Informe Técnico

Software para el sistema láser de medición HP5528A

Benjamín Valera Orozco
Gerardo Antonio Ruíz Botello
Sergio Padilla Olvera
Rigoberto Nava Sandoval

Julio 2021

Proyecto PAPIME
Aplicaciones del láser en la metrología dimensional. Prácticas de calibración y medición

Financiamiento PAPIME PE103720

Software para el sistema láser de medición HP5528A

Benjamín Valera Orozco, Gerardo Antonio Ruíz Botello, Sergio Padilla Olvera y Rigoberto Nava Sandoval

Instituto de Ciencias Aplicadas y Tecnología
Universidad Nacional Autónoma de México, Circuito Exterior S/N, Ciudad Universitaria AP 70-186, C.P. 04510, México D.F. México

Resumen

El sistema de medición láser HP5528A es un instrumento de gran precisión para la medición dimensional que está constituido por: una lámpara láser, una consola de control y un conjunto de accesorios ópticos. El instrumento se utiliza principalmente como estándar para la calibración de máquinas y herramientas de uno o más ejes de desplazamiento lineal o angular. Debido a su alta precisión, también puede ser utilizado para calibrar otros instrumentos de medición de alta exactitud como barras patrón y polígonos ópticos de referencia angular. Recientemente, con la incorporación del sistema láser de medición a los cursos regulares de licenciatura en ingeniería mecánica, se ha decido por incrementar la pulcritud con la que se manipula el equipo con el objetivo de reducir los posibles daños que el personal académico o alumnos pudieran ocasionar a este costoso equipo. En este informe se reporta el desarrollo de un software de interfaz entre el láser HP5528A y el usuario. El software fue pensado como una estrategia para aminorar las pulsaciones sobre los botones de su consola de control y que estas operaciones se lleven a cabo desde el programa de interfaz ejecutándose desde una computadora portátil. En este sentido la carga de trabajo en la consola de operación es reducida, aumentando, por lo tanto, la vida útil del láser HP5528A en su conjunto. En este concepto, y tomando en cuenta los costos de mantenimiento del láser, es más sencillo y económico exponer a una computadora portátil a la carga de trabajo que requiere la intervención humana como parte de la operación del equipo, sobre todo tomando en cuenta la gran cantidad de operadores que puede haber en un curso de ingeniería. En este informe se describe el software desarrollado en Visual Studio 2019 para Windows 2010 que no solo permite la sustitución de la consola de control por la interfaz de operación, sino que también facilita la adquisición de las mediciones al transportarlas dinámicamente hacia una hoja de cálculo de Excel 365. Se presenta como resultado final el software desarrollado ejecutándose en tiempo real en una computadora LapTop con conexión GPIB con el láser HP5528A.

Nomenclatura

Símbolo, término o abreviatura	Significado
COM	Component Object Model. Es una plataforma de programación que permite la comunicación entre procesos de aplicaciones.
Cliente	Recurso de computación que solicita servicios en un enlace entre procesos.
Servidor	Recurso de computación que pone recursos a disposición a través de un enlace entre procesos.
GPIB	General Purpose Instrumentation Bus. Estándar de comunicación entre equipos de cómputo e instrumentos de medición.
USB	Universal Serial Bus. Método de conexión para equipos de cómputo.
IEE-488.2	Estándar de comunicación sinónimo de GPIB.
ISA	Industry Standard Architecture. Conexión estándar para computadoras XT de 8 bits.
XT	Extended Technology. Nombre proporcionado a las primeras computadoras personales de la marca IBM.
EISA	Extended Industry Standard Architecture. Conexión estándar para computadoras de 32 bits.
OLE	Object Linking and Embedding. Es un protocolo de Microsoft que permite incrustar objetos de programación en otros programas.
MFC	Microsoft Foundation Classes. Es un conjunto de clases jerarquizadas que Microsoft pone a la disposición de los programadores para la elaboración de aplicaciones en ambiente Windows.
DLL	Dynamic Link Library. Son librerías de enlace dinámico que se ejecutan por demanda de otro programa en el entorno de algún sistema operativo, comúnmente Windows.
V.O.L.	Velocity Of Ligth. Velocidad de la luz.

Introducción

Definición del problema

En fechas recientes, en el Laboratorio de Ingeniería de precisión y Metrología del ICAT UNAM se decidió incluir al sistema láser de medición HP5528A como parte del instrumental requerido para realizar prácticas de laboratorio en los cursos regulares de la Facultad de Ingeniería UNAM. Debido a los altos costos tanto del sistema láser como de su mantenimiento, se ha optado por reducir el contacto humano con el equipo lo menos posible, con el fin de disminuir los costos de mantenimiento y evitar desgastes y accidentes que los operadores pudieran ocasionar. Con esta idea en mente se resolvió que una manera de prolongar la vida útil del láser HP5528A es disminuir el uso de su consola de control al remplazarla por un software de interfaz de tal manera que el contacto físico entre el operador y el equipo sea a través de una computadora personal. En este enfoque es más sencillo y económico remplazar una computadora personal que la consola de control del láser.

Entorno actual

Debido a la gran precisión y costo del sistema láser de medición HP5528A, en el Laboratorio de Ingeniería de Precisión y Metrología siempre ha sido una preocupación fundamental mantener este equipo lo menos expuesto a eventualidades que pudieran ocasionar el mal funcionamiento o desgaste del equipo.

En concordancia con esta preocupación, en los años de 1997 (Valera et al, 1997) y 2002 (Valera, Ruiz, 2002) se desarrollaron proyectos enfocados a disminuir la exposición del equipo a la intervención humana. En 1997 se desarrolló un programa para computadora con sistema operativo DOS que permitía operar el láser desde la computadora. La tecnología utilizada en esa fecha fue: Programación en Pascal, tarjeta GPIB con conexión ISA y computadora de escritorio XT.

Para el año de 2002 y con los avances tecnológicos en pleno crecimiento, el esquema desarrollado en 1997 fue renovado con los recursos más actuales para esa fecha, es decir: Programación Visual C++ V6.0, tarjeta GPIB para bus EISA, sistema operativo Windows XP y computadora de escritorio Pentium 486.

Para la fecha actual, 2021, las tecnologías mencionadas en los dos párrafos anteriores han ido desapareciendo y el surgimiento de nuevos sistemas ahora incluyen computadoras con decima generación de procesadores, medios de comunicación USB y software y sistemas operativos con alto respaldo en la nube de internet como Windows 10 y Visual Studio 2019. En este sentido, el avance tecnológico implica una renovación constante y sostenida de la infraestructura en el Laboratorio de Ingeniería de Precisión y Metrología.

Esta estrategia adoptada ha comprobado su efectividad, la prueba de ello es que se ha logrado mantener, sin descomposturas mayores, al sistema láser de medición HP5528A a lo largo de mas de 30 años de vida útil.

Descripción del problema a resolver y motivación

En resumen, el problema a resolver consiste en desarrollar software que pueda utilizarse en una computadora portátil como interfaz de operación, control y adquisición de mediciones para el sistema láser de medición HP 5528A. Aunque el láser HP5528A es un modelo antiguo que data de la década de 1990, éste sigue manteniendo vigencia y compatibilidad con el actual estándar de comunicaciones GPIB o IEE488.2 La tecnología a utilizar consiste en la más reciente infraestructura para la automatización de equipos de medición de gama alta:

- Interfaz de comunicaciones GPIB mediante puerto USB.
- Software de desarrollo Visual Studio 2019.
- Microsoft Excel 365.
- Sistema operativo Windows 10.
- Computadora portátil con procesador de décima generación.

Con esta infraestructura mencionada, se garantiza la vida útil del láser HP5528A por aproximadamente 5 años más.

El software de interfaz debe ser capaz de ejecutar la funcionalidad completa del láser HP5528A y, adicionalmente, permitir la adquisición de mediciones mediante el transporte de los datos hacia una hoja de cálculo en Excel 365.

Relevancia y limitaciones

La principal relevancia del proyecto radica en el continuo mantenimiento que se ha brindado al láser HP5528A desde el año de 1997 que nos ha permitido prolongar su vida útil más de lo esperado. Adicionalmente, este equipo puede ser incorporado a las prácticas de laboratorio en los cursos regulares de la Facultad de Ingeniería UNAM con la completa confianza de que su desgaste será mínimo.

La principal limitación en el desarrollo del presente trabajo es la restricción sanitaria que se ha impuesto a partir de la pandemia por COVID 19 y que ha impedido el ingreso del personal al Laboratorio de Ingeniería de Precisión y Metrología en donde se tienen las condiciones adecuadas y únicas para operar este instrumento de alta precisión.

Relación con otras áreas

El proyecto descrito en este informe está relacionado estrechamente con las áreas de metrología dimensional, automatización de procesos y desarrollo de software.

Método

El método por seguir en el presente proyecto de desarrollo se muestra esquemáticamente en la figura 1.

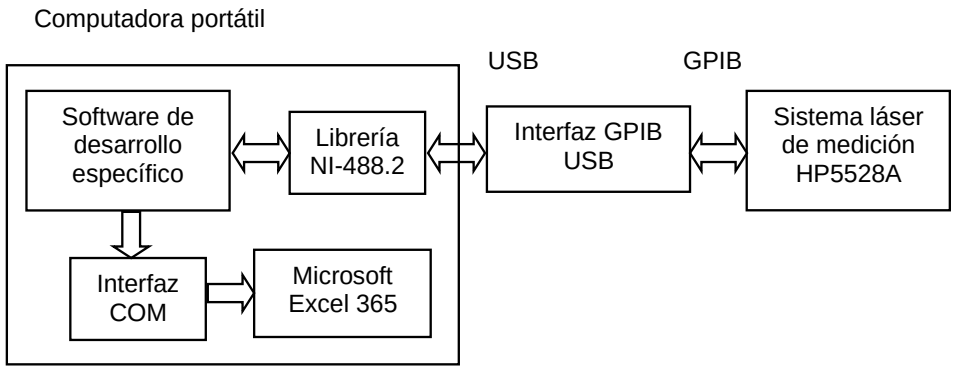


Figura 1. Método por emplear.

En el esquema de la figura 1 una computadora portátil con software específico constituye la interfaz de operación con el sistema láser de medición HP5528A. El software específico, desarrollado en lenguaje C para Visual Studio 2019, se encarga de ejecutar las instrucciones definidas en la librería NI-488.2 o GPIB que provee el fabricante de la Interfaz GPIB USB. La interfaz GPIB a USB es un dispositivo electrónico comercializado por la empresa National Instruments y cumple con los estándares de conexión entre equipos de medición con interfaz GPIB. Este dispositivo es conectado en un extremo al puerto USB de la computadora y en el otro extremo al conector GPIB del láser HP5528A. En este esquema es posible controlar la funcionalidad completa del láser HP5528A desde el software de desarrollo específico al enviar comandos de lectura y escritura propios del láser HP5528A. Por otra parte, el software de desarrollo específico despliega y envía las mediciones provenientes del láser HP5528A a una hoja de cálculo en Excel 365 mediante el concepto de programación COM que provee el sistema operativo Windows para la automatización de la paquetería Office 365 de Microsoft. En este esquema, el operador humano no tiene contacto físico con el láser HP5528A y su funcionalidad se lleva a cabo desde la computadora portátil con el software específico.

Objetivos y resultados esperados

El principal resultado de este trabajo es el software de desarrollo propio que permite controlar el láser HP5528A y transportar sus mediciones a una hoja de cálculo.

Organización del informe

La sección de introducción plantea el problema, el método empleado para resolverlo y expone los resultados esperados. El capítulo 1 del informe describe el desarrollo, la instalación y la operación del software específicamente elaborado para el presente proyecto. El capítulo 2 muestra el resultado del software específico ejecutándose en tiempo real y conectado al láser HP5528A. En este capítulo se incluye una breve guía para la detección de las fallas comúnmente encontradas. El informe es completado con sus respectivas secciones de referencias bibliográficas y conclusiones. Se incluyen

anexos conteniendo el código fuente desarrollado y las funciones comunes tanto de la librería NI-488.2 como del propio láser HP5528A.

1 Software para el láser HP5528A

El software para el láser HP5528A está desarrollado utilizando exclusivamente herramientas básicas de desarrollo. De esta manera se facilita su instalación en diversos equipos y se prescinde de software de terceras partes y el pago de costosas licencias. En primer lugar, se utiliza Visual Studio 2019 para el desarrollo de la interfaz de usuario. En segundo lugar, se utiliza la librería de programación NI-488.2 suministrada en conjunto con la tarjeta de interfaz GPIB a USB GPIB-USB-HS del fabricante National Instruments. En tercer lugar, se aprovechan las facilidades de colaboración entre procesos que ofrece el concepto COM OLE de Microsoft para el envío de mediciones a una hoja de cálculo en Excel 365.

1.1 Descripción general

La figura 2 muestra el esquema general en el desarrollo del software para el sistema de medición láser HP5528A.

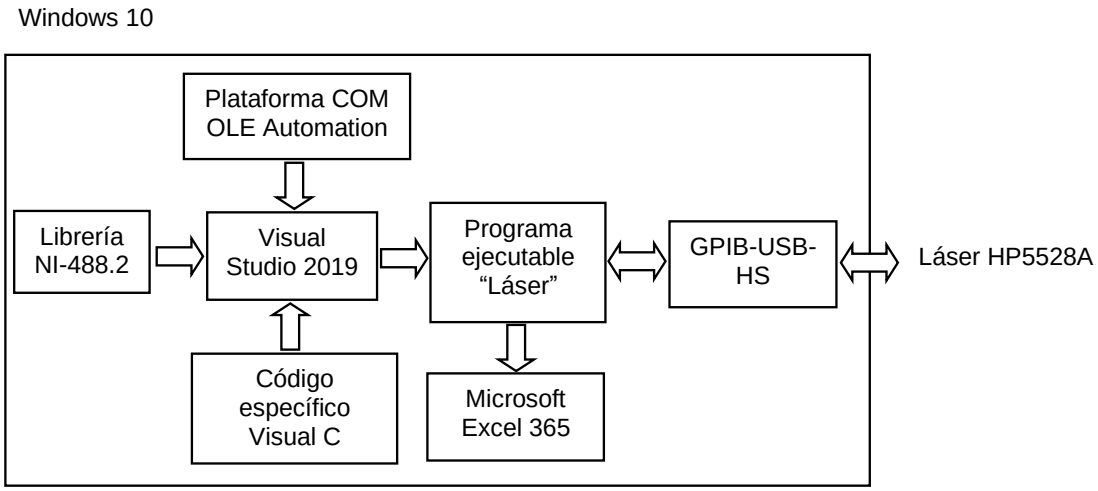


Figura 2. Esquema general para el desarrollo del software.

Con relación a la figura 2, Visual Studio 2019 es la herramienta básica para el desarrollo del programa ejecutable denominado “Láser”. En la compilación del archivo ejecutable Laser.exe se utilizan diversos recursos de programación que definen el comportamiento del programa. En primer lugar, la plataforma COM, que es un recurso de Microsoft para la comunicación entre procesos, habilita a la aplicación para la automatización de Excel mediante el envío de información desde el programa “Láser” hacia una hoja de cálculo en blanco. En segundo lugar, la librería NI-488.2 proporcionada por el fabricante de la tarjeta de interfaz GPIB-USB-HS, es el controlador que permite la programación de esta tarjeta desde un programa específico desarrollado en ambiente Windows. En tercer lugar, el código específico en lenguaje C de programación es el conjunto de instrucciones que conforman el algoritmo de medición y control desarrollado particularmente como parte del trabajo reportado en este informe. Adicionalmente, el código específico contiene el diseño de la pantalla o interfaz entre el usuario y el sistema láser HP5528A. El resultado final del proceso mostrado en la figura

2 es el programa ejecutable “Láser” que implementa la interfaz de usuario en tiempo real para el control y registro de mediciones del láser HP5528A y el transporte de estas mediciones hacia una hoja de cálculo en Excel 365.

1.2 Sistema de medición láser HP5528A

El sistema de medición láser HP5528A es un instrumento de alta precisión utilizado en mediciones dimensionales de distancia, velocidad, ángulo y rectitud (Hewlett Packard, 1986). Los tres componentes principales de este equipo son:

- Consola de control, figura 3.
- Lámpara láser, figura 4.
- Óptica auxiliar a la medición, figura 5.

En conjunto, el equipo es capaz de realizar mediciones de gran precisión en la verificación dimensional de instrumentos y patrones de longitud y ángulo y máquinas/herramientas de uno o varios ejes lineales o angulares.

Figura 3. Consola de control del láser HP5528A.

Figura 4. Lámpara láser del HP5528A.

Figura 5. Óptica auxiliar a la medición de longitud.

El láser HP5528A puede ser operado de dos formas:

- Mediante su consola de control de la figura 3 a través de la pulsación directa de los botones y la visualización en el despliegue alfanumérico.
- Mediante la conexión de un equipo externo controlador, como la interfaz GPIB-USB-H (National Instruments, 2021) de la figura 6, a su interfaz GPIB de la figura 7 “IEE STD GPIB PORT” y el uso de comandos de programación de lectura y escritura compatibles con el estándar GPIB.

Figura 6. Interfaz GPIB-USB-HS de National Instruments.

Adicionalmente, ver figura 7, el láser HP5528A cuenta con la capacidad de conectar tres sensores de temperatura “MATERIAL TEMPERATURE” y un sensor de condiciones del aire “AIR SENSOR”, variables ambientales que se pueden utilizar para compensar sus mediciones dimensionales.

Figura 7. Vista posterior de la consola de control en donde se encuentran localizados los puertos de conexión.

1.3 Descripción en el ámbito del programador

El programa “Láser” es una aplicación MFC (Kruglinski et al, 1988) en forma de caja de diálogo que implementa la interfaz de usuario y los algoritmos de medición y control para el láser HP5528A. El algoritmo en el programa “Láser” utiliza dos recursos de computación que deben incluirse como parte del proyecto “Laser” en visual Studio 2019. En primer lugar, se debe incluir la clase para la automatización de Excel a partir del registro de Windows (Programmer Sought, 2021). Esto se realiza mediante la inclusión de los archivos excel9.h y excel9.cpp como parte de la solución, como se muestra en la figura 8. Estos archivos contienen la definición de métodos y variables que se pueden utilizar en el código específico para ejecutar Excel y enviar información a las diferentes celdas en la hoja de cálculo. Los archivos excel9.h y excel.cpp requieren compatibilidad con el archivo de cabecera precompilado stdafx.h por lo que en el proyecto de Visual Studio 2019 no se pueden utilizar los nuevos controles mejorados de caja de diálogo definidos en el archivo precompilado pch.h. Por esta razón en el proyecto en Visual Studio 2019 se deben modificar todas las referencias a los métodos y variables definidos en pch.h y sustituirlos por referencias al antiguo archivo precompilado stdafx.h. Esta labor puede ser tediosa debido a que no solo implica la modificación al código fuente, sino que también se requieren modificaciones a las opciones de compilación. Una estrategia más simple sería crear el proyecto “Láser” en una versión antigua de Visual Studio, por ejemplo, Visual Studio 6.0, para luego transportarlo a Visual Studio 2019.

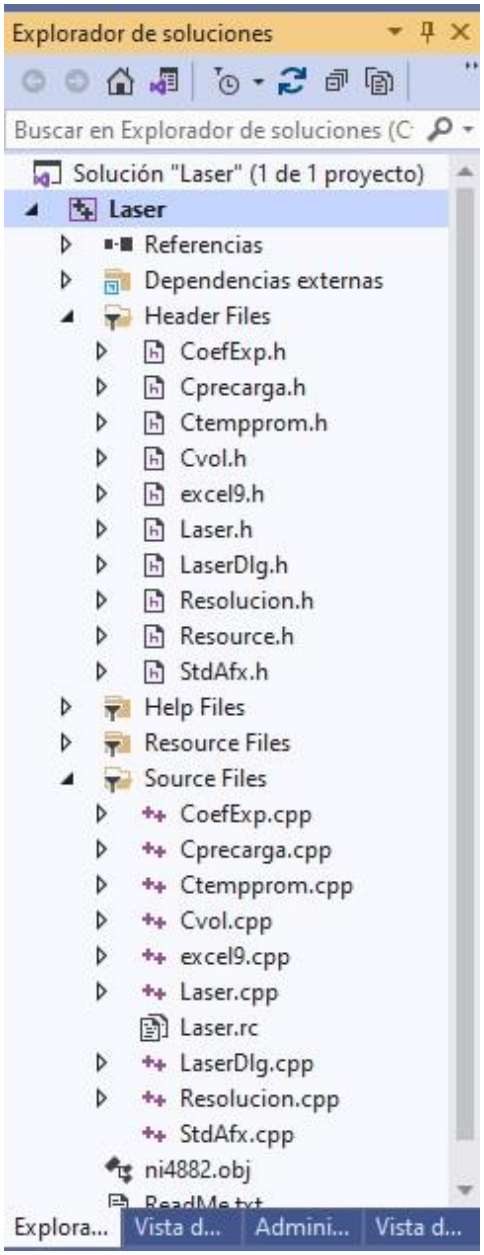
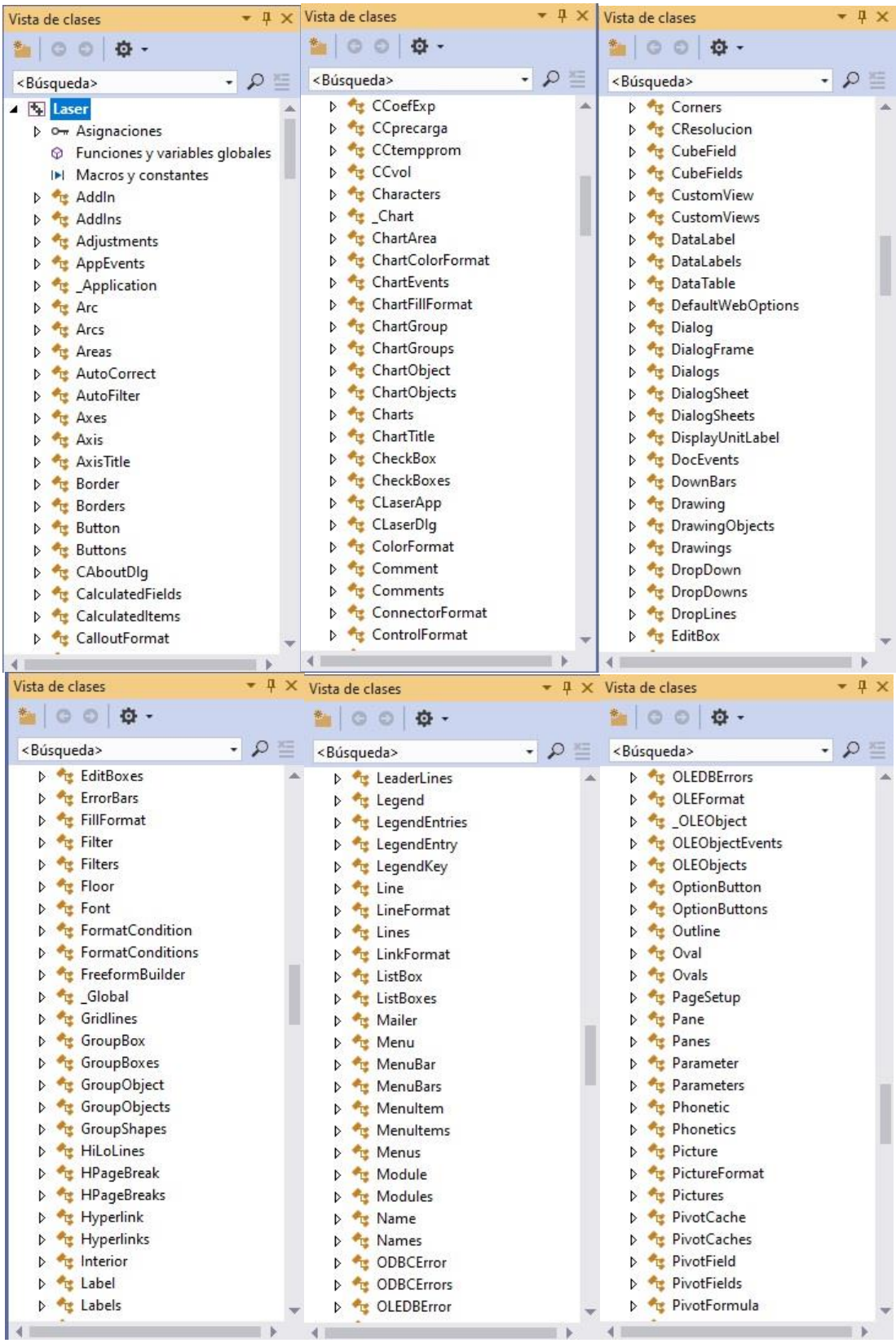


Figura 8. Explorador de soluciones de Visual Studio 2019 para el software "Láser".

Por otra parte y en segundo lugar, el proyecto "Láser" debe contener la librería NI-488.2 para que el código específico pueda acceder a la funcionalidad de la tarjeta GPIB-USB-HS y enviar comandos de escritura y lectura al láser HP5528A. La inclusión del componente ni4882.obj mostrado en la figura 8 se realiza al copiar los archivos ni4882.h y ni4882.obj, suministrados por el fabricante, al directorio del proyecto "Láser" y posteriormente ejecutar la opción "Proyecto/Agregar elemento existente…/ni4882.obj" del menú de opciones en Visual Studio 2019.

La figura 9 muestra la vista de clases para el proyecto "Laser" conteniendo 212 clases que serán descritas a continuación


Vista de clases
<Búsqueda>
Laser
Asignaciones
Funciones y variables globales
Macros y constantes
AddIn
AddIns
Adjustments
AppEvents
_Application
Arc
Arcs
Areas
AutoCorrect
AutoFilter
Axes
Axis
AxisTitle
Border
Borders
Button
Buttons
CAboutDlg
CalculatedFields
CalculatedItems
CalloutFormat
CCoefExp
CCprecarga
CCtempprom
CCvol
Characters
_Chart
ChartArea
ChartColorFormat
ChartEvents
ChartFillFormat
ChartGroup
ChartGroups
ChartObject
ChartObjects
Charts
ChartTitle
CheckBox
CheckBoxes
CLaserApp
CLaserDlg
ColorFormat
Comment
Comments
ConnectorFormat
ControlFormat
Corners
CResolucion
CubeField
CubeFields
CustomView
CustomViews
DataLabel
DataLabels
DataTable
DefaultWebOptions
Dialog
DialogFrame
Dialogs
DialogSheet
DialogSheets
DisplayUnitLabel
DocEvents
DownBars
Drawing
DrawingObjects
Drawings
DropDown
DropDowns
DropLines
EditBox
EditBoxes
ErrorBars
FillFormat
Filter
Filters
Floor
Font
FormatCondition
FormatConditions
FreeformBuilder
_Global
Gridlines
GroupBox
GroupBoxes
GroupObject
GroupObjects
GroupShapes
HiLoLines
HPageBreak
HPageBreaks
Hyperlink
Hyperlinks
Interior
Label
Labels
LeaderLines
Legend
LegendEntries
LegendEntry
LegendKey
Line
LineFormat
Lines
LinkFormat
ListBox
ListBoxes
Mailer
Menu
MenuBar
MenuBars
MenuItem
MenuItems
Menus
Module
Modules
Name
Names
ODBCError
ODBCErrors
OLEDBError
OLEDBErrors
OLEFormat
_OLEObject
OLEObjectEvents
OLEObjects
OptionButton
OptionButtons
Outline
Oval
Ovals
PageSetup
Pane
Panes
Parameter
Parameters
Phonetic
Phonetics
Picture
PictureFormat
Pictures
PivotCache
PivotCaches
PivotField
PivotFields
PivotFormula

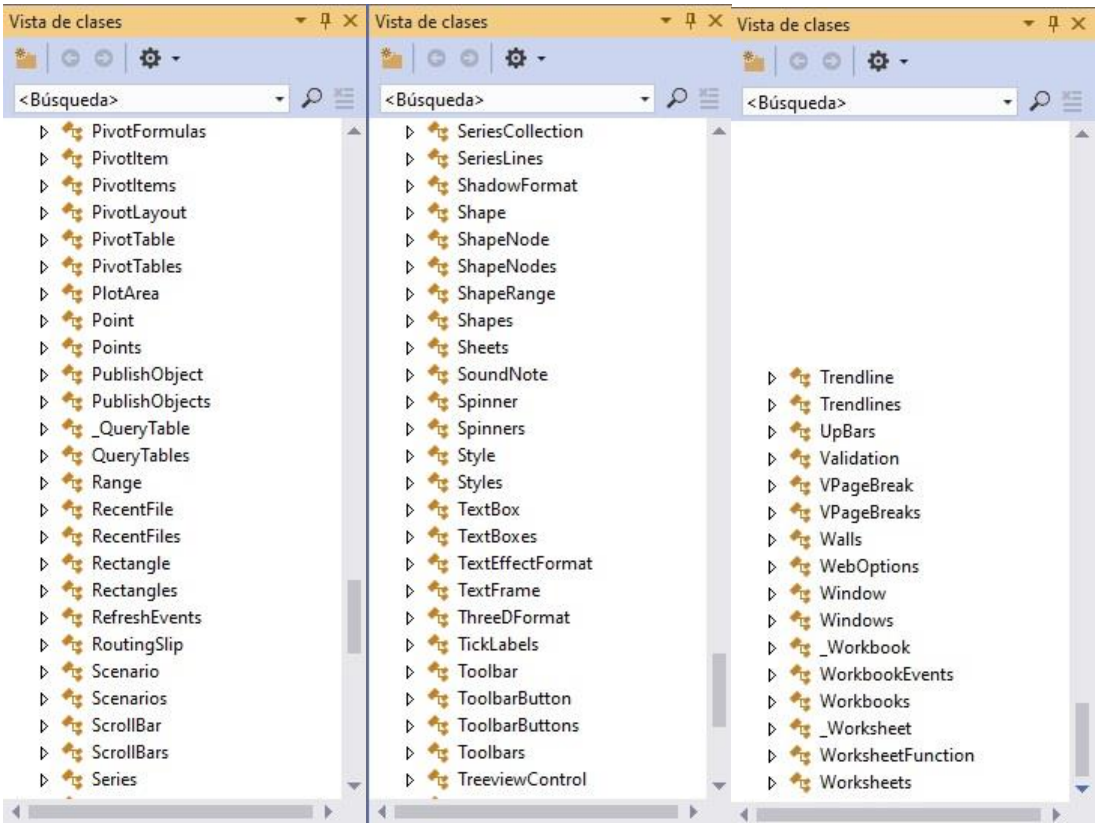


Figura 9. Vista de clases para el proyecto “Láser”.

1.3.1 Clases por omisión insertadas por el ambiente de programación

De las 212 clases de la figura 9, solamente 3 son insertadas por el asistente de creación del nuevo proyecto “Láser” en Visual Studio 2019 y, de estas 3, solamente dos son las que contienen el código específico que fue creado para proporcionarle funcionalidad a la aplicación.

CAboutDlg. Contiene los métodos CAboutDlg() y DoDataExchange() para el despliegue opcional de una caja de diálogo que muestre la información acerca de los derechos de los autores. No se modifica el código creado por omisión.

CLaserApp. Contiene los métodos CLaserApp() e InitInstance() que se utilizan para desplegar la caja de diálogo principal de la aplicación “Láser”. El método InitInstance() fue modificado para que cumpliera la importante función de habilitar el contenedor de la automatización OLE mediante las funciones AfxOleInit() y AfxEnableControlContainer() que inicializan el sistema de DLLs OLE COM y verifican que las DLLs corresponden con la versión correcta. El código completo de esta clase se puede consultar en el anexo A.

CLaserDlg. Contiene gran parte de la funcionalidad de la aplicación. Debido a su gran importancia, esta clase será descrita con mayor detalle en la sección 1.3.3.

1.3.2 Clases por omisión de la automatización

De las 212 clases de la figura 9, 204 son insertadas por el ambiente de programación al incluir los archivos excel9.h y excel9.cpp como parte del proyecto. En estas clases no se modifica en lo absoluto su código fuente y solamente son utilizadas parcialmente para cumplir con las funciones requeridas en el código específico de la aplicación. La tabla 1 muestra las clases mencionadas.

AddIn	ColorFormat	Floor	MenuBars	PivotFields	Shapes
AddIns	Comment	Font	MenuItem	PivotFormula	Sheets
Adjustments	Comments	FormatCondition	MenuItems	PivotFormulas	SoundNote
AppEvents	ConnectorFormat	FormatConditions	Menus	PivotItem	Spinner
_Application	ControlFormat	FreeformBuilder	Module	PivotItems	Spinners
Arc	Corners	_Global	Modules	PivotLayout	Style
Arcs	CubeField	Gridlines	Name	PivotTable	Styles
Areas	CubeFields	GroupBox	Names	PivotTables	TextBox
AutoCorrect	CustomView	GroupBoxes	ODBCError	PlotArea	TextBoxes
AutoFilter	CustomViews	GroupObject	ODBCErrors	Point	TextEffectFormat
Axes	DataLabel	GroupObjects	OLEDBError	Points	TextFrame
Axis	DataLabels	GroupShapes	OLEDBErrors	PublishObject	ThreeDFormat
AxisTitle	DataTable	HiLoLines	OLEFormat	PublishObjects	TickLabels
Border	DefaultWebOptions	HPageBreak	OLEObject	_QueryTable	Toolbar
Borders	Dialog	HPageBreaks	OLEObjectEvents	QueryTables	ToolbarButton
Button	DialogFrame	Hyperlink	_OLEObjects	Range	ToolbarButtons
Buttons	Dialogs	Hyperlinks	OptionButton	RecentFile	Toolbars
CalculatedFields	DialogSheet	Interior	OptionButtons	RecentFiles	TreeviewControl
CalculatedItems	DialogSheets	Label	Outline	Rectangle	Trendline
CalloutFormat	DisplayUnitLabel	Labels	Oval	Rectangles	Trendlines
Characters	DocEvents	LeaderLines	Ovals	RefreshEvents	UpBars
_Chart	DownBars	Legend	PageSetup	RoutingSlip	Validation
ChartArea	Drawing	LegendEntries	Pane	Scenario	VPageBreak
ChartColorFormat	DrawingObjects	LegendEntry	Panes	Scenarios	VPageBreaks
ChartEvents	Drawings	LegendKey	Parameter	ScrollBar	Walls
ChartFillFormat	DropDown	Line	Parameters	ScrollBars	WebOptions
ChartGroup	DropDowns	LineFormat	Phonetic	Series	Window
ChartGroups	DropLines	Lines	Phonetics	SeriesCollection	Windows
ChartObject	EditBox	LinkFormat	Picture	SeriesLines	_Workbook
ChartObjects	EditBoxes	ListBox	PictureFormat	ShadowFormat	WorkbookEvents
Charts	ErrorBars	ListBoxes	Pictures	Shape	Workbooks
ChartTitle	FillFormat	Mailer	PivotCache	ShapeNode	_Worksheet
CheckBox	Filter	Menu	PivotCaches	ShapeNodes	WorksheetFunction
CheckBoxes	Filters	MenuBar	PivotField	ShapeRange	Worksheets

Tabla1. Clases insertadas por omisión por la automatización.

1.3.3 Clases de la interfaz de operación

De las 212 clases totales del proyecto “Láser” de la figura 9, 5 son las clases específicamente desarrolladas para completar la funcionalidad del código fuente en el presente trabajo. A estas 5 clases habría que añadir la clase fuertemente modificada, CLaserDlg, que fue añadida por el ambiente de desarrollo Visual Studio 2019 en el momento de la creación del proyecto de programación “Láser”. Las 6 clases mencionadas se listan a continuación y se describen de manera detallada en lo que resta de la presente sección.

- CLaserDlg
- CCoefExp

- CCprecarga
- CCtempprom
- CCvol
- CResolucion

CLaserDlg

La figura 10 muestra los 29 métodos implementados como parte de la clase CLaserDlg.

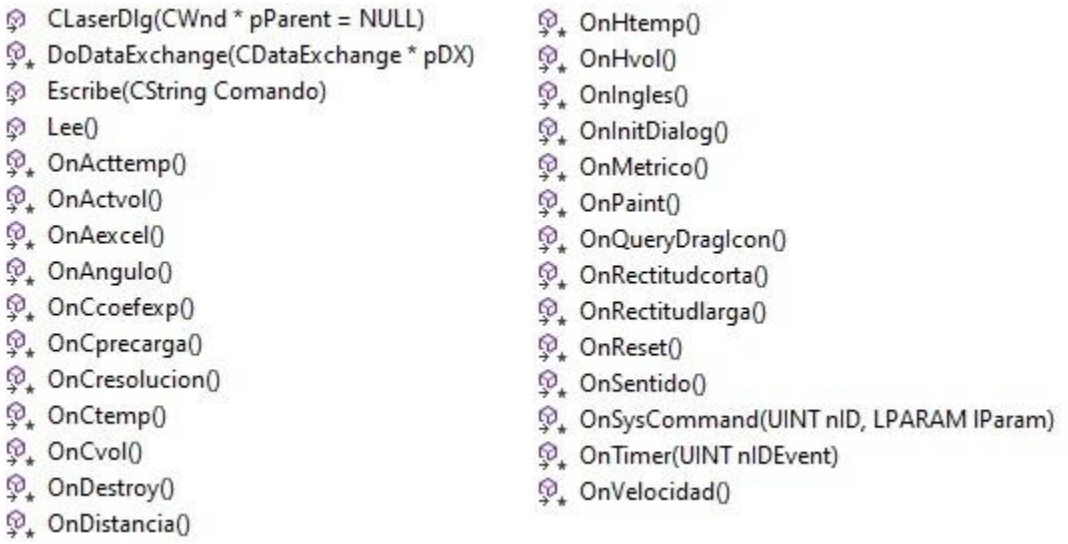


Figura 10. Métodos de la clase CLaserDlg.

A continuación, la descripción detallada de cada uno de los métodos de la figura 10. El código fuente completo de la clase se puede revisar en el anexo A.

CLaserDlg(CWnd* pParent = NULL); Es el constructor estándar de la aplicación en donde se inicializan variables utilizadas para el despliegue o para desempeñar el algoritmo de medición y control.

virtual void DoDataExchange(CDataExchange* pDX); Proporciona el soporte DDX/DDV para asignar variables y validación a los controles de la caja de diálogo de la aplicación principal.

BOOL Escribe(CString Comando); El método recibe la variable del tipo cadena, Comando, para enviarla al láser HP5528A mediante el comando GPIB “ibwrt” que se explica en el anexo B. Adicionalmente controla la aparición de errores mediante la variable GPIB “ibsta”.

CString Lee(); El método regresa un valor del tipo cadena resultado de efectuar una operación de lectura sobre el láser HP5528A mediante los comandos GPIB “ibtrg” e “ibrd” que se explican en el anexo B. Adicionalmente controla la aparición de errores en el proceso mediante la variable GPIB “ibsta”.

afx_msg void OnActtemp(); Es el método que responde al evento por presionar el botón “Actualiza temperaturas” de la caja de diálogo principal de la aplicación. Como la descripción lo indica, el método se encarga de enviar los comandos específicos GPIB del anexo C al láser HP5528A, mediante los métodos Lee() y Escribe(), para obtener los resultados de las mediciones de temperatura y desplegar los valores actualizados de los sensores en los controles de texto de la caja de diálogo.

afx_msg void OnActvol(); Es el método que responde al evento por presionar el botón “Actualiza datos” de la caja de diálogo principal de la aplicación. Como la descripción lo indica, el método se encarga de enviar los comandos específicos GPIB del anexo C al láser HP5528A, mediante los métodos Lee() y Escribe(), para obtener los resultados de las mediciones de presión, humedad, temperatura y V.O.L. para desplegar los valores actualizados en los controles de texto de la caja de diálogo.

afx_msg void OnAexcel(); Es el método que responde al evento por presionar el botón “Enviar lectura” de la caja de diálogo principal de la aplicación. Como la descripción lo indica, el método se encarga de enviar las lecturas del despliegue principal hacia la hoja de cálculo en Excel 365. La posición de la celda en la hoja de cálculo es controlada por las variables índices “m_iColumnas” y “m_iRenglones” que se incrementan con cada envío de acuerdo con las banderas m_bIncRenglón” y “m_bIncColumna” asociadas a los dos controles “Incrementa” de la misma caja de diálogo.

afx_msg void OnAngulo(); Es el método que responde al evento por presionar el radio botón “Angulo” de la caja de diálogo principal de la aplicación. Como la descripción lo indica, el método se encarga de establecer en el láser HP5528A el modo de medición en “Angulo”, mediante los comandos del anexo C, y actualizar los controles apropiados de la caja de diálogo principal de la aplicación para que proporcionen las unidades correctas.

afx_msg void OnCcoefexp(); Es el método que responde al evento por presionar el botón “Cambiar…” “Coef. Exp”. de la caja de diálogo principal de la aplicación. Como la descripción lo indica, el método se encarga de desplegar la caja de diálogo auxiliar “Nuevo coeficiente de expansión” en donde el usuario puede establecer dicho coeficiente y enviarlo mediante los comandos GPIB del anexo C hacia el láser HP5528A.

afx_msg void OnCprecarga(); Es el método que responde al evento por presionar el botón “Cambiar…” “Precarga” de la caja de diálogo principal de la aplicación. Como la descripción lo indica, el método se encarga de desplegar la caja de diálogo auxiliar “Nueva precarga” en donde el usuario puede establecer dicho valor de precarga o desplazamiento y enviarlo mediante los comandos GPIB del anexo C hacia el láser HP5528A.

afx_msg void OnCresolucion(); Es el método que responde al evento por presionar el botón “Cambiar…” “Resolución” de la caja de diálogo principal de la aplicación. Como la descripción lo indica, el método se encarga de desplegar la caja de diálogo auxiliar

“Nueva resolución” en donde el usuario puede establecer dicho valor de resolución en dígitos decimales significativos y enviarlo mediante los comandos GPIB del anexo C hacia el láser HP5528A.

afx_msg void OnCtemp(); Es el método que responde al evento por presionar el botón “Cambiar…” “Promedio” de la caja de diálogo principal de la aplicación. Como la descripción lo indica, el método se encarga de desplegar la caja de diálogo auxiliar “Nueva temperatura promedio” en donde el usuario puede establecer dicho valor de temperatura y enviarlo mediante los comandos GPIB del anexo C hacia el láser HP5528A.

afx_msg void OnCvol(); Es el método que responde al evento por presionar el botón “Cambiar V.O.L…” de la caja de diálogo principal de la aplicación. Como la descripción lo indica, el método se encarga de desplegar la caja de diálogo auxiliar “Nuevo número V.O.L.” en donde el usuario puede establecer dicho valor de V.O.L. y enviarlo mediante los comandos GPIB del anexo C hacia el láser HP5528A.

afx_msg void OnDestroy(); Es el manejador del evento por terminar la aplicación. En este método se cierra la caja de diálogo principal y se detiene el temporizador que ejecuta tareas periódicas.

afx_msg void OnDistancia(); Es el método que responde al evento por presionar el radio botón “Distancia” de la caja de diálogo principal de la aplicación. Como la descripción lo indica, el método se encarga de establecer en el láser HP5528A el modo de medición en “Distancia” mediante los comandos GPIB del anexo C y actualizar los controles apropiados de la caja de diálogo principal de la aplicación para que proporcionen las unidades correctas.

afx_msg void OnHtemp(); Es el método que responde a los eventos por presionar los radio botones “Habilitar” “Sensor 1 a 3” de la caja de diálogo principal de la aplicación. Como la descripción lo indica, el método se encarga de habilitar la compensación de la medición por variaciones en las lecturas de los sensores de temperatura 1 a 3 y envía la instrucción mediante los comandos GPIB del anexo C al láser HP5528A.

afx_msg void OnHvol(); Es el método que responde al evento por presionar el radio botón “Habilita compensación” de la caja de diálogo principal de la aplicación. Como la descripción lo indica, el método se encarga de habilitar la compensación V.O.L. de la medición por variaciones en las condiciones ambientales del aire y envía la instrucción mediante los comandos GPIB del anexo C al láser HP5528A.

afx_msg void OnIngles(); Es el método que responde al evento por presionar el radio botón “Inglés” de la caja de diálogo principal de la aplicación. Como la descripción lo indica, el método se encarga de establecer el sistema inglés de unidades y envía la instrucción mediante los comandos GPIB del anexo C al láser HP5528A.

virtual BOOL OnInitDialog(); Es el evento por omisión que se ejecuta al iniciar la caja de diálogo principal de la aplicación. En este método se inicializa la interfaz GPIB mediante

los comandos “ibdev” e “ibclr” que se explican en el anexo B. Además, se establece la conexión COM con Excel a partir del registro de Windows. También se inicializa un temporizador de 200 ms de periodo que permite la ejecución de eventos periódicos que son atendidos por el manejador de eventos “OnTimer”.

afx_msg void OnMetrico(); Es el método que responde al evento por presionar el radio botón “Métrico” de la caja de diálogo principal de la aplicación. Como la descripción lo indica, el método se encarga de establecer el sistema métrico de unidades y envía la instrucción mediante los comandos GPIB del anexo C al láser HP5528A.

afx_msg void OnPaint(); Es el dispositivo de contexto por omisión que introduce el ambiente de programación para pintar en la aplicación. Su código original no es modificado.

afx_msg HCURSOR OnQueryDragIcon(); Es el método por omisión que introduce el ambiente de programación para desplegar el cursor mientras el usuario desplaza la ventana minimizada. Su código original no es modificado.

afx_msg void OnRectitudcorta(); Es el método que responde al evento por presionar el radio botón “Rectitud corta” de la caja de diálogo principal de la aplicación. Como la descripción lo indica, el método se encarga de establecer en el láser HP5528A el modo de medición en “Rectitud corta” mediante los comandos GPIB del anexo C y actualizar los controles apropiados de la caja de diálogo principal de la aplicación para que proporcionen las unidades correctas.

afx_msg void OnRectitudlarga(); Es el método que responde al evento por presionar el radio botón “Rectitud larga” de la caja de diálogo principal de la aplicación. Como la descripción lo indica, el método se encarga de establecer en el láser HP5528A el modo de medición en “Rectitud larga” mediante los comandos GPIB del anexo C y actualizar los controles apropiados de la caja de diálogo principal de la aplicación para que proporcionen las unidades correctas.

afx_msg void OnReset(); Es el método que responde al evento por presionar el botón “Reset” de la caja de diálogo principal de la aplicación. Como la descripción lo indica, el método se encarga de enviar el comando GPIB “Reset” del anexo C hacia el láser HP5528A.

afx_msg void OnSentido(); Es el método que responde al evento por presionar el botón “Sentido” de la caja de diálogo principal de la aplicación. Como la descripción lo indica, el método se encarga de enviar el comando GPIB del anexo C para establecer el sentido de la medición, ya sea positivo o negativo respecto al desplazamiento de la óptica asociada a la medición del láser HP5528A.

afx_msg void OnSysCommand(UINT nID, LPARAM lParam); Es el método por omisión que introduce el ambiente de programación para administrar el despliegue de la caja de diálogo acerca de los derechos de autor. Su código original no es modificado.

afx_msg void OnTimer(UINT nIDEvent); Es el mensaje de Windows que se ejecuta periódicamente para actualizar la medición en el despliegue de la caja de diálogo principal de la aplicación.

afx_msg void OnVelocidad(); Es el método que responde al evento por presionar el radio botón “Velocidad” de la caja de diálogo principal de la aplicación. Como la descripción lo indica, el método se encarga de establecer en el láser HP5528A el modo de medición en “Velocidad” mediante los comandos GPIB del anexo C y actualizar los controles apropiados de la caja de diálogo principal de la aplicación para que proporcionen las unidades correctas.

CCoefExp

La figura 11 muestra los métodos y variable de la clase CCoefExp.

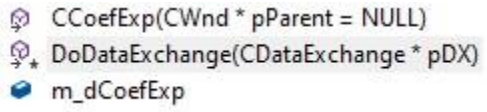


Figura 11. Métodos de la clase CCoefExp.

A continuación, la descripción detallada de cada uno de los métodos de la figura 11. El código fuente completo de la clase se puede revisar en el anexo A.

CCoefExp(CWnd* pParent = NULL); Es el constructor estándar de la caja de diálogo “Nuevo coeficiente de expansión” usado para modificar la variable m_dCoefExp.

virtual void DoDataExchange(CDataExchange* pDX); Es el método estándar que permite el intercambio del dato en la variable m_dCoefExp con otras clases del proyecto.

CCprecarga

La figura 12 muestra los métodos y variables de la clase CCprecarga.

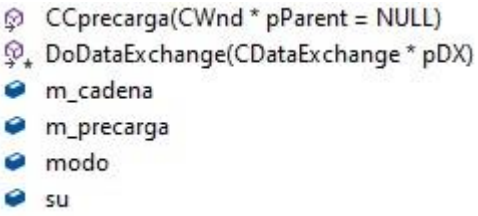


Figura 12. Métodos de la clase CCprecarga.

A continuación, la descripción detallada de cada uno de los métodos de la figura 12. El código fuente completo de la clase se puede revisar en el anexo A.

CCprecarga(CWnd* pParent = NULL); Es el constructor estándar de la caja de diálogo “Nueva precarga” usado para modificar la variable m_precarga.

virtual void DoDataExchange(CDataExchange* pDX); Es el método estándar que permite el intercambio del dato en la variable m_precarga con otras clases del proyecto, dependiendo del modo de medición.

CCtempprom

La figura 13 muestra los métodos y variables de la clase CCtempprom.

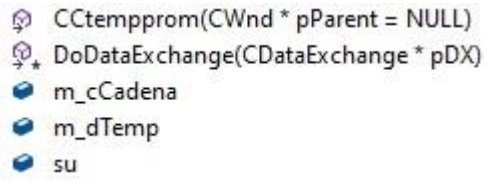


Figura 13. Métodos de la clase CCtempprom.

A continuación, la descripción detallada de cada uno de los métodos de la figura 13. El código fuente completo de la clase se puede revisar en el anexo A.

CCtempprom(CWnd* pParent = NULL); Es el constructor estándar de la caja de diálogo “Nueva temperatura promedio” usado para modificar la variable m_dTemp.

virtual void DoDataExchange(CDataExchange* pDX); Es el método estándar que permite el intercambio del dato en la variable m_dTemp con otras clases del proyecto, dependiendo del sistema de unidades en uso.

CCvol

La figura 14 muestra los métodos y variable de la clase CCvol.

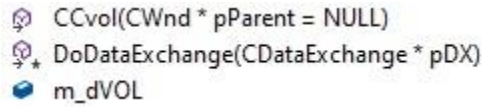


Figura 14. Métodos de la clase CCvol.

A continuación, la descripción detallada de cada uno de los métodos de la figura 14. El código fuente completo de la clase se puede revisar en el anexo A.

CCvol(CWnd* pParent = NULL); Es el constructor estándar de la caja de diálogo “Nuevo número V.O.L.” usado para modificar la variable m_dVOL.

virtual void DoDataExchange(CDataExchange* pDX); Es el método estándar que permite el intercambio del dato en la variable m_dVOL con otras clases del proyecto.

CResolucion

La figura 15 muestra los métodos y variables de la clase CResolución.

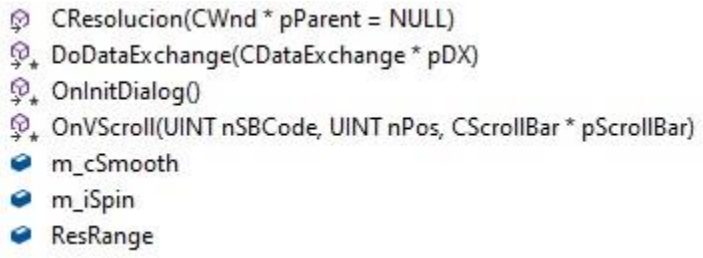


Figura 15. Métodos de la clase CResolucion.

A continuación, la descripción detallada de cada uno de los métodos de la figura 15. El código fuente completo de la clase se puede revisar en el anexo A.

CResolucion(CWnd* pParent = NULL); Es el constructor estándar de la caja de diálogo “Nueva resolución” usado para modificar las variables m_cSmooth y m_iSpin.

virtual void DoDataExchange(CDataExchange* pDX); Es el método estándar que permite el intercambio de los datos en las variables m_cSmooth y m_iSpin con otras clases del proyecto, dependiendo del modo de medición en el HP5528A.

virtual BOOL OnInitDialog(); Es el manejador del evento por crear la caja de diálogo “Nueva resolución” en donde se inicializa un control del tipo “Spin” que permite incrementar o decrementar su valor asociado a la variable m_iSpin mediante la pulsación de flechas hacia arriba o hacia abajo.

afx_msg void OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar); Es el manejador de los eventos por pulsar hacia arriba o abajo las flechas del control “Spin”.

1.3.4 Recursos del proyecto

La figura 16 muestra la vista de recursos conteniendo las cajas de diálogo desarrolladas en el proyecto de programación “Láser”. La lista contiene 7 cajas de diálogo utilizadas para interactuar con el usuario en los momentos en donde se requiere mostrar o ingresar información. En estas cajas de diálogo se utilizan diversos controles estándar de los programas elaborados sobre la base de la plataforma Windows, como son: cajas de texto, radio botones, botones, cajas de verificación, cajas combo, cajas de agrupación y controles del tipo spin. Para que el código fuente en lenguaje de programación C pueda interactuar con estos controles, se debe asignar a cada uno de estos recursos un número de identificación, ID, para posteriormente establecer los

métodos o variables que los representan en el código fuente. En el esquema de programación adoptado para el proyecto “Láser”, la pantalla principal de operación es una caja de diálogo que a su vez manipula otras cajas de diálogo principalmente para ingresar información. La jerarquización entre las siete cajas de diálogo desarrolladas se muestra en la figura 17.

A continuación, una descripción de cada caja de diálogo y los controles que contienen.

Figura 16. Vista de recursos del proyecto “Láser”.

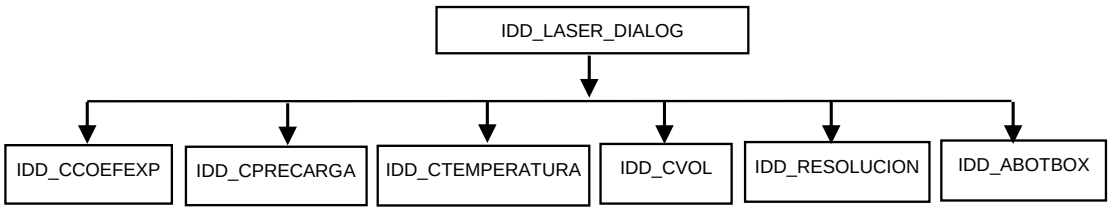


Figura 17. Jerarquía entre las cajas de diálogo del proyecto “Láser”.

IDD_ABOUTBOX

La caja de diálogo “Acerca de Láser”, mostrada en la figura 18, es manipulada mediante el identificador IDD_ABOUTBOX. Es un diálogo simple que muestra información acerca de los derechos de los autores y contiene el botón “Aceptar” para cerrar el diálogo.

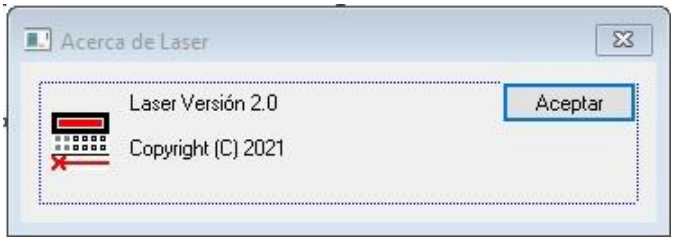


Figura 18. Caja de diálogo “Acerca de Láser”.

IDD_CCOEFEXP

La caja de diálogo “Nuevo coeficiente de expansión”, mostrada en la figura 19, es manipulada mediante el identificador IDD_CCOEFEXP y su clase asociada es CCoefExp, las cual fue descrita en la sección 1.3.3. Esta caja de diálogo es invocada desde la pantalla principal del programa “Láser” para modificar el coeficiente de expansión con el cual opera el láser HP5528A.

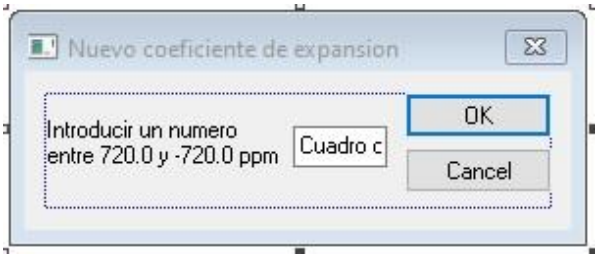


Figura 19. Caja de diálogo “Nuevo coeficiente de expansión”.

IDD_CPRECARGA

La caja de diálogo “Nueva precarga”, mostrada en la figura 20, es manipulada mediante el identificador IDD_CPRECARGA y su clase asociada es Cprecarga, la cual fue descrita en la sección 1.3.3. Esta caja de diálogo es invocada desde la pantalla principal del programa “Láser” para modificar la precarga u “offset” con el cual opera el láser HP5528A.

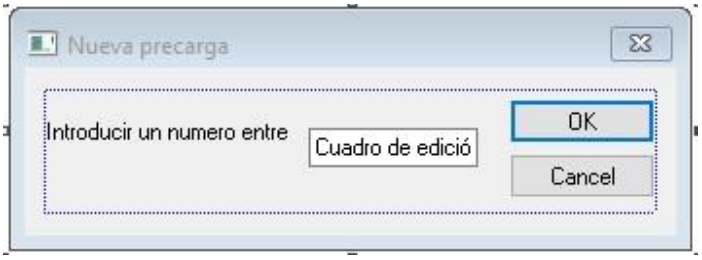


Figura 20. Caja de diálogo “Nueva precarga”.

IDD_CTEMPERATURA

La caja de diálogo “Nueva temperatura promedio”, mostrada en la figura 21, es manipulada mediante el identificador IDD_CTEMPERATURA y su clase asociada es Ctempprom, la cual fue descrita en la sección 1.3.3. Esta caja de diálogo es invocada desde la pantalla principal del programa “Láser” para modificar el valor de la temperatura promedio con el cual opera el láser HP5528A.

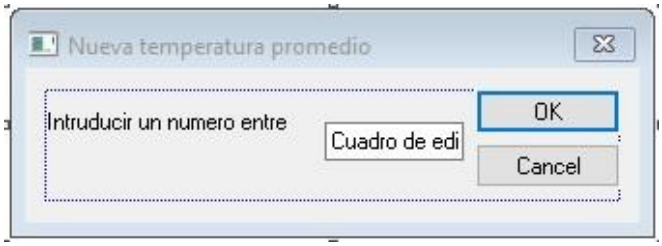


Figura 21. Caja de diálogo "Nueva precarga".

IDD_CVOL

La caja de diálogo "Nuevo número V.O.L.", mostrada en la figura 22, es manipulada mediante el identificador IDD_CVOL y su clase asociada es CCvol, la cual fue descrita en la sección 1.3.3. Esta caja de diálogo es invocada desde la pantalla principal del programa "Láser" para modificar el valor del número V.O.L con el cual opera el láser HP5528A.

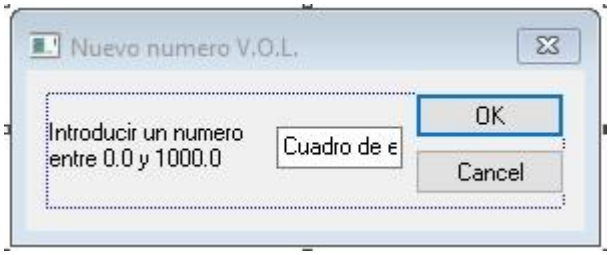


Figura 22. Caja de diálogo "Nuevo número V.O.L.".

IDD_LASER_DIALOG

La caja de diálogo "Láser", mostrada en la figura 23, es la pantalla principal de la aplicación a partir de la cual se invocan las restantes cajas de diálogo en la jerarquía de la figura17. En esta caja de diálogo existen 37 controles, identificados con su número ID con texto en color rojo, que pueden ser manipulados a través de eventos o variables para controlar el flujo del algoritmo de medición y control de la aplicación "Láser".

En primer lugar, existen 10 botones, excluyendo los botones de "Ayuda" y "Cancelar", que están asociados a eventos programados como parte de la clase CLaserDlg descrita en la sección 1.3.3. La tabla 2 muestra los números de identificación de cada uno de los botones de la figura 23 y sus eventos asociados.

En segundo lugar, existen 11 radio botones que también están asociados a eventos programados en la misma clase CLaserDlg que se describió en la sección 1.3.3. La tabla 3 muestra los números de identificación de cada uno de los radio botones de la figura 23 y sus eventos asociados.

En tercer lugar, 9 de los radio botones pueden o no tener un evento asociado pero tienen una variable asociada que es utilizada en el algoritmo de medición y control para desempeñar adecuadamente el flujo del programa y actualizar variables en los despliegues. La tabla 4 muestra los números de identificación de cada uno de los radio botones de la figura 23 y sus variables asociadas.

En cuarto lugar, existen 14 cajas de texto que se utilizan para desplegar información y que están asociadas a variables que son modificadas de acuerdo con el flujo del algoritmo de medición y control de la clase CLaserDlg, como se describió en la sección 1.3.3. La tabla 5 muestra los números de identificación de cada una de las cajas de texto de la figura 23 y sus variables asociadas.

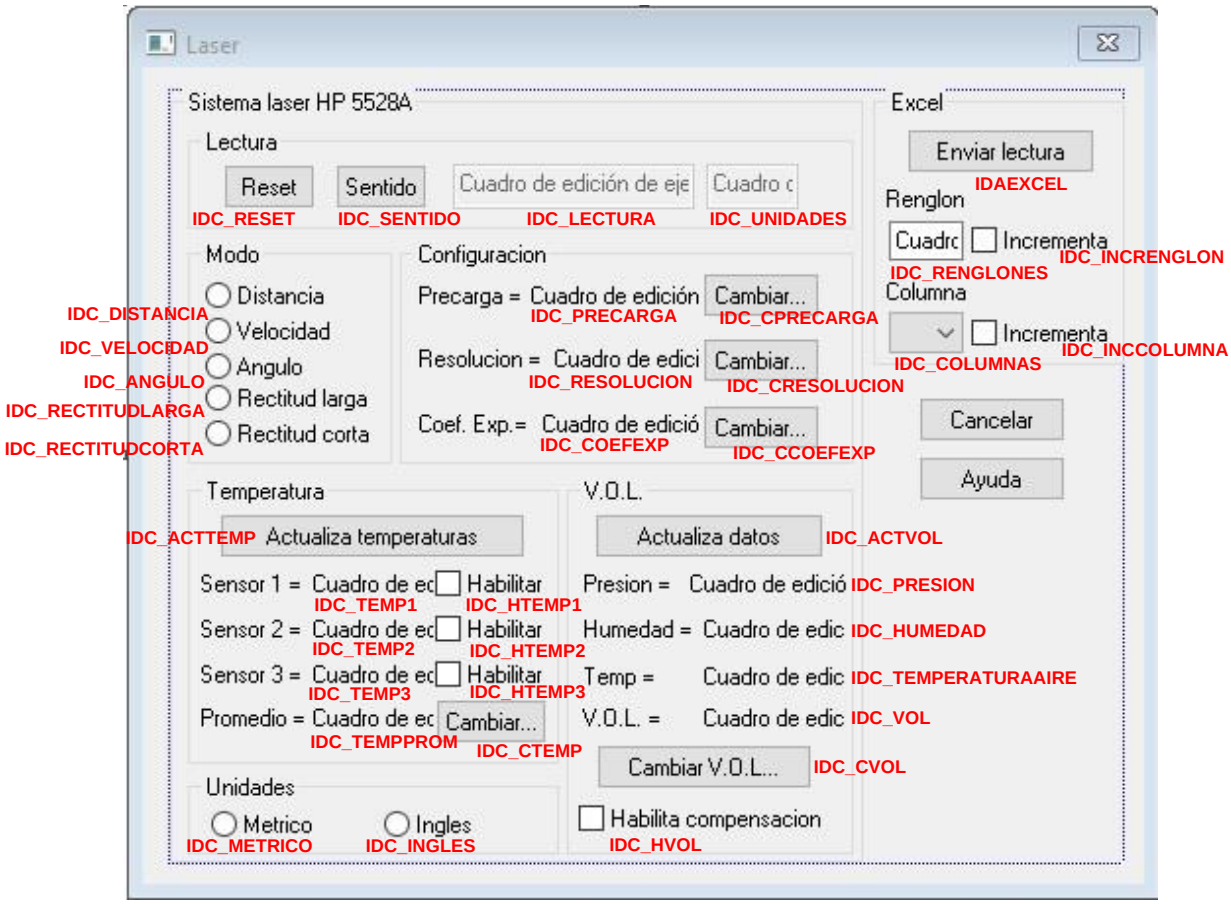


Figura 23. Caja de diálogo "Láser" y controles con número de identificación en texto de color rojo.

Número de identificación	**Evento en la clase CLaserDlg**
IDAEXCEL	OnAexcel
IDC_RESET	OnReset
IDC_SENTIDO	OnSentido
IDC_CCOEFEXP	OnCcoefexp
IDC_CRESOLUCION	OnCresolucion

IDC_CPRECARGA	OnCprecarga
IDC_ACTTEMP	OnActtemp
IDC_CTEMP	OnCtemp
IDC_ACTVOL	OnActvol
IDC_CVOL	OnCvol

Tabla 2. Botones de la caja de diálogo "Láser" y sus eventos asociados.

Número de identificación	**Evento en la clase CLaserDlg**
IDC_DISTANCIA	OnDistancia
IDC_METRICO	OnMetrico
IDC_INGLES	OnIngles
IDC_VELOCIDAD	OnVelocidad
IDC_ANGULO	OnAngulo
IDC_RECTITUDLARGA	OnRectitudlarga
IDC_RECTITUDCORTA	OnRectitudcorta
IDC_HTEMP1	OnHtemp
IDC_HTEMP2	OnHtemp
IDC_HTEMP3	OnHtemp
IDC_HVOL	OnHvol

Tabla 3. Radio botones de la caja de diálogo "Láser" y sus eventos asociados.

Número de identificación	**Variable en la clase CLaserDlg**
IDC_INCCOLUMNA	m_bIncColumna
IDC_INCRENGLON	m_bIncRenglon
IDC_COLUMNAS	m_iColumnas
IDC_DISTANCIA	m_iModo
IDC_METRICO	m_iSU
IDC_HTEMP1	m_bhtemp1
IDC_HTEMP2	m_bhtemp2
IDC_HTEMP3	m_bhtemp3
IDC_HVOL	m_HVOL

Tabla 4. Radio botones de la caja de diálogo "Láser" y sus variables asociadas.

Número de identificación	**Variable en la clase CLaserDlg**
IDC_RENGLONES	m_iRenglones
IDC_LECTURA	m_Lectura
IDC_UNIDADES	m_CUnidades
IDC_COEFEXP	m_cCoefExp
IDC_RESOLUCION	m_cResolucion
IDC_PRECARGA	m_precarga
IDC_TEMP1	m_cTemp1
IDC_TEMP2	m_cTemp2

IDC_TEMP3	m_cTemp3
IDC_TEMPPROM	m_cTempProm
IDC_PRESION	m_cPresion
IDC_HUMEDAD	m_cHumedad
IDC_TEMPERATURAAIRE	m_cTempAire
IDC_VOL	m_cVOL

Tabla 5. Cajas de texto de la caja de diálogo "Láser" y sus variables asociadas.

IDD_RESOLUCION

La caja de diálogo "Nueva resolución", mostrada en la figura 24, es manipulada mediante el identificador IDD_RESOLUCION y su clase asociada CResolucion descrita en la sección 1.3.3. Esta caja de diálogo es invocada desde la pantalla principal del programa "Láser" para modificar la resolución o cantidad de dígitos decimales de las mediciones con las que opera el láser HP5528A.

Figura 24. Caja de diálogo "Nueva resolución".

1.4 Procedimiento de instalación

Se instaló un conjunto de aplicaciones en una computadora LapTop nueva para destinarla, como interfaz de operación, al sistema láser de medición HP5528A. El sistema operativo de la LapTop es Windows 10 y las aplicaciones que se instalaron son:

- Visual Studio 2019.
- Office 365.
- NI-488.2.

Posteriormente a las instalaciones mencionadas, se procedió a elaborar el código fuente descrito en la sección 1.3. A continuación, una descripción de la paquetería instalada.

1.4.1 Visual Studio 2019

Para la instalación de Visual Studio 2019 se adquirió la licencia del software con un proveedor de Microsoft en México. La respuesta recibida a la compra fue un correo

electrónico de Microsoft en donde se proporciona un acceso mediante un enlace a su VLSC (Volume Licensing Service Center, Centro de Licenciamiento por Volumen), previo registro y validación de los datos del comprador. Al acceder al enlace se debe iniciar sesión ingresando el correo electrónico proporcionado al proveedor y la contraseña creada para tal efecto. La figura 25 muestra la página principal del VLSC de Microsoft después de haber iniciado sesión.

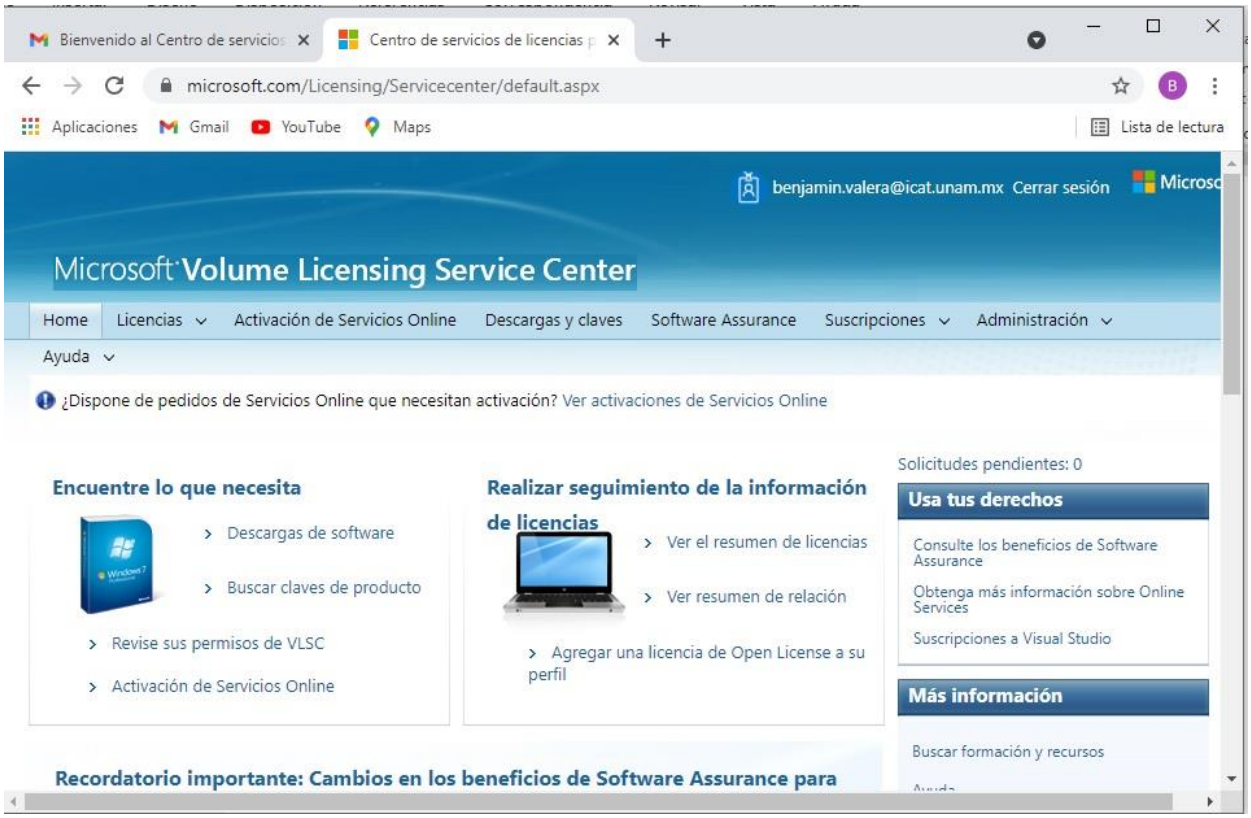


Figura 25. Página principal de VLSC.

En la página de la figura 25 se debe ir al enlace “Descargas de software” para acceder a la página “Descargas y claves” de la figura 26. En esta página se deberán realizar dos procesos:

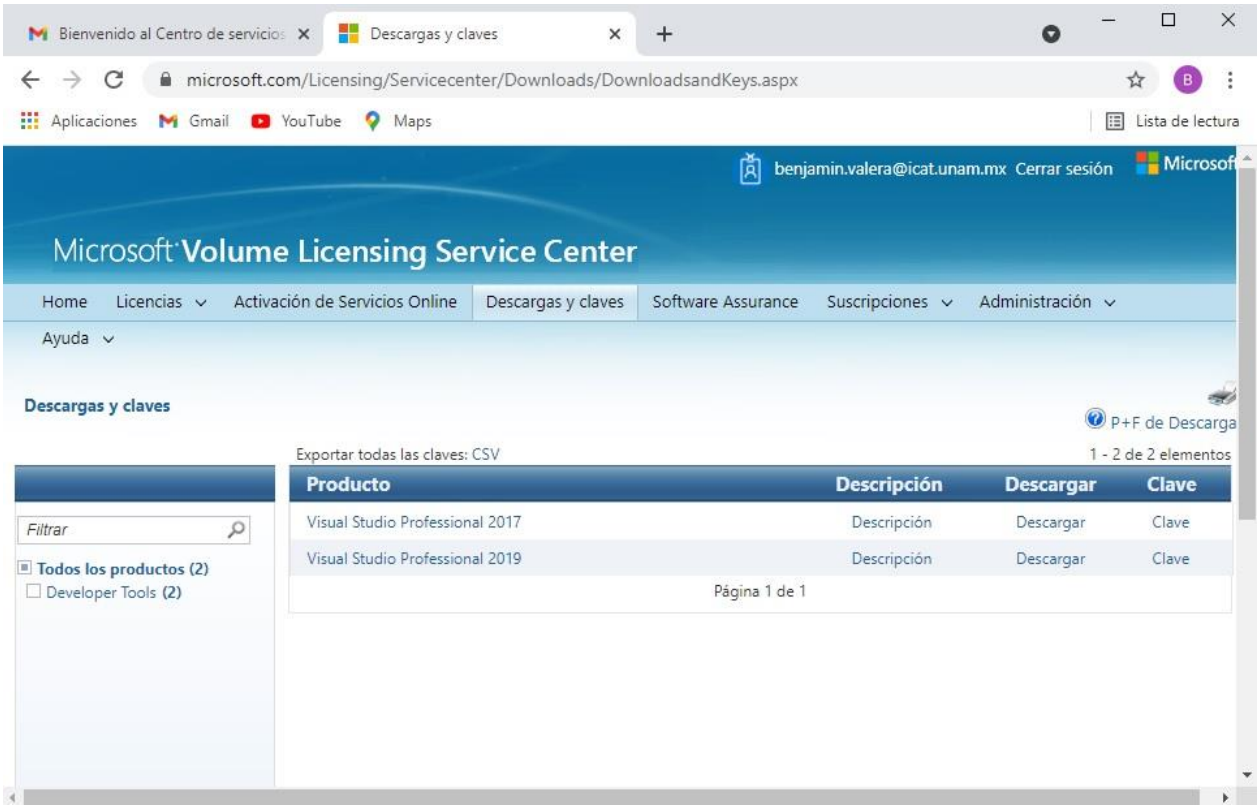


Figura 26. Página de descargas y claves de VLSC.

En primer lugar, presionar sobre el enlace “Descargar” del producto en cuestión y descargar el archivo ejecutable instalador del software. Posteriormente se debe ejecutar el archivo descargado, en modo administrador, y seguir las instrucciones hasta completar la instalación de Visual Studio 2019. En algún punto del proceso de instalación se desplegará una ventana similar a la de la figura 27 en donde se deberá seleccionar “Desktop Development with C++” y la totalidad de opciones en el panel derecho “Summary”, con el fin de asegurar la correcta instalación del software asociado a la programación en lenguaje de Visual C++.

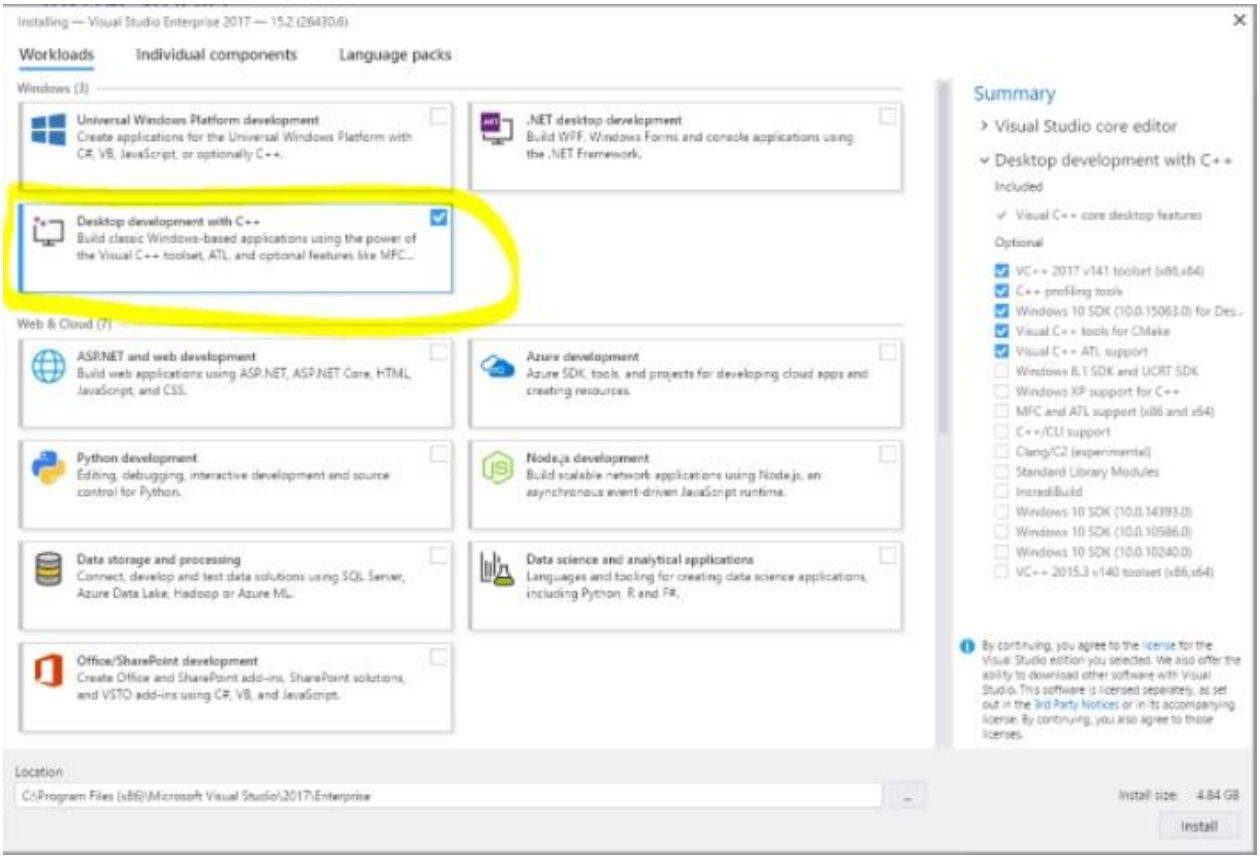


Figura 27. Página de instalación de Visual Studio 2019.

El segundo proceso por realizar es presionar el enlace “Clave” del producto en cuestión de la figura 26 para obtener la pantalla de la figura 28. En la pantalla de la figura 28 se podrá obtener la clave del producto para posteriormente registrarlo en la opción del menú “Ayuda/Registrar Producto” de Visual Studio 2019.

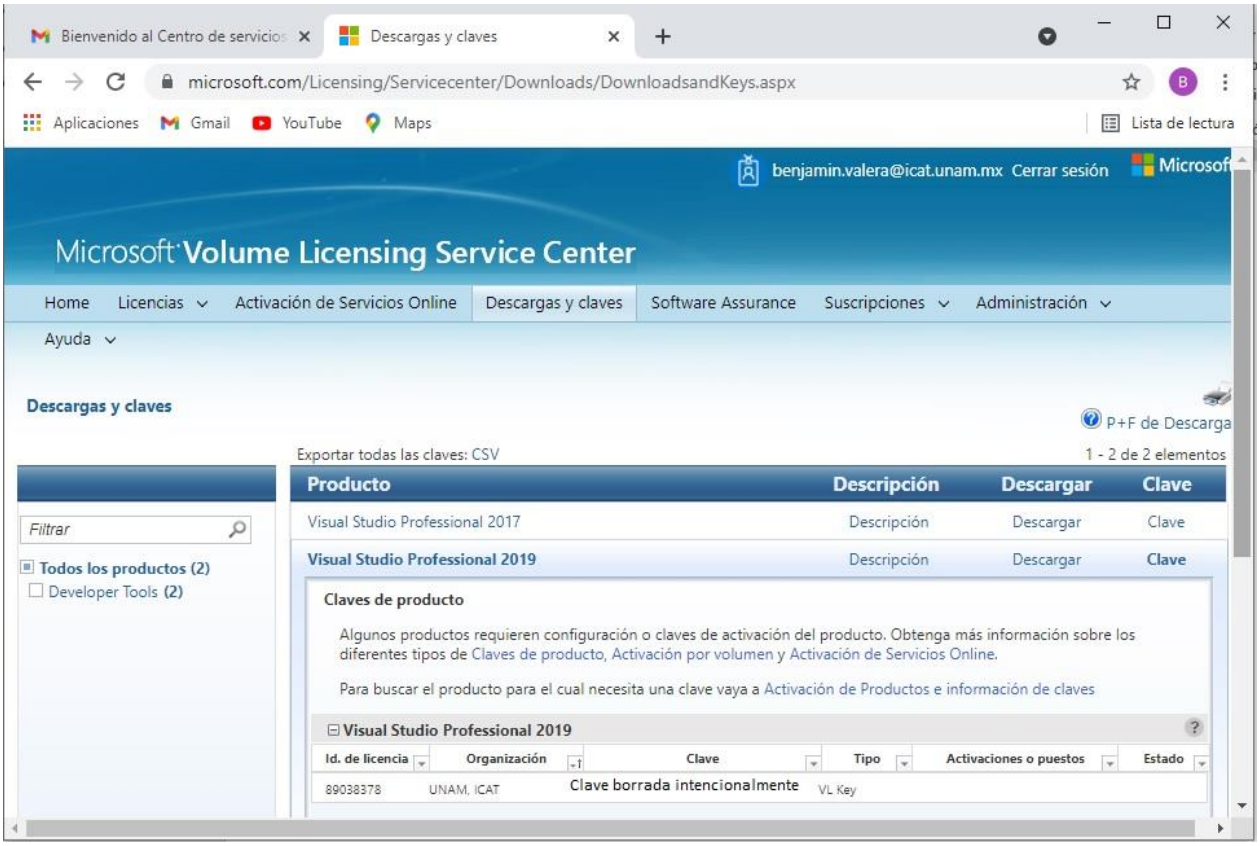


Figura 28. Página para obtener la clave de Visual Studio 2019.

1.4.2 Office 365

La instalación de Office 365 se realiza a partir de una cuenta en el dominio de comunidad.unam.mx, ya que nuestra institución, la UNAM, ofrece este servicio gratuito a la comunidad de alumnos y académicos. La figura 29 muestra el inicio de sesión en la cuenta del usuario en “Comunidad UNAM”. En esta pantalla se deberá seleccionar la opción del icono redondo en la parte superior derecha para acceder a la segunda opción “Ver cuenta”, como se muestra en la figura 29. Al realizar el proceso descrito, se abrirá una ventana emergente como la mostrada en la figura 30, en donde se podrá acceder a la instalación de Office 365 mediante el enlace “ADMINISTRAR” de “Aplicaciones de Office”. En éste último enlace, se podrá acceder al enlace final que permite la descarga del ejecutable instalador para posteriormente ejecutarlo en modo administrador y seguir las instrucciones para finalizar el proceso.

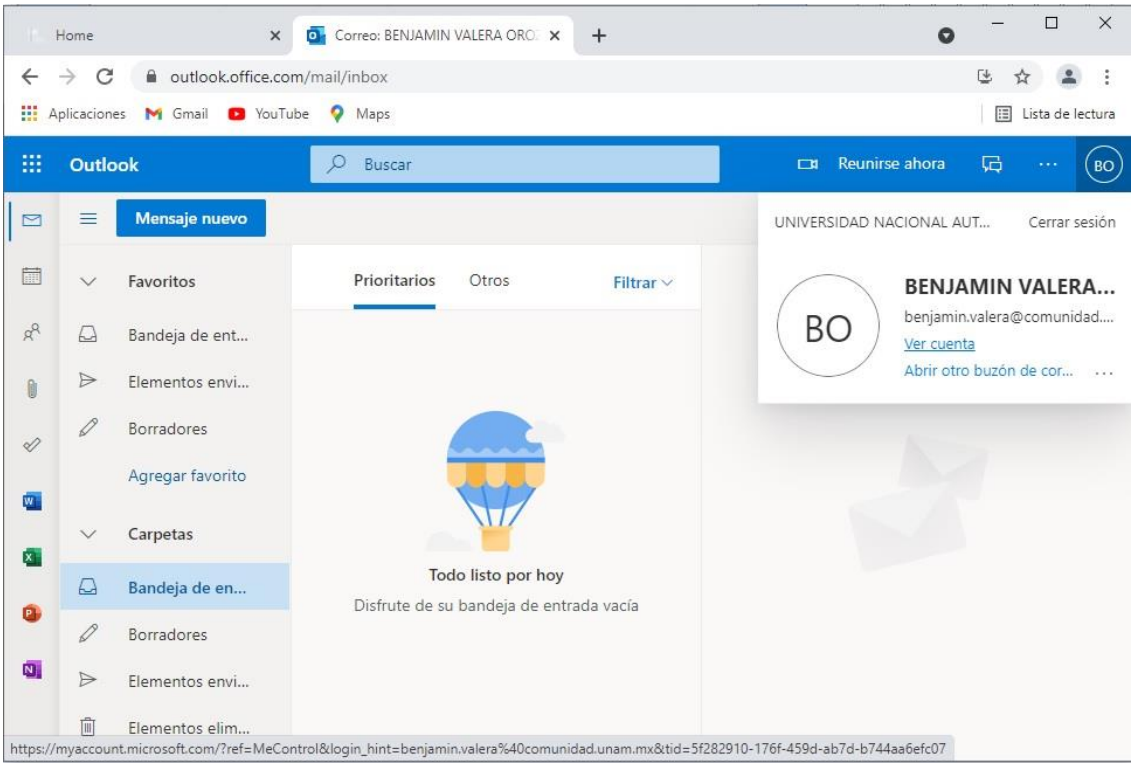


Figura 29. Inicio de sesión en la cuenta de "Comunidad UNAM".

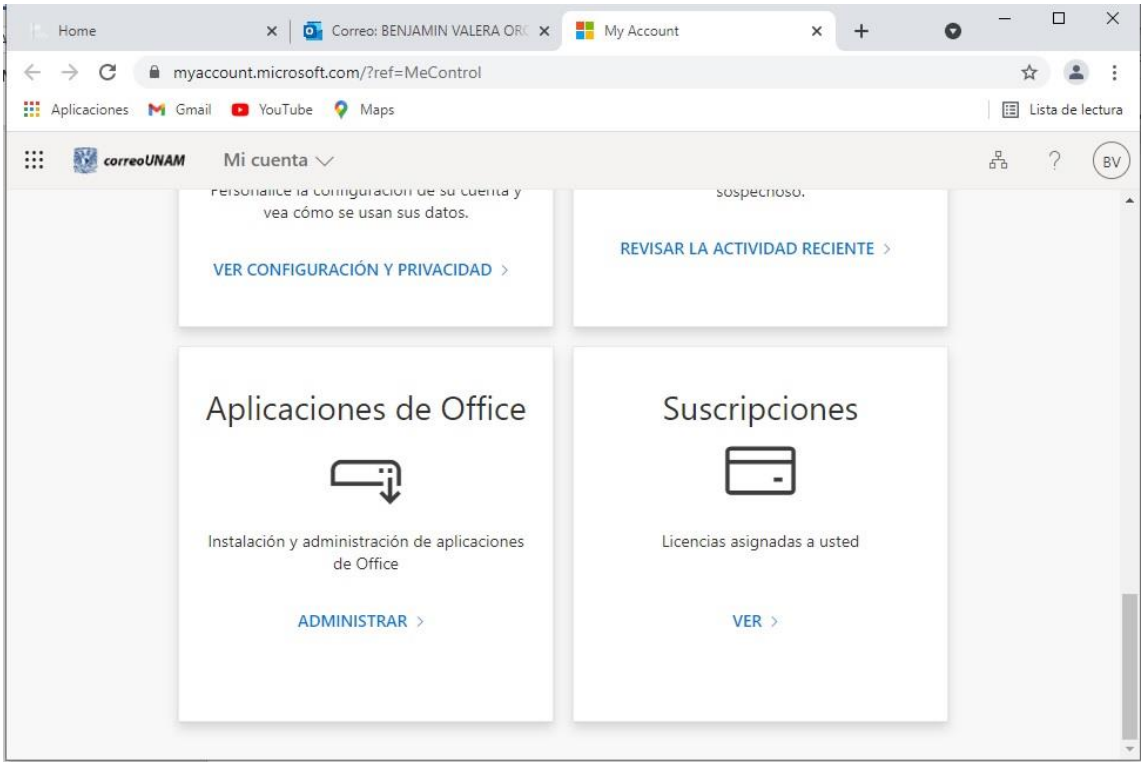


Figura 30. Ventana emergente "My Account" de "Comunidad UNAM".

1.4.3 NI-488.2

Para la instalación de la tarjeta de interfaz GPIB-USB-HS de la figura 6, se requiere ejecutar el programa instalador proporcionado por el fabricante National Instruments en disco óptico, o en el caso en donde la computadora no tenga unidad de disco, copiar el disco a una memoria USB y ejecutarlo desde este nuevo medio. El proceso de instalación es sencillo y simplemente se requiere esperar a que finalice la instalación para reiniciar la computadora y conectar la tarjeta de interfaz al conector USB de la computadora huésped. En este momento se deberá comprobar la correcta instalación al ejecutar el software “NI MAX” “Measurement & Automation Explorer”, como se muestra en la figura 31.

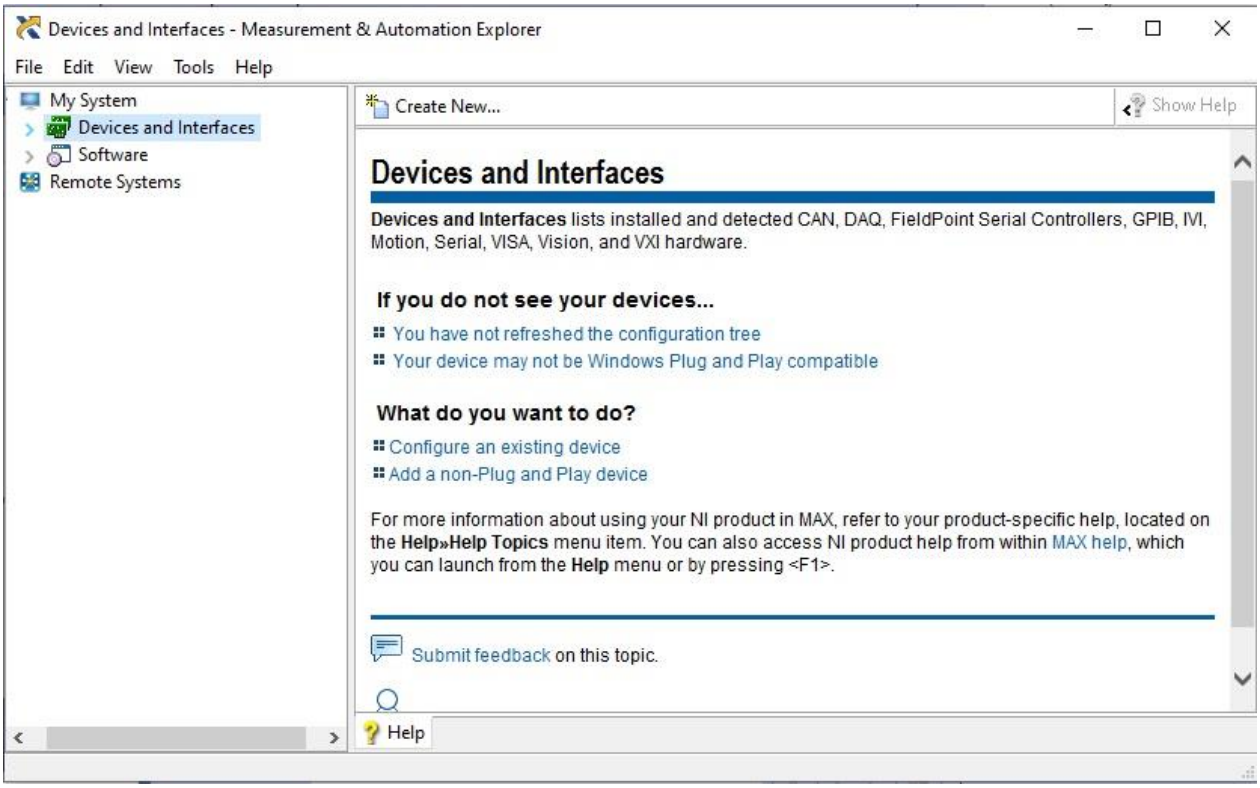


Figura 31. Measurement & Automation Explorer.

En esta pantalla se debe seleccionar la opción “Devices and Interfaces” del panel izquierdo para desplegar la pantalla de la figura 32 en donde, si la instalación fue correcta, se deberá desplegar la opción “NI-GPIB-USB-HS+ GPIB0”.

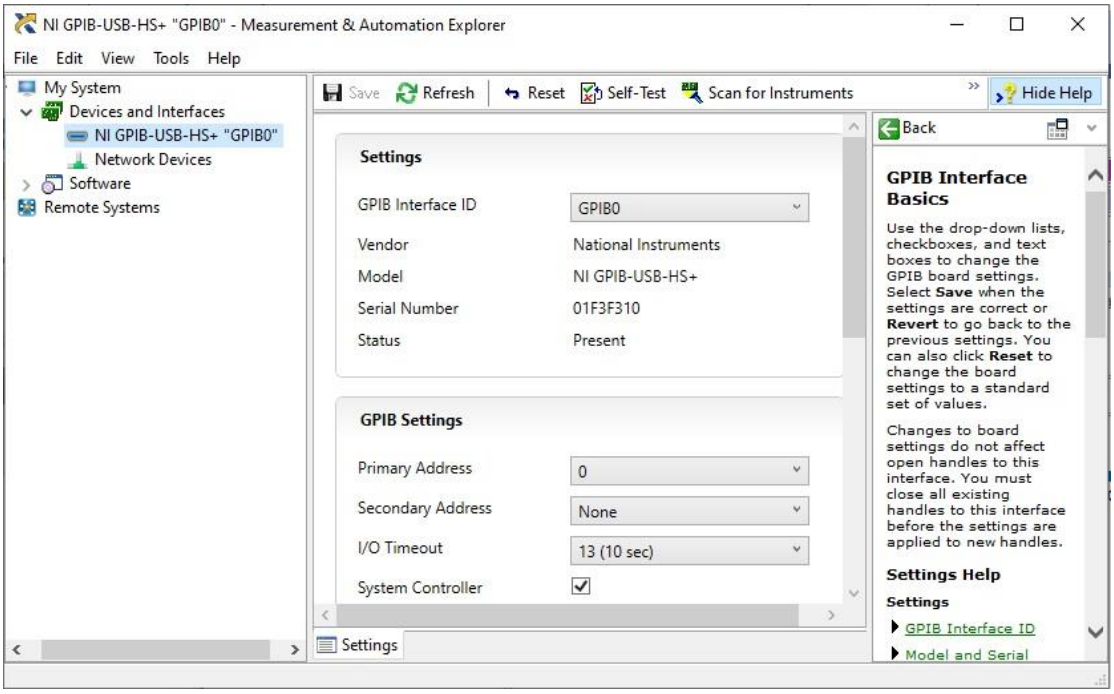


Figura 32. Devices and Interfaces.

En esta pantalla, al expandir la opción “NI-GPIB-USB-HS+ GPIB0” se mostrará información importante acerca de la dirección asignada por omisión a la tarjeta. La figura 33 indica que la tarjeta fue asignada con la dirección 3, es decir, “Primary Address 3”. Esta dirección será utilizada tanto en las pruebas de conexión como en el software específico para tener acceso a la funcionalidad y programación de la tarjeta.

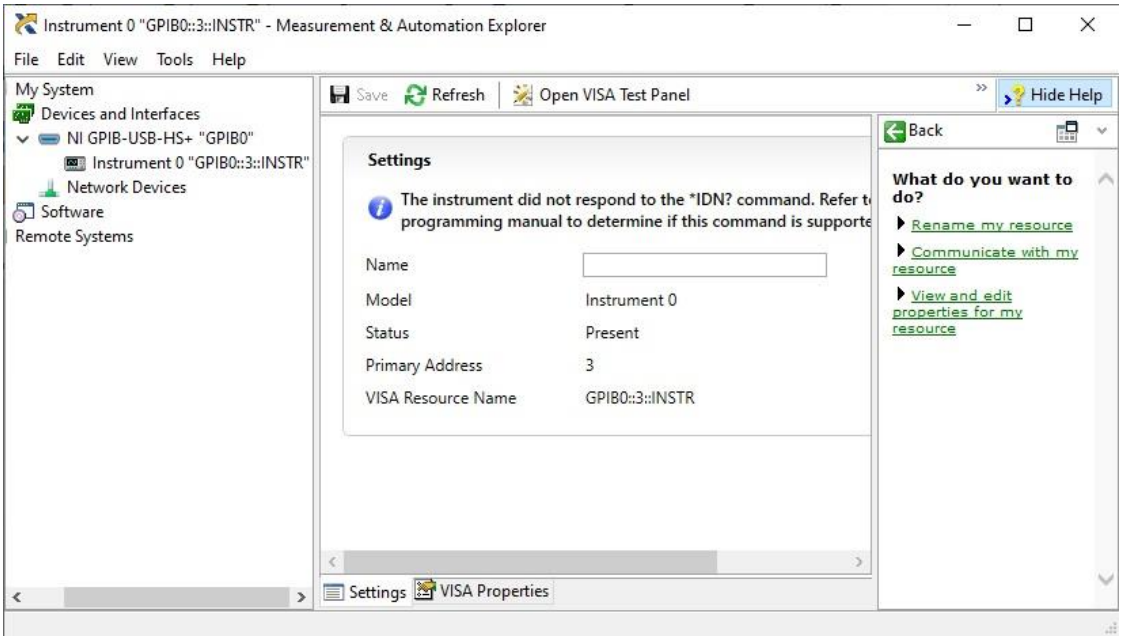


Figura 33. NI-GPIB-USB-HS+ GPIB0.

En esta pantalla se deberá presionar el botón derecho del ratón sobre la opción “Instrument 0 GPIB0::3::INSTR” para mostrar el menú flotante de la figura 34, en donde se podrá ingresar al software “Interactive Control” para interactuar con la tarjeta mediante comandos estándar del protocolo GPIB.

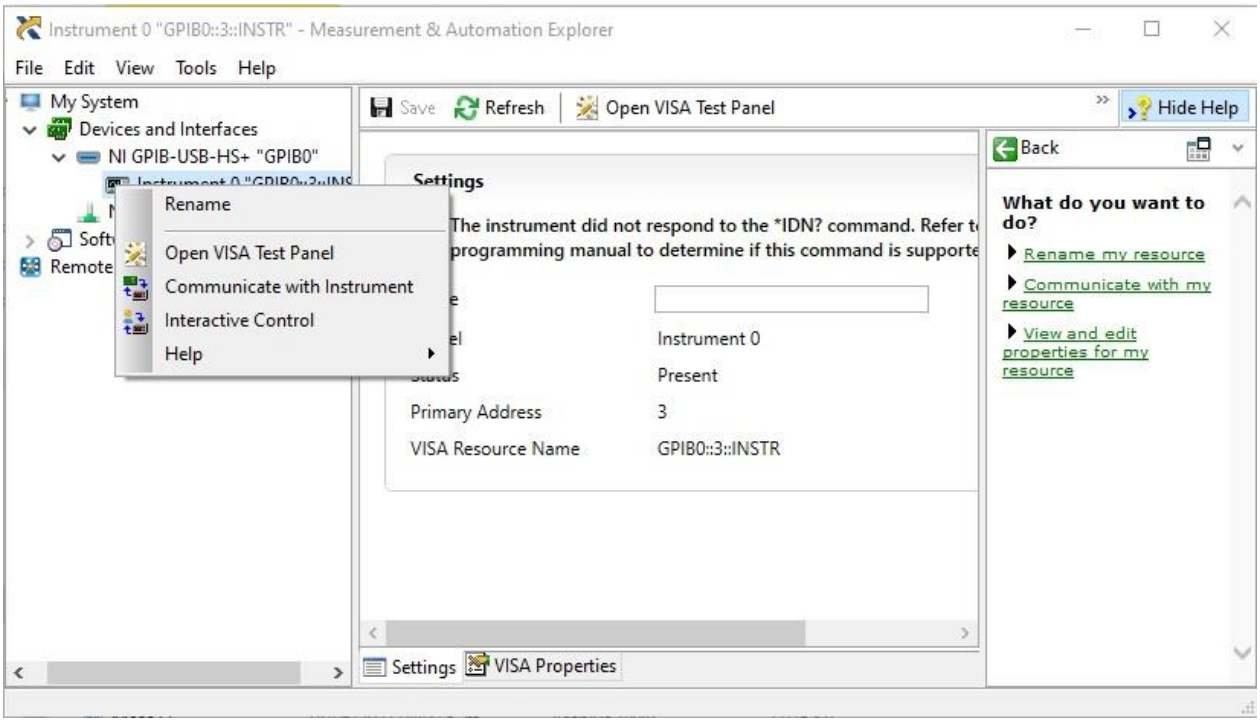


Figura 34. Interactive Control.

Al realizar el proceso descrito, se abrirá una nueva pantalla en la línea de comandos como se muestra en la figura 35. En este momento se podrán ingresar los siguientes comandos GPIB que comprueban el funcionamiento de la tarjeta:

- ibclr. Envía la instrucción para limpiar el dispositivo.
- ibtrg. Envía la instrucción de disparo al dispositivo.
- ibrd. Envía la instrucción para la lectura de bytes de datos desde el dispositivo.

Si las operaciones fueron exitosas y libres de error, entonces el despliegue debe ser conforme a la figura 35.

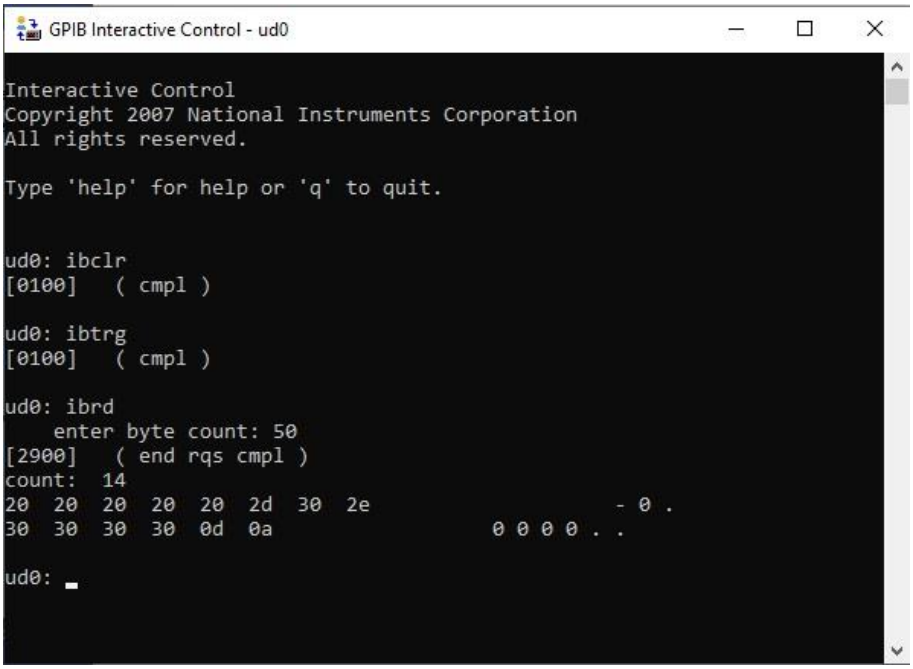


Figura 35. Comunicación GPIB con la tarjeta.

Una forma alternativa y más completa de probar la comunicación con la tarjeta es ingresar al software “Interactive Control” directamente desde el enlace “NI-GPIB-USB-HS+ GPIB0”, como se muestra en la figura 36.

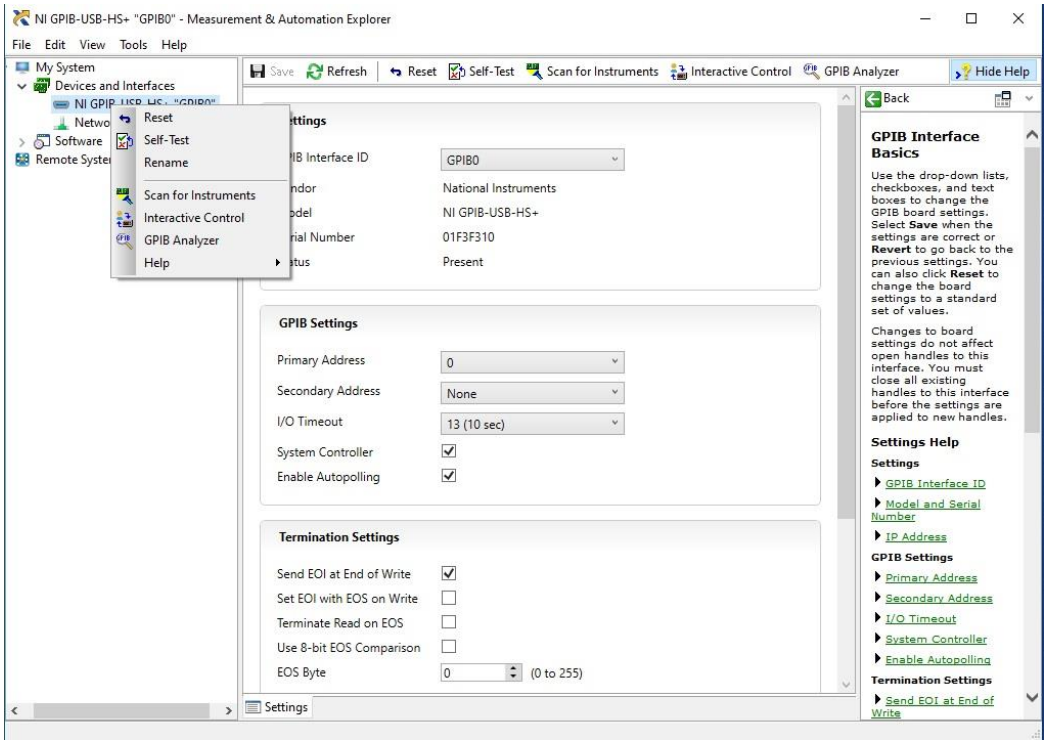


Figura 36. Interactive Control en un nivel superior.

En este momento se abrirá la pantalla de la figura 37 para ingresar la siguiente lista de comandos:

- ibdev. Obtiene el descriptor del dispositivo 0, en la dirección primaria 3, en la dirección secundaria 0, con un tiempo de espera de 10s, último byte 1 y modo de fin de cadena 0.
- ibclr. Envía la instrucción para limpiar el dispositivo.
- ibtrg. Envía la instrucción de disparo al dispositivo.
- ibrd. Envía la instrucción para la lectura de bytes de datos desde el dispositivo.

Si las operaciones fueron exitosas y libres de error, entonces el despliegue debe ser conforme a la figura 37.

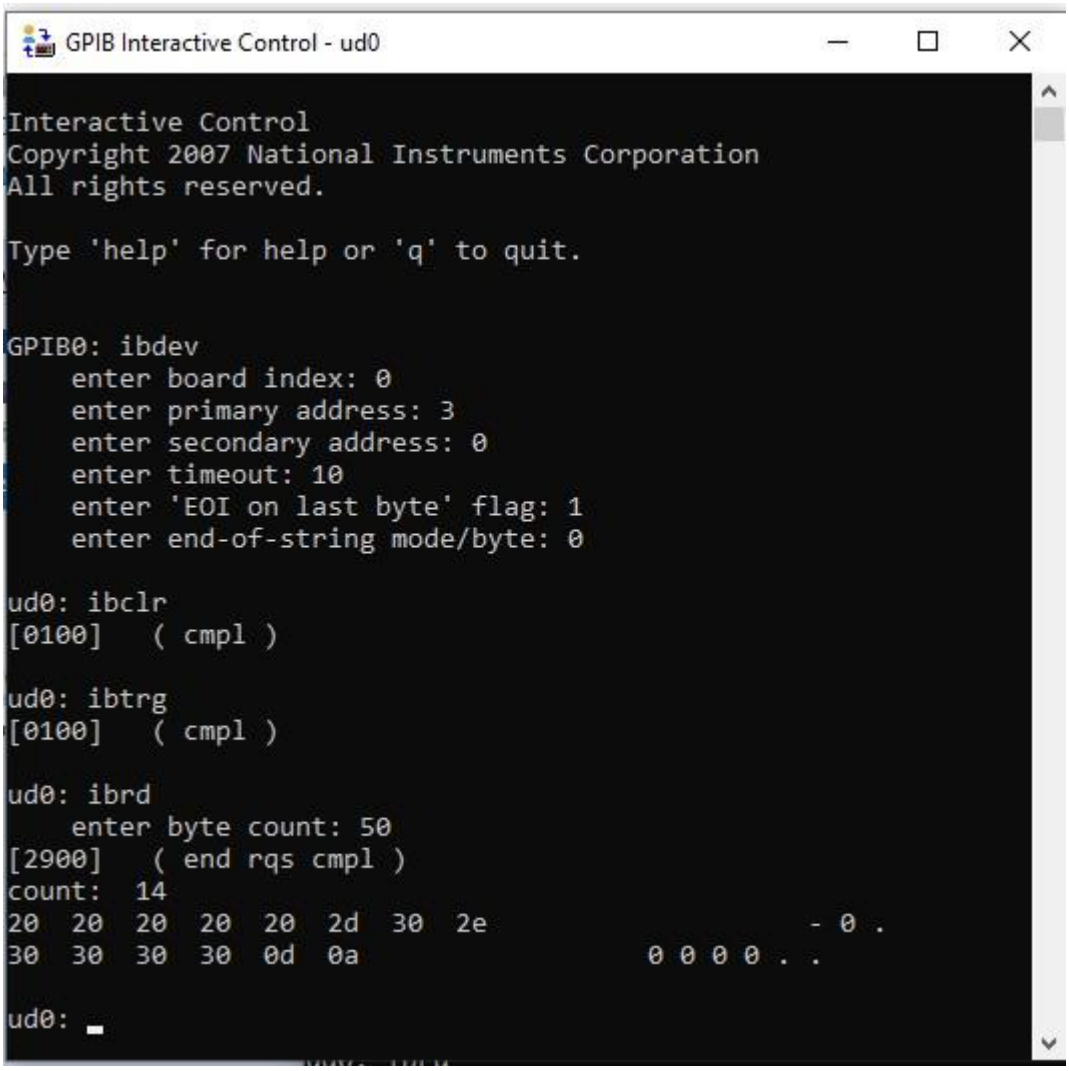


Figura 37. Comunicación GPIB con la tarjeta en un nivel superior.

1.5 Procedimiento de operación

Para operar el sistema láser de medición HP5528A en conjunto con el software “Láser”, primero se deben realizar las conexiones físicas como se muestra en la figura 38.

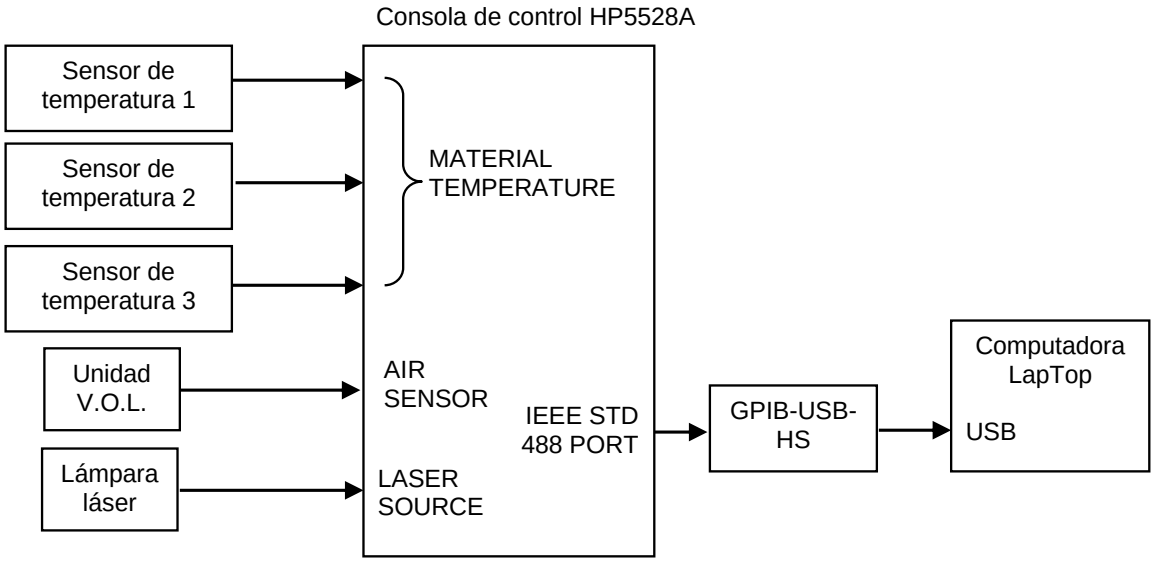


Figura 38. Esquema de conexiones para el láser HP5528A y software “Láser”.

Los conectores en la consola de control HP5528A son inconfundibles de manera que no hay posibilidad de error en las conexiones, como se muestra en la figura 39.

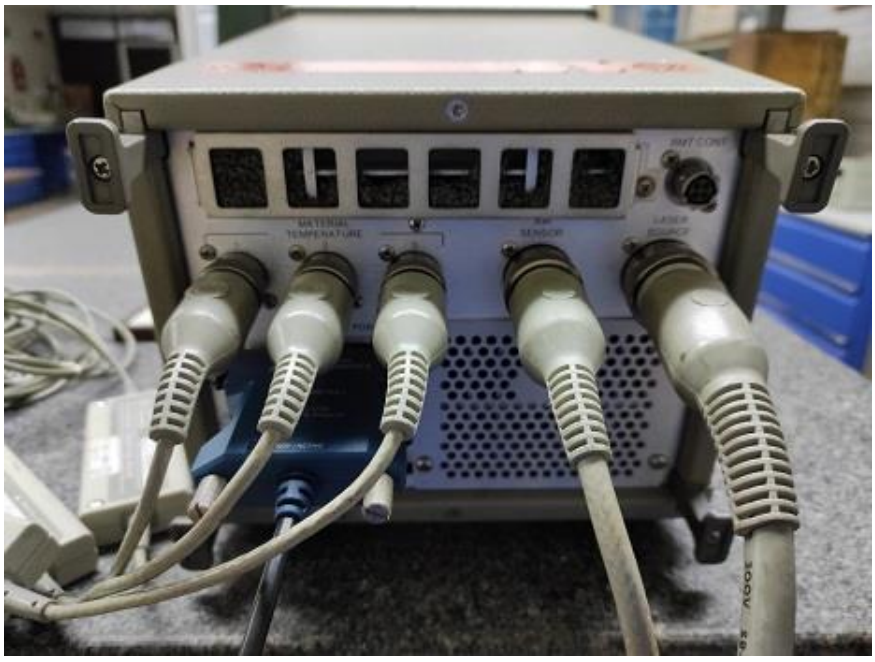


Figura 39. Vista posterior de la consola del láser HP5528A y con conexiones.

Posteriormente se debe realizar un procedimiento de alineación con la óptica asociada a la medición requerida: distancia, velocidad, ángulo, rectitud larga o rectitud corta. Para este procedimiento refiérase al manual de operación del láser HP5528A (Hewlett Packard, 1986).

Una vez realizadas las conexiones de las figuras 38 y 39 y las alineaciones requeridas para el modo de medición elegido, el sistema puede energizarse al presionar el botón “Line ON” de la consola de control HP5528A para iniciar la rutina de precalentamiento. Pasados unos segundos, el despliegue de la consola mostrará el mensaje “Laser Up” lo que indica que se pueden realizar mediciones después de presionar el botón “RESET” de la consola.

En este momento se podrá iniciar el programa “Láser” en la computadora y obtener una pantalla como la mostrada en la figura 40.

Una vez que las condiciones son las indicadas, el usuario tiene una gran diversidad de opciones para realizar mediciones, casi siempre al posicionar la óptica asociada sobre el objeto a medir. Estas mediciones se reflejarán en tiempo real en el cuadro de “Lectura” del programa “Láser”.

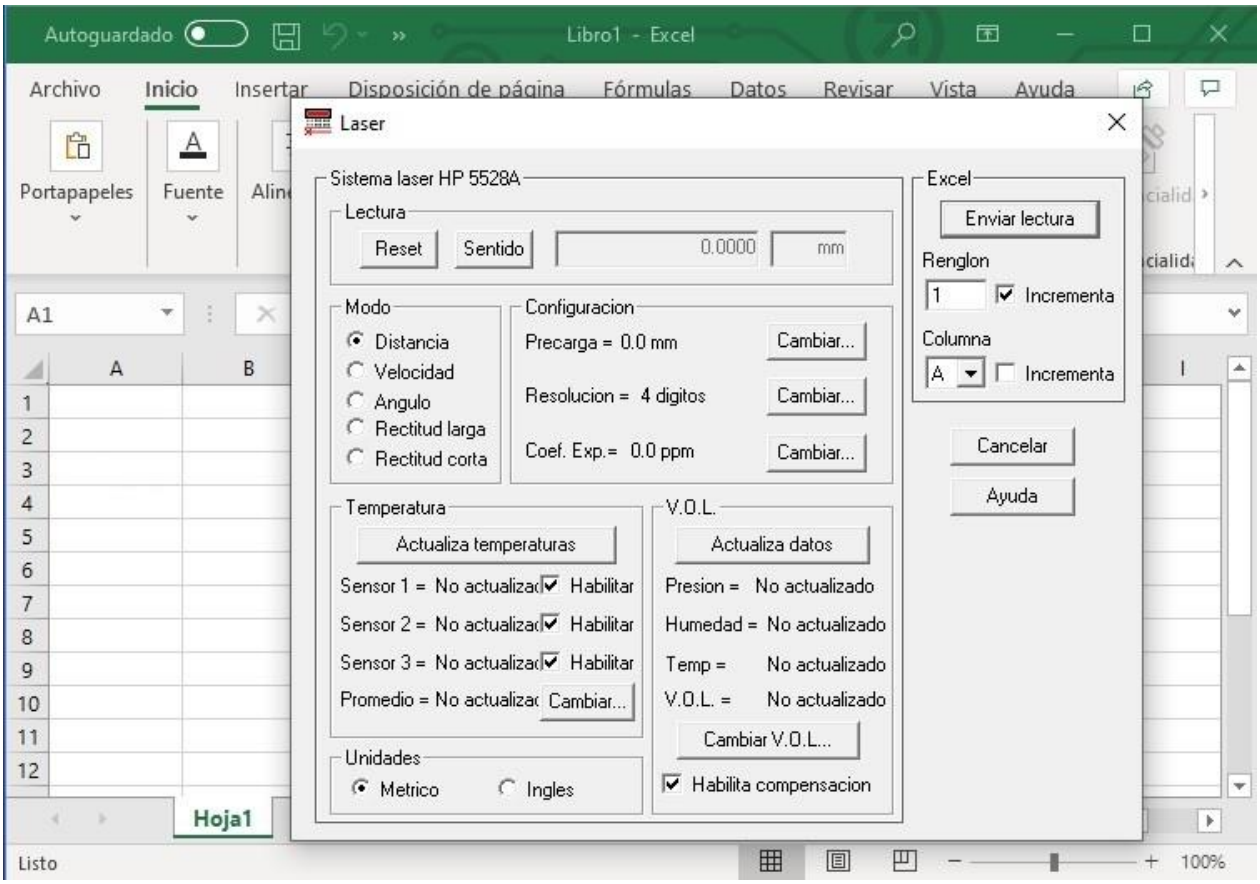


Figura 40. Láser HP5528A y software “Láser”.

Las opciones estándar del sistema láser HP5528A al realizar mediciones son proporcionadas por los siguientes siete botones:

- Reset. Se utiliza para poner a cero la lectura actual o para ignorar una condición anómala de operación reportada en la consola.
- Sentido. Se utiliza para alternar el sentido de la lectura entre positivo o negativo con relación a la posición de la óptica asociada.
- Configuración Cambiar… Son tres botones que permiten cambiar los valores de precarga, resolución y coeficiente de expansión con los que opera el algoritmo de medición en el láser HP5528A.
- Temperatura Cambiar… Se utiliza para cambiar la temperatura promedio con la que el láser HP5528A compensa las mediciones por variaciones en la temperatura del material a medir.
- Cambiar V.O.L. Se utiliza para cambiar el valor V.O.L. con el que el láser HP5528A compensa las mediciones por variaciones en la calidad del aire.

Las opciones adicionales que provee el programa “Láser” a la funcionalidad del láser HP5528A son proporcionadas por los siguientes 3 botones:

- Actualiza temperaturas. Al pulsar este botón, las temperaturas de los sensores 1 a 3 se actualizan y despliegan en los controles de cajas de texto. Adicionalmente, al seleccionar las casillas de verificación correspondientes, se consideran o no para el cálculo del promedio de temperaturas con el cual el láser HP5528A compensa sus mediciones por variaciones en la temperatura del material.
- Actualiza datos. Al pulsar este botón, los valores de presión, humedad temperatura y número V.O.L se actualizan y despliegan en los controles de cajas de texto.
- Enviar lectura. En el grupo de controles “Excel” existe una serie de elementos que transportan las mediciones del despliegue “Lectura” hacia una hoja de cálculo en Excel 365. Las casillas de verificación, “Incrementa”, en los controles “Renglón” y “Columna” permiten incrementar estos parámetros con cada pulsación del botón “Envía lectura” de modo que las lecturas anteriores no sean sobre-escritas en la misma celda de la hoja de cálculo. De esta manera, se consigue que nuevas lecturas se vayan organizando por renglones o columnas en la hoja de cálculo.

2 Resultados y discusión

En esta sección se exponen los resultados obtenidos en la elaboración del presente proyecto de desarrollo, una discusión acerca del trabajo realizado y una breve guía de fallas que se ha detectado que se pudieran presentar en la operación del sistema descrito.

2.1 Resultados

El principal resultado del proyecto es el programa “Laser” de la figura 41, que permite operar a el sistema láser de medición HP5528A en tiempo real. La figura 41 muestra en primer plano la pantalla principal del programa “Láser” y en segundo plano Excel 365. Es importante mencionar que Excel es ejecutado automáticamente por el programa “Láser” y es conveniente no abrir Excel previamente.

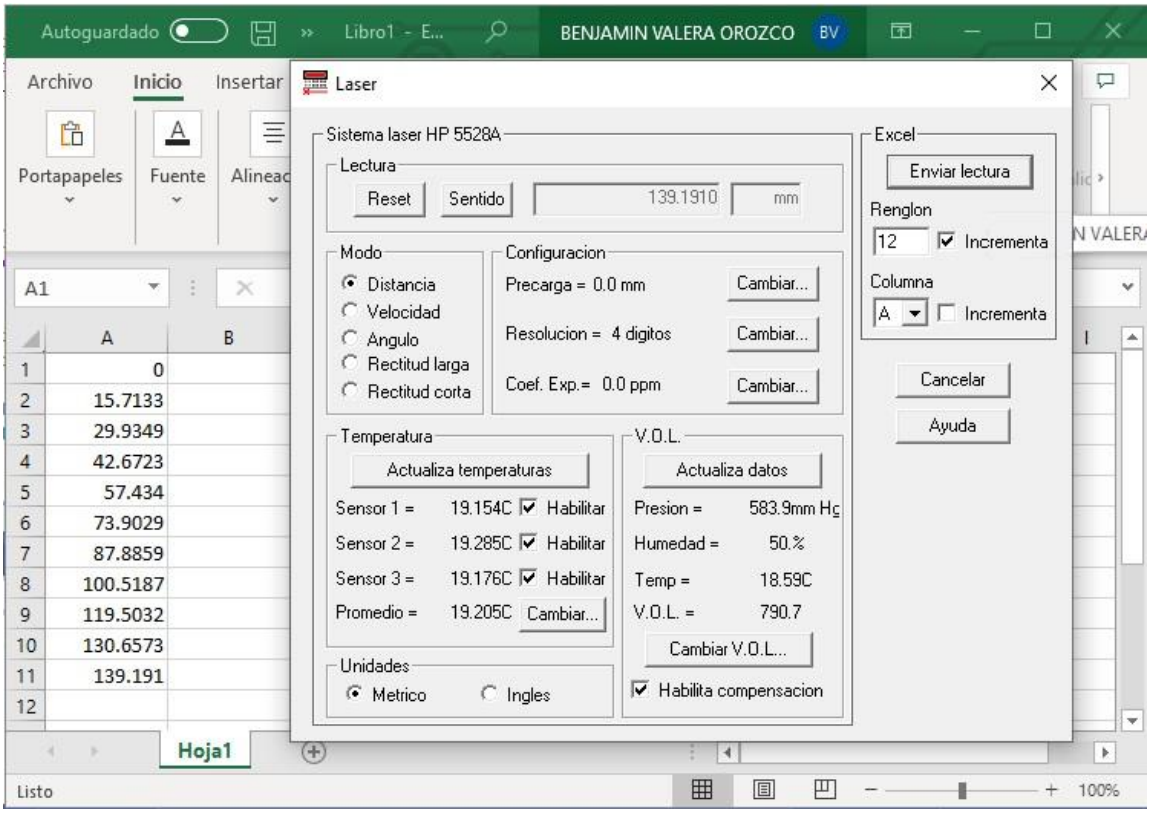


Figura 41. Programa “Láser”.

Otro resultado importante es que se logró crear infraestructura de medición que permite la participación de alumnos y académicos en la operación del láser HP5528A para la elaboración de prácticas de laboratorio en las carreras de licenciatura en ingeniería mecánica de la UNAM. La figura 42 muestra esta infraestructura desarrollada.

La figura 43 muestra un detalle del arreglo instrumental de la figura 42.

Figura 42. Infraestructura para el desarrollo de prácticas de laboratorio.

Figura 43. Infraestructura para el desarrollo de prácticas de laboratorio (detalle).

2.2 Guía para la detección de fallas

Una de las fallas que pudiera presentarse al ejecutar el programa “Láser” es que se despliegue el mensaje de error de la figura 44.

Figura 44. Mensaje de error en la ejecución del programa “Láser”.

Esta situación indica que no existe comunicación entre el programa “Láser” y la tarjeta de interfaz GPIB-USB-HS o el láser HP5528A. Para detectar la falla y corregir el problema se deben revisar las siguientes situaciones:

- Verificar la correcta conexión entre la tarjeta GPIB-USB-HS y la computadora, como se describió en la sección 1.5.
- Verificar la correcta conexión entre la tarjeta GPIB-USB-HS y el láser HP5528A, como se describió en la sección 1.5.
- Verificar que el láser HP5528A esté energizado y éste haya pasado la rutina de precalentamiento y listo para realizar mediciones al presionar su botón de “Reset”, como se describió en la sección 1.5.
- Verificar que la tarjeta GPIB-USB-HS esté correctamente identificada por el sistema operativo de la computadora. Esto se puede realizar al ejecutar el software “NI MAX” “Measurement & Automation Explorer”, como se describió en la sección 1.4.3.
- Verificar en “NI MAX” “Measurement & Automation Explorer” que la dirección primaria del dispositivo sea igual a 3. En caso contrario, cambiar la dirección para que corresponda con la 3 establecida por omisión en el programa “Láser”.

Otra falla que se pudiera presentar al ejecutar el programa “Láser” consiste en que Excel 365 no pueda ser ejecutado en segundo plano. Esto se puede deber a que Excel 365 no ha sido correctamente instalado y no esté presente en el registro de Windows. Para verificar que Excel 365 esta presente en el registro de Windows se debe ejecutar la utilería de Windows “Editor del registro” al teclear “reg” en la pestaña de búsqueda de Windows. La figura 45 muestra el “Editor del registro” cuando se ha navegado hasta las siguientes direcciónes:

HKEY_CLASSES_ROOT\Excel.Application” y
HKEY_CLASSES_ROOT\Excel.Application.16”

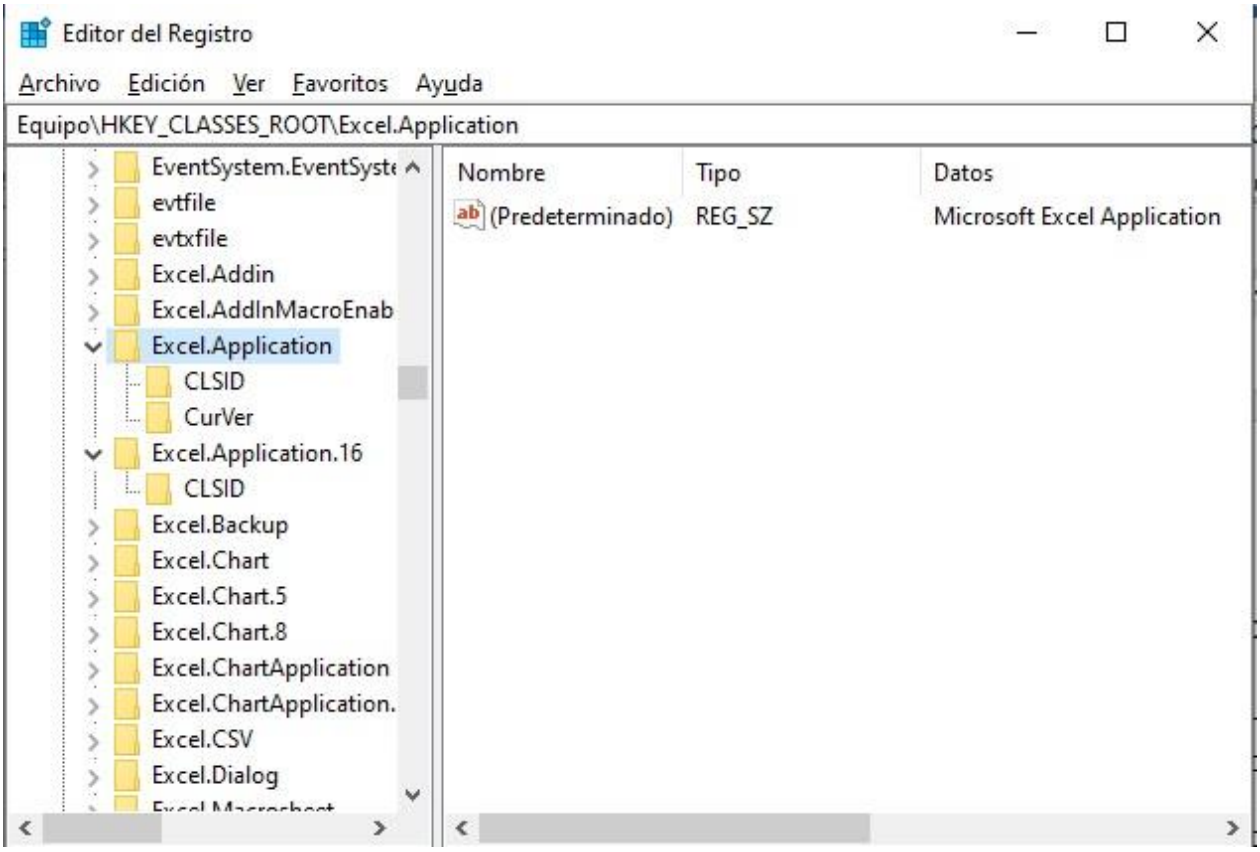


Figura 45. Registro de Windows para Excel 365.

En estas direcciones se debe verificar que los valores de los registros sean como se muestra en la tabla 6.

Ruta	**Nombre**	**Tipo**	**Datos**
HKEY_CLASSES_ROOT\ Excel.Application	(Predeterminado)	REG_SZ	Microsoft Excel Application
HKEY_CLASSES_ROOT\ Excel.Application\CLSID	(Predeterminado)	REG_SZ	00024500-0000-0000-C000-000000000046
HKEY_CLASSES_ROOT\ Excel.Application\CurVer	(Predeterminado)	REG_SZ	Excel.Application.16
HKEY_CLASSES_ROOT\ Excel.Application.16	(Predeterminado)	REG_SZ	Microsoft Excel Application
HKEY_CLASSES_ROOT\ Excel.Application.16\CLSID	(Predeterminado)	REG_SZ	00024500-0000-0000-C000-000000000046
\HKEY_CLASSES_ROOT\CLSID\ {00024500-0000-0000-C000-000000000046}	(Predeterminado)	REG_SZ	Microsoft Excel Application
HKEY_CLASSES_ROOT\CLSID\ {00024500-0000-0000-C000-000000000046}\Implemented Categories	(Predeterminado)	REG_SZ	(valor no establecido)
HKEY_CLASSES_ROOT\CLSID\ {00024500-0000-0000-C000-000000000046}\InprocHandler32	(Predeterminado)	REG_SZ	Ole32.dll
HKEY_CLASSES_ROOT\CLSID\ {00024500-0000-0000-C000-000000000046}\InprocServer32	(Predeterminado)	REG_SZ	(valor no establecido)
	Assembly	REG_SZ	Microsoft.Office.Interop.Excel, Version…

	Class	REG_SZ	Microsoft.Office.Interop.Excel.ApplicationClass…
	RuntimeVersion	REG_SZ	V2.050727
HKEY_CLASSES_ROOT\CLSID\{00024500-0000-0000-C000-000000000046}\LocalServer32	(Predeterminado)	REG_SZ	C:\Program Files\Microsoft Office\Root\Office16\EXCEL.EXE/automation
HKEY_CLASSES_ROOT\CLSID\{00024500-0000-0000-C000-000000000046}\ProgID	(Predeterminado)	REG_SZ	Excel.Application.16
HKEY_CLASSES_ROOT\CLSID\{00024500-0000-0000-C000-000000000046}\\VersionIndependentProgID	(Predeterminado)	REG_SZ	Excel.Application

Tabla 6. Valores de los registros de Windows para Excel 365.

Adicionalmente, se debe buscar la existencia del siguiente archivo en la ruta mencionada:

C:\Program Files\Microsoft Office\Root\Office16\EXCEL.EXE

Si existiera alguna discrepancia con lo mencionado, no es conveniente alterar los registros de Windows manualmente. En lugar de ello se recomienda reinstalar Office 365 hasta que se haya cumplido con lo que aquí se menciona.

2.3 Discusión y trabajo a futuro

Son dos los aspectos que se pueden presentar para la discusión en la presentación y desempeño del presente trabajo de desarrollo.

En primer lugar, la pertinencia y eficacia de adoptar el enfoque consistente en sustituir la consola del láser HP5528A por una computadora con software de interfaz con el usuario. Es cierto que un análisis superficial de tal situación indicaría que resulta caro destinar recursos adicionales para la compra de los dispositivos que implica este enfoque de sustitución. La infraestructura necesaria al menos involucra la compra de una tarjeta de interfaz GPIB a USB y una computadora. Lo cierto es que en la actualidad es extremadamente difícil encontrar a un proveedor de servicios de mantenimiento para el láser HP5528A, dado que el equipo ha sido descontinuado hace tiempo e inclusive el fabricante original, HP, cambio de nombre a Agilent. En el caso en que el láser HP5528A presentara una descompostura, sería prácticamente imposible solucionar el problema en la actualidad. Entonces el enfoque adoptado comprueba su utilidad ya que resulta más benéfico exponer una computadora y tarjeta de interfaz al desgaste que exponer a la misma consola del HP5528A a la pesada carga de trabajo que implica el uso de este instrumento de medición en la enseñanza o prestación de servicios. La prueba de que el enfoque ha resultado benéfico, es que se ha logrado mantener en operación el láser HP5528A a lo largo de casi 30 años a partir de su adquisición y del primer proyecto de automatización en el año de 1997.

En segundo lugar, la posibilidad de instalar el software desarrollado en este trabajo en otro equipo de cómputo prescindiendo de la paquetería básica de desarrollo como

Visual Studio y NI-488.2 tendría como beneficios el evitar los costos de adquirir otra licencia para Visual Studio y la disminución de la carga del software de National Instruments que estaría subutilizado. Esta posibilidad puede ser enfrentada en un futuro dado que eventualmente se requerirán cambios en la configuración de los equipos de cómputo en el Laboratorio de Metrología del ICAT UNAM.

Como trabajo a futuro planteamos las siguientes líneas de trabajo:

- Explorar la posibilidad de ejecutar el software “Láser” prescindiendo de la instalación de Visual Studio 2019 y NI-488.2. Esta posibilidad implicaría la creación de un archivo ejecutable robusto que contuviera las librerías requeridas y la instalación de la tarjeta GPIB-USB-HS a partir de sus manejadores básicos.

- Observar el desempeño del sistema desarrollado en las clases de laboratorio para detectar oportunidades de mejora o posibles fallas.

Conclusiones

Se desarrolló el programa para computadora Windows 10, “Láser”, que sustituye a la consola HP5528A como interfaz de operación con el sistema láser de medición. Se utilizaron herramientas básicas de desarrollo para crear el programa “Láser” lo cual evita los costos en la compra de licenciamientos de programas de terceros para la automatización de equipos de laboratorio. Adicionalmente, al usar herramientas básicas de desarrollo se cuenta con un control completo de los algoritmos de control y medición que pueden ser optimizados en cualquier situación.

El programa “Láser” permite la adquisición de lecturas en tiempo real en una hoja de cálculo en Excel, lo cual facilita el trabajo experimental cuando se utiliza el láser HP5528A.

El programa y automatización del sistema láser de medición HP5528A se plantean como alternativa para disminuir el desgaste de su consola de operación, ya que se prevé que el número de usuarios de este equipo se incremente ante su inminente incorporación como parte de los cursos de laboratorio de ingeniería mecánica en la UNAM.

Los resultados alcanzados aseguran el buen desempeño del equipo y la confiabilidad de la automatización implementada.

Agradecimientos

Al proyecto PAPIME PE103720 “Aplicaciones del láser en la metrología dimensional. Prácticas de calibración y medición” por el financiamiento recibido y sin el cual habría sido imposible la realización de este trabajo y la compra de los equipos y software.

Referencias

Hewlett Packard. (1986) *Láser measurement system service manual*. Hewlett Packard, USA, 419 pp.

Kruglinski DJ, Shepherd G y Wingo S. (1988) *Programming Microsoft Visual C++*. Microsoft Press, USA, 1153 pp.

National Instruments (2021) GPIB-USB-HS Dispositivo de Control de Instrumentos GPIB [Internet], National Instruments, Disponible desde <https://www.ni.com/es-mx/support/model.gpib-usb-hs.html> [Acceso 26 de junio de 2021].

Programmer Sought, (2021) C++ export data to Excel [Internet], Programmer Sought. Disponible desde <https://www.programmersought.com/article/90945649270/> [Acceso 26 de junio de 2021].

Valera O. B., Ruiz B. G. (2002) *Software para el sistema de medición láser HP5528A Informe Técnico*, Centro de Instrumentos UNAM, pp 27.

Valera O. B., Padilla O. S., Ruiz B. G. (1997) *Programa "Láser": Interfaz para PC del Sistema de Medición Láser HP5528A Informe Técnico*, Centro de Instrumentos UNAM, pp 31.

Anexo A Código fuente

CoefExp.h

```
#if !defined(AFX_COEFEXP_H__52DD5860_5382_11D6_AB8C_00A024175880__INCLUDED_)
#define AFX_COEFEXP_H__52DD5860_5382_11D6_AB8C_00A024175880__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// CoefExp.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CCoefExp dialog

class CCoefExp : public CDialog
{
// Construction
public:
	CCoefExp(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
	//{{AFX_DATA(CCoefExp)
	enum { IDD = IDD_CCOEFEXP };
	double m_dCoefExp;
	//}}AFX_DATA


// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CCoefExp)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:

	// Generated message map functions
	//{{AFX_MSG(CCoefExp)
		// NOTE: the ClassWizard will add member functions here
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif // !defined(AFX_COEFEXP_H__52DD5860_5382_11D6_AB8C_00A024175880__INCLUDED_)
```

Cprecarga.h

```
#if !defined(AFX_CPRECARGA_H__C490E200_55D6_11D6_AB8C_00A024175880__INCLUDED_)
#define AFX_CPRECARGA_H__C490E200_55D6_11D6_AB8C_00A024175880__INCLUDED_
```

```
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// Cprecarga.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CCprecarga dialog

class CCprecarga : public CDialog
{
// Construction
public:
	int su;
	int modo;
	CCprecarga(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
	//{{AFX_DATA(CCprecarga)
	enum { IDD = IDD_CPRECARGA };
	CString	m_cadena;
	double m_precarga;
	//}}AFX_DATA


// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CCprecarga)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:

	// Generated message map functions
	//{{AFX_MSG(CCprecarga)
		// NOTE: the ClassWizard will add member functions here
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif // !defined(AFX_CPRECARGA_H__C490E200_55D6_11D6_AB8C_00A024175880__INCLUDED_)
```

Ctempprom.h

```
#if !defined(AFX_CTEMPPROM_H__C490E204_55D6_11D6_AB8C_00A024175880__INCLUDED_)
#define AFX_CTEMPPROM_H__C490E204_55D6_11D6_AB8C_00A024175880__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
```

```
// Ctempprom.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CCtempprom dialog

class CCtempprom : public CDialog
{
// Construction
public:
	int su;
	CCtempprom(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
	//{{AFX_DATA(CCtempprom)
	enum { IDD = IDD_CTEMPERATURA };
	CString	m_cCadena;
	double m_dTemp;
	//}}AFX_DATA


// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CCtempprom)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:

	// Generated message map functions
	//{{AFX_MSG(CCtempprom)
		// NOTE: the ClassWizard will add member functions here
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif // !defined(AFX_CTEMPPROM_H__C490E204_55D6_11D6_AB8C_00A024175880__INCLUDED_)
```

Cvol.h

```
#if !defined(AFX_CVOL_H__C490E205_55D6_11D6_AB8C_00A024175880__INCLUDED_)
#define AFX_CVOL_H__C490E205_55D6_11D6_AB8C_00A024175880__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// Cvol.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CCvol dialog
```

```
class CCvol : public CDialog
{
// Construction
public:
	CCvol(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
	//{{AFX_DATA(CCvol)
	enum { IDD = IDD_CVOL };
	double m_dVOL;
	//}}AFX_DATA


// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CCvol)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:

	// Generated message map functions
	//{{AFX_MSG(CCvol)
		// NOTE: the ClassWizard will add member functions here
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif // !defined(AFX_CVOL_H__C490E205_55D6_11D6_AB8C_00A024175880__INCLUDED_)
```

Laser.h

```
// Laser.h : main header file for the LASER application
//

#if !defined(AFX_LASER_H__A2965A84_4B14_11D6_AB8C_00A024175880__INCLUDED_)
#define AFX_LASER_H__A2965A84_4B14_11D6_AB8C_00A024175880__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#ifndef __AFXWIN_H__
	#error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"		// main symbols

/////////////////////////////////////////////////////////////////////////////
// CLaserApp:
```

```
// See Laser.cpp for the implementation of this class
//

class CLaserApp : public CWinApp
{
public:
	CLaserApp();

// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CLaserApp)
	public:
	virtual BOOL InitInstance();
	//}}AFX_VIRTUAL

// Implementation

	//{{AFX_MSG(CLaserApp)
		// NOTE - the ClassWizard will add and remove member functions here.
		//    DO NOT EDIT what you see in these blocks of generated code !
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};


/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif // !defined(AFX_LASER_H__A2965A84_4B14_11D6_AB8C_00A024175880__INCLUDED_)
```

LaserDlg.h

```
// LaserDlg.h : header file
//

#if !defined(AFX_LASERDLG_H__A2965A86_4B14_11D6_AB8C_00A024175880__INCLUDED_)
#define AFX_LASERDLG_H__A2965A86_4B14_11D6_AB8C_00A024175880__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

/////////////////////////////////////////////////////////////////////////////
// CLaserDlg dialog

#include "excel9.h"

class CLaserDlg : public CDialog
{

private:
   _Application m_app;
   Range        m_range[5];
   _Worksheet   m_worksheet;
```

```
    Workbooks    m_workbooks;
    Worksheets   m_worksheets;

// Construction
public:
        BOOL CompensaVOL;
        double NumVol;
        double TempIn;
        double TempMe;
        double CoefExp;
        int ResM1Me;
        int ResM1In;

        int ResM2Me;
        int ResM2In;

        int ResM3Me;
        int ResM3In;

        int ResM4Me;
        int ResM4In;

        int ResM5Me;
        int ResM5In;

        double PreM1Me;
        double PreM1In;

        double PreM2Me;
        double PreM2In;

        double PreM3Me;
        double PreM3In;

        double PreM4Me;
        double PreM4In;

        double PreM5Me;
        double PreM5In;

        CString Lee();
        BOOL Escribe(CString Comando);
        BOOL Sentido;
        CLaserDlg(CWnd* pParent = NULL);  // standard constructor

// Dialog Data
        //{{AFX_DATA(CLaserDlg)
        enum { IDD = IDD_LASER_DIALOG };
        CButton       m_CAExcel;
        short  m_iRenglones;
        BOOL   m_bIncColumna;
        BOOL   m_bIncRenglon;
        int           m_iColumnas;
        CString       m_Lectura;
        CString       m_CUnidades;
        int           m_iModo;
        int           m_iSU;
        CString       m_cCoefExp;
```

```
	CString		m_cResolucion;
	CString		m_precarga;
	CString		m_cTemp1;
	CString		m_cTemp2;
	CString		m_cTemp3;
	CString		m_cTempProm;
	BOOL	m_bhtemp1;
	BOOL	m_bhtemp2;
	BOOL	m_bhtemp3;
	CString		m_cPresion;
	CString		m_cHumedad;
	CString		m_cTempAire;
	CString		m_cVOL;
	BOOL	m_HVOL;
	//}}AFX_DATA

	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CLaserDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);	// DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:
	HICON m_hIcon;

	// Generated message map functions
	//{{AFX_MSG(CLaserDlg)
	virtual BOOL OnInitDialog();
	afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
	afx_msg void OnDestroy();
	afx_msg void OnPaint();
	afx_msg HCURSOR OnQueryDragIcon();
	afx_msg void OnAexcel();
	afx_msg void OnTimer(UINT nIDEvent);
	afx_msg void OnReset();
	afx_msg void OnSentido();
	afx_msg void OnDistancia();
	afx_msg void OnMetrico();
	afx_msg void OnIngles();
	afx_msg void OnVelocidad();
	afx_msg void OnAngulo();
	afx_msg void OnRectitudlarga();
	afx_msg void OnRectitudcorta();
	afx_msg void OnCcoefexp();
	afx_msg void OnCresolucion();
	afx_msg void OnCprecarga();
	afx_msg void OnActtemp();
	afx_msg void OnCtemp();
	afx_msg void OnHtemp();
	afx_msg void OnActvol();
	afx_msg void OnCvol();
	afx_msg void OnHvol();
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
```

```
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif // !defined(AFX_LASERDLG_H__A2965A86_4B14_11D6_AB8C_00A024175880__INCLUDED_)
```

Resolucion.h

```
#if !defined(AFX_RESOLUCION_H__E1D3BBA0_53A4_11D6_AB8C_00A024175880__INCLUDED_)
#define AFX_RESOLUCION_H__E1D3BBA0_53A4_11D6_AB8C_00A024175880__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// Resolucion.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CResolucion dialog

class CResolucion : public CDialog
{
// Construction
public:
	int ResRange;
	CResolucion(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
	//{{AFX_DATA(CResolucion)
	enum { IDD = IDD_RESOLUCION };
	BOOL   m_cSmooth;
	int           m_iSpin;
	//}}AFX_DATA


// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CResolucion)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:

	// Generated message map functions
	//{{AFX_MSG(CResolucion)
	virtual BOOL OnInitDialog();
	afx_msg void OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar);
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif // !defined(AFX_RESOLUCION_H__E1D3BBA0_53A4_11D6_AB8C_00A024175880__INCLUDED_)
```

CoefExp.cpp

```
// CoefExp.cpp : implementation file
//

#include "stdafx.h"
#include "Laser.h"
#include "CoefExp.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CCoefExp dialog


CCoefExp::CCoefExp(CWnd* pParent /*=NULL*/)
	: CDialog(CCoefExp::IDD, pParent)
{
	//{{AFX_DATA_INIT(CCoefExp)
	m_dCoefExp = 0.0;
	//}}AFX_DATA_INIT
}


void CCoefExp::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CCoefExp)
	DDX_Text(pDX, IDC_DCOEFEXP, m_dCoefExp);
	DDV_MinMaxDouble(pDX, m_dCoefExp, -720., 720.);
	//}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CCoefExp, CDialog)
	//{{AFX_MSG_MAP(CCoefExp)
		// NOTE: the ClassWizard will add message map macros here
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CCoefExp message handlers
```

Cprecarga.cpp

```
// Cprecarga.cpp : implementation file
//

#include "stdafx.h"
#include "Laser.h"
#include "Cprecarga.h"
```

```
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CCprecarga dialog


CCprecarga::CCprecarga(CWnd* pParent /*=NULL*/)
	: CDialog(CCprecarga::IDD, pParent)
{
	//{{AFX_DATA_INIT(CCprecarga)
	m_cadena = _T("");
	m_precarga = 0.0;
	//}}AFX_DATA_INIT
}


void CCprecarga::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CCprecarga)
	DDX_Text(pDX, IDC_CADENA, m_cadena);
	DDX_Text(pDX, IDC_PRECARGA, m_precarga);
//	DDV_MinMaxDouble(pDX, m_precarga, -50000., 50000.);
	//}}AFX_DATA_MAP
	switch(su)
	{
		case 0:
			switch(modo)
			{
				case 0:
					DDV_MinMaxDouble(pDX, m_precarga, -50000., 50000.);
					break;
				case 1:
					DDV_MinMaxDouble(pDX, m_precarga, -20000., 20000.);
					break;
				case 2:
					DDV_MinMaxDouble(pDX, m_precarga, 0., 2.);
					break;
				case 3:
					DDV_MinMaxDouble(pDX, m_precarga, 0., 3.);
					break;
				case 4:
					DDV_MinMaxDouble(pDX, m_precarga, 0., 3.);
					break;
			}
			break;
		case 1:
			switch(modo)
			{
				case 0:
					DDV_MinMaxDouble(pDX, m_precarga, -2000., 2000.);
					break;
				case 1:
					DDV_MinMaxDouble(pDX, m_precarga, -720., 720.);
```

```
                        break;
                case 2:
                        DDV_MinMaxDouble(pDX, m_precarga, 0., 2.);
                        break;
                case 3:
                        DDV_MinMaxDouble(pDX, m_precarga, 0., 3.);
                        break;
                case 4:
                        DDV_MinMaxDouble(pDX, m_precarga, 0., 3.);
                        break;
            }
            break;
    }
}


BEGIN_MESSAGE_MAP(CCprecarga, CDialog)
    //{{AFX_MSG_MAP(CCprecarga)
        // NOTE: the ClassWizard will add message map macros here
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CCprecarga message handlers
```

Ctempprom.cpp

```
// Ctempprom.cpp : implementation file
//

#include "stdafx.h"
#include "Laser.h"
#include "Ctempprom.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CCtempprom dialog


CCtempprom::CCtempprom(CWnd* pParent /*=NULL*/)
    : CDialog(CCtempprom::IDD, pParent)
{
    //{{AFX_DATA_INIT(CCtempprom)
    m_cCadena = _T("");
    m_dTemp = 0.0;
    //}}AFX_DATA_INIT
}


void CCtempprom::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
```

```
	//{{AFX_DATA_MAP(CCtempprom)
	DDX_Text(pDX, IDC_TEMPPROM, m_cCadena);
	DDX_Text(pDX, IDC_NTEMPPROM, m_dTemp);
//	DDV_MinMaxDouble(pDX, m_dTemp, -100., 100.);
	//}}AFX_DATA_MAP
	switch(su)
	{
	case 0:
		DDV_MinMaxDouble(pDX, m_dTemp, 0., 50.);
		break;
	case 1:
		DDV_MinMaxDouble(pDX, m_dTemp, 32., 122.);
		break;
	}
}


BEGIN_MESSAGE_MAP(CCtempprom, CDialog)
	//{{AFX_MSG_MAP(CCtempprom)
		// NOTE: the ClassWizard will add message map macros here
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CCtempprom message handlers
```

Cvol.cpp

```
// Cvol.cpp : implementation file
//

#include "stdafx.h"
#include "Laser.h"
#include "Cvol.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CCvol dialog


CCvol::CCvol(CWnd* pParent /*=NULL*/)
	: CDialog(CCvol::IDD, pParent)
{
	//{{AFX_DATA_INIT(CCvol)
	m_dVOL = 0.0;
	//}}AFX_DATA_INIT
}


void CCvol::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
```

```
	//{{AFX_DATA_MAP(CCvol)
	DDX_Text(pDX, IDC_NVOL, m_dVOL);
	DDV_MinMaxDouble(pDX, m_dVOL, 0., 1000.);
	//}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CCvol, CDialog)
	//{{AFX_MSG_MAP(CCvol)
		// NOTE: the ClassWizard will add message map macros here
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CCvol message handlers
```

Laser.cpp

```
// Laser.cpp : Defines the class behaviors for the application.
//

#include <afxdisp.h>
#include "stdafx.h"
#include "Laser.h"
#include "LaserDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CLaserApp

BEGIN_MESSAGE_MAP(CLaserApp, CWinApp)
	//{{AFX_MSG_MAP(CLaserApp)
		// NOTE - the ClassWizard will add and remove mapping macros here.
		//    DO NOT EDIT what you see in these blocks of generated code!
	//}}AFX_MSG
	ON_COMMAND(ID_HELP, CWinApp::OnHelp)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CLaserApp construction

CLaserApp::CLaserApp()
{
	// TODO: add construction code here,
	// Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CLaserApp object

CLaserApp theApp;
```

```
/////////////////////////////////////////////////////////////////////////////
// CLaserApp initialization

BOOL CLaserApp::InitInstance()
{
	if(!AfxOleInit())
	{
		AfxMessageBox("No se puede iniciar la COM dll");
		return FALSE;
	}

	AfxEnableControlContainer();

	// Standard initialization
	// If you are not using these features and wish to reduce the size
	//  of your final executable, you should remove from the following
	//  the specific initialization routines you do not need.

#ifdef _AFXDLL
	Enable3dControls();			// Call this when using MFC in a shared DLL
#else
	Enable3dControlsStatic();	// Call this when linking to MFC statically
#endif

	CLaserDlg dlg;
	m_pMainWnd = &dlg;
	int nResponse = dlg.DoModal();
	if (nResponse == IDOK)
	{
		// TODO: Place code here to handle when the dialog is
		//  dismissed with OK
	}
	else if (nResponse == IDCANCEL)
	{
		// TODO: Place code here to handle when the dialog is
		//  dismissed with Cancel
	}

	// Since the dialog has been closed, return FALSE so that we exit the
	//  application, rather than start the application's message pump.
	return FALSE;
}
```

LaserDlg.cpp

```
// LaserDlg.cpp : implementation file
//

#include "stdafx.h"
#include "Laser.h"
#include "LaserDlg.h"
#include "coefexp.h"
#include "resolucion.h"
#include "cprecarga.h"
#include "ctempprom.h"
#include "cvol.h"
```

```
#include "ni4882.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

int ud;

void gpiberr(char *);        // ADDED- function prototype for gpiberr.

/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
	CAboutDlg();

// Dialog Data
	//{{AFX_DATA(CAboutDlg)
	enum { IDD = IDD_ABOUTBOX };
	//}}AFX_DATA

	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CAboutDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:
	//{{AFX_MSG(CAboutDlg)
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
	//{{AFX_DATA_INIT(CAboutDlg)
	//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
		// No message handlers
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

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

```
// CLaserDlg dialog

CLaserDlg::CLaserDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CLaserDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CLaserDlg)
	m_iRenglones = 1;
	m_bIncColumna = FALSE;
	m_bIncRenglon = TRUE;
	m_iColumnas = 0;
	m_Lectura = _T("");
	m_CUnidades = "mm";
	m_iModo = 0;
	m_iSU = 0;
	m_cCoefExp = "0.0 ppm";
	m_cResolucion = "4 digitos";
	m_precarga = "0.0 mm";
	m_cTemp1 = "No actualizado";
	m_cTemp2 = "No actualizado";
	m_cTemp3 = "No actualizado";
	m_cTempProm = "No actualizado";
	m_bhtemp1 = TRUE;
	m_bhtemp2 = TRUE;
	m_bhtemp3 = TRUE;
	m_cPresion = "No actualizado";
	m_cHumedad = "No actualizado";
	m_cTempAire = "No actualizado";
	m_cVOL = "No actualizado";
	m_HVOL = TRUE;
	//}}AFX_DATA_INIT
	// Note that LoadIcon does not require a subsequent DestroyIcon in Win32
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
	Sentido=TRUE;

	CoefExp=0.0;

	ResM1Me=4;
	ResM1In=5;

	ResM2Me=0;
	ResM2In=1;

	ResM3Me=0;
	ResM3In=0;

	ResM4Me=3;
	ResM4In=4;

	ResM5Me=4;
	ResM5In=5;

	PreM1Me=0.0;
	PreM1In=0.0;

	PreM2Me=0.0;
	PreM2In=0.0;

	PreM3Me=1.0;
```

```
	PreM3In=1.0;

	PreM4Me=1.0;
	PreM4In=1.0;

	PreM5Me=1.0;
	PreM5In=1.0;

	TempMe=20.00;
	TempIn=68.00;

	NumVol=728.8;

	CompensaVOL=TRUE;
}

void CLaserDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CLaserDlg)
	DDX_Control(pDX, IDAEXCEL, m_CAExcel);
	DDX_Text(pDX, IDC_RENGLONES, m_iRenglones);
	DDV_MinMaxInt(pDX, m_iRenglones, 1, 100);
	DDX_Check(pDX, IDC_INCCOLUMNA, m_bIncColumna);
	DDX_Check(pDX, IDC_INCRENGLON, m_bIncRenglon);
	DDX_CBIndex(pDX, IDC_COLUMNAS, m_iColumnas);
	DDX_Text(pDX, IDC_LECTURA, m_Lectura);
	DDX_Text(pDX, IDC_UNIDADES, m_CUnidades);
	DDX_Radio(pDX, IDC_DISTANCIA, m_iModo);
	DDX_Radio(pDX, IDC_METRICO, m_iSU);
	DDX_Text(pDX, IDC_COEFEXP, m_cCoefExp);
	DDX_Text(pDX, IDC_RESOLUCION, m_cResolucion);
	DDX_Text(pDX, IDC_PRECARGA, m_precarga);
	DDX_Text(pDX, IDC_TEMP1, m_cTemp1);
	DDX_Text(pDX, IDC_TEMP2, m_cTemp2);
	DDX_Text(pDX, IDC_TEMP3, m_cTemp3);
	DDX_Text(pDX, IDC_TEMPPROM, m_cTempProm);
	DDX_Check(pDX, IDC_HTEMP1, m_bhtemp1);
	DDX_Check(pDX, IDC_HTEMP2, m_bhtemp2);
	DDX_Check(pDX, IDC_HTEMP3, m_bhtemp3);
	DDX_Text(pDX, IDC_PRESION, m_cPresion);
	DDX_Text(pDX, IDC_HUMEDAD, m_cHumedad);
	DDX_Text(pDX, IDC_TEMPERATURAAIRE, m_cTempAire);
	DDX_Text(pDX, IDC_VOL, m_cVOL);
	DDX_Check(pDX, IDC_HVOL, m_HVOL);
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CLaserDlg, CDialog)
	//{{AFX_MSG_MAP(CLaserDlg)
	ON_WM_SYSCOMMAND()
	ON_WM_DESTROY()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDAEXCEL, OnAexcel)
	ON_WM_TIMER()
	ON_BN_CLICKED(IDC_RESET, OnReset)
	ON_BN_CLICKED(IDC_SENTIDO, OnSentido)
```

```
	ON_BN_CLICKED(IDC_DISTANCIA, OnDistancia)
	ON_BN_CLICKED(IDC_METRICO, OnMetrico)
	ON_BN_CLICKED(IDC_INGLES, OnIngles)
	ON_BN_CLICKED(IDC_VELOCIDAD, OnVelocidad)
	ON_BN_CLICKED(IDC_ANGULO, OnAngulo)
	ON_BN_CLICKED(IDC_RECTITUDLARGA, OnRectitudlarga)
	ON_BN_CLICKED(IDC_RECTITUDCORTA, OnRectitudcorta)
	ON_BN_CLICKED(IDC_CCOEFEXP, OnCcoefexp)
	ON_BN_CLICKED(IDC_CRESOLUCION, OnCresolucion)
	ON_BN_CLICKED(IDC_CPRECARGA, OnCprecarga)
	ON_BN_CLICKED(IDC_ACTTEMP, OnActtemp)
	ON_BN_CLICKED(IDC_CTEMP, OnCtemp)
	ON_BN_CLICKED(IDC_HTEMP1, OnHtemp)
	ON_BN_CLICKED(IDC_ACTVOL, OnActvol)
	ON_BN_CLICKED(IDC_CVOL, OnCvol)
	ON_BN_CLICKED(IDC_HTEMP2, OnHtemp)
	ON_BN_CLICKED(IDC_HTEMP3, OnHtemp)
	ON_BN_CLICKED(IDC_HVOL, OnHvol)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CLaserDlg message handlers

BOOL CLaserDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Add "About..." menu item to system menu.

	// IDM_ABOUTBOX must be in the system command range.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
		}
	}

	// Set the icon for this dialog.  The framework does this automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon

	// TODO: Add extra initialization here

	//Inicia GPIB
	ud=ibdev(0, 3, 0, T10s, 1, 0);
	if(ud<0)
	{
		gpiberr("ibdev Error");
```

```
            m_Lectura="No existe manejador de GPIB";
//          m_CAExcel.EnableWindow(FALSE);
            UpdateData(FALSE);
//          return FALSE;
      }
      else
      {
            ibclr (ud);
            if (ibsta & ERR)
            {
                  gpiberr("ibclr Error");
                  m_Lectura="Laser no responde";
//                m_CAExcel.EnableWindow(FALSE);
                  UpdateData(FALSE);
//                return FALSE;
            }
            else Escribe("UM\x00d\x00a\0");
      }

      // si Excel ya se está ejecutando, asociarnos a él, en caso contrario,
      //  iniciarlo

   LPDISPATCH pDisp;
   LPUNKNOWN pUnk;
   CLSID clsid;
   TRACE("Entrando en CControlComView::OnExcelCargar\n");
   BeginWaitCursor();
   ::CLSIDFromProgID(L"Excel.Application.16", &clsid); // del Registro
   if(::GetActiveObject(clsid, NULL, &pUnk) == S_OK) {
      VERIFY(pUnk->QueryInterface(IID_IDispatch,
            (void**) &pDisp) == S_OK);
      m_app.AttachDispatch(pDisp);
      pUnk->Release();
      TRACE(" asociación finalizada\n");
        }

   else {
         if(!m_app.CreateDispatch("Excel.Application.16")) {
                  AfxMessageBox("Programa Excel 16 no encontrado");
         }
       TRACE(" creación finalizada\n");
   }
   EndWaitCursor();

// Ejecuta excel
   LPDISPATCH pWorkbooks;

   CWnd* pWnd = CWnd::FindWindow("XLMAIN", NULL);
   if (pWnd != NULL) {
     TRACE("Ventana de Excel encontrada\n");
     pWnd->ShowWindow(SW_SHOWNORMAL);
     pWnd->UpdateWindow();
     pWnd->BringWindowToTop();
   }

   m_app.SetSheetsInNewWorkbook(1);

   VERIFY(pWorkbooks = m_app.GetWorkbooks());
```

```
   m_workbooks.AttachDispatch(pWorkbooks);

   LPDISPATCH pWorkbook = NULL;
   if (m_workbooks.GetCount() == 0) {
      // Add devuelve un puntero a libro de trabajo (Workbook), pero
      //  no tenemos una clase Workbook
      pWorkbook = m_workbooks.Add(); // Guarda el puntero para
                                     //  una versión posterior
   }

   LPDISPATCH pWorksheets = m_app.GetWorksheets();
   ASSERT(pWorksheets != NULL);
   m_worksheets.AttachDispatch(pWorksheets);
   LPDISPATCH pWorksheet = m_worksheets.GetItem(COleVariant((short) 1));

   m_worksheet.AttachDispatch(pWorksheet);
   m_worksheet.Select();

    // liberación

   if (pWorkbook != NULL) {
         pWorkbook->Release();
   }

//     SetTimer(1, 200, NULL);

       return TRUE;  // return TRUE  unless you set the focus to a control
}

void CLaserDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
       if ((nID & 0xFFF0) == IDM_ABOUTBOX)
       {
              CAboutDlg dlgAbout;
              dlgAbout.DoModal();
       }
       else
       {
              CDialog::OnSysCommand(nID, lParam);
       }
}

void CLaserDlg::OnDestroy()
{
       WinHelp(0L, HELP_QUIT);
       KillTimer(1);
       CDialog::OnDestroy();
}

// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.

void CLaserDlg::OnPaint()
{
       if (IsIconic())
       {
              CPaintDC dc(this); // device context for painting
```

```
		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialog::OnPaint();
	}
}

// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CLaserDlg::OnQueryDragIcon()
{
	return (HCURSOR) m_hIcon;
}

void CLaserDlg::OnAexcel()
{

	UpdateData(TRUE);

    char cadenaA[50];
	char cadenaaux[50]="ABCDEFGHIJKLMNOPQRSTUVWXYZ";

    LPDISPATCH pRange;

    wsprintf(cadenaA, "%c%d", cadenaaux[m_iColumnas], m_iRenglones);

    VERIFY(pRange = m_worksheet.GetRange(COleVariant(cadenaA)));
    m_range[0].AttachDispatch(pRange);

    m_range[0].SetValue(COleVariant(m_Lectura));

    COleVariant vaResult0 = m_range[0].GetValue();
    if (vaResult0.vt == VT_BSTR)
    {
	CString str = vaResult0.bstrVal;
	TRACE("vaResult0 = %s\n", (const char*) str);
	}

	if (m_bIncRenglon & (m_iRenglones<100)) m_iRenglones++;
	if (m_bIncColumna & (m_iColumnas<25)) m_iColumnas++;

	UpdateData(FALSE);

}
```

```
// ADDED- This function will alert you that a NI-488 function failed
//        by printing an error message.  The status variable IBSTA
//        will also be printed in hexadecimal along with the
//        mnemonic meaning of the bit position. The status variable
//        IBERR will be printed in decimal along with the mnemonic
//        meaning of the decimal value. The status variable IBCNTL
//        will be printed in decimal.
//
//        The NI-488 function IBONL is called to disable the
//        hardware and software.
void gpiberr(char *msg)
{
   char str[100];
   char tempbuf[20];

   // Start the Message with the failing GPIB call.
   sprintf(str, msg);

   // Add ibsta information to the Message String.
   strcat(str, "  ibsta = &H");
   itoa(ibsta, tempbuf, 16);
   strcat(str, tempbuf);
   strcat(str, " <");
   if (ibsta & ERR )  strcat(str, " ERR");
   if (ibsta & TIMO)  strcat(str, " TIMO");
   if (ibsta & END )  strcat(str, " END");
   if (ibsta & SRQI)  strcat(str, " SRQI");
   if (ibsta & RQS )  strcat(str, " RQS");
   if (ibsta & CMPL)  strcat(str, " CMPL");
   if (ibsta & LOK )  strcat(str, " LOK");
   if (ibsta & REM )  strcat(str, " REM");
   if (ibsta & CIC )  strcat(str, " CIC");
   if (ibsta & ATN )  strcat(str, " ATN");
   if (ibsta & TACS)  strcat(str, " TACS");
   if (ibsta & LACS)  strcat(str, " LACS");
   if (ibsta & DTAS)  strcat(str, " DTAS");
   if (ibsta & DCAS)  strcat(str, " DCAS");
   strcat(str, " >");

   // Add iberr information to the Message String.
   strcat(str, "\niberr = ");
   itoa(iberr, tempbuf, 10);
   strcat(str, tempbuf);
   if (iberr == EDVR) strcat(str, " EDVR <DOS Error>");
   if (iberr == ECIC) strcat(str, " ECIC <Not CIC>");
   if (iberr == ENOL) strcat(str, " ENOL <No Listener>");
   if (iberr == EADR) strcat(str, " EADR <Address error>");
   if (iberr == EARG) strcat(str, " EARG <Invalid argument>");
   if (iberr == ESAC) strcat(str, " ESAC <Not Sys Ctrlr>");
   if (iberr == EABO) strcat(str, " EABO <Op. aborted>");
   if (iberr == ENEB) strcat(str, " ENEB <No GPIB board>");
   if (iberr == EOIP) strcat(str, " EOIP <Async I/O in prg>");
   if (iberr == ECAP) strcat(str, " ECAP <No capability>");
   if (iberr == EFSO) strcat(str, " EFSO <File sys. error>");
   if (iberr == EBUS) strcat(str, " EBUS <Command error>");
   if (iberr == ESTB) strcat(str, " ESTB <Status byte lost>");
   if (iberr == ESRQ) strcat(str, " ESRQ <SRQ stuck on>");
   if (iberr == ETAB) strcat(str, " ETAB <Table Overflow>");
```

```
   // Add ibcntl information to the Message String.
   strcat(str, "\nibcntl = ");
   ultoa(ibcntl, tempbuf, 16);
   strcat(str, tempbuf);
   AfxMessageBox(str, MB_ICONSTOP, 0);

   // Take the device offline
   ibonl(ud, 0);
}

void CLaserDlg::OnTimer(UINT nIDEvent)
{
       UpdateData(TRUE);

       m_Lectura=Lee();

       UpdateData(FALSE);

       CDialog::OnTimer(nIDEvent);
}

void CLaserDlg::OnReset()
{
       Escribe("RS\x00d\x00a\0");
}

void CLaserDlg::OnSentido()
{
       if(Sentido)
       {
              if(Escribe("D2\x00d\x00a\0")) Sentido=FALSE;
       }
       else
       {
              if(Escribe("D1\x00d\x00a\0")) Sentido=TRUE;
       }
}

BOOL CLaserDlg::Escribe(CString Comando)
{
       long cuenta;
       char cadena[20];
//     CString str;

       cuenta=Comando.GetLength();
       strcpy(cadena, Comando);

//     str.Format("Longitud de la cadena = %d", cuenta);
//     AfxMessageBox(Comando+" "+str, MB_ICONSTOP, 0);

       ibwrt (ud, cadena, cuenta);
    if (ibsta & ERR)
       {
              gpiberr("ibwrt Error");
              return FALSE;
       }
       return TRUE;
```

```
}

CString CLaserDlg::Lee()
{
	char ReadBuffer[20];
	CString Resultado;

    ibtrg (ud);
    if (ibsta & ERR)
    {
        gpiberr("ibtrg Error");
        Resultado="Laser no responde";
    }
       else
       {
              ibrd (ud, ReadBuffer, 20L);
              if (ibsta & ERR)
              {
                     gpiberr("ibrd Error");
                     Resultado="Laser no responde";
              }
              else
              {
                     ReadBuffer[ibcntl-2]='\0';
                     Resultado=ReadBuffer;
              }
       }
       return Resultado;
}

void CLaserDlg::OnMetrico()
{
       UpdateData(TRUE);
       Escribe("UM\x00d\x00a\0");
       switch(m_iModo)
       {
              case 0:
                     m_CUnidades="mm";
                     m_cResolucion.Format("%d digitos", ResM1Me);
                     m_precarga.Format("%11.5f mm", PreM1Me);
                     break;
              case 1:
                     m_CUnidades="mm/min";
                     m_cResolucion.Format("%d digitos", ResM2Me);
                     m_precarga.Format("%11.1f mm/min", PreM2Me);
                     break;
              case 2:
                     m_CUnidades="arc-sec";
                     m_cResolucion.Format("%d digitos", ResM3Me);
                     m_precarga.Format("%11.1f arc-sec", PreM3Me);
                     break;
              case 3:
                     m_CUnidades="mm";
                     m_cResolucion.Format("%d digitos", ResM4Me);
                     m_precarga.Format("%11.4f mm", PreM4Me);
                     break;
              case 4:
                     m_CUnidades="mm";
```

```
                    m_cResolucion.Format("%d digitos", ResM5Me);
                    m_precarga.Format("%11.5f mm", PreM5Me);
                    break;
        }
        UpdateData(FALSE);
}

void CLaserDlg::OnIngles()
{
        UpdateData(TRUE);
        Escribe("UE\x00d\x00a\0");
        switch(m_iModo)
        {
                case 0:
                        m_CUnidades="in";
                        m_cResolucion.Format("%d digitos", ResM1In);
                        m_precarga.Format("%11.6f in", PreM1In);
                        break;
                case 1:
                        m_CUnidades="in/min";
                        m_cResolucion.Format("%d digitos", ResM2In);
                        m_precarga.Format("%11.2f in/min", PreM2In);
                        break;
                case 2:
                        m_CUnidades="arc-sec";
                        m_cResolucion.Format("%d digitos", ResM3In);
                        m_precarga.Format("%11.1f arc-sec", PreM3In);
                        break;
                case 3:
                        m_CUnidades="in";
                        m_cResolucion.Format("%d digitos", ResM4In);
                        m_precarga.Format("%11.5f in", PreM4In);
                        break;
                case 4:
                        m_CUnidades="in";
                        m_cResolucion.Format("%d digitos", ResM5In);
                        m_precarga.Format("%11.6f in", PreM5In);
                        break;
        }
        UpdateData(FALSE);
}

void CLaserDlg::OnDistancia()
{
        UpdateData(TRUE);
        Escribe("M1\x00d\x00a\0");
        switch(m_iSU)
        {
                case 0:
                        m_CUnidades="mm";
                        m_cResolucion.Format("%d digitos", ResM1Me);
                        m_precarga.Format("%11.5f mm", PreM1Me);
                        break;
                case 1:
                        m_CUnidades="in";
                        m_cResolucion.Format("%d digitos", ResM1In);
                        m_precarga.Format("%11.6f in", PreM1In);
                        break;
```

```
	}
	UpdateData(FALSE);
}


void CLaserDlg::OnVelocidad()
{
	UpdateData(TRUE);
	Escribe("M2\x00d\x00a\0");
	switch(m_iSU)
	{
		case 0:
			m_CUnidades="mm/min";
			m_cResolucion.Format("%d digitos", ResM2Me);
			m_precarga.Format("%11.1f mm/min", PreM2Me);
			break;
		case 1:
			m_CUnidades="in/min";
			m_cResolucion.Format("%d digitos", ResM2In);
			m_precarga.Format("%11.2f in/min", PreM2In);
			break;
	}
	UpdateData(FALSE);
}

void CLaserDlg::OnAngulo()
{
	UpdateData(TRUE);
	Escribe("M3\x00d\x00a\0");
	switch(m_iSU)
	{
		case 0:
			m_CUnidades="arc-sec";
			m_cResolucion.Format("%d digitos", ResM3Me);
			m_precarga.Format("%11.1f arc-sec", PreM3Me);
			break;
		case 1:
			m_CUnidades="arc-sec";
			m_cResolucion.Format("%d digitos", ResM3In);
			m_precarga.Format("%11.1f arc-sec", PreM3In);
			break;
	}
	UpdateData(FALSE);
}

void CLaserDlg::OnRectitudlarga()
{
	UpdateData(TRUE);
	Escribe("M4\x00d\x00a\0");
	switch(m_iSU)
	{
		case 0:
			m_CUnidades="mm";
			m_cResolucion.Format("%d digitos", ResM4Me);
			m_precarga.Format("%11.4f mm", PreM4Me);
			break;
		case 1:
			m_CUnidades="in";
```

```
                    m_cResolucion.Format("%d digitos", ResM4In);
                    m_precarga.Format("%11.5f in", PreM4In);
                    break;
        }
        UpdateData(FALSE);
}

void CLaserDlg::OnRectitudcorta()
{
        UpdateData(TRUE);
        Escribe("M5\x00d\x00a\0");
        switch(m_iSU)
        {
                case 0:
                        m_CUnidades="mm";
                        m_cResolucion.Format("%d digitos", ResM5Me);
                        m_precarga.Format("%11.5f mm", PreM5Me);
                        break;
                case 1:
                        m_CUnidades="in";
                        m_cResolucion.Format("%d digitos", ResM5In);
                        m_precarga.Format("%11.6f in", PreM5In);
                        break;
        }
        UpdateData(FALSE);
}

void CLaserDlg::OnCcoefexp()
{
        char cadena[20];
        CCoefExp dlg;

        dlg.m_dCoefExp=CoefExp;
        if (dlg.DoModal() == IDOK)
        {
                CoefExp=dlg.m_dCoefExp;
                sprintf(cadena, "%1.1fEC\x00d\x00a\0",CoefExp);
                Escribe(cadena);
                sprintf(cadena, "%1.1f ppm",CoefExp);
                m_cCoefExp = cadena;
                UpdateData(FALSE);
        }
}

void CLaserDlg::OnCresolucion()
{
        CResolucion dlg;
        BOOL smooth;
        CString val;
        int prefijo;

        switch(m_iSU)
        {
                case 0:
                        switch(m_iModo)
                        {
                                case 0:
                                        dlg.ResRange=5;
```

```
                        dlg.m_iSpin=ResM1Me;
                        break;
                case 1:
                        dlg.ResRange=1;
                        dlg.m_iSpin=ResM2Me;
                        break;
                case 2:
                        dlg.ResRange=1;
                        dlg.m_iSpin=ResM3Me;
                        break;
                case 3:
                        dlg.ResRange=4;
                        dlg.m_iSpin=ResM4Me;
                        break;
                case 4:
                        dlg.ResRange=5;
                        dlg.m_iSpin=ResM5Me;
                        break;
        }
        break;
    case 1:
        switch(m_iModo)
        {
                case 0:
                        dlg.ResRange=6;
                        dlg.m_iSpin=ResM1In;
                        break;
                case 1:
                        dlg.ResRange=2;
                        dlg.m_iSpin=ResM2In;
                        break;
                case 2:
                        dlg.ResRange=1;
                        dlg.m_iSpin=ResM3In;
                        break;
                case 3:
                        dlg.ResRange=5;
                        dlg.m_iSpin=ResM4In;
                        break;
                case 4:
                        dlg.ResRange=6;
                        dlg.m_iSpin=ResM4In;
                        break;
        }
        break;
}

if (dlg.DoModal() == IDOK)
{
    switch(m_iSU)
    {
        case 0:
                switch(m_iModo)
                {
                        case 0:
                                ResM1Me=dlg.m_iSpin;
                                prefijo=ResM1Me;
                                smooth=dlg.m_cSmooth;
```

```
                        val="R1\x00d\x00a\0";
                        break;
                case 1:
                        ResM2Me=dlg.m_iSpin;
                        prefijo=ResM2Me;
                        smooth=dlg.m_cSmooth;
                        val="R2\x00d\x00a\0";
                        break;
                case 2:
                        ResM3Me=dlg.m_iSpin;
                        prefijo=ResM3Me;
                        smooth=dlg.m_cSmooth;
                        val="R3\x00d\x00a\0";
                        break;
                case 3:
                        ResM4Me=dlg.m_iSpin;
                        prefijo=ResM4Me;
                        smooth=dlg.m_cSmooth;
                        val="R4\x00d\x00a\0";
                        break;
                case 4:
                        ResM5Me=dlg.m_iSpin;
                        prefijo=ResM5Me;
                        smooth=dlg.m_cSmooth;
                        val="R5\x00d\x00a\0";
                        break;
        }
        break;
case 1:
        switch(m_iModo)
        {
                case 0:
                        ResM1In=dlg.m_iSpin;
                        prefijo=ResM1In;
                        smooth=dlg.m_cSmooth;
                        val="R1\x00d\x00a\0";
                        break;
                case 1:
                        ResM2In=dlg.m_iSpin;
                        prefijo=ResM2In;
                        smooth=dlg.m_cSmooth;
                        val="R2\x00d\x00a\0";
                        break;
                case 2:
                        ResM3In=dlg.m_iSpin;
                        prefijo=ResM3In;
                        smooth=dlg.m_cSmooth;
                        val="R3\x00d\x00a\0";
                        break;
                case 3:
                        ResM4In=dlg.m_iSpin;
                        prefijo=ResM4In;
                        smooth=dlg.m_cSmooth;
                        val="R4\x00d\x00a\0";
                        break;
                case 4:
                        ResM5In=dlg.m_iSpin;
                        prefijo=ResM5In;
```

```
                                        smooth=dlg.m_cSmooth;
                                        val="R5\x00d\x00a\0";
                                        break;
                                }
                                break;
                        }
                        CString cadena;
                        if(smooth) cadena.Format("%d",prefijo);
                        else cadena.Format("%d",-1*prefijo);
                        Escribe(cadena+val);
                        m_cResolucion.Format("%d digitos", prefijo);
                        UpdateData(FALSE);
                }
}

void CLaserDlg::OnCprecarga()
{
        CCprecarga dlg;
        CString val;
        double prefijo;

        dlg.modo=m_iModo;
        dlg.su=m_iSU;
        switch(m_iSU)
        {
                case 0:
                        switch(m_iModo)
                        {
                                case 0:
                                        dlg.m_cadena="-50000 50000 mm";
                                        dlg.m_precarga=PreM1Me;
                                        break;
                                case 1:
                                        dlg.m_cadena="-20000 20000 mm/min";
                                        dlg.m_precarga=PreM2Me;
                                        break;
                                case 2:
                                        dlg.m_cadena="0 2 arc-sec";
                                        dlg.m_precarga=PreM3Me;
                                        break;
                                case 3:
                                        dlg.m_cadena="0 3 mm";
                                        dlg.m_precarga=PreM4Me;
                                        break;
                                case 4:
                                        dlg.m_cadena="0 3 mm";
                                        dlg.m_precarga=PreM5Me;
                                        break;
                        }
                        break;
                case 1:
                        switch(m_iModo)
                        {
                                case 0:
                                        dlg.m_cadena="-2000 2000 in";
                                        dlg.m_precarga=PreM1In;
                                        break;
                                case 1:
```

```
                        dlg.m_cadena="-720 -720 in/min";
                        dlg.m_precarga=PreM2In;
                        break;
                case 2:
                        dlg.m_cadena="0 2 arc-sec";
                        dlg.m_precarga=PreM3In;
                        break;
                case 3:
                        dlg.m_cadena="0 3 in";
                        dlg.m_precarga=PreM4In;
                        break;
                case 4:
                        dlg.m_cadena="0 3 in";
                        dlg.m_precarga=PreM5In;
                        break;
        }
        break;
}
if (dlg.DoModal() == IDOK)
{
        switch(m_iSU)
        {
                case 0:
                        switch(m_iModo)
                        {
                                case 0:
                                        PreM1Me=dlg.m_precarga;
                                        m_precarga.Format("%1.5f mm", PreM1Me);
                                        prefijo=PreM1Me;
                                        val="P1\x00d\x00a\0";
                                        break;
                                case 1:
                                        PreM2Me=dlg.m_precarga;
                                        m_precarga.Format("%1.1f mm/min", PreM2Me);
                                        prefijo=PreM2Me;
                                        val="P2\x00d\x00a\0";
                                        break;
                                case 2:
                                        PreM3Me=dlg.m_precarga;
                                        m_precarga.Format("%1.1f arc-sec", PreM3Me);
                                        prefijo=PreM3Me;
                                        val="P3\x00d\x00a\0";
                                        break;
                                case 3:
                                        PreM4Me=dlg.m_precarga;
                                        m_precarga.Format("%1.4f mm", PreM4Me);
                                        prefijo=PreM4Me;
                                        val="P4\x00d\x00a\0";
                                        break;
                                case 4:
                                        PreM5Me=dlg.m_precarga;
                                        m_precarga.Format("%1.5f mm", PreM5Me);
                                        prefijo=PreM5Me;
                                        val="P5\x00d\x00a\0";
                                        break;
                        }
                        break;
                case 1:
```

```
                        switch(m_iModo)
                        {
                                case 0:
                                        PreM1In=dlg.m_precarga;
                                        m_precarga.Format("%1.6f in", PreM1In);
                                        prefijo=PreM1In;
                                        val="P1\x00d\x00a\0";
                                        break;
                                case 1:
                                        PreM2In=dlg.m_precarga;
                                        m_precarga.Format("%1.2f in/min", PreM2In);
                                        prefijo=PreM2In;
                                        val="P2\x00d\x00a\0";
                                        break;
                                case 2:
                                        PreM3In=dlg.m_precarga;
                                        m_precarga.Format("%1.1f arc-sec", PreM3In);
                                        prefijo=PreM3In;
                                        val="P3\x00d\x00a\0";
                                        break;
                                case 3:
                                        PreM4In=dlg.m_precarga;
                                        m_precarga.Format("%1.5f in", PreM4In);
                                        prefijo=PreM4In;
                                        val="P4\x00d\x00a\0";
                                        break;
                                case 4:
                                        PreM5In=dlg.m_precarga;
                                        m_precarga.Format("%1.6f in", PreM5In);
                                        prefijo=PreM5In;
                                        val="P5\x00d\x00a\0";
                                        break;
                        }
                        break;
                }
                CString cadena;
                cadena.Format("%1.6f",prefijo);
                Escribe(cadena+val);
                UpdateData(FALSE);
        }
}

void CLaserDlg::OnActtemp()
{
        CString uni, temp;

        UpdateData(TRUE);
        switch(m_iSU)
        {
        case 0:
                uni="C";
                break;
        case 1:
                uni="F";
                break;
        }

        Escribe("T1\x00d\x00a\0");
```

```
	temp=Lee();
	if(temp.Find("999999999")==2)
		m_cTemp1="No conectado";
	else
		m_cTemp1=temp+uni;

	Escribe("T2\x00d\x00a\0");
	temp=Lee();
	if(temp.Find("999999999")==2)
		m_cTemp2="No conectado";
	else
		m_cTemp2=temp+uni;

	Escribe("T3\x00d\x00a\0");
	temp=Lee();
	if(temp.Find("999999999")==2)
		m_cTemp3="No conectado";
	else
		m_cTemp3=temp+uni;

	Escribe("TA\x00d\x00a\0");
	temp=Lee();
	if(temp.Find("999999999")==2)
		m_cTempProm="No conectado";
	else
		m_cTempProm=temp+uni;

	switch(m_iModo)
	{
	case 0:
		Escribe("M1\x0d\x0a");
		break;
	case 1:
		Escribe("M2\x0d\x0a");
		break;
	case 2:
		Escribe("M3\x0d\x0a");
		break;
	case 3:
		Escribe("M4\x0d\x0a");
		break;
	case 4:
		Escribe("M5\x0d\x0a");
		break;
	}

	UpdateData(FALSE);
}

void CLaserDlg::OnCtemp()
{
	CCtempprom dlg;
	double prefijo;

	dlg.su=m_iSU;
	switch(m_iSU)
	{
	case 0:
```

```
			dlg.m_cCadena="0.00 50.00 C";
			dlg.m_dTemp=TempMe;
			break;
		case 1:
			dlg.m_cCadena="32.00 122.00 F";
			dlg.m_dTemp=TempIn;
			break;
		}

		if (dlg.DoModal() == IDOK)
		{
			switch(m_iSU)
			{
			case 0:
				TempMe=dlg.m_dTemp;
				prefijo=TempMe;
				break;
			case 1:
				TempIn=dlg.m_dTemp;
				prefijo=TempIn;
				break;
			}
			CString cadena;
			cadena.Format("%1.2f",prefijo);
			Escribe(cadena+"TA\x00d\x00a\x0");
			UpdateData(FALSE);
		}
}

void CLaserDlg::OnHtemp()
{
		int cuenta=0;

		UpdateData(TRUE);
		if(m_bhtemp1) cuenta=cuenta+1;
		if(m_bhtemp2) cuenta=cuenta+2;
		if(m_bhtemp3) cuenta=cuenta+4;

		if(cuenta>0)
		{
			CString str;

			str.Format("%d", cuenta);
			Escribe(str+"TP\x00d\x00a\x0");
		}
}

void CLaserDlg::OnActvol()
{
		CString uni1, uni2, temp;

		UpdateData(TRUE);
		switch(m_iSU)
		{
		case 0:
			uni1="C";
			uni2="mm Hg";
			break;
```

```
	case 1:
		uni1="F";
		uni2="in Hg";
		break;
	}

	Escribe("AP\x00d\x00a\0");
	temp=Lee();
	if(temp.Find("999999999")==2)
		m_cPresion="No conectado";
	else
		m_cPresion=temp+uni2;

	Escribe("AH\x00d\x00a\0");
	temp=Lee();
	if(temp.Find("999999999")==2)
		m_cHumedad="No conectado";
	else
		m_cHumedad=temp+"%";

	Escribe("AT\x00d\x00a\0");
	temp=Lee();
	if(temp.Find("999999999")==2)
		m_cTempAire="No conectado";
	else
		m_cTempAire=temp+uni1;

	Escribe("VL\x00d\x00a\0");
	temp=Lee();
	if(temp.Find("999999999")==2)
		m_cVOL="No conectado";
	else
		m_cVOL=temp;

	switch(m_iModo)
	{
	case 0:
		Escribe("M1\x0d\x0a");
		break;
	case 1:
		Escribe("M2\x0d\x0a");
		break;
	case 2:
		Escribe("M3\x0d\x0a");
		break;
	case 3:
		Escribe("M4\x0d\x0a");
		break;
	case 4:
		Escribe("M5\x0d\x0a");
		break;
	}

	UpdateData(FALSE);
}

void CLaserDlg::OnCvol()
{
```

```
	CCvol dlg;

	dlg.m_dVOL=NumVol;

	if (dlg.DoModal() == IDOK)
	{
		NumVol=dlg.m_dVOL;

		CString cadena;
		cadena.Format("%1.1f",NumVol);
		Escribe(cadena+"VL\x00d\x00a\x0");
		UpdateData(FALSE);
	}
}

void CLaserDlg::OnHvol()
{
	if(CompensaVOL)
	{
		if(Escribe("A0\x00d\x00a\0")) CompensaVOL=FALSE;
	}
	else
	{
		if(Escribe("A1\x00d\x00a\0")) CompensaVOL=TRUE;
	}
}
```

Resolucion.cpp

```
// Resolucion.cpp : implementation file
//

#include "stdafx.h"
#include "Laser.h"
#include "Resolucion.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CResolucion dialog


CResolucion::CResolucion(CWnd* pParent /*=NULL*/)
	: CDialog(CResolucion::IDD, pParent)
{
	//{{AFX_DATA_INIT(CResolucion)
	m_cSmooth = TRUE;
	m_iSpin = 0;
	//}}AFX_DATA_INIT
}


void CResolucion::DoDataExchange(CDataExchange* pDX)
```

```
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CResolucion)
	DDX_Check(pDX, IDC_SMOOT, m_cSmooth);
	DDX_Text(pDX, IDC_SDIGITO, m_iSpin);
	//}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CResolucion, CDialog)
	//{{AFX_MSG_MAP(CResolucion)
	ON_WM_VSCROLL()
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CResolucion message handlers

BOOL CResolucion::OnInitDialog()
{
	CDialog::OnInitDialog();

	// TODO: Add extra initialization here

	CSpinButtonCtrl *pSpin=
		(CSpinButtonCtrl*) GetDlgItem(IDC_SPIN1);
	pSpin->SetRange(0,ResRange);
	pSpin->SetPos(m_iSpin);

	return TRUE;  // return TRUE unless you set the focus to a control
		      // EXCEPTION: OCX Property Pages should return FALSE
}

void CResolucion::OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)
{
	// TODO: Add your message handler code here and/or call default

	if(nSBCode==SB_ENDSCROLL)
	{
		return;
	}
	if(pScrollBar->GetDlgCtrlID()==IDC_SPIN1)
	{
		CString strValue;
		strValue.Format("%d", (int) nPos);
		m_iSpin=nPos;
		UpdateData(FALSE);
	}

	CDialog::OnVScroll(nSBCode, nPos, pScrollBar);
}
```

Anexo B Funciones comunes del manejador NI-488.2

Función `ibfnd`	
Propósito	Abre un dispositivo y regresa el descriptor de unidad asociado con el nombre proporcionadp
Sintaxis	`int ibfind (char *udname)`
Descripción	`udname` es una cadena conteniendo un nombre de tarjeta por omisión o configurable. `ibfind` regresa un número que es usado en cada función para identificar una tarjeta o dispositivo en particular

Función: `ibtrg`	
Propósito	Dispara el dispositivo seleccionado
Sintaxis	`int ibtrg (int ud)`
Descripción	`ud` especifica un dispositivo, retomado por `ibfind`

Función: `ibrd`	
Propósito	Lee datos de un dispositivo hacia una cadena
Sintaxis	`int ibrd (int ud, void *rd, int cnt)`
Descripción	`ud` especifica un dispositivo, retomado por `ibfind` `rd` es la variable de almacenamiento de datos `cnt` es el número de bytes a leer

Función: `ibclr`	
Propósito	Limpia el dispositivo seleccionado
Sintaxis	`int ibclr (int ud)`
Descripción	`ud` especifica un dispositivo, retomado por `ibfind`

Función: `ibwrt`	
Propósito	Escribe datos desde una cadena
Sintaxis	`int ibwrt (int ud, int *wrt, int cnt)`
Descripción	`ud` especifica un dispositivo, retomado por `ibfind` `wrt` es la variable de almacenamiento de datos `cnt` es el número de bytes a escribir

Variable: `ibsta`	
Propósito	Variable de 16 bits que se renueva con cada operación del NI-488.2 para mantener vigilancia constante de los errores

Tabla 7. Funciones comunes del manejador NI-488.2.

Anexo C Funciones comunes del HP5528A

Código de la función del instrumento	Función del HP5528A
M1	Modo de distancia
M2	Modo de velocidad
M3	Modo angular
M4	Modo de rectitud larga
M5	Modo de rectitud corta
P1*	Pre-establecer/Modificar distancia
P2*	Pre-establecer/Modificar velocidad
P3*	Pre-establecer/Modificar ángulo
P4*	Pre-establecer/Modificar rectitud larga
P5*	Pre-establecer/Modificar rectitud corta
R1*	Despliega/Modifica resolución de distancia
R2*	Despliega/Modifica resolución de velocidad
R3*	Despliega/Modifica resolución de ángulo
R4*	Despliega/Modifica resolución de rectitud larga
R5*	Despliega/Modifica resolución de rectitud corta
T1	Despliega temperatura de material en el sensor 1
T2	Despliega temperatura de material en el sensor 2
T3	Despliega temperatura de material en el sensor 3
TA*	Despliega/Modifica temperatura de material promedio
AP	Presión del aire
AH	Humedad del aire
AT	Temperatura del aire
VL*	Despliega/Modifica número de compensación V.O.L.
EC*	Despliega/Modifica el coeficiente de expansión
RC*	Registra salida IEEE-488
RS	Re-inicio
D1	Sentido de dirección al encendido
D2	Sentido de dirección inversa al encendido
UE	Unidades inglesas
UM	Unidades métricas
C0	Deshabilita número de compensación
C1	Habilita número de compensación
TP*	Habilita sensor por temperatura del material

Tabla 8. Funciones comunes del manejador HP5528A.