al DSPIC, que guarda como el SetPoint; este valor es la distancia deseada del desplazamiento del disco y esta dada en nanómetros; al igual que las constantes Ki y Kp, el valor de este SetPoint aparece en la parte inferior del botón “Enviar Datos Control”.

El botón “Enviar Datos Control”, envía los valores de las constantes Ki y Kp al DSPIC y el botón “Enviar SetPoint”, envía el valor del desplazamiento o el SetPoint que deseamos establecer en el DSPIC.

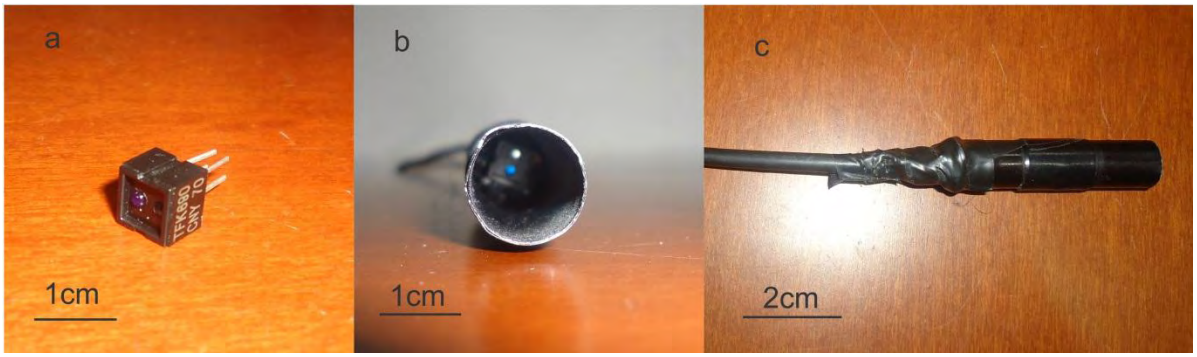
Los botones “Abrir Puerto” y “Cerrar Puerto” sirven para abrir y cerrar el puerto de comunicación del PC; para este caso el COM8.

Cuando el DSPIC esta funcionando mediante las interrupciones por Timer1, existe la opción de leer el dato correspondiente en cada interrupción; esto se activa y desactiva con los botones “Activar Datos” y “Desactivar Datos” respectivamente. También existe la posibilidad de leer un solo dato y ser mostrado en el cuadro de texto grande; esto se realiza con el botón “Recibir 1 Dato”.

Cuando se realiza cualquiera de las opciones “Iniciar Adquisición”, “Adquisición sin control”, “Leer Dato”, “Recibir 1 Dato”; estos se registran en el cuadro de dialogo grande, ubicado en el centro de la interfaz; si en cualquier momento se desea borrar esta información, solo basta con presionar el botón “Borrar Texto”.

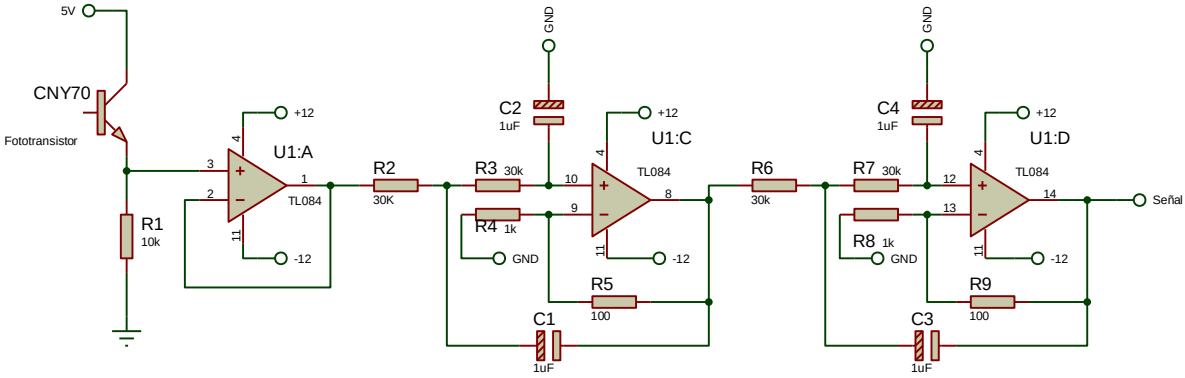
2.1.2.3 Sensor Para realizar la adquisición del fotograma se utiliza un sensor óptico, ubicado en el lugar donde se proyecta el interferograma; el sensor seleccionado es el CNY70, como se muestra en la Figura 18. Éste cuenta con un fototransistor de pequeño tamaño, permitiendo tomar el valor de una zona mas reducida y así no promediar el valor de un lugar tan grande; además se ha ubicado este sensor dentro de un tubo para disminuir los efectos de las luces externas al interferómetro y ha sido acoplado con un cable coaxial.

Figura 18. Sensor CNY70, a) Sensor CNY70 al descubierto, b) Vista del sensor dentro del tubo para disminuir efectos de luces externas al interferómetro, c) Vista superior del tubo contenedor del sensor



La señal generada por el sensor es introducida al sistema mediante el modulo ADC del DSPIC; acondicionada con un sistema de tres etapas como se indica en la Figura 19. En la primera etapa, un seguidor de voltaje; en la segunda y tercera etapa, filtros pasa bajos con la implementación Sallen-Key de segundo orden, con frecuencia de corte en 5,3 Hz. Teniendo en cuenta que los desplazamientos del disco son muy lentos en este proyecto, se puede diseñar filtros pasa bajos con frecuencias de corte bajas; de esta manera garantizamos disminuir el ruido de altas frecuencias presentadas en el sistema, así como los de bajas frecuencias que se puedan presentar por factores externos.

Figura 19. Circuito para adecuar y filtrar la señal del sensor



2.1.3 Sistema óptico Para un buen desarrollo del proyecto es importante el contar con buenos elementos ópticos para su ejecución. Debido al elevado costo de estos instrumentos, se decide diseñarlos y fabricarlos para obtener un interferómetro económico pero confiable.

Con la base en metal y su marco en madera para que sea más pesada, se sitúa debajo de esta una espuma de 10cm de ancho para reducir las vibraciones.

2.1.3.1 Bases Las bases constan de un juego de engranes y perillas, que permiten ajustar cuidadosamente la posición y orientación de los espejos; debido a que se debe orientar muy bien la posición de los espejos para lograr que converjan los dos haces de luz y así crear el interferograma. El haz de luz combinado se hace pasar por un lente que incrementa su tamaño, así se puede ubicar el sensor y se puede realizar la adquisición. En la Figura 20 se muestran las bases para los espejos, en la Figura 21 se muestra la base para el divisor de haz y en la Figura 22 se muestra la base para el láser.

Figura 20. Bases del interferómetro para los espejos

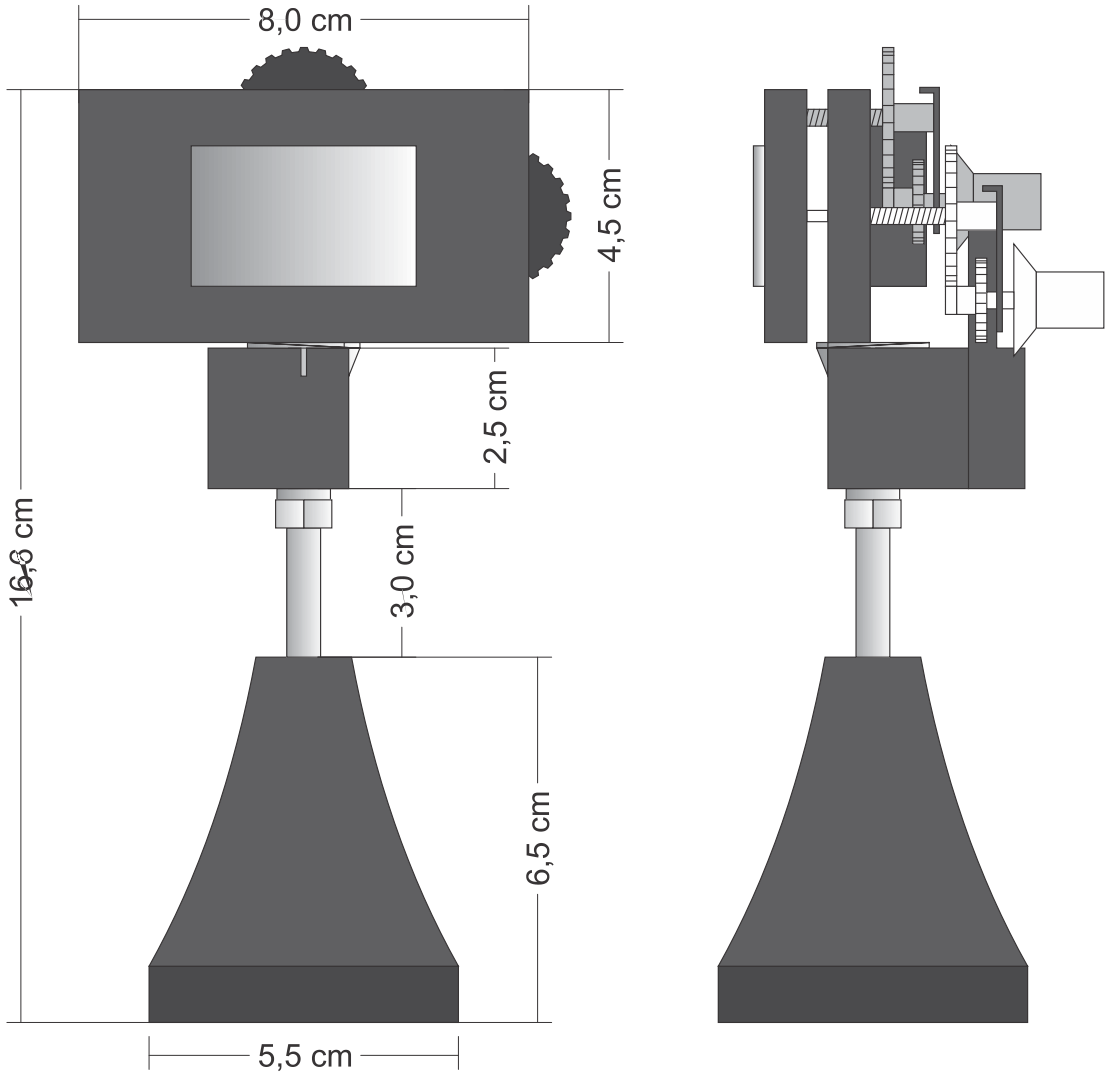


Figura 21. Base del interferómetro para el divisor de haz

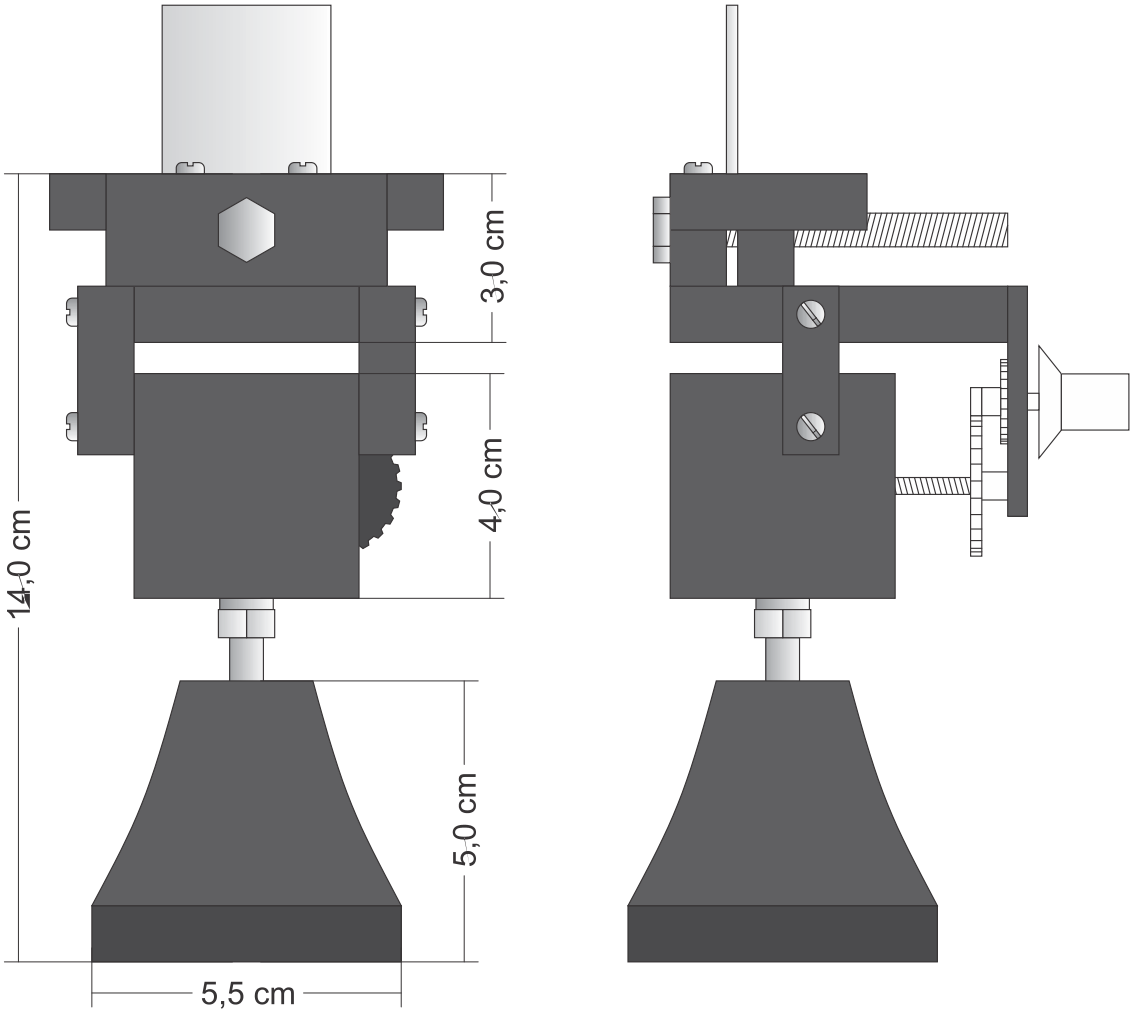
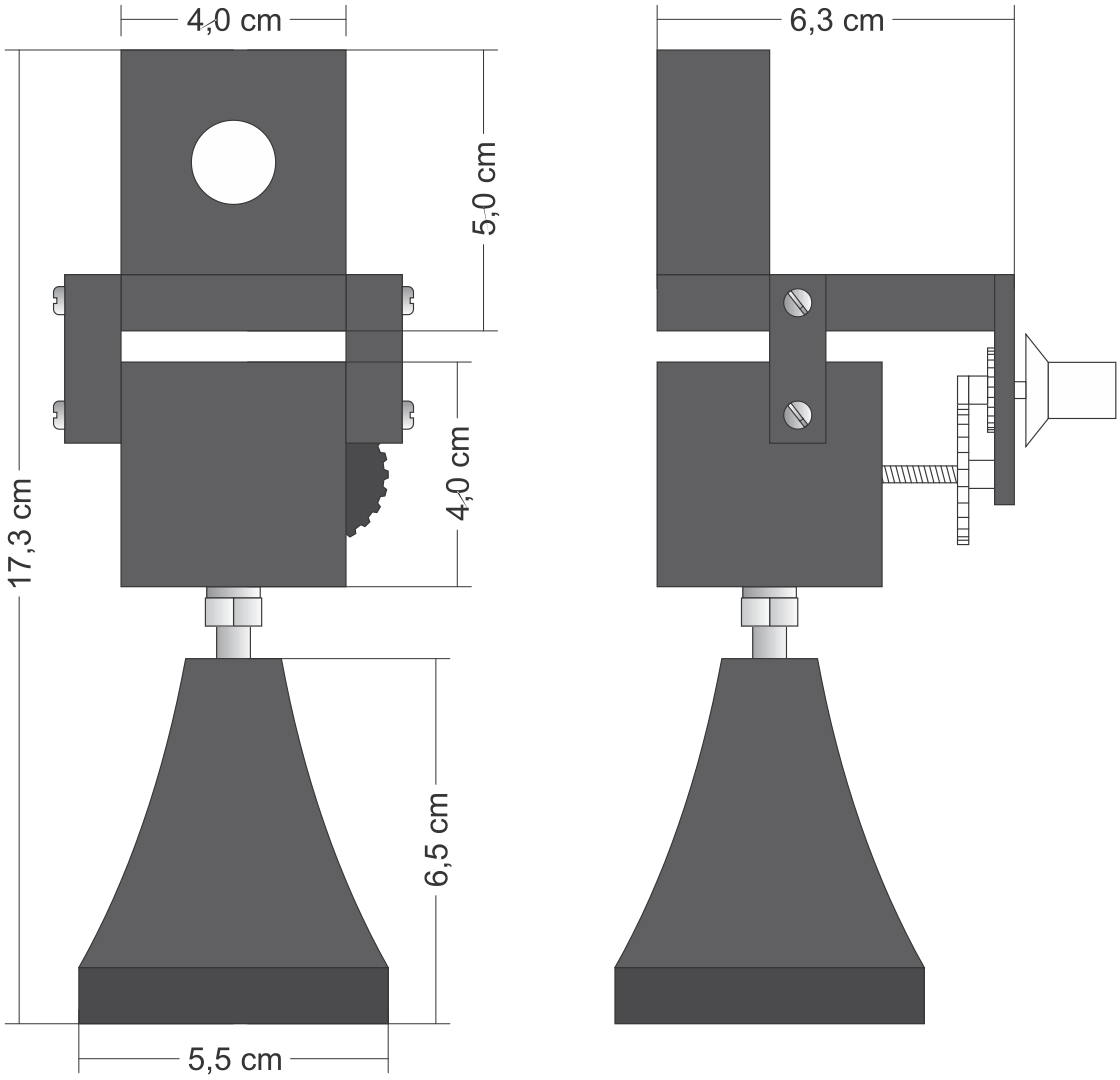


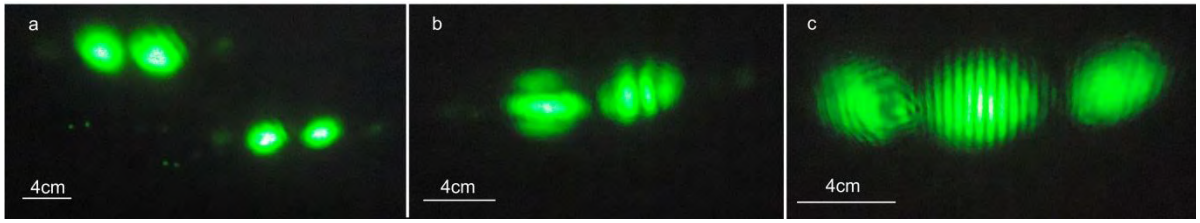
Figura 22. Base del interferómetro para el láser



2.1.3.2 Espejos Los espejos que se pueden utilizar en el interferómetro pueden ser los comerciales; simplemente hay que tener en cuenta que el tamaño del espejo que se sitúa sobre el disco, sea menor al tamaño del disco piezoeléctrico.

Como divisor de haz, puede ser utilizado un semi espejo; también se puede utilizar un vidrio transparente sin ninguna tonalidad de algún color en específico. Solo hay que tener en cuenta que por el grosor del vidrio, se generan 2 puntos del láser al pasar por éste; esto no afecta en nada al interferómetro como tal y no altera los patrones de interferencia ni su comportamiento; solo se tiene un punto adicional, en lugar de uno, en la proyección de los puntos del láser después de ser ampliados con el lente. Esto se observa mejor en la Figura 23.

Figura 23. Puntos proyectados por el interferómetro, a) Puntos proyectados por los dos espejos sin hacerlos converger en un mismo punto, b) Puntos proyectados por los dos espejos haciendo converger los 2 puntos, c) Puntos proyectados por los dos espejos haciendo converger solo un par de ellos



2.1.3.3 Láser El láser seleccionado para el interferómetro, es uno de la fabrica China CNI, la referencia de éste es PGL-FS-532. Este láser genera un haz de luz de color verde, con una longitud de onda de 532±1 nm; tiene una potencia de salida que varía entre los 10 y 80 mW, es importante no tener un láser más potente, ya que puede calentar y/o dañar los elementos del interferómetro; estas y otras características del láser se muestran en la Tabla 1.

Tabla 1. Especificaciones láser CNI PGL-FS-532

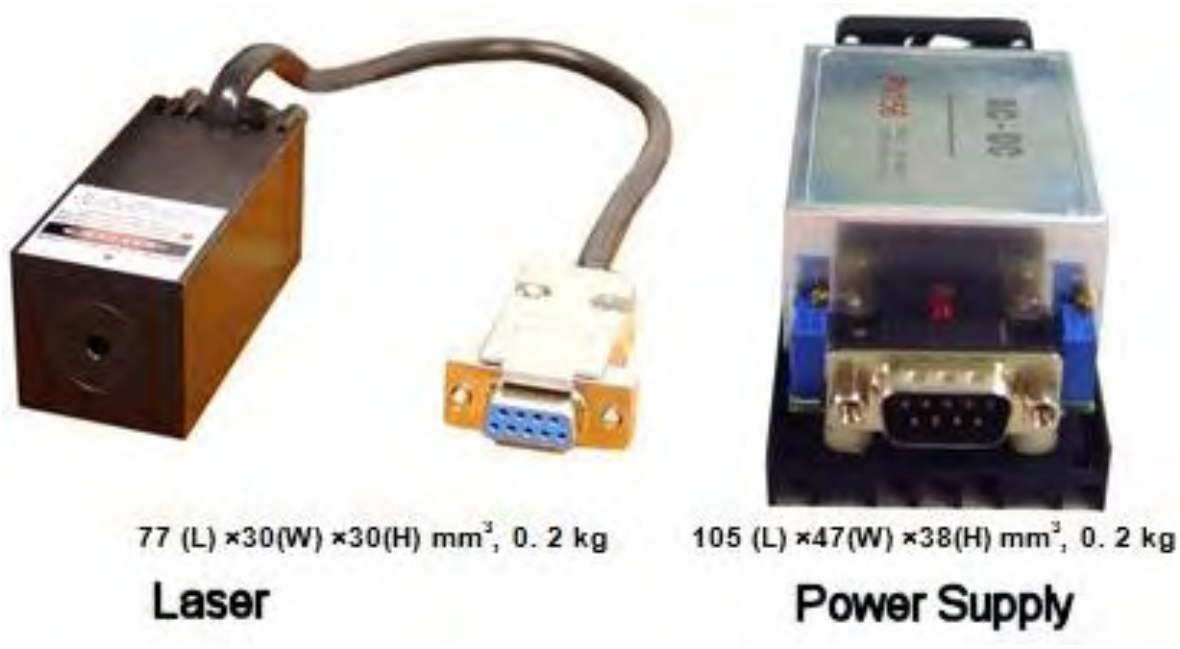
Especificación	Valor
Longitud de onda (nm)	532 ± 1
Potencia de salida (mW)	10 ~ 80
Modo transversal	TEM00
Estabilidad de potencia (rms, mas de 4 horas)	<5%, <10%, <20%
Tiempo de calentamiento (minutos)	<10
Diámetro del haz en la abertura (mm)	~ 1.5
Altura del haz desde la placa base (mm)	15
Temperatura de funcionamiento (°C)	10 ~ 35
Fuente de alimentación (90-264 VAC)	PSU-V-OEM
Tiempo estimado de vida (horas)	5000
Garantía	1 año

* PSU-V-OEM es la referencia de la fuente de alimentación, que es suministrada por el fabricante

PGL-FS-532/10~80mW. CNI Changchun New Industries Optoelectronics Tech. Co..Ltd.[citado en 2012-12-21] Disponible en internet: http://www.cnilaser.com/PDF/PGL-FS-532.pdf

El láser seleccionado se muestra en la Figura 24.

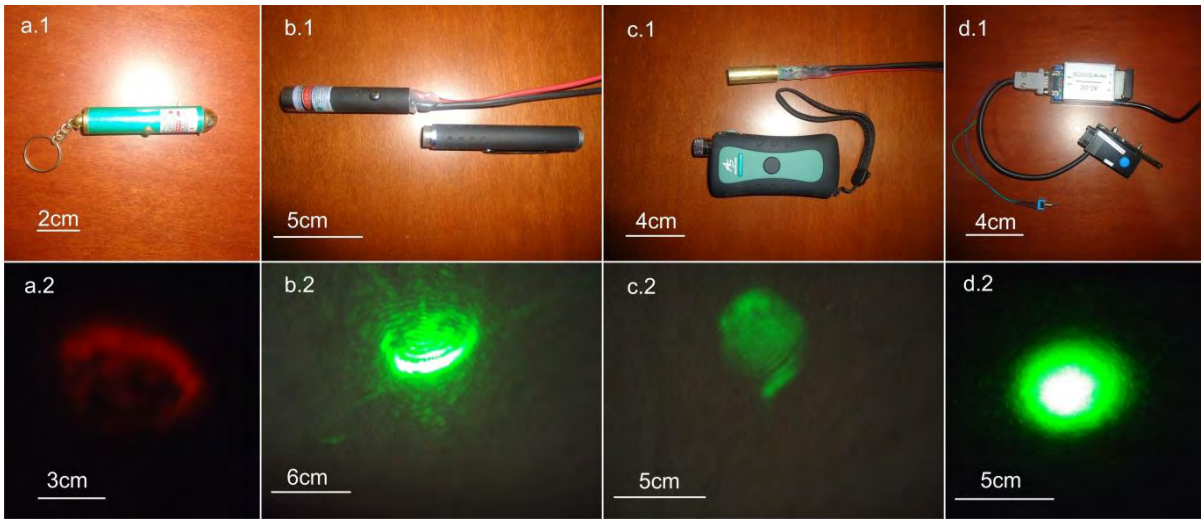
Figura 24. Láser CNI PGL-FS-532, fuente de alimentación PSU-V-OEM



PGL-FS-532/10~80mW. CNI Changchun New Industries Optoelectronics Tech. Co.Ltd.[citado en 2012-12-21] Disponible en internet: http://www.cnilaser.com/PDF/PGL-FS-532.pdf

Es importante tener un láser con poco error en su longitud de onda. El error en la longitud de onda representa un error en el algoritmo para determinar el desplazamiento; ya que en el interferograma, el desplazamiento en el disco se determina mediante la longitud de onda del láser, como se observó en las ecuaciones anteriores. También un buen láser tiene un punto constante y sin variaciones; los láser económicos no proyectan un punto totalmente redondo, sino que en su lugar proyectan una franja o un punto con mayor concentración en un extremo, como se muestra en la Figura 25.

Figura 25. Distintos tipos de láser con su punto proyectado ampliado mediante una lupa, a) Láser comercial de juguete color rojo, b) Apuntador láser color verde tipo bolígrafo, c) Apuntador láser color verde d) Láser CNI PGL-FS-532



El láser comercial de juguete rojo, presenta un error en su longitud de onda de ± 25 nm, cambiando así su longitud de onda entre 630 y 680 nm; al ser un cambio tan grande, el interferograma no es fácilmente apreciable, por esto no es muy confiable su medición. El apuntador láser verde tipo bolígrafo, presenta un error en su longitud de onda de ± 10 nm, cambiando así la longitud de onda entre 522 y 542 nm; no es un cambio muy grande, pero en el interferograma, en algunas ocasiones, se observa pequeñas vibraciones debido a este error; debido a esto se decide utilizar el láser CNI PGL-FS-532, que solo tiene ± 1 nm de error en su longitud de onda.

2.1.4 Pruebas del interferómetro construido Una vez construido el interferómetro es fundamental saber su confiabilidad; por esto, la caracterización de éste es fundamental.

Para poder realizar la caracterización, se necesita un dispositivo confiable que permita desplazar uno de los espejos del interferómetro una distancia establecida; para comparar el resultado del desplazamiento medido, con el aplicado al espejo. Con los resultados, se puede saber si:

- Se cumple la ecuación de Lloyd y Paetkau[1] propuesta para determinar el desplazamiento.

- Los patrones de interferencia del interferograma se comportan de acuerdo a lo estudiado.

- Si los algoritmos propuestos son correctos.

Para realizar la caracterización del interferómetro construido, se decide utilizar el microscopio de fuerza atómica Asylum Research MFP-3D-BIO de la Universidad de los Andes. Se acopla el montaje del interferómetro, al AFM como se muestra en la Figura 26; para poder desplazar con el mecanismo del AFM, una de las bases del interferómetro con su espejo.

Este AFM cuenta con un actuador piezoeléctrico, que mueve la muestra en el eje X y en el eje Y; sobre este actuador, se coloca una de las bases del interferómetro para desplazarla en el eje X, realizando varios barridos a diferentes velocidades y distancias.

La adquisición se realiza mediante los equipos con los que cuenta el AFM, obteniendo una imagen de franjas paralelas parecida al interferograma; con el software del AFM, se extraen los datos correspondientes a una de las pruebas realizadas. Se analizan los datos para los desplazamientos de 10 µm; de estos se

[1] LLOYD, Samantha y PAETKAU, Mark. Characterization of a Piezoelectric Buzzer Using a Michelson Interferometer. Thompson Rivers University, Kamloops, BC, Canadá

selecciona uno, para realizar un estudio detallado, obteniendo los resultados mostrados en la Figura 27.

Figura 26. Montaje del interferómetro en el microscopio de fuerza atómica Asylum Research MFP-3D-BIO de la Universidad de los Andes

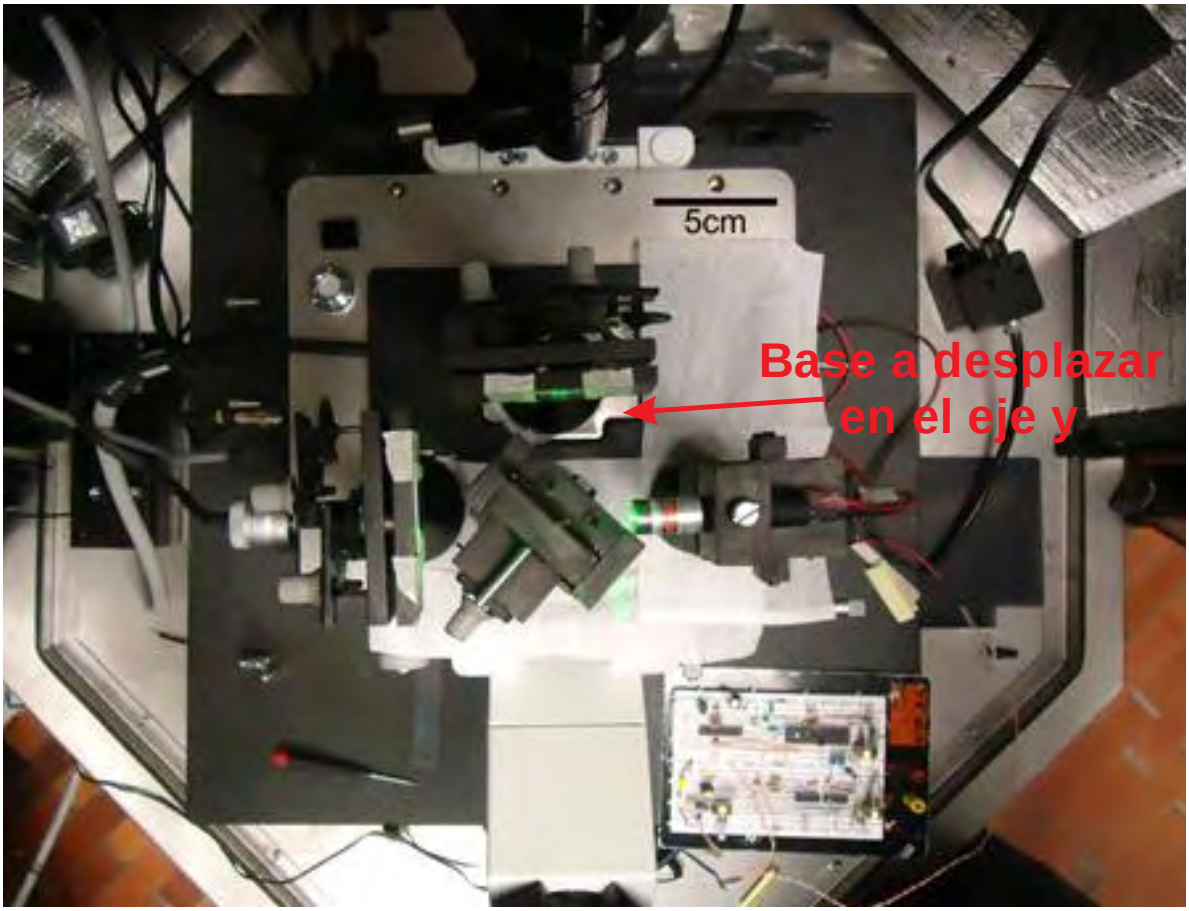
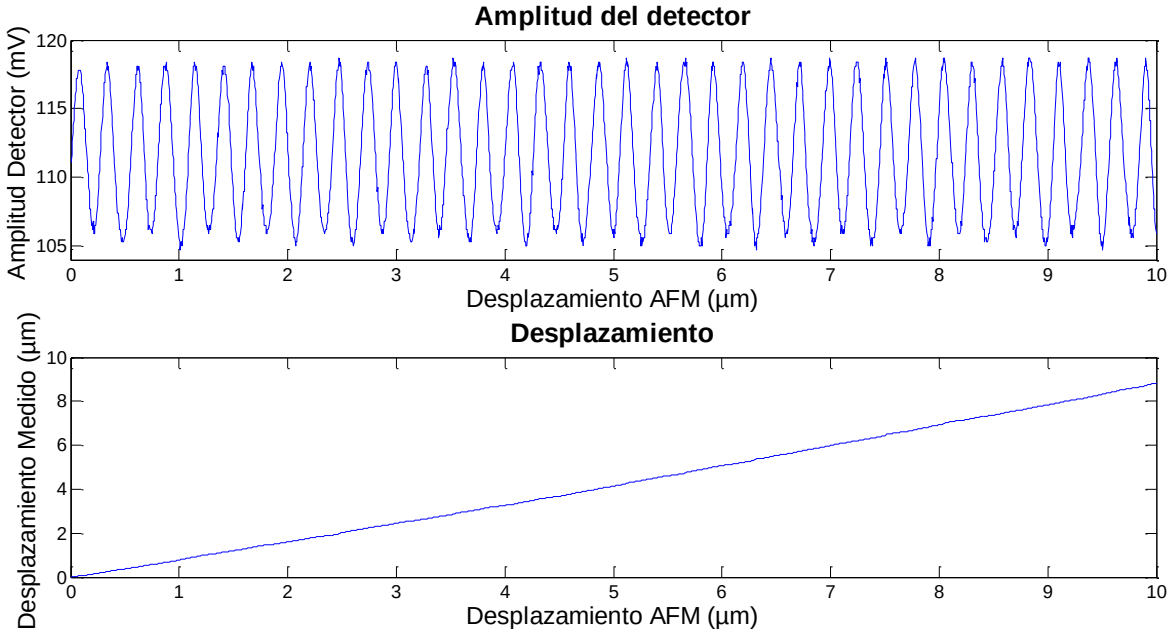


Figura 27. Caracterización del interferómetro construido con el AFM Asylum Research MFP-3D-BIO de la Universidad de los Andes



Teniendo como resultado final, según la grafica inferior, que el espejo se desplazó 8,823 µm; obteniendo un error relativo del 11,77%, que es muy alto, por lo tanto, se decide contar manualmente las longitudes de onda, de la gráfica superior, para verificar el desplazamiento que debería indicar la medición. Se cuenta un total de 37,7 franjas, que al remplazar en la ecuación de Lloyd y Paetkau[1], se obtiene un desplazamiento igual a 10,0282 µm; éste genera un error relativo del 0,282%. Esto indica que el sistema si funciona; además que el interferómetro construido es un dispositivo muy precisa para medir desplazamientos a nano y micro escalas.

Para evitar el error presentado, solo se debe mejorar el sistema. Con los datos obtenidos se procede a analizar dónde se encuentra el error; se percibe que la

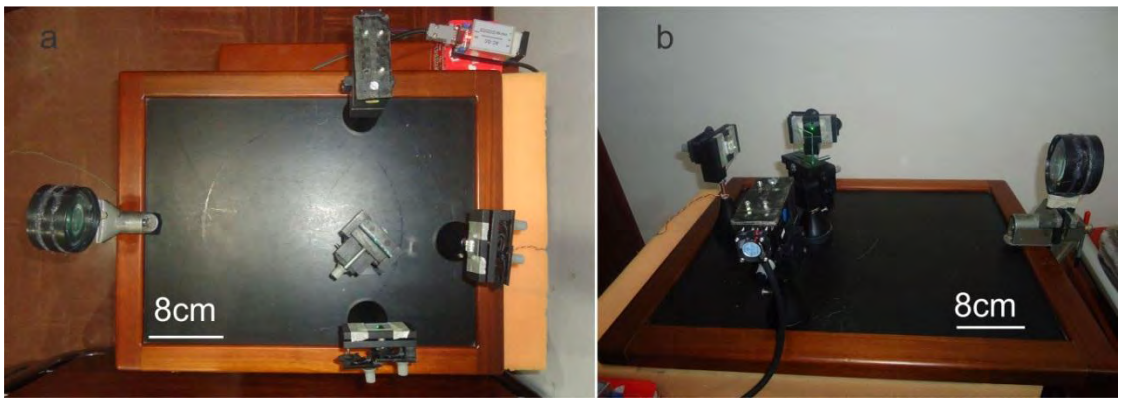
[1] LLOYD, Samantha y PAETKAU, Mark. Characterization of a Piezoelectric Buzzer Using a Michelson Interferometer. Thompson Rivers University, Kamloops, BC, Canadá

amplitud de las ondas captadas por el sensor no es constante; esto debido a varios factores como al láser utilizado, que es un apuntador láser verde tipo bolígrafo, que presenta un error en su longitud de onda de ±10 nm; También es posible este error debido a que se esté presentando un ruido en el sistema del sensor; por esto se decide utilizar un cable blindado para transmitir su señal; además de adecuarle una serie de filtros, explicados anteriormente; todo esto con el fin que la amplitud de la onda, sea siempre constante, tanto en los picos inferiores como superiores y así garantizar una medición mas precisa; tener una frecuencia de muestro, significativamente mayor a la frecuencia de la onda, obtenida por el sensor; esto con el fin de garantizar capturar, los máximos y mínimos de las franjas del interferograma; así el algoritmo de conversión de la onda seno a desplazamiento no tendrá un error tan grande.

Finalmente, con todas las correcciones realizadas, como se indica en los circuitos mostrados en este proyecto; el sistema de adecuación de señal del sensor con sus respectivos filtros, y utilizando el láser CNI PGL-FS-532, se procede a realizar la medición del desplazamiento del disco piezoeléctrico; y posteriormente implementar el controlador para corregir los problemas de histéresis y arrastre.

El interferómetro final se muestra en la Figura 28.

Figura 28. Interferómetro final, a) Vista superior, b) Vista lateral



3. RESULTADOS

Es importante tener en cuenta ciertos aspectos para realizar la caracterización del disco:

- El primero es evitar gente caminando o factores que puedan crear vibraciones en el interferómetro; también los ruidos provocados por dispositivos multimedia, parlantes, televisores, todo lo que provoque sonidos muy fuertes.

- Se debe tener cuidado con la intensidad del haz de luz que llega al interferograma, ya que si es muy fuerte, éste puede saturar el sensor; esto se puede calibrar haciendo una medición con el sistema y observando sus resultados en la grafica que genera la interfaz, sobre los datos adquiridos por el sensor.

- El interferómetro debe estar ubicado sobre una superficie firme para evitar vibraciones. Se debe evitar que en el cuarto ocurran cambios de luminosidad, como el abrir una puerta, ventana o cortina; el encender o apagar una luz. También es importante que nada interfiera entre el interferómetro y el lugar donde se ubica el sensor.

Las recomendaciones anteriores, son necesarias para obtener un error menor al 1% como se especifica en los alcances; de lo contrario la medida puede fallar y los cálculos realizados por el sistema pueden ser erróneos.

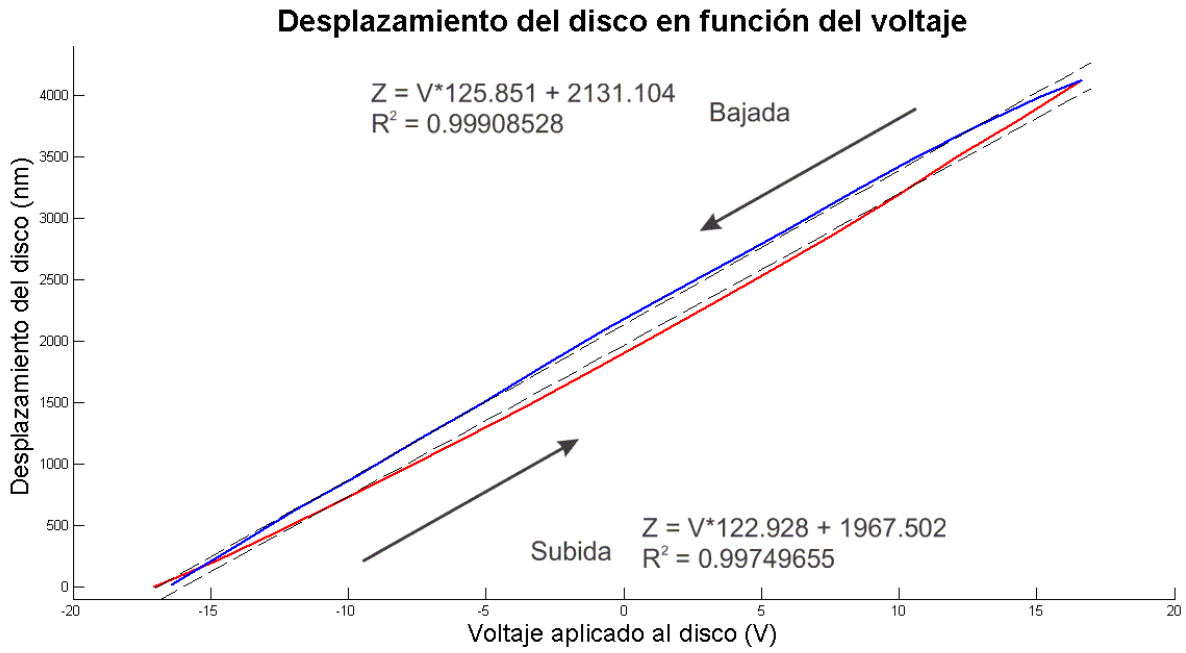
El costo de los materiales para el interferómetro: base, bases espejos, base divisor de haz; es de aproximadamente de $300.000, el láser tiene un costo de $750.000, los materiales para la fabricación de los circuitos un costo de $150.000 aproximadamente.

El sistema puede indicar el valor del desplazamiento, con una resolución 0.1 nm; pero se ha limitado a mostrar valores con resoluciones de 1 nm; debido al error de la longitud de onda del láser, las posibles perturbaciones que no se puedan corregir por completo en el momento de realizar la medida, entre otros; por lo cual siempre existirá una incertidumbre en la medida.

3.1 MEDICIÓN DEL DESPLAZAMIENTO DEL DISCO PIEZOELÉCTRICO EN FUNCIÓN DEL VOLTAJE

Teniendo en cuenta las recomendaciones, se realizan varias mediciones de desplazamiento del disco en función del voltaje sin activar el control. La medida se realiza para 2 desplazamientos continuos, primero de forma ascendente e inmediatamente después de forma descendente; este procedimiento se repite varias veces; se observa que el comportamiento del disco es similar en las diferentes pruebas y se analizan los datos obtenidos. Se muestra a continuación el estudio para una de las pruebas; se analiza y grafica solamente 16 de los datos, para no saturar el dibujo; obteniendo como resultado los datos presentados en la Figura 29.

Figura 29. Desplazamiento del disco piezoeléctrico en función del voltaje, sin control para dos desplazamientos continuos, primero ascendente (rojo) y enseguida descendente (azul), las líneas punteadas corresponde a la regresión lineal de cada desplazamiento



Como se observa en la imagen, el desplazamiento del disco tiende a ser lineal. Se efectúa los cálculos para la regresión lineal y se obtiene una R^2 de 0.99749655 para el desplazamiento ascendente y una R^2 de 0.99908528 para el desplazamiento descendente. R es el índice de correlación; este índice mide el grado de ajuste de los datos experimentales a la recta de la regresión lineal. Se procede a calcular el error de la siguiente manera:

- Se obtiene la ecuación de la recta correspondiente a la regresión lineal.
- Se establece como dato teórico el correspondiente a la regresión lineal y como dato experimental el dato medido con el interferómetro, se calcula el

error absoluto y con éste el error relativo de cada punto de la muestra final (los 16 datos).

- Una vez obtenido el error relativo de cada uno de los puntos se procede a promediar estos valores, obteniendo como error promedio un valor de 11.22% para el desplazamiento ascendente y 6.78% para el desplazamiento descendente.

Los datos analizados se presentan en la Tabla 2.

Tabla 2. Datos de desplazamientos del disco piezoeléctrico en función de voltaje, para dos desplazamientos continuos

Desplazamiento ascendente		**Desplazamiento descendente**	
Voltaje (V)	Desplazamiento (nm)	Voltaje (V)	Desplazamiento (nm)
-17,065	1	-16,413	11
-14,818	206	-14,166	316
-12,571	449	-11,919	625
-10,323	691	-9,672	902
-8,076	944	-7,424	1197
-5,829	1199	-5,177	1485
-3,582	1463	-2,930	1794
-1,335	1736	-0,683	2092
0,913	2012	1,564	2372
3,160	2297	3,811	2645
5,407	2580	6,059	2919
7,654	2874	8,306	3206
9,901	3178	10,553	3487
12,149	3503	12,800	3744

Se realizan 6 desplazamientos consecutivos, intercalados entre ascendente y descendente, para observar el *drift* o arrastre; los resultados se pueden observar en la Figura 30.

Figura 30. a) Efecto de drift o arrastre del disco piezoeléctrico para 6 desplazamientos consecutivos, intercalados entre ascendente y descendente; líneas rojas: desplazamientos ascendentes, líneas azules: desplazamientos descendentes; línea discontinua: primer desplazamiento, línea punteada: segundo desplazamiento, línea de punto y línea: tercer desplazamiento; b) Detalle de la gráfica

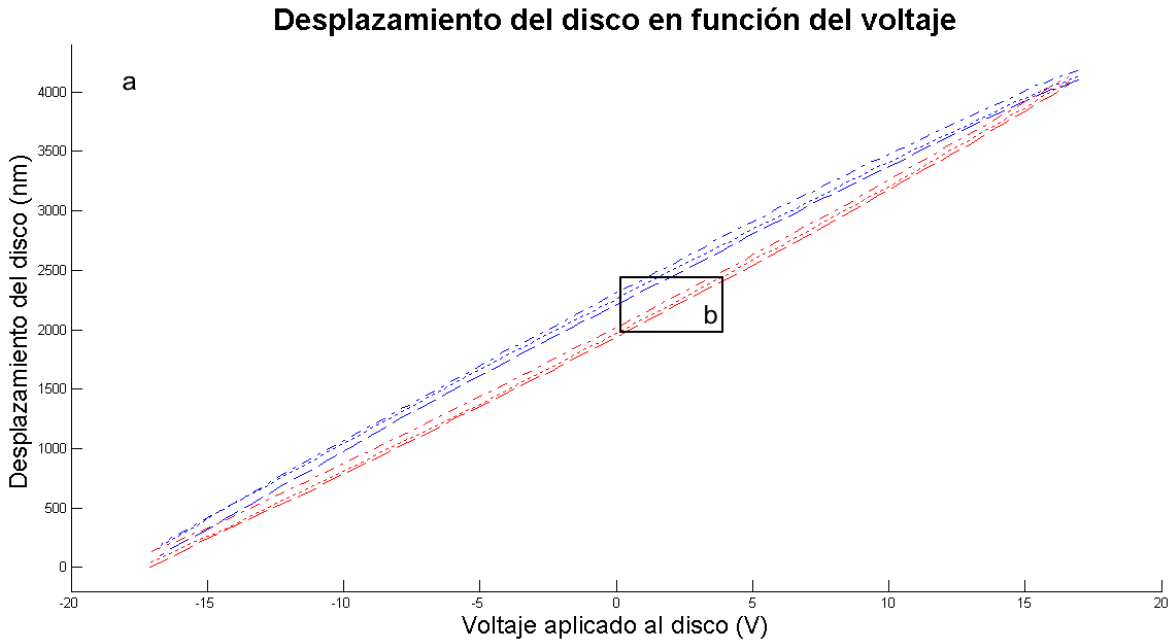
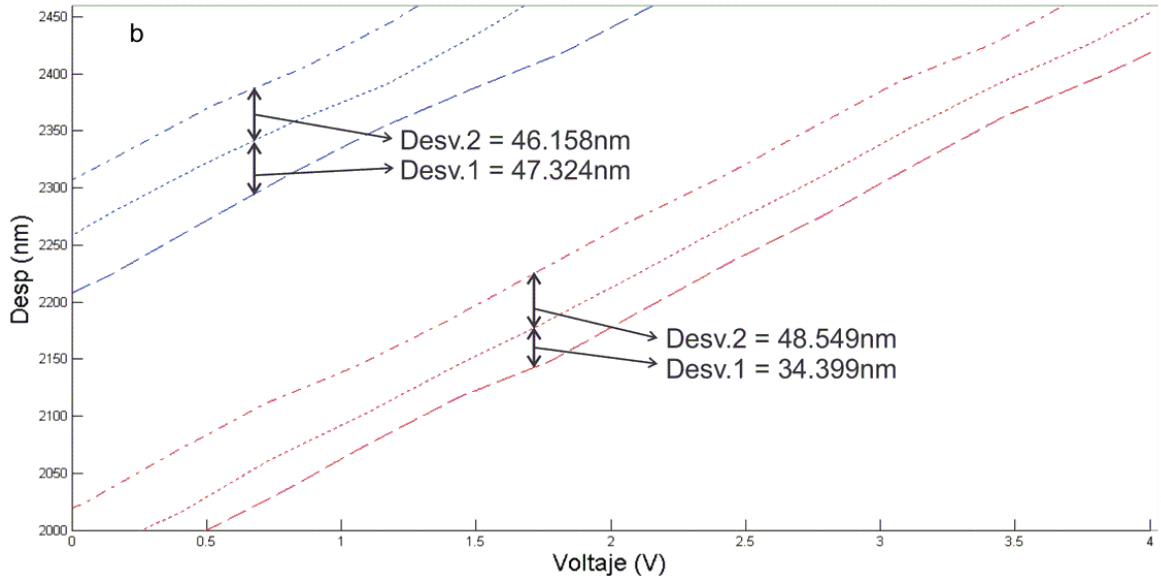


Figura 30. (Continuación)

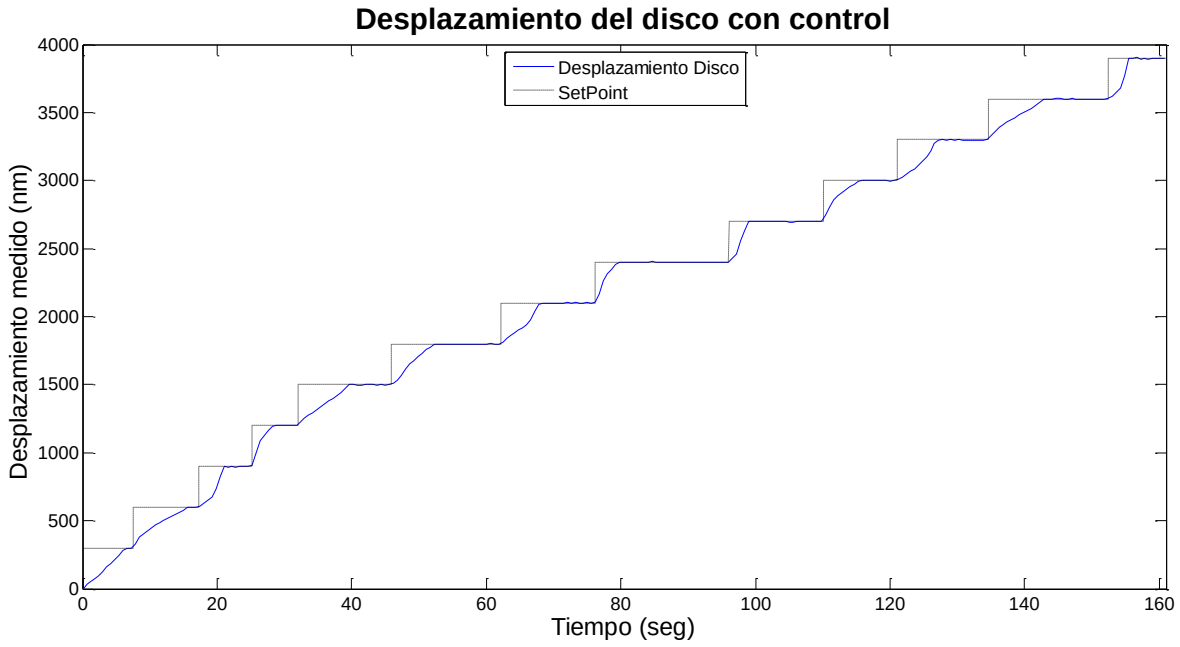


3.2 CARACTERIZACIÓN DEL DISCO PIEZOELÉCTRICO CON CONTROLADOR IMPLEMENTADO

Para realizar la corrección de histéresis y arrastre presentado por el disco, se procede a realizar los desplazamientos con el control PI implementado. Debido a la complejidad del modelo matemático del comportamiento del disco piezoeléctrico, como se explicó anteriormente; se decide sintonizar el control PI mediante prueba y error; obteniendo que para desplazamientos menores a 1000 nm es recomendable utilizar kp = 0.001, y ki = 0.003, si los desplazamientos son mas altos a 1000 nm hasta 2000 nm se recomienda utilizar la mitad del valor de las constantes mencionados anteriormente, si el valor del desplazamiento es mayor a 2000 nm se recomienda utilizar la tercera parte de dicho valor en las constantes; esto con el fin de no efectuar desplazamientos muy rápidos, para evitar suprimir con los filtros análogos, la onda seno captada por el sensor.

Se establece las condiciones iniciales del sistema y se determina un SetPoint de 300 nm. El sistema realiza la función del controlador cada 20 milisegundos; una vez alcanza el valor deseado, se incrementa el SetPoint en 300 nm y así sucesivamente; se analizan los valores cuando el sistema ya ha alcanzado el SetPoint, obteniendo la grafica presentada en la Figura 31, correspondiente al desplazamiento solicitado y el medido; se realizó un muestreo de 50Hz, un total de 8051 datos, la muestra total tardó 161,02 segundos. Los voltajes entre cada desplazamiento que se quiera realizar, van a ser diferentes para los mismos SetPoint, debido a los problemas presentados por histéresis y arrastre en el disco piezoeléctrico.

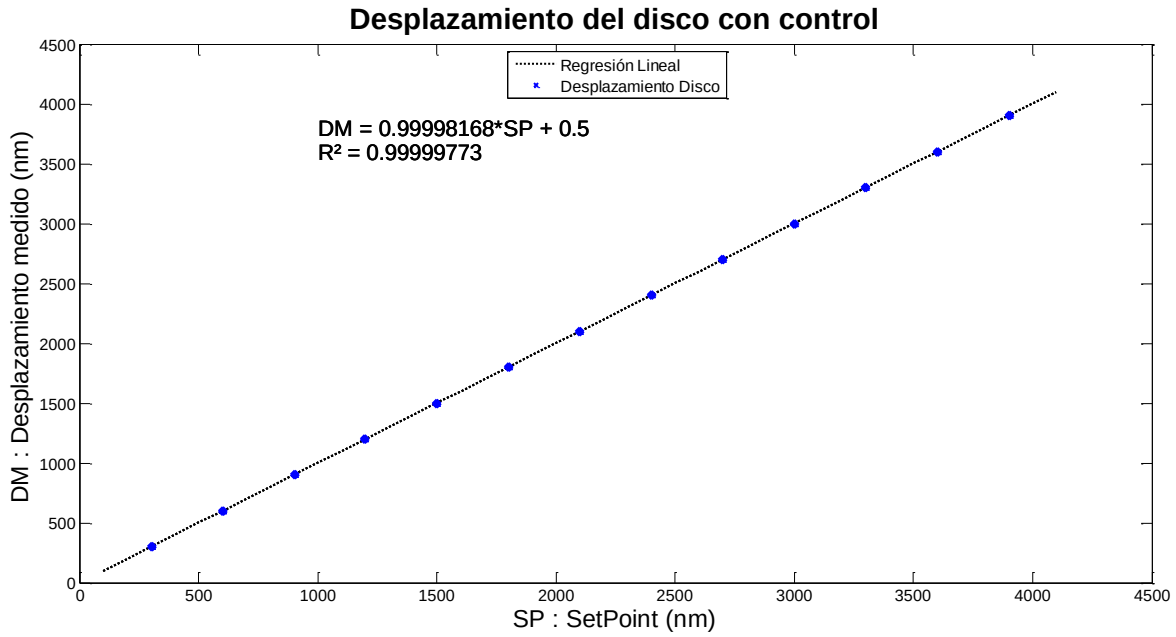
Figura 31. Desplazamiento del disco piezoeléctrico, con corrección de histéresis y arrastre; desplazamiento del disco piezoeléctrico contra el SetPoint solicitado (azul); la línea punteada negra corresponde al SetPoint



Finalizado este proceso, se hace un análisis a la señal; se selecciona un dato para cada cambio de SetPoint, cuando el desplazamiento del disco piezoeléctrico

medido ya ha alcanzado el valor solicitado; obteniendo la grafica presentada en la Figura 32.

Figura 32. Desplazamiento del disco piezoeléctrico, con corrección de histéresis y arrastre; desplazamiento del disco piezoeléctrico contra el SetPoint solicitado (puntos azules); la línea punteada negra corresponde a la recta de la regresión lineal



Se efectúa los cálculos para la regresión lineal y se obtiene una R^2 de 0.99999773; donde R es el índice de correlación, se procede a calcular el error de la siguiente manera:

- Se obtiene la ecuación de la recta correspondiente a la regresión lineal como se observa en la Figura 32.

- Se establece como dato teórico el correspondiente a la regresión lineal y como dato experimental el dato medido con el interferómetro; se calcula el error absoluto y con éste, el error relativo de cada punto de la muestra.

- Una vez obtenido el error absoluto de cada uno de los puntos se procede a promediar estos valores, obteniendo como error promedio un valor de 0.1299%.

Los datos analizados se presentan en la Tabla 3.

Tabla 3. Datos del desplazamiento del disco piezoeléctrico con el controlador PI implementado

SetPoint (nm)	**Desplazamiento medido (nm)**
300	302
600	600
900	904
1200	1199
1500	1499
1800	1799
2100	2099
2400	2399
2700	2700
3000	3001
3300	3299
3600	3601
3900	3903

Los resultados de los errores del proyecto y R^2 de la regresión lineal, se pueden analizar en la Tabla 4.

Tabla 4. Errores de los diferentes desplazamientos

Desplazamiento	Error	R^2
Desplazamiento ascendente	11.22%	0.99749655
Desplazamiento descendente	6.78%	0.99908528
Desplazamiento con control	0.1299%	0.99999773

3.3 ELABORACIÓN DE ARTÍCULO PARA REVISIÓN Y PUBLICACIÓN

Finalmente con los resultados del proyecto se elabora el artículo para publicación internacional, que se presenta en el ANEXO D. El artículo se muestra sin el titulo del anexo, para poder posicionarlo con los márgenes correspondientes dentro de la hoja; también se omite en estas hojas del anexo, la numeración de las páginas, de tal manera que se observe el artículo tal como se presentaría.

4. CONCLUSIONES

Se construyó un interferómetro a bajo costo, que permite realizar la medición del desplazamiento a micro y nano escalas en uno de sus espejos; con precisiones menores al 1%, siempre y cuando se tengan en cuenta las recomendaciones dadas.

Los discos piezoeléctricos cambian su comportamiento y su desplazamiento con respecto al voltaje, a la repetitividad de las medidas debido a las histéresis y arrastre; también por la temperatura y los ruidos presentes en el ambiente.

La histéresis y el arrastre se pueden corregir implementando un control en lazo cerrado, que genera un sistema de desplazamiento más confiable y preciso; pasando de errores de un 11.22% para desplazamientos ascendentes y 6.78% para desplazamientos descendentes, a un error menor al 1% sin importar el sentido del desplazamiento.

Para obtener resultados más confiables es necesario contar con un láser monocromático y coherente con muy poco error en su longitud de onda; por otra parte se puede mejorar esto, con la aplicación de filtros análogos al adecuar la señal del sensor.

Los interferómetros profesionales son muy costosos, pero se pueden fabricar según la necesidad que se tenga; obteniendo un sistema confiable y con una

reducción de costos muy significativa dependiendo de los parámetros establecidos para el interferómetro como la base, las bases de los espejos, el láser y los demás componentes.

Es importante tener en cuenta las recomendaciones dadas para obtener resultados más confiables; ya que una intervención externa al sistema, puede efectuar errores en los cálculos del DSPIC, al interpretar los datos modificados debido a estos factores; tales como cambios en la luminosidad del cuarto al abrir una puerta o ventana, al encender una luz; vibraciones externas al interferómetro, entre otras.

El control PI implementado, permitió corregir los errores de histéresis y arrastre del disco piezoeléctrico; además se puede sintonizar el control mediante prueba y error.

Se puede realizar los algoritmos, cálculos y acción del control del disco en tiempo real, como se muestra en una de las rutinas; la mayor parte del proceso es realizada por el DSPIC; con unas pequeñas modificaciones se puede prescindir del PC, si no se desea observar los gráficos y simplemente deseamos controlar el desplazamiento del disco.

5. TRABAJO POR REALIZAR

Este proyecto se puede tomar de base para permitir realizar más investigación en el área de la nanotecnología; a continuación se menciona una serie de sugerencias sobre el trabajo futuro que se puede realizar; así como las posibles mejoras que se pueden efectuar al sistema diseñado.

Realizar un estudio matemático más profundo sobre el comportamiento de los materiales piezoeléctricos, no solo de los discos; con el fin de obtener un modulo matemático más detallado; permitiendo así, simular los diferentes componentes del proyecto, como el controlador o los filtros; y poder tener una representación en espacio de estados del sistema.

Mejorar el sistema mecánico de las bases del interferómetro; adecuando servomotores, a los juegos de engranes, para posicionar los espejos; también se puede implementar un sistema electrónico, que automáticamente posiciones los puntos reflejados por cada espejo para crear el interferograma.

Diseñar e implementar diferentes tipos de controladores, para mejorar la acción de control en el disco piezoeléctrico; con el fin de encontrar el mejor diseño para reducir al máximo los errores de histéresis y arrastre, y mejorar la respuesta del controlador.

Crear una estructura, acoplando diferentes discos piezoeléctricos, para crear un posicionador fino en tres dimensiones; también se puede simplificar el tamaño de los elementos para hacerlo más portable.

El sistema ha sido diseñado para realizar la mayor parte de las acciones mediante el DSPIC; el computador, se utiliza principalmente para observar los resultados de las mediciones y analizar posteriormente los datos; se puede implementar un teclado y una pantalla, para prescindir del PC y tener un sistema mas portable; se

puede implementar en el código del DSPIC, una rutina para guardar los datos en su memoria o guardarlos en una memoria USB, para después analizarlos en el computador.

Se puede cambiar el sensor CNY70 por una cámara, que pueda capturar todo el interferograma; de esta forma se puede evitar la parte inicial del programa, que establece las condiciones iniciales; ya que con los datos de todo el interferograma, se puede saber inmediatamente el máximo y mínimo, de la intensidad en las franjas creadas en el interferograma.

Diseñar nuevos algoritmos, para convertir la onda seno en el desplazamiento del disco; evitando disminuir la velocidad del desplazamiento del disco, para aumentar la frecuencia de muestreo y así realizar los cálculos sin errores tan significativos.

Ubicar un disco piezoeléctrico pequeño sobre uno más grande; con el fin de aumentar el rango de desplazamiento del sistema, sin perder la resolución de desplazamiento del sistema actual.

BIBLIOGRAFÍA

BUNCH, Bryan H y HELLEMANS, Alexander. The history of science and technology. Houghton Mifflin Harcourt. Abril 2004. 695 p.

CADY, Walter Guyton. Piezoelectricity: An Introduction to the Theory and Applications of Electromechanical Phenomena in Crystals, Dover Press. 1964. 822 p.

DORF, Richard y BISHOP, Robert. Sistemas de control moderno. Traducido por Sebastián Dormido Canto y Raquel Dormido Canto. Revisión técnica por Sebastian Dormido Bencomo. 10 ed. Madrid: Pearson Prentice Hall. 2005. 928 p. ISBN: 84-205-4401-9

Fleming, Andrew J. y Leang, Kam K. Evaluation of Charge Drives for Scanning Probe Microscope Positioning Stages. En: American Control Conference (11-13, Junio, 2008). Westin Seattle Hotel, Seattle, Washington, USA

G. Binnig, C.F. Quate y Ch. Gerber. Atomic force microscope. Physical review letters. 3 Marzo 1986. vol. 56, no. 9 [citado en 2012 12 31], p. 930-933

HARIHARAN, P. Basics of interferometry. 2 ed. USA: Elsevier science, 2007. 226 p. ISBN 0-12-373589-0

HARIHARAN, P. Optical interferometry. 2 ed. USA: Elsevier science, 2003. 351 p. ISBN 0-12-311630-9

Interferometría básica, disponible en la web en http://mecfunnet.faii.etsii.upm.es/difraccion/interferencia/interf_michelson.html

Interferómetro de Michelson, video en línea disponible en http://www.youtube.com/watch?v=e2JtSQkPlnk

Keyence. Ultra High-Speed/High-Accuracy. Laser Displacement Sensor LK-G5000 series. Catálogo por internet, Disponible en internet: http://www.keyence.com/dwn/lkg5000_ka.pdf

Láser, [citado en 2012-12-24] Disponible en internet: http://www.itlalaguna.edu.mx/academico/carreras/electronica/opteca/OPTOPDF5_archivos/UNIDAD5TEMA1.pdf

LLOYD, Samantha y PAETKAU, Mark. Characterization of a Piezoelectric Buzzer Using a Michelson Interferometer. Thompson Rivers University, Kamloops, BC, Canadá

MICHELSON, Albert y Morley, Edward. On the Relative Motion of the Earth and the Luminiferous Ether. *American Journal of Science* 34 (203): 333–345.

Michelson Interferometer Operation. Block Engineering. [citado en 2013-1-2]. Disponible en internet: http://blockeng.com/technology/ftirtechnology.html

PGL-FS-532/10~80mW. CNI Changchun New Industries Optoelectronics Tech. Co..Ltd.[citado en 2012-12-21]. Disponible en internet: http://www.cnilaser.com/PDF/PGL-FS-532.pdf

Principios básicos, [citado en 2012-12-24] Disponible en internet: http://www.itlalaguna.edu.mx/academico/carreras/electronica/opteca/OPTOPDF5_archivos/UNIDAD5TEMA2.pdf

RADHESH. PID Controller Simplified. (en línea). Mayo 11, 2008. [Citado en 2013-12-24]. Disponible en internet: http://radhesh.wordpress.com/2008/05/11/pid-controller-simplified/

RUGE, Ilber Adonayt. Método básico para implementar un controlador digital PID en un microcontrolador PIC para desarrollo de aplicaciones a bajo costo.[citado en 2012-12-21]. Disponible en internet: http://www.edutecne.utn.edu.ar/microcontrol_congr/industria/MTODOB~1.PDF

RUIZ ROSERO, Juan Pablo. Diseño e implementación de un AFM para visualización y fabricación de nanoestructuras. Tesis de maestría. Bogotá: Universidad de los Andes, 2012

SÁNCHEZ, José Oscar, SÁNCHEZ, Domingo Vidal. Estabilización de valores iniciales de un circuito rc con PID. (en linea). Trabajo de la asignatura: Controles de sistemas Automáticos. Universidad Tecnológica de Santiago. Republica Dominicana. 19 de Abril del 2001. [Citado en 2013-01-11]. Disponible en internet: http://www.monografias.com/trabajos84/control-pid/control-pid.shtml

Sistema láser, [citado en 2012-12-24] Disponible en internet: http://www.itlalaguna.edu.mx/academico/carreras/electronica/opteca/OPTOPDF5_archivos/UNIDAD5TEMA3.pdf

Tipos de láser, [citado en 2012-12-24] Disponible en internet: http://www.itlalaguna.edu.mx/academico/carreras/electronica/opteca/OPTOPDF5_archivos/UNIDAD5TEMA4.pdf

WNAG, Yu-Chi, CHEN, Li-Kang y HUNG, Shao-Kang. The Design and Characteristic Study of a 3-dimensional Piezoelectric Nano-positioner. En: SICE Annual Conference (18-21, Agosto, 2010) The Grand Hotel, Taipei, Taiwan

ZHOU, Huixing, BRIAN Henson y BELL, Andrew. Linear piezo-actuator and its applications. The 5th International Conference on Frontiers of Designand Manufacturing (ICFDM 2002), Dalian, China, Vol. 1, pp 16-20. (Julio 2002). [citado en 2013-01-08] Disponible en internet: http://zhouhx.tripod.com/piezopaper.pdf

ANEXOS

ANEXO A. Código Fuente Del Microcontrolador DSPIC30F4013

```
////////////////////////////////////////////////////////////////////////
//     **********         Programa DSPIC30F4013         **********     //
//  Programa para DSPIC30F4013, diseñado por Jose Eduardo Ruiz Rosero  //
//  para el proyecto de investigación denominado:                      //
//                                                                     //
//  "CARACTERIZACIÓN DEL DESPLAZAMIENTO DE UN DISCO PIEZOELÉCTRICO     //
//  EN FUNCIÓN DEL VOLTAJE CON CORRECCIÓN DE HISTÉRESIS Y ARRASTRE     //
//  MEDIANTE UN CONTROL EN LAZO CERRADO, UTILIZANDO INTERFEROMETRÍA."   //
//                                                                     //
//                 Universidad de Nariño 2012 - 2013                   //
//                    San Juan de Pasto - Colombia                     //
////////////////////////////////////////////////////////////////////////

#include <30F4013.h>

#DEVICE ADC=12
#fuses HS2_PLL8,NOWDT,NOPROTECT,DEBUG
//Cristal de 20MHz divido entre 2 y multiplicado por 8=80MHz
#use delay(clock=80M)
#INCLUDE <stdio.h>
#INCLUDE <stdlib.h>
#INCLUDE <math.h>

// Definición de variables ********************************************
int ii,marca1,mandardato=0,cont1=0,mmax=0;
int cuentacontrol=0;
char x,x1,x2,recibido[16];
unsigned int duty,total,lectura,min,max,maxa;
float divi,funa,res,rad,des1,des2,dest,deste;
float kp=0.0003,ki=0.0030,sp,sumres=1;
float Tp;
float Ti;
float error_sum = 0;
float merror_sum;
float error;
float desired_power;
float temp_float;
float setpoint=1500;
float power;
float datoa;
// ********************************************************************


// Definición de puertos **********************************************
#define LEDR PIN_F0   // Pin 30. LED-Rojo
#define LEDA PIN_F1   // Pin 29. LED-Amarillo
#define LEDV PIN_F4   // Pin 28. LED-Verde
```

```
// ********************************************************************


// Definiciones para la comunicación ***********************************
#use rs232(baud=115200, UART1)
#define uart_putc(c) putc(c)
#define uart_getc  getc
#define uart_kbhit kbhit
#define uart_printf  printf
// ********************************************************************


// Programa de interrupcion por timer 1 ********************************

#INT_TIMER1 // Interrupción del timer 1
void timer1_isr(void) {
setup_timer1(TMR_DISABLED);         //deshabilitar interrupción del timer
1
set_timer1(63973);                  //setear timer 1 para interrupción
setup_timer1(TMR_INTERNAL | TMR_DIV_BY_256);    //configurar timer1

      output_toggle(LEDV);          //cambio de estado del LED-Verde
      read_adc(ADC_START_ONLY);     //inicia la conversión del ADC
      lectura=read_adc(ADC_READ_ONLY);    //Guarda el valor leído del ADC
      funa=lectura;
      funa=lectura-res;
      rad=funa/maxa;
      des2=(asin(rad));
      des2=((((des2+1)*0.81)*1000)/(40*0.532));
      if (des1==0){
            dest=0;
      }
      else{
            dest=dest+((abs(des1-des2))*sumres);
      }
      des1=des2;
      marca1=marca1+1;
      deste=lectura;

// Para mandar los datos cada vez que entre a interrupción por Timer1
      if(mandardato==1){
            //mandar el valor del desplazamiento
            lectura=dest+500f;
            x1=make8(lectura,1);
            x2=make8(lectura,0);
            if (x2==10){
                  x2=11;
            }
            if (x2==13){
                  x2=12;
            }
            if (x1==10){
                  x1=254;
            }
```

```
            if (x1==13){
                  x1=255;

            }
            putc(x1);
            putc(x2);

            //mandar el valor del PWM
            x1=make8(total,1);
            x2=make8(total,0);
            if (x2==10){
                  x2=11;
            }
            if (x2==13){

                  x2=12;
            }
            if (x1==10){
                  x1=254;
            }
            if (x1==13){
                  x1=255;
            }
            putc(x1);
            putc(x2);
      }

// Para mandar una vez los datos al entrar a la interrupción por Timer1
      if(mandardato==2){
            //mandar el valor del desplazamiento
            lectura=dest+500f;
            x1=make8(lectura,1);
            x2=make8(lectura,0);
            if (x2==10){
                  x2=11;
            }
            if (x2==13){
                  x2=12;
            }
            if (x1==10){
                  x1=254;
            }
            if (x1==13){
                  x1=255;
            }
            putc(x1);
            putc(x2);

            //mandar el valor del PWM
            x1=make8(total,1);
            x2=make8(total,0);
            if (x2==10){
                  x2=11;
            }
```

```
            if (x2==13){
                  x2=12;
            }
            if (x1==10){
                  x1=254;
            }
            if (x1==13){
                  x1=255;
            }
            putc(x1);
            putc(x2);

            mandardato=0;     // Para deshabilitar el mandar datos y que
                                    // efectivamente solo se mande uno
                                    // (el que se termino de enviar)
            }
//********************** Inicia el controlador **********************//

      datoa=dest;
      error = setpoint - datoa;           //Calcula el error

      if(error>=0){    //Si el desplazamiento esta por encima del
SetPoint
            output_low(LEDA);       //Prende el Led-Amarillo
            sumres=1;
      }
      else{                         // de lo contrario
            output_high(LEDA);      //Apaga el Led-Amarillo
            sumres=-1;
      }

      // Cálculo del termino proporcional
      Tp = Kp * error;

      // Cálculo del termino integral
      error_sum = error_sum + (int32)error;
      merror_sum = (int32)32000f/Ki;

      if(error_sum>=merror_sum)
            error_sum=merror_sum;
      if(error_sum<=0)
            error_sum=0;

      temp_float = error_sum;
      Ti = Ki * temp_float;

      // Caalculo de la potencia deseada
      desired_power = (Tp + Ti);

// set correcto del PWM
      if (desired_power < 0){ //para que no pueda ser menor a 0
            total = 0;
    }
    else if (desired_power > 64000){ // Para que no pueda ser mayor a 64k
```

```
            total = 64000;
      }
        else {
            total = (desired_power*10f)+2570f;  //Adecua el total
        }
        set_pwm_duty(4, total);
        delay_ms(10);
//************************ Fin del controlador ************************//
        }           // Fin de interrupción por Timer1

#int_rda   // Programa de interrupción por puerto serial ***************
void rda_isr(void){
disable_interrupts(int_rda);       //Deshabilita interrupción

if  (kbhit()) {                                 //Si hay dato recibido en
el puerto
            x = getc();                         //Grabar el dato recibido
en x
      }                                               //fin del if

      switch(x) {
//Inicio para cálculos, viene de botón "Adquisición"
            case '1':
                  output_low(LEDA);
                  output_low(LEDV);
                  for(ii=0;ii<2;ii++){              //prende y apaga el
LED-Rojo
                        output_toggle(LEDR);
                        delay_ms(200);
                        }
                  output_high(LEDR);
                  mmax=1;
                  total=35270;
//Coloca 32570 al PWM para posicionar el disco
                  set_pwm_duty(4, total);
                  delay_ms(200);
                  dest=0;
                max=0;
                  min=9999999;
                  marca1=0;

//empieza a desplazar el disco rápidamente acercándolo a la posición d=0
                  for(ii=0;ii<440;ii++){
                        total=total-30;
                        set_pwm_duty(4, total);
                  }

//continua el desplazamiento del disco pero un poco más lento

                  for(ii=0;ii<1500;ii++){
                        total=total-5;
                        set_pwm_duty(4, total);
                        delay_ms(1);
                  }
```

```
                delay_ms(100);

// *** en este ciclo FOR se realizan los cálculos de las diferentes
      ***
// *** constantes para la conversión de onda seno a desplazamiento
      ***


                for(ii=0;ii<200;ii++){
                      total=total-10;
                      set_pwm_duty(4, total);
                      read_adc(ADC_START_ONLY);
                      lectura=read_adc(ADC_READ_ONLY);
                      delay_ms(10);
                      output_toggle(LEDV);
                }

                for(ii=0;ii<1100;ii++){ //1000
                      total=total-10;
                      set_pwm_duty(4, total);
                      read_adc(ADC_START_ONLY);
                      lectura=read_adc(ADC_READ_ONLY);
                      delay_ms(10);
                      if (lectura>max){
                            max=lectura;
                      }
                      if (lectura<min){
                            min=lectura;
                      }
                      output_toggle(LEDV);
                }
                res=min+((max-min)/2);
                divi=(max-min)/2;
                divi=divi+0.1;

                maxa=max-res;
                maxa=maxa+2;

                des1=0;
                dest=0;
                power=0;
                error_sum=0;
                cuentacontrol=0;
                cont1=0;
                temp_float=0;
                Ti=0;
                Tp=0;
                desired_power=0;
                marca1=0;

                while (true)
                {
                      if (kbhit()) {
```

```
                              duty = getc();
                        }
                        if (duty=='p'){
//cada vez que termine la adquisición de datos, JAVA manda
//una P para indicar salir de esta parte del programa
                              output_low(LEDR);
                              break;
                        }
                        if (duty=='r'){
//cada vez que se da en salir de adquisición para pasar los 50 datos
//una r para indicar salir de esta parte del programa
                              for(ii=1;ii<50;ii++){
                                          delay_ms(10);
                                          total=total+(20);
                                          cont1=cont1+2;
                                          output_toggle(LEDA);
                                    }
                              delay_ms(300);
                              output_low(LEDR);
                              break;
                        }
                        if (duty=='4'){
//cada dato que se quiera adquirir con control el programa en
//JAVA manda un 4 una vez este en el modo adquisición

                              if (cont1<50){
                                    for(ii=1;ii<50;ii++){
                                          delay_ms(10);
                                          total=total+(20);
                                          cont1=cont1+2;
                                          output_toggle(LEDA);
                                    }
                              }
                              read_adc(ADC_START_ONLY);
                              lectura=read_adc(ADC_READ_ONLY);
                              delay_ms(3);
                              funa=lectura;
                              funa=lectura-res;
                              rad=funa/maxa;
                              des2=(asin(rad));
                              des2=((((des2+1)*0.81)*1000)/(40*0.532));

//este es el valor del desplazamiento en comparación al punto anterior,
//el 0.532 es la longitud de onda del laser dividida entre mil, ya que
//antes esta multiplicado por mil, el 40 es por que el total se divide
//entre 4 por adecuacion de formulas para adecuar el algoritmo
                              if (des1==0){
                                    dest=0;
                              }
                              else{
                                    dest=dest+((abs(des1-des2))*sumres);
//sumres es para identificar si esta avanzando o retrocediendo el disco
//piezoeléctrico, el termino sale del controlador que puede ser +1 o -1
                              }
```

```
                              des1=des2;
//guarda el desplazamiento para compararlo con el siguiente punto

                              deste=lectura;



//mandar el valor del desplazamiento
                              lectura=dest+500f;
// se suma 500 al desplazamiento para evitar valores negativo al
//enviarlos desde el DSPIC, en programa en JAVA resta los 500 una
//vez recibido el dato
                  x1=make8(lectura,1);
                  x2=make8(lectura,0);

                              if (x2==10){
                                    x2=11;
                              }
                              if (x2==13){
                                    x2=12;
                              }
                              if (x1==10){
                                    x1=254;
                              }
                              if (x1==13){
                                    x1=255;
                              }
                              putc(x1);
                              putc(x2);

                              output_toggle(LEDV);

//mandar el valor del PWM
                              x1=make8(total,1);
                              x2=make8(total,0);

                              if (x2==10){
                                    x2=11;
                              }
                              if (x2==13){
                                    x2=12;
                              }
                              if (x1==10){
                                    x1=254;
                              }
                              if (x1==13){
                                    x1=255;
                              }
                              putc(x1);
                              putc(x2);


//mandar el valor de la lectura del sensor por el ADC
                              lectura=deste;
```

```
                        x1=make8(lectura,1);
                x2=make8(lectura,0);

                        if (x2==10){
                              x2=11;
                        }
                        if (x2==13){
                              x2=12;
                        }
                        if (x1==10){
                              x1=254;
                        }
                        if (x1==13){
                              x1=255;
                        }
                        putc(x1);
                        putc(x2);


//********************** Inicia el controlador **********************//

      datoa=dest;
      error = setpoint - datoa;                 //Calcula el error

      if(error>=0){//Si el desplazamiento esta por encima del SetPoint
            output_low(LEDA); //Prende el Led-Amarillo
            sumres=1;
      }
      else{             // de lo contrario
            output_high(LEDA);      //Apaga el Led-Amarillo
            sumres=-1;
      }

      // Cálculo del termino proporcional
      Tp = Kp * error;

      // Cálculo del termino integral
      error_sum = error_sum + (int32)error;
      merror_sum = (int32)32000f/Ki;

      if(error_sum>=merror_sum)
            error_sum=merror_sum;
      if(error_sum<=0)
            error_sum=0;

      temp_float = error_sum;
      Ti = Ki * temp_float;

      // Cálculo de la potencia deseada
      desired_power = (Tp + Ti);

// set correcto del PWM
      if (desired_power < 0){ //para que no pueda ser menor a 0
            total = 0;
```

```
    }
    else if (desired_power > 64000){ // Para que no pueda ser mayor a 64k
           total = 64000;
    }
      else {
           total = (desired_power*10f)+2570f;  //Adecua el total
      }
      set_pwm_duty(4, total);
      delay_ms(10);
//************************ Fin del controlador ************************//
                        }  // Fin del if (duty=='4')

                        if (duty=='5'){
                              //Realizar la adquisición pero sin control

                              if (mmax==1){
                                    lectura=max;
                                    x1=make8(lectura,1);
                        x2=make8(lectura,0);

                                    if (x2==10){
                                          x2=11;
                                    }
                                    if (x2==13){
                                          x2=12;
                                    }
                                    if (x1==10){
                                          x1=254;
                                    }
                                    if (x1==13){
                                          x1=255;
                                    }
                              putc(x1);
                              putc(x2);

                              lectura=min;
                                    x1=make8(lectura,1);
                        x2=make8(lectura,0);

                                    if (x2==10){
                                          x2=11;
                                    }
                                    if (x2==13){
                                          x2=12;
                                    }
                                    if (x1==10){
                                          x1=254;
                                    }
                                    if (x1==13){
                                          x1=255;
                                    }
                              putc(x1);
                              putc(x2);
                              }
```

```
                              mmax=50;


                              read_adc(ADC_START_ONLY);
                              lectura=read_adc(ADC_READ_ONLY);
                              delay_ms(10);
                              output_toggle(LEDR);
                              funa=lectura;
                              funa=lectura-res;
                              rad=funa/maxa;
                              if (cont1==50){
                                    dest=0;
                              }
                              cont1=cont1+1;
                              des2=(asin(rad));
                              des2=((((des2+1)*0.81)*1000)/(40*0.532));
                              if (des1==0){
                                    dest=0;
                              }
                              else{
                                    dest=dest+((abs(des1-des2))*sumres);
                              }
                              des1=des2;
                              deste=lectura;


//mandar el valor del desplazamiento
                              lectura=dest+500f;

                  x1=make8(lectura,1);
                  x2=make8(lectura,0);

                              if (x2==10){
                                    x2=11;
                              }
                              if (x2==13){
                                    x2=12;
                              }
                              if (x1==10){
                                    x1=254;
                              }
                              if (x1==13){
                                    x1=255;
                              }
                              putc(x1);
                              putc(x2);


//mandar el valor del PWM
                              x1=make8(total,1);
                  x2=make8(total,0);

                              if (x2==10){
                                    x2=11;
```

```
                                    }
                                    if (x2==13){
                                          x2=12;
                                    }
                                    if (x1==10){
                                          x1=254;
                                    }
                                    if (x1==13){
                                          x1=255;
                                    }
                                    putc(x1);
                                    putc(x2);


//mandar el valor de la lectura
                                    lectura=deste;
                                    x1=make8(lectura,1);
                        x2=make8(lectura,0);

                                    if (x2==10){
                                          x2=11;
                                    }
                                    if (x2==13){
                                          x2=12;
                                    }
                                    if (x1==10){
                                          x1=254;
                                    }
                                    if (x1==13){
                                          x1=255;
                                    }
                                    putc(x1);
                                    putc(x2);

                                    total=total+(20*sumres);
                                    if (marca1>50){
                                          marca1=0;
                                          output_toggle(LEDA);
                                    }
                                    marca1=marca1+1;
                                    set_pwm_duty(4, total);
                                    if (total>63156){//Si llega a este valor
                                    sumres=-1;  //el disco empieza a retroceder
                                    }
                                    if (total<2570){  //Si llega a este valor
                                    sumres=1;   //el disco empieza a avanzar
                                    }
                              }  // Fin del if (duty=='5')

                              else {
                                    duty=duty;
                              }
                        }           //Fin del While
                  break;            //Fin del Case 1
```

```
            case '3': // Recibe datos de las constantes para el
controlador
                        delay_ms(20);
                        output_high(LEDR);
                        if (kbhit()) {
                              recibido[0] = getc();
                              delay_ms(5);
                              recibido[1] = getc();
                              delay_ms(5);
                              recibido[2] = getc();
                              delay_ms(5);
                              recibido[3] = getc();
                              delay_ms(5);
                        }
                  kp=(((recibido[0]*256)+recibido[1])/10000f);

                  ki=(((recibido[2]*256)+recibido[3])/10000f);
                  output_low(LEDR);
            break;

            case '4':   // Envía lo que lee instantáneamente con el sensor
                  output_high(LEDR);
                  read_adc(ADC_START_ONLY);
                  lectura=read_adc(ADC_READ_ONLY);
                  delay_ms(5);

                  x=lectura;
            x1=make8(lectura,1);
            x2=make8(lectura,0);

                  if (x2==10){
                        x2=11;
                  }
                  if (x2==13){
                        x2=12;
                  }
                  if (x1==10){
                        x1=20;
                  }
                  if (x1==13){
                        x1=26;
                  }
                  putc(x1);
                  putc(x2);
                  output_low(LEDR);
            break;

            case '6': // Recibe datos del SetPoint
                  delay_ms(20);
                  output_high(LEDR);
                  if (kbhit()) {
                              recibido[0] = getc();
                              delay_ms(5);
```

```
                        recibido[1] = getc();
                        delay_ms(5);
                  }
                  if (recibido[0]==253)
                        recibido[0]=10;
                  if (recibido[0]==254)
                        recibido[0]=13;
                  sp=(((recibido[0]*256)+recibido[1]));

                  setpoint=sp;
                  output_low(LEDR);
            break;

            case '7': // Activar mandar dato en interrupción
                  delay_ms(20);
                  mandardato=1;
            break;

            case '8': // desactivar mandar dato en interrupción
                  delay_ms(20);
                  mandardato=0;
            break;

            case '9': // activa para que mande solo un dato
                  delay_ms(20);
                  mandardato=2;
            break;

            default:
                  for(ii=0;ii<10;ii++)
                  {
                        output_toggle(LEDR);
                        delay_ms(100);
                  }
            }
enable_interrupts(int_rda);                  // Habilita interrupciones
}                 //Final de interrupción por recibir datos por serial
// ********************************************************************


// Principal ***********************************************************
void main(void){

      enable_interrupts(int_rda);
      enable_interrupts(INTR_GLOBAL);          // habilita interrupciones

      for(ii=1;ii<10;ii++){ //Parpadean 5 veces todos los LED
            output_toggle(LEDR);
            output_toggle(LEDA);
            output_toggle(LEDV);
            delay_ms(30);
            }
            output_low(LEDR);
            output_low(LEDA);
```

```
            output_low(LEDV);

// Configuración de PWM ************************************************
      setup_timer3(TMR_INTERNAL,65535);
      setup_compare(4, COMPARE_PWM | COMPARE_TIMER3);
//*********************************************************************

// Configuración del timer para la interrupción ************************
      setup_timer1(TMR_INTERNAL | TMR_DIV_BY_256); //Configura el Timer1
      set_timer1(1);                        // Timer para el control PI
      setup_timer1(TMR_DISABLED);
      set_timer1(64754);
//*********************************************************************

// Configuración de ADC ************************************************
      setup_adc(ADC_CLOCK_DIV_16);
      setup_adc_ports(sAN3);
      SET_ADC_CHANNEL(3);
//*********************************************************************

      duty=0;
      lectura=0;
      total=0;
      set_pwm_duty(4, 0);

// Ejecuta lo mismo que interrupción para adquisición, para configurar
// constantes, para cálculos de conversión de onda seno a desplazamiento

      output_high(LEDR);
      total=35270;
      set_pwm_duty(4, total);
      delay_ms(200);
      dest=0;
      max=0;
      min=9999999;
      marca1=0;
      output_high(LEDR);
      for(ii=0;ii<440;ii++){
            total=total-30;
            set_pwm_duty(4, total);
      }

      for(ii=0;ii<1500;ii++){
            total=total-5;
            set_pwm_duty(4, total);
            delay_ms(1);
      }

      delay_ms(100);

      for(ii=0;ii<1500;ii++){
            total=total-8;
            set_pwm_duty(4, total);
            read_adc(ADC_START_ONLY);
```

```
            lectura=read_adc(ADC_READ_ONLY);
            delay_ms(1);
            if (lectura>max){
                  max=lectura;
            }
            if (lectura<min){
                  min=lectura;
            }
            output_toggle(LEDV);
      }
      res=min+((max-min)/2);
      divi=(max-min)/2;
      divi=divi+0.1;

      maxa=max-res;
      maxa=maxa+2;

      des1=0;
      dest=0;
      power=0;
      error_sum=0;
      cuentacontrol=0;
      temp_float=0;
      Ti=0;
      Tp=0;
      desired_power=0;
      output_low(LEDR);
      setup_timer1(TMR_INTERNAL | TMR_DIV_BY_256);
      enable_interrupts(INT_TIMER1);     // Habilita interrupción por
Timer1
      total=0;//borrar
      while (true){
            delay_ms(1);

      }
} //End del main*********************************************************
```

ANEXO B. Código Fuente Del Microcontrolador PIC18F2550

```
////////////////////////////////////////////////////////////////////////
//     **********     Programa CDS USB PIC 18F2550      **********     //
//  Programa para PIC 18F2550, compatible con Windows 7 de 64 bits,    //
//  para el proyecto de investigación denominado:                      //
//                                                                     //
//  "CARACTERIZACIÓN DEL DESPLAZAMIENTO DE UN DISCO PIEZOELÉCTRICO     //
//  EN FUNCIÓN DEL VOLTAJE CON CORRECCIÓN DE HISTÉRESIS Y ARRASTRE     //
//  MEDIANTE UN CONTROL EN LAZO CERRADO, UTILIZANDO INTERFEROMETRÍA."   //
//                                                                     //
//  ESTE CÓDIGO ESTA BASADO EN CÓDIGOS LIBRES ENCONTRADOS EN INTERNET  //
//                                                                     //
//                 Universidad de Nariño 2012 - 2013                   //
//                    San Juan de Pasto - Colombia                     //
////////////////////////////////////////////////////////////////////////

#include <18F2550.h>

#fuses HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL5,CPUDIV1,VREGEN
#use delay(clock=48000000)
#use rs232(baud=115200, UART1, errors)
   #define uart_putc(c) putc(c)
   #define uart_getc  getc
   #define uart_kbhit kbhit
   #define uart_printf  printf
#ifndef HW_INIT
#define HW_INIT()
#endif


//leds ordered from bottom to top
#DEFINE LED1 PIN_A5  //green
#define LED2 PIN_B4  //yellow
#define LED3 PIN_B5  //red
#define BUTTON_PRESSED() !input(PIN_A4)

//see section below labeled USB_CABLE_IS_ATTACHED
#define PIN_USB_SENSE   PIN_B2

#define HW_ADC_CONFIG   ADC_CLOCK_INTERNAL
#define HW_ADC_CHANNEL  0
#define HW_ADC_PORTS    AN0

#include "usb_cdc.h"

void usb_debug_task(void)
{
   static int8 last_connected;
   static int8 last_enumerated;
   int8 new_connected;
```

```
   int8 new_enumerated;
   static int8 last_cdc;
   int8 new_cdc;

   new_connected=usb_attached();
   new_enumerated=usb_enumerated();
   new_cdc=usb_cdc_connected();

   if (new_enumerated)
      output_high(LED1);
   else
      output_low(LED1);

   if (new_cdc)
      output_high(LED2);
   else
      output_low(LED2);

   if (usb_cdc_carrier.dte_present)
      output_high(LED3);
   else
      output_low(LED3);

   if (new_connected && !last_connected)
      uart_printf("USB connected, waiting for enumaration...\r\n\n");
   if (!new_connected && last_connected)
      uart_printf("USB disconnected, waiting for connection...\r\n\n");
   if (new_enumerated && !last_enumerated)
      uart_printf("USB enumerated by PC/HOST\r\n\n");
   if (!new_enumerated && last_enumerated)
      uart_printf("USB unenumerated by PC/HOST, waiting for
enumeration...\r\n\n");
   if (new_cdc && !last_cdc)
      uart_printf("Serial program initiated on USB<->UART COM
Port\r\n\n");

   last_connected=new_connected;
   last_enumerated=new_enumerated;
   last_cdc=new_cdc;
}

void main(void)
{
   char c;

   HW_INIT();
   set_tris_c(0x00);

   output_low(LED1);
   output_low(LED2);
   output_low(LED3);

   uart_printf("\r\n\nCCS CDC (Virtual RS232) Example\r\n");
```

```
  #if defined(__PCH__)
   uart_printf("\r\nPCH: v");
   uart_printf(__PCH__);
  #elif defined(__PCD__)
   uart_printf("\r\nPCD: v");
   uart_printf(__PCD__);
  #else
   uart_printf("\r\n\PCM: v");
   uart_printf(__PCM__);
  #endif
   uart_printf("\r\n");

   usb_init_cs();

  #if !(__USB_PIC_PERIF__)
   uart_printf("USBN: 0x%X", usbn_get_version());
   uart_printf("\r\n\n");
  #endif

while (TRUE)
   {
      usb_task();
      usb_debug_task();

      if (uart_kbhit())
      {
      c=uart_getc();


         if (c=='\n') {usb_cdc_putc('\r'); usb_cdc_putc('\n');}
         else if (c=='\r') {usb_cdc_putc('\r'); usb_cdc_putc('\n');}
         else if (c=='!') {uart_printf(usb_cdc_putc,"!");}
         else {usb_cdc_putc(c);}
      }


      if (usb_cdc_kbhit())
      {
         c=usb_cdc_getc();
         if (c=='\n') {uart_putc('\r'); uart_putc('\n');}
         else if (c=='\r') {uart_putc('\r'); uart_putc('\n');}
         else {uart_putc(c);}
      }
   }
}
```

ANEXO C. Código del programa en JAVA para la interfaz de usuario

```
package main;


import java.io.*;
import java.util.*;
import gnu.io.*;
import java.awt.Color;
import java.util.Date;
import java.awt.Image;
import java.awt.Toolkit;
import main.NewJFrame;
import java.awt.event.KeyEvent;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.jfree.chart.*;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.general.DefaultPieDataset;
import org.jfree.data.*;
import org.jfree.data.category.DefaultCategoryDataset;


import java.awt.Color;
import java.io.File;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartUtilities;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.axis.*;
import org.jfree.chart.axis.NumberTickUnit;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.chart.plot.XYPlot;
import org.jfree.chart.renderer.xy.XYLineAndShapeRenderer;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;
import org.jfree.chart.plot.CategoryPlot;


import org.jfree.chart.ChartFactory;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.labels.StandardCategoryItemLabelGenerator;
import org.jfree.chart.plot.CategoryPlot;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.chart.renderer.category.BarRenderer;
import org.jfree.data.category.DefaultCategoryDataset;
import org.jfree.ui.RectangleInsets;


import java.text.DecimalFormat;
import org.jfree.ui.RectangleInsets;

public class NewJFrame extends javax.swing.JFrame {
    static Enumeration       portList;
```

```
    static CommPortIdentifier portId;
    static String       messageString ="a";
    static SerialPort         serialPort;
    static OutputStream       outputStream;
    static boolean            outputBufferEmptyFlag = false;
    InputStream               inputStreamrec;
    int x,x1,x2,x3,x4,x5,x6,pri,sec,ii,total1,total2,dest1,dest2,prueba,
            nkpd,nkpd1,nkpd2,
            nkid,nkid1,nkid2,
            nsp,nsp1,nsp2;
    char c;
    float adq,tiempoadq,datosadq,total,dest,marca50,max,min;
    String
valor,numii,totalstr,valpri,valsec,valread,valread1,valread2,valread3,

valread4,valread5,valread6,info="",tiempostr,datosstr,adqstr,deststr,
           skpd,skpd1,skpd2,dskpd,
           skid,skid1,skid2,dskid,
           ssp,ssp1,ssp2,dssp,timestr,
           voltstr,maxstr,minstr;
    double dnkpd,dnkid,dnsp,volt,time,inicio;
    DecimalFormat voltaje = new DecimalFormat("##.####");
    DecimalFormat voltsrtd = new DecimalFormat("##.###");
    DecimalFormat adqsrtd = new DecimalFormat("#####");
    DecimalFormat destsrtd = new DecimalFormat("#####");




    public NewJFrame() {
        initComponents();
    }


@Override
    public Image getIconImage() {
        Image retValue = Toolkit.getDefaultToolkit().

getImage(ClassLoader.getSystemResource("icono/icono2.png"));


        return retValue;
    }


    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jButton2 = new javax.swing.JButton();
        jFormattedTextField1 = new javax.swing.JFormattedTextField();
        kpd = new javax.swing.JTextField();
        jLabel5 = new javax.swing.JLabel();
```

```
        jButton12 = new javax.swing.JButton();
        jLabel4 = new javax.swing.JLabel();
        jTextField1 = new javax.swing.JTextField();
        jButton6 = new javax.swing.JButton();
        jLabel3 = new javax.swing.JLabel();
        jScrollPane2 = new javax.swing.JScrollPane();
        jTextPane1 = new javax.swing.JTextPane();
        kid = new javax.swing.JTextField();
        jLabel7 = new javax.swing.JLabel();
        jLabel8 = new javax.swing.JLabel();
        jLabel10 = new javax.swing.JLabel();
        sp = new javax.swing.JTextField();
        jLabelkp = new javax.swing.JLabel();
        jLabelsp = new javax.swing.JLabel();
        jLabelki = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();
        jButton8 = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        datoscontrol = new javax.swing.JButton();
        jLabel2 = new javax.swing.JLabel();
        jButton9 = new javax.swing.JButton();
        jButton4 = new javax.swing.JButton();
        jButton3 = new javax.swing.JButton();
        jButton11 = new javax.swing.JButton();
        Abrir = new javax.swing.JButton();
        jButton10 = new javax.swing.JButton();
        Cerrar = new javax.swing.JButton();
        jButton5 = new javax.swing.JButton();
        jLabel6 = new javax.swing.JLabel();

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
            .addGap(0, 100, Short.MAX_VALUE)
        );
        jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
            .addGap(0, 100, Short.MAX_VALUE)
        );

        jButton2.setText("jButton2");

        jFormattedTextField1.setText("jFormattedTextField1");

        setTitle("Interfaz Interferómetro - Disco Piezoeléctrico (UDENAR
2013)");
        setIconImage(getIconImage());
        setPreferredSize(new java.awt.Dimension(800, 450));
```

```
        kpd.setText("10");
        kpd.addCaretListener(new javax.swing.event.CaretListener() {
            public void caretUpdate(javax.swing.event.CaretEvent evt) {
                kpdCaretUpdate(evt);
            }
        });
        kpd.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                kpdActionPerformed(evt);
            }
        });

        jLabel5.setText("Cantidad de muestras:  650");

        jButton12.setText("Recibir 1 Dato");
        jButton12.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton12ActionPerformed(evt);
            }
        });

        jLabel4.setText("Cantidad de muestras");

        jTextField1.setText("650");
        jTextField1.addCaretListener(new
javax.swing.event.CaretListener() {
            public void caretUpdate(javax.swing.event.CaretEvent evt) {
                jTextField1CaretUpdate(evt);
            }
        });
        jTextField1.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jTextField1ActionPerformed(evt);
            }
        });

        jButton6.setText("Borrar Texto");
        jButton6.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton6ActionPerformed(evt);
            }
        });

        jLabel3.setText("0");

        jTextPane1.setAutoscrolls(false);
        jTextPane1.setDragEnabled(true);
        jTextPane1.setVerifyInputWhenFocusTarget(false);
        jScrollPane2.setViewportView(jTextPane1);

        kid.setText("30");
        kid.addCaretListener(new javax.swing.event.CaretListener() {
```

```
    public void caretUpdate(javax.swing.event.CaretEvent evt) {
        kidCaretUpdate(evt);
    }
});
kid.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        kidActionPerformed(evt);
    }
});

jLabel7.setText("Kp = 0.");

jLabel8.setText("Ki = 0.");

jLabel10.setText("SP =");

sp.setText("1500");
sp.addCaretListener(new javax.swing.event.CaretListener() {
    public void caretUpdate(javax.swing.event.CaretEvent evt) {
        spCaretUpdate(evt);
    }
});
sp.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        spActionPerformed(evt);
    }
});

jLabelkp.setText("Kp = 0.0");
jLabelkp.setToolTipText("");

jLabelsp.setText("SP = 0.0 nm");
jLabelsp.setToolTipText("");

jLabelki.setText("ki = 0.0");
jLabelki.setToolTipText("");

jButton1.setText("Enviar SetPoint");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

jButton8.setText("Leer Dato");
jButton8.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton8ActionPerformed(evt);
    }
});

jLabel1.setText("Mensaje");

datoscontrol.setText("Enviar Datos Control");
```

```
        datoscontrol.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                datoscontrolActionPerformed(evt);
            }
        });

        jLabel2.setText("0");

        jButton9.setText("Salir Adquisición");
        jButton9.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton9ActionPerformed(evt);
            }
        });

        jButton4.setText("Iniciar Adquisición");
        jButton4.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton4ActionPerformed(evt);
            }
        });

        jButton3.setText("Adquisición");
        jButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton3ActionPerformed(evt);
            }
        });

        jButton11.setText("Desactivar Datos");
        jButton11.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton11ActionPerformed(evt);
            }
        });

        Abrir.setText("Abrir Puerto");
        Abrir.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                AbrirActionPerformed(evt);
            }
        });

        jButton10.setText("Activar Datos");
        jButton10.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton10ActionPerformed(evt);
            }
        });

        Cerrar.setText("Cerrar Puerto");
        Cerrar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
                CerrarActionPerformed(evt);
            }
        });

        jButton5.setText("Adquisición sin control");
        jButton5.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton5ActionPerformed(evt);
            }
        });

        jLabel6.setText(".");

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(10, 10, 10)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
                    .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
                            .addComponent(jButton9)
                            .addComponent(jButton12,
javax.swing.GroupLayout.PREFERRED_SIZE, 109,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
                            .addGroup(layout.createSequentialGroup()
                                .addComponent(Abrir,
javax.swing.GroupLayout.PREFERRED_SIZE, 99,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGap(10, 10, 10)
                                .addComponent(jButton11))
                            .addGroup(layout.createSequentialGroup()
                                .addComponent(Cerrar)
                                .addGap(10, 10, 10)
                                .addComponent(jButton10,
javax.swing.GroupLayout.PREFERRED_SIZE, 116,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                        .addGap(0, 393, Short.MAX_VALUE))
                    .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
```

```
                            .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 156,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jLabel2,
javax.swing.GroupLayout.PREFERRED_SIZE, 90,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jLabel3,
javax.swing.GroupLayout.PREFERRED_SIZE, 90,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
                                .addComponent(jButton8,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 187,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING, false)
                                    .addComponent(jButton5,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                    .addComponent(jButton3,
javax.swing.GroupLayout.PREFERRED_SIZE, 137,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addGroup(layout.createSequentialGroup()
                                        .addGap(10, 10, 10)
                                        .addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE, 127,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                    .addComponent(jButton4,
javax.swing.GroupLayout.PREFERRED_SIZE, 187,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                .addComponent(jLabel4))
                            .addComponent(jLabel5,
javax.swing.GroupLayout.PREFERRED_SIZE, 187,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TR
AILING, false)
                            .addComponent(jButton6,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(jButton1,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(datoscontrol,
javax.swing.GroupLayout.Alignment.LEADING,
```

```
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.addGroup(javax.swing.GroupLayout.Alignment.LEADING,
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TR
AILING, false)

.addGroup(javax.swing.GroupLayout.Alignment.LEADING,
layout.createSequentialGroup()
                                        .addGap(19, 19, 19)
                                        .addComponent(jLabel8)
                                        .addGap(4, 4, 4)
                                        .addComponent(kid,
javax.swing.GroupLayout.PREFERRED_SIZE, 51,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addGroup(javax.swing.GroupLayout.Alignment.LEADING,
layout.createSequentialGroup()
                                        .addGap(22, 22, 22)
                                        .addComponent(jLabel10)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                        .addComponent(sp,
javax.swing.GroupLayout.PREFERRED_SIZE, 51,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                    .addComponent(jLabelkp,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.PREFERRED_SIZE, 108,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addComponent(jLabelki,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.PREFERRED_SIZE, 108,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addComponent(jLabelsp,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.PREFERRED_SIZE, 108,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                .addGroup(layout.createSequentialGroup()
                                    .addGap(15, 15, 15)
                                    .addComponent(jLabel7)
                                    .addGap(4, 4, 4)
                                    .addComponent(kpd,
javax.swing.GroupLayout.PREFERRED_SIZE, 51,
javax.swing.GroupLayout.PREFERRED_SIZE))))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING, false)
                            .addComponent(jScrollPane2,
javax.swing.GroupLayout.DEFAULT_SIZE, 407, Short.MAX_VALUE)
```

```
                            .addComponent(jLabel6,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))))
                .addContainerGap())
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(15, 15, 15)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
                    .addComponent(kpd,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(3, 3, 3)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
                            .addGroup(layout.createSequentialGroup()
                                .addComponent(jLabel1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                .addComponent(jLabel2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                .addComponent(jLabel3)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                .addComponent(jButton3)
                                .addGap(1, 1, 1)
                                .addComponent(jLabel4)
                                .addGap(18, 18, 18)
                                .addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                .addComponent(jLabel5))
                            .addGroup(layout.createSequentialGroup()
                                .addComponent(jLabel7)
                                .addGap(9, 9, 9)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)

.addGroup(layout.createSequentialGroup()
                                        .addGap(3, 3, 3)
                                        .addComponent(jLabel8))
```

```
                                        .addComponent(kid,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                    .addGap(6, 6, 6)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BA
SELINE)
                                        .addComponent(sp,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addComponent(jLabel10))
                                    .addGap(6, 6, 6)
                                    .addComponent(jLabelkp)
                                    .addGap(6, 6, 6)
                                    .addComponent(jLabelki)
                                    .addGap(5, 5, 5)
                                    .addComponent(jLabelsp)))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
                                .addGroup(layout.createSequentialGroup()
                                    .addComponent(jButton1)
                                    .addGap(6, 6, 6)
                                    .addComponent(datoscontrol)
                                    .addGap(6, 6, 6)
                                    .addComponent(jButton6))
                                .addGroup(layout.createSequentialGroup()
                                    .addComponent(jButton4)
                                    .addGap(6, 6, 6)
                                    .addComponent(jButton5)
                                    .addGap(6, 6, 6)
                                    .addComponent(jButton8))))
                        .addGroup(layout.createSequentialGroup()
                            .addComponent(jLabel6)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                            .addComponent(jScrollPane2,
javax.swing.GroupLayout.PREFERRED_SIZE, 216,
javax.swing.GroupLayout.PREFERRED_SIZE)))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BA
SELINE)
                            .addComponent(Abrir)
                            .addComponent(jButton12))
                        .addComponent(jButton11))
```

```
                .addGap(11, 11, 11)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
                    .addComponent(jButton9)
                    .addComponent(Cerrar)
                    .addComponent(jButton10))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>

    private void kpdCaretUpdate(javax.swing.event.CaretEvent evt) {
        skpd = kpd.getText();
        nkpd = Integer.parseInt(skpd);
        dnkpd = nkpd;
        dnkpd = dnkpd / 10000;
        dskpd = Double.toString(dnkpd);

        jLabelkp.setText("kp = " + dskpd);
    }

    private void kpdActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jButton12ActionPerformed(java.awt.event.ActionEvent evt)
{
        // envia para que el DSPic mande un dato
        try {
            outputStream.write("9".getBytes());
        } catch (IOException e) {
        }


////////////////////////////////////////////////////////////////////////
////////
// recibir el valor del desplazamiento
        try {
            inputStreamrec = serialPort.getInputStream();
        } catch (IOException e) {
        }
        try {
            messageString = Integer.toString(inputStreamrec.read());
        } catch (IOException ex) {
            Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE,
null, ex);
        }
        dest1 = Integer.parseInt(messageString);

        try {
            inputStreamrec = serialPort.getInputStream();
```

```
        } catch (IOException e) {
        }
        try {
            messageString = Integer.toString(inputStreamrec.read());
        } catch (IOException ex) {
            Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE,
null, ex);
        }
        dest2 = Integer.parseInt(messageString);



        if (dest1 == 254) {
            dest1 = 10;
        }
        if (dest1 == 255) {
            dest1 = 13;
        }


        dest = ((dest1 * 256) + dest2) - 500;

////////////////////////////////////////////////////////////////////////
////////
// recibir el valor del PWM
        try {
            inputStreamrec = serialPort.getInputStream();
        } catch (IOException e) {
        }
        try {
            messageString = Integer.toString(inputStreamrec.read());
        } catch (IOException ex) {
            Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE,
null, ex);
        }
        total1 = Integer.parseInt(messageString);

        try {
            inputStreamrec = serialPort.getInputStream();
        } catch (IOException e) {
        }
        try {
            messageString = Integer.toString(inputStreamrec.read());
        } catch (IOException ex) {
            Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE,
null, ex);
        }
        total2 = Integer.parseInt(messageString);



        if (total1 == 254) {
            total1 = 10;
        }
```

```
        if (total1 == 255) {
            total1 = 13;
        }
        total = (total1 * 256) + total2;
        volt = (total / 1780) - 18.52;

        totalstr = Float.toString(total);
        deststr = Float.toString(dest);
        voltstr = Double.toString(volt);



        //ssp = sp.getText();
        info = jTextPane1.getText();
        //info = info + ssp + "," + voltstr + "," + deststr + ";" + "\n";
        info = info + ssp + "\t" + voltsrtd.format (volt) + "\t" +
destsrtd.format (dest) + ";" + "\n";
        jLabel6.setText("SetPoint           Voltaje
Desplazamiento");




        jTextPane1.setText(info);
    }

    private void jTextField1CaretUpdate(javax.swing.event.CaretEvent evt)
{
        datosstr = jTextField1.getText();
        datosadq = Float.parseFloat(datosstr) + 1;


        jLabel5.setText("Cantidad de muestras: " + datosstr);
    }

    private void jTextField1ActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
    }

    private void jButton6ActionPerformed(java.awt.event.ActionEvent evt)
{
        jTextPane1.setText("");
        info = "";
    }

    private void kidCaretUpdate(javax.swing.event.CaretEvent evt) {
        skid = kid.getText();
        nkid = Integer.parseInt(skid);
        dnkid = nkid;
        dnkid = dnkid / 10000;
        dskid = Double.toString(dnkid);

        jLabelki.setText("ki = " + dskid);
```

```
    }

    private void kidActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void spCaretUpdate(javax.swing.event.CaretEvent evt) {
//        ssp = sp.getText();
        //nsp = Integer.parseInt(ssp);
        dnsp = nsp;
        dnsp = dnsp;
        dssp = Double.toString(dnsp);

//        jLabelsp.setText("SP = " + dssp + " nm");
    }

    private void spActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{
        try {
            outputStream.write("6".getBytes());
        } catch (IOException e) {
        }

        ssp = sp.getText();
        nsp = Integer.parseInt(ssp);

        jLabelsp.setText("SP = " + ssp + " nm");
////////////////////////////////////////////////////////////////////////
////

// Conversiones para el Set Point
**********************************************
        nsp1 = nsp / 256;
        nsp2 = nsp - (nsp1 * 256);
        if (nsp1 == 10) {
            nsp1 = 253;
        }
        if (nsp1 == 13) {
            nsp1 = 254;
        }
        if (nsp2 == 10) {
            nsp2 = 11;
        }
        if (nsp2 == 13) {
            nsp2 = 12;
        }
        ssp1 = Integer.toString(nsp1);
        ssp2 = Integer.toString(nsp2);

        try {
```

```
            outputStream.write(nsp1);
        } catch (IOException e) {
        }
        try {
            Thread.sleep(10);
        } catch (InterruptedException ex) {
            Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE,
null, ex);
        }
        try {
            outputStream.write(nsp2);
        } catch (IOException e) {
        }
        try {
            Thread.sleep(10);
        } catch (InterruptedException ex) {
            Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE,
null, ex);
        }

        System.out.println("Set Point enviado por: " +
serialPort.getName());

    }

    private void jButton8ActionPerformed(java.awt.event.ActionEvent evt)
{
        for (ii = 1; ii < datosadq; ii++) {
//////////////////////////////////////////////////////////////////////
////////
// recibir el valor del desplazamiento
            try {
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            dest1 = Integer.parseInt(messageString);

            try {
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            dest2 = Integer.parseInt(messageString);
```

```
            if (dest1 == 254) {
                dest1 = 10;
            }
            if (dest1 == 255) {
                dest1 = 13;
            }



            dest = ((dest1 * 256) + dest2) - 500;

////////////////////////////////////////////////////////////////////////
////////
// recibir el valor del PWM
            try {
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            total1 = Integer.parseInt(messageString);

            try {
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            total2 = Integer.parseInt(messageString);



            if (total1 == 254) {
                total1 = 10;
            }
            if (total1 == 255) {
                total1 = 13;
            }
            total = (total1 * 256) + total2;
            volt = (total / 1780) - 18.52;

            totalstr = Float.toString(total);
            deststr = Float.toString(dest);
```

```
            voltstr = Double.toString(volt);

            //ssp = sp.getText();
            info = jTextPane1.getText();
            //info = info + ssp + "," + voltstr + "," + deststr + ";" +
"\n";
            info = info + ssp + "\t" + voltsrtd.format (volt) + "\t" +
destsrtd.format (dest) + ";" + "\n";
            jLabel6.setText("SetPoint           Voltaje
Desplazamiento");
        jTextPane1.setText(info);
        }
    }

    private void datoscontrolActionPerformed(java.awt.event.ActionEvent
evt) {

        try {
            outputStream.write("3".getBytes());
        } catch (IOException e) {
        }

        skpd = kpd.getText();
        nkpd = Integer.parseInt(skpd);

        skid = kid.getText();
        nkid = Integer.parseInt(skid);

        jLabelkp.setText("kp = " + dskpd);
        jLabelki.setText("ki = " + dskid);
////////////////////////////////////////////////////////////////////////
////

// Conversiones para kp
************************************************
        nkpd1 = nkpd / 256;
        nkpd2 = nkpd - (nkpd1 * 256);
        if (nkpd1 == 10) {
            nkpd1 = 11;
        }
        if (nkpd1 == 13) {
            nkpd1 = 12;
        }
        if (nkpd2 == 10) {
            nkpd2 = 11;
        }
        if (nkpd2 == 13) {
            nkpd2 = 12;
        }
        skpd1 = Integer.toString(nkpd1);
        skpd2 = Integer.toString(nkpd2);
```

```
// Conversiones para ki
************************************************
        nkid1 = nkid / 256;
        nkid2 = nkid - (nkid1 * 256);
        if (nkid1 == 10) {
            nkid1 = 11;
        }
        if (nkid1 == 13) {
            nkid1 = 12;
        }
        if (nkid2 == 10) {
            nkid2 = 11;
        }
        if (nkid2 == 13) {
            nkid2 = 12;
        }
        skid1 = Integer.toString(nkid1);
        skid2 = Integer.toString(nkid2);


        try {
            outputStream.write(nkpd1);
        } catch (IOException e) {
        }
        try {
            Thread.sleep(10);
        } catch (InterruptedException ex) {
            Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE,
null, ex);
        }
        try {
            outputStream.write(nkpd2);
        } catch (IOException e) {
        }
        try {
            Thread.sleep(10);
        } catch (InterruptedException ex) {
            Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE,
null, ex);
        }
        try {
            outputStream.write(nkid1);
        } catch (IOException e) {
        }
        try {
            Thread.sleep(10);
        } catch (InterruptedException ex) {
            Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE,
null, ex);
        }
        try {
            outputStream.write(nkid2);
        } catch (IOException e) {
        }
```

```
        try {
            Thread.sleep(10);
        } catch (InterruptedException ex) {
            Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE,
null, ex);
        }


        System.out.println("Datos enviados por: " +
serialPort.getName());

    }

    private void jButton9ActionPerformed(java.awt.event.ActionEvent evt)
{
        try {
            outputStream.write("r".getBytes());
        } catch (IOException e) {
        }
    }

    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt)
{

        DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        DefaultCategoryDataset dataset1 = new DefaultCategoryDataset();
        XYSeries serie1 = new XYSeries("Sensor");
        XYSeries serie2 = new XYSeries("Desplazamiento");
        x = 10000;

inicio = System.currentTimeMillis();
        for (ii = 1; ii < datosadq; ii++) {
            try {
                outputStream.write("4".getBytes());
            } catch (IOException e) {
            }
            try {
                outputStream.write("h".getBytes());
            } catch (IOException e) {
            }

            if (marca50<20) {
                try {Thread.sleep(500);
                } catch (Exception e) {}
            }
            marca50=100;




////////////////////////////////////////////////////////////////////////
////////
// recibir el valor del desplazamiento
            try {
```

```
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            dest1 = Integer.parseInt(messageString);

            try {
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            dest2 = Integer.parseInt(messageString);



            if (dest1 == 254) {
                dest1 = 10;
            }
            if (dest1 == 255) {
                dest1 = 13;
            }



            dest = ((dest1 * 256) + dest2) - 500;

////////////////////////////////////////////////////////////////////////
////////
// recibir el valor del PWM
            try {
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            total1 = Integer.parseInt(messageString);

            try {
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
```

```
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            total2 = Integer.parseInt(messageString);



            if (total1 == 254) {
                total1 = 10;
            }
            if (total1 == 255) {
                total1 = 13;
            }
            total = (total1 * 256) + total2;
            volt = (total / 1780) - 18.52;


////////////////////////////////////////////////////////////////////////
////////
// recibir el valor de la lectura del ADC
            try {
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            x1 = Integer.parseInt(messageString);

            try {
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            x2 = Integer.parseInt(messageString);



            if (x1 == 254) {
                x1 = 10;
            }
            if (x1 == 255) {
```

```
                x1 = 13;
            }
            adq = (x1 * 256) + x2;




       time = System.currentTimeMillis() - inicio;
       time=time/1000;

            adqstr = Float.toString(adq);
            totalstr = Float.toString(total);
            deststr = Float.toString(dest);
            voltstr = Double.toString(volt);
            numii = Integer.toString(ii);
            //info = info + numii + "         " + voltsrtd.format (volt)
+ "                    " + adqsrtd.format (adq) + "
" + destsrtd.format (dest) + ";" + "\n";
            info = info + time + "\t   " + voltsrtd.format (volt) +
"\t\t" + adqsrtd.format (adq) + "\t" + destsrtd.format (dest) + ";" +
"\n";
            //info = info + numii + "," + voltstr + "," + adqstr + "," +
deststr + ";" + "\n";
            timestr = Double.toString(time);
            Comparable desp = null;
            dataset.addValue(adq, "valor", timestr);
            dataset1.addValue(dest, "desp", timestr);
            serie1.add(time, adq);
            serie2.add(time, dest);
            jLabel6.setText("Tiempo(seg)        Voltaje(V)        Lectura
sensor        Desplazamiento(nm)");



        }

        double fin = System.currentTimeMillis() - inicio;
        fin=fin/1000;
        System.out.println("El tiempo transcurrido es " + fin + "
segundo");
        try {
            outputStream.write("p".getBytes());
        } catch (IOException e) {
        }


        jLabel1.setText(deststr);
        jTextPane1.setText(info);

///////////////
    XYSeriesCollection collection1 = new XYSeriesCollection();
```

```
    collection1.addSeries(serie1);

    JFreeChart chart1 = ChartFactory.createXYLineChart(
                "Adquisición del sensor", // Titulo
                "Tiempo (seg)", // Etiqueta eje X
                "Sensor", // Etiqueta eje Y
                collection1, // Datos
                PlotOrientation.VERTICAL,// orientacion
                false, // Incluye leyenda
                true, // Incluye tooltips
                false // urls
                );
        ChartFrame frame1 = new ChartFrame("Adquisición del sensor",
chart1);
        frame1.pack();
        frame1.setVisible(true);

        frame1.setBackground(Color.white);
        //NumberAxis rangeAxis = (NumberAxis) plot.getRangeAxis();

//rangeAxis.setStandardTickUnits(NumberAxis.createIntegerTickUnits());
        //rangeAxis.setTickUnit(new NumberTickUnit(1));
        //XYPlot plot1=chart1.getXYPlot();
        //NumberAxis domainAxis = (NumberAxis) plot1.getDomainAxis();
        //domainAxis.setTickUnit(new NumberTickUnit(1));


///////////////
    XYSeriesCollection collection2 = new XYSeriesCollection();
    collection2.addSeries(serie2);

    JFreeChart chart2 = ChartFactory.createXYLineChart(
                "Desplazamiento del disco", // Titulo
                "Tiempo (seg)", // Etiqueta eje X
                "Desplazamiento (nm)", // Etiqueta eje Y
                collection2, // Datos
                PlotOrientation.VERTICAL,// orientacion
                false, // Incluye leyenda
                true, // Incluye tooltips
                false // urls
                );
        ChartFrame frame2 = new ChartFrame("Desplazamiento del disco",
chart2);
        frame2.setBackground(Color.white);
        frame2.pack();
        frame2.setVisible(true);

        chart2.setBackgroundPaint(Color.white);



        try {
            String lineaArchivo;
```

```
            String fuenteArchivo = info;
            BufferedReader fuenteSalida;
            fuenteSalida = new BufferedReader(
                    new StringReader(fuenteArchivo));
// Se define un stream de salida (PrintWriter)
// que tomara los datos de memoria (BufferedWriter)
// y los escribira en un archivo (FileWriter)

            PrintWriter archivoSalida;
            archivoSalida = new PrintWriter(
                    new BufferedWriter(
                    new FileWriter("salida.txt")));

            while ((lineaArchivo = fuenteSalida.readLine()) != null) {
                archivoSalida.println(lineaArchivo);
            }
            archivoSalida.close(); // Se cierra el stream de salida
        } catch (IOException e) {
            System.out.println("Excepcion Entrada/Salida");
        }




    }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt)
{
        try {
            outputStream.write("1".getBytes());
        } catch (IOException e) {
        }
        try {
            outputStream.write("z".getBytes());
        } catch (IOException e) {
        }
        marca50=10;
    }

    private void jButton11ActionPerformed(java.awt.event.ActionEvent evt)
{
        // desactivar envio de datos en interrupcion
        try {
            outputStream.write("8".getBytes());
        } catch (IOException e) {
        }
    }

    private void AbrirActionPerformed(java.awt.event.ActionEvent evt) {
        boolean portFound = false;
        String defaultPort = "COM8";
        portList = CommPortIdentifier.getPortIdentifiers();
```

```
        while (portList.hasMoreElements()) {
            portId = (CommPortIdentifier) portList.nextElement();

            if (portId.getPortType() == CommPortIdentifier.PORT_SERIAL) {

                if (portId.getName().equals(defaultPort)) {
                    System.out.println("Found port " + defaultPort);

                    portFound = true;

                    try {
                        serialPort =
                                (SerialPort) portId.open("SimpleWrite",
2000);
                    } catch (PortInUseException e) {
                        System.out.println("Port in use.");

                        continue;
                    }

                    try {
                        outputStream = serialPort.getOutputStream();
                    } catch (IOException e) {
                    }

                    try {
                        serialPort.setSerialPortParams(115200,
                                SerialPort.DATABITS_8,
                                SerialPort.STOPBITS_1,
                                SerialPort.PARITY_NONE);

                    } catch (UnsupportedCommOperationException e) {
                    }


                    try {
                        serialPort.notifyOnOutputEmpty(true);
                    } catch (Exception e) {
                        System.out.println("Error setting event
notification");
                        System.out.println(e.toString());
                        System.exit(-1);
                    }

                    try {
                        Thread.sleep(2000);  // Be sure data is xferred
before closing
                    } catch (Exception e) {
                    }
                }
            }
        }
        jLabel1.setText("Puerto Abierto");
    }
```

```
    private void jButton10ActionPerformed(java.awt.event.ActionEvent evt)
{
        // activar envio de datos en interrupcion
        try {
            outputStream.write("7".getBytes());
        } catch (IOException e) {
        }
    }

    private void CerrarActionPerformed(java.awt.event.ActionEvent evt) {
        serialPort.close();
        jLabel1.setText("Puerto Cerrado");
    }

    private void jButton5ActionPerformed(java.awt.event.ActionEvent evt)
{

        DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        DefaultCategoryDataset dataset1 = new DefaultCategoryDataset();

        XYSeries serie1 = new XYSeries("Sensor");
        XYSeries serie2 = new XYSeries("Desplazamiento");
        x = 10000;


            try {
                outputStream.write("5".getBytes());
            } catch (IOException e) {
            }
            try {
                outputStream.write("h".getBytes());
            } catch (IOException e) {
            }


// recibir el valor de max y min
            try {
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            x1 = Integer.parseInt(messageString);

            try {
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
            }
            try {
```

```
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            x2 = Integer.parseInt(messageString);


            if (x1 == 254) {
                x1 = 10;
            }
            if (x1 == 255) {
                x1 = 13;
            }
            max = (x1 * 256) + x2;

            try {
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            x1 = Integer.parseInt(messageString);

            try {
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            x2 = Integer.parseInt(messageString);


            if (x1 == 254) {
                x1 = 10;
            }
            if (x1 == 255) {
                x1 = 13;
            }
            min = (x1 * 256) + x2;

            minstr = Float.toString(min);
            maxstr = Float.toString(max);
            jLabel2.setText(minstr);
            jLabel3.setText(maxstr);
```

```
////////////////////////////////////////////////////////////////////////
////////
// recibir el valor del desplazamiento
            try {
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            dest1 = Integer.parseInt(messageString);

            try {
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            dest2 = Integer.parseInt(messageString);



            if (dest1 == 254) {
                dest1 = 10;
            }
            if (dest1 == 255) {
                dest1 = 13;
            }



            dest = ((dest1 * 256) + dest2) - 500;

////////////////////////////////////////////////////////////////////////
////////
// recibir el valor del PWM
            try {
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            total1 = Integer.parseInt(messageString);
```

```
            try {
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            total2 = Integer.parseInt(messageString);



            if (total1 == 254) {
                total1 = 10;
            }
            if (total1 == 255) {
                total1 = 13;
            }
            total = (total1 * 256) + total2;
            volt = (total / 1780) - 18.52;



////////////////////////////////////////////////////////////////////////
////////
// recibir el valor de la lectura del ADC
            try {
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            x1 = Integer.parseInt(messageString);

            try {
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            x2 = Integer.parseInt(messageString);
```

```
            if (x1 == 254) {
                x1 = 10;
            }
            if (x1 == 255) {
                x1 = 13;
            }
            adq = (x1 * 256) + x2;



        for (ii = 1; ii < 50; ii++) {
            try {
                outputStream.write("5".getBytes());
            } catch (IOException e) {
            }
            try {
                outputStream.write("h".getBytes());
            } catch (IOException e) {
            }



////////////////////////////////////////////////////////////////////////
////////
// recibir el valor del desplazamiento
            try {
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            dest1 = Integer.parseInt(messageString);

            try {
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            dest2 = Integer.parseInt(messageString);
```

```
            if (dest1 == 254) {
                dest1 = 10;
            }
            if (dest1 == 255) {
                dest1 = 13;
            }



            dest = ((dest1 * 256) + dest2) - 500;

////////////////////////////////////////////////////////////////////////////
////////
// recibir el valor del PWM
            try {
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            total1 = Integer.parseInt(messageString);

            try {
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            total2 = Integer.parseInt(messageString);



            if (total1 == 254) {
                total1 = 10;
            }
            if (total1 == 255) {
                total1 = 13;
            }
            total = (total1 * 256) + total2;
            volt = (total / 1780) - 18.52;



////////////////////////////////////////////////////////////////////////////
////////
```

```
// recibir el valor de la lectura del ADC
            try {
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            x1 = Integer.parseInt(messageString);

            try {
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            x2 = Integer.parseInt(messageString);



            if (x1 == 254) {
                x1 = 10;
            }
            if (x1 == 255) {
                x1 = 13;
            }
            adq = (x1 * 256) + x2;
        }


 inicio = System.currentTimeMillis();
        for (ii = 1; ii < datosadq; ii++) {
            try {
                outputStream.write("5".getBytes());
            } catch (IOException e) {
            }
            try {
                outputStream.write("h".getBytes());
            } catch (IOException e) {
            }




////////////////////////////////////////////////////////////////////////
////////
// recibir el valor del desplazamiento
```

```
            try {
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            dest1 = Integer.parseInt(messageString);

            try {
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            dest2 = Integer.parseInt(messageString);



            if (dest1 == 254) {
                dest1 = 10;
            }
            if (dest1 == 255) {
                dest1 = 13;
            }



            dest = ((dest1 * 256) + dest2) - 500;

////////////////////////////////////////////////////////////////////////
////////
// recibir el valor del PWM
            try {
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            total1 = Integer.parseInt(messageString);

            try {
                inputStreamrec = serialPort.getInputStream();
```

```
            } catch (IOException e) {
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            total2 = Integer.parseInt(messageString);



            if (total1 == 254) {
                total1 = 10;
            }
            if (total1 == 255) {
                total1 = 13;
            }
            total = (total1 * 256) + total2;
            volt = (total / 1780) - 18.52;



////////////////////////////////////////////////////////////////////////
////////
// recibir el valor de la lectura del ADC
            try {
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            x1 = Integer.parseInt(messageString);

            try {
                inputStreamrec = serialPort.getInputStream();
            } catch (IOException e) {
            }
            try {
                messageString = Integer.toString(inputStreamrec.read());
            } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            x2 = Integer.parseInt(messageString);



            if (x1 == 254) {
                x1 = 10;
```

```
            }
            if (x1 == 255) {
                x1 = 13;
            }
            adq = (x1 * 256) + x2;




//System.out.println (voltstrd.format (3.43242383));

        time = System.currentTimeMillis() - inicio;
       time=time/1000;

            adqstr = Float.toString(adq);
            totalstr = Float.toString(total);
            deststr = Float.toString(dest);
            voltstr = Double.toString(volt);
            numii = Integer.toString(ii);
            //info = info + numii + "         " + voltsrtd.format (volt)
+ "                    " + adqsrtd.format (adq) + "
" + destsrtd.format (dest) + ";" + "\n";
            info = info + time + "\t   " + voltsrtd.format (volt) +
"\t\t" + adqsrtd.format (adq) + "\t" + destsrtd.format (dest) + ";" +
"\n";
            //info = info + numii + "," + voltstr + "," + adqstr + "," +
deststr + ";" + "\n";
            timestr = Double.toString(time);
            Comparable desp = null;
            dataset.addValue(adq, "valor", timestr);
            dataset1.addValue(dest, "desp", timestr);
            serie1.add(time, adq);
            serie2.add(time, dest);
            jLabel6.setText("Tiempo(seg)        Voltaje(V)        Lectura
sensor        Desplazamiento(nm)");




        }

        double fin = System.currentTimeMillis() - inicio;
        fin=fin/1000;
        System.out.println("El tiempo transcurrido es " + fin + "
segundo");
        try {
            outputStream.write("p".getBytes());
        } catch (IOException e) {
        }


        jLabel1.setText(deststr);
```

```
        jTextPane1.setText(info);

///////////////
    XYSeriesCollection collection1 = new XYSeriesCollection();
    collection1.addSeries(serie1);

    JFreeChart chart1 = ChartFactory.createXYLineChart(
                "Adquisición del sensor", // Titulo
                "Tiempo (seg)", // Etiqueta eje X
                "Sensor", // Etiqueta eje Y
                collection1, // Datos
                PlotOrientation.VERTICAL,// orientacion
                false, // Incluye leyenda
                true, // Incluye tooltips
                false // urls
                );
        ChartFrame frame1 = new ChartFrame("Adquisición del sensor",
chart1);
        frame1.pack();
        frame1.setVisible(true);

        frame1.setBackground(Color.white);
        //NumberAxis rangeAxis = (NumberAxis) plot.getRangeAxis();

//rangeAxis.setStandardTickUnits(NumberAxis.createIntegerTickUnits());
        //rangeAxis.setTickUnit(new NumberTickUnit(1));
        //XYPlot plot1=chart1.getXYPlot();
        //NumberAxis domainAxis = (NumberAxis) plot1.getDomainAxis();
        //domainAxis.setTickUnit(new NumberTickUnit(1));


///////////////
    XYSeriesCollection collection2 = new XYSeriesCollection();
    collection2.addSeries(serie2);

    JFreeChart chart2 = ChartFactory.createXYLineChart(
                "Desplazamiento del disco", // Titulo
                "Tiempo (seg)", // Etiqueta eje X
                "Desplazamiento (nm)", // Etiqueta eje Y
                collection2, // Datos
                PlotOrientation.VERTICAL,// orientacion
                false, // Incluye leyenda
                true, // Incluye tooltips
                false // urls
                );
        ChartFrame frame2 = new ChartFrame("Desplazamiento del Disco",
chart2);
        frame2.setBackground(Color.white);
        frame2.pack();
        frame2.setVisible(true);

        chart2.setBackgroundPaint(Color.white);

        try {
```

```
            String lineaArchivo;
            String fuenteArchivo = info;
            BufferedReader fuenteSalida;
            fuenteSalida = new BufferedReader(
            new StringReader(fuenteArchivo));
// Se define un stream de salida (PrintWriter)
// que tomara los datos de memoria (BufferedWriter)
// y los escribira en un archivo (FileWriter)

            PrintWriter archivoSalida;
            archivoSalida = new PrintWriter(
                    new BufferedWriter(
                    new FileWriter("salida.txt")));

            while ((lineaArchivo = fuenteSalida.readLine()) != null) {
                archivoSalida.println(lineaArchivo);
            }
            archivoSalida.close(); // Se cierra el stream de salida
        } catch (IOException e) {
            System.out.println("Excepcion Entrada/Salida");
        }




        /*
         * EN CASO DE QUERE EXPORTAR COMO TXT try{ String lineaArchivo;
String
         * fuenteArchivo=info; BufferedReader fuenteSalida; fuenteSalida
= new
         * BufferedReader( new StringReader(fuenteArchivo)); // Se define
un
         * stream de salida (PrintWriter) // que tomara los datos de
memoria
         * (BufferedWriter) // y los escribira en un archivo (FileWriter)
         *
         * PrintWriter archivoSalida; archivoSalida = new PrintWriter(
new
         * BufferedWriter( new FileWriter("salida.txt")));
         *
         * while ((lineaArchivo = fuenteSalida.readLine()) != null)
         * archivoSalida.println(lineaArchivo); archivoSalida.close(); //
Se
         * cierra el stream de salida } * catch (IOException e) {
         * System.out.println("Excepcion Entrada/Salida"); } *
         */



        /*
         * public grabarArchivo() { try { FileOutputStream salida = new
         * FileOutputStream("NOMBRE DEL ARCHIVO", true);
         *
```

```
         * salida.write(" STRING DESEADO ");
         *
         * salida.close(); } catch (Exception e) { JOptionPane jOP = new
         * JOptionPane(); jOP.showMessageDialog(null, "Error de
escritura,
         * archivo de errores","", 0); } }
         *
         *
         *
         *
         * DefaultCategoryDataset dataset = new DefaultCategoryDataset();
         * dataset.addValue(212, "Usuarios", "Agosto");
dataset.addValue(504,
         * "Usuarios", "Septiembre"); dataset.addValue(1520, "Usuarios",
         * "Octubre"); dataset.addValue(1842, "Usuarios", "Noviembre");
         * dataset.addValue(2991, "Usuarios", "Diciembre");
         *
         *
         * JFreeChart chart = ChartFactory.createLineChart( "Grafica
Lineal", //
         * Titulo "Mes", // Etiqueta de datos "Usuarios", // Etiqueta de
valores
         * dataset, // Datos PlotOrientation.VERTICAL,// orientacion
false, //
         * Incluye leyenda true, // Incluye tooltips false // urls ); *
         * ChartFrame frame = new ChartFrame("Graficador", chart);
frame.pack();
         * frame.setVisible(true);
         */




    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {



        System.out.println("Inicia el programa");
      boolean portFound = false;
        String  defaultPort = "COM8";

      if (args.length > 0) {
          defaultPort = args[0];
      }

      portList = CommPortIdentifier.getPortIdentifiers();

      while (portList.hasMoreElements()) {
```

```
            portId = (CommPortIdentifier) portList.nextElement();

            if (portId.getPortType() == CommPortIdentifier.PORT_SERIAL) {

              if (portId.getName().equals(defaultPort)) {
                  System.out.println("Found port " + defaultPort);

                  portFound = true;

                  try {
                    serialPort =
                        (SerialPort) portId.open("SimpleWrite", 2000);
                  } catch (PortInUseException e) {
                    System.out.println("Port in use.");

                    continue;
                  }

                  try {
                    outputStream = serialPort.getOutputStream();
                  } catch (IOException e) {}

                  try {
                    serialPort.setSerialPortParams(115200,
                                                   SerialPort.DATABITS_8,
                                                   SerialPort.STOPBITS_1,
                                                   SerialPort.PARITY_NONE);
                  } catch (UnsupportedCommOperationException e) {}

                  try {
                    serialPort.notifyOnOutputEmpty(true);
                  } catch (Exception e) {
                    System.out.println("Error setting event notification");
                    System.out.println(e.toString());
                    System.exit(-1);
                  }


//    System.out.println("Writing \""+messageString+"\" to
"+serialPort.getName());

//    try {outputStream.write(messageString.getBytes());}
//    catch (IOException e) {}

                  try {
                     Thread.sleep(500);  // Be sure data is xferred before
closing
                  } catch (Exception e) {}


              }
            }
      }
```

```
      if (!portFound) {
          System.out.println("port " + defaultPort + " not found.");
      }

        //<editor-fold defaultstate="collapsed" desc=" Look and feel
setting code (optional) ">
        /*
         * If Nimbus (introduced in Java SE 6) is not available, stay
with the
         * default look and feel. For details see
         *
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(NewJFrame.class.getName()).log(java.ut
il.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(NewJFrame.class.getName()).log(java.ut
il.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(NewJFrame.class.getName()).log(java.ut
il.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(NewJFrame.class.getName()).log(java.ut
il.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /*
         * Create and display the form
         */


        java.awt.EventQueue.invokeLater(new Runnable() {

            public void run() {
                new NewJFrame().setVisible(true);
            }

        });
        }
```

```
    // Variables declaration - do not modify
    private javax.swing.JButton Abrir;
    private javax.swing.JButton Cerrar;
    private javax.swing.JButton datoscontrol;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton10;
    private javax.swing.JButton jButton11;
    private javax.swing.JButton jButton12;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JButton jButton4;
    private javax.swing.JButton jButton5;
    private javax.swing.JButton jButton6;
    private javax.swing.JButton jButton8;
    private javax.swing.JButton jButton9;
    private javax.swing.JFormattedTextField jFormattedTextField1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel10;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabelki;
    private javax.swing.JLabel jLabelkp;
    private javax.swing.JLabel jLabelsp;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JTextPane jTextPane1;
    private javax.swing.JTextField kid;
    private javax.swing.JTextField kpd;
    private javax.swing.JTextField sp;
    // End of variables declaration
}
```

ANEXO D. Articulo para publicación

Con los resultados del proyecto se elabora el artículo para publicación internacional el cual se presenta a partir de la siguiente pagina para poder posicionarlo con los márgenes correspondientes dentro de la hoja, también se omite en estas hojas del anexo la numeración de las paginas, de tal manera que se observe el anexo tal como se presentaría.