UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA

ELECTRICA – INSTRUMENTACION

Automatización de un sistema de irradiación de superficies con láser pulsado e implementación de monitoreo óptico del proceso a tiempo real.

TESIS

QUE PARA OPTAR POR EL GRADO DE:

MAESTRO EN INGENIERÍA

PRESENTA:

RICARDO RODRIGUEZ GONZALEZ

TUTOR (ES) PRINCIPAL(ES)

NASER QURESHI, ICAT

CITLALI SANCHEZ AKÉ, ICAT

CIUDAD UNIVERSITARIA, CD. MX., AGOSTO DE 2019

JURADO ASIGNADO:

Presidente:	Dr. Garduño Mejía Jesús
Secretario:	Dr. Pérez Ruiz Santiago Jesús
Vocal:	Dr. Qureshi Naser
1 er. Suplente:	Dra. Sánchez Aké Citlali
2 d o. Suplente:	Dr. Velasco Herrera Víctor Manuel

Lugar o lugares donde se realizó la tesis: Ciudad Universitaria, UNAM, México.

TUTOR DE TESIS:

Dr. Qureshi Naser

FIRMA

AGRADECIMIENTOS

"Después de la tormenta todo se ve frio, oscuro y silencioso, aunque el día este soleado y despejado."

Hoy no hay dedicatoria especial a mi familia, solo a ti, que no solo eres mi sangre, sino parte de mi corazón, juntos nos hemos visto crecer. A veces pienso que tú eres el mayor, porque soportaste mejor los golpes de la vida, hasta ofreciste tu hombro para desahogarme. Espero que sigamos creciendo ¿juntos o separados?, ¡qué importa!, cada uno tiene que seguir hacia delante, porque al final del camino nos volveremos a encontrar, para contarnos las anécdotas de nuestro destino.

En esta nueva travesía de camino a mi sueño conocí personas raras, inteligentes y muy agradables, de distintos estados y nacionalidades, que me han compartido sus experiencias y conocimiento, brindándome un apoyo incondicional hasta en mis peores momentos. Tal vez sea suerte o casualidad, el rodearme de buenas personas, porque no siempre las merezco.

Solo puedo decir:

- *"Gracias a todos los que han confiado en mí."*

Por último, agradezco al Consejo Nacional de Ciencia y Tecnología (CONACyT) por la beca otorgada para la realización de mis estudios de maestría, al Proyecto PAPIIT IG100418 financiado por la DGAPA-UNAM y al proyecto del Consejo Nacional de Ciencia y Tecnología (CONACyT 253754).

RESUMEN

El presente trabajo consiste en la automatización de un sistema de procesamiento de materiales mediante la irradiación de superficies por láser pulsado en el régimen de nanosegundos, con la posibilidad de monitorear en tiempo real los cambios en las propiedades ópticas de la superficie y la remoción de material, debido a los pulsos láser. Este sistema permite modificar la energía del láser, número de pulsos y posición de la muestra, por lo que puede ser utilizado para numerosas aplicaciones como la limpieza, modificación de rugosidad y propiedades ópticas de una superficie, entre otras. En particular, se presenta una aplicación: remoción de películas metálicas depositadas previamente sobre vidrios. Nuestro interés en esta aplicación es utilizarla en el futuro para la fabricación de nanopartículas a partir de la irradiación de películas delgadas metálicas previamente depositadas sobre vidrios y la limpieza de vidrios, como se ha hecho en el laboratorio anteriormente de manera manual y semiautomática [1, 2, 3]

El sistema automatizado permite monitorear los cambios en la superficie (reflexión o transmisión de una longitud de onda en particular), en función del número de pulsos láser, constituyendo así una herramienta eficiente no sólo para la fabricación de materiales sino también para el estudio de los procesos físicos que ocurren durante la interacción láser-materia. Para tal objetivo se desarrolló una interfase de usuario que facilita el manejo del sistema de la técnica de irradiación laser, controlando cada uno de los instrumentos, sensores y actuadores en una sola aplicación. El desarrollo de la programación fue realizado en el software de ingeniería Labview 2013 de National Instruments, debido a su enfoque en el control de instrumentos. El sistema consta de un láser pulsado Nd:YAG de la firma EKSPLA modelo NL300 de una longitud de onda de 1064 nm, un osciloscopio Tektronix TDS 5054B, una tarjeta de adquisición de datos (DAQ) USB-6001, una fuente HM7044 de Hameg , fotodiodos de respuesta rápida DET10A/M de Thor Labs (tiempo de respuesta de 1 ns) y un sistema de movimiento (x,y,z) compuesto por tres platinas manuales PT1M de Thor Labs acopladas cada una por un motor paso a paso (PAP) NEMA 17.

La interfaz de usuario está dividida en tres pestañas:

La primera pestaña inicializa la comunicación de la aplicación con los instrumentos y actuadores del sistema, además de configurar los parámetros de inicio de cada uno de ellos para su posterior ejecución en la técnica de procesamiento por láser.

La segunda pestaña controla la técnica de irradiación por láser mediante una rutina que sincroniza todos los instrumentos y actuadores, además almacena los datos proporcionados para su posterior análisis.

La rutina consiste en que cada vez que un pulso de laser NL300 de Ekspla irradia una parte de la muestra, se detectan dos señales con la ayuda de dos fotodiodos rápidos conectados a un osciloscopio Tektronix de modelo TDS 5054B, que lee y envía los datos obtenidos en el ordenador. Las señales de los fotodiodos detectan a tiempo real los cambios en la superficie, uno puede ser para detectar la reflexión o transmisión de la zona irradiada y el otro para verificar si hay ablación del material mediante la técnica conocida como deflectometría. Una vez irradiada un área de la muestra a una posición fija, se procede a mover las platinas (y,z) con la ayuda de motores paso a paso (PAP), para irradiar la siguiente zona, y así se repite el mismo proceso hasta acabar de irradiar por completo la superficie de la muestra, la cual no debe exceder los 25 mm de desplazamiento máximo de las platinas.

La tercera pestaña da la posibilidad de controlar de manera manual, la posición de la muestra en los tres ejes, y controla el modo y disparo del láser.

En este trabajo se describe el sistema automatizado y se presentan experimentos preliminares de irradiación de películas metálicas sobre vidrio

ÍNDICE

ACRONIMOS

Al: Aluminio

ASCII: American Standard Code for Information Interchange

Au: Oro

CAN: Controller Area Network

DAQ: Data Acquisition

Nd:YAG: Neodymium:Doped Yttrium Aluminium Garnet

OPC: Operation Complete

PAP: Paso a Paso

PCB: Pinted Circuitos Board

TTL: Transistor-Transistor Logic

VREF: Voltaje de Referencia

INTRODUCCIÓN

El procesamiento de materiales con láser es un campo en expansión de gran interés tanto para la ciencia básica como la ingeniería. El aspecto más importante para la ciencia es entender los mecanismos de interacción entre el haz láser y los materiales. Mientras que, para la ingeniería, el láser no sólo representa una herramienta de manufactura económica, rápida y limpia, sino que también abre un abanico opciones de fabricación que no son factibles usando los métodos convencionales. En particular, el procesamiento de superficies por láser tiene un gran potencial en la fabricación directa (sin pasos posteriores) de micro y nanoestructuras para diferentes aplicaciones [4].

Los cambios generados en una superficie como consecuencia de la estructuración de superficies irradiación con láser pulsado depende completamente de la densidad de potencia incidente, el número de pulsos y el tipo de material que se está irradiando [5]. Estos parámetros determinan si hay fusión, evaporación o simplemente modificación de la superficie, por lo que controlar las variables experimentales que involucra el láser es un factor clave para la aplicación de este método. Más aún, el haz láser interacciona de manera distinta con cada material, por lo que contar con una herramienta de monitoreo in situ y a tiempo real es fundamental para conocer la transformación o posible daño del material durante la irradiación.

Este proyecto consiste en la automatización de un sistema de irradiación láser, que permita controlar, en un rango amplio, la densidad de energía del láser y el número de pulsos. En paralelo, se implementa un sistema óptico para determinar los cambios en la superficie irradiada a tiempo real. La irradiación de superficies se ha utilizado ampliamente de manera manual y semiautomática en el laboratorio de Fotofísica y películas delgadas del Instituto de Ciencias Aplicadas y Tecnología [1, 2, 3]. Se ha utilizado para la irradiación de películas delgadas metálicas sobre substratos dieléctricos, pues estudios recientes han mostrado que como resultado de la irradiación se puede producir dos tipos de nanoestructuras: i) nanopartículas metálicas adheridas a la superficie del substrato [4, 6, 7, 8], ii) perforaciones de tamaño nanométrico en la superficie del substrato [9, 10]. Debido a que la modificación de la superficie depende de la densidad de energía del láser y el número de pulsos aplicados, la implementación de técnicas ópticas para la caracterización in situ y a tiempo real de la

superficie irradiada, permitirá controlar el proceso y garantizar su reproducibilidad. En particular, se propone medir la intensidad de luz transmitida y reflejada por la zona irradiada durante y después de cada disparo láser. Asimismo, se propone utilizar un haz láser continuo rasante a la superficie irradiada para monitorear el proceso in situ. La desviación de este haz indicará si hay remoción del material irradiado, permitiendo así minimizar posibles daños y controlar el proceso.

Como antecedente, las técnicas ópticas mencionadas han sido utilizadas en la caracterización de procesos de interacción láser con materia sólida [8, 2, 11]. Sin embargo, para el caso particular de nanoestructuración por irradiación láser de películas metálicas, hay escasos reportes del uso de estas técnicas [8]. Utilizar otras dos caracterizaciones ópticas simultáneamente (transmitancia y deflexión de haz rasante), derivará en mejorar el entendimiento del proceso y mejorar esta técnica para manufactura. Nuestro grupo de investigación, Fotofísica y Películas Delgadas del Instituto de Ciencias Aplicadas y Tecnología, ya ha mostrado que la combinación de estas dos metodologías aporta información de la escala temporal de los procesos involucrados y pueden ser usados para controlar la reproducibilidad del procesamiento.

CAPITULO 1: PROCESAMIENTO DE MATERIALES CON LASER

El procesamiento de materiales con láser es un campo en expansión de gran interés tanto para la ciencia básica como la ingeniería, teniendo aplicaciones diversas como en el micro maquinado, análisis elemental, la producción de nanopartículas, etc. El procesamiento con láser es el nombre que se otorga a la modificación de propiedades físicas y químicas de los materiales mediante irradiación láser, dotándolos de las características que se requieren para diversas aplicaciones tecnológicas. El procesamiento por láser depende principalmente de los parámetros del láser y del material, por lo que cada caso es diferente dependiendo del objetivo a realizar [12].

1.1 Irradiación por pulso láser

Una de las técnicas más populares para el procesamiento de materiales es la irradiación con láser pulsado, debido a que tiene la posibilidad de evaporar compuestos de elevada complejidad en un área controlada. Esta evaporación permite la limpieza de superficies o la modificación de las mismas, por ejemplo, con la formación de canales y agujeros que cambian las propiedades ópticas y mojabilidad de superficies. Adicionalmente, la evaporación conserva la estequiometría del material, por lo que esta técnica también se utiliza para la fabricación de películas delgadas y nanopartículas, pues la limpieza del proceso otorga bajo nivel de incorporación de impurezas. Además, debido a la corta duración del proceso, que por lo general tiene una duración de unos cuantos microsegundos, el procesamiento se realiza en un área local del material mientras que el resto permanece inalterado [13].

El equipo necesario para utilizar esta técnica requiere de un láser de alta potencia como fuente de excitación y una base motorizada capaz de cambiar la zona de irradiación de la muestra. El sistema de movimiento se elige dependiendo de la aplicación requerida, por ejemplo, para desplazar la superficie con precisión micrométrica para ser utilizados incluso dentro de sistemas de vacío o bien equipos robustos que para el desplazamiento de muestras es de metros cuadrados de área. En cuanto a la fuente de irradiación, se utilizan distintos tipos de láseres en operación continua o pulsada y con frecuencias desde el infrarrojo hasta el cercano ultravioleta. Los láseres en operación pulsada con duración de pulso en el rango de los

nanosegundos, como el excímero y los Nd:YAG son los más usados para esta técnica porque una amplia gama de materiales absorbe bien la longitud de onda de éstos.

Una de las técnicas para la fabricación de nanopartículas metálicas se basa en el uso de irradiación de superficies, en particular en una capa metálica depositada previamente sobre un substrato no conductor. Misma que al ser irradia con pulsos cortos (“ns”) o pulsos ultracortos (“fs”), deriva en la formación de nanoclusters y nanoestruturas como resultado de la fusión del metal y la poca adherencia entre éste y el substrato.

1.1.1 Fabricación de nanopartículas

La técnica de irradiación laser que consiste en irradiar una película delgada depositada previamente, con pulsos láser de corta duración ("ns") o pulsos ultracortos ("fs"). El depósito de las películas delgadas se puede hacer por distintas técnicas como ablación laser o sputtering, el aspecto importante es que se depositen sobre substratos con poca adherencia al metal [14]. Si la energía del pulso láser es suficiente para fundir el metal, se formarán gotas sobre el substrato debido a la baja adherencia metal-substrato, que al enfriarse se convertirán en nano-islas o nanopartículas. El tamaño de las nanopartículas es de decenas de nanómetros y con las condiciones adecuadas se puede obtener partículas de menor tamaño y forma regular [6]. En este proceso intervienen muchas variables que definen la estructura de las nanopartículas, como son la presión, la temperatura y la humedad, además de las características del láser como la duración del pulso y la fluencia. También intervienen las características del material irradiado como su grosor, sus propiedades térmicas, etc.

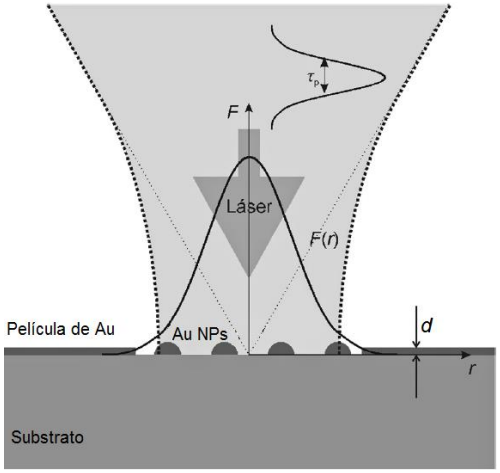


Figura 1: *Proceso de formación de nanopartículas por irradiación de un pulso láser de duración (τ) y fluencia $F(r)$con una película delgada de oro de nanometros de grosor (d)* [15].

En la *figura 1* se puede apreciar un esquema del proceso que se lleva a cabo en un material irradiado (Au) con un pulso, este material está sobre la superficie de un sustrato dieléctrico, en forma de una capa delgada definida por un grosor d, éste se funde como consecuencia de la energía entregada por el haz de un láser pulsado. El metal fundido tiende a formar gotas si hay poca adherencia entre éste y el substrato, generando así las nanopartículas cuando las gotas metálicas se enfrían.

Para monitorear a tiempo real e in situ la formación de nanopartículas (y así lograr un mejor control de la técnica), se utilizaron dos métodos de caracterización ópticos: (i) medición de la intensidad transmitida de un haz láser continuo antes y después del procesamiento láser y (ii) reflectometría de un haz rasante a la muestra.

1.1.2 Métodos ópticos de monitoreo y caracterización

Todas las sustancias tienen propiedades ópticas que están definidas por las partículas que las componen, como la absorción, transmitancia y reflectancia. Se han creado diversos métodos capaces de aprovechar la lectura de estas propiedades para monitorear y caracterizar procesos de manera no destructiva, como la espectroscopia, en el análisis de los espectros de emisión. Para irradiación laser, la espectroscopia es muy usada cuando la energía es suficiente para generar un plasma. En cambio, si la fluencia es baja se pueden usar otros métodos como la transmitancia, reflectancia y deflectometría de un haz laser, además cuentan con la ventaja de poder ser adaptados al proceso, tomando medidas de manera in situ [1].

1.1.2.1 Transmitancia

La transmitancia estudia los cambios de la intensidad de la luz transmitida a través de la zona tratada durante el proceso, es decir, que la transmitancia (T) es la razón entre la intensidad de luz transmitida por la muestra (I) y la intensidad de luz incidente (I_0).

$$T = I/I_0 \qquad (1)$$

Donde T depende la longitud de onda. En este trabajo, para la medición de T, se coloca un láser He-Ne (632.8nm) delante de la muestra y atrás un fotodetector DET10A/M de Thor

Labs para que detecte las variaciones del haz que atraviese la muestra durante el proceso (*Figura 2*), es decir antes, durante y después de la irradiación con el láser pulsado. Una desventaja de este método es que no puede ser utilizado en muestras opacas, es decir la transmitancia puede estudiar películas delgadas cuyo espesor máximo sea del orden de ~$3\alpha^{-1}$, donde α^{-1} es la longitud de absorción o penetración óptica y α el coeficiente de absorción [16].

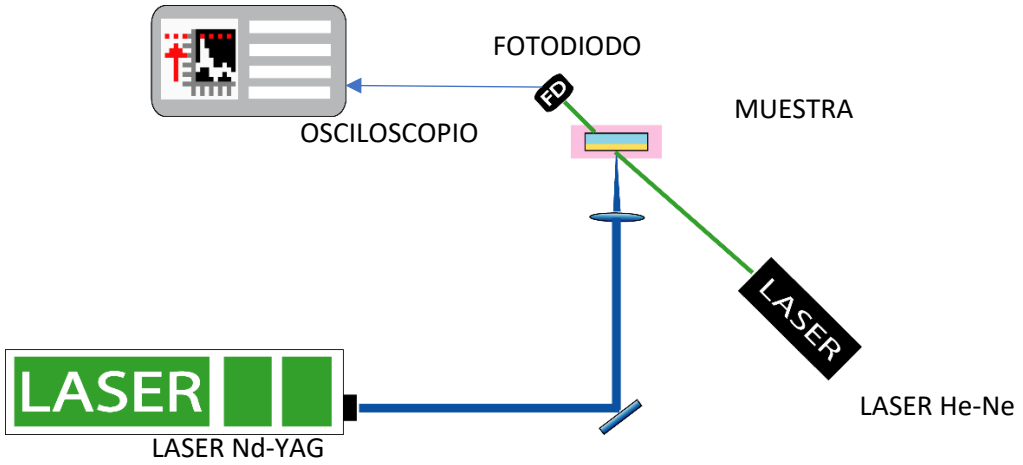


Figura 2: *Sistema de medición para la técnica de transmitancia. a) Osciloscopio, b) Fotodiodo de respuesta rápida (1 ns), c) Laser Helio-Neón, d) Laser pulsado de Neodimio YAG.*

1.1.2.2 Deflectometría

La técnica de deflectometría se basa en el efecto "*mirage*" el cual ocurre cuando un haz de luz se propaga por medio cuyo índice de refracción no es constante, resultando que su trayectoria se curva. Las variaciones pueden ocurrir debido a los cambios de temperatura del medio, debido a la transferencia de calor desde la película metálica irradiada, cambio en el índice de refracción debido al paso de una onda de choque, o al esparcimiento del haz continúo debido a la eyección de la materia desde la película o sustrato [17].

Dado a que el índice de refracción (n) está en función de la temperatura y que el calor generado en el material se propaga también a través del gas que lo rodea, un haz que se propague por el aire cerca de la zona iluminada sufrirá una desviación periódica, el diferencial del ángulo ($d\phi$) que se desvía el haz reflejado queda dado por la variación del índice de refracción:

$$d\phi = 1/n\ (\partial n/\ \partial x)\ dy \qquad (2)$$

Para su estudio se usó un láser continuo He-Ne paralelo a la superficie a irradiar (*Figura 3*), para obtener el haz rasante sobre la muestra y detectar las variaciones del ángulo del haz, por medio de la energía detectada al otro extremo por el fotodiodo.

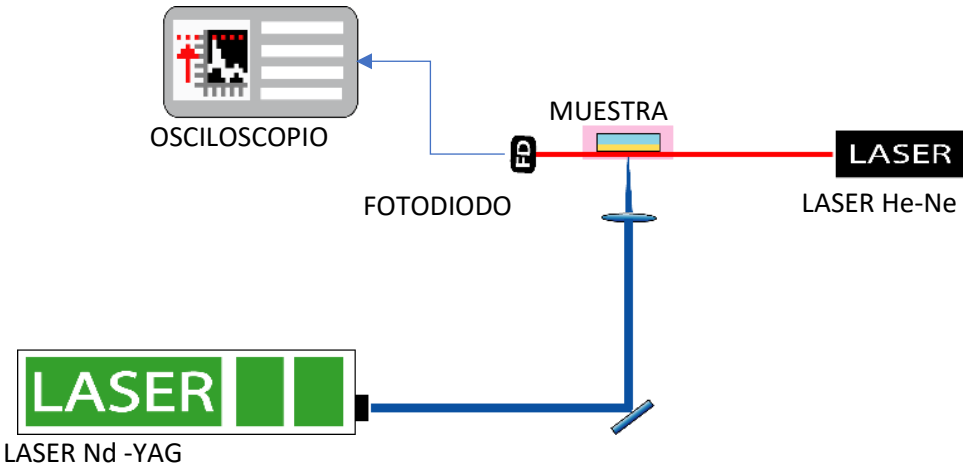


Figura 3: *Sistema de medición para la técnica de deflectometría a) Osciloscopio, b) Fotodiodo de respuesta rápida (1 ns), c) Laser Helio-Neón, d) Laser pulsado de Neodimio YAG.*

1.2 Laser Nd:YAG

El láser de Nd:YAG es de clase IV de estado sólido en un sistema de cuatro niveles, su medio activo es el granate oxido de itrio y aluminio dopado con neodimio para producir una emisión más fuerte, su emisión característica se encuentra en el infrarrojo, con una longitud de onda de 1064nm.

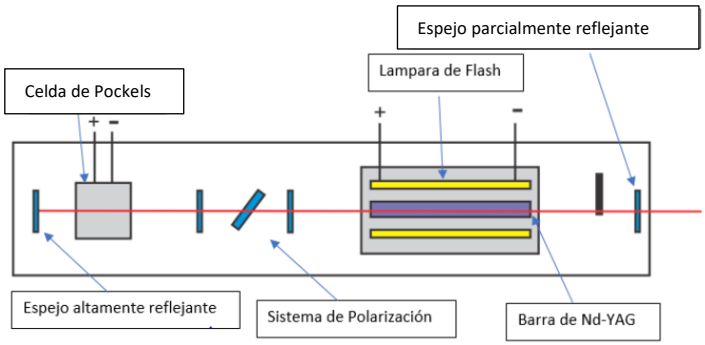


Figura 4: *Esquema de un láser de Nd-YAG pulsado.*

Este láser funciona de manera pulsada, el cual está compuesto por una barra solida de Nd-YAG, encapsulada con una lampara de flash, la cual excitara el medio activo para la generación de irradiación laser. Un arreglo de dos espejos compone la cavidad, uno altamente reflejante y otro parcialmente reflejante, entre ellos se encuentra la celda de Pockels, la cual contiene un cristal líquido que, al aplicar una diferencia de potencial, cambia su índice de refracción, dejando pasar la luz. Con este sistema, denominado Q-switch, el láser opera de manera pulsada en el rango de ns (*Figura 4*).

1.2.1 Laser NL300 de Ekspla

El láser NL300 de la firma Ekspla es un láser de Nd:YAG de pulsos de duración de ns, que trabaja a una longitud de onda de 1064nm. Su principio de funcionamiento se describe en la *figura 5*, en donde se puede ver que se manda una señal TTL (1) en un tiempo T1, para activar la lampara flash (2) y aplicar un voltaje alto a la celda Pockels , que por la presencia de este, el Q switch (3) se mantendrá cerrado induciendo perdidas de energía elevadas en el resonador, hasta el tiempo T2, en donde la varilla de Nd:YAG alcance la inversión máxima de población, en ese mismo momento el Q-switch se abre, minimizando las pérdidas de energía del resonador, generando un pulso óptico muy corto y energético(4).

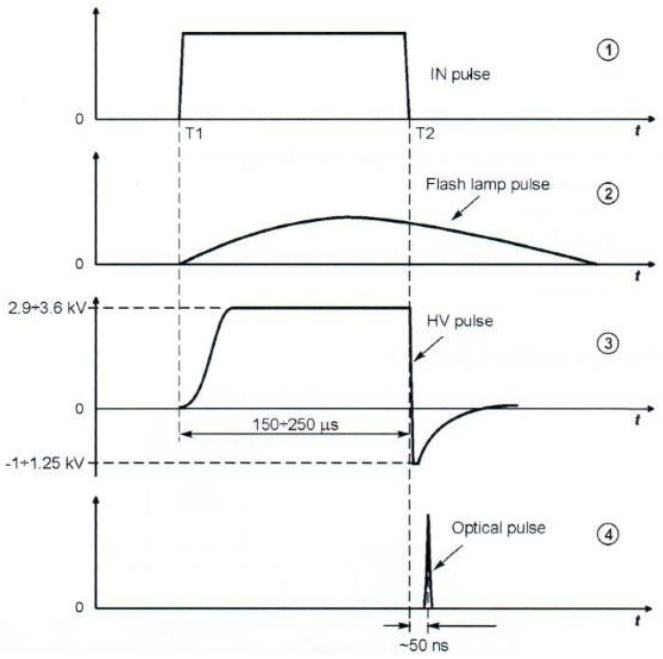


Figura 5: *Principio de funcionamiento del láser- NL300 de Ekspla* [18].

La señal que controla el pulso del láser es una señal de una frecuencia de 10 Hz, con una amplitud máxima de 8 volt, la cual es generada por la electrónica interna del láser, aunque también puede ser recibida de manera externa.

El láser NL300 funciona de dos modos: usando los controles del teclado de control o de manera remota mediante un controlador externo. El fabricante te proporciona los drivers, el manual y un programa de prueba para el control externo del láser.

En el diagrama de la *figura 6* se aprecia la estructura del funcionamiento de los archivos que hacen posible la comunicación con el ordenador.

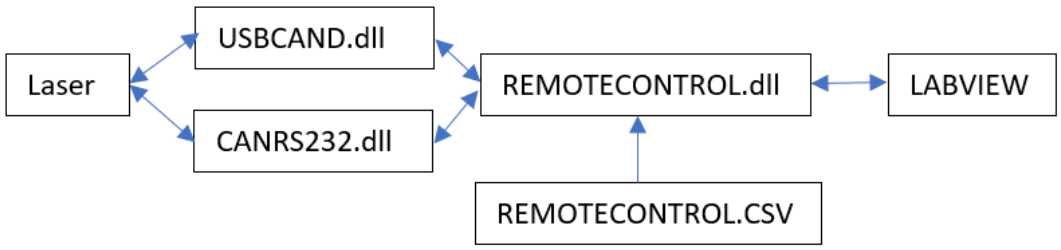


Figura 6: *Diagrama de la comunicación del láser con Labview.*

REMOTECONTROL.dll es una biblioteca compartida que administra la comunicación entre la aplicación y el dispositivo. Traduce las funciones enviadas a paquetes de datos enviados y recibidos a través de la interfaz USB. Las aplicaciones de control utilizan esta biblioteca para comunicarse.

USBCAND.dll Es una biblioteca que implementa la comunicación de la interfaz RS232-USB.

CANRS232.dll: Es una biblioteca que implementa la comunicación de la interfaz RS232-CAN

REMOTECONTROL.csv: Contiene una lista de registros de los módulos del láser que se pueden controlar o leer.

REMOTECONTROL.h Es un archivo de encabezados que contiene declaraciones directas de subrutinas y variables de la biblioteca REMOTECONTROL.dll.

Del láser NL300 pueden controlarse las funciones de la tabla de la figura 7 con ayuda del control remoto.

Función	Subfuncion	Descripción
Output level	E. max	Se opera el láser en la máxima energía que el pulso puede brindar.
	E adjust	La energía del pulso del láser es aproximadamente el 10% del valor nominal.
	E. OFF	El láser no emite ningún pulso.
Burst Mode	Continuous	Activa modo continuo
	Burst Mode	Activa el modo burst
	Burst trigger	Genera los pulsos del modo burst
Burst Length		Indica el número de pulsos en el modo ráfaga, este modo solo soporta de 0 a 99 pulsos.
Pump voltage		Controla el voltaje de la lampara flash.
Set Cooling T		Este menú presenta la temperatura del agua de refrigeración
Read Cooling T		Este menú presenta el monitoreo de la temperatura del agua de refrigeración.
Pulse Counter		Permite calcular el número total de pulsos del láser multiplicado por 1000.

Synchronization	Mode (Extenal or Internal)	Cambia entre el modo interno o externo del láser.
	EO ON/OFF	Apaga o enciende la señal electro-óptica del láser.
	SynOut delay	Se ajusta el retardo entre el Q switching y OUT sinc, en un rango de +- 8ms y para el modo externo de 8 ms a -1ms en operación.
	Best EO delay	Es el retardo entre es oscilador de la lampara flash y la electro- óptica.
	Adj. EO delay	Es el retardo entre el oscilador de la lampara flash y la electro-óptica, en el modo de operación adjust.
Safety		Puede activar las advertencias de errores durante la operación del láser.
Display Layout		Configura la organización del menú del control remoto del láser.
Freq. Divider		Es un numero del 0 al 99, que divide la frecuencia del pulso laser.

Figura 7: *Tabla de las funciones del láser controladas manualmente.*

CAPITULO 2: DESARROLLO

Para explicar mejor el desarrollo del sistema lo dividiremos en tres partes, como se puede ver en la figura 8. En la primera parte (1) se encuentra el sistema de movimiento de tres ejes, compuesto por tres platinas PT1M de ThorLabs acopladas cada una a un motor PAP NEMA 17, encargados de posicionar la zona de la muestra a irradiar, enfrente de ella se encuentra un lente divergente el cual enfoca el haz hacia la superficie de la muestra, esto nos permitirá aumentar la fluencia en la zona a irradiar. En la segunda parte (2), se encuentra el láser NL300 de Ekspla, en que se controlará la energía, número de pulsos y frecuencia de repetición y cuyo haz será direccionado hacia la lente. Por último, se presenta la parte 3 del sistema, en ella se encuentran los fotodiodos que permitirán el monitoreo del proceso, con la ayuda del osciloscopio TDS 5054B de Tektronix.

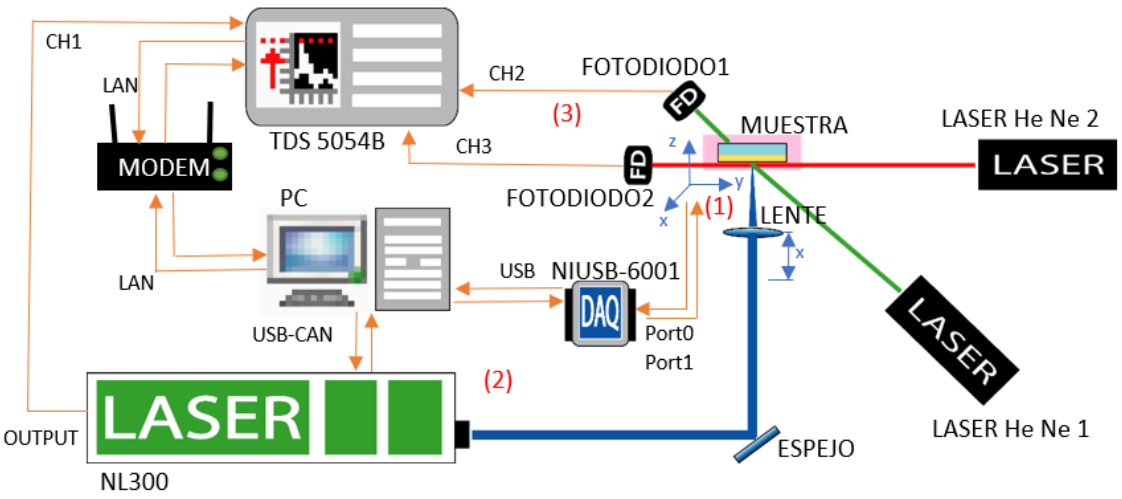


Figura 8: *Arreglo experimental*

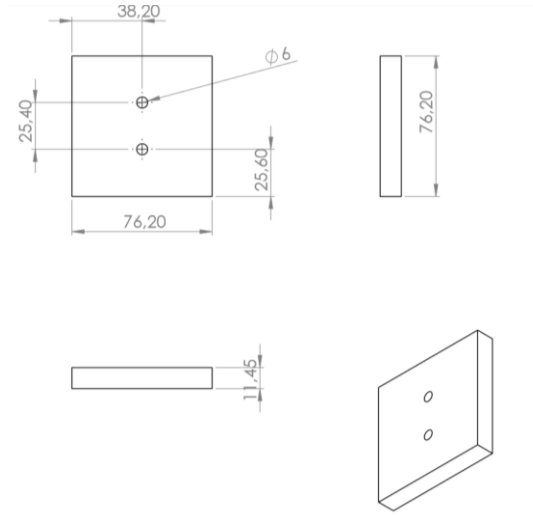
2.1 Sistema de Movimiento

Se desarrolló un sistema de posicionamiento de tres ejes (x,y,z), este sistema consta de 3 motores PAP NEMA 17 de 1.8º a 1.3 A por bobina, de 3 platinas PT1M de ThorLabs de 2.5um de resolución y 25mm de desplazamiento y de aditamentos que ayudan a embonar las piezas del sistema.

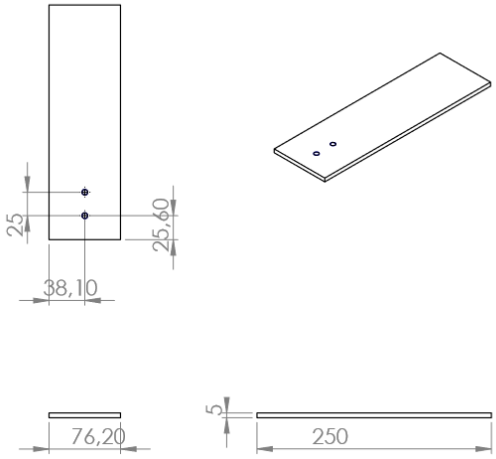
Se realizaron varios bosquejos para ver cómo adaptar los motores a las platinas, posteriormente desarrollamos los planos en Solidworks (*Figura 9*), para tener una mayor visualización de las piezas a realizar en el taller, la idea era lograr que la flecha del motor

quede lo más alineada al centro del tornillo milimétrico de cada platina para evitar cualquier irregularidad en él movimiento de los motores PAP. Se decidió usar acoplamientos comerciales a los cuales se les hicieron pequeñas modificaciones para conectar el tornillo milimétrico con la flecha de motor, debido a que funcionan como resorte compensan el juego que hay entre la platina y el motor PAP.

Base Niveladora



Soporte de Motor



Cople Motor-Platina

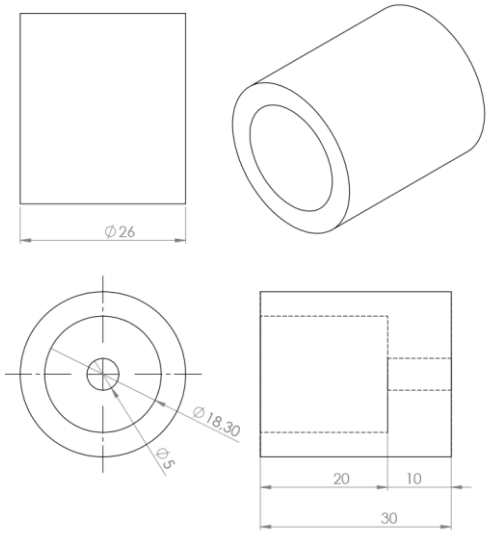


Figura 9: *Plano de las piezas de acoplamiento*

En la *figura 10* se aprecia el ensamblado del sistema de movimiento desarrollado en Solidworks, el eje “y” y “z” serán los encargados de posicionar la muestra, mientras el eje “x” será el encargado de definir la distancia entre el lente de enfoque y la muestra. También se anexaron 3 *push bottom*, para poder definir la posición de inicio de cada motor y conocer la posición de la platina en todo momento.

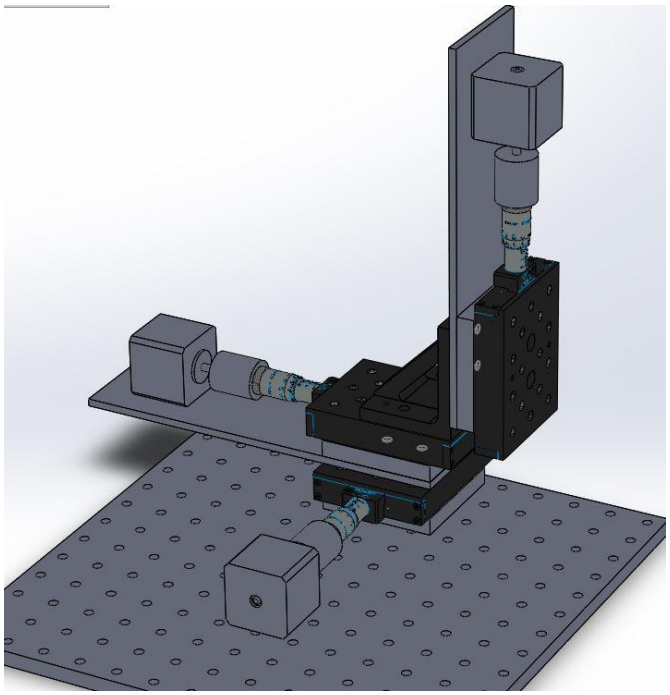


Figura 10: *Sistema de movimiento de tres ejes*

2.1.1 Control de motores PAP

Para el control de los motores PAP que están acoplados a las platinas, se utilizó el controlador comercial DRV-8825 de Pololu Robotics and Electronics el cual nos ofrece el control de un motor bipolar PAP de 8v a 45v de alimentación y soporta hasta 1.5 A de corriente eléctrica por bobina y con refrigeración hasta 2.2 A, también se puede regular la corriente, con el potenciómetro que se encuentra en la placa, con la ayuda de la fórmula 3 proporcionada por el fabricante.

$$\text{Límite de corriente} = \text{VREF} \times 2 \qquad (3)$$

En la figura 11 se puede apreciar el circuito para paso completo del controlador DRV8825 y el nombre de todos los pines que lo componen. El pin ENABLE, RESET y SLEEP están activos mientras su estado este en LOW, el pin ENABLE habilita el controlador y los pines RESET y SLEEP evitan que funcione el controlador. Por tal motivo se conectan a la fuente de voltaje del sistema lógico (microcontrolador, Arduino, DAQ, etc). El controlador DRV8825 es alimentado por una segunda fuente de mayor potencia, la cual le dará la fuerza suficiente para mover los motores PAP.

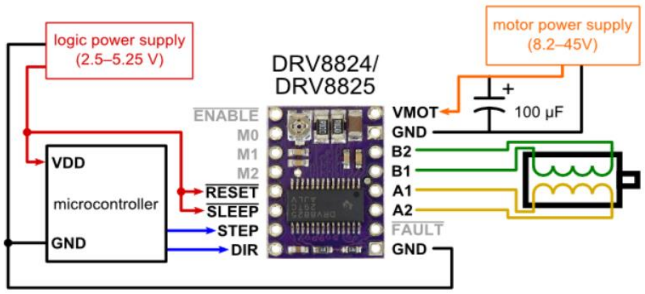


Figura 11: *Circuito para conectar DRV-8825 configurado de paso completo* [19].

Al pin STEP se le envía una señal cuadrada, el controlador se encargará de descomponer esta señal a cuatro señales cuadradas, dos de ellas desfasadas para ser enviadas a las bobinas y activarlas en diferente orden y producir el movimiento al motor PAP. El tamaño del periodo de la señal cuadrada determinará la velocidad del motor PAP. Con el pin DIR se controla la dirección del motor y con los pines M0, M1 Y M2 se configuran los tipos de micropasos los cuales son encargados de multiplicar el número total de pasos de una vuelta del motor PAP, y aumentar la precisión acosta de un decremento de velocidad y torque, el controlador es capaz de soportar pasos de 1/2, 1/4, 1/8, 1/16 y de 1/32. Los pines donde se conectarán bobinas son B2, B1, A1 y A2. La configuración de los micropasos se puede ver en la tabla de la figura 12.

Resolución de Micro pasos	**Modo 0**	**Modo 1**	**Modo 3**
Paso completo	Bajo	Bajo	Bajo
Medio Paso	Alto	Bajo	Bajo
1/4 de paso	Bajo	Alto	Bajo

1/8 de paso	Alto	Alto	Bajo
1/16 de paso	Bajo	Bajo	Alto
1/32 paso	Alto	Bajo	Alto
1/32 paso	Bajo	Alto	Alto
1/32 paso	Alto	Alto	Alto

Figura 12: *Configuración de pines para la resolución de micropasos*

2.1.2 Diseño de PCB

Debido a la falta de entradas y salidas digitales de la DAQ, se decidió unir los pines de DIR, M0, M1 y M2 de los tres controladores, para tener las mismas características en los tres motores además de tomar como referencia el circuito proporcionado por el fabricante, manteniendo el controlador siempre activo mientras esté conectado a una fuente de voltaje.

Se diseño el esquemático (*Figura 13*) y la PCB (*Figura 14*) en el programa Eagle, donde se creó la librería DRV-8825 especificando las características físicas y pines del controlador de motores PAP, para facilitar el diseño de la placa Pinted Circuitos Board (PCB). Se agregaron planos de tierra para evitar cualquier tipo de interferencia entre las señales de control.

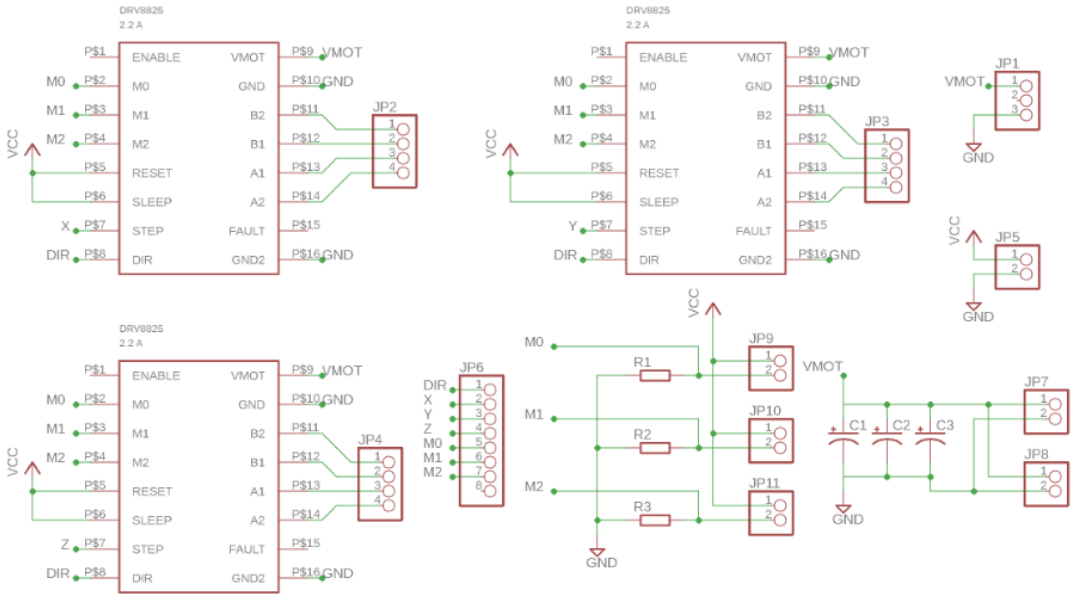
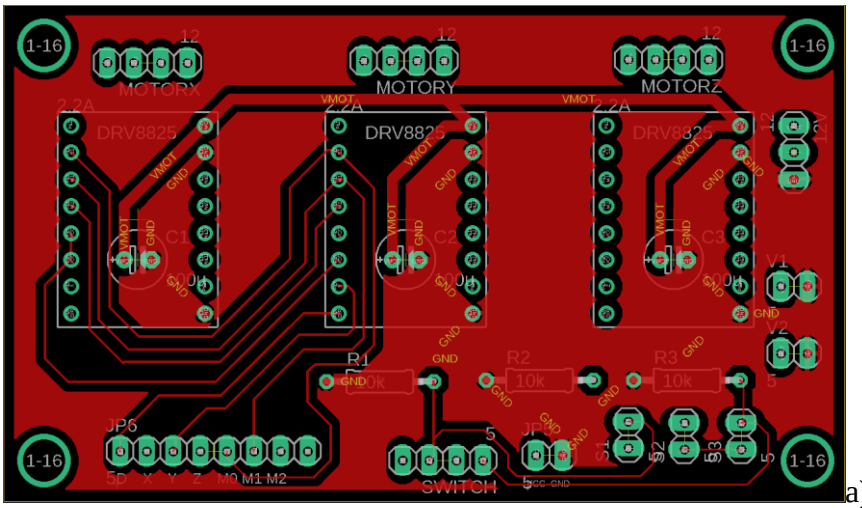
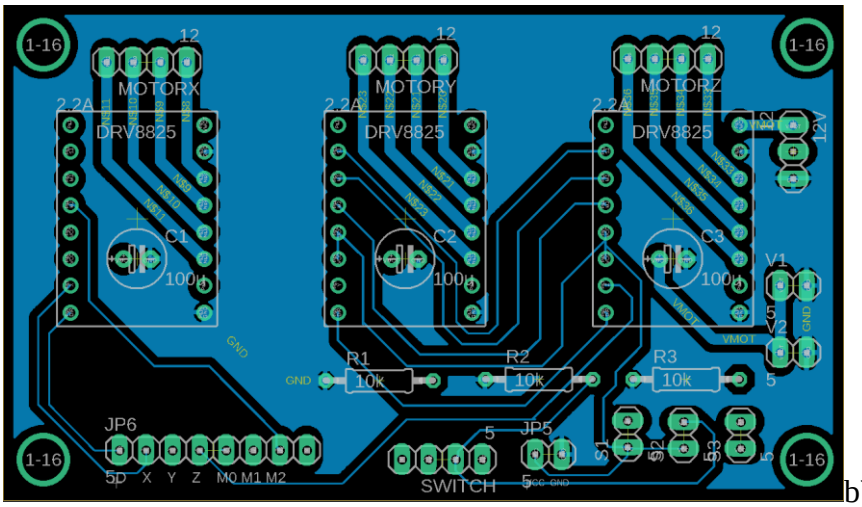


Figura 13: *Esquemático del circuito del Sistema de Movimiento (x,y,z)*



a)



b)

***Figura 14**: Diseño de PCB. Capa TOP (a), Capa BOTTOM (b)*

2.2 Interfaz de Usuario

Para el control del sistema se propuso desarrollar 3 programas por separado en Labview 2013 para manipular cada uno de los instrumentos involucrados en el proceso. Estos instrumentos son el Laser NL300, el osciloscopio Tektronix TDS 5054B, y una tarjeta de adquisición de datos NIUSB – 6001 de National Instruments. En el proceso se optó por ocupar la versión de 32 bits de Labview debido a que los controladores del láser no son compatibles con el sistema de 64 bits.

2.2.1 Programa de control de motores a PAP

El primer programa contempla el control de los motores PAP con una tarjeta de adquisición de datos mandando un tren de pulsos al pin STEP al controlador de motor PAP para controlar

el movimiento, la velocidad modificando el ancho del pulso y una señal digital al pin DIR para controlar la dirección del motor PAP.

Como se puede apreciar en el esquemático (*Figura 15*) del programa de control de motores PAP, lo primero que se hizo fue definir las salidas del puerto de la tarjeta gráfica, después se activó el envío de datos para posteriormente ejecutar la rutina del programa, en donde hemos agregado un ciclo for con la intensión de controlar el número de pulsos. Dentro de él se modifican los estados de las entradas (P0.1, P0.2 y P0.3), de HIGH a LOW, con el propósito de crear una señal cuadrada en los tres pines por separado, la cual será enviada a las entradas STEP del controlador, para generar el movimiento del motor PAP. A su vez podrán modificarse la entrada (P0.0) en cada repetición del ciclo para modificar la dirección del motor PAP, enviando la señal a DIR, al controlador. El ciclo también contiene un WAIT que puede ser modificado para variar la velocidad de los motores PAP. Debido al arreglo que se tiene para enviar las señales a DIR y STEP, los motores serán controlados uno por uno, evitando que trabajen simultáneamente, esto no perjudica en nada el proceso, posteriormente a que finalice la rutina se comprobaran si no hubo errores en la tarjeta de datos para poder finalizar nuestro programa.

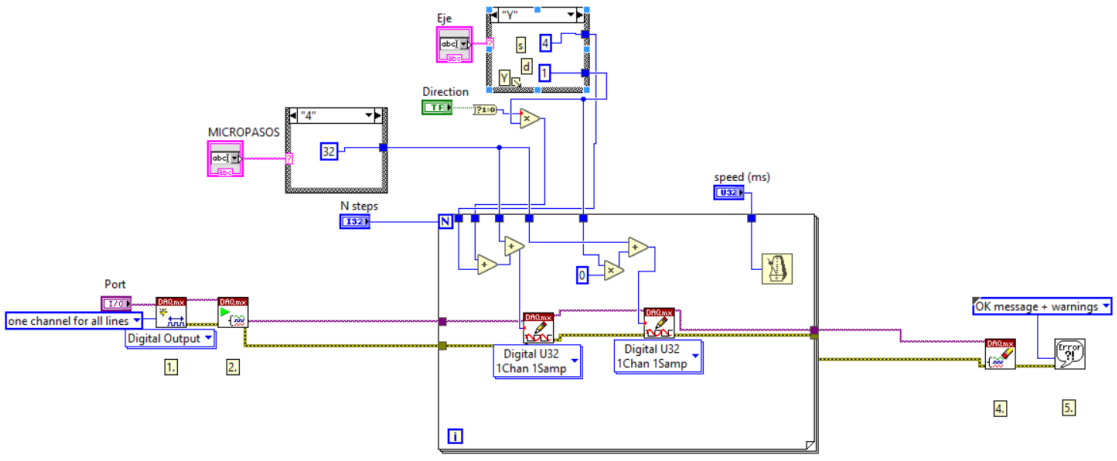


Figura 15: *Esquemático del Programa de control de motores PAP.*

2.2.1.1 Programa de inicio de motores PAP

Posteriormente se creó un subprograma de inicio, (Figura 16) tomando como referencia el primer programa en donde se recibe la señal de tres *push botton* para ubicar a los motores

PAP en las coordenadas (0,0,0), así tener considerado moverse solo los 25 mm que tienen como permitido las platinas y no forzar ni dañar a los motores PAP y ni el sistema mecánico de las platinas. Esto se logra cambiando el ciclo *for* por ciclo *white* condicionándolo por las señales digital de los *push botton*, indicando que se detenga los motores cuando detecte un HIGH en la señal.

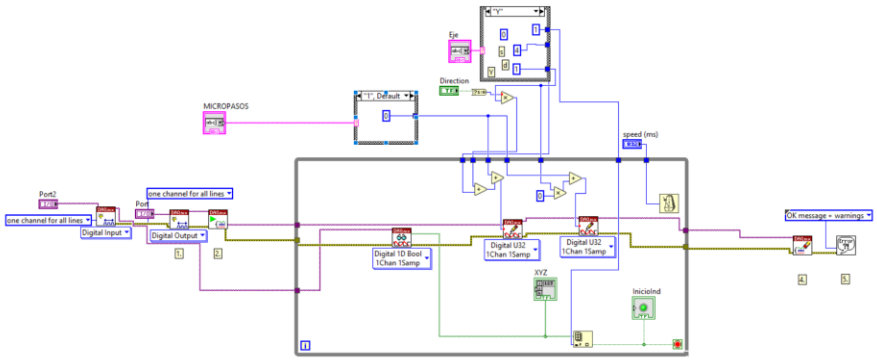


Figura 16: *Esquematico del Programa de Inicio de motores PAP.*

Las entradas del subprograma de inicio, se define el puerto 0 de la tarjeta DAQ como salidas para enviar las señales de los motores PAP, y el puerto 1 se define como entradas a los primeros 3 pines (P1.0, P1.1, P1.2), que adquieran las señales de los *push botton* para identificar la posición de inicio de los motores PAP. También se le agregaron indicadores a los mismos para ver su estado lógico.

2.2.2 Programa de control del laser

El siguiente programa está enfocado en controlar la fluencia de láser, la frecuencia de repetición, el número de pulsos y el ancho de los pulsos enviados a la lampara. Para ello se usó el programa de prueba, los controladores y el manual, proporcionados por el fabricante Ekspla.

Primero reconocemos nuestro dispositivo con el subprograma RCConnect (*Figura 17*) que llama a la función rcConnect(int32_t arg1, int32_t arg2) de la librería de REMOTECONTROL.dll, en donde se especifica el tipo de conexión (RS232 o CAN) y el puerto COM en el que está conectado el instrumento, para su posterior identificación o envió de un error en caso de no lograrse una buena comunicación. (*Apéndice A*)

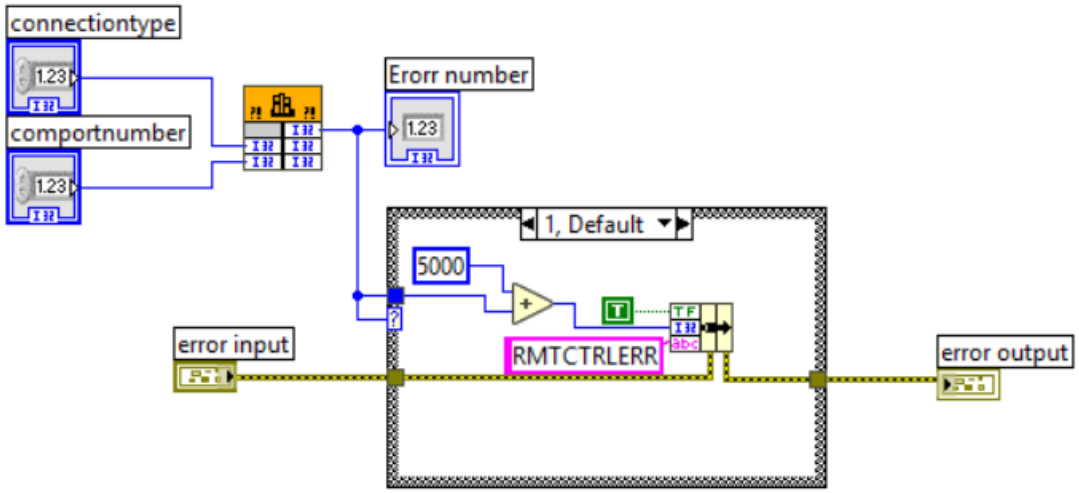


***Figura 17**: Subprograma rcConect*

Posteriormente se inicializa el láser con el subprograma Initializarlaser.vi, en donde se actualiza el archivo REMOTECONTROL.INI, que contiene la lista de todos los registros de lectura y escritura del Laser NL300. Algunos de los registros con sus respetivos valores que controlan los módulos internos del láser se encuentran en la tabla de la *figura 18*.

Registro	Valor
State	SLEEP, STOP, RUN y Fault
Outputlevel	OFF, Adjustm y Max
Continuous/ Burst mode/Trigger	Continuous, Burst y Trigger
Burst length, pulses	
SynOut delay	
Divider, every Nth pulse	
Set cooling temperarture	
Lamp pulse counter	
Repetition Rate	
Best EO delay	
Adjusment EO delay	

***Figura 18:** Tabla de inicialización del laser.*

Con la ayuda del subprograma EscribirString.vi (Figura 19), y de los registros obtenidos anteriormente, se mandan los comandos en formato string. El EscribirString.vi incluye la librería dinámica REMOTECONTROL.dll, en donde se le envían valores a las variables de la función rcSetRegFromStringA(const char *devname, const char *regname, const char *value, int forcelimits). La variable devname se le envía NL30x:8 que es el nombre del láser, a regname se le pueden agregar los registros State, Outputlevel, Continuous/Burst mode/ Trigger y Burst length, pulses encontrados en la tabla de inicialización del láser.

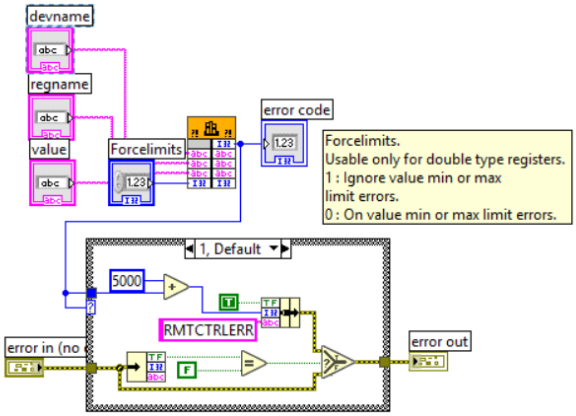


Figura 19: *Subprograma EscribirString.vi*

Para mandar los comandos restantes que no sean de consulta usamos el subprograma Programar.vi (*Figura 20*), el cual incluye la librería dinámica REMOTECONTROL.dll en donde se le envían los valores a la función rcSetRegNVFromDouble(const CStr arg1, const CStr arg2, double arg3); en donde arg1 será el modelo del instrumento, arg2 el registro a ejecutar y arg3 el valor del registro en formato double.

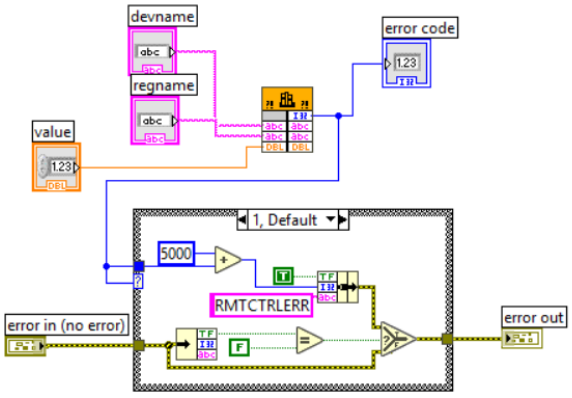


Figura 20: *Subprograma Programar.vi*

Para leer los datos del láser se ocupa el subprograma LeerString.vi (*Figura 21*) el cual usa la librería REMOTECONTROL.dll para llamar a la función rcGetRegAsString(const CStr arg1, const CStr arg2, CStr arg3, int32_t arg4, int32_t arg5, int32_t *arg6), donde especificamos el nombre del instrumento(arg1) y definimos el registro (arg2) del que deseamos saber su valor, además de definir el tamaño del string, el tiempo de espera y el tiempo en el que fue recibida la orden.

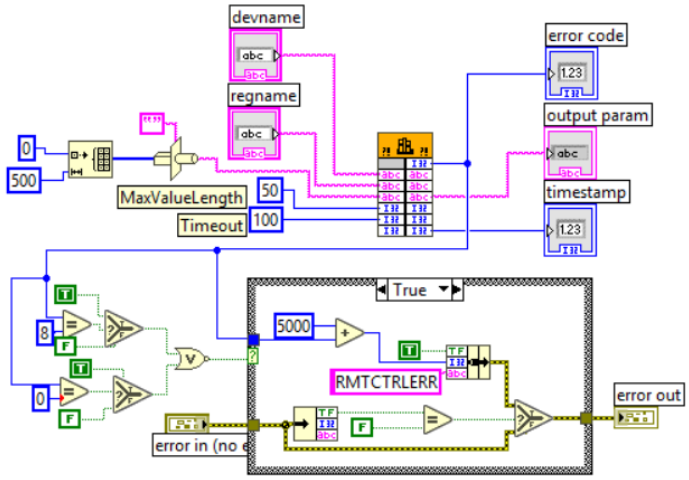


Figura 21: Subprograma LeerString.vi

Se desarrollo una aplicación para el control general del láser donde las variables que estamos controlando son: el tipo de conexión, el puerto COM, el modo de disparo, el divisor de frecuencia, el número de los pulsos, el Best EO delay, y el estado del láser, además de consultar el número total de pulsos de la lampara (*Figura 22*).

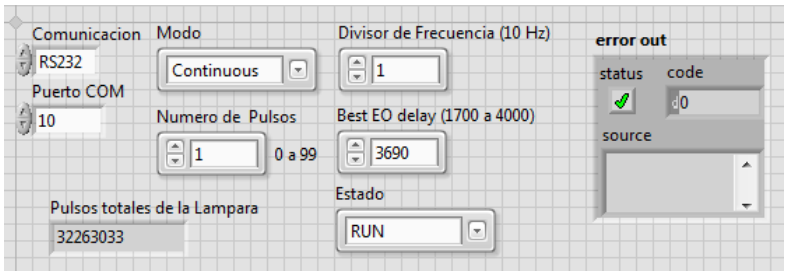


Figura 22: *Aplicación del laser*

Para desarrollar la aplicación del láser NL300 primero se indica el puerto en que está conectado el instrumento y el tipo de comunicación. Posteriormente llamamos a

Inicializarlaser.vi para generar el archivo de inicio REMOTECONTROL.INI que contiene el nombre de cada uno de los registros del instrumento. Después se escogerá el modo de trabajo del láser (Continuous o Burst), en el caso del modo continuo los pulsos empezaran a generarse a 10 Hz aproximadamente 5 segundos después de que se envió la instrucción RUN y se encendió la lampara y parará hasta que le mandemos la instrucción de STOP. En caso contrario con el modo Burst al mandar la instrucción RUN solo activaremos la lampara y no generamos ningún pulso hasta enviar la instrucción Trigger burst al registro Continuous / Burst mode / Trigger burst. La cantidad de pulsos dependerá del registro Burst length, pulses, el cual solo podrá generar como máximo 100 pulsos seguidos. Posteriormente se configurará el divisor de frecuencia considerando que la frecuencia de la lámpara es de 10 Hz. Después configuraremos la energía del pulso con la ayuda del registro Best EO Delay, el cual modifica el retardo de disparo entre la lampara y el Q-switch del láser. Finalmente mandamos la instrucción de RUN o STOP al registro State, para generar o detener los pulsos laser (*Figura 23*).

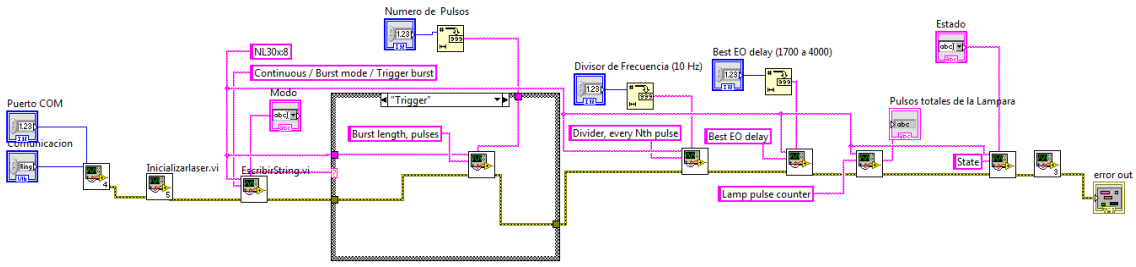


Figura 23: *Subprograma de la aplicación del láser.*

2.2.3 Programa de control del Osciloscopio

El desarrollo de esta etapa consistió en la generación de subprogramas que inicializan, configuran y adquieren las señales detectadas por el osciloscopio. Tales subprogramas se desarrollaron con base en la información del manual del usuario y la programación del equipo TDS5000, para posteriormente generar los subprogramas que inicializan, configuran y adquieren las señales detectadas por el osciloscopio.

Para la generación del programa, se optó en ocupar la comunicación Ethernet de 100 Mbps con la ayuda de un router linksys E1200 de Cisco, debido a que es la comunicación más

rápida que soporta nuestra tarjeta del ordenador, esto nos reducirá un poco más el tiempo para la adquisión de datos. El osciloscopio se controla mediante comandos específicos encontrados en el manual del programador, en donde se especifica el formato, funciones y limitaciones de tales comandos. Para poder realizar la comunicación de la interfaz de ethernet se ocupó los controladores NI-488.2 y NIVISA de National Instruments.

2.2.3.1 Configuración de la Interfaz LAN

El sistema operativo con el que se está trabajando es Windows 10, el cual tiene desactivado el protocolo de redes compartidas de manera predeterminada. Para activarlo vamos a Panel de Control: Programas: Activar o desactivar las características de Windows: Compatibilidad con el protocolo para compartir archivos SMB 1.0/ CIFS y marcamos las casillas:

- Cliente SMB 1.0/CIFS
- Eliminación Automática de SMB 1.0/CIFS
- Servidor SMB 1.0/ CIFS

Con estas configuraciones se podrá conectar a la red local, que se creó con el osciloscopio y el modem.

También es necesario configurar el osciloscopio para la transferencia de datos de la interfaz del ethernet de manera manual para ello hay que ir a la barra del menú en la sección de utilidades dar clic izquierdo en LAN SERVER STATUS y aparecerá una ventana del lado derecho que dice VXI-11 en esta parte solo le damos en la opción de STAR para habilitar la transferencia de datos, este procedimiento es necesario realizarlo cada vez que se enciende el Osciloscopio.

Después hay que configurar el Osciloscopio abriendo el programa NI-MAX, en la parte izquierda de la pantalla desplegamos las opciones de Devices and Interfaces, clic derecho en Network Devices y seleccionamos Create New VISA TCP/ IP Resource y seleccionamos la opción Auto-detect of LAN Instrument y nos aparecerá el canal del Osciloscopio y le damos en finalizar para terminar las configuraciones.

2.2.3.2 Programa de Inicio del Osciloscopio

El osciloscopio se programa con dos tipos de instrucciones; los comandos que modifican las configuraciones o indican al instrumento una acción específica a realizar y las consultas que

hacen que el instrumento devuelva datos e información de estado. Las consultas se diferencian por un "**?**" de los comandos, por ejemplo **ADQuire:MODe SAMple** es un comando que configura el modo de adquisición de datos en este caso el modo SAMPLE donde solo se adquiere una señal cuando se activa el trigger y **ADQuire:MODe?** es una consulta ya que el instrumento te envía el modo de adquisición de datos en el que está configurado. Cada vez que manda un comando principal se inicia con "**:**" y para concatenar varios comandos se ocupan los "**;**". En la figura 24 se aprecia el envío de los comandos del instrumento, estos comandos se envían en un formato string con ayuda del subprograma de VISA, posteriormente esperamos un momento a que se haya concretado la acción, para después verificar si no hubo un error en la transferencia de datos. En esencia todos los subprogramas mostrados para el control del osciloscopio cumplen la misma función, mandar un paquete de datos, los cuales incluyen los comandos o consultas para realizar ciertas acciones en el instrumento y monitorizar su estado.

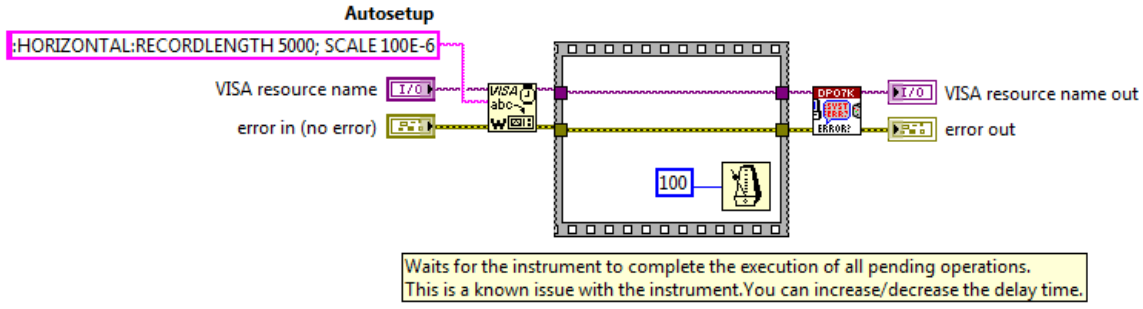


Figura 24: *Programa de configuraciones del eje horizontal del Osciloscopio.*

Se desarrollo un programa de inicio (*Figura 25*), en que se encuentran varios subprogramas que envían comandos que preparan al instrumento para la adquisión de datos. Primero inicializamos el instrumento con Initialize.vi (*Apéndice B*), indicando el canal de comunicación el cual es el encargado de identificar el modelo del instrumento y verifica si no se ha generado un error en la comunicación. Después indicamos el tipo de adquisión SAMPLE a tiempo real con el subprograma Configure Acquisition.vi (*Apéndice B*), el cual manda el comando **ACQuire:MODe SAMple**. Posteriormente se mandan a llamar el subprograma Configure Channel.vi (*Apéndice B*) en donde se configuran y activan los canales CH1, CH2 y CH3 del instrumento, con una impedancia de 1Mohms, con acoplamiento DC para la detección del pulso del láser y AC para las señales captada por los

fotodiodos, y el ajuste del rango de la amplitud. Después se configuro el tipo de adquisición de las muestras con el subprograma Configure Continuous Acquisition.vi (*Apéndice B*) del modo SEQUENCE, posteriormente se llamó al subprograma Timebase.vi (*Apéndice B*) donde se configuro el eje del tiempo del instrumento controlando el tiempo base, la posición de la señal en la pantalla y la cantidad de muestras de la señal. También se configuró el trigger con el subprograma Configure Trigger(Edge).vi (*Apéndice B*) en donde se habilitó el trigger en el canal 1, con un nivel de 8 volts, de tipo EDGE RISE.

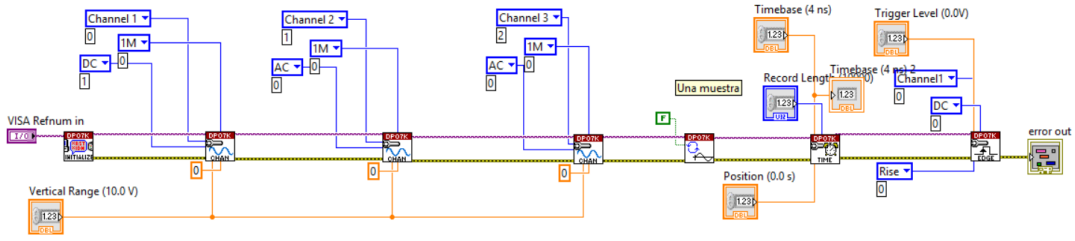


Figura 25: *Programa de inicialización del osciloscopio.*

La interfaz del inicio del osciloscopio del programa final al usuario se le da la tarea de buscar el canal donde se encuentra conectado el instrumento, además se le permite controlar el número de muestras a adquirir y configurar los rangos de amplitud y tiempo del instrumento (*Figura 26*)

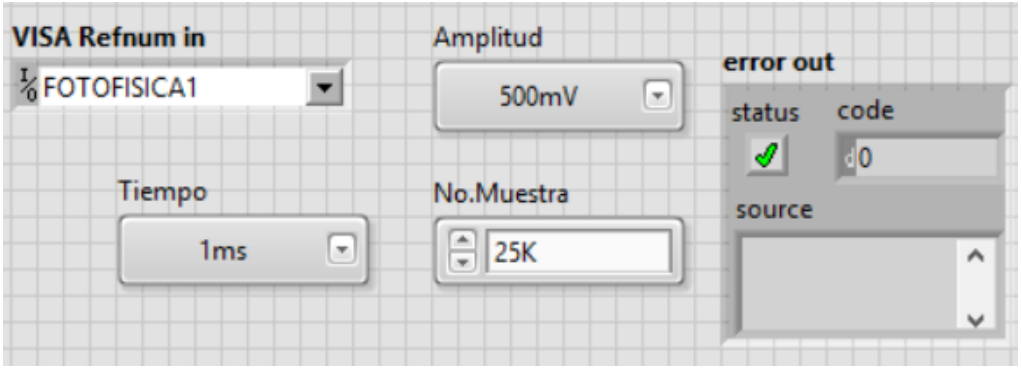


Figura 26: *Aplicación de las configuraciones de inicio del osciloscopio*

2.2.4 Programa de control de la técnica de Irradiación por pulso laser

En el segundo apartado de la aplicación final se programaron todos los instrumentos para realizar una rutina que automatiza la técnica de irradiación por pulsos laser, en donde se

configura la cantidad de pulsos irradiados en la muestra en una posición fija, el intervalo de movimiento (vertical y horizontal), velocidad y posición de las platinas.

Lo primero que hace esta rutina es ubicar los motores PAP en su posición inicial. Después se activa el láser para irradiar una cantidad de n pulsos y capturar cada una de las señales emitidas a los fotodiodos en cada pulso. Posteriormente se moverán los motores en el eje X a la siguiente posición a irradiar de la muestra. Se repetirá el proceso N veces hasta cubrir el ancho de la muestra, y bajará M cantidad de veces de manera vertical repitiendo el mismo proceso hasta irradiar toda la superficie de la muestra (*Figura 27*).

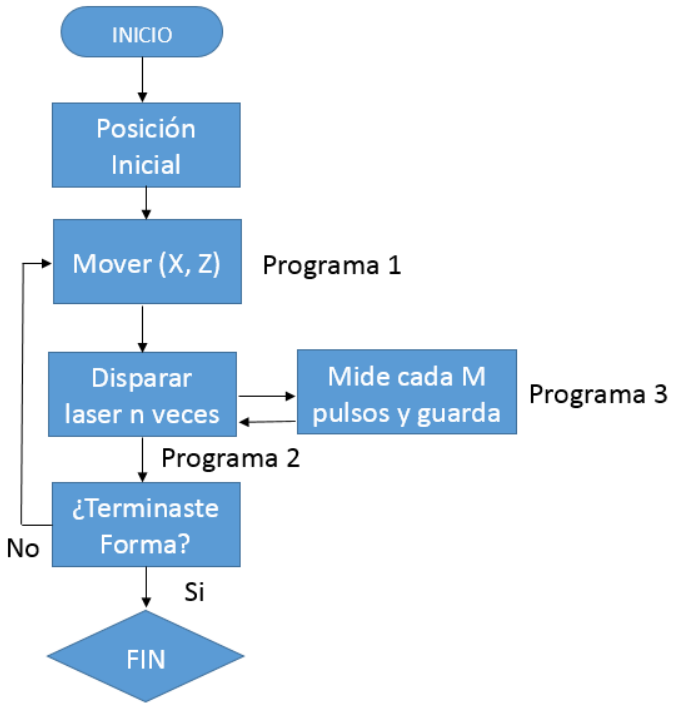


Figura 27: *Diagrama de flujo*

2.2.4.1 Detección de un Pulso Laser

Para generar la primera parte de la rutina de irradiación laser, se creó un programa capaz de detectar el pulso del láser, pero antes se creó el subprograma Triggerestado.vi (*Apéndice B*) para el comando de consulta **:Trigger:STATE?** Con éste le enviamos la instrucción y leemos 3 bytes del buffer, estos pueden indicar los siguientes estados del instrumento:

ARMed: Esto indica que el instrumento está adquiriendo información previa al encendido.

AUTo: Esto indica que el instrumento está en el modo automático y adquiere datos incluso en ausencia del trigger.

PARcial: Esto indica que se ha producido el disparo A y que el instrumento está esperando que se produzca el disparo B.

READy: Esto indica que se ha adquirido toda la información previa al arranque y que el instrumento está listo para aceptar un disparador.

SAVe: Esto indica que el instrumento está en modo de guardar y no está adquiriendo datos.

TRIGger: Esto indica que el instrumento se activó y está adquiriendo la información del post-disparo.

En el estado **SAVe** se indica que el trigger ya fue activado y la onda ya fue adquirida por el instrumento, por el subprograma Triggerestado.vi (Apendice).

Para el programa de la detección del pulso laser (Figura 28), primero se encendió la lámpara y agregamos un delay para después disparar el pulso laser. Posteriormente se creó un ciclo *while* para esperar el pulso. Condicionando la salida del ciclo *while* con la respuesta del instrumento adquirida por el subprograma Triggerestado.vi, si los datos leídos son un **SAV** termina el ciclo, para posteriormente detener el láser.

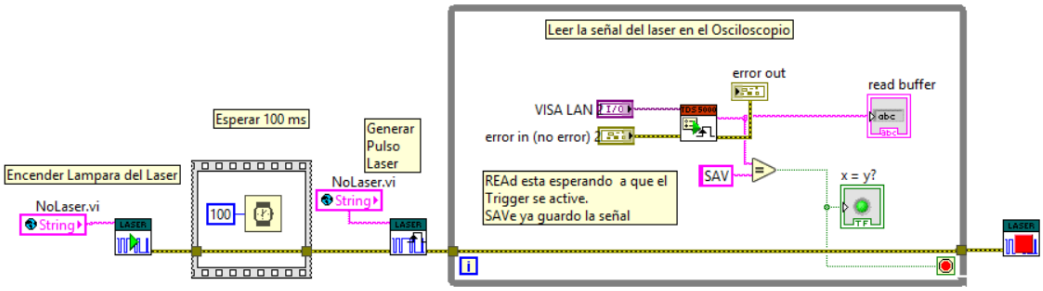


Figura 28: *Programa de Inicio y detección de un pulso laser.*

Después de que se haya detectado el pulso seguirá la adquisición de datos de la señal, debido a que el trigger del osciloscopio ya fue activado por el pulso del láser. En esta parte se adquieren 3 señales (del láser, de transmitancia y de deflectometria).

2.2.4.2 *Sincronización de pulsos laser con adquisición de datos*

El siguiente paso fue agregar la adquisición de datos del pulso y de las señales de transmitancia y deflectometria (*Figura 29*), tomando el programa anterior, agregamos el subprograma initiate.vi, el cual contiene el comando **:ACQ:STATE ON** el cual activa la adquisión de datos, y el instrumento queda en espera de la señal del trigger, este subprograma se agregó antes de revisar el estado del trigger, debido a que estamos en modo SAMPLE en donde solo adquirimos una muestra de la señal y la adquisición se detiene, esto nos permitirá tomar las 3 señales de cada N cantidad de pulsos, al finalizar el ciclo ***while***, se agregó una estructura ***case***, la cual la condicionamos con la respuesta del subprograma Triggerestado.vi para evitar que el programa genere archivos demás en la adquisión de datos, antes de que se active el trigger. El case contiene el subprograma Read(Multiple Waveforms) encargado de la adquisición de datos de las tres señales. El ***case*** y ***while*** están encerrados en un ciclo ***for*** el cual indicara la cantidad de pulsos que serán emitidos en cada instrucción.

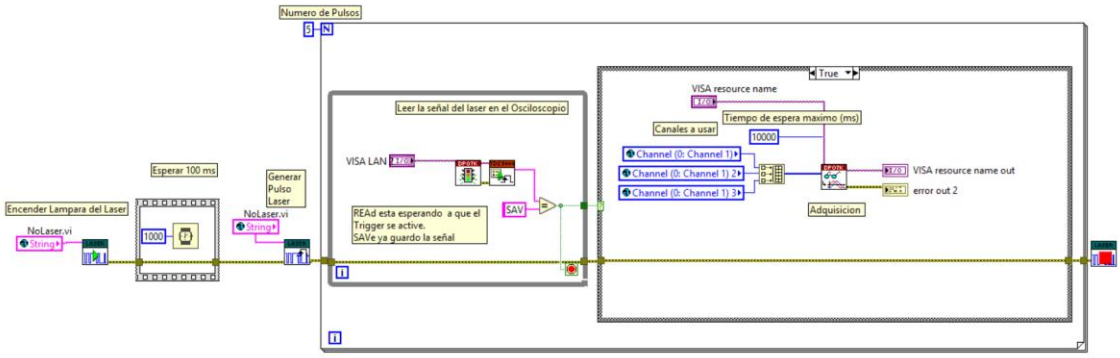


Figura 29: *Detección y adquisición de N pulsos*

2.2.4.3 *Adquisión de una forma de onda*

Para la adquisión de las tres ondas se ocupó el subvi Read(Multiple Waveforms).vi (*Figura 30*) en donde se controla el tiempo máximo de adquisión, y se verifica que ninguna otra acción, se esté ejecutando en el osciloscopio gracias al subvi OPC (Operation Complete) el cual manda un bit al registro de errores para corroborar el estado de la operación. El osciloscopio solo puede obtener una onda a la vez por ello el subvi Fetch.vi fue puesto adentro de un ciclo for para recolectar una a una las señales de los tres canales.

Al inicio del programa se inicia la adquisición de datos, esta se ejecutará cuando el trigger lo indique el cual se configuro anteriormente en las configuraciones de inicio del osciloscopio.

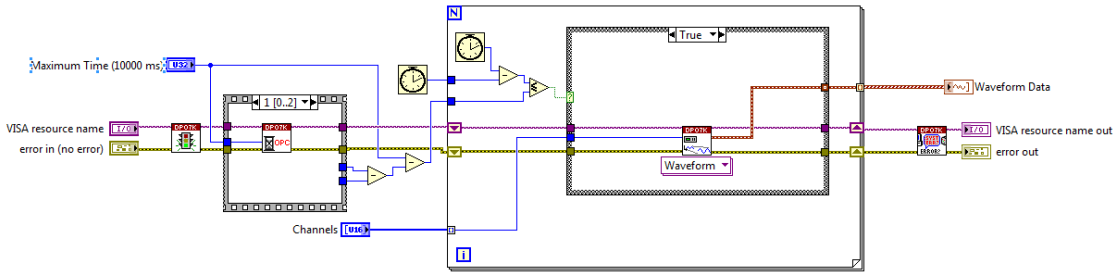


Figura 30: *Programa Waveform (multiple)*

En él se encuentra el subvi Fetch.vi el cual es el encargado de realizar la adquisión de datos de una onda y guardarla. Para adquirir la forma de onda es necesario ubicar el canal (CH1, CH2, CH3 y CH4), indicar el modo de adquisición (Sample) y el formato, es decir; la configuración del número de byte por muestra de la onda (2 bytes). Antes de realizar la adquisición es necesario especificar la cantidad de muestras a adquirir. Para después iniciar la transferencia de datos con el comando **CURV?**, donde hemos configurado el buffer para que reciba 1 millón de muestras en cada paquete y finalmente se guarden en el ordenador en un archivo ".txt".

2.2.4.4 Lectura de los archivos de las señales

Los archivos txt, tienen un formato particular para agilizar la adquisición de ondas, los datos los guarda de la siguiente manera en el orden que se presentan en la siguiente lista:

- XZero;XIncrement;YMultiplier;YOffset;YZero
- Width;Date Length
- Waveform Data

Las muestras de la onda se guardan en formato ASCII porque no consideran las especificaciones de los ejes X y Y, por tal motivo para utilizar la onda es necesario darle formato, a todas las muestras de la señal. Los datos de la onda que no describen los ejes solo es necesario sumarles YZero, restarle YOffset y multiplicarlo por YMultiplier para obtener el eje Y con formato, mientras para el eje X solo tienes que empezar a darle valor desde XZero e ir incrementanto Xincrement hasta que se acaben las muestras. El fabricante ya nos

proporciona un programa para traducir los archivos de señales solo basta con abrir el archivo y ubicarlo.

Después de haber obtenido todos los datos de las señales y acabado de irradiar parte de la muestra, el programa les indica a los motores que muevan a la platina del eje x a la siguiente posición, para posteriormente repetir el mismo proceso ya mencionado, hasta acabar con toda la sección del eje x, se moverá la platina del eje z para repetir todo el proceso hasta irradiar toda la muestra por completo para finalizar la rutina (*Figura 31*).

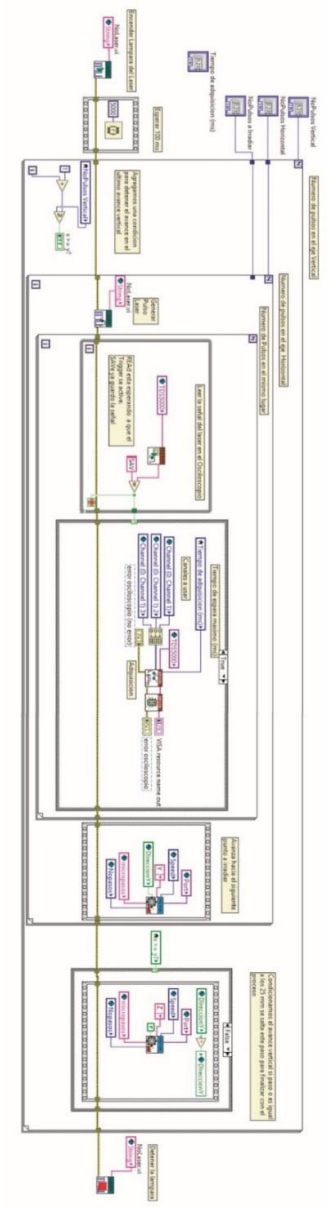


Figura 31: *Rutina de ablación laser*

CAPITULO 3: MEDICIONES Y RESULTADOS

3.1 Aplicación de Usuario

Para la parte final del proyecto se conjuntaron todos los programas realizados, en una sola aplicación que es capaz de controlar cada uno de los instrumentos y hacer de manera automatizada todos los procesos de la técnica de irradiación laser.

La aplicación esta divida en tres bloques. El primero de ellos realiza todas las configuraciones iniciales de los instrumentos (*Figura 32*), el segundo bloque incluye la rutina, que automatiza el proceso de irradiación laser (*Figura 33*), y el tercer bloque controla el modo manual del láser NL300 y los motores PAP. (*Figura 34*)

En la parte de inicio, inicializamos la comunicación de los instrumentos NL300 y el TDS-5054B, controlado por unos botones ubicados en la parte inferior de cada sección. Con el botón enviar mandamos las configuraciones iniciales de cada instrumento para realizar el proceso de irradiación laser. Para verificar que la conexión se realizó de manera adecuada se agregaron indicadores de error para cada uno de los instrumentos.

Para el caso de los motores es necesario definir los puertos de entrada y salida de la tarjeta de adquisición de datos por el usuario, y en esta parte solo se pueden enviar los motores a la posición de inicio.

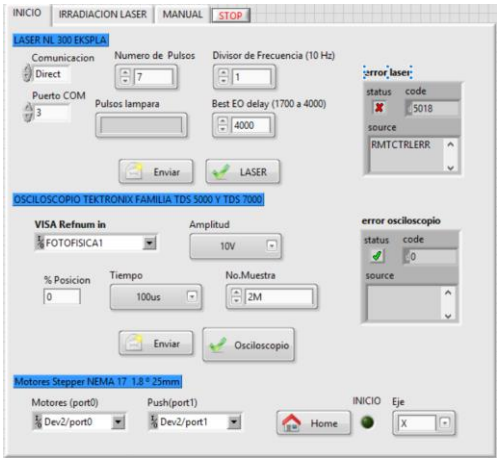


Figura 32: *Inicio (Aplicación de Usuario)*

En la parte de irradiación laser se puede modificar el número de pulsos para irradiar la misma zona, la cantidad de pulsos a disparar en los ejes horizontal y vertical, el control de la distancias entre pulso y pulso, control de la distancia máxima de cada eje (y,z), la resolución de los motores PAP y se consultara el estado del trigger del instrumento TDS-5054B, el tiempo máximo establecido para la adquisición de la señal, y la ubicación de cada una de las platinas (Figura 33).

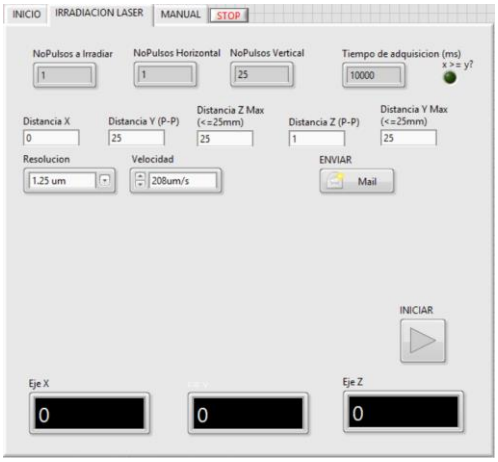


Figura 33: *Irradiación Laser (Aplicación de Usuario)*

En el modo manual se controla el láser a placer, siendo el usuario capaz de modificar modo de funcionamiento (Continuo o tren de pulsos), generar el pulso laser con un solo clic, modificar el tren de pulsos, el divisor de frecuencia y cambiar la energía del láser. En este apartado también se le da al usuario la opción de mover las platinas a gusto (*Figura 34*).

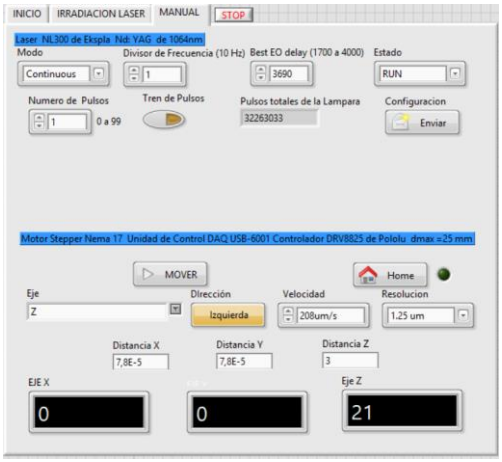


Figura 34: *Modo Manual (Aplicación de Usuario.)*

3.1.1 Resolución

Para calcular la resolución del sistema se tomó el recorrido de las platinas PTM1 por vuelta y dividió entre la cantidad de pulsos necesarios para mover el motor 360 grados logrando una resolución mínima de 0.08 µm aproximadamente (*Figura 35*)

Pasos	**1**	**2**	**4**	**8**	**16**	**32**
Distancia lineal	2,50E-06	1,25E-06	6,25E-07	3,13E-07	1,56E-07	7,81E-08

Figura 35: *Tabla de resolución de micropasos*

Se ha limitado al usuario a usar ciertas velocidades, debido a que los retardos entre pulso y pulso solo se pueden variar en una escala de milisegundos, debido a que Labview solo garantiza el retardo mínimo a un milisegundo. Las velocidades programadas se pueden ver en la *figura 36*, en donde ampliamos estas velocidades con la resolución de los micropasos, teniendo un catálogo de 35 velocidades que van desde 8.68 µm/s a los 2.5mm/s, tomando en cuenta que no se podrán adquirir todas las velocidades en cada resolución, las velocidades más altas se encontrarán en las resoluciones más bajas. Debido a que para nuestra aplicación solamente se requieren resoluciones de µm, se usarán velocidades superiores a los 10^{-4} m/s.

	Micropasos					
Retardo (ms)	1	2	4	8	16	32
1	2,50E-03	1,25E-03	6,25E-04	3,13E-04	1,56E-04	7,81E-05
2	1,25E-03	6,25E-04	3,13E-04	1,56E-04	7,81E-05	3,91E-05
3	8,33E-04	4,17E-04	2,08E-04	1,04E-04	5,21E-05	2,60E-05
4	6,25E-04	3,13E-04	1,56E-04	7,81E-05	3,91E-05	1,95E-05
5	5,00E-04	2,50E-04	1,25E-04	6,25E-05	3,13E-05	1,56E-05
6	4,17E-04	2,08E-04	1,04E-04	5,21E-05	2,60E-05	1,30E-05
7	3,57E-04	1,79E-04	8,93E-05	4,46E-05	2,23E-05	1,12E-05
8	3,13E-04	1,56E-04	7,81E-05	3,91E-05	1,95E-05	9,77E-06
9	2,78E-04	1,39E-04	6,94E-05	3,47E-05	1,74E-05	8,68E-06
10	2,50E-04	1,25E-04	6,25E-05	3,13E-05	1,56E-05	7,81E-06

Figura 36: *Tabla de velocidades del sistema de irradiación laser*

3.1.2 Tiempo de adquisición de datos

El tiempo de adquisición de datos es una de las variables más importantes para nuestro sistema, ya que nos ayudara a sincronizar cada una de las acciones con el pulso laser, y delimitar la frecuencia de trabajo de cada pulso para darle tiempo al instrumento de adquirir las señales.

La manera que definimos el tiempo máximo de adquisición de datos del osciloscopio TDS 5054B fue tomando en cuenta el divisor de frecuencia del láser NL300. La lampara flash que excita al laser trabaja a una frecuencia de 10 Hz, es decir, cada 0.1s se genera un pulso laser. El láser contiene un divisor de frecuencia que es un numero entero en el intervalo de 0 a 99, que divide la frecuencia de trabajo de la lampara flash sin modificarla, para cambiar la frecuencia del pulso emitido por el láser. Por ejemplo: si la frecuencia de trabajo de la lampara es de 10 Hz y configuramos el divisor de frecuencia a 1, la frecuencia del pulso laser será 10 Hz. Pero si tenemos un divisor de frecuencia de 2, la frecuencia del pulso laser será de 5Hz.

La idea fue adquirir las señales de 100 pulsos empezando por una frecuencia de 10Hz, verificando que el instrumento sea capaz de adquirir todas las señales en caso de que no sea así, quiere decir que el instrumento no estaba preparado para adquirir la siguiente señal en la frecuencia antes mencionada, es decir hay que disminuir la frecuencia de disparo para darle tiempo al instrumento de captar todas las señales, los resultados se pueden apreciar en la tabla de la *figura 37*.

Muestras	**500-50K**	**100K**	**250K**	**500K**	**1M**	**2M**
Divisor (10 Hz)	1	2	4	6	12	24

Figura 37: *Tabla de divisor de frecuencias*

En donde podemos notar que el instrumento puede adquirir menos de 50, 000 muestras en menos de 100 ms y lo máximo en que demora el instrumento es 2.4 s al recolectar 2, 000, 000 de muestras.

Considerando la velocidad del instrumento, se dispuso a medir la adquisión de 100 pulsos, usando los subprogramas creados, y el reloj interno del ordenador. La medición abarca desde que captamos el pulso laser hasta el guardado de las señales en el ordenador. Se obtuvo la

gráfica de la *figura 38*, en donde se ve el tiempo de adquisición en función de la cantidad de las muestras.

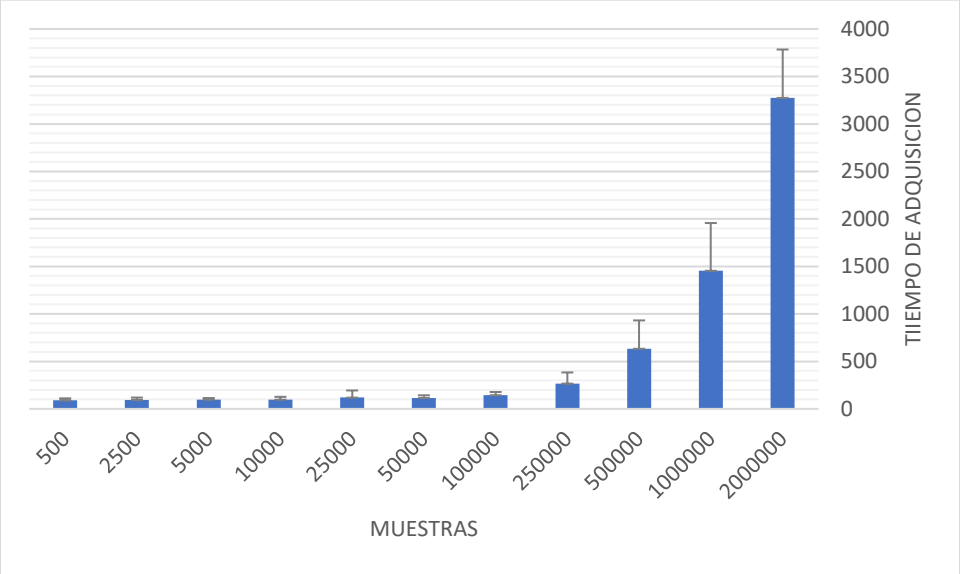


Figura 38: *Grafica de tiempos de adquisición de tres señales*

Estas mediciones nos permitirán definir el *divisor de frecuencias* del láser a ocupar dependiendo del tamaño de la señal. Como se puede ver en el gráfico, somos capaces de adquirir 3 señales de 500 muestras en menos de 100 µs sin usar el divisor de frecuencia, pero conforme se incrementa el tamaño de las señales es necesario usar el divisor de frecuencias. El tiempo máximo aproximadamente en captar 3 señales de 2,000,000 muestras es inferior a 4 s.

Por último, se armó todo el sistema experimental de irradiación (*figura 8 y figura 40*), para realizar una prueba con una película delgada de Al sobre un sustrato de vidrio (*figura 39*) y verificar que el proceso se está ejecutando correctamente. Esto consistió en alinear el haz de un láser He-Ne que apunta hacia la zona irradiada por el láser de Nd:YAG para medir transmitancia antes y después de irradiar con el pulso láser alta potencia. El otro haz, proveniente de un láser He-Ne, pasa rasante a la zona irradiada de la muestra para medir deflexión debida a la eyección de material y al cambio de temperatura cerca de la superficie, de modo que ambos coincidan con el área de irradiación del haz de alta potencia. El área de la zona irradiada está definida por el spot del láser de alta potencia, el cual depende de la distancia del lente y la muestra. Cabe señalar que los láseres He-Ne utilizados trabajan a una

longitud de onda de 633 nm, mientras que el Nd:YAG está trabajando a una longitud de onda de 355 nm.

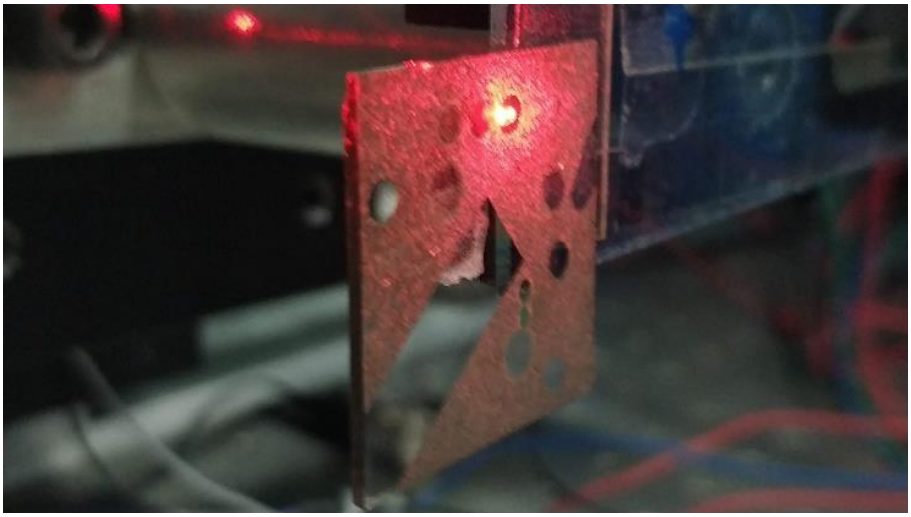


Figura 39: *Muestra de Prueba*

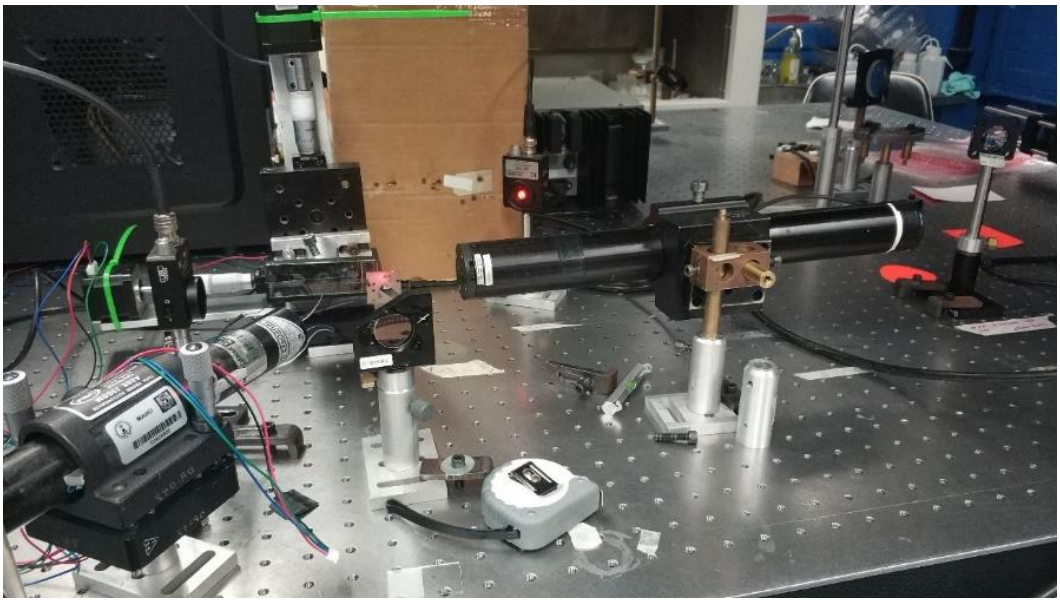
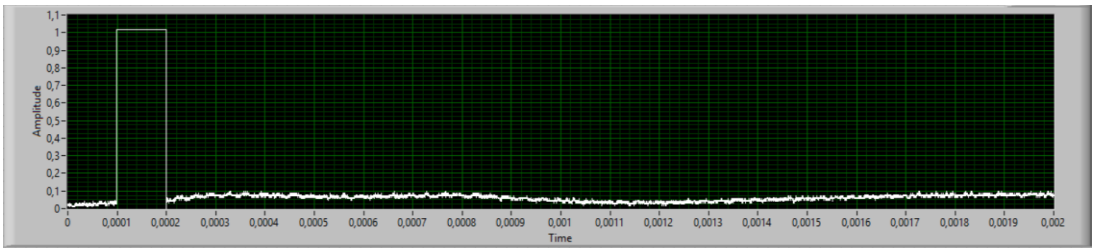


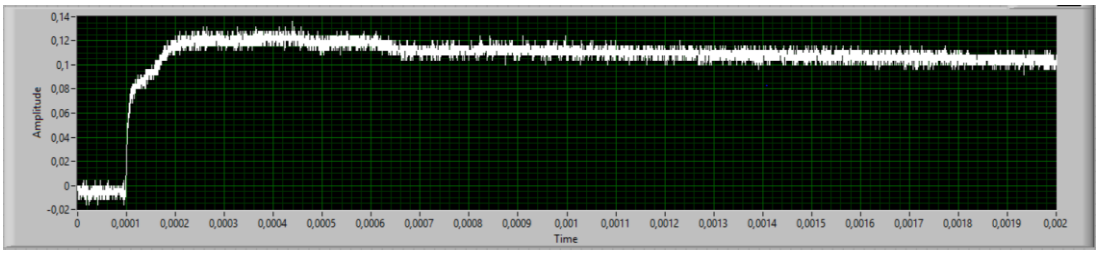
Figura 40: *Sistema de irradiación laser*

En la figura 39 se observan las señales adquiridas en el osciloscopio en cada pulso láser, en el inciso (a) se muestra la señal TTL de salida del láser Nd:YAG utilizada para disparar el osciloscopio. La subida de esta señal coincide con el pulso láser. En el inciso (b) se muestra

la señal de transmitancia, donde se puede apreciar que cuando llega el pulso láser, el fotodiodo detecta el aumento de la intensidad de luz transmitida que se debe a la remoción de la película delgada.



a)



b)

Figura 41: *Señal TTL del pulso laser(a), Señal de Transmitancia(b)*

CAPITULO 4: CONCLUSIONES Y PERSPECTIVAS

En este trabajo se automatizó un sistema de irradiación láser para la modificación de superficies. Se construyó un sistema de movimiento con platinas manuales acopladas a motores paso a paso. Este sistema puede ser aplicado en diferentes experimentos de laboratorio debido a su resolución de 0.08 µm, gracias a las propiedades de los motores PAP. Comparándolo con sistemas comerciales motorizados como el de Thor Labs [20], encontramos que éste usa servomotores continuos el cual alcanza una velocidad de 2.6 mm/s a una resolución mínima de 0.05 µm, mientras nuestro sistema cuenta con una velocidad máxima de 2.5 mm/s a una resolución mínima de 0.08 µm. Siendo nuestro sistema muy parecido a los comerciales, pero más económico y controlado con una interface que permite controlar simultáneamente un láser pulsado de alta potencia y un osciloscopio.

Se generó una aplicación para el control del láser NL300 de Ekspla, en el cual se pueden controlar todas las variables disponibles por el fabricante, como la frecuencia de repetición de la lámpara de flash, frecuencia de salida del pulso láser, energía por pulso, modo de oración (continuo o tren de pulsos), entre otras. También se desarrolló la comunicación con el osciloscopio TDS 5054B mediante ethernet con una velocidad de 100 Mbps, en donde podemos guardar 6 millones de muestras en menos de 4 s, además que el programa desarrollado puede ser usado con diferentes modelos de osciloscopios ya que es compatible con las familias TDS 5000 y la DPO 7000 de Tektronix.

El funcionamiento del sistema automatizado se comprobó con la irradiación laser de muestras con recubrimientos metálicos. Entre las ventajas que ofrece este sistema está la facilidad de configuración por el usuario, gracias a su abanico de opciones y su interfaz amigable, además de la automatización completa de la técnica.También podrá ser usado en técnicas que involucren el uso del osciloscopio, debido a que puede leer de forma automatizada las señales, o para aplicaciones que requieran conocer en todo momento la ubicación de la muestra, o solo quieran controlar el disparo del láser. Este sistema nos permite irradiar muestras con un área máxima de 25 x 25 mm^2, y configurar la energía del láser y su frecuencia de repetición. La densidad de energía (o fluencia) de irradiación también se controla de manera automatizada mediante el motor PAP del eje X, que varía la distancia

entre el lente y la muestra. Además, se guardan todas las mediciones en 4 canales del osciloscopio en tiempo real.

Cabe mencionar que el sistema solo fue probado para verificar la automatización correcta del mismo, con muestras de películas delgadas de Al adheridas a un portaobjetos para ver las señales de transmitancia y reflectancia. Por lo tanto, en este trabajo no se analizó ninguna de las muestras para verificar la formación de nanopartículas. Este análisis corresponderá al que use posteriormente el sistema de irradiación láser.

El sistema de irradiación laser puede ser mejorado mucho más, pero esto involucraría costos más altos, pero beneficiaria su manufactura. En el aspecto mecánico, se podría desarrollar un sistema de tres ejes más grande, pero de menor o igual resolución, ampliando el área de trabajo para su aplicación en superficies de grandes dimensiones. En la adquisición de señales, si se pretende mejorar la velocidad del proceso, es necesario de un mejor instrumento de adquisición de señales, que demore menos tiempo en la lectura de mayor cantidad de muestras, además de tener un buen equipo de cómputo, que realice las multitareas lo más rápido posible.

Referencias

[1] A. CANALES, Síntesis de nanopartículas de Au por irradiación láser (Licenciatura), Mexico: Universidad Nacional Autonoma de Mexico.

[2] A. C.-R. T. G.-F. M. V.-M. C. Sánchez-Aké, «Nanosecond pulsed laser nanostructuring of Au thin films: comparison between irradiation at low and atmospheric pressure,» *Appl. Surf. Sci., 403, 448-454,* 2017.

[3] O. K. I. G. B. J. A. R.-E. G. M. Sánchez-Aké T. Cesca, «Buffer-layer-assisted morphological manipulation of metal nanoparticle arrays by laser irradiation.,» *Applied Surface Science ,* nº 487, pp. 726-733, 2019.

[4] M. K. T. S. T. C. M. E. K. Z. B. H. P. Lorenz, «Dynamics of the laser-induced nanostructuring of thin metyal layers: experiment and theory,» *Mat. Res. Express 2, 026501,* 2015.

[5] D. Bäuerle, Laser processing and chemistry, New York: Springer 4th ed., 2011.

[6] J. C. S. S. S.J. Henley, «Pulsed-laser-induced nanoscale island formation in thin metal-on-oxide films,» *Phys. Rev. B 72, 195408,* 2005.

[7] S. H. S. H. K. G. A. A. E. S. E. K. S. S. J. Beliatis, «Organic solar cells with plasmonic layers formed by laser nanofabrication,» *Phys. Chem. Chem. Phys. 15, 8237,* 2013.

[8] R. P. C. A. S. R. P. L. D. J.-R. C.E. Rodríguez, «Plasmonic response and transformation mechanism upon single laser exposure of metal discontinuous films,» *Appl. Surf. Sci., 302, 32-36,* 2014.

[9] J. Z. L. B. M. E. K. Z. P. Lorenz, «Nanodrilling of fused silica using nanosecond laser irradiation,» *Appl. Surf. Sci. 351, 935-945,* 2015.

[10] C. G. M. E. L. B. K. Z. P. Lorenz, «Nanostructuring of fused silica assisted by laser-shaped metal triangles using nanosecond laser,» *Phys. Proc. 83, 62-73,* 2016.

[11] J. S. S. P. M. R. H. S. O. K. E. N. E. Matthias, «In-situ investigation of laser ablation of thin films,» *Thin Solid Films 254, 139-146,* 1995.

[12] P. M. Ossi, Advances in the Application of Lasers in Materials Science, Ed. Springer Series in Materials Science ISBN 978-3-319-96845-2, 2018.

[13] K. H. B. Chrisey, «Pulsed Laser Deposition of Thin Films,» *Ed. John Wiley & Sons Inc,* pp. 567-578, 1994.

[14] J. P. C. E. NIETO, «Películas delgadas: fabricación y aplicaciones,» *BOLETÍN DE LA SOCIEDAD ESPAÑOLA DE Ceramica y vidrio,* pp. 245-258, 1994.

[15] G. M. R. G. G. A. Ratautas K, «Nanoparticle formation after nanosecond-laser irradiation of thin gold films,» *Journal of Applied Physics 112:013108.1,* 2012.

[16] A. S.-L. J. T. Y. M. G. A. SALAZAR, «Aplicación de las técnicas fototérmicas al estudio de materiales,» *BOLETIN DE LA SOCIEDAD ESPAÑOLA DE Cerámica y Vidrio,* pp. 584-588, 2000.

[17] D. F. a. J. B. A. C. Boccara, «Thermo-optical spectroscopy: Detection by the ''mirage effect'',» *Journal of Applied Physics,* pp. 130-132, 1970.

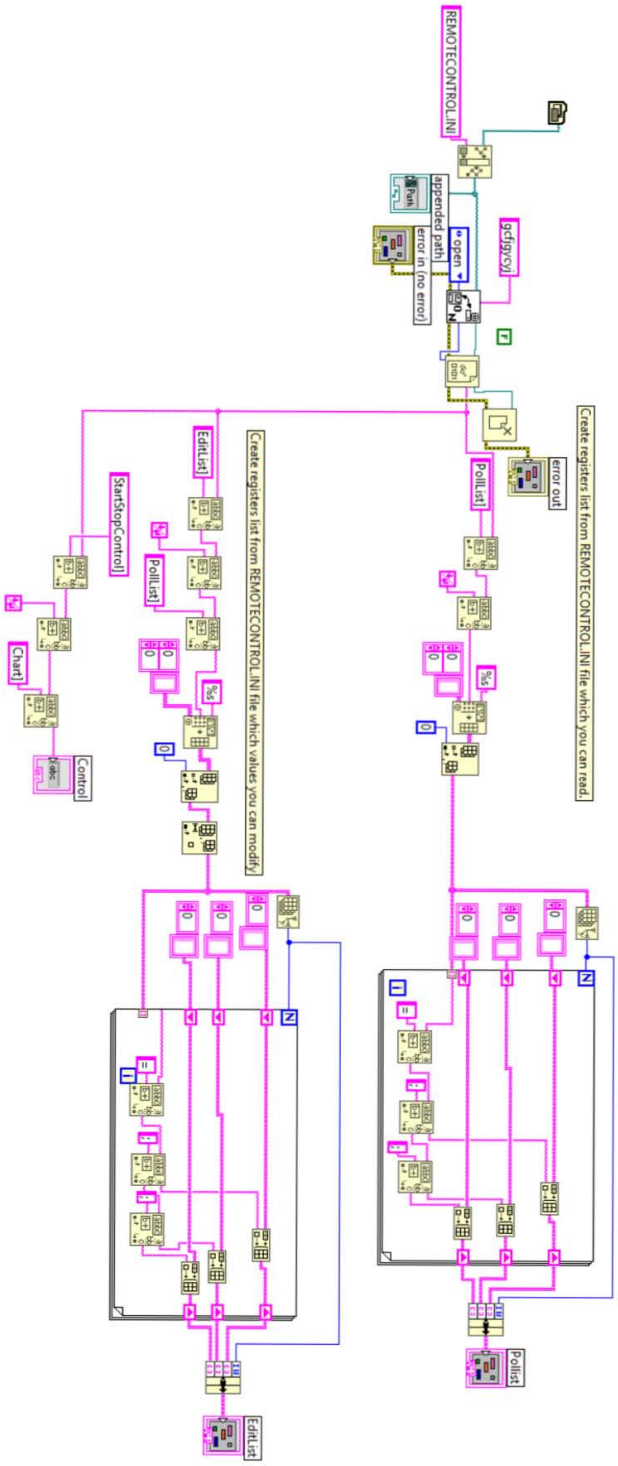
[18] «NL300 Series Laser Technical Description and User Manual,» *ThorLabs, Lithuania,* 2014.

[19] «Pololu Robotics and Electronics,» 2001–2019 . [En línea]. Available: https://www.pololu.com/.

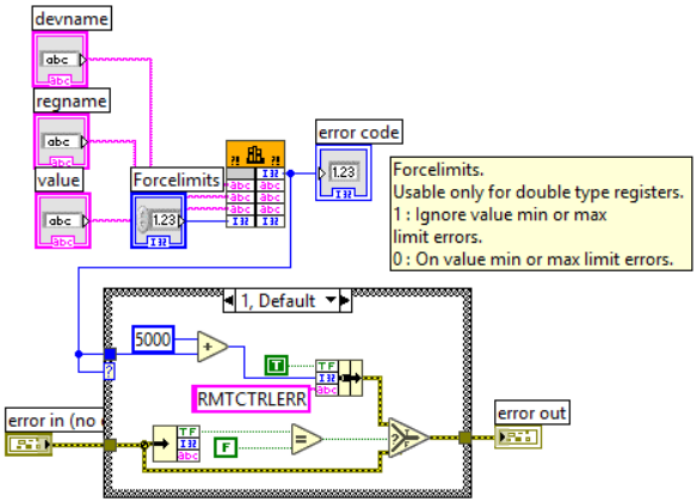
[20] Thor Labs, «PT1/M - 25 mm Translation Stage with Standard Micrometer,» https://www.thorlabs.com/.

APÉNDICE A: Subprogramas del instrumento NL300 de Ekspla

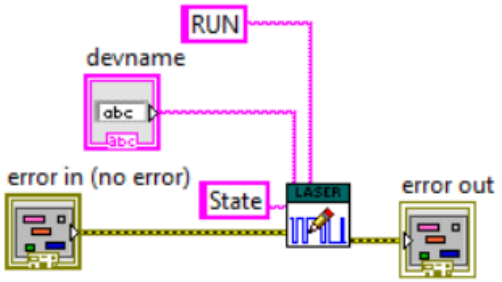
Inicializarlaser.vi



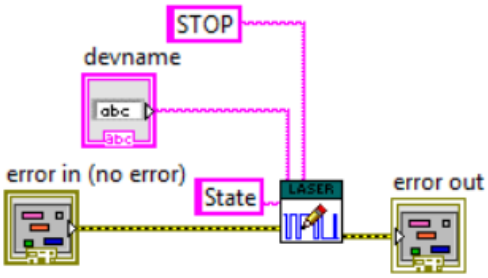
EscribirString.vi



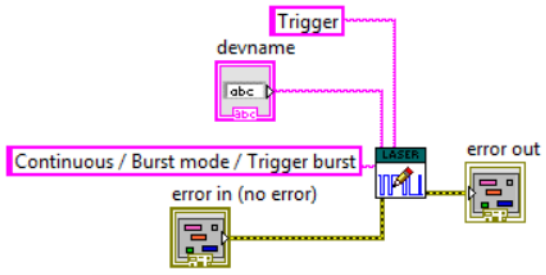
RUNlaser.vi



STOPlaser.vi

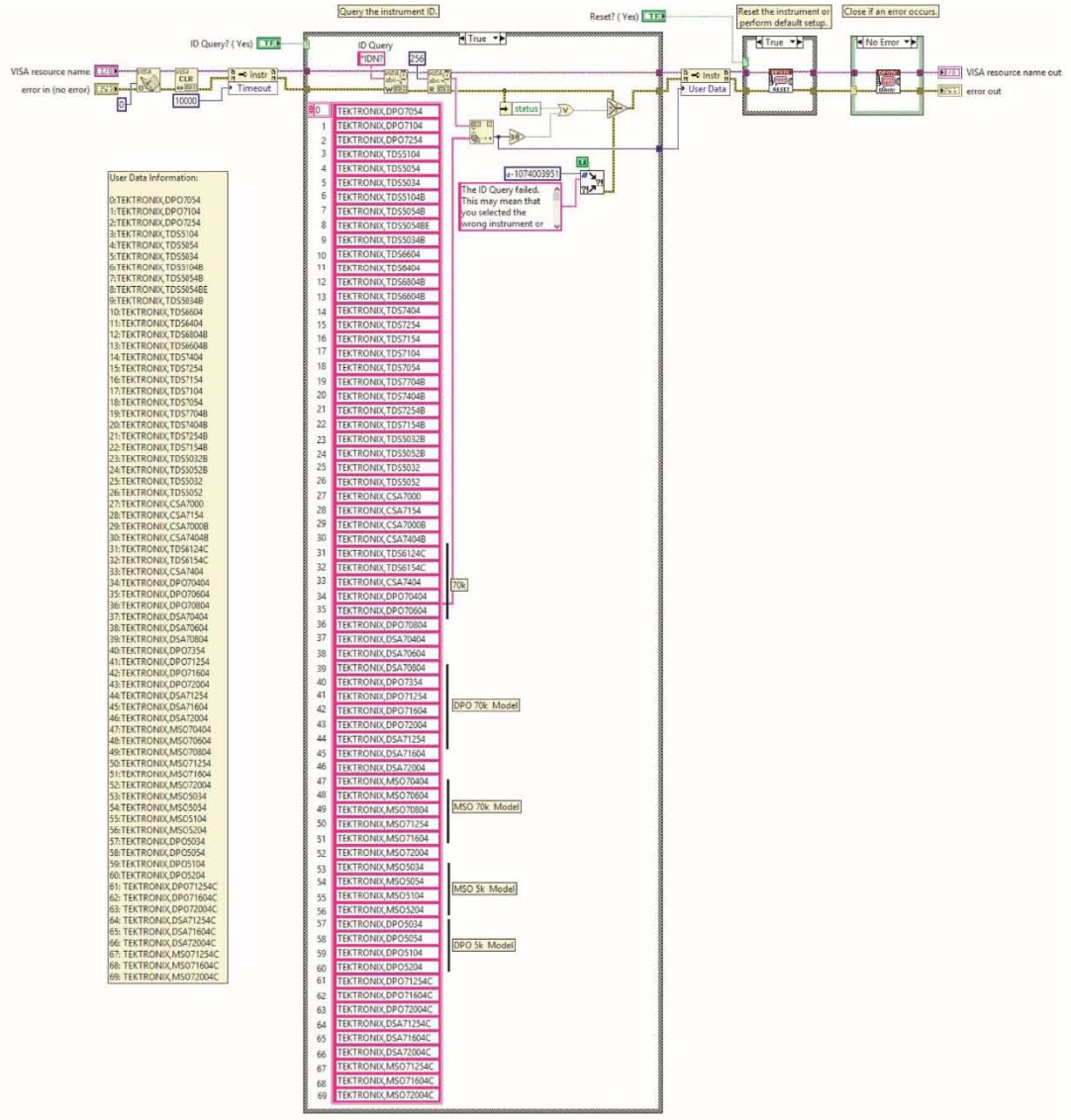


Triggerlaser.vi

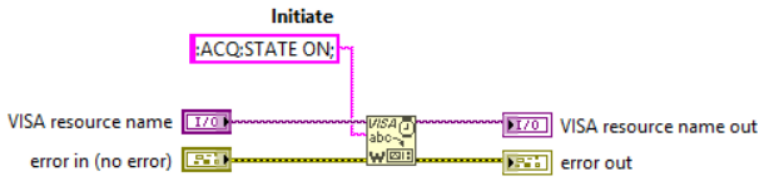


APÉNDICE B: Subprogramas del instrumento TDS 5054B

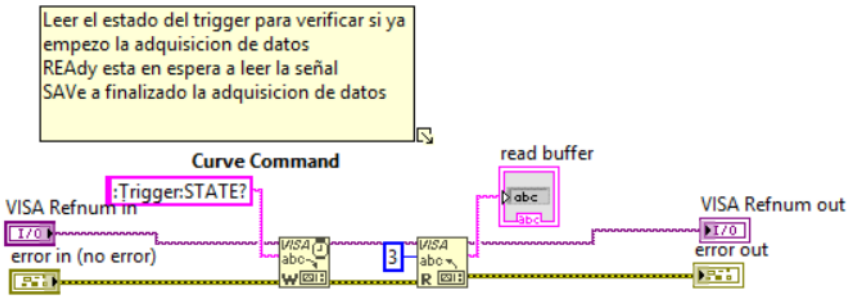
Subprograma Initialize.vi



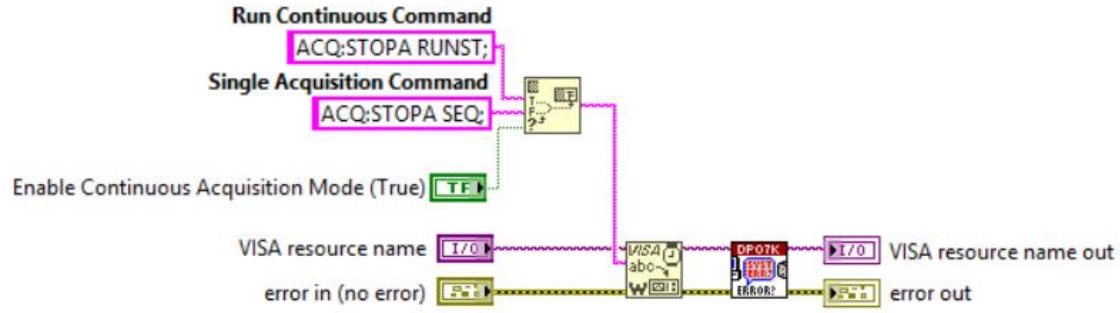
Subprograma Initiate.vi



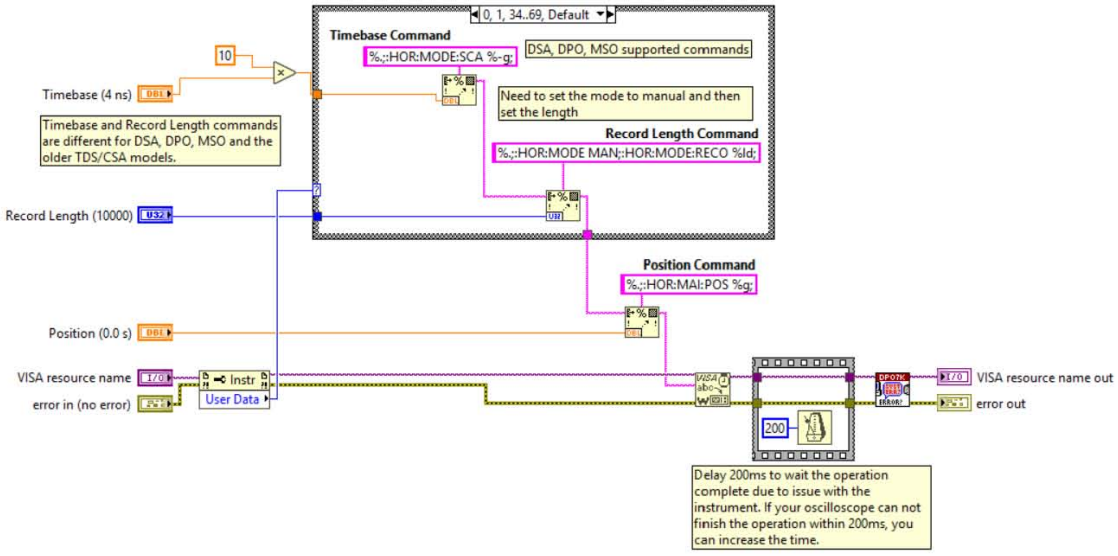
Programa de Monitoreo del estado del instrumento TDS 5054B



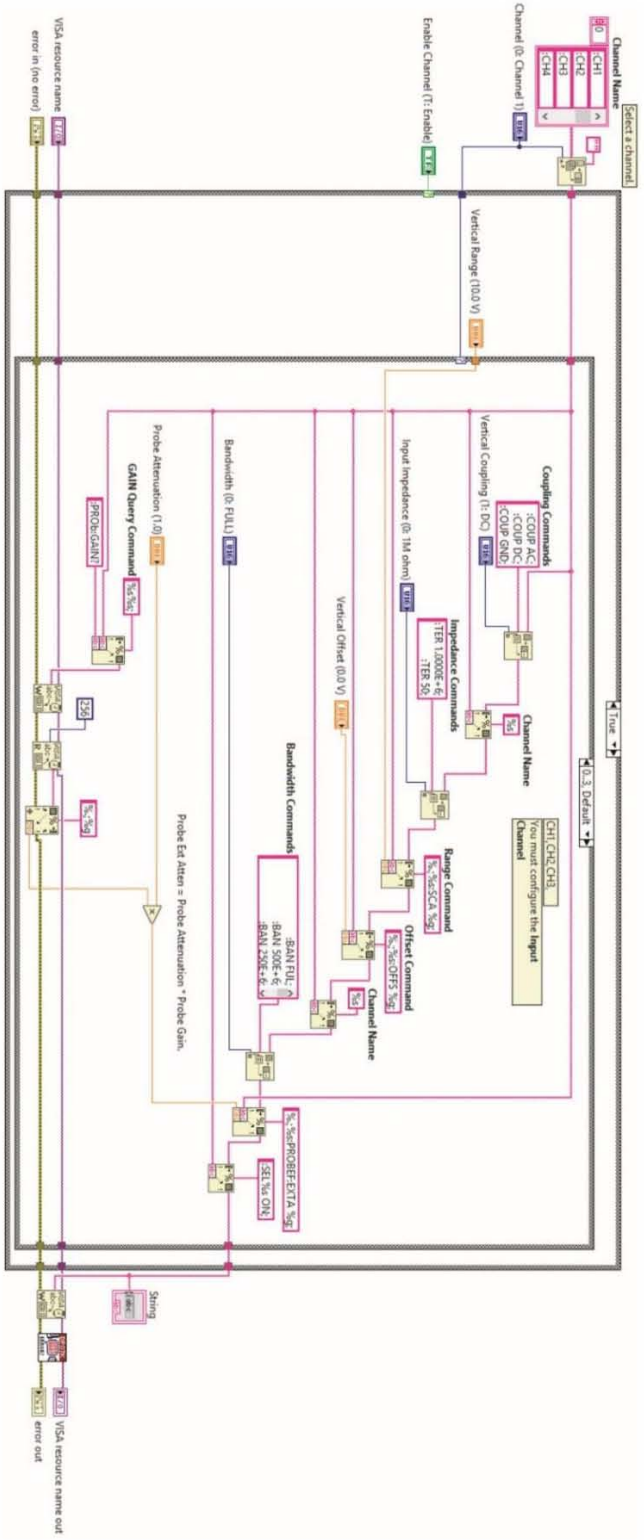
Subprograma Configure Continuous Acquisition.vi



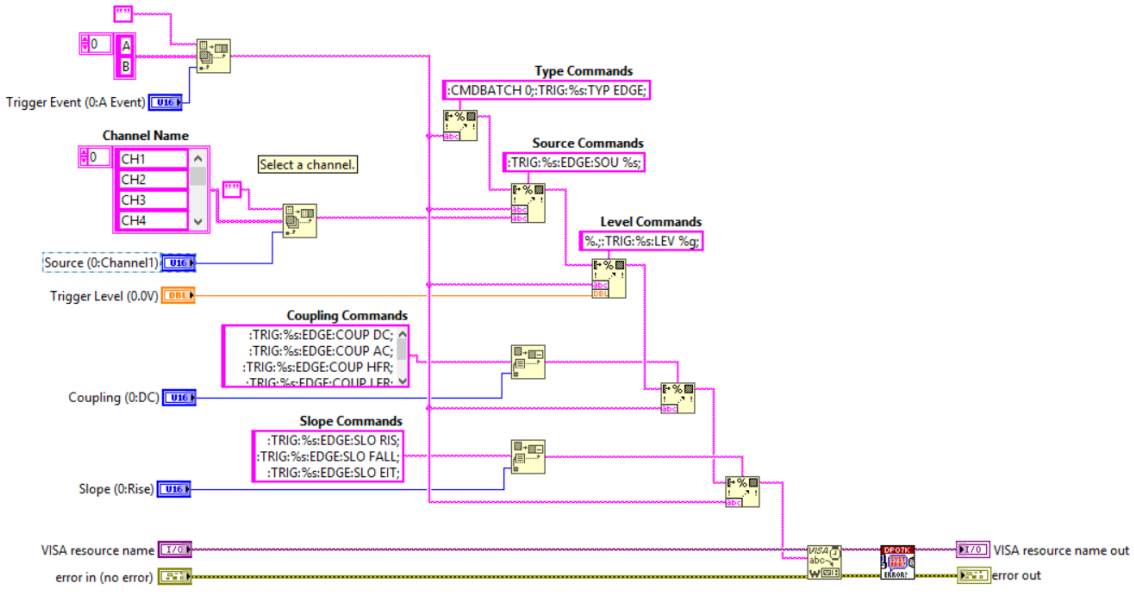
Subprograma Timebase.vi



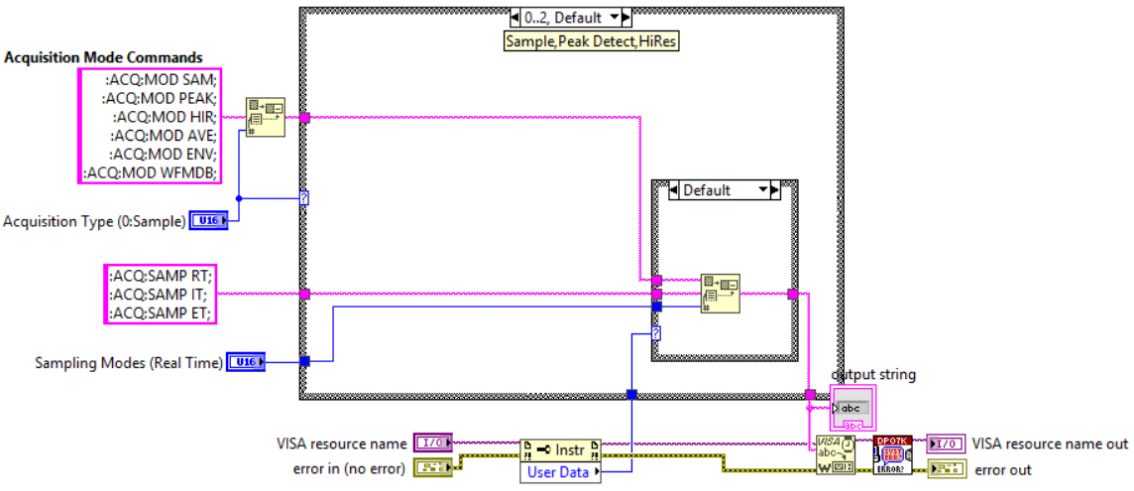
Subprograma Channel.vi



Subprograma Configure Trigger(Edge).vi



Subprograma Configure Acquisition.vi



APÉNDICE C: Archivo REMOTECONTROL.h

```
/*
 *    EKSPLA REMOTECONTROL.DLL
 *
 */
/*
 * REMOTECONTROL.H revision history
 *
 *   - DLL version v1.6.0.0 and above:
 *          Added Register type functions
 *          Added rcSetRegFromDoubleA and rcSetRegFromStringA
 */
#ifndef __REMOTECONTROL_H
#define __REMOTECONTROL_H
#include "windows.h"
#ifdef _MSC_VER
#define DLLimport __declspec(dllimport)
#else
#define DLLimport
#endif
              ;int bbb;
#ifdef __cplusplus
extern "C" {
#endif
/*
 *   Function return codes
```

```
 */
#define RMTCTRLERR_OK              0  // Success, no error
#define RMTCTRLERR_NOMOREDATA      1  // End of registers list or enumerated values list
#define RMTCTRLERR_NOCFGFILE       2  // No config file found
#define RMTCTRLERR_WRONGCFGFILE    3  // wrong CFG file
#define RMTCTRLERR_BUFFERTOOSHORT  4  // application provided return buffer is too short
#define RMTCTRLERR_NOSUCHDEVICE    5  // no such device name
#define RMTCTRLERR_NOSUCHREGISTER  6  // no such register name
#define RMTCTRLERR_CANTCONNECT     7
#define RMTCTRLERR_TIMEOUT         8  // timeout waiting for device answer
#define RMTCTRLERR_READONLY        9  // register is read only
#define RMTCTRLERR_NOT_NV          10  // register is not NV
#define RMTCTRLERR_HILIMIT         11  // attempt to set value above upper limit
#define RMTCTRLERR_LOLIMIT         12  // attempt to set value below bottom limit
#define RMTCTRLERR_NOSUCHVALUE     13  // attempt to set not allowed value
#define RMTCTRLERR_NOTLOGGED       14  // register is not being logged
#define RMTCTRLERR_MEMORYFULL      15  // not enough memory
#define RMTCTRLERR_LOGISEMPTY      16  // no data in the queue yet
#define RMTCTRLERR_ALREADYCONNECTED 17  // already connected, please disconnect first
#define RMTCTRLERR_NOTYETCONNECTED  18  // not connected, please connect first
/*
  Function rcGetRegFromLogAsDouble and rcGetRegFromLogAsString
  may or return value with RMTCTRLERR_LOGOVERFLOW.
  This happens when data retrieval succeeds, but Log FIFO
  overflow is detected.
*/
```

```
#define RMTCTRLERR_LOGOVERFLOW  0x80000000 // Log FIFO overrun occured

/*

 *  Functions list


 Connection functions:

   rcConnect

   rcDisconnect


 Devices and registers enumeration functions:

   rcGetFirstDeviceName

   rcGetNextDeviceName

   rcGetFirstRegisterName

   rcGetNextRegisterName


 Register classification functions

   rcIsRegisterWriteable

   rcIsRegisterNV

   rcGetRegFirstEnumValue

   rcGetRegNextEnumValue


 Register access functions:

   rcGetRegAsDouble

   rcSetRegFromDouble

   rcSetRegFromDoubleA

   rcSetRegNVFromDouble

   rcGetRegAsString
```

```
  rcSetRegFromString
  rcSetRegFromStringA
  rcSetRegNVFromString

 Register log and access functions:
  rcLogRegStart
  rcLogRegStop
  rcGetRegFromLogAsDouble
  rcGetRegFromLogAsString
 *
 */
/*****************************************************************
   Function:     Connect
   Description:  Makes a connection attempt
   Arguments:    connectiontype: 0 - direct
                                 1 - rs232
                 comportnumber:  1 - COM1
                                 2 - COM2
                                 ...
   Return value: error code, one of RMTCTRLERR_xxx list
 */
int __stdcall rcConnect(int connectiontype, int comportnumber);
/*****************************************************************
   Function:     Disconnect
   Description:  Disconnects
   Arguments:    none
```

```
   Return value: error code, one of RMTCTRLERR_xxx list
 */
int __stdcall rcDisconnect(void);
/*****************************************************************
   Function:     GetFirstDeviceName
   Description:  Get the name of first device in the list. Use
                 GetNextDeviceName() in a loop to retrieve all
                 device names.
   Arguments:    devname:       pointer to C string. GetFirstDeviceName will
                                write to *devname
                 maxdevnamelen: Maximum length of devname string including
                                terminatiion character.
   Return value: error code, one of RMTCTRLERR_xxx list
 */
int __stdcall rcGetFirstDeviceName(char *devname, int maxdevnamelen);
/*****************************************************************
   Function:     GetNextDeviceName
   Description:  Get the name of next device in the list. Use
                 GetFirstDeviceName() to restart retrieval from
                 the beginning of device names list
   Arguments:    devname:       pointer to C string. GetFirstDeviceName will
                                write to *devname
                 maxdevnamelen: Maximum length of devname string including
terminatiion character.
   Return value: error code, one of RMTCTRLERR_xxx list
                 On the end of devices list, GetNextDeviceName will
```

```
            return RMTCTRLERR_NOMOREDATA
 */
int __stdcall rcGetNextDeviceName(char *devname, int maxdevnamelen);
/*****************************************************************
   Function:     GetFirstRegisterName
   Description:  Get first register name of specific device. Use
            GetNextRegName() in a loop to retrieve all device
            register names.
   Arguments:    devname:       pointer to device name string.
            regname:       pointer to register name string. GetFirstRegisterName will
                  write to *regname
            maxregnamelen: Maximum length of string including terminatiion character.
   Return value: error code, one of RMTCTRLERR_xxx list
 */
int __stdcall rcGetFirstRegisterName(const char * devname, char *regname, int maxregnamelen);
/*****************************************************************
   Function:     GetNextRegisterName
   Description:  Get next register name of specific device. Use GetFirstRegisterName()
            to reset retrival of register names from the beginning.
   Arguments:    regname:       pointer to C string. GetNextParName will
                  write to *parname
            maxregnamelen: Maximum length of string including terminatiion
                  character.
   Return value: error code, one of RMTCTRLERR_xxx list.
            On the end of registers list, GetNextRegName will
            return RMTCTRLERR_NOMOREDATA
```

```
 */
int __stdcall rcGetNextRegisterName(char *regname, int maxparlen);
/*****************************************************************
   Function:     rcIsRegisterWriteable
   Description:  Determine if register is writeable
   Arguments:    devname:     pointer to device name string
                 regname:     pointer to register name string.
                 isWriteable: pointer to integer. On success
                              rcIsRegisterWriteable will write:
                               1 - in case register is writeable
                               0 - in case register is not writeable
   Return value: error code, one of RMTCTRLERR_xxx list.
 */
int __stdcall rcIsRegisterWriteable(const char *devname, const char *regname,
                            int *isWriteable);
/*****************************************************************
   Function:     rcIsRegisterNV
   Description:  Determine if register is NV (nonvolatile)
   Arguments:    devname:     pointer to device name string
                 regname:     pointer to register name string.
                 isNV:        pointer to integer. On success
                              rcIsRegisterNV will write:
                               1 - in case register is NV
                               0 - in case register is not NV
   Return value: error code, one of RMTCTRLERR_xxx list.
 */
```

```
int __stdcall rcIsRegisterNV(const char *devname, const char *regname,
                        int *isNV);
/*****************************************************************
   Function:     rcGetRegFirstEnumValue
   Description:  For registers having list of values, like ON, OFF, DISABLE etc,
                 this function returns first available value.
                 Attempt to get enumerated value of numeric register
                 returns RMTCTRLERR_NOMOREDATA.
                 Use rcGetRegNextEnumValue() in a loop to
                 retrieve all available enumerated register values.
   Arguments:    devname:        pointer to device name string.
                 regname:        pointer to register name string.
                 enumname:       pointer to value. rcGetRegFirstEnumValue will
                                 write value string to *enumname
                 maxenumnamelen: Maximum length of enumname string including terminatiion
                                 character.
   Return value: error code, one of RMTCTRLERR_xxx list
 */
int __stdcall rcGetRegFirstEnumValue(const char * devname, const char *regname,
                                char * enumname, int maxenumnamelen);
/*****************************************************************
   Function:     rcGetRegNextEnumValue
   Description:  See rcGetRegFirstEnumValue.
                 rcGetRegNextEnumValue() will return next available listed
                 value.
   Arguments:    devname:        pointer to device name string.
```

```
            regname:        pointer to register name string.
            enumname:       pointer to value. rcGetRegNextEnumValue will
                            write value string to *enumname
            maxenumnamelen: Maximum length of enumname string including terminatiion
                            character.
   Return value: error code, one of RMTCTRLERR_xxx list
 */
int __stdcall rcGetRegNextEnumValue(const char * devname, const char *regname,
                                    char * enumname, int maxenumnamelen);
/*****************************************************************
   Function:    GetRegAsDouble
   Description:  Return register value as double.
   Arguments:   devname:   pointer to device name string
            regname:   pointer to register name string.
            value:     pointer to double, rcGetRegAsDouble will overwrite
                       on success
            timeout:   timeout in milliseconds. (-1) - infinite timeout
            timestamp: pointer to int. In case timestamp is not a NULL,
                       rcGetRegAsDouble will write to *timestamp the time
                       stamp of received message
   Return value: error code, one of RMTCTRLERR_xxx list.
 */
int __stdcall rcGetRegAsDouble(const char *devname, const char *regname,
                               double *value,       int timeout,
                               int *timestamp);
/*****************************************************************
```

```
   Function:     SetRegFromDouble

   Description:  Set register value from double variable.

   Arguments:    devname:   pointer to device name string

                 regname:   pointer to register name string.

                 value:     value

   Return value: error code, one of RMTCTRLERR_xxx list.

 */

int __stdcall rcSetRegFromDouble(const char *devname, const char *regname, double value);

/*****************************************************************

   Function:     SetRegFromDoubleA

   Description:  Set register value from double variable.

   Arguments:    devname:     pointer to device name string

                 regname:     pointer to register name string.

                 value:       value

                 forcelimits: forcelimits==0 - complain when value is out of limits

                              forcelimits!=0 - force value limits

   Return value: error code, one of RMTCTRLERR_xxx list.

 */

int __stdcall rcSetRegFromDoubleA(const char *devname, const char *regname, double value,

                                  int forcelimits);

/*****************************************************************

   Function:     SetRegNVFromDouble


   Description:  Set register NV value from double variable.


   Arguments:    devname:   pointer to device name string
```

```
            regname:   pointer to register name string.
            value:     value
   Return value: error code, one of RMTCTRLERR_xxx list.
 */
int __stdcall rcSetRegNVFromDouble(const char *devname, const char *regname, double value);
/*****************************************************************
   Function:     GetRegAsString
   Description:  Return register value as string.
   Arguments:    devname:   pointer to device name string
            regname:   pointer to register name string.
            value:     pointer to string, GetRegAsString will overwrite
            maxvallen: maximum length of value string, including terminating character
            timeout:   timeout in milliseconds. (-1) - infinite timeout
            timestamp: pointer to int. In case timestamp is not a NULL,
                  rcGetRegAsDouble will write to *timestamp the time
                  stamp of received message
   Return value: error code, one of RMTCTRLERR_xxx list.
 */
int __stdcall rcGetRegAsString(const char *devname, const char *regname,
                  char *value, int maxvallen, int timeout,
                  int *timestamp);
/*****************************************************************
   Function:     SetRegFromString
   Description:  Set register value from string.
   Arguments:    devname:   pointer to device name string
            regname:   pointer to register name string.
```

```
            value:     pointer to string
   Return value: error code, one of RMTCTRLERR_xxx list.
 */
int __stdcall rcSetRegFromString(const char *devname, const char *regname, const char *value);
/*****************************************************************
   Function:     SetRegFromStringA

   Description:  Set register value from string.
   Arguments:    devname:     pointer to device name string
                 regname:     pointer to register name string.
                 value:       pointer to string
                 forcelimits: forcelimits==0 - complain when value is out of limits
                              forcelimits!=0 - force value limits
   Return value: error code, one of RMTCTRLERR_xxx list.
 */
int __stdcall rcSetRegFromStringA(const char *devname, const char *regname, const char *value,
                                  int forcelimits);
/*****************************************************************
   Function:     SetRegNVFromString
   Description:  Set register value from string.
   Arguments:    devname:   pointer to device name string
                 regname:   pointer to register name string.
                 value:     pointer to string
   Return value: error code, one of RMTCTRLERR_xxx list.
 */
int __stdcall rcSetRegNVFromString(const char *devname, const char *regname, const char *value);
```

```
/*****************************************************************
   Function:     rcLogRegStart
   Description:  Start logging register
   Arguments:    devname:   pointer to device name string
                 regname:   pointer to register name string.
                 queuesize: Size of memory to allocate
                        for data queue.
   Return value: error code, one of RMTCTRLERR_xxx list.
 */
int __stdcall rcLogRegStart(const char *devname, const char *regname,
                        int queuesize);
/*****************************************************************
   Function:     rcLogRegStop
   Description:  Stop logging register
   Arguments:    devname:   pointer to device name string
                 regname:   pointer to register name string.
   Return value: error code, one of RMTCTRLERR_xxx list.
 */
int __stdcall rcLogRegStop(const char *devname, const char *regname);
/*****************************************************************
   Function:     rcGetRegFromLogAsDouble
   Description:  Get register value from log queue as double value.
   Arguments:    devname:   pointer to device name string
                 regname:   pointer to register name string.
                 value:     pointer to double, rcGetRegAsDouble will overwrite
                 timestamp: pointer to int. In case timestamp is not a NULL,
```

```
                    rcGetRegAsDouble will write to *timestamp the time
                    stamp of received message
   Return value: error code, one of RMTCTRLERR_xxx list.
 */
int __stdcall rcGetRegFromLogAsDouble(const char *devname, const char *regname,
                     double *value,  int *timestamp);
/*************************************************************
   Function:     rcGetRegFromLogAsString
   Description:  Get register value from log queue as string.
   Arguments:    devname:   pointer to device name string
                 regname:   pointer to register name string.
                 value:     pointer to string, GetRegAsString will overwrite
                 maxvallen: maximum length of value string, including terminating character
                 timestamp: pointer to int. In case timestamp is not a NULL,
                       rcGetRegAsDouble will write to *timestamp the time
                       stamp of received message
   Return value: error code, one of RMTCTRLERR_xxx list.
 */
int __stdcall rcGetRegFromLogAsString(const char *devname, const char *regname,
                     char *value, int maxvallen, int *timestamp);
#ifdef __cplusplus
} // extern "C"
#endif
#endif //__REMOTECONTROL_H
```