**REPORTE DE RESIDENCIA PROFESIONAL**
**INSTITUTO TEGNOLOGICO DE TUXTLA GUTIERREZ**

**TITULO DEL PROYECTO:**

*"ESTUDIO Y DESARROLLO DE INSTRUMENTACIÓN PARA LA AUTOMATIZACIÓN DE TOMA DE DATOS DE INTERFERÓMETROS DE NANO-MEDICIONES"*

**PERIODO:**

DEL 19 DE ENERO AL 19 DE MAYO DEL 2009

**NOMBRE DE LA INSTITUCION:**

CENTRO DE INVESTIGACIONES EN ÓPTICA, A. C.

**NOMBRE DEL ALUMNO**

Daniel Antonio Pérez Villatoro .

**ASESOR INTERNO**

M. en C. RAUL MORENO RINCON

**ASESOR EXTERNO**

DR. MOISES CYWIAK GARBARCEWICZ

## ÍNDICE

### CAPITULO I



### CAPITULO II

**FUNDAMENTO TEORICO**

## CAPITULO III

## PROCEDIMIENTO Y DESCRIPCION DE LAS ACTIVIDADES REALIZADAS



## CAPITULO IV

## RESULTADOS

# CAPITULO I

## 1.1.- INTRODUCCIÓN

El trabajo que se desarrolla a continuación pretende introducirlo a los conceptos de acondicionamiento, generación y medición de señales de manera automatizada utilizando C++ Builder como plataforma de programación y aparatos de alto desarrollo tecnológico como son el uso de tarjetas de generación y adquisición de datos del tipo DAQ (Data Acquisition) y GPIB (General-Purpose Instrumentation Bus) de National Instrument, e instrumentos como Lock-ins (de Stanford Research systems) y osciloscopios digitales.

Se mostraran los pasos que se siguieron para programar la tarjeta tipo DAQ, de igual forma se mostrara paso a paso como programar la tarjeta GPIB para adquirir datos de un Lock-in y un osciloscopio digital que cuentan con este protocolo de comunicación utilizando C++ Builder.

En la parte anexa del trabajo se explica con detalle mediante una imagen la utilidad que tendrá la programación de estas dos tarjetas al combinarlas en un solo programa.

## 1.2.- JUSTIFICACIÓN

Con la programación de estas 2 tarjetas se pretende automatizar la medición de datos de un interferómetro que se explica en la parte anexa de este trabajo además de poder almacenar en un archivo de texto estas mediciones para el uso que sea necesario, ya son esta mediciones las que permitirán conocer las deformaciones que presente el objeto que recordemos estas deformaciones están en valores de micras y no son visibles al ojo humano y muchas veces ni siquiera a un microscopio.

Automatizar la toma de datos del interferómetro además de ser un método no invasivo permitirá conocer casi al instante las deformaciones que el objeto que se está analizando sin necesidad de utilizar aparatos de alto costo y que además muchos son métodos invasivos que pueden llegar a dañar el objeto a analizar

La DAQ servirá para generar una señal triangular en la cual se podrá controlar la amplitud que podrá variar de 1 a 10v, así como el tiempo en que esta tardara en generarse, es decir en subir y bajar, ejemplo si quiero una señal triangular de 1 v y que tarde en generarse en 1 minuto el programa calculara el tiempo necesario para que la señal tarde 30 segundos en generar la señal de subida o la pendiente de subida de 0v a 1v y 30 segundos en generar la señal de bajada o la pendiente de bajada de 1v a 0v, completando así 1 minuto en generar la señal triangular completa a 1v.

Esta señal servirá para mover un nanoposicionador que es un pizoelectrico y que sostendrá al objeto a analizar, el nanoposicionador moverá al objeto hacia arriba y hacia abajo al mismo tiempo que oscila debido a la frecuencia de referencia del lock-in, el lock-in tomara los datos mientras la DAQ genere la señal, al hacer esto al hacer esto se tiene una especie de aparato que escanea la superficie del objeto y que al graficarlos mostrara un perfil de la superficie del objeto.

## 1.3.- OBJETIVOS

El objetivo es programar 2 tarjetas una tipo DAQ y otra GPIB.

La DAQ debe generar una señal triangular donde se pueda controlar el voltaje y el tiempo de la señal con un retraso en la bajada de 1 segundo.

La tarjeta GPIB servirá para tomar datos automáticamente de un lock-in y un osciloscopio digital y poder almacenar estas mediciones en un archivo de texto.

Por último ambos programas se juntaran en uno solo (la tarjeta DAQ y la GPIB) con el fin de que el lock-in mediante la tarjeta GPIB estará tomando datos durante todo el tiempo que tarde la señal que genera la tarjeta DAQ.

## 1.4.- CARACTERIZACION DEL AREA DE TRABAJO

El objetivo del Departamento de Metrología Óptica es el cálculo de diversas cantidades físicas (temperatura, velocidad, presión, distancia, desplazamiento, esfuerzo, etc.) de objetos y/o regiones de observación. El reconocimiento de patrones también es importante. Esto último incluye el cálculo de forma, textura, color, dimensiones, etc., de objetos Para llevar a cabo este objetivo, se usan arreglos ópticos que incluyen dispositivos opto-electrónicos, fuentes de luz y software. Generalmente, se capturan imágenes del objeto bajo estudio y éstas a su vez son procesadas para recuperar en forma cuantitativa la información de interés.

Los campos en los que se lleva a cabo investigación en el departamento son pruebas no destructivas, inspección de procesos, control y automatización, perfilometría, reconocimiento de patrones, análisis de deformación, análisis aerodinámico, detección de fractura, calibración, visión por computadora, análisis de vibraciones, etc. Estos campos implican el uso combinado de técnicas experimentales y numéricas.

El laboratorio de Metrología Heterodina se ha destacado por realizar investigación y desarrollo a nivel internacional de sistemas ópticos con aplicaciones a la metrología como son mediciones de superficies de muy alta calidad óptica en áreas micrométricas logrando obtener resoluciones verticales de unos cuantos nanómetros y con resoluciones laterales muy cercanos a la longitud de onda de la fuente de iluminación.

## 1.5.- PROBLEMAS A RESOLVER

1.- Programación de la tarjeta DAQ para la generación de la señal triangular que servirá para mover un nanoposicionador

2.- Programación de la tarjeta GPIB para controlar un osciloscopio.

3.- Programación de la tarjeta GPIB para controlar un lock-in.

4.-Combinar los programas de generación de señal triangular mediante la tarjeta DAQ con el de adquisición de datos del lock-in mediante la tarjeta GPIB.

## 1.6.- ALCANCES

Automatizar la toma de datos del interferómetro que se explica en la parte anexa de este trabajo ya que mediante la programación y la combinación de ambas tarjetas la DAQ que genera la señal triangular y la GPIB que tomara los datos que reciba el Lock-in en tiempo real, mediante estas mediciones del lock-in se pretende conocer las deformaciones que el objeto presenta ya que al graficar esos datos del lock-in se conocerá casi al instante.

Con la programación del la tarjeta GPIB para el osciloscopio se pretende poder capturar las señales que queramos o que estemos midiendo y almacenarlas en la computadora y poder darles cualquier tipo de tratamiento matemático que nosotros o el usuario considere necesario para su fines.

## 1.7.- LIMITACIONES

Aunque el protocolo GPIB es estándar para cualquier instrumento que cuente con este tipo de comunicación surgen algunos detalles que por lo cual los programas solo se podrán utilizar en los instrumentos de la marca que se menciona.

El programa que se utiliza para programar las tarjetas es en builder c++ y por el momento la tarjeta GPIB solo se podrá utilizar en el lock-in de la marca Stanford Research systems modelo SR850 ya que el lock-in tiene sus propias instrucciones o comandos para que desde la computadora y mediante la tarjeta lo podamos controlar o pedir los datos que nosotros necesitemos y estos podrían variar con respecto a otro instrumento igual, pero diferente marca.

Al igual que el lock-in el osciloscopio es de la marca tektronix TDS 2012 y también cuenta con sus propios comandos o instrucciones para que este reconozca que es lo que le estamos pidiendo ya sea el valor de la señal medida (CURV?) o el valor de la escala en v/d del canal 1 ("CH1:SCA?") por lo tanto estos comando podrían variar con respecto a otro osciloscopio.

Debido a que el fabricante proporciona los comandos que necesita su instrumento para poder controlarlos es necesario checar el manual de programación del instrumento.

Otra de las limitaciones que podemos encontrar es que para poder ejecutar los programas es necesario que la computadora cuente con las tarjetas y los drivers instalados debido si no cuentan con estos los programas no podrán ejecutarse.

# CAPITULO II

# FUNDAMENTO TEORICO

## 2.1.- CONCEPTOS BASICOS

### 2.1.1.- LUZ

La luz es una radiación electromagnética tiene una naturaleza dual como partícula (fotón) y como onda, puede ser caracterizada en términos de su longitud de onda (distancia sucesiva entre dos ondas), frecuencia (número de ondas por espacio de tiempo) y amplitud (diferencia entre los picos máximos y mínimos). La longitud de onda es medida en unidades métricas µm, nm, Angstrom(Å)(0.1nm).

**Reflexión.**

Cuando los rayos de luz llegan a un cuerpo en el cual no pueden continuar propagándose, salen desviados en otra dirección, es decir, se reflejan. La forma en que esto ocurre depende del tipo de superficie sobre la que inciden y del ángulo que forman sobre la misma.

**Absorción.**

Existen superficies y objetos que absorben la mayor parte de las radiaciones luminosas que les llegan. Estos objetos se ven de color negro. Otros tipos de superficies y objetos, absorben sólo unas determinada gama de longitudes de onda, Y reflejan el resto.

**Refracción**.

El cambio de dirección que sufren los rayos luminosos al pasar de un medio a otro, donde su velocidad es distinta, da lugar a los fenómenos de refracción. Así si un haz de rayos luminosos incide sobre la superficie de un cuerpo transparente, parte de ellos se reflejan mientras que otra parte se refracta, es decir penetran en el cuerpo transparente experimentando un cambio en su dirección de movimiento.

**Difracción**

En física, la difracción es un fenómeno característico de las ondas que consiste en la dispersión y curvado aparente de las ondas cuando encuentran un obstáculo.

## 2.1.2.- LASER

(Light Amplification by Stimulated Emission of Radiation, Amplificación de Luz por Emisión Estimulada de Radiación)

Hay cuatro procesos básicos que se producen en la generación del láser, denominados bombeo, emisión espontánea de radiación, emisión estimulada de radiación y absorción.

### Bombeo

Se provoca mediante una fuente de radiación como puede ser una lámpara, el paso de una corriente eléctrica o el uso de cualquier otro tipo de fuente energética que provoque una emisión.

### Emisión espontánea de radiación

Los electrones que vuelven al estado fundamental emiten fotones. Es un proceso aleatorio y la radiación resultante está formada por fotones que se desplazan en distintas direcciones y con fases distintas generándose una radiación monocromática incoherente.

### Emisión estimulada de radiación

La emisión estimulada, base de la generación de radiación de un láser, se produce cuando un átomo en estado excitado recibe un estímulo externo que lo lleva a emitir fotones y así retornar a un estado menos excitado. El estímulo en cuestión proviene de la llegada de un fotón con energía similar a la diferencia de energía entre los dos estados. Los fotones así emitidos por el átomo estimulado poseen fase, energía y dirección similares a las del fotón externo que les dio origen. La emisión estimulada descrita es la raíz de muchas de las características de la luz láser. No sólo produce luz coherente y monocroma, sino que también "amplifica" la emisión de luz, ya que por cada fotón que incide sobre un átomo excitado se genera otro fotón.

### Absorción

Proceso mediante el cual se absorbe un fotón. El sistema atómico se excita a un estado de energía más alto, pasando un electrón al estado metaestable. Este fenómeno compite con el de la emisión estimulada de radiación.

## 2.2.- INTERFERENCIA

Es el efecto que se produce cuando dos o más ondas se entrecruzan. Cuando las ondas interfieren entre sí, la amplitud (intensidad o tamaño) de la onda resultante depende de las frecuencias, fases relativas (posiciones relativas de crestas y valles) y amplitudes de las ondas iníciales.

Por ejemplo, la interferencia constructiva se produce en los puntos en que dos ondas de la misma frecuencia que se solapan o entrecruzan están en fase; es decir, cuando las crestas y los valles de ambas ondas coinciden. En ese caso, las dos ondas se refuerzan mutuamente y forman una onda cuya amplitud es igual a la suma de las amplitudes individuales de las ondas originales.

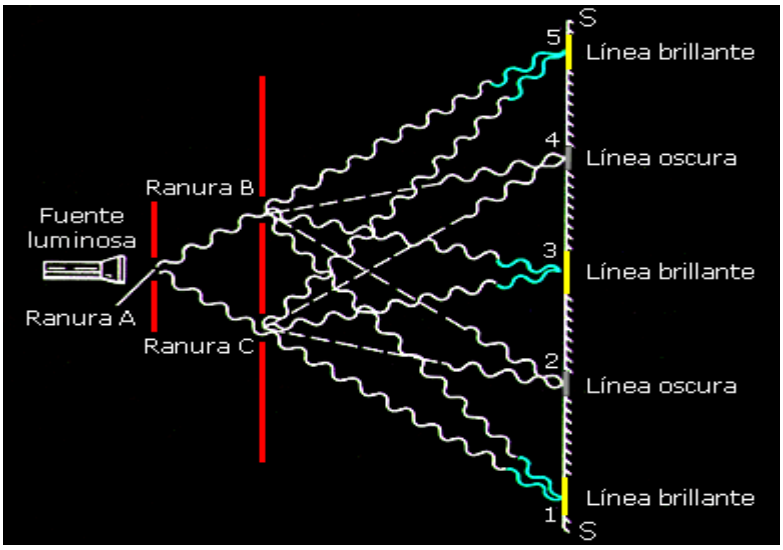
La interferencia destructiva se produce cuando dos ondas de la misma frecuencia están completamente desfasadas una respecto a la otra; es decir, cuando la cresta de una onda coincide con el valle de otra. En este caso, las dos ondas se cancelan mutuamente. Cuando las ondas que se cruzan tienen frecuencias diferentes o no están exactamente en fase ni desfasadas, el esquema de interferencia puede ser más complejo. La luz visible está formada por ondas electromagnéticas que pueden interferir entre sí. La interferencia de ondas de luz causa, por ejemplo, las irisaciones que se ven a veces en las burbujas de jabón. La luz blanca está compuesta por ondas de luz de distintas longitudes de onda. Las ondas de luz reflejadas en la superficie interior de la burbuja interfieren con las ondas de esa misma longitud reflejadas en la superficie exterior. En algunas de las longitudes de onda, la interferencia es constructiva, y en otras destructivas. Como las distintas longitudes de onda de la luz corresponden a diferentes colores, la luz reflejada por la burbuja de jabón aparece coloreada. El fenómeno de la interferencia entre ondas de luz visible se utiliza en holografía e interferometría.

La interferencia puede producirse con toda clase de ondas, no sólo ondas de luz. Las ondas de radio interfieren entre sí cuando rebotan en los edificios de las ciudades, con lo que la señal se distorsiona. Cuando se construye una sala de conciertos hay que tener en cuenta la interferencia entre ondas de sonido, para que una interferencia destructiva no haga que en algunas zonas de la sala no puedan oírse los sonidos emitidos desde el escenario. Arrojando objetos al agua estancada se puede observar la interferencia de ondas de agua, que es constructiva en algunos puntos y destructiva en otros.

### 2.2.1.- Interferencia de ondas luminosas - Experiencia de Tomás Young

Young demostró que en determinadas circunstancias, los haces luminosos podían interferir entre si, dando lugar a franjas luminosas alternadas con otras oscuras. Luego de demostrarlo y explicarlo de forma convincente, constituyó una prueba definitiva de la naturaleza ondulatoria de la luz.

La experiencia de Young se basaba en colocar una pantalla con una ranura (A), iluminada con una luz monocromática, próxima a ella colocó otra pantalla con dos rendijas B y C paralelas entre sí y a poca distancia una de otra, un poco mas distante se colocaba otra pantalla (S) sobre la que se forman una serie de franjas brillantes y oscuras que se llaman franjas de interferencia.




Cada una de las ranuras B y C se convierten en fuentes secundarias de luz, cuyas ondulaciones, según el principio de Huyghens, se propagan en todos sentidos. Al utilizar una fuente luminosa y dos ranuras, se tiene la seguridad de obtener dos trenes de ondas con la misma fase, igual frecuencia (longitud de onda) y amplitud.

En el punto de mayor intensidad luminosa (3), el campo luminoso es el doble del que habría si en la pantalla hubiera un solo orificio (franja brillante), debido a que las ondas que provienen de B y C siguen caminos de igual longitud, sumándose en fase entre sí.

En los puntos 2 y 4 donde la intensidad es mínima (franjas oscuras), los campos luminosos están en oposición de fase y por lo tanto el campo resultante es nulo, debido a que los trenes de onda que llegan de B y C recorren trayectorias que se diferencian en media longitud de onda, restándose destructivamente. Es decir, esta distribución de intensidad es debida a la superposición de las ondas provenientes de cada uno de los orificios existentes en la pantalla y que, llegado a ciertos puntos de observación con una diferencia de fase, se suman destructivamente.

Cuando las frecuencias son iguales, como las velocidades de propagación son las mismas, en un punto cualquiera del espacio, la diferencia de fase entre las vibraciones que provienen de cada una de las fuentes que emiten ondas ilimitadas

permanece constante en el tiempo, y sólo depende del punto considerado. En las regiones en las que estas vibraciones llegan en fase, las amplitudes se suman y hay un máximo de intensidad. En otras regiones, las vibraciones están en oposición de fase y se restan destructivamente, de manera de producir allí un mínimo de intensidad.
Esto constituye el fenómeno de interferencia, y su constatación es una medida de la coherencia de las fuentes que la producen.

## 2.3.- INTERFEROMETRO

El Interferómetro, es un instrumento que emplea la interferencia de ondas de luz para la medida ultraprecisa de longitudes de onda de la luz misma, de distancias pequeñas y de determinados fenómenos ópticos.

Existen muchos tipos de interferómetros, pero en todos ellos hay dos haces de luz que recorren dos trayectorias ópticas distintas, determinadas por un sistema de espejos y placas, que finalmente se unen para formar franjas de interferencia. Para medir la longitud de onda de una luz monocromática se utiliza un interferómetro dispuesto de tal forma que un espejo situado en la trayectoria de uno de los haces de luz puede desplazarse una distancia pequeña, que puede medirse con precisión, y varía así la trayectoria óptica del haz. Cuando se desplaza el espejo una distancia igual a la mitad de la longitud de onda de la luz, se produce un ciclo completo de cambios en las franjas de interferencia. La longitud de onda se calcula midiendo el número de ciclos que tienen lugar cuando se mueve el espejo una distancia determinada.

### 2.3.1.- USOS DE LOS INTERFEROMETROS

Cuando se conoce la longitud de onda de la luz empleada, pueden medirse distancias pequeñas en la trayectoria óptica analizando las interferencias producidas. Esta técnica se emplea para medir el contorno de la superficie de los espejos de los telescopios. Los índices de refracción de una sustancia también pueden medirse con el interferómetro, y se calculan a partir del desplazamiento en las franjas de interferencia causado por el retraso del haz.

El principio del interferómetro también se emplea para medir el diámetro de estrellas grandes relativamente cercanas, como por ejemplo Betelgeuse. Como los interferómetros modernos pueden medir ángulos extremadamente pequeños, se emplean, también en este caso en estrellas gigantes cercanas, para obtener imágenes de variaciones del brillo en la superficie de dichas estrellas.

El principio del interferómetro se ha extendido a otras longitudes de onda, y en la actualidad está generalizado su uso en radioastronomía.

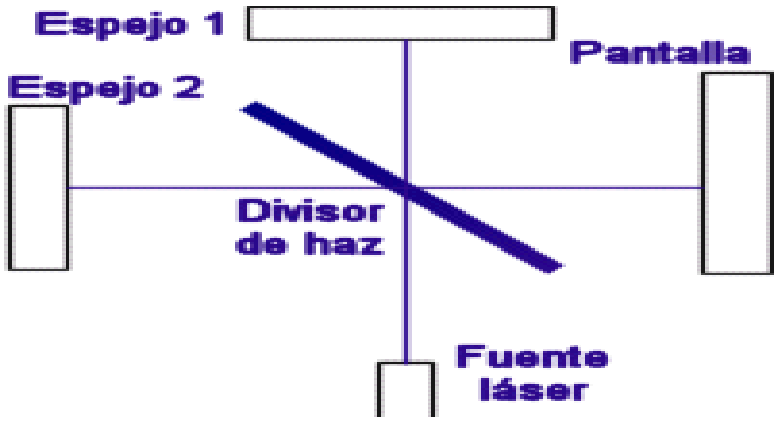
### 2.3.2.- EXPERIMENTO DE MICHELSON - MORLEY

Históricamente, el interferómetro más conocido es el diseñado alrededor de 1887 por el físico estadounidense Albert Michelson para un experimento que llevó a cabo con el químico estadounidense Edward Morley.

Es un interferómetro que permite medir distancias con una precisión muy alta. Su funcionamiento se basa en la división de un haz coherente de luz en dos haces para que recorran caminos diferentes y luego converjan nuevamente en un punto. De esta forma se obtiene lo que se denomina la figura de interferencia que permitirá medir pequeñas variaciones en cada uno de los caminos seguidos por los haces

El experimento estaba diseñado para medir el movimiento absoluto de la Tierra a través de una sustancia hipotética denominada éter, que según se suponía equivocadamente era el portador de las ondas de luz. Si la Tierra se desplazara a través de un éter estacionario, la luz que avanza en una trayectoria paralela a la dirección de movimiento de la Tierra tardaría un tiempo distinto en recorrer una distancia determinada que la luz que recorriera esa misma distancia en una trayectoria perpendicular al movimiento de la Tierra.

El interferómetro se construyó de forma que un haz de luz se dividía en dos trayectorias perpendiculares entre sí; después, los rayos se reflejaban y volvían a combinarse, formando franjas de interferencia. Si fuera correcta la hipótesis del éter, ambos haces de luz intercambiarían sus respectivos papeles al girar el aparato 90 grados (el haz que viajaba más rápido en la primera posición sería el más lento en la segunda posición), y se produciría un desplazamiento de las franjas de interferencia. Michelson y Morley no observaron ningún desplazamiento, y los experimentos posteriores confirmaron este resultado negativo. Hoy, el concepto de la propagación de ondas electromagnéticas a través del espacio vacío ha sustituido a la idea del éter.




En un principio, la luz es dividida por un divisor de haz en dos haces. El primero es reflejado y se proyecta hasta el espejo, del cual vuelve, atraviesa la superficie del divisor de haz y llega al detector. El segundo rayo atraviesa el divisor de haz, se refleja en el espejo luego es reflejado en el divisor de haz hacia abajo y llega al detector.

El espacio entre el divisor de haz y cada uno de los espejos se denomina brazo del interferómetro. Usualmente uno de estos brazos permanecerá inalterado durante un experimento, mientras que en el otro se colocarán las muestras a estudiar.

Hasta el observador llegan dos haces, que poseen una diferencia de fase dependiendo fundamentalmente de la diferencia de camino óptico entre ambos rayos. Esta diferencia de camino óptico puede depender de la posición de los espejos o de la colocación de diferentes materiales en cada uno de los brazos del interferómetro.

En general se emplean lentes para ensanchar el haz y que sea fácilmente detectable por un fotodiodo o proyectando la imagen en una pantalla. De esta forma el observador ve una serie de anillos, y al desplazar uno de los espejos notará que estos anillos comienzan a moverse. En esta forma se puede explicar la conservación de la energía, ya que la intensidad se distribuirá en regiones oscuras y regiones luminosas, sin alterar la cantidad total de energía.

# 2.4.- ADQUISICION DE DATOS

La Adquisición de Datos, consiste en la toma de muestras del mundo real (sistema analógico) para generar datos que puedan ser manipulados por un ordenador (sistema digital). Consiste, en tomar un conjunto de variables físicas, convertirlas en tensiones eléctricas y digitalizarlas de manera que se puedan procesar en una computadora. Se requiere una etapa de acondicionamiento, que adecua la señal a niveles compatibles con el elemento que hace la transformación a señal digital. El elemento que hace dicha transformación es el módulo o tarjeta de Adquisición de Datos (**DAQ**).

Los datos adquiridos son mostrados, analizados, y almacenados en una computadora, usando el software suministrado por el vendedor, o el control de esta se puede desarrollar usando varios lenguajes de programación de fines generales tales como BASIC, C, FORTRAN, Java, balbucean, Pascal. Los lenguajes de programación especializados usados para de adquisición de datos incluyen las EPICS, usadas para construir sistemas de adquisición de datos de gran escala, LabVIEW, que ofrece un ambiente de programación gráfica optimizado para de adquisición de datos, y MATLAB que proporciona un lenguaje de programación, y también las herramientas y las bibliotecas gráficas incorporadas para de adquisición de datos y el análisis

La adquisición de datos comienza con el fenómeno físico o la propiedad física del objeto que será medido. Esta propiedad puede ser la temperatura, intensad de luz, presión, etc. Un sistema de adquisición de datos efectivo puede medir todos estos diferentes fenómenos o propiedades.

Un transductor es un dispositivo que convierte una propiedad o fenómeno físico a una señal eléctrica medible.la habilidad de un sistema de adquisición de datos para medir estos diferentes fenómenos depende de los transductores para convertir los fenómenos físicos en señales medibles por el hardware de adquisición de datos. Los transductores son sinónimos con sensores en un sistema DAQ. Hay diferentes transductores para diferentes aplicaciones, como medir temperatura, presión, o fluidos. La DAQ también despliega diversas técnicas de acondicionamiento de señal para modificar adecuadamente diferentes señales eléctricas en el voltaje que puede ser digitalizada usando ADCs.

Las señales pueden ser digitales o analógicas dependiendo de la tarjeta. El acondicionamiento de la señal puede ser necesario si la señal no es conveniente para que la DAQ sea utilizada. La señal se puede amplificar o desamplificar, o puede requerir de un filtrado, o un amplificador del lock-in se incluye para realizar la desmodulación.

# 2.5.- LA TARJETA DAQ

La tarjeta DAQ es una interfaz entre la señal y la PC. Esta puede ser en forma de módulos que se conectan a la computadora por los puertos (paralelo, serial, USB, etc...), o tarjetas que se conectan a las ranuras de la tarjeta madre (PCI, ISA, PCIE, etc...).

Las tarjetas DAQ contienen múltiples componentes (multiplexores, ADC, DCA, TTL-I/O, RAM). Estos son accesibles vía bus por micro controladores, los cuales pueden correr con pequeños programas.

Ejemplos de Sistemas de Adquisición y control: DAQ para recoger datos (datalogger) medioambientales (energías renovables e ingeniería verde). · DAQ para audio y vibraciones (mantenimiento, test). · DAQ + control de movimiento (corte con laser). · DAQ + control de movimiento+ visión artificial (robots modernos).

### 2.5.1.- Etapa de acondicionamiento

Podemos encontrar estas etapas, aunque no todas están siempre presentes:

**Amplificación** Es el tipo más común de acondicionamiento. Para conseguir la mayor precisión posible la señal de entrada deber ser amplificada de modo que su máximo nivel coincida con la máxima tensión que el convertidor pueda leer.

**Aislamiento** - Otra aplicación habitual en el acondicionamiento de la señal es el aislamiento eléctrico entre el transductor y el ordenador, para proteger al mismo de transitorios de alta tensión que puedan dañarlo. Un motivo adicional para usar aislamiento es el garantizar que las lecturas del convertidor no son afectadas por diferencias en el potencial de masa o por tensiones en modo común.

**Multiplexado** - El multiplexado es la conmutación de las entradas del convertidor, de modo que con un sólo convertidor podemos medir los datos de diferentes canales de entrada. Puesto que el mismo convertidor está midiendo diferentes canales, su frecuencia máxima de conversión será la original dividida por el número de canales muestreados.

**Filtrado** - El fin del filtro es eliminar las señales no deseadas de la señal que estamos observando.

**Excitación** - La etapa de acondicionamiento de señal a veces genera excitación para algunos transductores, como por ejemplos las galgas "extesométricas", "termistores" o "RTD", que necesitan de la misma, bien por su constitución interna, (como el termistor, que es una resistencia variable con la temperatura) o bien por la configuración en que se conectan (como el caso de las galgas, que se suelen montar en un puente de Wheatstone).

**Linealización** - Muchos transductores, como los termopares, presentan una respuesta no lineal ante cambios lineales en los parámetros que están siendo medidos. Aunque la linealización puede realizarse mediante métodos numéricos en el sistema de adquisición de datos, suele ser una buena idea el hacer esta corrección mediante circuitería externa.

### 2.5.2.- Especificaciones de la tarjeta DAQ que se utilizara

# Specifications — NI 671x, NI 673x

These specifications are typical at 25 °C unless otherwise stated.

## Analog Output

### Output Characteristics

Number of channels
- NI 6715/6713/6733 ..... 8 voltage outputs
- NI 6711/6731 ..... 4 voltage outputs

Resolution ..... 12 bits, 1 in 4,096 (NI 671x), 16 bits, 1 in 65,536 (NI 673x)

| | Maximum Update Rate (NI 671x/673x) | | Max Update Rate (NI 6715) | |
|---|---|---|---|---|
| Number of Channels | Using Local FIFO (kS/s)[1] | Using Host Memory (kS/s)[2] | Using Local FIFO (kS/s)[1] | Using Host Memory (kS/s)[2] |
| 1 | 1,000 | 1,000 | 1,000 | 800 |
| 2 | 1,000 | 1,000 | 850 | 400 |
| 3 | 1,000 | 1,000 | 750 | 266 |
| 4 | 1,000 | 1,000 | 650 | 200 |
| 5 | 1,000 | 1,000 | 600 | 160 |
| 6 | 952 | 1,000 | 550 | 133 |
| 7 | 833 | 869 | 510 | 114 |
| 8 | 740 | 769 | 480 | 100 |

[1]These numbers apply to continuous waveform generation, and do not change irrespective of the number of devices in the system. [2]These numbers may change when using more devices or when other CPU or bus activity is taking place.

FIFO buffer size
- NI 6713/6733 ..... 16,384 samples
- NI 6711/6715/6731 ..... 8,192 samples

Data transfers ..... DMA, interrupts, programmed I/O

DMA modes (PXI/PCI only) ..... Scatter-gather

### Voltage Output

Ranges ..... ±10.0 V, ±AO EXT REF

Output coupling ..... DC

Protection ..... Short-circuit to ground

## Digital I/O

Number of channels ..... 8 input/output

Compatibility ..... 5 V TTL/CMOS

Power-on state ..... Input (high-impedance)

Data transfers ..... Programmed I/O, DMA (NI 673x), interrupts (NI 673x)

Input buffer ..... 2048 B (NI 673x)

Output buffer ..... 2048 B (NI 673x)

Transfer rate ..... 10 Mwords/s (NI 673x)

## Timing I/O

### General-Purpose Up/Down Counter/Timers

Number of channels ..... 2

Resolution ..... 24 bits

Compatibility ..... 5 V TTL/CMOS

| Level | Minimum | Maximum |
|---|---|---|
| Input low voltage | 0 V | 0.8 V |
| Input high voltage | 2 V | 5 V |
| Output low voltage ($I_{out}$ = 5 mA) | — | 0.4 V |
| Output high voltage ($I_{out}$ = -3.5 mA) | 4.35 V | — |

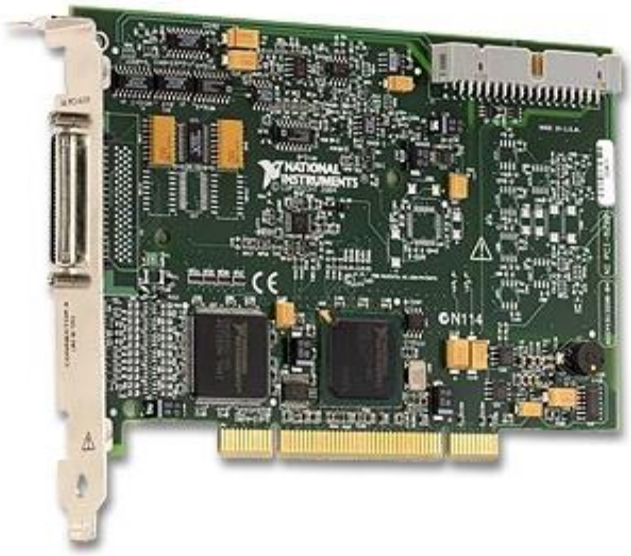
| | |
|---|---|
| Digital logic levels | |
| Base clocks available | 20 MHz and 100 kHz |
| Data transfers | DMA (except DAQCard-6715), interrupts, programmed I/O |
| DMA modes | Scatter-gather |

**Digital Trigger**

| | |
|---|---|
| Purpose | |
| Analog output | Start trigger, gate, clock |
| General-purpose counter/timers | Source, gate |
| Source | (except NI 6715) |
| PCI | RTSI <0...6> |
| PXI | PFI <0...9> |
| Slope | Positive or negative; software-selectable |
| Compatibility | 5 V TTL/CMOS |

**Materiales que se utilizaron para la DAQ**

*SH68-68-EP*
*Shielded Cable*




*Tarjeta DAQ NI6713*

*Conector SCB-68 externo para la tarjeta DAQ*

| | | | |
|---|---|---|---|
| AO_GND | 34 | 68 | NC |
| NC | 33 | 67 | AO_GND |
| AO_GND | 32 | 66 | AO_GND |
| AO_GND | 31 | 65 | AO_7¹ |
| AO_6¹ | 30 | 64 | AO_GND |
| AO_GND | 29 | 63 | AO_GND |
| AO_5¹ | 28 | 62 | NC |
| AO_GND | 27 | 61 | AO_GND |
| AO_GND | 26 | 60 | AO_4 |
| AO_3 | 25 | 59 | AO_GND |
| AO_GND | 24 | 58 | AOGND |
| AO_GND | 23 | 57 | AO_2 |
| AO_0 | 22 | 56 | AO_GND |
| AO_1 | 21 | 55 | AO_GND |
| EXTREF | 20 | 54 | AO_GND |
| P04 | 19 | 53 | D_GND |
| D_GND | 18 | 52 | DIO0 |
| P01 | 17 | 51 | P05 |
| P06 | 16 | 50 | D_GND |
| D_GND | 15 | 49 | DIO2 |
| +5 V | 14 | 48 | P07 |
| D_GND | 13 | 47 | P03 |
| D_GND | 12 | 46 | NC |
| PFI_0 | 11 | 45 | EXTSTROBE |
| PFI_1 | 10 | 44 | D_GND |
| DBND | 9 | 43 | PFI_2 |
| +5 V | 8 | 42 | PFI_3/ CTR_1_SOURCE |
| D_GND | 7 | 41 | PFI_4/ CTR_1_GATE |
| PFI_5/AO_SAMP_CLK | 6 | 40 | CTR_1_OUT |
| PFI_6/AO_START_TRIG | 5 | 39 | D_GND |
| D_GND | 4 | 38 | PFI_7 |
| PFI_9/CTR_0_GATE | 3 | 37 | PFI_8/CTR_0_SOURCE |
| CTR_0_OUTT | 2 | 36 | D_GND |
| FREQ_OUT | 1 | 35 | D_GND |

¹ No Connect on 6711 or 6731

*Figure 1. NI 671x and NI 673x I/O Connector*

# 2.6.- TARJETA GPIB

El Hewlett-Packard Instrument Bus (HP-IB) es un estándar bus de datos digital de corto rango desarrollado por Hewlett-Packard en los años 1970 para conectar dispositivos de test y medida (por ejemplo multímetros, osciloscopios, etc) con dispositivos que los controlen como un ordenador. Otros fabricantes copiaron el HP-IB, llamando a su implementación General-Purpose Instrumentation Bus (GP-IB). En 1978 el bus fue estandarizado por el Institute of Electrical and Electronics Engineers (IEEE) como el IEEE-488 (488.1).

El IEEE-488 permite que hasta 15 dispositivos inteligentes compartan un simple bus paralelo de 8 bits, mediante conexión en cadena, con el dispositivo más lento determinando la velocidad de transferencia. La máxima velocidad de transmisión está sobre 1 Mbps en el estándar original y en 8 Mbps con IEEE-488.1-2003 (HS-488).

Las 16 líneas que componen el bus están agrupadas en tres grupos de acuerdo con sus funciones: 8 de bus de datos, 3 de bus de control de transferencia de datos y 5 de bus general. Algunas de ellas tienen retornos de corriente común y otras tienen un retorno propio, lo que provoca un aumento del número de líneas totales (8 masas).

### 2.6.1.- Conceptos generales sobre GPIB

El bus GPIB fue inventado por Hewlett Packard a finales de los años 1960. La intención era crear un bus fiable, especialmente diseñado para conectar computadoras e instrumentos en una configuración de red que poseyera las características requeridas por un equipo de medida.

El control remoto de los instrumentos es un aspecto relevante del bus, pero hay otros más importantes como el reconocimiento de recepción de datos (“data hardware handshake”), que dota a las operaciones de fiabilidad; o la capacidad de respuesta en tiempo real.

El principal objetivo del bus GPIB consiste en gestionar la transferencia de información entre dos o más dispositivos. Antes de enviar los datos hacia los dispositivos (instrumentos conectados al bus) éstos deben configurarse de acuerdo con este protocolo de transmisión de información. Entre los parámetros relativos al protocolo se encuentra la asignación de direcciones a los instrumentos interconectados.

La numeración del dispositivo, o asignación de su dirección, se realiza desde el panel frontal o alterando la conexión de los puentes de su tarjeta interfaz, que suele ser accesible desde la parte posterior del instrumento.

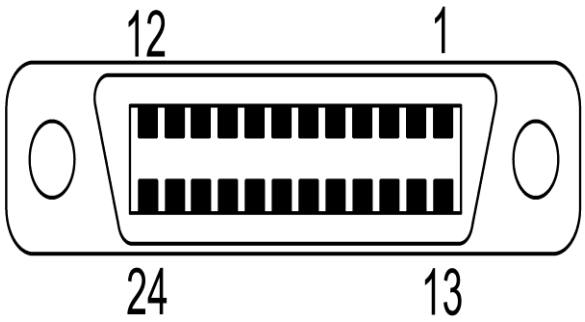
El elemento controlador del equipo GPIB es único (generalmente la tarjeta controladora instalada en un PC, en cuyo caso se le asigna la dirección 0), supervisa todas las operaciones que se realizan en el bus, y determina el dispositivo que envía la información y el momento en que se realiza su envío. El controlador puede designar un sustituto si en un determinado momento no puede atender los requisitos de control. El nuevo controlador recibe el nombre de controlador activo.

El controlador asegura que no puede haber dos o más instrumentos enviando información al bus simultáneamente. Además, establece los dispositivos que permanecen en estado de recepción o escucha, ya que no todos los instrumentos están siempre interesados en captar la información del bus. Esta función la realiza "despertando" a los dispositivos en estado de "latencia" mediante una solicitud de reafirmación, y mediante órdenes que especifican los nuevos receptores y el nuevo emisor.

Cuando el proceso de transmisión-recepción ha finalizado, el controlador del equipo se asegura de que todos los receptores han recibido la información enviada al bus por el emisor mediante el "data hardware handshake" o control de transferencia de datos. Este protocolo permite asegurar la recepción de la información por parte de los dispositivos más lentos. Como consecuencia, el dispositivo más lento limita la velocidad de operación del equipo GPIB.

En resumen, se consideraran los siguientes elementos o conceptos más relevantes y específicos, involucrados en un equipo red de instrumentación mediante el protocolo GPIB:

- Controlador del equipo controlador activo.
- Dispositivos conectados al bus.
- Dispositivo fuente.
- Dispositivos destino.
- Comandos y funciones.



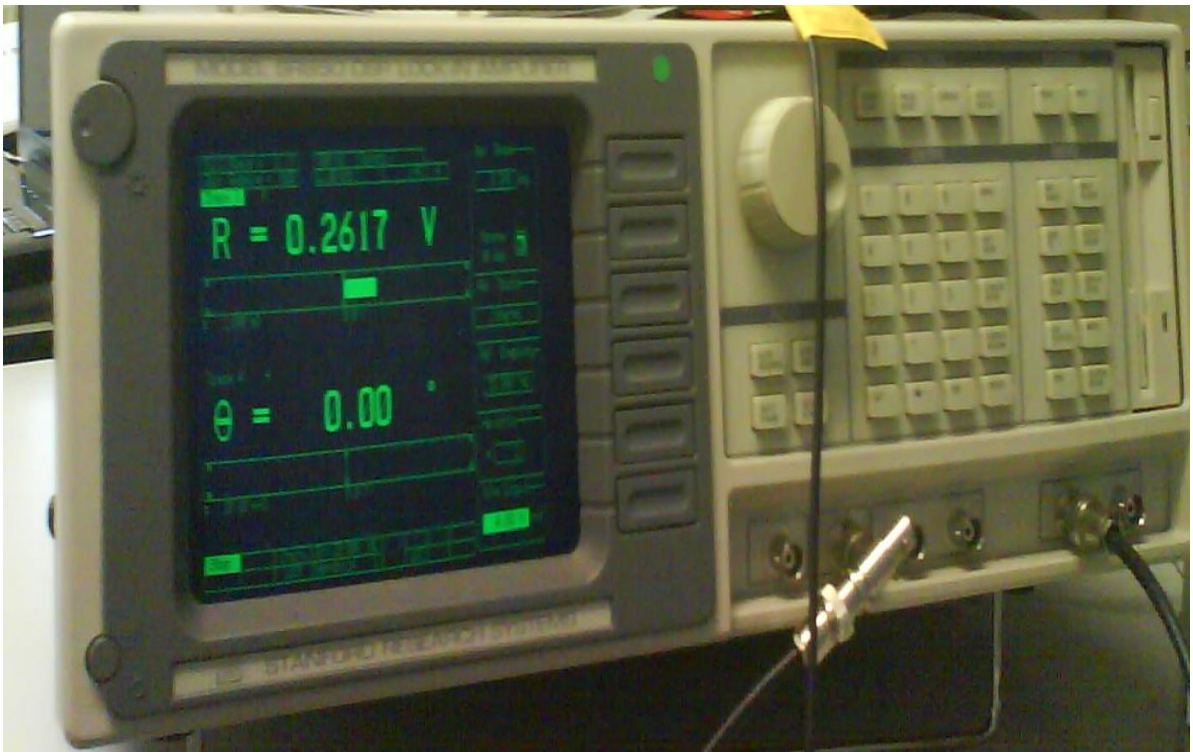

| | | |
|---|---|---|
| **Pin 1** | DIO1 | Data input/output bit. |
| **Pin 2** | DIO2 | Data input/output bit. |
| **Pin 3** | DIO3 | Data input/output bit. |
| **Pin 4** | DIO4 | Data input/output bit. |
| **Pin 5** | EOI | End-or-identify. |
| **Pin 6** | DAV | Data valid. |
| **Pin 7** | NRFD | Not ready for data. |
| **Pin 8** | NDAC | Not data accepted. |
| **Pin 9** | IFC | Interface clear. |
| **Pin 10** | SRQ | Service request. |
| **Pin 11** | ATN | Attention. |
| **Pin 12** | SHIELD | |
| **Pin 13** | DIO5 | Data input/output bit. |
| **Pin 14** | DIO6 | Data input/output bit. |
| **Pin 15** | DIO7 | Data input/output bit. |
| **Pin 16** | DIO8 | Data input/output bit. |
| **Pin 17** | REN | Remote enable. |
| **Pin 18** | GND | (emparejado con DAV) |
| **Pin 19** | GND | (emparejado con NRFD) |
| **Pin 20** | GND | (emparejado con NDAC) |
| **Pin 21** | GND | (emparejado con IFC) |
| **Pin 22** | GND | (emparejado con SRQ) |
| **Pin 23** | GND | (emparejado con ATN) |

# 2.7.- LOCK-IN

Los amplificador lockin son instrumentos diseñados para recobrar señales que están inmersas en el ruido. Estos requieren una señal de referencia igual a la frecuencia de la señal que está siendo medida y después la usa para desmodular la señal de entrada antes del filtrado

Las dos funciones principales que hacen del lock-in algo tan útil es que es capaz de detectar la diferencia de fase entre dos señales y es capaz de atenuar el ruido de una señal débil aunque aquel tenga una intensidad del orden de la propia señal, devolviéndonos una señal continua cuyo valor es el máximo de la señal alterna que estemos midiendo.

El lock-in detecta una señal proveniente del experimento y la compara con una señal de referencia que el mismo es capaz de generar. Esta señal de referencia puede ser usada solo como tal o ser útil.

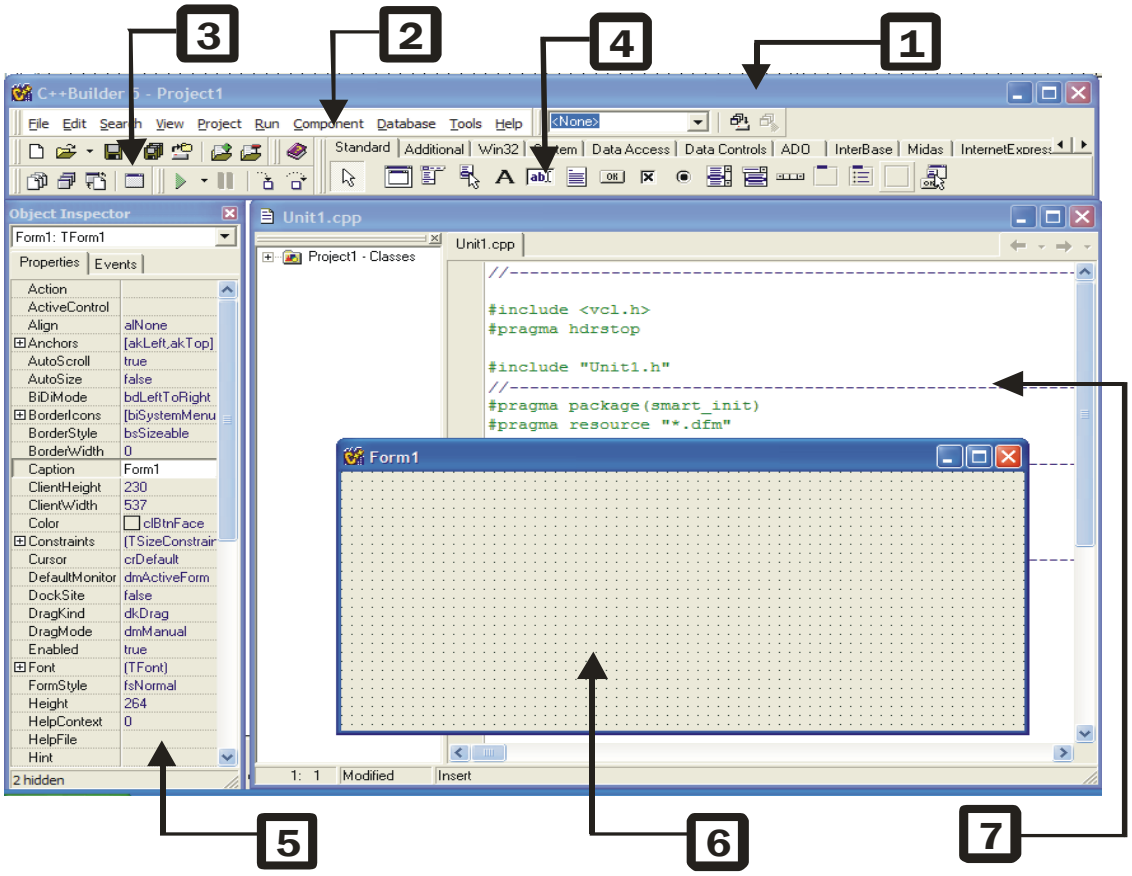


# 2.8.- BUILDER C++

C++ Builder es un flexible, simple y potente compilador que permite usar cualquiera de los tres enfoques de programación en la resolución de problemas en las que deba ser utilizada la computadora. Con Builder C++ se pueden realizar aplicaciones sencillas de entender al usuario y que además tengan una gran eficiencia en el control de procesos tanto internos como externos, ya que cuenta con un gran número de componentes prefabricados que facilitan de forma notable la creación de cualquier aplicación.

### 2.8.1.- Entorno del lenguaje C++ Builder

A continuación se muestra el aspecto inicial de Builder C++, así como el nombre y descripción de cada parte que lo compone:



1. VENTANA PRINCIPAL

La mayor parte de las opciones de trabajo y configuración se encuentran accesibles en la ventana principal de C++ Builder, que por regla general, está en la parte superior de la pantalla, de extremo izquierdo a derecho. En la Barra de título podemos ver el nombre del proyecto sobre el que estamos trabajando.

2. MENÚ DE OPCIONES

Este elemento, común en la práctica de aplicaciones Windows, contiene las diferentes opciones del programa agrupadas en varias principales. (File, Edit, Search, View, Project, Run, Component, etc.

3. PALETA DE BOTONES O ACCIONES RÁPIDAS

Nos permite ejecutar de forma directa algunas de las opciones del menú sin necesidad de pasar por las distintas ventanas, bastando la pulsación de un botón. Mediante ellos podremos recuperar un proyecto, guardar el actual, ejecutar el programa, etc.

4. PALETA DE COMPONENTES

Desde esta paleta podemos seleccionar los elementos que vamos a insertar en un formulario.

5. EL INSPECTOR DE OBJETOS

Su finalidad es facilitar la edición de las propiedades y eventos correspondientes al componente que se seleccione.

6. EL FORMULARIO

Es la ventana del entorno Windows en la que se alojarán los distintos controles encargados de interactuar con el usuario. Este elemento también es conocido como ficha.

7. EDITOR DE CÓDIGO

El código asociado a la respuesta a eventos se escribe en esta ventana. Por ejemplo, las tareas que realizará el programa cuando se presione un botón (evento), se escribirán en código C++ en un área preasignada en este editor de código.

# CAPITULO III

# PROCEDIMIENTO Y DESCRIPCION DE LAS ACTIVIDADES REALIZADAS

## 3.1.- Pasos necesarios para la programación de e inicialización de la tarjeta DAQ

El siguiente diagrama a bloques de la fig.3.1.1 hace representación de los pasos necesarios que se requieren para inicializar la tarjeta DAQ y para la programación de la misma. Ya que se necesitan,

- Tener los drivers instalados ya que son necesarios para que la computadora reconozca el dispositivo o la tarjeta.

- Instalar la tarjeta DAQ a la computadora.

- Copiar las librerías de la tarjeta al proyecto que se va a crear ya que son necesarias porque son los comandos que la tarjeta utiliza para comunicarse con el dispositivo y con la computadora y estas aparecen al momento de instalar el software de la tarjeta.

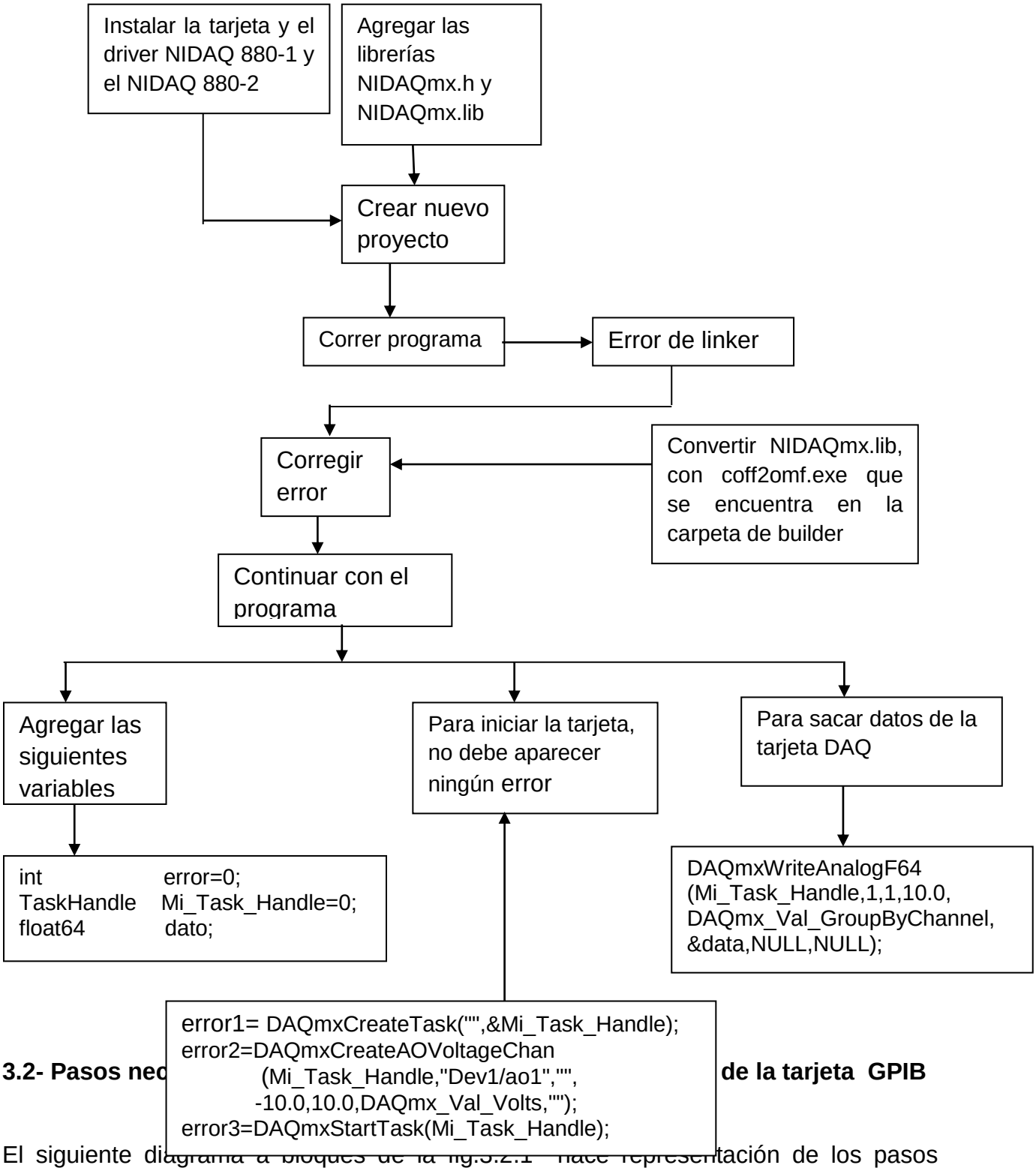
- Corregir un error de compatibilidad que surge con C++ Builder y corresponde a una librería que se tiene que convertir utilizando un builder y MSDOS.

- Parte del código que se necesita para realizar el programa para facilitar la programación (opcional).

- Las variables necesarias para la programación de la misma de igual forma sirve para facilitar la programación (opcional).

En la parte anexa del trabajo se explica con detalle un pequeño manual para la programación e inicialización de la misma ya que con esto se facilitara la programación de la tarjeta DAQ y algunas variables que se utilizaron en el programa.

### 3.1.2- Diagrama a bloque para la instalación de la tarjeta DAQ




### 3.2- Pasos nec de la tarjeta GPIB

El siguiente diagrama a bloques de la fig.3.2.1 hace representación de los pasos necesarios que se requirieron para inicializar la tarjeta GPIB y para la programación de la misma. Algunos de los pasos son los que se presentan a continuación:

- Tener los drivers instalados ya que son necesarios para la comunicación entre la tarjeta, la computadora y el instrumento.

- Instalar la tarjeta GPIB en la computadora

- Copiar las librerías de la tarjeta al proyecto que se va a crear ya que son necesarias porque son los comandos que la tarjeta utiliza para comunicarse con el dispositivo y con la computadora y estas aparecen al momento de instalar el software de la tarjeta.

- Como iniciar la comunicación con el instrumento debido a que, como es un protocolo de comunicación estándar, se tiene que inicializar deacuerdo al estándar establecido

- Indicarle al instrumento el numero que le corresponde ya que una sola tarjeta puede manejar hasta 15 instrumento diferentes

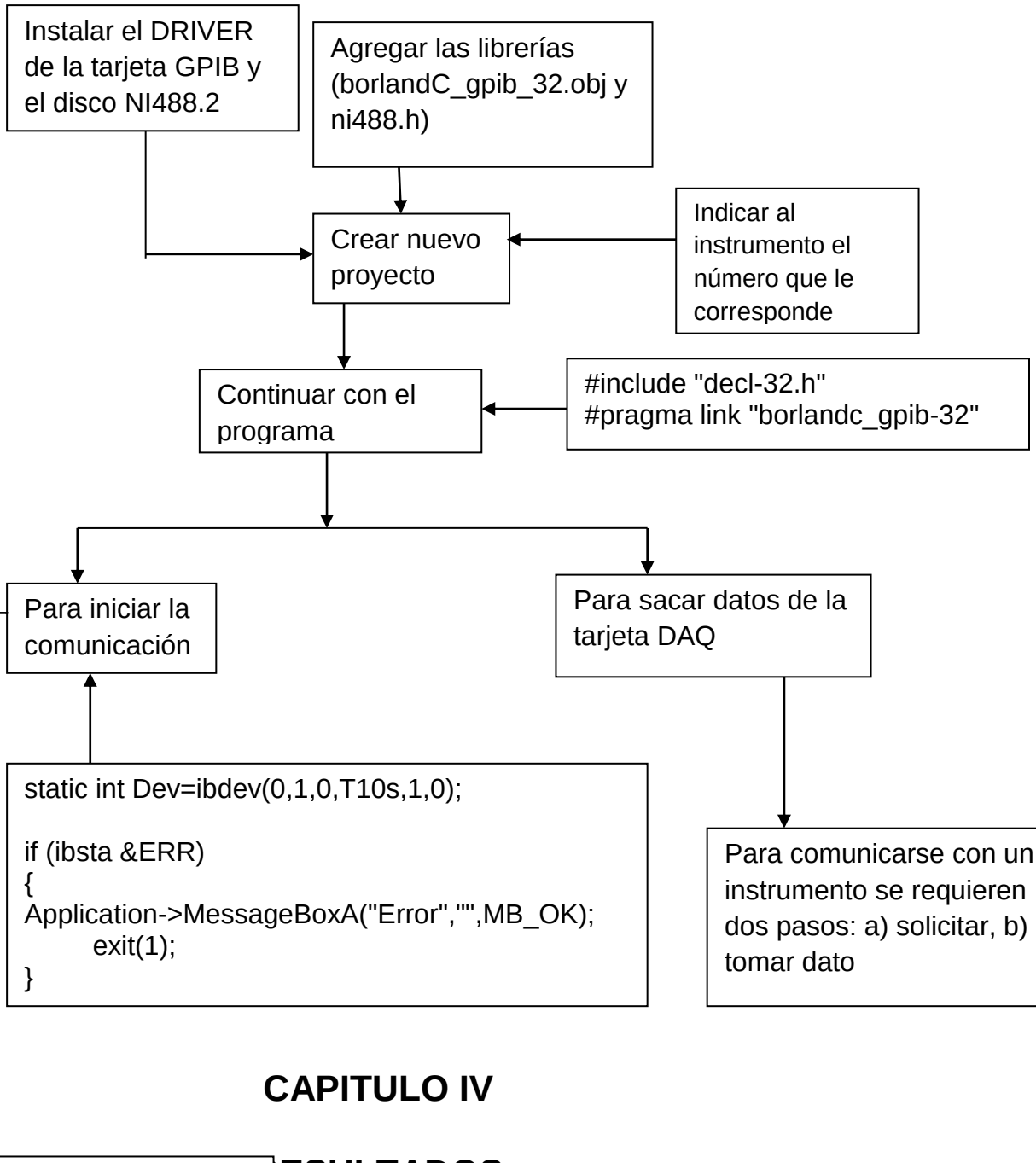
- Parte del código que se necesita para realizar el programa para facilitar la programación (opcional).

- Las variables necesarias para la programación de la misma de igual forma sirve para facilitar la programación (opcional)

- Y como finalizar la comunicación entre la tarjeta y el instrumento que se esta utilizando ya que es necesario en el protocolo de comunicación.

En la parte anexa del trabajo se explica con detalle un pequeño manual para la programación e inicialización de la misma ya que con esto se facilitara la programación de la tarjeta GPIB y algunas variables que se utilizaron en el programa.

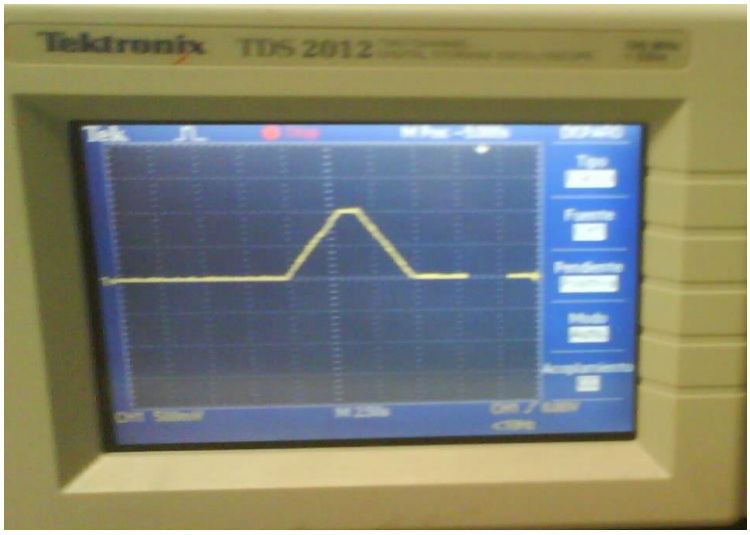
### 3.2.1.- Diagrama a bloques para la instalación de la tarjeta GPIB



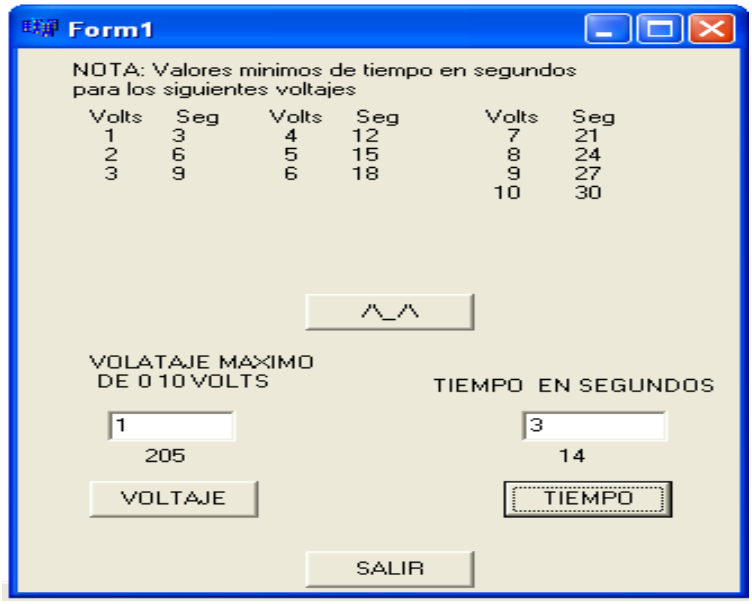

# CAPITULO IV

# RESULTADOS

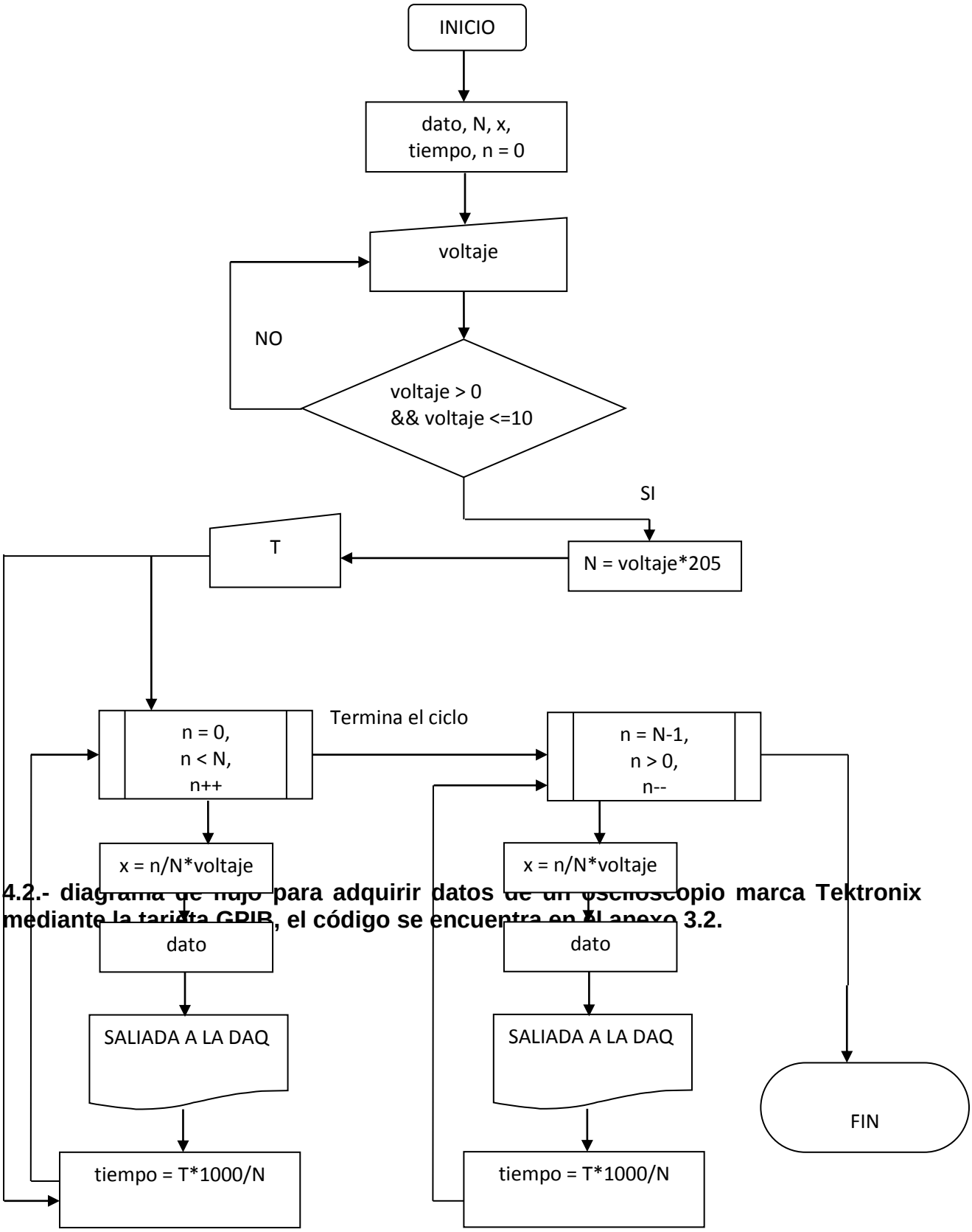
**4.1.- A… el diagrama de flujo de la programación de la tarjeta … crear una señal triangular donde se podrá variar la amp… esta tardara en subir y bajar, el código del programa se encuentra en el anexo 3.1 del trabajo.**




**Señal que se genera la tarjeta DAQ**




**Vista del programa para la tarjeta DAQ**




**4.2.- diagrama de flujo para adquirir datos de un osciloscopio marca Tektronix mediante la tarjeta GPIB, el código se encuentra en el anexo 3.2.**

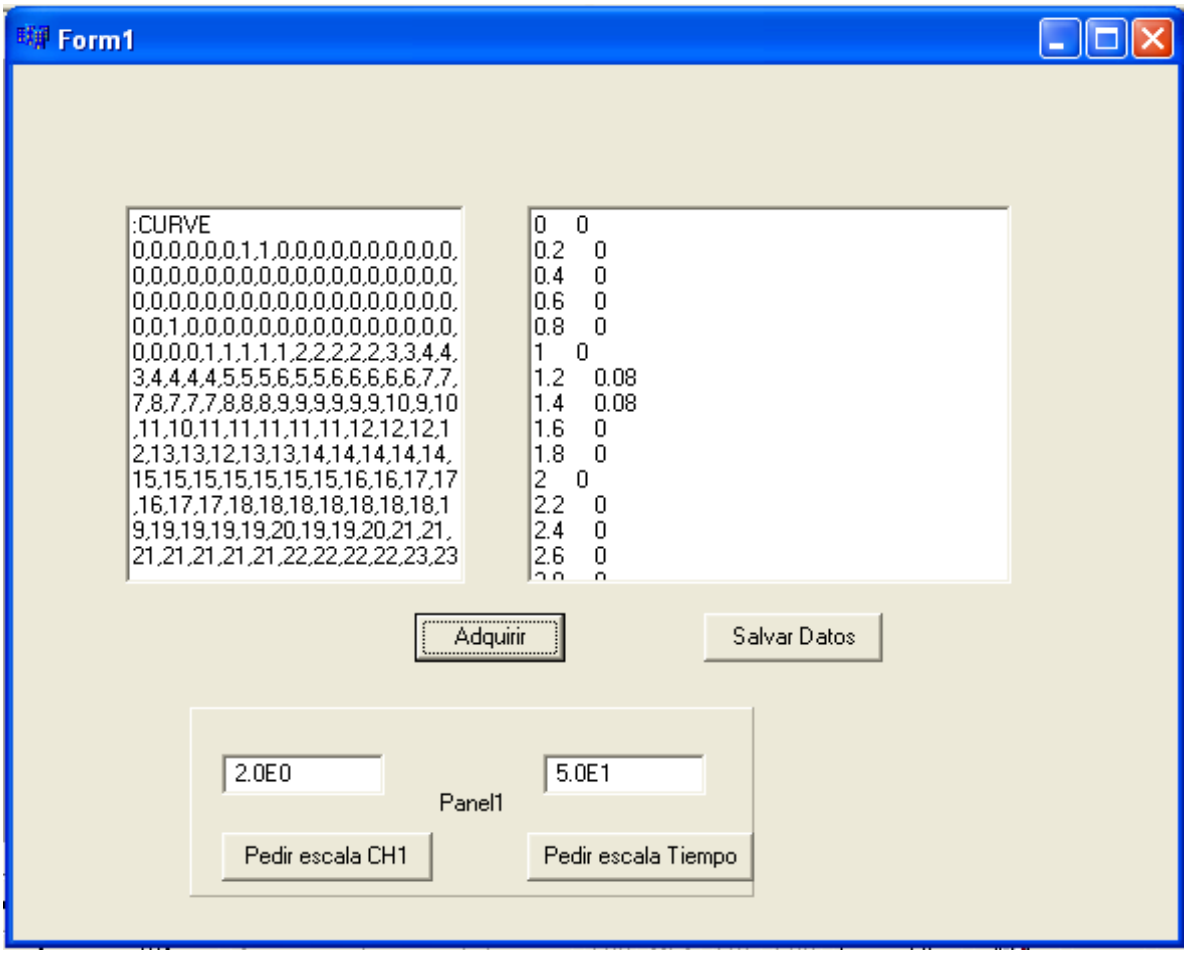
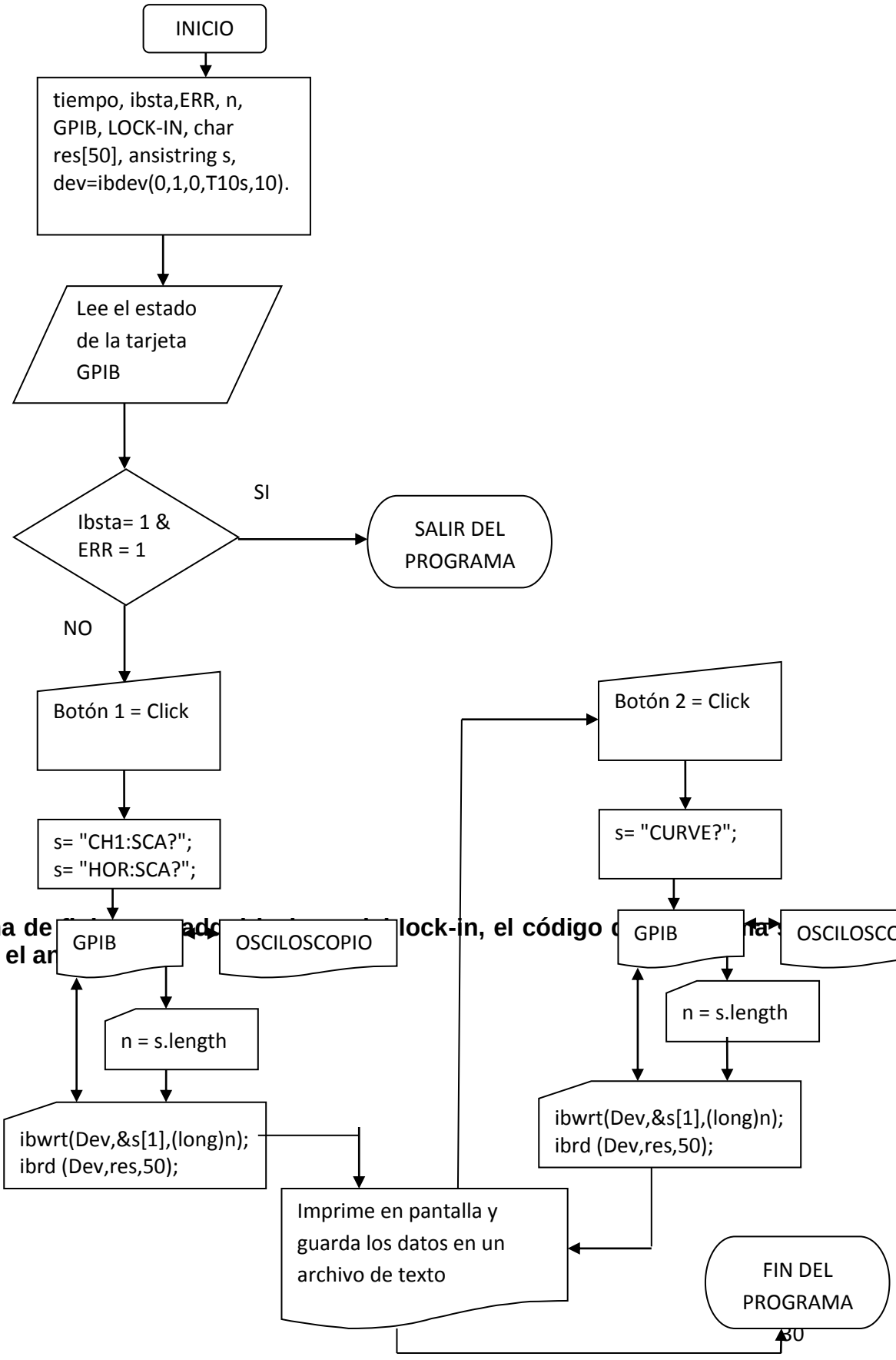
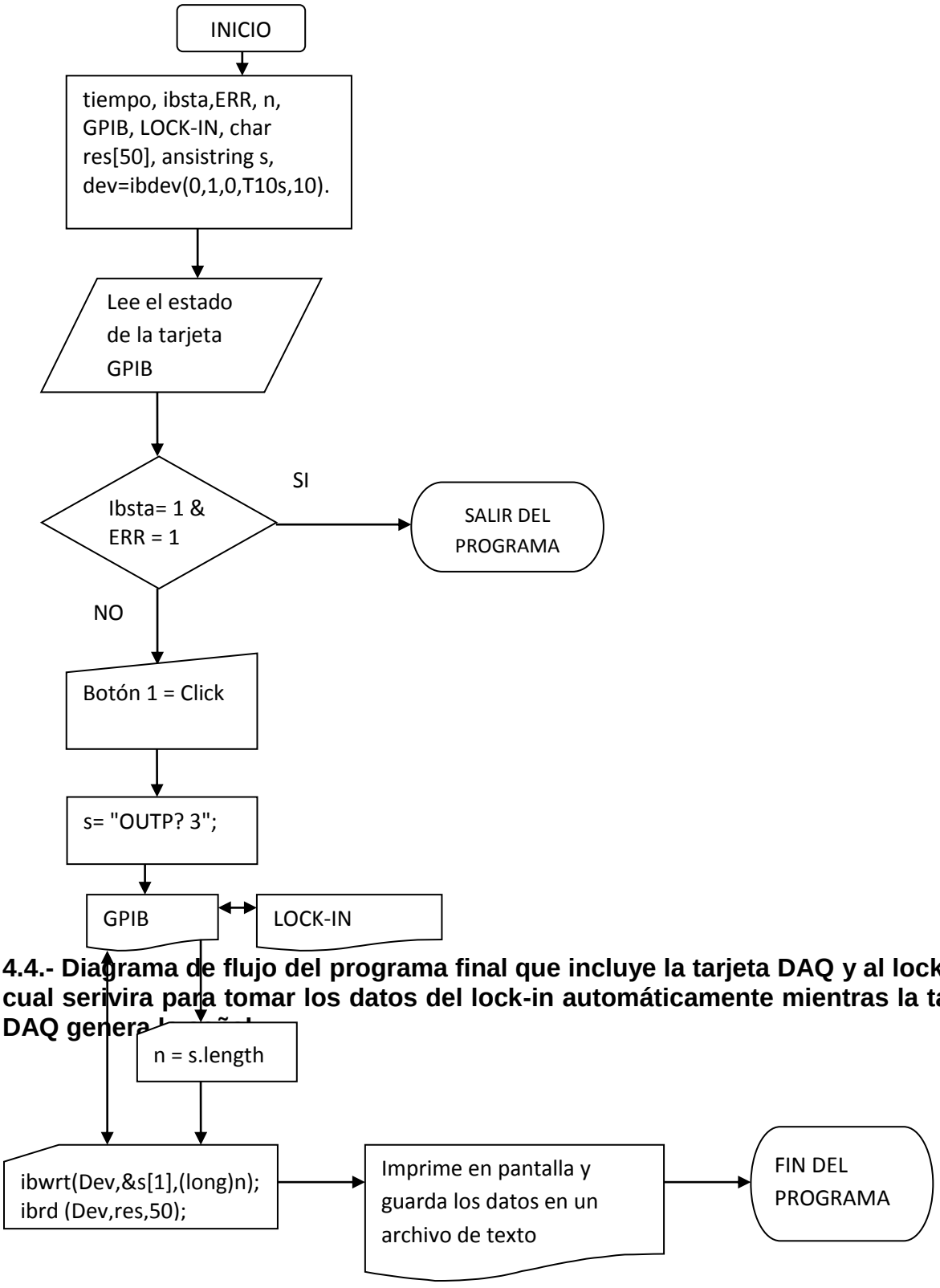



**Imagen del programa para adquirir datos de un osciloscopio digital**

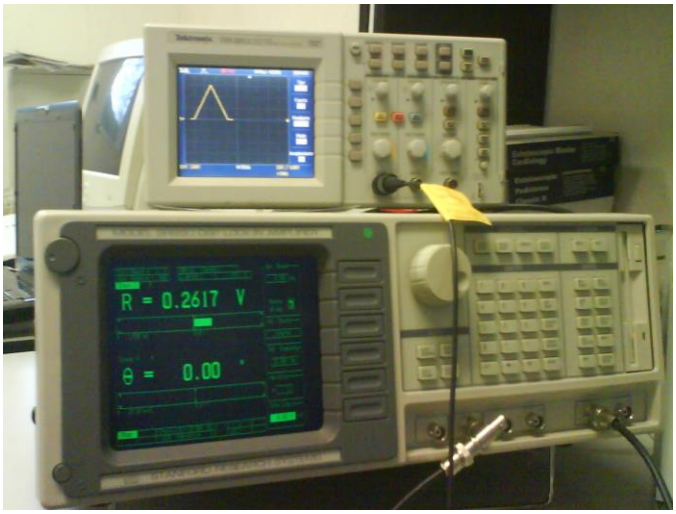
En la página siguiente se muestra el diagrama de flujo del programa:



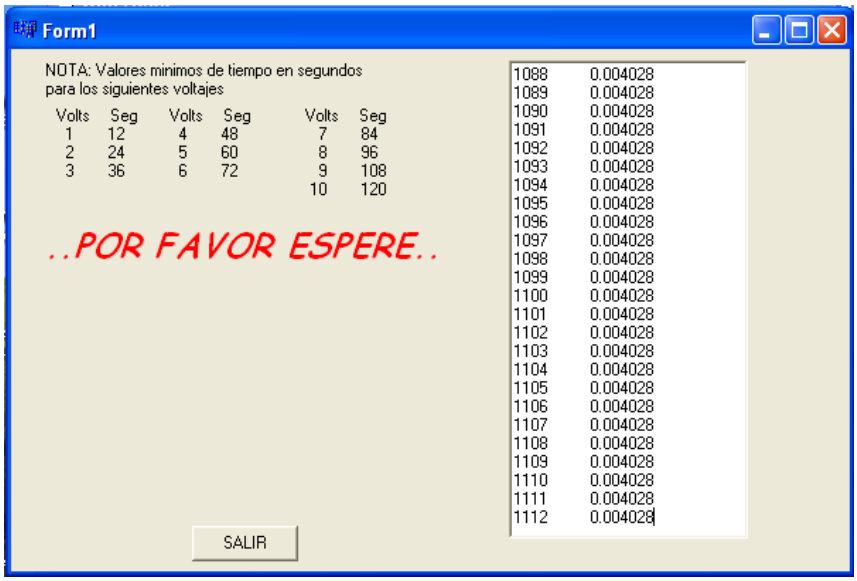

**4.3.- Diagrama de [illegible] lock-in, el código [illegible] encuentra en el ar[illegible]**



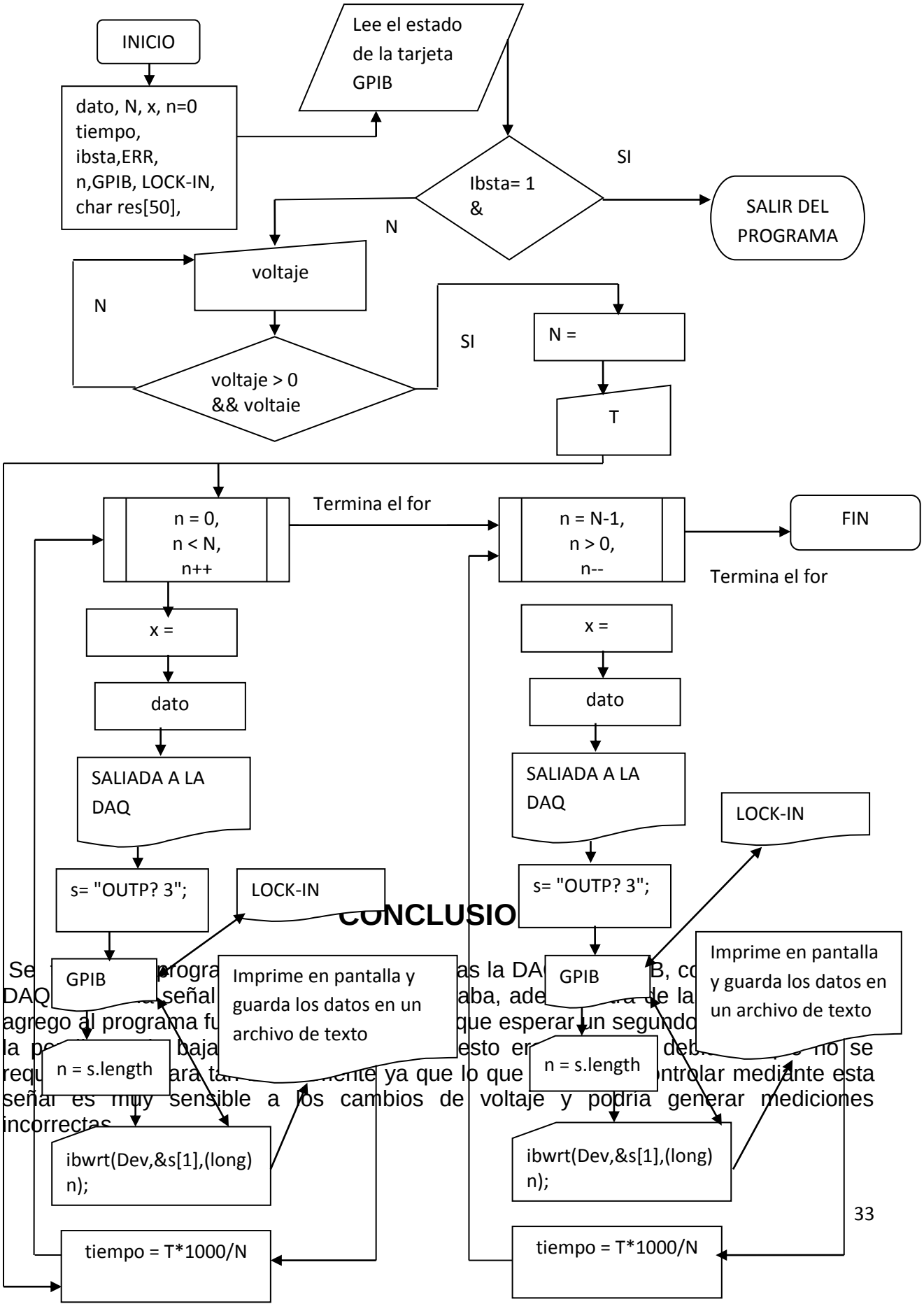

**4.4.- Diagrama de flujo del programa final que incluye la tarjeta DAQ y al lock-in el cual serivira para tomar los datos del lock-in automáticamente mientras la tarjeta DAQ genera la señal**




**En el osciloscopio se muestra la señal generada por la tarjeta DAQ, debajo del osciloscopio se encuentra el lock-in del cual cada pulso que genera la DAQ el programa está tomando datos provenientes del lock-in los cuales pertenecen a alguna señal externa.**




**Aquí podemos observar que mientras se está generando la señal atreves de la tarjeta DAQ también se están tomando los datos provenientes del lock-in.**




## CONCLUSIO

Se ... progra... as la DA... B, co... DAQ ... señal ... aba, ade... la de la ... agrego al programa fu... que esperar un segundo ... la po... baja ... esto er... debi... no se requ... ara ta... ente ya que lo que ... ontrolar mediante esta señal es muy sensible a los cambios de voltaje y podria generar mediciones incorrectas.

Otra de las cosas que se observa es que el programa marca unos valores mínimos de tiempo para cada voltaje que se le aplica esto se debe a que la tarjeta la GPIB que controla la toma de datos del lock-in hace que consuma tiempo, lo que se quiere decir con esto es que cada ves que nosotros le decimos al programa que queremos 1V de salida o una amplitud de la señal triangular de 1V, la tarjeta DAQ genera 205 veces 1 voltaje de 4.88mV para completar 1V, entonces el lock-in toma la misma cantidad de datos que la tarjeta DAQ genera para 1V, por lo tanto cada ves que se genera 4.88mv de la DAQ el lock-in toma un dato y asi sucesivamente hasta que la tarjeta DAQ termine la señal triangular completa.

El numero de datos que se toman corresponde a una división de el voltaje que queremos que nos genere entre 4.88mv.

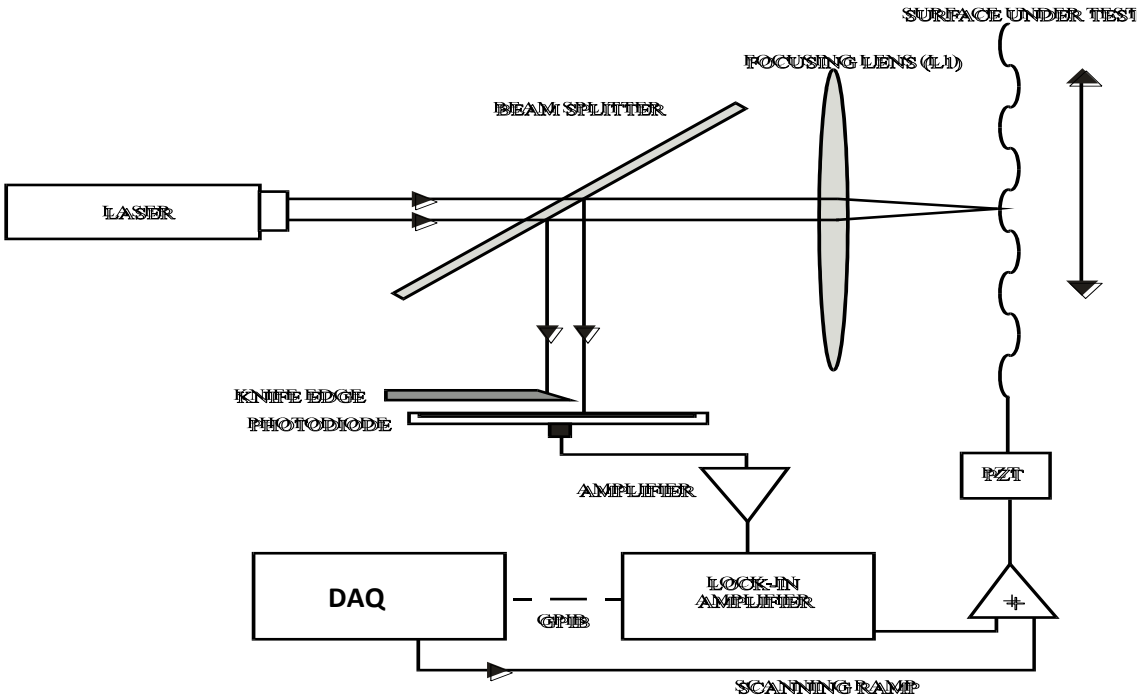
La programación del osciloscopio mediante la tarjeta GPIB se utilizo para capturar unas señales que se midieron de un amplificador pero no corresponden al proyecto en si, más que nada fue como una practica para comenzar a programar el lock-in.

Al final se logro la terminación del programa de ambas tarjetas y se realizaron algunas pruebas con una señal externa que se introdujo al lock-in, el lock-in se mantuvo a la frecuencia que le asignamos o la frecuencia que tenia de referencia y solo tomo datos de la amplitud de solo esa frecuencia que correspondían a la variación del voltaje que se le hacia a esa señal.

Debido a la falta de tiempo no se pudo llevar a cabo el experimento con el interferómetro pero con este programa se trata de automatizar la toma de dato irregularidades de un objeto.

# ANEXOS

## 1) APLICACIÓN DE LA PROGRAMACIÓN DE LA TARJETA DAQ Y GPIB



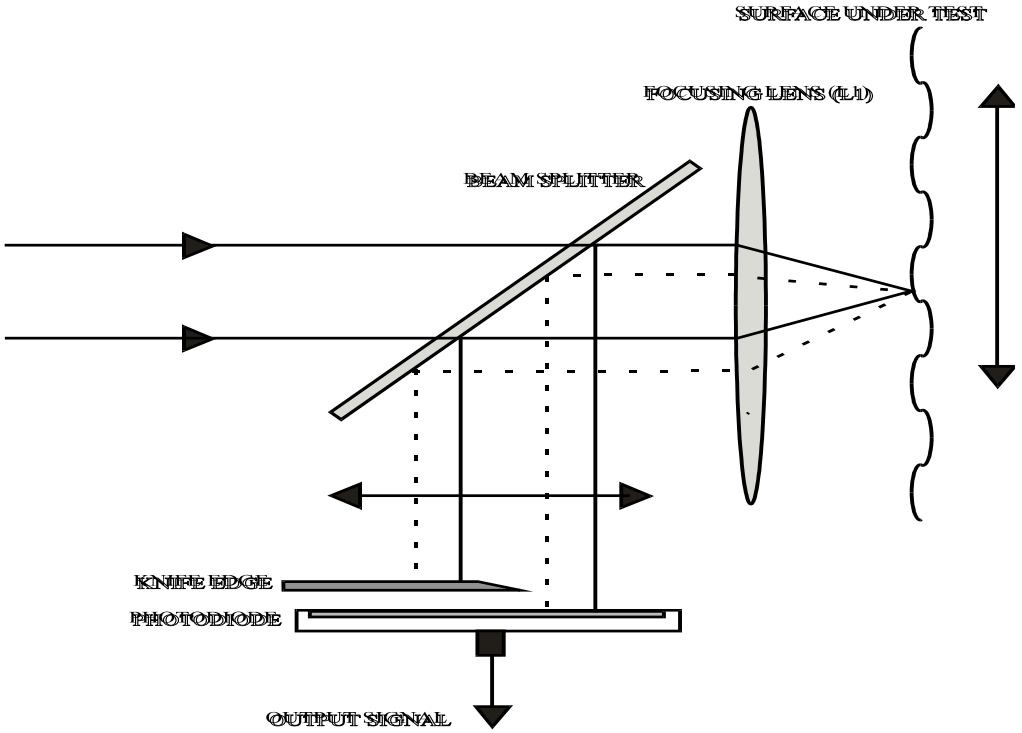

**Módulo de Adquisición con el Sistema de Borde de Navaja**

En el esquema (arriba) podemos observar para que servirá la programación de las tarjetas a continuación se explica el funcionamiento o aplicación que tendrá la programación de ambas tarjetas.

En el esquema se observa el haz del laser y la trayectoria que sigue, este pasa por un divisor de haz que divide el haz del laser en 2 haces uno que se va al objeto pasando por una lente, esta lente sirve para enfocar el haz del laser contra el objeto, el haz que pasa por la lente es reflejado de vuelta hacia el divisor de haz que nuevamente se divide en dos haces pero el que nos interesa es el que regresa al fotodiodo ahí podemos observar que el fotodiodo está cubierto hasta la mitad por una navaja con la intensión de que el fotodiodo reciba la mitad de la luz que le es llegada del haz reflejado es decir el fotodiodo entregara solo la mitad de voltaje que si estuviera totalmente descubierto, ejemplo si suponemos que el fotodiodo no esta cubierto por la navaja y el haz de laser reflejado le llega directamente al fotodiodo, este entregara 10volts de salida, pero si cubrimos la mitad del fotodiodo con la navaja y el haz de laser

lo posicionamos exactamente a la mitad entre la navaja y el fotodiodo la salida que obtendremos será de solamente 5 volts.

El lock-in generara una señal de referencia de forma senoidal entre 23 y 30hertz que se sumara mediante un OPAM a la señal que generara la tarjeta DAQ, la salida del OPAM será enviada a un nanoposicionador que se desplaza 1micra por cada 1v que se le aplica, el nanoposionador como se observa está sosteniendo un objeto (que en esta esquema esta exageradamente ampliado ya que las ondulaciones que se observan son de micras y corresponden a las deformaciones del objeto pero que por razones de explicación fueron ampliadas) este objeto debido a la señal que genera la suma de la DAQ con la de referencia que genera el lock-in harán que el objeto se desplace hacia arriba y al mismo tiempo oscile a la frecuencia que genera el lock-in.




**Sistema de Borde de Navaja (KED)**

En la imagen (arriba) ampliada se observa que el haz del laser mediante una lente es enfocada al objeto y es reflejada por el mismo objeto, como se había mencionado anteriormente solo la mitad de luz del laser es la que le llega al fotodiodo entonces al mover el objeto con el nanoposicionador este se va desplazando y oscila al mismo tiempo. Debido a que el objeto presenta deformaciones en su superficie el haz de laser es reflejado en diferente ángulo.

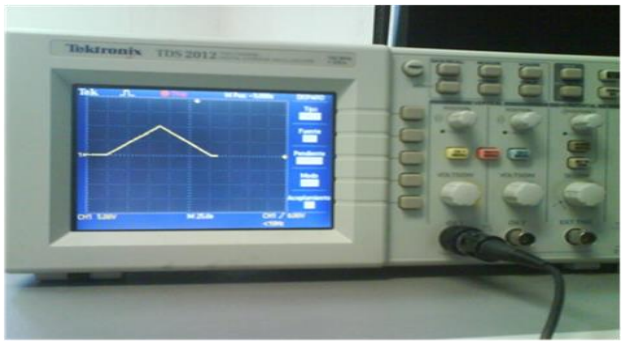
Como se observa en la imagen la línea punteada corresponde al haz del laser reflejado pero debido a un pequeño desplazamiento del objeto y alas irregularidades del mismo el haz del laser se refleja en diferente ángulo lo que provoca que en este caso la

cantidad de luz que le llega al fotodiodo sea menor a la que tenia antes de que el objeto comenzara a desplazarse (línea continua), como la cantidad que se refleja en el fotodiodo es menor a la que tenia anteriormente y como al mismo tiempo que se desplaza el objeto esta oscilando el lock-in, como ya se había mencionado anteriormente solo captura los cambios de voltaje que le ocurren a la frecuencia de referencia, que en este caso esta entre 23 y 30 hertz y elimina cualquier otra frecuencia que este interfiriendo (ruido) y es aquí donde se requiere de la captura de datos del lock-in, lo que hace la tarjeta GPIB es que cada 4.88mv que la tarjeta DAQ genera el lock-in toma el valor que muestre el desplazamiento del haz del laser o dicho en otras palabras toma el voltaje que entrega el fotodiodo debido al ángulo en que se refleja el haz del laser y como al mismo tiempo que se refleja esta oscilando a la frecuencia de referencia el lock-in elimina cualquier otra frecuencia y solo toma el dato del voltaje que esta frecuencia de referencia tenga y almacena el dato.

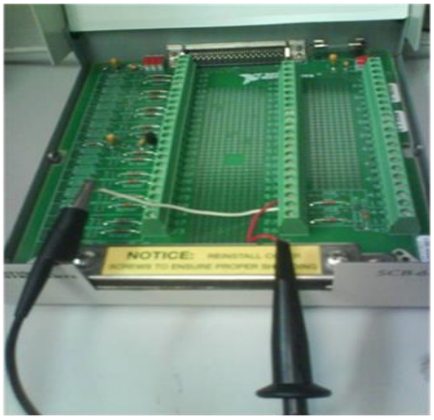
Con esta oscilación se garantiza que el lock-in solo tomara los cambios en amplitud de la señal de referencia a la que el objeto oscila aun cuando hayan ruidos externos ya que el arreglo del interferómetro es demasiado sensible a cualquier movimiento incluso a un soplido.

Si no se presentara esta oscilación el lock-in simplemente no podría tomar ningún valor ya que necesita de una frecuencia de referencia y estos datos que toma son los que servirán para conocer las irregularidades del objeto.

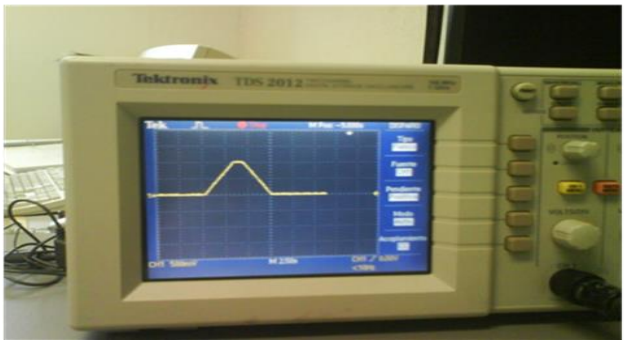
## 2) IMÁGENES TOMADAS DE LA TARJETA DAQ FUNCIONANDO




10v ; 75s




Modulo DAQ




1v ; 7.5s

## 3) CÓDIGO Y PROCEDIMIENTO PARA CREAR LOS PROGRAMAS QUE SE REALIZARON CON LA TARJETA DAQ Y GPIB

### 3.1.- Pasos para la instalación de la tarjeta DAQ (pci- 6713)

1. Instalar el driver NIDAQ 880-1 y el NIDAQ 880-2 y la tarjeta.

2. En el nuevo proyecto que se va a crear se tienen que agregar al proyecto las librerías NIDAQmx.h y NIDAQmx.lib. De preferencia, estas librerías deben ir en el directorio del proyecto, estos archivos se encuentran en la carpeta donde se instalo la tarjeta DAQ.

3. Después se corre el programa aunque no tenga nada escrito más que las librerías agregadas.

4. Al correr el programa se observará un error del LINKER : “Contains invalid OMF record, type 0x21 (Possibly COFF). Entonces se debe convertir el archivo NIDAQmx.lib de la siguiente manera:

   a) copiar el archivo NIDAQqmx.lib a la carpeta del proyecto
   b) copiar el archivo coff2omf.exe que está en builder en la carpeta BIN, a la carpeta del proyecto
   c) convertir el archivo usando la orden: coff2omf nidaqmx.lib nidaqmx1.lib. Esto se hace desde el MS-DOS, esta instrucción creara un archivo nuevo llamado nidaqmx1.lib.
   d) renombrar el nidaqmx1.lib a nidaqmx.lib

5. Una vez hecho esto se puede volver a correr el programa y el error debe desaparecer, para programar la tarjeta DAQ hay que declarar las siguientes TRES variables:

   a) int error=0;
      TaskHandle Mi_Task_Handle=0;
      float64 data;

6. Para usar estas variables y sacar datos por la tarjeta DAQ hay que usar las instrucciones siguientes:

```
error1= DAQmxCreateTask("",&Mi_Task_Handle);
error2=DAQmxCreateAOVoltageChan(Mi_Task_Handle,"Dev1/ao1","",-10.0,10.0,
                                                        DAQmx_Val_Volts,"");
error3=DAQmxStartTask(Mi_Task_Handle);
```

NOTA:

- Con esta inicialización tenemos salidas de –10.00 a 10.00 Volts.

Si algún entero error 1, 2 ó 3 es diferente de cero, algo pasó y no se pudo iniciar correctamente la tarjeta. Si las tres variables error están en cero la tarjeta se inicializo correctamente y podemos proceder a usar la tarjeta.

NOTAS:

- Dev1 se refiere a la primera tarjeta DAQ, por si hubiera más tarjetas DAQ.
- a01 se refiere al Analog Output 1, si se desea por ejemplo el Analog Output 3, se pondría ao3.

Para sacar un dato, una vez ya inicializada la tarjeta, se usa:

DAQmxWriteAnalogF64(Mi_Task_Handle,1,1,10.0,DAQmx_Val_GroupByChannel,&data,NULL,NULL);

Para sacar el dato, hay que darle un valor a dato, por ejemplo, dato=1.35, y luego usar la instrucción. En este ejemplo ANALOG OUTPUT 1 se pondrá en 1.35 Volts y así permanecerá hasta hacerle un cambio

NOTA

- Recordar que las salidas posibles están entre –10.0 y 10.0 Volts

### 3.2.- Programa para la tarjeta DAQ

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
#include "NIDAQmx.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TaskHandle MiTaskHandle=0;
float64 dato;
TForm1 *Form1;
int N;
float voltaje;
int tiempo,T;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
```

```
}

//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  if(voltaje>0&&voltaje<=10)
  {
  N= voltaje*205;
  Form1->Label1->Caption=N;
  }
Form1->Edit2->Visible= true;
Form1->Button4->Visible= true;
Form1->Label3->Visible= true;
Form1->Label5->Visible= true;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
dato=0;
DAQmxWriteAnalogF64(MiTaskHandle,1,1,10.0,
                    DAQmx_Val_GroupByChannel,&dato,NULL,NULL);
exit(0);
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button3Click(TObject *Sender)
{
int n;
float x;

Form1->Edit2->Visible= false;
Form1->Button4->Visible= false;
Form1->Button3->Visible= false;
Form1->Edit1->Visible= false;
Form1->Button1->Visible= false;
Form1->Label1->Visible= false;
Form1->Label2->Visible= false;
Form1->Label3->Visible= false;
Form1->Label5->Visible= false;
Form1->Label4->Visible= true;

  for (n=0;n<N;n++)
   {
    x=(double)n/N*voltaje;
    dato=x;
    DAQmxWriteAnalogF64(MiTaskHandle,1,1,10.0,
```

```
                DAQmx_Val_GroupByChannel,&dato,NULL,NULL);
    Application->ProcessMessages();
    Sleep(tiempo);
   }
   Sleep(1000);

  for (n=N-1;n>0;n--)
   {
    x=(double)n/N*voltaje;
    dato=x;
    DAQmxWriteAnalogF64(MiTaskHandle,1,1,10.0,
                DAQmx_Val_GroupByChannel,&dato,NULL,NULL);
    Application->ProcessMessages();
    Sleep(tiempo);
    }
Form1->Edit1->Visible= true;
Form1->Button1->Visible= true;
Form1->Button3->Visible= true;
Form1->Label4->Visible= false;
Form1->Label1->Visible= true;
Form1->Label2->Visible= true;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::FormCreate(TObject *Sender)
{
int error1, error2, error3;

Form1->Label7->Caption= "NOTA: Valores minimos de tiempo en segundos \npara los
siguientes voltajes" ;
Form1->Label6->Caption= "Volts     Seg\n  1      3\n  2       6\n  3       9" ;
Form1->Label8->Caption= "Volts     Seg\n  4      12\n  5       15\n  6       18" ;
Form1->Label9->Caption= "Volts  Seg\n 7      21\n   8    24\n   9      27\n 10      30" ;
Form1->Edit2->Visible= false;
Form1->Button3->Visible= false;
Form1->Button4->Visible= false;
Form1->Label4->Visible= false;
Form1->Label3->Visible= false;
Form1->Label5->Visible= false;

Form1->Label2->Caption="VOLATAJE MAXIMO \n DE 0 10 VOLTS";
error1= DAQmxCreateTask("",&MiTaskHandle);
error2= DAQmxCreateAOVoltageChan(MiTaskHandle,"Dev1/ao1","",
                        0.0,10.0, DAQmx_Val_Volts,"");
error3=DAQmxStartTask(MiTaskHandle);
if(error1!=0 ||error2!=0||error3!=0)
```

```
{
Application->MessageBoxA("ERROR","FALLO DE TARJETA",0 );
exit(1);
}
else
Application->MessageBoxA("EXITO","COMUNICACION EXITOSA",0 );

//Form1->Button3->Visible=true;
Form1->Button2->Visible=true;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit1Change(TObject *Sender)
{
if ( Form1->Edit1->Text != "" )
    voltaje=StrToInt(Form1->Edit1->Text);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit2Change(TObject *Sender)
{
if ( Form1->Edit2->Text != ""  )
    T=StrToInt(Form1->Edit2->Text);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit1KeyPress(TObject *Sender, char &Key)
{
if ((Key < '0' || Key > '9') && Key !=8 && Key !=46)
  {
  Key=0;
  }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit2KeyPress(TObject *Sender, char &Key)
{
if ((Key < '0' || Key > '9') && Key != 8 && Key !=46)
  {
  Key=0;
  }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button4Click(TObject *Sender)
{
tiempo = T*1000/N;
Form1->Label5->Caption= tiempo;
Form1->Button3->Visible= true;
}
```

//---------------------------------------------------------------------------

### 3.3.- Pasos para la instalación de la tarjeta GPIB

1. Se requiere instalar el DRIVER de la tarjeta. Para esto, instalar el disco NI488.2 para Windows de la NATIONAL.

2. Una vez instalado el disco, aparecerá una carpeta: c:\Archivos de Programa\NationalInstruments\NI.4882\Languages\BorlandC

3. En la carpeta anterior aparecerán tres archivos: borlandC_gpib-32.obj, decl-32.h y ni488.h

4. Para programar la tarjeta GPIB, primero debe indicársele al instrumento qué número de GPIB le corresponderá. En el programa en c se puede declarar una variable “int MiInstrumento = #”.

5. Antes de iniciar la programación, copiar en la carpeta de trabajo los archivos: borlandC_gpib_32.obj y ni488.h. En seguida, renombrar el ni488.h a decl-32.h

6. Ahora el programa debe iniciar con:

```
#include "decl-32.h"
#pragma link "borlandc_gpib-32"
```

7. Para iniciar la comunicación usar:

```
     static int Dev=ibdev(0,1,0,T10s,1,0);
       if (ibsta &ERR)
{
      Application->MessageBoxA("Error","",MB_OK);
        exit(1);
 }
```

Si no hubo error, procedemos con el programa, como se indicará a continuación.
Para terminar de usar la tarjeta, es NECESARIO liberarla, y se hace de la siguiente manera:

ibclr(Dev) que debe ser la última instrucción.

8. Para comunicarse con un instrumento hay que conocer los comandos de dicho instrumento. Supongamos que a un osciloscopio se le pide su identificación así: “ID?”

Entonces se requieren dos pasos: a) solicitar, b) tomar dato

y se hace de la siguiente manera:

```
AnsiString s;
 int n;
char arreglo[1000];
s="ID?";
 n=s.Length();
ibwrt(Dev,&s[1],(long) n);
 ibrd(Dev,arreglo,50);
 arreglo[ibcnt]='\0'; ó = NULL; Aquí terminamos el arreglo con final nulo.
Edit1->Text=arreglo; Aquí desplegamos en algún lugar el resultado.
```

**3.4.- Programa para adquirir datos del osciloscopio marca tektronix mediante la tarjeta GPIB.**

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include <stdio.h>
#include "Unit1.h"
#include "decl-32.h"
#pragma link "borlandc_gpib-32"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
int Dev;
char res[25000];
double T1;
double V1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
AnsiString s;
int n;
Dev= ibdev(0,1,0,T10s,1,0);
if(ibsta & ERR)
  {
  Application->MessageBoxA("ERROR","",MB_OK);
```

```
  exit(1);
  }
  Application->MessageBoxA("EXITO","",MB_OK);


s="CH1:SCA?";         // con esta instrucción le pedimos
                        al osciloscopio la escala de voltaje en el canal 1
n=s.Length();
ibwrt(Dev,&s[1],(long)n);
ibrd (Dev,res,50);
res[ibcnt]=NULL;
  //s.sprintf(res);
Edit2->Text=res;
n=Edit2->Text.Length();
Edit2->Text=Edit2->Text.SubString(11,n-11);
//V1=StrToFloat(Edit2->Text);
Form1->Button3->Visible= true;
Form1->Edit3->Visible= true;
ibclr(Dev);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
AnsiString s;
int n;
s="HOR:SCA?"; // con esta instrucción  le pedimos al osciloscopio la escala en tiempo
n=s.Length();
ibwrt(Dev,&s[1],(long)n);
ibrd (Dev,res,50);
res[ibcnt]=NULL;
  //s.sprintf(res);
Edit3->Text=res;
n=Edit3->Text.Length();
Edit3->Text=Edit3->Text.SubString(19,n-19);
//T1= StrToFloat(Edit3->Text);
Form1->Button1->Visible= true;
ibclr(Dev);
}
//---------------------------------------------------------------------------
```

**3.5.- Programa para adquirir datos del lock-in mediante la tarjeta GPIB.**

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include <stdio.h>
#include "Unit1.h"
```

```
#include "decl-32.h"
#pragma link "borlandc_gpib-32"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
int Dev;
char res[25000];
double T1;
double V1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
AnsiString s;
int n;
Dev= ibdev(0,1,0,T10s,1,0);
if(ibsta & ERR)
  {
  Application->MessageBoxA("ERROR","",MB_OK);
  exit(1);
  }
  Application->MessageBoxA("EXITO","",MB_OK);

s="*IDN?";         // con esta instrucción le pedimos
                      al lock-in su identificación (marca ,version, etc..)
n=s.Length();
ibwrt(Dev,&s[1],(long)n);
ibrd (Dev,res,50);
res[ibcnt]=NULL;
  //s.sprintf(res);
Edit2->Text=res;
n=Edit2->Text.Length();
Edit2->Text=Edit2->Text.SubString(11,n-11);
//V1=StrToFloat(Edit2->Text);
Form1->Button3->Visible= true;
Form1->Edit3->Visible= true;
ibclr(Dev);
}
//---------------------------------------------------------------------------
```

### 3.6.- Programa final para controlar la tarjeta DAQ y tomar datos del lock-in al mismo tiempo.

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include <stdio.h>
#include "Unit1.h"
#include "NIDAQmx.h"
#include "decl-32.h"
#pragma link "borlandc_gpib-32"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
// VARIABLES PARA LA TARJETA GPIB
int Dev;
char res[25000];
// VARIABLES PARA LA TARJETA DAQ
TaskHandle MiTaskHandle=0;
float64 dato;
TForm1 *Form1;
int N;
float voltaje;
int tiempo,T;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  if(voltaje>0&&voltaje<=10)
  {
  N= voltaje*205;
  Form1->Label1->Caption=N;
  }
Form1->Edit2->Visible= true;
Form1->Button4->Visible= true;
Form1->Label3->Visible= true;
Form1->Label5->Visible= true;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
dato=0;
DAQmxWriteAnalogF64(MiTaskHandle,1,1,10.0,
                    DAQmx_Val_GroupByChannel,&dato,NULL,NULL);
```

```
exit(0);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{
int n;
double x;
AnsiString s;
int a;
double Lockin;

Form1->Edit2->Visible= false;
Form1->Button4->Visible= false;
Form1->Button3->Visible= false;
Form1->Edit1->Visible= false;
Form1->Button1->Visible= false;
Form1->Label1->Visible= false;
Form1->Label2->Visible= false;
Form1->Label3->Visible= false;
Form1->Label5->Visible= false;
Form1->Label4->Visible= true;

  for (n=0;n<N;n++)
   {
    x=(double)n/N*voltaje;
    dato=x;
    DAQmxWriteAnalogF64(MiTaskHandle,1,1,10.0,
                DAQmx_Val_GroupByChannel,&dato,NULL,NULL);
s="OUTP? 3";
a=s.Length();
ibwrt(Dev,&s[1],(long)a);
ibrd (Dev,res,50);
res[ibcnt-1]=NULL;
Lockin=StrToFloat(res);
s.sprintf("%d          %f",n,Lockin);
Form1->RichEdit1->Lines->Add(s);
//ibclr(Dev);
Application->ProcessMessages();
Sleep(tiempo/4);
   }
   Sleep(1000);

  for (n=N-1;n>0;n--)
   {
    x=(double)n/N*voltaje;
    dato=x;
    DAQmxWriteAnalogF64(MiTaskHandle,1,1,10.0,
                DAQmx_Val_GroupByChannel,&dato,NULL,NULL);
s="OUTP? 3";
a=s.Length();
```

```
ibwrt(Dev,&s[1],(long)a);
ibrd (Dev,res,50);
res[ibcnt-1]=NULL;
Lockin=StrToFloat(res);
s.sprintf("%d          %f",n,Lockin);
Form1->RichEdit1->Lines->Add(s);
Application->ProcessMessages();
Sleep(tiempo/4);
   }
Form1->Edit1->Visible= true;
Form1->Button1->Visible= true;
Form1->Button3->Visible= true;
Form1->Label4->Visible= false;
Form1->Label1->Visible= true;
Form1->Label2->Visible= true;
 }
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
int error1, error2, error3;
DecimalSeparator='.';
Form1->RichEdit1->Text=" ";

Form1->Label7->Caption= "NOTA: Valores minimos de tiempo en segundos \npara los
siguientes voltajes" ;
Form1->Label6->Caption= "Volts     Seg\n  1       12\n  2       24\n  3       36" ;
Form1->Label8->Caption= "Volts     Seg\n  4       48\n  5       60\n  6       72" ;
Form1->Label9->Caption= "Volts     Seg\n   7       84\n   8       96\n   9       108\n 10       120" ;
Form1->Edit2->Visible= false;
Form1->Button3->Visible= false;
Form1->Button4->Visible= false;
Form1->Label4->Visible= false;
Form1->Label3->Visible= false;
Form1->Label5->Visible= false;

Dev= ibdev(0,1,0,T10s,1,0);
if(ibsta & ERR)
  {
  Application->MessageBoxA("ERROR","",MB_OK);
  exit(1);
  }

Form1->Label2->Caption="VOLATAJE MAXIMO \n DE 0 10 VOLTS";
error1= DAQmxCreateTask("",&MiTaskHandle);
error2= DAQmxCreateAOVoltageChan(MiTaskHandle,"Dev1/ao1","",
                          0.0,10.0, DAQmx_Val_Volts,"");
error3=DAQmxStartTask(MiTaskHandle);
if(error1!=0 ||error2!=0||error3!=0)
{
Application->MessageBoxA("ERROR","FALLO DE TARJETA",0 );
```

```
exit(1);
}
Form1->Button2->Visible=true;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit1Change(TObject *Sender)
{
if ( Form1->Edit1->Text != "" )
    voltaje=StrToInt(Form1->Edit1->Text);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit2Change(TObject *Sender)
{
if ( Form1->Edit2->Text != ""  )
    T=StrToInt(Form1->Edit2->Text);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit1KeyPress(TObject *Sender, char &Key)
{
if ((Key < '0' || Key > '9') && Key !=8 && Key !=46)
  {
  Key=0;
  }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit2KeyPress(TObject *Sender, char &Key)
{
if ((Key < '0' || Key > '9') && Key != 8 && Key !=46)
  {
  Key=0;
  }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button4Click(TObject *Sender)
{
tiempo = T*1000/N;
Form1->Label5->Caption= tiempo/4;
Form1->Button3->Visible= true;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button5Click(TObject *Sender)
{
int Respuesta;
 Respuesta=Application->MessageBoxA("¿Salvar Datos?","Salvando",MB_OKCANCEL);
 if (Respuesta==ID_OK){
  if( Form1->SaveDialog1->Execute() )
      Form1->RichEdit1->Lines->SaveToFile(Form1->SaveDialog1->FileName);
}
}
//---------------------------------------------------------------------------
```

## BIBLIOGRAFIA

http://es.wikipedia.org/wiki/Adquisici%C3%B3n_de_datos

http://es.wikipedia.org/wiki/IEEE_488

http://en.wikipedia.org/wiki/Data_acquisition

http://www2.uca.es/grup-invest/instrument_electro/ppjjgdr/Electronics_Instrum/Electronics_Instrum_Files/temas/T7_Instrum_Prog.PDF