UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Diseño de un módulo electrónico e interfaz para el control de un posicionador 3D

TESIS

Que para obtener el título de

Ingeniero Mecatrónico

P R E S E N T A

Ramses Castro Cabrera

DIRECTOR(A) DE TESIS

Dr. Pedro Jesús Acevedo Contla

Ciudad Universitaria, Cd. Mx., 2017

Agradecimientos

Agradezco a la **Universidad Nacional Autónoma de México**.

A la **Facultad de Ingeniería**.

Al **Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas (IIMAS)** y al **Departamento de Ingeniería en Sistemas Computacionales y Automatización (DISCA).**

A la DGAPA proyecto PAPIIT-IN106016, por el apoyo otorgado para la realización de esta tesis.

Al **Ing. Adalberto Joel Durán Ortega** por su extensa asesoría y constante apoyo durante cada etapa de la realización de esta tesis, sin su asesoría este proyecto no habría podido realizarse.

Y especialmente a mi asesor de tesis, **Dr. Pedro Acevedo Contla** por brindarme las facilidades, oportunidad, asesoría y apoyo para poder realizar esta tesis.

Dedicatoria

Con mucho cariño agradezco y dedico esta tesis a mi madre Martha Rebeca Cabrera Soria por su apoyo en todas y cada una de las etapas de mi vida, como persona y como estudiante, y que gracias a ella soy todo lo que soy ahora. Por brindarme el amor y el apoyo de una madre maravillosa, valiente, fuerte e inteligente, por enseñarme las bondades de la vida, el trabajo duro y la familia. Y por mostrarme que todo es posible mientras se quiera y se trabaje por ello, a nunca rendirme, quererme a mí mismo y a la gente que me rodea. Por siempre procurar mi bienestar y felicidad, sacrificándose por mí y mi hermana asegurándose que nunca nos falte nada para nuestro futuro y nuestra felicidad. Te amo mamá.

A mi hermana Isis Castro Cabrera que siempre ha sido mi motivo de orgullo e inspiración y que ha estado a mi lado en todos los momentos alegres y tristes de mi vida. Que comparte conmigo mis sueños y mis ambiciones, que me ha enseñado el valor del esfuerzo y ha sido mi compañera en incontables aventuras. Por siempre defenderme y cuidarme, incluso de mí mismo. Y aunque tal vez nuestras vidas nos dirijan a esquinas diferentes del mundo, siempre seremos hermanos. Siempre contarás conmigo San, sin importar lo que pase ni dónde te encuentres.

A mi abuela María de los Ángeles Soria Contreras que siempre ha estado a mi lado con su amor y ha sido siempre un motivo de alegría en mi vida. Que me ha apoyado siempre en mi formación y me ha brindado su cariño. Por preocuparse siempre por mi bienestar, felicidad y cuidarme desde mi infancia. Y a pesar de que los años pasen, me sigue amando y viendo como al niño que mecía en sus piernas. Vita, te adoro con todo mi corazón.

A mi Golda, Karen Gabriela Sánchez Escamilla con quien he compartido risas y lágrimas, que siempre ha estado ahí para escucharme, entenderme y darme siempre un consejo. Que me ha brindado su confianza y cariño. Y que ha sido mi cómplice e inquisidora a través de todos estos años.

A mis estimados Bajos Pineda, José Manuel, Mauricio y Paulina, y también a Wicho López Hernández, mis amigos de Veracruz que me han brindado su confianza y una amistad impresionante y me han acompañado en grandes aventuras en etapas buenas y malas, que han sabido apoyarme, me han dado su confianza y me han hecho ver diferente la vida. Han sido una fuente de alegría para mí y sé, siempre podré contar con ellos.

A mi querido Julio C. Chávez Argüelles con quien he encontrado siempre una palabra de apoyo y una opinión sincera. Que con sus bromas y comentarios me ha alegrado la vida y me ha demostrado el valor de una amistad sincera. Y que sé, puedo confiar en él en todo momento de mi vida.

A Chio Estrada Castro que con su alegría y apertura me ha hecho ver el mundo de una manera distinta, por darme su confianza, apoyarme y escucharme. Que me ha hecho reír, divertir y querer aprender más de mí mismo y de los demás.

A mi buen José Miguel Ortiz Sánchez, Mike, con quien he pasado tantos momentos felices. Que a través de su buen humor y buena voluntad me ha hecho ser mejor persona, por escucharme, entenderme y respetarme aún si las situaciones se tornaron difíciles. Por las horas de conversación. A él y a su familia, por depositar su confianza en mí y recibirme siempre con calidez y alegría.

A los hermanos Rivera Mota, Israel y Melissa, que me han dado alegría, apoyo y confianza con su particular humor. Que me han brindado su confianza, cariño y una sincera y desinteresada amistad. Definitivamente mi vida no sería igual sin ellos.

A mi Universidad Nacional Autónoma de México, la UNAM, definitivamente la mejor universidad de este país, que me ha dado tanto desde tiempos de preparatoria y me ha educado con excelentes valores y principios. Particularmente agradezco a mi Facultad, la honorable e imponente Facultad de Ingeniería que me ha preparado a través de sus profesores, aulas, laboratorios y talleres con la mejor educación profesional para un ingeniero y me ha brindado las herramientas necesarias para competir a nivel mundial con cualquier profesionista. Es un orgullo pertenecer a esta formidable Institución.

Y agradezco a todas aquellas personas que son o fueron parte de mi vida y que directa o indirectamente me han hecho elegir este camino. Amigos y familiares, muchas gracias a todos.

“Somos quienes somos por las personas y experiencias que tocan nuestra vida, las buenas y las malas. Eso define quiénes somos y a dónde iremos, eso define nuestro camino. Es necesario agradecerles pues no nos encontraríamos donde estamos sin ellas.”

Ramses Castro Cabrera

Introducción

Dentro de las diferentes líneas de investigación del Departamento de Ingeniería de Sistemas Computacionales y Automatización (DISCA) del Instituto de Investigaciones en Matemáticas y en Sistemas (IIMAS), se trabaja con tecnologías para la aplicación de imagenología ultrasónica. La utilización de manipuladores y posicionadores cartesianos en pruebas no destructivas de esta índole busca más cada día una correcta evaluación por medio de inspecciones ultrasónicas; no obstante la interpretación a menudo es altamente dependiente de factores subjetivos tales como la fatiga visual, el estado de ánimo y cansancio físico del evaluador, lo que repercute en la calidad de la inspección. Si se desea realizar pruebas con ultrasonido, se requiere de una extensa inspección de superficies y/o áreas de interés. Para este propósito se necesita de un conjunto de muestras ordenadas que deben adquirirse con la ayuda de un sistema de alta resolución que usualmente resulta muy costoso.

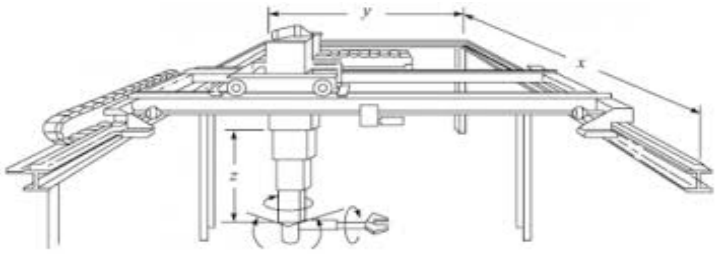


Figura I. *Sistema básico manipulador cartesiano [1].*

El DISCA contaba anteriormente con un posicionador XYZ de alta resolución controlado desde una computadora a través de una tarjeta de adquisición de datos de National Instruments. Su manipulación era hecha por medio de un sistema de lazo abierto, configurable desde una interfaz en LabVIEW. Dentro de los aspectos más relevantes de este posicionador se encontraban su resolución en unidades micrométricas y la versatilidad que implica al trabajar con diferentes tipos de pruebas.

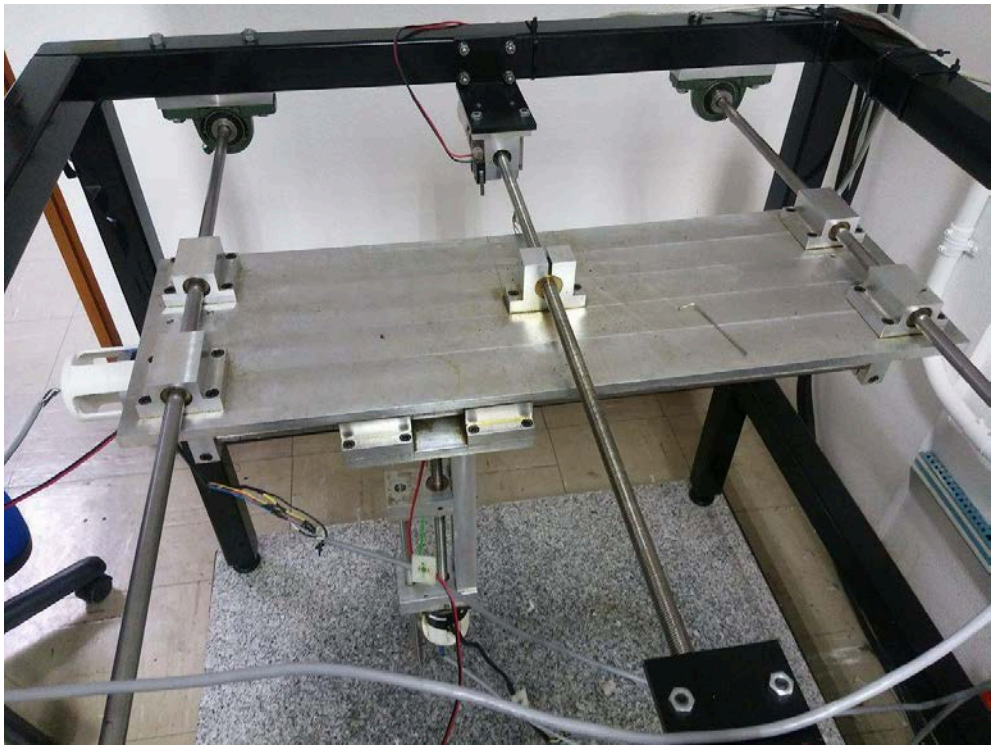


Figura II. *Posicionador XYZ del DISCA*

La visión a futuro con este manipulador es modificar su sistema para ser utilizado con otro tipo de tarjetas, eliminando la dependencia a las tarjetas de National Instruments, por lo que se requiere de la creación de una tarjeta que contenga el circuito eléctrico y su anexión a un controlador digital. Igualmente se pretende poder realizar un mapeo confiable a través de una retroalimentación que permita evitar errores propios de un lazo abierto. Se desea el algoritmo de elección de tipo de movimiento y distancia de desplazamiento considerando que el usuario pueda modificar estas variables, y se configure y observe todo en tiempo real a través de una interfaz generada en LabVIEW.

El proyecto realizado en esta tesis comprende la creación de una tarjeta de circuitos impresos que sea capaz de comunicar los tres motores del posicionador tomando en cuenta la alimentación para los motores y los sensores necesarios. La segunda parte comprende la lógica del posicionador, la cual incluye la interfaz hombre-máquina para manipular el posicionador, también se agrega la programación para hacer funcionar los motores, el algoritmo requerido para la realización de los distintos tipos de movimiento y finalmente la lectura de los canales de un osciloscopio así como su almacenamiento en una base de datos. Físicamente el posicionador deberá contar con un receptor de pulsos ultrasónicos, propiedad del DISCA, como punto terminal, el cual será desplazado en diferentes direcciones y distancias, según sea la elección del usuario, mientras se toman lecturas de este actuador por medio del osciloscopio y se almacenarán en un archivo de texto.

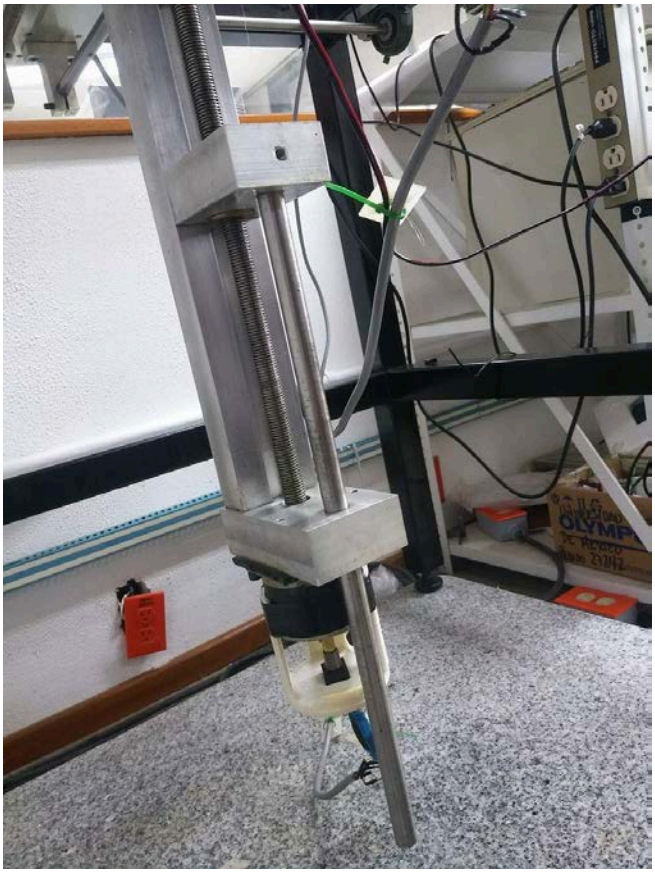


Figura III. *Barra terminal final del posicionador.*

La lógica seguida en las etapas de operación de este posicionador comprende primeramente ubicar al posicionador en un punto de arranque absoluto y re-ubicarlo en un segundo origen; una vez ahí comenzará el desplazamiento seleccionado mientras toma las lecturas del osciloscopio conectado y habrá de guardarlas automáticamente en un archivo. Una vez finalizado el proceso de lectura-desplazamiento el actuador debe volver a la posición de arranque inicial absoluta. Como resultado se busca obtener un archivo con las lecturas realizadas mostrando las coordenadas donde fueron tomadas y el posicionador de vuelta en su ubicación de origen.

Objetivos

- Diseñar y construir un sistema eléctrico para accionar los elementos que conforman el posicionador (motores y sensores), a partir de las exigencias de voltaje y corriente de cada uno de estos. Construir una tarjeta de circuitos impresos no mayor a 18 x 11 centímetros para colocarse dentro de una caja contenedora.

- Diseñar y construir una interfaz en LabVIEW para la comunicación hombre máquina de fácil comprensión y visualmente atractiva e interactiva.

- Crear un algoritmo de programación para la operación de los motores y sensores necesarios para el funcionamiento del posicionador a través del uso de LabVIEW y una tarjeta Arduino MEGA.

- Crear un algoritmo de programación para la selección de diferentes distancias y tipo de movimiento: lineal, planar o volumétrico. Además su vinculación con la interfaz en LabVIEW y un osciloscopio Tektronix DPO 3014.

- Facilitar el almacenamiento de los datos de las señales provenientes del osciloscopio en distintas coordenadas del desplazamiento dentro de un archivo de texto.

El presente trabajo está dividido en 6 capítulos, cuyo contenido es el siguiente:

Capítulo 1. Se da una breve explicación sobre lo que son los sistemas de instrumentación y control, así mismo se comentan las ventajas de utilizar elementos virtuales para la instrumentación y automatización.

Capítulo 2. Se comentan los conceptos básicos de los elementos físicos, utilizados para el desarrollo de este proyecto y por qué fueron seleccionados contra otros en el mercado.

Capítulo 3. Se explican de forma breve los sistemas de adquisición de datos, también se informa brevemente el funcionamiento e importancia del microcontrolador a utilizar.

Capítulo 4. Este capítulo se explican los pasos a seguir en el sistema electrónico, su selección de dispositivos y diseño, posteriormente se da la explicación de la implementación y funcionamiento de la tarjeta de circuitos impresos.

Capítulo 5. Se da una explicación del proceso de programación empleado en el presente trabajo, la labor de lectura de señales analógicas y digitales, procesamiento de datos, programación de algoritmos de manipulación del posicionador y despliegue de información.

Capítulo 6. Se describen las pruebas hechas y se discuten los resultados así como su relación con los objetivos planteados.

Anexos. Se muestra el código fuente del proyecto, diagramas electrónicos y guía de usuario.

CAPÍTULO I
Sistemas de Instrumentación

1.1 Introducción

A lo largo de la historia el ser humano ha deseado poder medir variables físicas para múltiples propósitos, la medición ha surgido a partir de la necesidad de querer saber el tamaño de un objeto, su peso, su temperatura, el tiempo de una acción, entre otras medidas físicas. Era de esperarse que dicha variable debiera ser comparada con algún elemento físico y estandarizado y no sólo por la percepción personal utilizando las facultades sensoriales humanas. A partir de este punto surgieron los instrumentos, dispositivos o formas de medir una variable, de manera estandarizada logrando que quien utilizara un instrumento pudiera obtener el mismo resultado repetidas veces.

Con el paso de los años y el desarrollo de la tecnología han sido desarrollados instrumentos electrónicos donde a partir de su diseño y composición, junto con la ayuda de sensores o incluso computadoras nos entregan resultados más precisos, con una menor posibilidad de error y con la facultad de no sólo medir, sino procesar los datos de nuestras medidas. De aquí surgen los sistemas de instrumentación, estructuras complejas que agrupan un conjunto de instrumentos, programas que se encargan de automatizar el proceso, las conexiones entre estos elementos que garantizan la repetitividad de las medidas. El objetivo básico de un sistema de instrumentación es la adquisición de información del mundo físico a la máxima velocidad posible, con la mayor exactitud que se pueda obtener.

1.2 Instrumentación

La instrumentación trata de las técnicas, los recursos, y métodos relacionados con la concepción de dispositivos para mejorar o aumentar la eficacia de los mecanismos de percepción y comunicación del hombre [5]. La instrumentación comprende dos campos principales: instrumentación de medida e instrumentación de control. En general, en el diseño de los sistemas de instrumentación de medida la atención se centra en el tratamiento de las señales o magnitudes de entrada, mientras que en los sistemas de control se da especial importancia al tratamiento de las señales de salida. En el primer caso son de interés los captadores o sensores y los transductores, mientras que en el segundo los dispositivos más relevantes son los accionadores o actuadores.

Existen instrumentos de diversos tipos según sea la aplicación que se desee dar. Bajo esta idea podemos encontrar dentro de los sistemas de instrumentación de medida a distintos dispositivos cuya función sea la de determinar la presencia o ausencia de una variable física (como los sensores de humo), otros que servirán para indicar visualmente

una variable (indicadores con aguja variable) o sencillamente que sirvan para medir una cierta variable en un determinado tiempo y hacer un registro de ella (sismógrafo por ejemplo). Dentro de los instrumentos de control se pueden encontrar aquellos que transmiten señales a fin de mantener una variable en una condición deseada; en este caso es común encontrarse con transmisores, transductores y actuadores finales como motores o pistones.

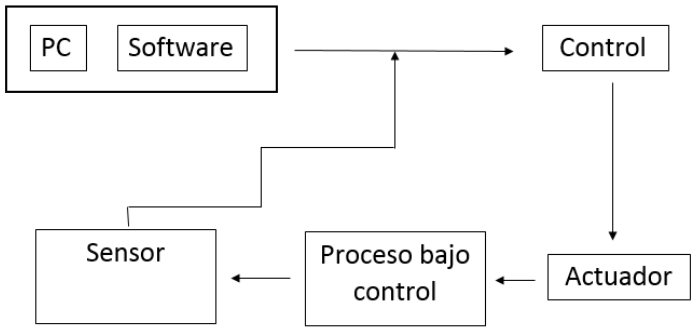


Figura 1.1 *Diagrama general de un sistema de instrumentación y control*

Es importante aclarar que las señales de entrada recibida en sistemas de instrumentación siempre serán analógicas o digitales. Estas señales se pueden manipular para que su procesamiento se realice de una manera más adecuada. En muchas ocasiones al medir una variable, esta puede ser muy débil para el captador por tanto se hace uso de diversos filtros y amplificadores de señal. También se da el caso de querer convertirse una señal proveniente del medio ambiente en una señal capaz de ser procesada por una computadora, para estos casos son ocupados los sistemas de conversión analógico-digital.

1.2.1 Datos analógicos

Un dato analógico es una onda producida por la oscilación o la aceleración de una carga eléctrica y que conforme transcurre el tiempo tiene un valor. Se refiere a la transmisión electrónica que se consigue añadiendo señales de frecuencia o amplitud variables a ondas transportadoras de corriente electromagnética alterna con una frecuencia dada. Las ondas electromagnéticas tienen componentes eléctricas y magnéticas [6].

Habitualmente las ondas que rigen nuestro planeta se presentan todas de forma ondulatoria y cada una tiene sus únicas propiedades, no obstante también entran en conflicto con otras ondas y objetos físicos presentando los efectos de difracción e interferencia haciendo difícil su clasificación y aislamiento. Los elementos más conocidos de una onda son la amplitud, el periodo y la frecuencia.

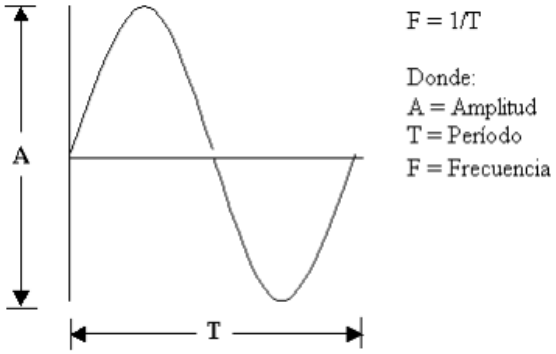


Figura 1.2 *Señal analógica [2]*

Una ventaja de las señales analógicas es que presentan datos de manera continua y pueden ser propagadas por diferentes medios, incluso en el vacío si se trata de ondas electromagnéticas. Igualmente al presentar características únicas es mucho más sencilla su diferenciación de otras ondas, su clasificación y el filtrado.

1.2.2 Datos digitales

Se dice que una señal es digital cuando sus magnitudes se representan mediante valores discretos en lugar de variables continuas como por ejemplo un interruptor de la luz que solo puede tomar valores o estados de abierto o cerrado [6]. Por abstracción dichos valores se sustituyen por ceros y unos, lo que facilita la aplicación de la lógica y la aritmética binaria. Si el nivel alto se representa por 1 y el bajo por 0 (Figura 1.3), se habla de lógica positiva y en caso contrario de lógica negativa.

Una señal digital puede ser obtenida de una señal analógica mediante un proceso de conversión analógico digital (DAC) donde se realiza un muestreo de la onda en sus diferentes valores de tensión o voltaje, luego los valores continuos de la onda sinusoidal se convierten en series de valores numéricos decimales discretos correspondientes a los diferentes niveles o variaciones de voltajes, la cuantificación de estos valores tomados en diferentes puntos de la onda sinusoidal permite medirlos y asignarles sus correspondientes valores en el sistema numérico decimal.

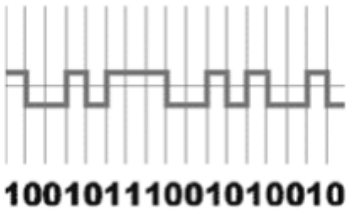


Figura 1.3 *Señal digital [2]*

Un dato digital se puede distinguir por ciertas características que continuación se enlistan:

- Los datos digitales toman valores discretos.

- Se suelen representar por una serie de pulsos de tensión que representan los valores binarios de la señal.
- La transmisión digital tiene el problema de que la señal se atenúa y distorsiona con la distancia, por lo que a cada cierta distancia hay que introducir repetidores de señal.

1.2.3 Principios básicos de instrumentación

Los instrumentos se utilizan para monitorear y controlar variables de procesos. Dependiendo del tipo de procesos, se seleccionan los componentes del mismo. Dentro de la instrumentación toman importancia los diferentes elementos de cómputo junto con transductores o sensores. Así también dependiendo de las condiciones de trabajo, velocidad de operación deseada, aspectos de seguridad y precisión habrán de ser elegidos los captadores y actuadores necesarios.

Igualmente es importante evaluar la compatibilidad con elementos de cómputo cuando se trata de sistemas de instrumentación con cierto nivel de automatización. Finalmente un aspecto importante para elegir un instrumento es la velocidad de acción o de lectura; se debe considerar la capacidad del instrumento para hacer lecturas, la velocidad a la que la computadora procesa esas lecturas y los datos son procesados, evaluados y se toma una resolución para tomar una decisión de acción, se espera un tiempo mínimo entre la lectura y la respuesta, la velocidad de ambos aspectos depende muchas veces de las limitaciones físicas en los dispositivos tanto de lectura (sensores) como de acción (actuadores).

Independiente de la aplicación, todo instrumento de medición cuenta con una serie de términos que se enuncian a continuación:

- *Exactitud*: Aproximación con la cual la lectura de un instrumento se acerca al valor real de la variable medida.
- *Precisión*: Medida de la reproductibilidad de las mediciones; esto es, dado el valor fijo de una variable, la precisión de una medida del grado con la cual las mediciones sucesivas difieren una de otra.
- *Sensibilidad*: Relación de la señal de salida o respuesta del instrumento respecto al cambio de la entrada o variable medida.
- *Resolución*: Cambio más pequeño en el valor medido al cual responde el instrumento.
- *Error*: Desviación a partir del valor real de la variable medida.

Dentro del aspecto de los errores, estos pueden surgir por tres categorías diferentes: Errores gruesos, sistemáticos y aleatorios. Los primeros se limitan a aquellos de origen humano como errores en la lectura, ajuste incorrecto así como equivocaciones de cálculos [7]. Los segundos, errores sistemáticos, se deben a fallas de los instrumentos como unidades defectuosas o desgastadas así como efectos ambientales que modifican la lectura. Finalmente los errores aleatorios ocurren por causas que no se pueden

establecer directamente debido a variaciones en los parámetros en los sistemas de medición.

Una parte importante de los sistemas de instrumentación es el análisis estadístico y operaciones matemáticas para determinar un resultado. El análisis estadístico de datos es una práctica cuyo objetivo es determinar analíticamente la incertidumbre del resultado final. El resultado de un método de medición se puede predecir con base en el muestreo de datos sin tener información detallada de todos los factores de perturbación; para realizar métodos estadísticos e interpretaciones generalmente se necesita un gran número de mediciones y la aplicación de modelos matemáticos como por ejemplo, regresión lineal. Por el lado de las operaciones matemáticas se tienen el conjunto de modelos matemáticos y algoritmos a nivel de programación que permiten a partir de un dato de entrada o una serie de datos, la conversión o manipulación para obtener como resultado final una variable de salida que dependa del valor de la variable de entrada, de un proceso matemático y de los valores fijos propuestos por el programador.

1.2.4 Instrumentación virtual y real

Actualmente la instrumentación con dispositivos electrónicos es fuertemente utilizada en distintos ámbitos, desde mediciones con elementos como sensores o transductores hasta complejos sistemas de instrumentación en aplicaciones de automatización industrial. Los diferentes tipos de instrumentos permiten a través de su diseño, funcionamiento y aplicación la lectura de una variable con el fin de ser utilizada dentro de un sistema computarizado o autónomo. No obstante la necesidad de disminuir la cantidad de aparatos de medición, la facilidad de transporte y el creciente mejoramiento de sofisticados softwares de medición hacen más común el uso de herramientas virtuales como aparatos de medición.

Desde hace ya algunos años se ha visto cómo en softwares de audio y video, así como programas como Matlab, Wolfram Mathematica, Proteus (Figura 1.4) por mencionar algunos, se ha impulsado el uso de interfaces que simulan la apariencia y funcionamiento de un dispositivo físico como puede ser un osciloscopio, voltímetro, cronómetros, termómetros, etc. Sin embargo no basta con tener la apariencia de un instrumento de medición, también se debe comportar como uno, de este concepto surgen los instrumentos virtuales.

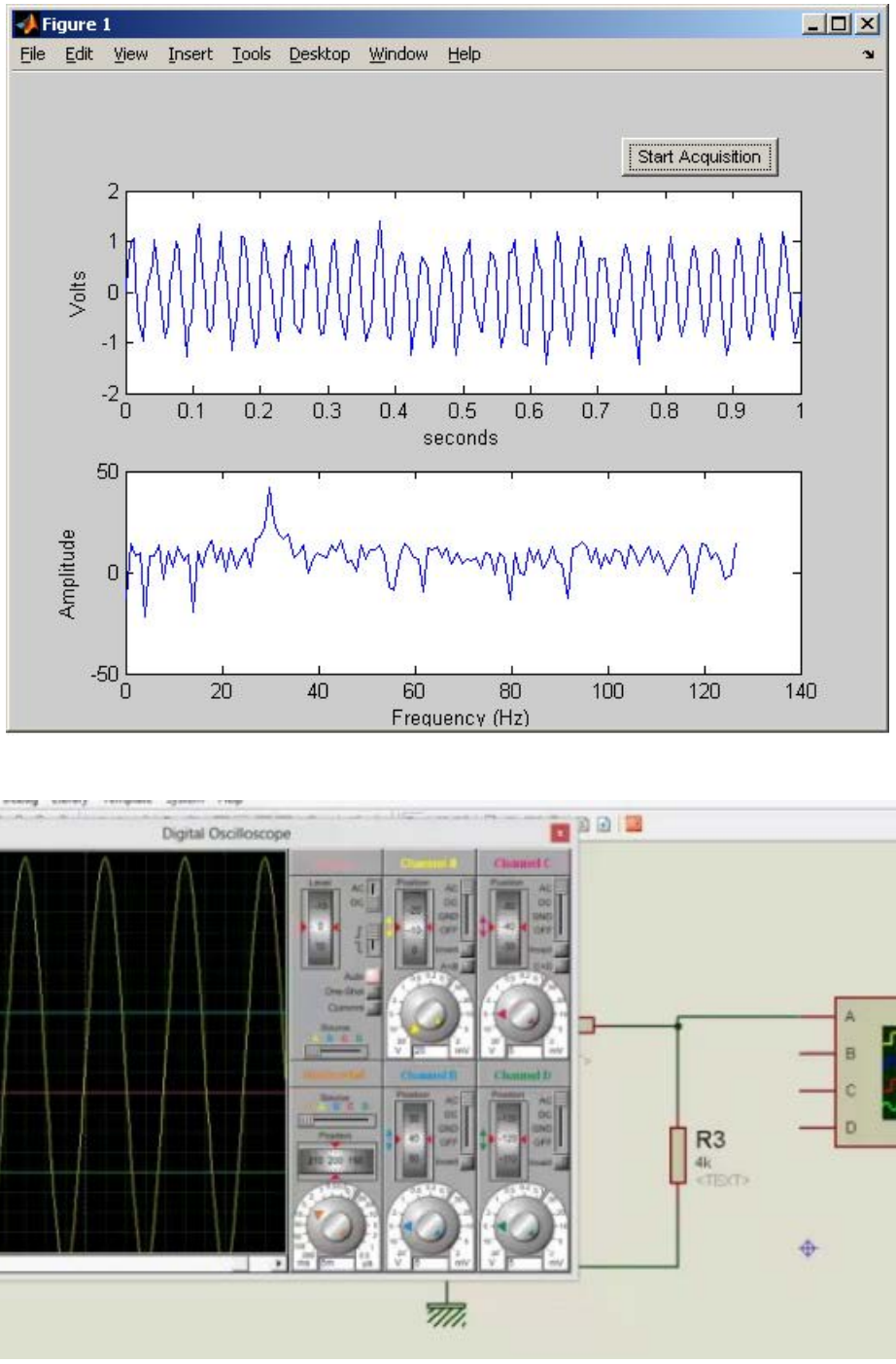


Figura 1.4 *Osciloscopios virtuales de Matlab (arriba) y Proteus (abajo).*

A diferencia de los dispositivos físicos, los dispositivos virtuales son todos aquellos que viven dentro del entorno de una computadora, una interfaz gráfica o bien cualquier otro elemento que exista gracias a una implementación gráfica. Se entiende como virtual a todo aquello con lo que nosotros como humanos podamos interactuar fuera del plano físico en el que vivimos y que exista gracias a distintas tecnologías gráficas y de información sin la necesidad de la existencia física del mismo. La manera en cómo nos

comunicamos de una manera cómoda, directa, versátil y entendible con este tipo de sistemas recibe el nombre de interfaz. Mediante la interfaz se conectan ambos mundos a través de su propio lenguaje gracias al uso de diversos dispositivos electrónicos así como importantes elementos de programación y electrónica de manera que una maquina o sistema funcione con sólo un toque de la computadora.

1.3 Principios básicos de control automático

Primeramente se debe entender que el control para sistemas en ingeniería es un tema muy extenso por lo que sólo se tocarán los conceptos básicos en este subtema. Entendido esto se puede empezar diciendo que hoy en día múltiples máquinas con cierto nivel de sofisticación y de autonomía también poseen en cierta medida un nivel de control o una programación de control.

El control en sí como elemento dentro de la ingeniería es una operación o serie de operaciones que se realizan a partir de un dato o conjunto de datos de entrada, de manera que se obtenga como resultado una o varias operaciones [8]. Visto de un modo más simplista, a partir de una información dada como entrada, empleando distintas metodologías, se espera un resultado de salida. Dentro de los sistemas de control podemos encontrar dos pilares fundamentales: El control de lazo abierto (Figura 1.5) y control de lazo cerrado (Figura 1.6). El primero se refiere a sistemas en los cuales la salida no tiene efecto sobre la acción de control, dicho de otra manera no existe una forma de comparar la variable de salida con la variable de entrada. En el caso de control de lazo cerrado se mantiene una relación determinada entre la salida y la entrada de referencia, comparándolas y usando la diferencia como medio para modificar el valor de la salida, volviendo al sistema insensible a las perturbaciones externas y a las variaciones internas en los parámetros del sistema [8].

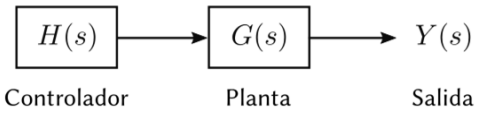


Figura 1.5 *Sistema de control Lazo Abierto [8]*

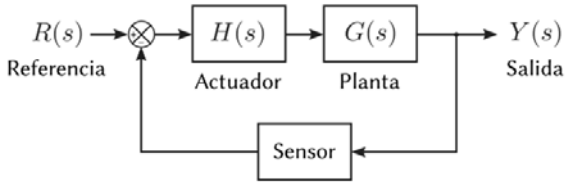


Figura 1.6 *Sistema de control Lazo Cerrado [8]*

Ya entendida la diferencia entre ambos tipos de control existen ciertos conceptos dentro de la teoría de control que vale la pena puntualizar:

Variable controlada: La variable controlada es la cantidad o condición que se mide y controla. Normalmente es la variable de entrada al sistema.

Variable de control: La señal de control o variable manipulada es la cantidad o condición que el controlador modifica para afectar el valor de la variable controlada. Normalmente, la variable controlada es la salida del sistema.

Planta: Una planta puede ser una parte de un equipo, un conjunto de los elementos de una máquina que funcionan juntos, y cuyo objetivo es efectuar una operación particular.

Procesos: Se define como proceso a una operación o un desarrollo natural progresivamente continuo, marcado por una serie de cambios graduales que se suceden unos a otros de una manera determinada y que conducen a un resultado o propósito determinados.

Sistema: Un sistema es una combinación de componentes que actúan juntos y realizan un objetivo determinado. Un sistema no está necesariamente limitado a los sistemas físicos. El concepto de sistema se puede aplicar a fenómenos abstractos y dinámicos, como los que se encuentran en la economía.

Perturbaciones: Una perturbación es una señal que tiende a afectar negativamente el valor de la salida de un sistema. Si la perturbación se genera dentro del sistema se denomina interna, mientras que una perturbación externa se genera fuera del sistema y se considera como parte de la entrada.

Control realimentado: El control realimentado se refiere a una operación que, en presencia de perturbaciones, se ponen en relación la salida y la entrada al sistema, de esta manera el resultado de las operaciones dentro del sistema generan una salida la cual es tomada como una referencia nuevamente para su re-operación.

1.4 LabVIEW

LabVIEW Laboratory Virtual Instrument Engineering Workbench, es una herramienta gráfica para pruebas, control y diseño mediante la programación. El lenguaje que usa se llama lenguaje G. Es una plataforma capaz de crear entornos para la comunicación e interconexión de diversos sistemas, dispositivos, tarjetas controladoras y subrutinas lógicas para el desarrollo de una aplicación [9].

La idea de National Instruments al crear LabVIEW fue la concepción de una plataforma con una sintaxis de programación gráfica que facilita visualizar, crear y codificar sistemas de ingeniería, LabVIEW ofrece reducir tiempos de pruebas, análisis de negocio basado en datos recolectados y facilitar cualquier trabajo de ingeniería. Está diseñado para incorporarse con otro software, ya sea métodos alternativos de desarrollo o plataformas de fuente abierta, para dar al usuario el mayor número de herramientas para la implementación de un proyecto.

En los últimos años National Instruments ha mostrado una alta capacidad para desarrollar tecnología tanto para la lectura de señales como para su procesamiento y manipulación. Actualmente el software LabVIEW puede mostrarse como elemento de vanguardia en el campo de instrumentación virtual siendo capaz de analizar señales provenientes de diversos instrumentos, comunicarse con tarjetas y microcontroladores avanzados, simular la apariencia y uso de instrumentos así como hacer uso de diversas herramientas de control para múltiples actuadores y señales (Figura 1.7).

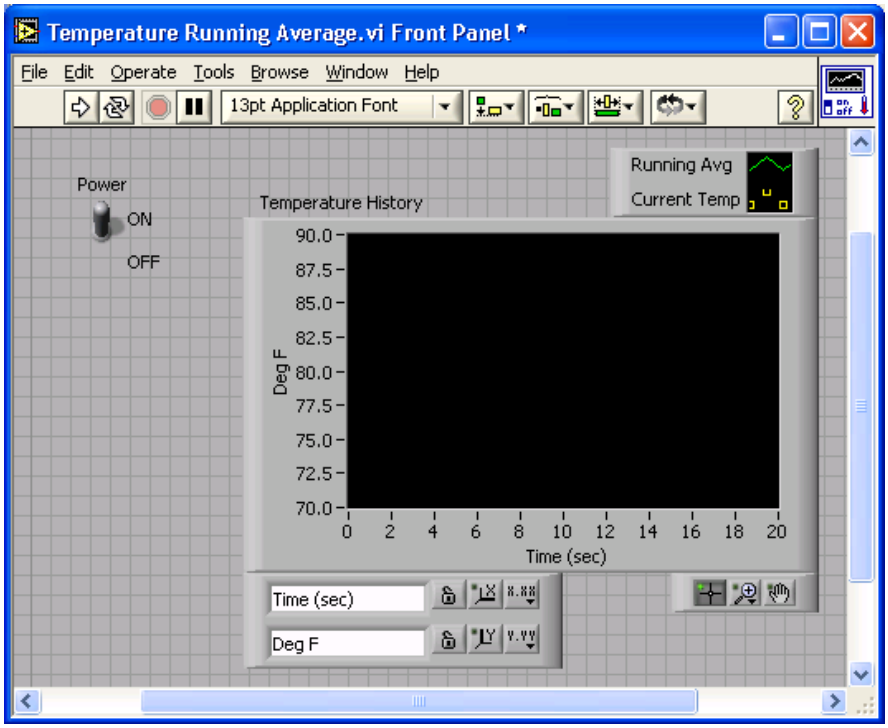


Figura 1.7 *Panel frontal de LabVIEW con controles e indicadores básicos (ejemplo) [5].*

Por el lado de desarrollo de interfaces, LabVIEW cuenta con varios elementos de control y muestreo así como de comunicación con otros sistemas e instrumentos periféricos, de esta manera es relativamente práctico diseñar, programar, implementar y utilizar un sistema complejo de ingeniería en esta plataforma. Los elementos que proporciona LabVIEW para el diseño de interfaces suelen ser variados y dinámicos permitiendo la modificación de cada elemento para mejor utilización del programador. Para el proyecto de tesis presente será este el software utilizado para la permisión de interacción hombre-máquina para el control del posicionador de tres dimensiones.

CAPÍTULO II
Actuadores y sensores

2.1 Introducción

Dentro de un sistema mecatrónico es común el uso de dispositivos que permitan la lectura de variables tales como desplazamiento, sentido, velocidad, peso, temperatura, etc. Para este fin son utilizadas diversas herramientas como sensores y transductores para determinar las condiciones del sistema. Además de estos últimos también es muchas veces necesaria la implementación de dispositivos que puedan modificar estas estas condiciones, para ello se hace uso de actuadores, dispositivos tales como motores, pistones, válvulas, bombas, resistencias eléctricas, etc. En conjunto, sensores y actuadores, permiten al sistema en cuestión realizar una serie de tareas modificando las condiciones iniciales.

Dentro del presente proyecto fueron utilizados motores para mover el posicionador y a su vez se hizo uso de sensores que han servido para determinar el sentido de giro de estos motores y el desplazamiento a través de una programación conjunta.

2.2 Sensores

Un sensor es un dispositivo capaz de detectar diferentes tipos de materiales, elementos o magnitudes físicas o químicas. Es un dispositivo que a partir de las condiciones del medio, proporciona una señal de salida en relación a lo que se desee medir. [10]

Los sensores son actualmente implementados en la mayoría de los dispositivos electrónicos; su uso depende de la aplicación que se desee y va desde formar parte de la instrumentación de un equipo, monitoreo, investigación e incluso a forma parte de juguetes. Hoy en día dentro de la selección de un sensor, se deben considerar diferentes factores, tales como: la forma de la carcasa, distancia operativa, datos eléctricos y conexiones.

2.3 Transductores

Un transductor es el dispositivo que transforma una magnitud física (mecánica, térmica, magnética, eléctrica, óptica, etc.) en otra magnitud, normalmente eléctrica. Lo que se busca con un transductor es convertir una lectura física obtenida en otro tipo de dato para su operación y procesamiento. [10]

Algunos de los sensores y transductores utilizados con más frecuencia son los calibradores de tensión (utilizados para medir la fuerza y la presión), los termopares (temperaturas), los velocímetros (velocidad). Cualquier sensor o transductor necesita estar calibrado para ser útil como dispositivos de medida.

La calibración es el procedimiento mediante el cual se establece la relación entre la variable medida y la señal de salida convertida. Los transductores, al igual que los

sensores, pueden clasificarse en dos tipos básicos, dependiendo de la forma de la señal convertida. Los dos tipos son:

- Transductores analógicos.
- Transductores digitales

Los transductores analógicos proporcionan una señal analógica continua como voltaje, corriente eléctrica o sonido. Esta señal puede ser tomada como el valor de la variable física que se mide. Los transductores digitales producen una señal de salida digital, en la forma de un conjunto de bits de estado en paralelo o formando una serie de pulsaciones que pueden ser contadas. En una u otra forma, las señales digitales representan el valor de la variable medida. Los transductores digitales suelen ofrecer la ventaja de ser más compatibles con las computadoras digitales que los analógicos en la automatización y en el control de procesos, no obstante la información de una onda sinusoidal se ve reducida considerablemente.

2.4 Codificadores (encoder)

Un codificador es un dispositivo electromecánico que puede medir el movimiento o la posición. La mayoría de los codificadores utilizan sensores ópticos para proporcionar señales eléctricas en forma de trenes de impulsos, los cuales pueden a su vez, traducirse en movimiento, dirección o posición. [11]

El principio de funcionamiento es muy sencillo: debido a que el codificador presenta ranuras, por medio de diferentes estrategias, se deja pasar una señal de un lado a otro del codificador con excepción en los espacios oscuros o sólidos que no permiten que la luz atraviese, o bien se utiliza un sensor opto reflectivo de manera que la luz “rebote” en los espacios donde existe material. De esta manera a medida que el codificador avanza o gira el cambio de pulsos de señal permite saber si el elemento en cuestión se mueve, qué tanto se mueve, qué tan rápido y en qué sentido por medio de la ayuda de una computadora o un contador digital.

2.4.1 Codificador rotativo

Los codificadores rotativos se utilizan para medir el movimiento de rotación de un eje. La figura 2.5 muestra los componentes fundamentales de un codificador giratorio, que consiste en un diodo emisor de luz (LED), un disco y un detector de luz en el lado opuesto del disco. El disco, que está montado en el eje de la rotación, tiene patrones de sectores opacos y transparentes codificados en él. Al girar el disco, los segmentos opacos bloquean a la luz y los transparentes la dejan pasar. Esto genera los pulsos de una forma de onda cuadrada, la cual puede luego ser interpretada como posición o movimiento. [11]

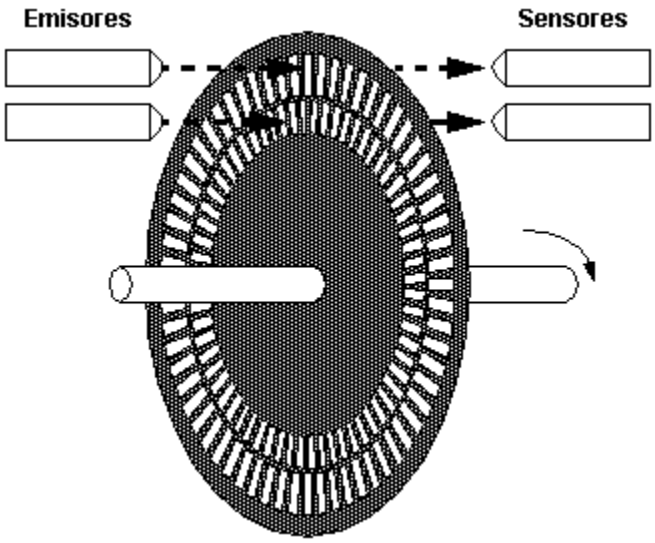


Figura 2.1 *Componentes de un codificador giratorio [11].*

Los codificadores suelen tener 100 segmentos por revolución. Esto significa que estos codificadores pueden proporcionar 3.6 grados de resolución.

2.4.2 Codificador de cuadratura

Los codificadores de cuadratura poseen dos sensores que proveen de dos canales, A y B, desfasadas 90º uno del otro (Figura 2.2). Cuando el codificador gira hacia una dirección el canal A adelanta al B, y cuando gira en dirección contraria el canal B adelanta al A. El principio de funcionamiento de estos sensores consiste en dejar pasar una señal infrarroja por medio de pequeños espacios en la periferia de un disco; para el caso particular de este encoder se ocupan dos sensores infrarrojos con un cierto desfase entre uno y otro (canales A y B). Estos contactos ópticos se van activando y desactivando en una secuencia que nos permite saber la dirección y el número de desplazamientos que han ocurrido en el encoder.

Figura 2.2 *Configuración de un codificador de cuadratura [11].*

Cuando se dispone de una tarjeta DAQ con dos contadores es sencillo medir velocidad y dirección, y así determinar posición. Sin embargo a menudo no se dispone de tarjetas o dispositivos con dos contadores, o que permiten ingresar directamente la señal del codificador. La solución es convertir la señal de cuadratura de 90º en dos señales: una de pulsos para determinar velocidad y cantidad de movimiento, y otra tipo digital On/Off para determinar la dirección. La desventaja de esta conversión es que en la transición pueden perderse pulsos y así contar erróneamente la cantidad de movimiento que ha habido en una dirección determinada.

	A	B
1	0	0
2	0	1
3	1	1
4	1	0

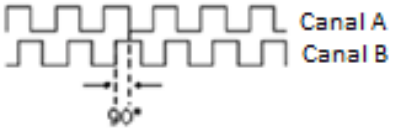


Figura 2.3 *Estados de un encoder de cuadratura.*

Para este proyecto ha sido utilizado el codificador de cuadratura BOURNS EM14 que posee un total de 256 pulsos por revolución. De esta manera se cuenta con una gran resolución pudiendo dividir una revolución en 256 segmentos (1.4º por cambio).

2.5 Motores a pasos

Un motor a pasos, o motor de velocidad gradual, es un dispositivo electromecánico que convierte pulsos eléctricos en movimientos mecánicos. Dependiendo de su diseño, un motor a pasos puede avanzar 90º, 45º, 18º o incluso una fracción de grado por pulso. Éstos se producen cuando se genera una inversión de corriente por los devanados, que a su vez es controlada por unos interruptores de estado sólido de potencia [12]. Los motores a pasos son utilizados mayormente en aplicaciones donde se requiera una buena precisión de velocidad o de posición.

La principal ventaja de los motores a pasos es su capacidad de convertir una excitación en una salida mecánica digital, o bien un incremento angular en el rotor denominado comúnmente “paso”. Los motores a pasos utilizan comúnmente corriente directa para funcionar por lo que son inherentemente digitales [14]. Poseen ciertas ventajas sobre los servomotores como la alta resolución, el torque (par mecánico) estable a bajas velocidades y una reducción de resonancia por la velocidad careciendo de manera obligada de un control de lazo cerrado y de un conjunto de dispositivos analógicos y digitales como amplificadores, convertidores, generadores, etc.

Los motores a pasos como se ha mencionado, pueden utilizarse con un control de lazo abierto, resultando más útiles cuando se trabaja con sistemas de bajas aceleraciones y con cargas estáticas. En contraparte el lazo cerrado es indispensable cuando se trabaja con aceleraciones y particularmente cuando se ven involucradas cargas variables como en los sistemas transportadores de elementos. Es importante hacer notar que si se aplica más carga de la que es capaz de soportar el motor, este puede perder pasos y por ende la posición o ángulo del sistema en general.

2.5.1 Principio de funcionamiento

El principio básico de los motores a pasos es la alineación por flujo magnético, de tal forma que el motor a pasos posee tres o más devanados o bobinas que generan un flujo

magnético, el cual para poder cerrar un circuito de flujo, debe alinear primeramente los dientes del rotor para que así el flujo magnético pueda circular (Figura 2.3).

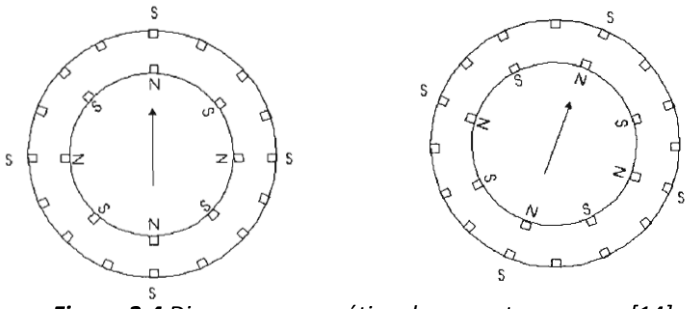


Figura 2.4 *Diagrama magnético de un motor a pasos [14].*

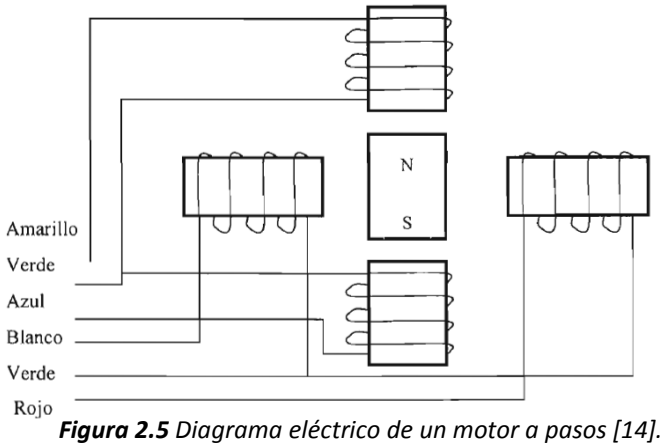


Figura 2.5 *Diagrama eléctrico de un motor a pasos [14].*

El motor a pasos, al excitar sus devanados, alinea los dientes del rotor con los dientes del devanado excitado de manera que al excitar los diferentes devanados en un orden específico resulta en un movimiento progresivo y uniforme [14]. A pesar de que los diferentes tipos de motores a pasos funcionan bajo un principio similar, existen diferentes tipos de motores dentro de los cuales se encuentran los motores magnetizados permanentemente, de reluctancia variable de unidades múltiples, de una sola unidad y los híbridos. Los motores a pasos más comunes son aquellos de reluctancia variable de varias unidades así como de una sola unidad.

2.5.2 Motor de reluctancia variable

En los motores de reluctancia variable su rotor es de hierro dulce laminado, con varios dientes en dirección radial, mientras que el estator tiene un número de polos diferente también en dirección radial (Figura 2.5). Si el número de fases del estator es q y su número de polos es Ns, entonces el número de dientes del rotor Nr, se suele elegir como Nr = Ns ± (Ns/q) .El número de dientes del rotor es menor que el número de dientes del estator, de modo que sólo un par de polos del estator y su correspondiente par de polos del rotor pueden estar alineados por fase. [12]

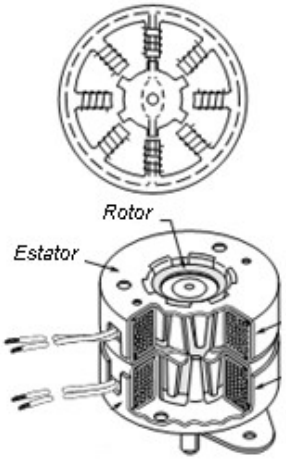
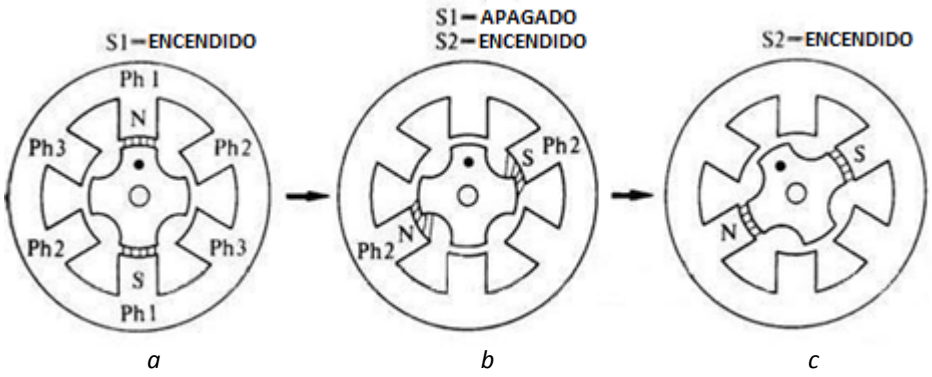


Figura 2.6 *Configuración de un motor de reluctancia variable [12].*

Las ranuras presentes en el rotor conllevan a una variación de la reluctancia en función de su posición angular. Al aplicar un impulso a la fase del estator, el rotor (hierro) se acerca al electroimán para disminuir el entrehierro, y con ello la reluctancia magnética en el circuito, y así facilitar el paso del flujo magnético. Partiendo de la posición de equilibrio con la fase 1 activada (Figura 2.6a), un paso se obtiene situando la fase 1 a estado OFF y la fase 2 a estado ON; en este instante, los polos del rotor más próximos a los polos de la fase dos del estator, son atraídos en el intento de circular las líneas de flujo magnético (Figura 2.6b), produciendo un movimiento de rotación entre ambos polos hasta quedar alineados (Figura 2.6c). Con esta operación obtenemos un paso del rotor, seguimos haciendo lo mismo sucesivamente para hacer el desplazamiento de más pasos [13]



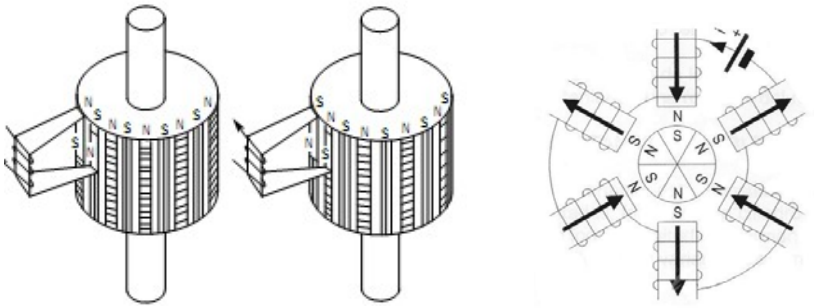
. ***Figura 2.7*** *Secuencia de excitación para producir un paso [13].*

El paso angular es:

$$\frac{360}{qN_r}$$

2.5.3 Motor de imán permanente

Estos motores tienen la característica de que utilizan un imán permanente cerámico cilíndrico en el rotor, es decir estos imanes están adheridos al rotor, los polos N y S son permanentes (Figura 2.7). El imán está magnetizado radialmente en una serie de polos. El estator está constituido por láminas de material ferro magnético, bobinado con el mismo número de polos que el rotor [13].



. ***Figura 2.8*** *Corte transversal de un motor a pasos de imán permanente [13].*

En la figura 2.8 se muestra el corte transversal de un motor a pasos de dos polos y dos fases (se llama así porque el estator tiene dos bobinas y el rotor dos polos magnéticos). Si se supone que el devanado *bs* se encuentra como un circuito abierto y que se aplica una corriente positiva constante por el devanado *as*. El resultado es que esta corriente establece un polo sur de estator, en el diente de estator donde está el devanado as1, y se establece el polo norte del estator en el diente en el que esta devanado as2. El rotor se colocará en *θrm = 0*. Ahora se desenergiza el devanado *as* al mismo tiempo que se energiza el devanado *bs*, con una corriente positiva. El rotor se mueve la longitud de un paso en dirección contraria a la de las manecillas del reloj. Para continuar los pasos en esa dirección, se desenergiza el devanado *bs* y se energiza el *as* con una corriente negativa. De esta manera es solamente necesario hacer una secuencia de paso de corriente para que el motor gire de manera constante, si se desea girar el motor en sentido contrario basta con invertir la secuencia de energización [12].

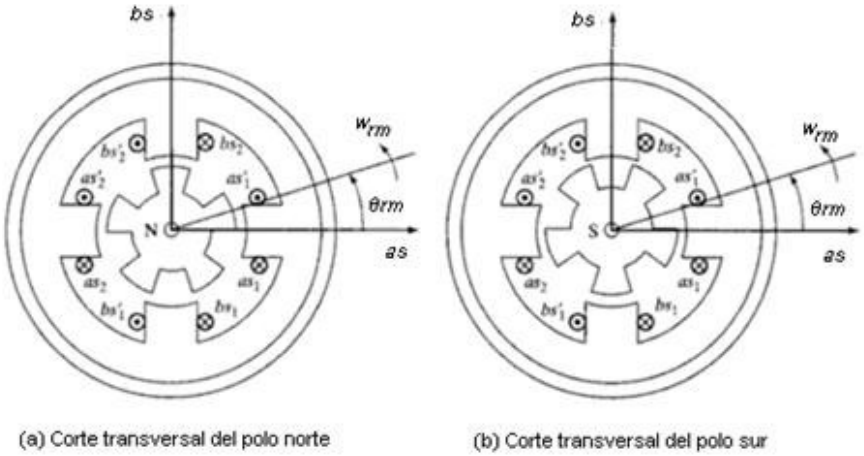


Figura 2.9 *Corte transversal de un motor a pasos [12]*

El ángulo de paso de este tipo de motores depende del número de polos del estator y el rotor. Debido a las características del material magnético utilizado en la construcción del rotor, el número de polos de éste es limitado, por lo que los ángulos que se consiguen con este tipo de motor son grandes. Por lo tanto el número de pasos viene dado por:

$$n = ne * np$$

Donde:

ne = número $de\ devanados\ (fases)$
np = número $de\ polos$

2.5.4 Motores Híbridos

Los motores híbridos son una combinación de los dos tipos mencionados anteriormente. Se denominan también motores de inductancia síncrona. Su estator consta de varias bobinas, mientras que su rotor consiste en un imán cilíndrico magnetizado axialmente (en la dirección del eje del motor), dispuesto entre dos piezas de hierro dulce laminado que tiene varios dientes, en número ligeramente distinto al de las bobinas del estator (figura 2.9). [12]

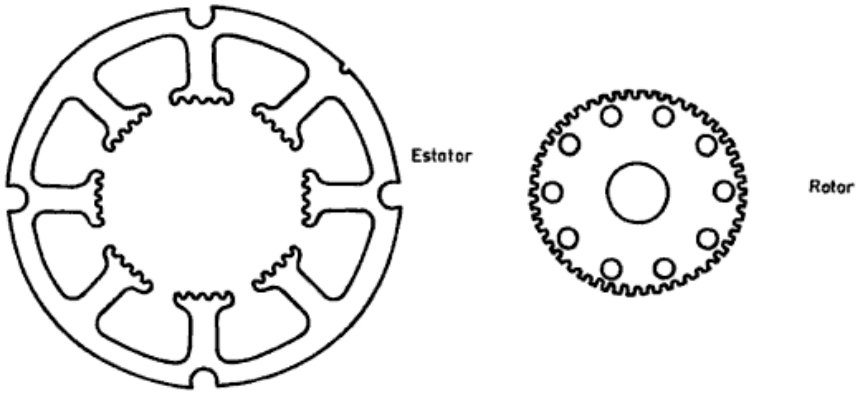
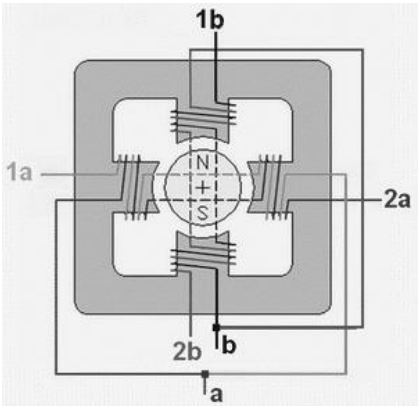


Figura 2.10 *Motor paso a paso hibrido [12]*

2.5.5 Motores unipolares y bipolares

Para los motores unipolares el bobinado por cada fase es doble (Figura 2.10), unido en el interior y puesto en serie nos entrega 6 hilos, agrupados de tres en tres para cada fase (uno de estos es el punto común). Existen tres tipos de secuencia para manejar estos tipos de motores (Figura 2.10), tipo wave drive, se activa una sola bobina a la vez, secuencia normal, se activan dos bobinas a la vez y secuencia de medio pasó [12].



	1	2	3	4
1	1	0	0	0
1	0	1	0	0
2	0	0	1	0
2	0	0	0	1

a

	1	2	3	4
1a	1	1	0	0
1b	0	1	1	0
2a	0	0	1	1
2b	1	0	0	1

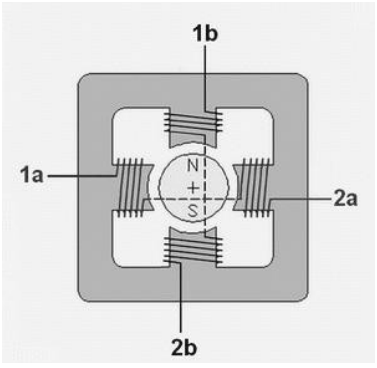
b

	1	2	3	4	5	6	7	8	9
1a	1	1	0	0	0	0	0	1	1
1b	0	1	1	1	0	0	0	0	0
2a	0	0	0	1	1	1	0	0	0
2b	0	0	0	0	0	1	1	1	0

c

Figura 2.11 *Secuencia de pulsos para un motor a pasos unipolar, a) Tipo wave drive, b) Secuencia normal. c) Secuencia de medio paso. [12]*

La configuración de los motores bipolares (Figura 2.11), requiere que las bobinas reciban corriente en uno y otro sentido, y no solamente un encendido-apagado como en los unipolares. Cada inversión de la polaridad provoca el movimiento del eje en un paso.



	1	2	3	4	5	6	7	8
1a	+	-			+	-		
1b	-	+			-	+		
2a			+	-			+	-
2b			-	+			+	-

Figura 2.12 *Secuencia de manejo para un motor a pasos bipolar [12]*

CAPITULO III
Adquisición de datos

3.1 Introducción

Cuando nos referimos a adquisición de datos nos referimos en esencia al proceso de recolectar la información dentro de una ventana de tiempo deseada. Esto es, tomar los datos procedentes de un sistema, ya sea analógico o digital, y registrarlos para su uso inmediato o bien dentro de una base de datos; todo este proceso habrá de realizarse dentro de un tiempo fijo con el fin de tomar un muestreo periódico y continuo.

Dentro de la naturaleza nos encontramos con múltiples fenómenos que pueden ser (y son comúnmente) registrados para su estudio. Entendemos que en estado natural estas señales se manejan de manera analógica: las ondas de luz que vemos en nuestro día a día son un ejemplo, las ondas de sonido, ondas de calor, la manera en cómo observamos los colores, etc. Vivimos sumergidos en un universo físico dominado por ondas que pueden ser cuantificadas y procesadas si tenemos el equipo adecuado.

Para recuperar los datos de un elemento del mundo físico es necesaria una fuerte comprensión de lo que ocurre con las ondas que lo rigen; todas las ondas son distintas una de la otra y tienen su propia longitud y frecuencia. Por ejemplo, en el proceso de distinción de colores, cada color tiene su propia frecuencia; el ojo humano nos permite diferenciar colores siempre y cuando se encuentren dentro del espectro de luz visible. Los diferentes fenómenos del mundo real son ondas analógicas que presentan una variación continua, esta variación puede ser muy rápida como en las odas de sonido o lenta como en las variaciones de temperatura.

La mejor manera de almacenar, manipular, recuperar, calcular e incluso procesar los datos de estas señales es mediante la tecnología analógica, apoyados en una computadora. Lamentablemente la computadora trabaja con señales digitales, volviendo incoherente la relación con las señales analógicas de nuestro mundo real. Para este proceso es necesaria la conversión de estos datos en señales analógicas a señales digitales vinculando ambos mundos mediante los llamados convertidores analógicos a digital (ADC: *Analog Digital Converter*) y el hardware necesario para obtener estos datos es llamado Sistema de adquisición de datos. Estas dos partes trabajan a la par comunicando ambos mundos para nuestro mayor aprovechamiento y estudio.

3.2 Controladores y microcontroladores

Recibe el nombre de controlador el dispositivo que se usa para dirigir uno o varios procesos. Por ejemplo, al medir la temperatura se hace uso de un sensor que obtiene información del sistema, cuando la temperatura registrada traspasa los límites prefijados

por el usuario, se generan las señales adecuadas que activan los actuadores que permiten modificar el valor de la temperatura dentro del rango estipulado.

Aunque el concepto de controlador ha permanecido invariable a través del tiempo, su implementación física ha variado frecuentemente. En la actualidad, todos los elementos del controlador se han incluido una placa, la cual recibe el nombre de microcontrolador. Realmente consiste en una sencilla computadora contenida en un circuito integrado.

Un microcontrolador habitualmente debe ser programado para poder ejecutar una operación en particular. Debido a que se asemeja a una computadora, un microcontrolador dispone normalmente de los siguientes componentes:

- Un procesador
- Memoria RAM.
- Memoria tipo ROM/PROM/EPROM.
- Terminales de E/S para comunicarse con el exterior.
- Módulos de conversión Analógico Digital o Digital Analógico
- Generador de pulsos de reloj.

Los productos que hacen uso de un microcontrolador a menudo presentan un mayor control en sus procesos, mayor fiabilidad, flexibilidad para modificar el proceso y la reducción de espacio en comparación con un circuito más extenso. El microcontrolador es en definitiva un circuito integrado que incluye todos los componentes de una computadora. Debido a su reducido tamaño es posible montar el controlador en el propio dispositivo al que gobierna.

3.3 Arduino

Arduino es una plataforma abierta para prototipos de fácil uso y acceso. Cuenta con una parte de software y una parte de hardware. En las placas de Arduino es muy sencilla la conexión con diversos dispositivos gracias a que cuenta con múltiples conectores que pueden funcionar como entradas o salidas, analógicas y digitales [18]. El entorno de arduino consta de un lenguaje de programación, basado primeramente en Wiring que a su vez se basa en lenguaje C. En cuanto a la IDE (*Integrated Development Environment), Arduino basa su funcionamiento en Processing.*

Actualmente Arduino es una herramienta muy útil para el desarrollo de varios proyectos. Originalmente creado para la difusión de conocimiento y aprendizaje de personas de todo tipo, relacionadas con el mundo de la electrónica o no, Arduino ha demostrado ser un fuerte utensilio en el desarrollo de proyectos a nivel profesional alcanzando la integración con distintos ambientes como la impresión 3D, vinculación con numerosos softwares de diseño, cálculo y procesamiento de imágenes, así como aplicaciones en espacios virtuales, automatización, seguridad, recreación, salud, entre otras.

La ventaja clara de esta plataforma ante competidores es la fácil manipulación de su software y hardware, los alcances que tiene debido a sus características físicas, la flexibilidad de trabajar con otros sistemas, la interconexión con distintos dispositivos y la

economía de sus placas. Una de las mayores ventajas para usar Arduino dentro de la realización de este proyecto es que su manipulación tanto de hardware como de software es muy sencilla. Se ha elegido el uso de una placa Arduino basándose en que posee una arquitectura abierta, es decir, en el uso de la IDE o bien de las tarjetas no se requieren permisos ni licencias, el software es libre para quien desee usarlo y por tanto existen muchos dispositivos compatibles con Arduino, hay una gran cantidad de información sobre las placas y programación disponible y su venta es sumamente popular en distintas partes del mundo.

Si se analizan los sistemas de Arduino como elementos de trabajo en proyectos, se encuentra como un entorno de fácil acceso y bajo precio. Las características propias de cada modelo de tarjeta generan un amplio rango de posibilidades de proyectos a realizar, según sea el propósito. La conexión de Arduino con una computadora es muy fácil y relativamente rápida (16 MHz) a través de un cable USB 2.0 de fácil adquisición. Además de estos detalles, el remplazo de los componentes de las placas o las placas mismas resulta barato y de fácil búsqueda a tal grado que, a partir de los elementos principales, se pueden diseñar nuevas tarjetas compatibles con la IDE de Arduino.

3.3.1 Arduino MEGA

El Arduino MEGA es una de las múltiples placas de Arduino, es un microcontrolador basado en ATmega 2560. Cuenta con 54 puertos o terminales de conexión "*pin" (* de su nombre en inglés) que pueden ser codificadas como entradas o salidas. Es una de las placas más conocidas y usadas de Arduino y tiene una ventaja ante sus hermanas debido a la cantidad de elementos que pueden manipularse así como la velocidad de procesamiento (16 Mhz), sus puertos de conexión, convertidor analógico digital, regulador de voltaje DC, un mayor número de terminales de puerto serial y la facilidad de conexión con otros dispositivos periféricos [18].

Figura 3.2 *Arduino MEGA [18].*

Una ventaja que presenta el Arduino MEGA comparado con sus homólogos (UNO, Demilanuove, micro, nano, Lillypad, etc.) es que posee una mayor cantidad de memoria flash así como un total de tres pares de terminales de comunicación serial. Además de un evidente gran número de pines disponibles, es capaz de manipular más cantidad de

corriente (40 mA) resultando en un mayor número de elementos de entrada y/o salida manipular, con mayor control y sin afectar el rendimiento.

Dentro de las características principales del Arduino MEGA se mencionan los siguientes datos técnicos:

Microcontrolador	ATmega 2560
Voltaje de operación	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (límites)	6-20V
Pines digitales I/O	54 (15 con salida PWM)
Pines digitales	16
Corriente DC por pin I/O	40 mA
Corriente DC para el pin 3.3V	50 mA
Memoria Flash	128 KB de los cuales 4 KB usados para bootloader
SRAM	8 KB
EEPROM	4 KB
Velocidad del reloj	16 MHz

Debido a que el proyecto de tesis presente requiere de un total de 16 pines para manipular adecuadamente el posicionador XYZ se ha optado por el uso de esta tarjeta como el elemento lógico para comunicar la computadora con el posicionador. Considerando que tarjetas más pequeñas como Arduino UNO o Leonardo tienen un máximo de 13 terminales de conexión digitales es imposible el uso de estas tarjetas para implementarse en este posicionador. Igualmente se ha considerado a Arduino como un dispositivo que, por medio de software, es fácilmente vinculable con LabVIEW y su velocidad de respuesta con dicho software es adecuada.

3.4 Conectores

Un conector eléctrico es, como su nombre lo sugiere, un dispositivo capaz de unir temporalmente o permanentemente dos terminales. En electrónica se puede observar que estos conectores pueden ser encontrados en múltiples formas y tamaños dependiendo de la aplicación final así como el número de hilos que se manejen.

La ventaja que presenta un conector desde las grandes industrias o plantas eléctricas hasta los hogares y dispositivos electrónicos es la facilidad de prescindir de una conexión permanente como la soldadura o trenzado; igualmente proporcionan una significativa sencillez en el intercambio de terminales, seguridad ante daños físicos así como facilidad de manipulación.

3.4.1 Cable RJ45

El cable tipo RJ45 es una interfaz ocupada popularmente en sistemas de telefonía e internet [13]. Por su tamaño y simplicidad de diseño en cuestiones de espacio y conexión es uno de los cables favoritos para aplicaciones de red Ethernet.

Figura 3.3 *Cable RJ45 [13].*

Las ventajas que ofrece ante otros cables es su reducido espacio de operación pues contiene ocho alambres de conexión en un espacio menor a dos centímetros cuadrados así como su facilidad de ensamble entre el cable de ocho hilos y el conector de la terminal. Es una ventaja para uso de aplicaciones donde se requieran cables de bajo calibre y una paso de corriente bajo.

Dentro de este proyecto de tesis se han usado tres conectores RJ45 donde cada uno conecta 4 cables necesarios para los sensores de los encoders. Se han utilizado conectores macho para los cables de los sensores y hembra para las terminales dirigidas al Arduino MEGA. De esta manera se asegura por medio de los conectores RJ45 una buena conexión con un mínimo movimiento entre las terminales. Se eligió el uso de este conector debido a su número de alambres internos así como por la facilidad que este presenta para conectarse.

3.4.2 Cable USB 2.0

El cable USB recibe su nombre por el protocolo que lleva su nombre (Universal Serial Bus) en donde las terminales del cable además de alimentar al dispositivo con que se conecte también tienen la facultad de transferir información a una velocidad relativamente rápida.

Actualmente estos cables han sido reemplazados por otros de mejores cualidades, diseño y velocidad de transferencia [16]. No obstante siguen siendo uno de los cables más utilizados en la mayoría de dispositivos electrónicos periféricos en computadoras como impresoras y cámaras web. En el caso del presente proyecto el cable USB 2.0 se encarga de la comunicación del Arduino MEGA con la computadora.

Figura 3.4 *Cable USB 2.0 [16].*

El cable USB 2.0 se usa principalmente para la conexión del Arduino con la computadora. A pesar de que ciertas tarjetas de pequeños tamaños ocupan otro tipo de cables, para el uso de placas grandes es bastante popular el uso del cable USB 2.0. Al ser un cable comercialmente conocido en dispositivos electrónicos como impresoras y cámaras, es muy fácil su adquisición o reposición.

CAPÍTULO IV
Diseño e implementación del sistema eléctrico

4.1 Introducción

Para el desarrollo del proyecto presente se ha buscado crear una tarjeta que sea capaz de controlar tres motores a pasos de manera independiente a partir del microcontrolador Arduino tomando en consideración las demandas de energía que cada elemento del circuito exige. Anteriormente este posicionador de tres dimensiones contaba con el apoyo de una tarjeta de National Instruments, la cual por sus características, permitía a una computadora con LabVIEW ser compatible con la tarjeta y facilitar los procesos de comunicación. Esta tarjeta funcionaba con puertos de entrada y salida permitiendo recibir información de un lado y comunicando en su salida a los motores del posicionador.

A pesar de que el posicionador funcionaba ya de una manera adecuada, el movimiento se hacía con un sistema de control de lazo abierto. La posición se determinaba considerando que cada paso del motor recorría una distancia micrométrica de tal modo que un elevado número de pasos resultaba en avance de milímetros y centímetros.

La tarjeta diseñada contiene un circuito para alimentar los motores del posicionador al igual que a los sensores de los encoders de cuadratura y switch bumper, los cuales permiten crear un sistema de lazo cerrado en el posicionador. Además se ha propuesto el uso de una placa Arduino como medio para comunicar la información de la computadora a los motores dejando de lado la tarjeta controladora de National Instruments.

4.2 Objetivos principales de la placa

Se requería una placa capaz de comunicarse con una placa Arduino MEGA, que reciba información de los pines de salida digital del Arduino y los dirija a los pines de señal de tres puentes H y por medio de ellos controlar el sentido de giro de tres motores a pasos. Se pedía de igual manera la alimentación de seis sensores cuya función es, de tres de

ellos supervisar el movimiento de los motores durante su giro y de los tres sobrantes definir las coordenadas físicas del punto de origen.

	REQUERIMENTOS	ESPECIFICACIONES
Tamaño	Necesario introducirse en una caja contenedora.	Medidas de 18X20X5 cm
Alimentación	Capaz de ser alimentada con un toma corriente. Debe alimentar tres motores a pasos y 6 motores.	-Alimentación de ±127V -Regulador de voltaje a 12V y 5V en CD.
Conexión	-Conectarse al tomacorriente con clavija. -Debe ser conectada a terminales de un Arduino MEGA.	-Uso de clavija convencional (Tipo A). -Uso de alambre de conexión/"jumpers"
Seguridad	-Se deben prever retornos de corriente que puedan dañar al Arduino y Sensores. -Las conexiones deben ser resistentes a movimiento.	-Uso de optoacopladores MOC3011 entre el Arduino y la tarjeta. -Uso de conectores de Ethernet RJ45.

4.3 Diseño del circuito electrónico

Como se ha mencionado, anteriormente se contaba con una Tarjeta de National Instruments encargada de la comunicación de datos y control de los motores. No obstante al decidirse el cambio hacia aquella tarjeta desarrollada en el IIMAS se ha solicitado que la tarjeta a diseñar fuera de características específicas en consideración del tamaño de la caja contenedora donde sería colocada para su protección.

Se realizó la tarjeta en una placa fenólica de cobre, las rutas para comunicar cada terminal entre los diversos dispositivos se han obtenido a partir del proceso de circuitos impresos por medio retiro de material con de óxido férrico. El proceso a seguir en la elaboración de la placa fue primeramente determinar los dispositivos necesarios para controlar la marcha y sentido de giro de un motor, la etapa de potencia y la alimentación del circuito, enseguida se realizó un circuito de prueba para los tres motores en una placa de pruebas o "protoboard" y comprobar su correcto funcionamiento con la ayuda de la placa Arduino, después generar el esquema de conexiones de cada elemento del circuito que irían montados en la placa PCB tomando en consideración los diagramas de conexión de cada elemento (datasheet) en el software Proteus 8 Professional, ISIS de Labcentere Electronics. Posteriormente se acudió al software ARES de Proteus para definir las rutas de conexión para la placa PCB.

4.3.1 Dispositivos utilizados

Como uno de los principales dispositivos del circuito se encuentra el motor a pasos 57BYG, un motor de tipo imán permanente, la ventaja al usar este motor es su par de (según el fabricante) 1.4 Nm a una corriente de 2.8A, lo que lo vuelve adecuado para sistemas de posicionamiento. Este motor es parte del diseño del posicionador previamente realizado en trabajos anteriores en el IIMAS. Otro elemento importante es el puente H L298 el cual se antepuso a su hermano L293 debido a su capacidad de manipulación de corriente, o a un arreglo de transistores tomando en cuenta que estos últimos necesitan de un mayor espacio y el uso de disipadores. También conocido como "puente completo", el puente H tiene su nombre debido a la forma que presenta dentro de un circuito esquemático simplificado. En la barra central se encuentra ubicado el motor y en cada "rama lateral" ascendente o descendente se ubican los conmutadores que, activados de manera apropiada, brindarán al sistema los movimientos necesarios para que el motor pueda girar en un sentido u otro. El uso de un puente H se considera adecuado para motores cuyo sentido de giro se desee controlar y en el caso de motores a pasos, las cuatro terminales del puente H son favorables para el control de los cuatro cables que controlan el motor.

Figura 4.1 *Puente H L298.*

Debido a que el puente H L298 no cuenta con diodos integrados, fue necesario diseñar el circuito considerando las conexiones necesarias para su adecuado funcionamiento. Además de los elementos extras al puente H (diodos, condensadores y resistores) se ha considerado la anexión de un transistor de regulación de voltaje 7805 para reducir el voltaje de 12V a 5V, necesarios para la habilitación del puente H (terminales ENABLE) así como para la alimentación de los sensores a cinco voltios, un exceso de voltaje en estos elementos provocaría que se quemasen. Ahora si bien es comprendido que se debe alimentar el circuito con un tomacorriente, ha sido necesario el uso de una fuente para regular el voltaje y obtener una salida de 12 voltios en corriente alterna; además se usó también un rectificador de voltaje (provisto como un elemento del proyecto de tesis anterior) que rectifica la corriente alterna y la convierte en una corriente directa de 12V necesaria para alimentar a los motores a pasos y al regulador de voltaje 7805.

Además de los elementos del circuito mencionados anteriormente, se ha adjuntado el uso de optoacopladores como medida de seguridad, también se ha pensado en las

terminales de entrada y salida de datos. Para las terminales de salida dirigidas a los cables de cada motor a pasos se han utilizado borneras tomando en cuenta el posible cambio de posición e incluso la probabilidad de accidentalmente jalar un cable del motor; el uso de borneras refuerza la conexión mejorando el agarre y evitando que al manipular el circuito estos cables sean desconectados. Por la parte de conexión con la placa Arduino se ha considerado utilizar *headers* o pines de conexión tanto macho como hembra; la consideración surge con base en la idea que la placa arduino no se encuentra montada en la placa PCB sino a un lado dentro de la caja que las contiene por lo que la manipulación de cables se ve reducida en comparación con la que se ve en las conexiones del motor. Además considerando la cercanía entre ambas placas es posible el uso de cables de conexión de corto alcance y de menor costo al mismo tiempo que el espacio en la superficie de la placa PCB es reducido y el uso de *headers* minimiza el espacio para las conexiones requeridas.

Finalmente la placa PCB se creó pensando en los diferentes elementos que necesitarían de una alimentación regulada a 5V de modo que se ha proviso de varios pines de conexión a lo largo de la placa PCB conectados a la terminal de voltaje de 5V y la tierra de la placa Arduino: Es importante resaltar que por motivos de seguridad la tierra de la placa Arduino está separada de la tierra que alimenta a los motores puesto que una falla en el circuito o una descarga eléctrica estropearía cada elemento del circuito, el Arduino MEGA y con una alta posibilidad al puerto USB de la computadora donde se encuentre conectado.

4.4 Placa PCB

Se conoce como placa PCB a la placa de circuito impreso, consiste en una placa de material aislante (comúnmente baquelita o fibra de vidrio) donde son “impresas” las pistas de un circuito electrónico (Figura 4.2). Estas placas pueden ser de distintos tamaños y pueden ser hechas en ambos lados; el objetivo principal de ellas es tener un lugar donde colocar los distintos circuitos integrados, resistores, condensadores u otros elementos de un circuito evitando el uso de alambres de conexión, tablas perforadas o matrices de puntos.

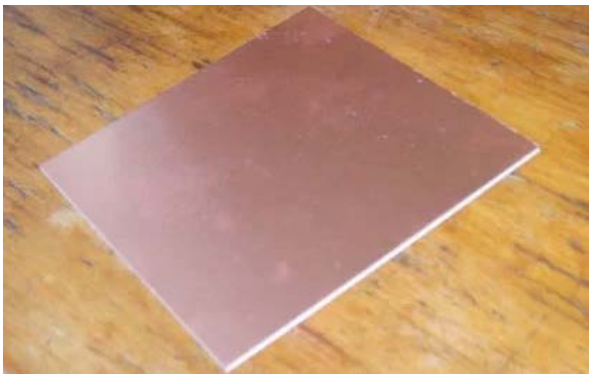


Figura 4.2 *Placa fenólica.*

Para hacer una placa PCB es necesario retirar el cobre de la placa por distintos métodos que van desde la corrosión por medio de químicos hasta el desbaste del metal hasta llegar al material aislante. Las ventajas que ofrece una de estas placas es su resistencia mecánica, su facilidad de transporte y conexión, no absorbe la humedad y es un método adecuado si se desea evitar problemas de conexión.

4.4.1 Distribución

La placa PCB debía ser colocada dentro de una caja de plástico que la contuviera, además se ha debido considerar la presencia del transformador de voltaje así como el circuito rectificador en la misma caja, por tal motivo la placa fenólica no podía ser de una medida mayor. La caja de plástico contenedora tiene una longitud de 20 cm, un ancho de 15 cm y una altura de 11 cm en su sección interna, las paredes tienen un grosor de 2.5 mm. Dadas estas medidas se consideró el uso de una placa fenólica de un largo de 15 cm y por 10 cm de ancho. De esta manera el espacio sobrante dentro de la caja fue suficiente para colocar el arduino Mega, el transformador de voltaje y el circuito rectificador.

Teniendo en cuenta las limitantes de espacio en la placa fenólica se tuvo que prever desde el software ARES la manera en que serían colocados los distintos dispositivos antes de crear la placa PCB. Uno de los principales puntos a considerar fueron las terminales de salida hacia los motores, se pensó colocar todas en un borde de la placa para tener una mejor organización y evitar el cruce de los cables del motor con otros propios del circuito. Igualmente se ha buscado que las tierras comunes se vean orientadas en un solo lado de la placa.

4.4.2 Rutas de la placa PCB

Se ha dicho que debido a la cantidad de conexiones necesarias la posibilidad de cruce entre las rutas se incrementa. Para este proyecto se ha necesitado conectar más de 120 pines de diferentes dispositivos ya sea entre ellos o a alguna terminal de voltaje o tierra. Por tal motivo el circuito presenta un total de conexiones externas o “puenteos” ajenas a las rutas obtenidas de la placa fenólica, un total de 16 terminales sin conectar han sido comunicadas con cables de conexión. Es importante aclarar que no es una situación grave utilizar puenteo, no obstante al crear una placa de circuitos impresos se busca que se requiera la menor cantidad de estos pues en efecto, el objetivo de una placa PCB es la eliminación (en este caso una reducción) del uso de alambres o cables.

Uno de los problemas comunes al crear una placa PCB es el ancho de la ruta, de ser muy grande resultará en un uso innecesario de espacio y en el caso de ser muy delgado puede causar que, al momento de retirar el material con óxido férrico, se elimine más cobre del deseado a causa de alguna imperfección y existan espacios sin

continuidad derivando en la necesidad de reparar dicha sección de la ruta añadiendo material extra como soldadura de estaño; para crear la placa PCB de este proyecto se usó un ancho de línea mínimo de 2mm. Igual que en el caso de las rutas delgadas, los puntos de conexión o “pads” donde son enterados los pines de un dispositivo, deben ser de un cierto grosor pues estos son perforados por un taladro de lado a lado de la placa y al momento de retirar el material es posible que se pierda el punto de conexión; se ha optado que todos los pads fueran de un diámetro igual a 4mm, de esta manera no importa si el pin es demasiado grueso, el punto de conexión no se perdería.

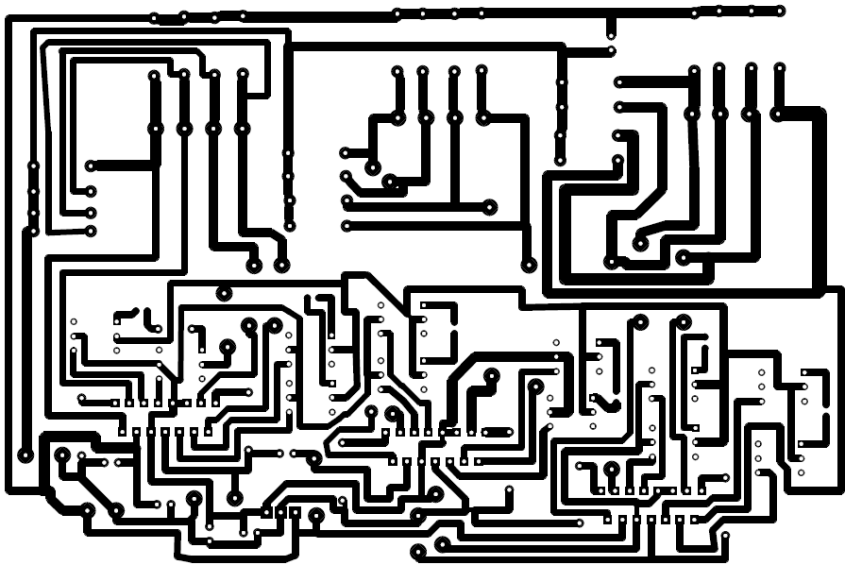


Figura 4.3 *Plano de rutas generado en Proteus ARES.*

Finalmente se contempló el uso de puentes de conexión entre las terminales que, por posibilidades físicas, no pudieron ser conectadas. Debido a que se genera el esquema base en computadora y posteriormente se imprime para entintar la placa fenólica, es recomendable tomar en cuenta la creación de ramificaciones en la ruta y/o crear puntos de conexión donde sean ingresados los extremos del alambre a conectar. De este modo si fueron 16 terminales sin conectar, fue necesario hacer ocho puentes que comunicaran dos terminales entre sí resultando en un circuito cerrado que permitiese el flujo de corriente y por ende el funcionamiento general de la placa de circuito impreso.

4.4.3 Modelo virtual

Una vez diseñado en Proteus ISIS el esquema electrónico, se genera automáticamente la lista de rutas (NETLIST). Una red o ruta es un grupo de pines interconectados entre sí y la lista de redes es una lista con todas las redes que forman el diseño. ARES es capaz de recibir esta lista de redes para diseñar, a partir de ella, nuestra placa de circuito impreso.

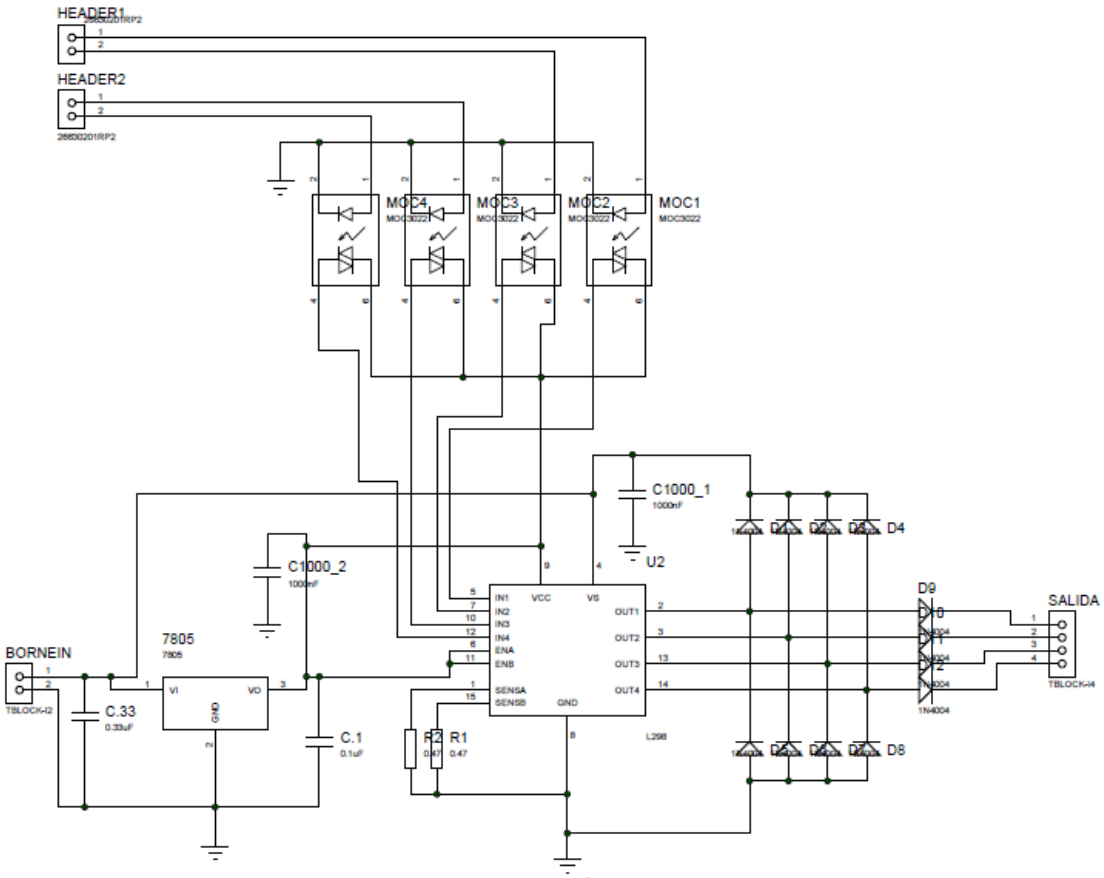


Figura 4.4 *Diagrama electrónico en Proteus ISIS.*

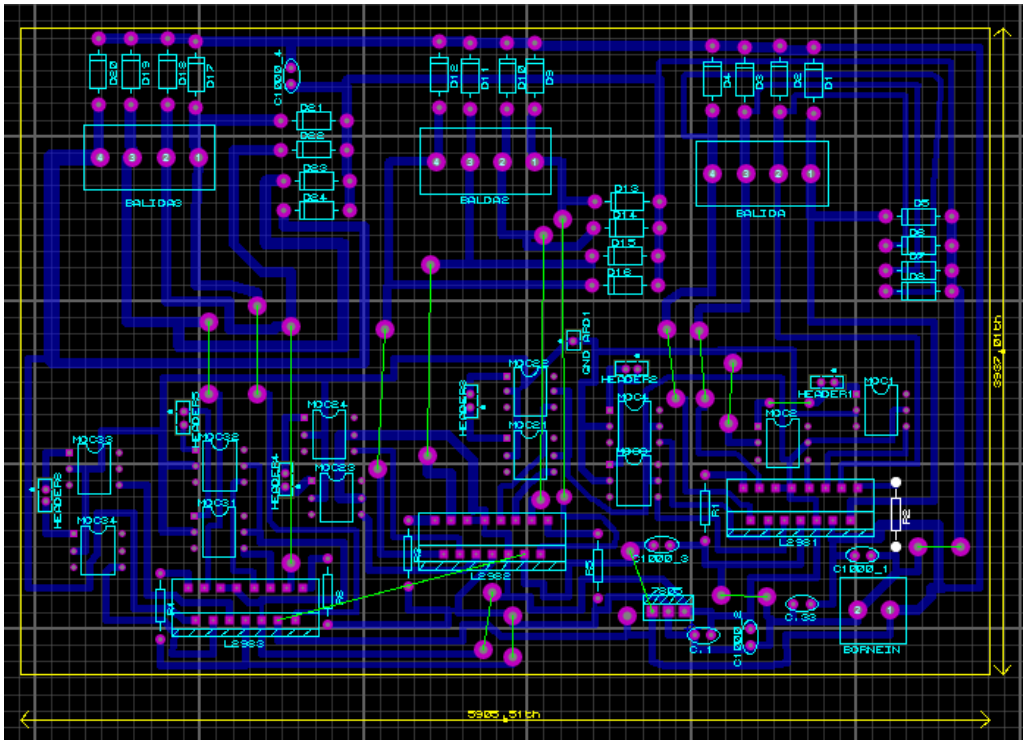


Figura 4.5 *Diagrama de rutas en Proteus Ares. Se aprecian en verde las terminales que necesitaron de un puente o salto de conexión.*

Ares incluye también un motor de presentaciones posibilitando la visualización en tres dimensiones de la tarjeta antes de realizar el prototipo. Esta capacidad no solo ayuda en el diseño de la placa, sino que también proporciona información de su posible altura. Se puede igualmente disponer de diferentes vistas de la imagen visualizada, girarla, hacer zoom sobre ella, etc. También es posible seleccionar el nivel de detalle de la visualización.

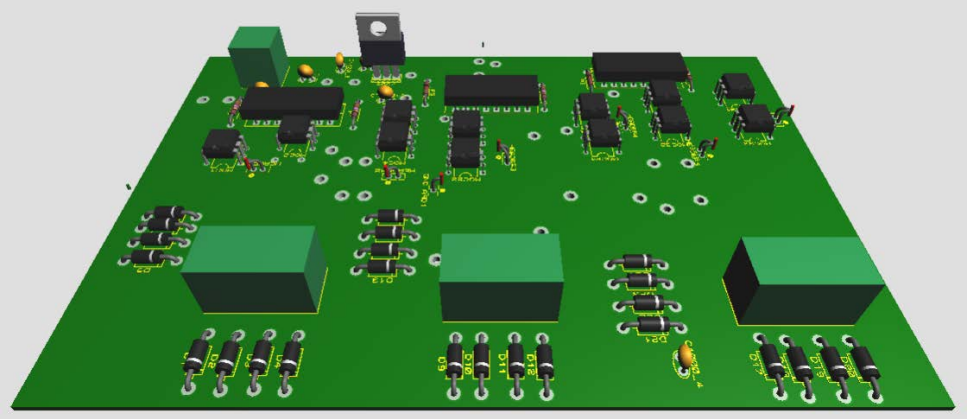


Figura 4.6 *Modelo virtual de la placa PCB (superior) generado en Proteus ARES.*

4.5 Fabricación de la placa PCB

El proceso para crear la placa PCB de este proyecto fue imprimir en papel el plano de las rutas generadas en ARES y por medio de calentamiento de tinta, transferir dicho plano a la placa fenólica; una vez entintada la placa fue sumergida en un recipiente con óxido férrico que hizo la labor de retirar el cobre excesivo definiendo al mismo tiempo las rutas de conexión.

Como se ha comentado previamente fue necesaria la anexión de material extra, en este caso soldadura de estaño, para reforzar las rutas más delgadas o completar la continuidad. Hay que tomar en consideración que la impresión en papel puede presentar errores pequeños así como los existen al momento de transferir la tinta del papel a la placa fenólica y finalmente también cuando se es retirado el material con el óxido férrico. Por los motivos mencionados es probable que exista un error en la continuidad o en la formación de una vía de cobre por lo que siempre es recomendable reforzar las vías menos definidas.

Finalmente se perforó la placa para la inserción de los pines de cada circuito integrado y realizar las conexiones y soldaduras pertinentes. Se optó por ocupar un mini drill con una broca para circuitos impresos de 0.8 mm. Se perforaron las cavidades para los distintos circuitos integrados y posteriormente se procedió a colocar cada dispositivo al mismo tiempo que se soldaban sus pines a la placa fenólica; cabe resaltar que este proceso se realizó de una forma progresiva y los circuitos integrados más pequeños

fueron los primeros en ser colocados de manera que la altura de los elementos más grandes no estorbase al colocar los demás elementos.

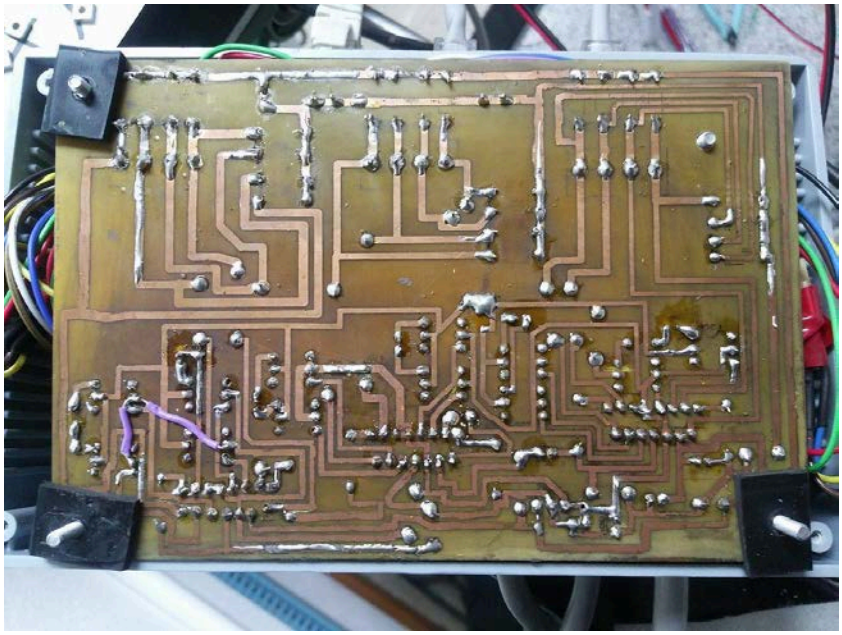


Figura 4.7 *Vista inferior de la placa de circuitos impresos.*

4.6 Elementos periféricos y seguridad

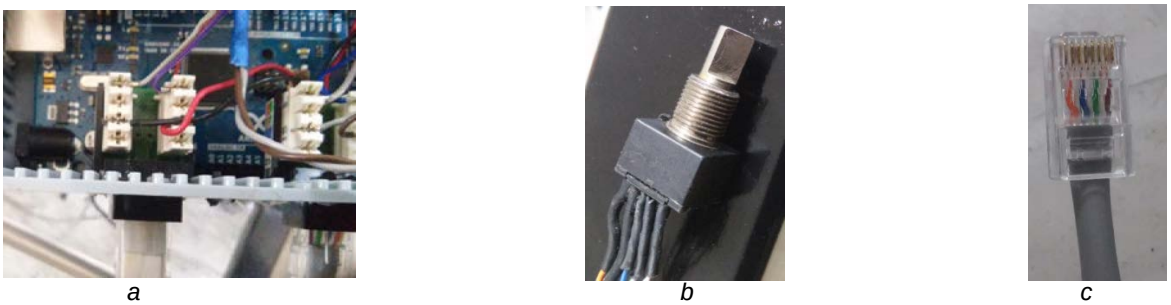
La placa de circuitos impresos es una de las partes más importantes del circuito eléctrico, alimentando los motores y regulando un voltaje fijo de cinco voltios. No obstante una parte fundamental para el funcionamiento de este proyecto son los elementos periféricos ajenos a la placa PCB. Dentro de estos elementos se encuentran seis sensores y el arduino MEGA.

En el caso de la placa Arduino, se considera como el elemento lógico externo a la computadora. Como microcontrolador que es, recibe las órdenes del computador y las transfiere en su salida como pulsos que, dependiendo de su orden y en combinación con los puentes H y demás, entran en contacto con la placa PCB y permiten el giro de los motores. Pero al no ser parte directa de la placa es considerado el Arduino MEGA como un elemento periférico, igualmente porque existe tanto salida como entrada de datos y es conectado a los pines de entrada de la placa fenólica por medio de alambre de conexión. Los motores en sentido estricto son ajenos a la placa PCB pero al ser sólo conectadas sus terminales en las salidas de la placa no será contemplado como elemento periférico

Los sensores que fungen como elementos periféricos son seis en total, tres de ellos son los integrados del encoder de cuadratura que son los que monitorean el giro de cada motor. Este sensor funciona con cuatro pines generales, dos para alimentación que habilitan el funcionamiento y los otros dos como salidas de los dos receptores infrarrojos

internos del circuito integrado. Los siguientes tres sensores son pulsadores normalmente abiertos; estos sensores cuentan con tres terminales de conexión, dos de ellas para alimentación y una tercera de señal, tiene la misma función que un botón pulsador y su función en este proyecto es definir un punto de origen físico en el posicionador. Cabe resaltar que todos los sensores son alimentados por los diferentes pines de voltaje fijo de cinco voltios de la placa de circuitos impresos.

Se ha tomado en consideración que los integrados de encoder de cuadratura cambian su posición con respecto al origen absoluto dependiendo de qué tanto avance el posicionador, es decir, a medida que avanza el posicionador dos de los integrados avanzan junto con el sistema. Por tal motivo y considerando que estos sensores son de gran importancia para el funcionamiento del posicionador, la conexión de estos sensores a la placa arduino (señales) y placa PCB (alimentación) se realizó por medio de conectores RJ45 (Figura 4.7). Estos conectores usados principalmente en aplicación de cable Ethernet, cuentan con ocho terminales, no obstante el encoder de cuadratura sólo requiere cuatro de ellas. Se han utilizado en total 3 pares de conectores (hembra y macho) de modo la conexión de cada encoder sea independiente, firme y segura sin importar si se mueve o no el posicionador, además se considera una medida preventiva en caso de que sea necesario el desacople del sensor y/o se requiera su cambio.



a *b* *c*

Figura 4.8 *Conexión de encoder con RJ45 a) Conector hembra. b) Encoder de cuadratura BOURNS EM14 con cuatro terminales. c) Conector RJ45 macho.*

La seguridad dentro del circuito de la placa PCB, Arduino y elementos periféricos se ha hecho primeramente considerando una descarga o un error en las conexiones. Se ha utilizado el modelo de optoacoplador MOC3011 en las terminales de señal de entrada de cada puente H. Se debe considerar que para un circuito de esta magnitud se trabaja con corrientes no tan altas pero lo suficientemente peligrosas para dañar los elementos más sensibles. Por tal motivo se necesita crear un aislamiento entre las terminales tanto de corriente como en las tierras involucradas.

Un optoacoplador combina, en un mismo integrado, un LED y un colector o fotodetector. El principio básico de un optoacoplador es que, a partir de la recepción de impulsos luminosos del LED el colector permite o no el paso de corriente en otra sección del circuito. La luz que se emite incide en el fotodiodo y crea una corriente inversa que pasa por el resistor [20], al ser un elemento óptico la velocidad de respuesta es mucho mayor en comparación con dispositivos mecánicos como relevadores, ofreciendo una respuesta rápida sin perjudicar la facultad de aislamiento.

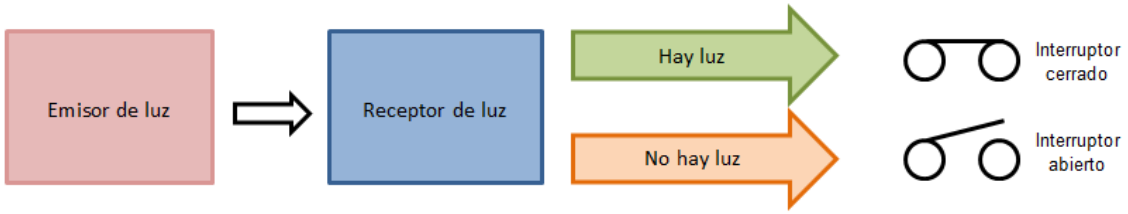


Figura 4.9 *Funcionamiento básico de un optoacoplador.*

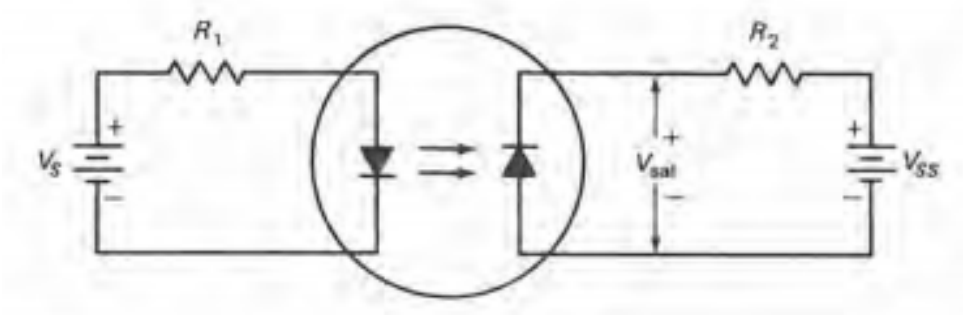


Figura 4.10 *Circuito de un optoacoplador [20].*

La principal ventaja del optoacoplador es el aislamiento eléctrico entre el circuito del fotodiodo; por este motivo también se le conoce como “optoaislador”, el único contacto entre los circuitos de entrada y salida es el haz luminoso, el cual representa una ventaja importante contra otros dispositivos aislantes cuando se habla de velocidad de respuesta.

En este proyecto se busca trabajar con tres motores de manera que la tierra que cierra el circuito de los sensores debe ser independiente de la tierra de los motores. Además se considera que el uso de aisladores mecánicos tales como relevadores resultaría de bajo aprovechamiento debido al bajo tiempo de respuesta que presentan. Por tal motivo un optoacoplador es una opción indicada manteniendo la seguridad esperada sin comprometer la velocidad de reacción.

4.7 Pruebas de funcionamiento

Se realizaron pruebas en la placa alimentando el circuito directamente del tomacorriente resultando en un correcto funcionamiento del circuito. El voltaje regulado del transformador y rectificador tiene una salida de 12.07 voltios en corriente directa, y el voltaje regulado por medio del integrado 7805 tiene una salida de corriente directa de 4.98 voltios.

Se realizaron pruebas con cada motor a pasos de manera independiente así como todos en conjunto resultando en un correcto funcionamiento y con el par suficiente para mover los ejes del posicionador sin pérdidas de pasos ni detenciones. De igual manera los seis sensores que se ocupan en el posicionador fueron correctamente alimentados de manera que su comunicación con la placa Arduino y la computadora no presenta fallas.

En el aspecto de seguridad el circuito no presenta un sobrecalentamiento a pesar de estar confinado dentro de una caja plástica. La placa PCB presenta un ligero aumento de la temperatura sin ser excesiva para derretir soldadura, dañar los dispositivos, fundir plásticos o representar un peligro de quemadura. Los optoacopladores han funcionado de manera correcta permitiendo el paso de señal cuando es requerido sin poner en riesgo el resto del circuito.

CAPÍTULO V
Programación y control del sistema

5.1 Introducción

Para el desarrollo de este proyecto se ha contemplado crear un circuito capaz de controlar tres motores, alimentar los sensores del sistema, que sea capaz de ser comunicado con una placa Arduino MEGA y ser alimentado de un tomacorriente. No obstante el posicionador requiere también de una programación para mover los motores en un orden y dirección deseados, también debe dar la oportunidad de elegir distancias, realizar lecturas y almacenar dichas lecturas en un archivo de texto.

Al ser programado en LabVIEW la parte lógica del proyecto fue escrita en leguaje G y C++. Por medio del uso de una interfaz, también creada en LabVIEW, se ingresan los datos necesarios para un buen funcionamiento. Bajo esta idea, toda la programación pertinente fue hecha en estrecha relación con la interfaz, haciendo que la comunicación hombre-máquina fuera práctica, no fuera necesario un conocimiento de programación para poder utilizar la interfaz y, por extensión, tampoco el posicionador.

La programación destinada para controlar el posicionador cuenta con tres etapas para que no existan errores al momento de hacer las lecturas y el movimiento de barrido. En primer lugar se busca partir de un origen absoluto, situado en una de las esquinas del posicionador; de este punto el posicionador moverá una barra donde será montado el emisor ultrasónico (o el receptor ultrasónico), que se considerará como punto terminal o punto terminal del posicionador, hasta el segundo origen seleccionado por el usuario. Cuando el punto terminal llegue a este segundo origen comenzará un desplazamiento igualmente seleccionado por el usuario, ya sea en línea recta, en un plano o en un volumen; mientras se mueve el posicionador en dicho movimiento, se habrán de realizar las lecturas de los canales del osciloscopio y se guardarán en un archivo de texto. Al final de este desplazamiento, el punto terminal debe regresar a su posición de origen absoluto en sus coordenadas iniciales, de esta manera el posicionador quedará listo para la próxima serie de lecturas.

A pesar de que por medio de la interfaz de LabVIEW se controla el posicionador se debe tomar en consideración que el uso de LabVIEW consume muchos recursos de una computadora. Se contemplaba mover el punto terminal a su segundo origen, realizar y guardar las lecturas y luego regresar al origen absoluto. Sin embargo al anexar ciertas partes de código tales como la lectura del osciloscopio o el almacenamiento de datos, el programa completo colapsaba al compilar imposibilitando cualquier acción. Hay que tomar en consideración que la lectura del osciloscopio y la comunicación con un Arduino hacen que LabVIEW requiera de más recursos, si la computadora utilizada no es suficientemente rápida resultará en una detención del programa o retraso del mismo. Por

tal motivo se ha optado por separar el proceso en dos programas. El primer programa comprenderá la ubicación del punto terminal en el segundo origen, también forma parte de este programa el retorno al origen. El segundo programa realizará el desplazamiento de barrido, la lectura de los canales del osciloscopio y el almacenamiento de datos. De este modo la tarea solicitada con el posicionador se completa sin sobrecargar de procesos la computadora.

5.2 Programación en LabVIEW

LabVIEW Laboratory Virtual Instrument Engineering Workbench, es una herramienta gráfica para pruebas, control y diseño mediante la programación. El lenguaje que usa se llama lenguaje G. Es una plataforma capaz de crear entornos para la comunicación e interconexión de diversos sistemas, dispositivos, tarjetas controladoras y subrutinas lógicas para el desarrollo de una aplicación [9].

La idea de National Instruments al crear LabVIEW fue la concepción de una plataforma con una sintaxis de programación gráfica que facilita visualizar, crear y codificar sistemas de ingeniería, LabVIEW ofrece reducir tiempos de pruebas, análisis de negocio basado en datos recolectados y facilitar cualquier trabajo de ingeniería. Está diseñado para incorporarse con otro software, ya sea métodos alternativos de desarrollo o plataformas de fuente abierta, para dar al usuario el mayor número de herramientas para la implementación de un proyecto.

Una ventaja que presenta LabVIEW ante otros softwares de programación como Processing o Java es la facultad de crear interfaces de manera muy sencilla. Al ser un lenguaje gráfico se permite más fácilmente la programación de los elementos que componen a interfaz, es decir, los mismos elementos visuales que ofrece LabVIEW son al mismo tiempo elementos de programación. Igualmente este software cuenta con un vasto número de bloques, cada uno con una función específica, que permiten mediante un diagrama realizar rutinas y procesos lógicos deseados. Cada parte dentro del Panel Frontal tiene su representación lógica en el diagrama de bloques, esto significa que cada elemento visual dentro de la ventana que usará el usuario tiene un bloque, modificable y programable, que lo representa dentro del código.

Con esas ideas presentes se puede decir que LabVIEW da la facilidad de sólo buscar un elemento que deseemos incluir en la interfaz, colocarlo donde se desee, y conectarlo con los demás elementos por medio de bloques. Al final de este proceso resultará una interfaz a gusto del programador o un usuario, capaz de realizar tareas complejas, realizada de una manera más interactiva, sencilla y sin la necesidad de líneas de código.

5.2.1 Herramientas básicas de LabVIEW

LabVIEW es una de las muchas herramientas de National Instruments que involucran una interfaz y un lenguaje de programación. Para efectos elementales LabVIEW es uno de los softwares más completos y proporciona una amplia gama de herramientas para crear un

entorno de desarrollo fácil de entender pero con grandes alcances. Cuenta principalmente con dos secciones importantes: el Panel Frontal y el diagrama de bloques.

El Panel Frontal de LabVIEW es la interfaz que entra directamente en contacto con el usuario. Es un conjunto de elementos gráficos como indicadores, botones y perillas que interactúan con la persona que manipula el programa. Estos controladores son terminales lógicas dentro del diagrama de bloques del Instrumento Virtual (Virtual Instrument VI) (programa gráfico sobre la plataforma LabVIEW).

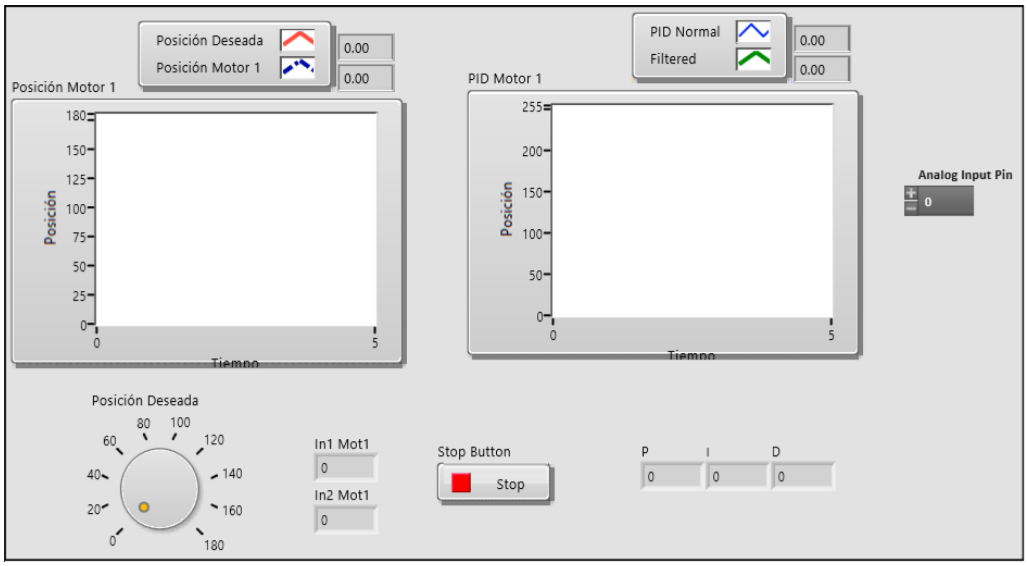


Figura 5.1 *Ejemplo de Panel Frontal de LabVIEW.*

La ventaja del Panel Frontal, aparte de ser la parte visual de la interfaz, es que puede reemplazar un panel de operación físico, es fácil modificar su distribución y puede simular el funcionamiento y apariencia de múltiples instrumentos o equipo de medición y comunica estos datos directamente con el diagrama de bloques. LabVIEW da la opción al programador de crear un Panel Frontal final que será el ejecutable del programa, pudiendo modificar el tamaño de los elementos, colores, distribución e incluso tamaño y diseño de la ventana de la aplicación.

LabVIEW tiene también la cualidad de trabajar con un sistema de bloques al que denomina “Diagrama de bloques”, dentro de este diagrama se pueden colocar los distintos elementos de programación que van desde bloques de funciones lógicas hasta bloques relacionados con la teoría de control.

Los bloques de LabVIEW pueden ser comunicados unos con otros de diversas maneras dependiendo del tipo de dato que se maneje. Es de considerar importante que existen ciertos bloques que sólo funcionan con tipos específicos de datos, es decir, bloques que manejan únicamente cifras de números enteros, valores booleanos o bien sólo de escritura. La manera en que estos bloques sean colocados así como las funciones que realizan, se evalúa por el compilador de LabVIEW, en caso de ser correcta y lógicamente aceptable el programa habrá de realizarse.

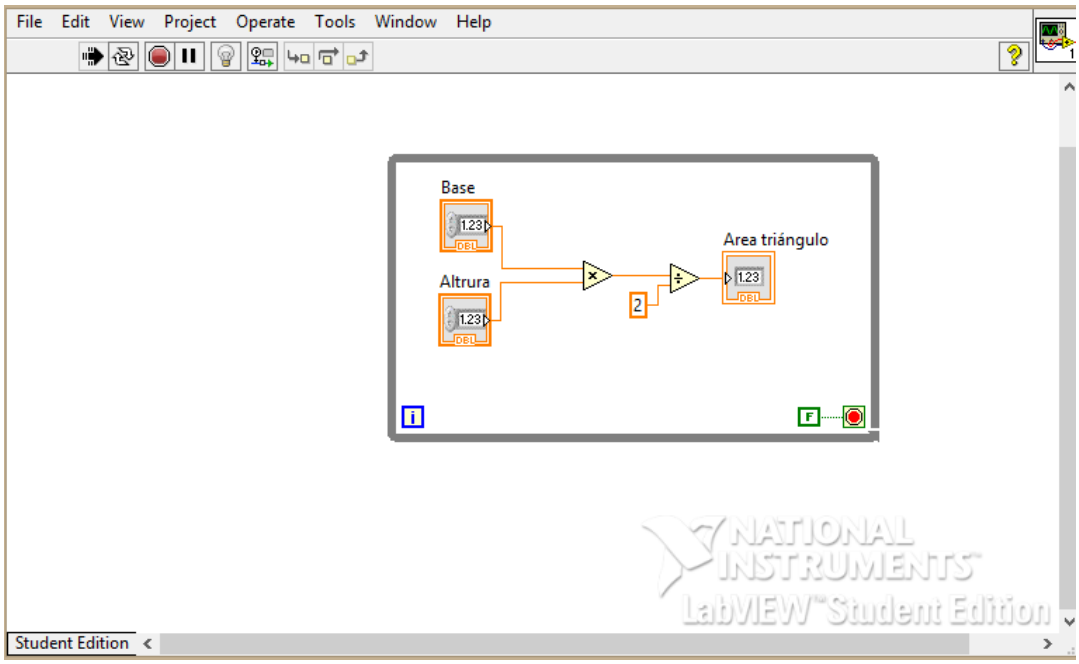


Figura 5.2 *Ejemplo de diagrama de bloques en LabVIEW.*

5.2.2 Comunicación con la tarjeta Arduino MEGA

En el caso de la relación de la plataforma Arduino y el software de LabVIEW se ha aprovechado la comunicación serial de las placas de arduino para poder comunicar ambos sistemas. Anteriormente esta comunicación recibía el nombre de “LIFA_base” en la cual era necesaria una instalación de drivers, toolkits y firmwares dentro de la computadora para poder comunicarla con una tarjeta. Además, era necesario cargar un programa específico en la tarjeta Arduino para que esta comunicación fuera posible. La desventaja sujeta a este antiguo método es que era necesario siempre contar con dicho programa, el cual tardaba mucho en compilar y cargar y que además solía presentar problemas de conexión al momento de trabajar con él. Por tal motivo desde el año 2014 LabVIEW ha recomendado el uso de un toolkit llamado “LINX” con notables mejoras para comunicar una gama amplia de tarjetas incluidas las de Arduino.

5.2.3 Configuración de MakerHub LINX

A partir de la serie de inconvenientes mencionados, National Instruments creó otro firmware y un segundo toolkit para vincular ambos sistemas. El LINX LabVIEW MakerHub hace más fácil conectarse con plataformas embebidas comunes como chipKIT, Arduino y NI myRIO, así como sensores comunes incluyendo acelerómetros, sensores de temperatura y sensores ultrasónicos de distancia. En este firmware es más fácil comunicar el arduino con LabVIEW a partir de opciones seleccionables desde la barra de herramientas del programa sin la necesidad de cargar programa alguno al Arduino de manera independiente [9]

Para la configuración del firmware dentro de LabView, en la ventana “Tools” → “MakerHub” → LINX → LINX Firmware Wizard se desplegará un submenú de selección de LINX.

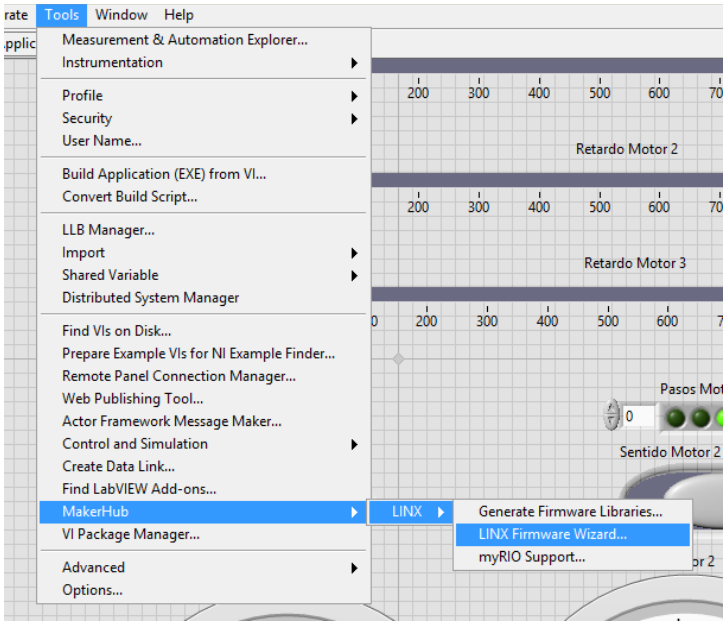


Figura 5.3 *Ejemplo de diagrama de bloques en LabVIEW.*

A partir del menú desplegado se puede seleccionar el tipo de tarjeta, modelo, puerto COM donde se encuentra conectado el dispositivo y el tipo de comunicación que será utilizado entre la computadora y la tarjeta. Al instalar LINX, se adquirió una nueva opción en el menú de bloques con la cual es posible inicializar, leer, escribir y cerrar la comunicación con alguno o varios de los pines en el arduino. De esta manera, sólo se necesitó inicializar el “Wizard” de la barra de herramientas para poder seleccionar el tipo de tarjeta y tipo de comunicación.

5.2.4 Comunicación entre LabVIEW y MakerHub LINX

Una vez que se realizó la comunicación física entre ambos sistemas por medio del cable USB y fue cargado el programa al arduino en el firmware, se pudo comenzar a programar en código G de manera adecuada. Para que se permita la comunicación entre el arduino y LabView y se pueda compilar el VI, se debe contar con una apertura y un cierre de comunicación. Primeramente se realizó la apertura de comunicación mediante el bloque *Initialize* dentro del menú de funciones de LabView. El bloque *Initialize* tiene como única función comenzar el intercambio de datos entre la tarjeta Arduino y LabView, en este bloque se elige el puerto COM con el que se trabajará y la velocidad de transmisión de datos.

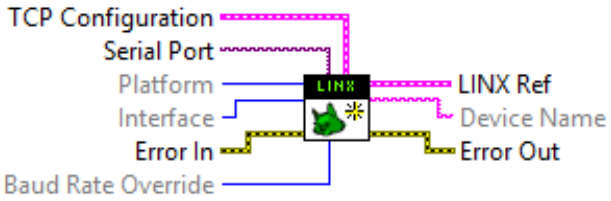


Figura 5.4 *Bloque Initialize.*

Una vez creado el canal de comunicación a través de la función *Initialize* se permitió la lectura y escritura en los pines del arduino mediante los bloques de lectura (*Read*) y escritura (*Write*) del toolkit de Makerhub.

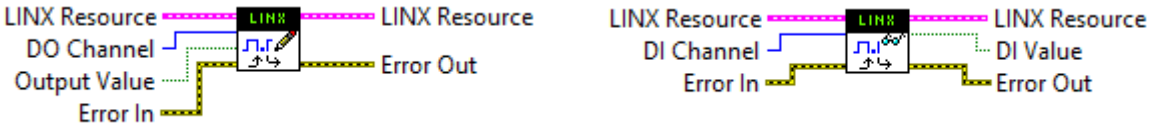


Figura 5.5 *Bloques Digital Write (izquierda) y Digital Read (derecha).*

Indistintamente de la ejecución de las partes, el flujo de datos debe dirigirse al bloque de cierre, necesario para un buen funcionamiento, de lo contrario el programa no podría compilar ni ejecutarse. Para esa labor se ocupó el bloque *Close* de MakerHub

Figura 5.6 *Bloque Close.*

5.2.5 Control de motores a pasos

En esta sección se verá la forma en que se controla sólo un motor a pasos por motivos de simplificación para evitar la repetitividad del mismo procedimiento para los otros dos motores. Siendo el motor a pasos un tipo de motor basado en la secuencia ordenada de señales de alto y bajo, se realizó este programa con una serie de arreglos y un registro de corrimiento.

Debe puntualizarse que cada estructura *For Loop* encargada del giro de uno de los motores es independiente de la lectura de los sensores o de los algoritmos de lectura o almacenamiento de datos. Esto se hace para que cada motor se controle de manera independiente.

5.2.5.1 Registro de corrimiento

El funcionamiento del motor paso a paso con LabView se realiza mediante un arreglo que cambia de índice constantemente. Se utilizó la herramienta *For Loop* ya que, al ser el giro

del motor un evento cíclico, este se repetirá hasta que se detenga la condición de marcha del motor.

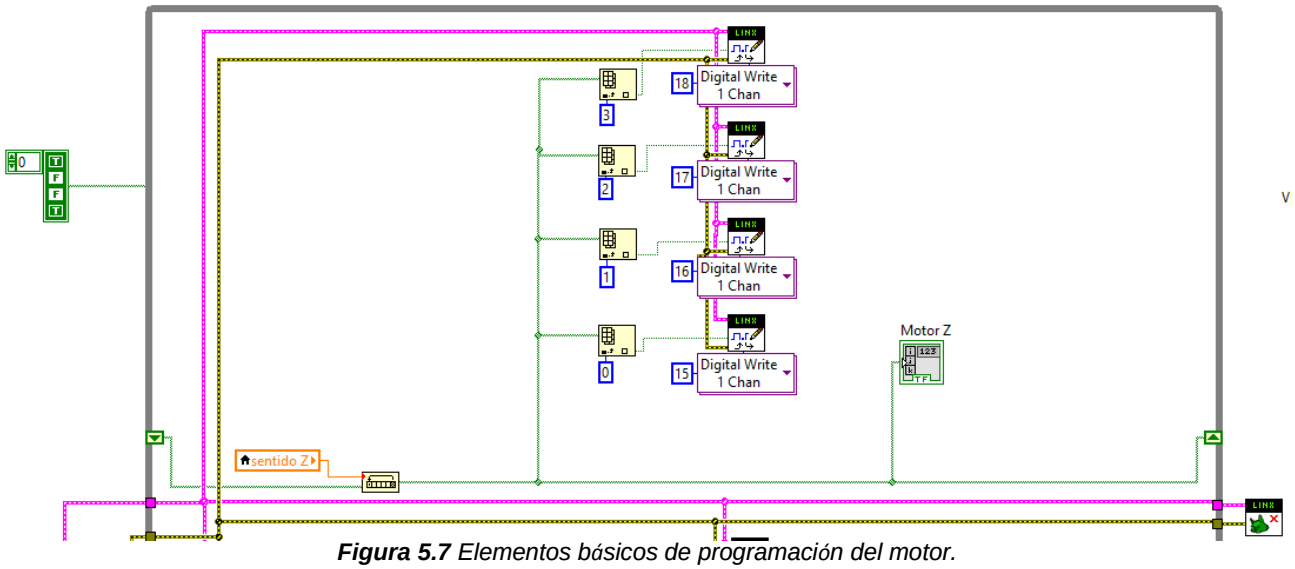


Figura 5.7 *Elementos básicos de programación del motor.*

Al momento de inicializarse el programa se manda una señal de alto al primer elemento del arreglo y este lo comunica al pin correspondiente del arduino. Por medio de la herramienta *shift register,* una vez finalizado el ciclo *For Loop* por primera vez, se modifica el índice del registro de corrimiento y se avanza una unidad en el índice del arreglo. De esta manera cuando se ejecute la estructura *For Loop* por segunda vez, el segundo elemento del arreglo sea el que se encuentre en alto y los demás elementos en bajo. Esta acción se repite hasta que el índice llega a un valor de tres donde el registro vuelve a tomar un valor de cero y el procedimiento completo se vuelve a ejecutar.

Visualmente esta variación es observable desde el panel de control a partir de un arreglo booleano en forma de barra de leds. Dependiendo qué elemento del arreglo se encuentre en alto, se modificará el led correspondiente.

Figura 5.8 *Bloque Close.*

Mediante el bloque *Rotate 1D Array* y con la ayuda de la función *shift register,* se permite el cambio o “rotación” de los valores del arreglo. De esta manera, siendo la estructura *Foor Loop* una estructura cíclica, el índice de cada arreglo cambia de manera infinita.

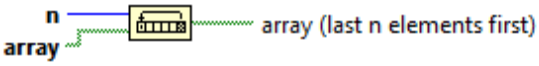


Figura 5.9 *Bloque Close.*

La ventaja al usar la función *Rotate 1D Array* es la permisión de “saltar” de elemento dentro del arreglo al asignar un número en el índice “n” del bloque; de este modo si se coloca un número 2 en dicho índice, el arreglo se moverá de dos en dos, si se coloca un 3, se moverá de tres en tres y así sucesivamente. El uso del bloque *Index Array* tiene la función de asignar una posición a un elemento de un arreglo, es decir, al escribir por ejemplo un número cuatro en la terminal de este bloque, se le estaría asignando la posición 4 del arreglo (hay que recordar que los arreglos se contabilizan desde cero). De este modo cada vez que el bloque *Rotate 1D Array* avance, se realizará una acción distinta dependiendo de qué número se haya asignado a cada bloque *Index Array* en el programa.

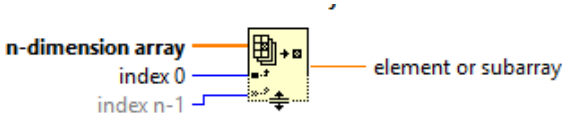


Figura 5.10 *Bloque Index Array.*

Dentro del algoritmo general del posicionador, el valor del índice “n” del bloque *Rotate 1D array* permanece constante en la unidad (positiva o negativa) considerando que se desea un cambio consecutivo de uno en uno dentro de este procedimiento. Ya que este cambio se comunica al Aruino y este a su vez al puente H y de ahí al motor, se traduce como una secuencia de pulsos con la cual el motor a pasos puede girar.

5.2.5.2 Velocidad de giro de los motores

La velocidad de giro se realizó mediante retrasos controlados por el usuario. Para la sección correspondiente al Panel Frontal, se ocupó un control numérico con el fin de modificar manualmente la velocidad de todos los motores en un intervalo en milisegundos propuesto por el usuario. De no ser colocado algún número dentro de la casilla de control numérico, las variaciones en los pulsos del motor se realizarán a la velocidad de la computadora (esto no significa que el motor girará a esa velocidad, simplemente el procesamiento se realizará sin detención alguna).

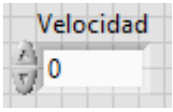


Figura 5.11 *Control de retrasos en el Panel Frontal.*

Dentro del código G de LabView, la variable de control de retardo se traduce como una variable del tipo double, la cual es dirigida al bloque *Wait Until Next ms Multiple* que se encarga de provocar un retardo en el programa. Es importante aclarar que este proceso de retardo, al estar contenido dentro del cuadro de una de las estructuras *For Loop,* sólo afectará a los elementos contenidos dentro de esa estructura, los demás cuadros de estructura *For Loop* destinados a la lectura de los sensores, osciloscopio o

demás procesos, permanecerán a su propia velocidad independiente. Este fenómeno de paralelismo entre estructuras es una muy importante herramienta de LabVIEW.

Figura 5.12 *Bloque Wait Until Next ms Multiple conectado al control de retraso.*

5.2.5.3 Escritura de los pines digitales

Cada vez que se modifica el índice de corrimiento (*shift register)*, se modifica la señal de alto con él, de manera que a medida que cambia el índice de corrimiento, la señal de alto se dirige a una posición diferente del arreglo. Este procedimiento continúa de manera cíclica hasta que se detiene el programa.

En la figura 5.13 se muestra la forma de conectar el bloque de escritura digital del Arduino. Se puede observar, aparte de las líneas de comunicación y de error, la línea que indica el pin del Arduino que se usará como salida digital; en este caso se ha elegido el pin 11 del Arduino MEGA como pin destinado a la escritura. Igualmente es apreciable la terminal del bloque llamada “Valor de entrada”, este elemento del bloque de escritura recibe la información que se desea escribir de manera binaria, es decir, si se escribe un 0 la placa Arduino enviará una señal de LOW, si se escribe 1 se enviará una señal de HIGH; estas señales se observan físicamente como un voltaje bajo cercano a cero o un voltaje alto alrededor de 5 voltios respectivamente. Es importante mencionar que por condición de diseño, una señal digital de escritura del Arduino tiene un voltaje constante de 5 volts.

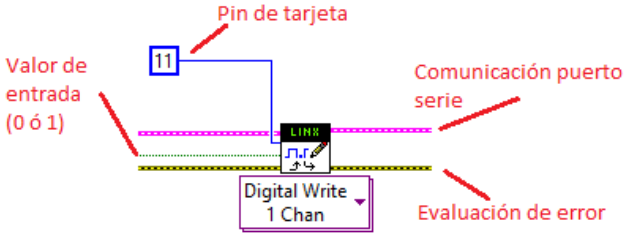


Figura 5.13 *Terminales del bloque Digital Write.*

Si se toma el valor de salida de algún bloque *Index array* (correspondiente a un elemento del arreglo) y se conecta esa salida al valor digital de entrada del bloque *Digital Write* (Figura 5.14), uno de los elementos del arreglo estará en alto mientras los otros están en bajo, significando que el Arduino envía un pulso a uno de los cuatro pines del motor, al realizarse una iteración, ese pulso se desplaza al siguiente pin y así sucesivamente formando un tren de pulsos y un ciclo de giro.

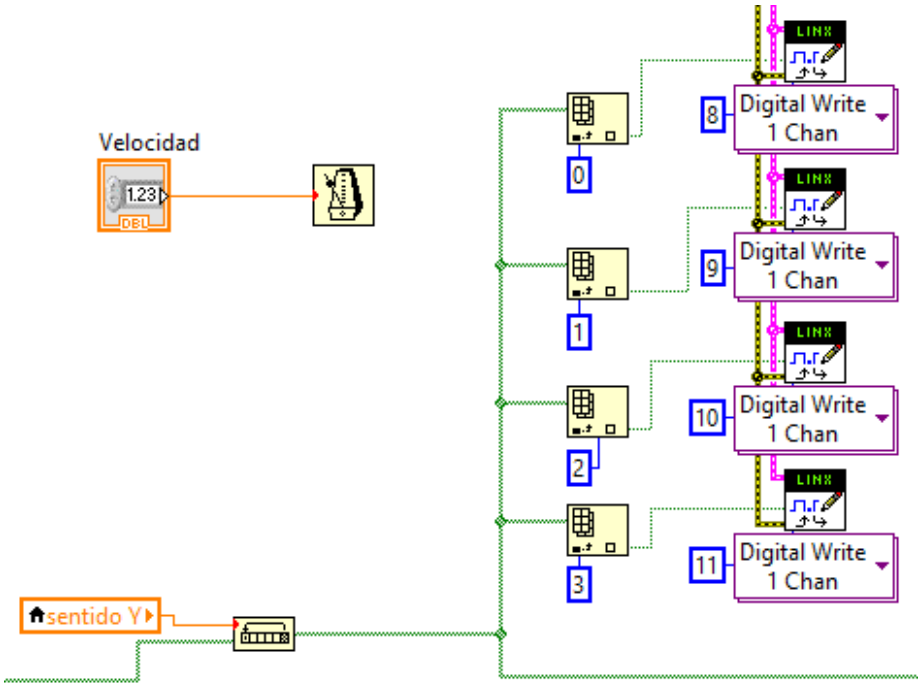


Figura 5.14 *Estructura de conexión en los cuatro pines del motor.*

Entendido cómo funciona este proceso y resumiendo, se tiene que el bloque *Rotate 1D array* asigna un valor unitario o cero que permite un "avance" en los elementos del arreglo (variable *sentido Y* en la figura 5.14). Según avancen (o retrocedan) dichos elementos, se realizará una acción dependiendo el elemento en juego en ese momento y el índice que fue asignado a uno de los bloques *Index array*, resultando en que, por ejemplo, si la variable sentido Y tiene un valor de 1, los elementos del arreglo avanzarán de uno en uno. Empezando por el índice del arreglo 0, se realizará la escritura del pin 8. Cuando el índice del arreglo sea 1, se desactivará el pin 8 y se activará el pin 9. Este proceso se repetirá hasta que el índice del arreglo llegue a 3, entonces todo el procedimiento se repetirá de nuevo.

5.2.5.4 Control de sentido de giro del motor

Para controlar el sentido de giro del motor se usó la función *Rotate 1D array*. Una peculiaridad de este bloque es que, al ser colocado en lugar de "n" un número positivo, se desplaza ese número de elementos dentro del arreglo; con un cero, el índice del arreglo permanece estático en el elemento donde se haya quedado. Con un número negativo, se invierte el sentido de desplazamiento dentro del arreglo y por ende, el cambio de sentido del motor.

Figura 5.15 *Sentido del motor controlado por la variable Sentido X.*

Bajo la idea planteada si se introduce un número -2 en la variable *sentido X*, el índice del arreglo se moverá de dos en dos en sentido contrario. Suponiendo que un momento dado el índice de un arreglo de cuatro elementos apunta al elemento 3, en la siguiente iteración de la estructura *For Loop* el índice de dicho arreglo ahora se encontraría en el elemento 1. Como se ha dicho previamente, se espera un avance de uno en uno con el fin de completar una secuencia para girar el motor, de modo que el único valor negativo tomado por la variable *sentido X* (o *sentido Y* o *sentido Z*) es el de -1.

Entonces si se hace un análisis de lo que sucede dentro del algoritmo de giro del motor, encontramos que dependiendo del valor del variable *sentido X*, el motor correspondiente al eje X del posicionador girará en un sentido positivo (dextrógiro) si el valor de *sentido X* es 1, se detendrá si es 0 y girará en sentido negativo (levógiro) si la variable toma el valor de -1. El mismo procedimiento sucede en para las variables *sentido Y* y *sentido Z* correspondientes a los ejes Y y Z respectivamente.

5.2.6 Lectura y despliegue de encoders

Esta sección del programa se dirige a la lectura de cada uno de los tres sensores que monitorean el giro de los motores, sus pines de lectura, su despliegue en el Panel Frontal y el conteo de sus pasos. Para este proyecto se han usado tres sensores ópticos BOURNS EM14.

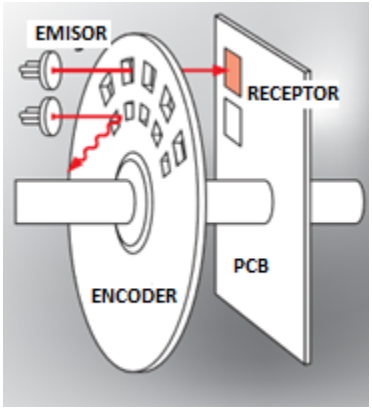


Figura 5.16 *Funcionamiento de un encoder de cuadratura.*

A pesar de que los encoders de cuadratura son muy útiles fue necesario reducir a la mitad la cantidad de lecturas. La principal razón fue que, trabajando bajo la IDE de Arduino no existía problema alguno con el conteo de los pasos, la velocidad de procesamiento de la computadora permitía que en un giro relativamente rápido fueran

captados todos los pulsos del encoder. Sin embargo al programar con LabVIEW se encontró que las lecturas se entorpecían. Se observó que a pesar de que la velocidad de giro en el encoder fuera baja, había una pérdida de lecturas al dar una revolución completa. Dicho de otra manera, el encoder de cuadratura BOURNS EM14 da un total de 256 pulsos por revolución si se trabaja con los dos canales al mismo tiempo, con la IDE del arduino se tomaba la lectura de esos 256 pulsos, en cambio en LabVIEW se contabilizaban menos pasos en una revolución. Se considera que este problema surge por la alta demanda de recursos de la computadora al momento de la ejecución de un VI en LabVIEW.

Para solucionar dicho problema se optó por reducir las lecturas a la mitad, capturando un total de 128 pulsos por revolución. Debido a que hay una mayor distancia entre un pulso y otro (180º en lugar de sólo 90º) y se reduce la lectura de un canal, es más fácil para la computadora procesar los datos y trabajar de manera más eficiente. Se consideró que si bien se perdía una importante cantidad de lecturas, la resolución del posicionador en general sería aún adecuada. Tomando en cuenta que por cada paso de un motor a pasos, se da un avance de 15.5 micrómetros y el motor tiene un total de 200 pasos por revolución, en una revolución se tendrían 3100 micrómetros. Con este hecho, la división de la distancia de una revolución (3100 micrómetros) entre el total de pasos del encoder (128 pasos), resultaría en un total de 24.21875 micrómetros por cada paso del encoder. La resolución es adecuada puesto que la usada en estudios ultrasónicos es de 25 micrómetros.

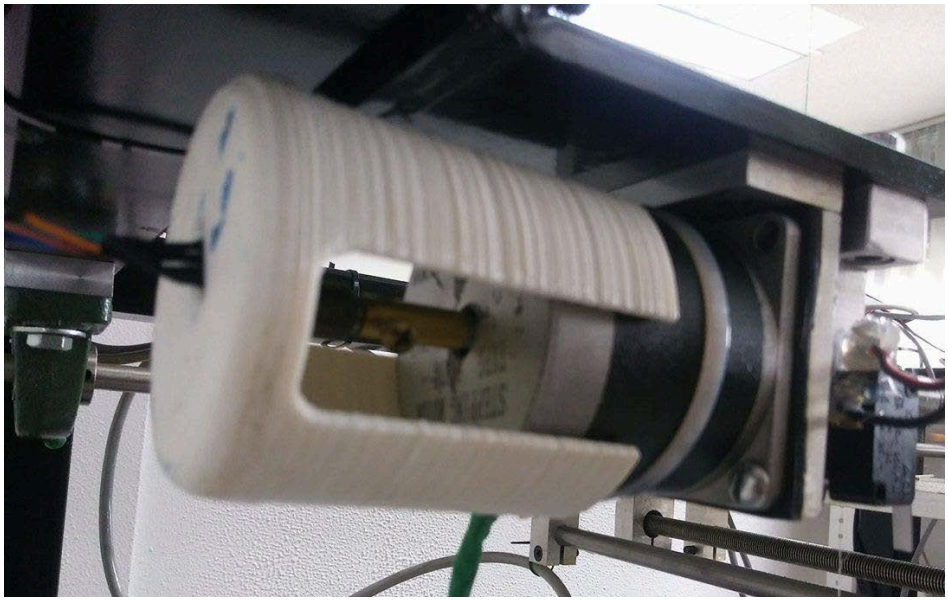


Figura 5.17 *Acoplamiento de encoder de cuadratura.*

5.2.6.1 Lectura de encoders de cuadratura con LabVIEW

La forma en que el sensor BOURNS EM14 funciona es a partir de un tren de pulsos de alto a bajo o viceversa de forma secuencial. Es decir, ambas señales no pueden cambiar al mismo tiempo de un estado a otro.

Con base en el razonamiento anterior se pueden tratar esos cambios de alto y bajo como señales digitales. Por este motivo dentro de LabView se ha contemplado utilizar la función *Digital Read* de manera que se identifiquen los estados de alto y bajo; cada vez que cambie un valor, habrá de contabilizarse dicho cambio. Al igual que su contraparte de escritura, el bloque de lectura *Digital Read*, además de sus terminales de comunicación y error, requiere de una terminal donde se indique el pin del Arduino que se desee “leer”. Como salida el bloque ofrece un dato de tipo booleano (TRUE/FALSE).

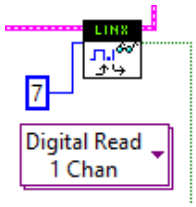


Figura 5.18 *Bloque Digital Read (Pin 7).*

Dentro del lenguaje G que se maneja en LabView, la lectura digital de los pines se hace de manera que el resultado de dicha lectura resulte en un valor booleano. Debido a que se ha planeado la utilización de la estructura de LabVIEW *Formula Node*, ha sido necesario convertir ese valor booleano en un número siendo, por convención, el cero para FALSE y el número uno para TRUE. Este cambio de tipo de variable se realizó con la función *Boolean To (0,1)*.

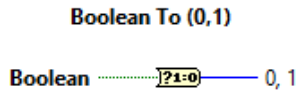


Figura 5.19 *Conversión de dato booleano a valor numérico.*

5.2.6.2 Conteo de pasos

Si bien en la interfaz de Arduino se pueden usar interrupciones, por motivos de facilidad, falta de existencia de dichas interrupciones en LabView y para mejorar el rendimiento en la lectura del programa, en lugar de usar interrupciones se usaron lo que en programación se conoce como “banderas” dentro de la función *Formula Node* de LabView. Esta estructura permite al programador escribir partes de código involucrando lenguaje C con el lenguaje G de LabView.

En sentido práctico las banderas de programación es una metodología en la cual, dentro de una estructura repetitiva, se posibilita o imposibilita la ejecución de una sección del código a partir de condiciones iniciales y variantes a lo largo de la ejecución (alto o bajo). Suponiendo, que existen dos procesos, A y B, en un mismo código y una variable X que toma los valores de 0 ó 1. El proceso A sólo se activa si X=0 y B sólo se activa si X=1. Al momento de arranque X es cero, entrando por default al proceso A. Si en algún punto del proceso A se hace cambiar el valor de X de 0 a 1, al ser una estructura repetitiva, en su siguiente iteración no se ejecutaría A pero se ejecutaría B puesto que X=1; si estando en B se hace cambiar el valor de X de 1 a 0, entonces en la siguiente iteración volvería a ejecutarse A ignorando el proceso B.

Para la sección correspondiente al conteo de pasos del encoder dentro del bloque *Formula Node* se siguió la lógica de jugar con las lecturas del sensor y las banderas de control. Si el sensor S1 entrega un valor de 1 significa que el encoder en esa posición deja pasar el halo de luz, si es 0 existe un bloqueo. Con el uso de banderas y por medio de la estructura *if-else* se realizó un contador que agrega (o sustrae en el caso de sentido negativo) una unidad a la variable *count*, encargada de contar el número de pasos en el sensor; inmediatamente la bandera de control que se encontraba levantada se vuelve a bajar de manera que se agregue un ±1 sólo vez cada que haya un cambio de alto a bajo o de bajo a alto. Este proceso se realiza ya que, debido a la velocidad de procesamiento de una computadora, se puede registrar más de una vez un valor positivo o negativo para una sola iteración y por tanto es necesario limitar la adición.

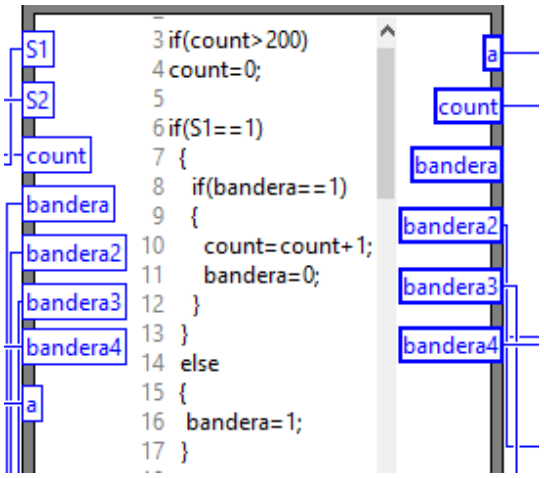


Figura 5.20 *Contador básico en estructura Formula Node.*

El conteo de los pasos del encoder de cuadratura es apreciable en el Panel Frontal de LabVIEW. A través de la variable count se muestra un indicador numérico que permite la observación de la cantidad de pasos que ha recorrido el encoder, estos pasos son sólo los cambios de alto a bajo o de bajo a alto que son registrados por el encoder.

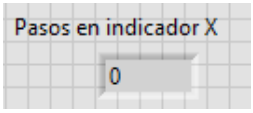


Figura 5.21 *Indicador del contador.*

Por otra parte se cuenta con un indicador en forma de perilla o aguja que muestra en qué punto de la revolución se encuentra girando el motor. Es apreciable un límite de 127 que refiere los 128 pasos del encoder (contando el cero). Y al igual que en el caso del indicador numérico, va ligado a la variable count de manera indirecta, se hace uso de una segunda variable *a_1* que limita el conteo. Con base en esta idea si el conteo de esta variable es mayor a 127, *a_1* toma nuevamente un valor de cero; si es menor a cero (suponiendo que el movimiento del motor es negativo), *a_1* tomará el valor de 127. Este procedimiento se realiza meramente para una apreciación de giro continuo en la aguja del indicador.

5.2.6.3 Control de sentido de giro

Una de los puntos a favor que tiene un encoder de cuadratura es que permite saber por medio de una programación el sentido en el cual está girando. No obstante se ha optado no usar esta estrategia en el proyecto presente debido a que no es completamente necesaria.

Si se analiza la funcionalidad del proyecto se puede identificar que el sentido de giro en cada motor es un dato que es proporcionado por la serie de algoritmos que definen el desplazamiento del posicionador y no es un dato que se desee obtener. Dicho de otra manera, a través de las elecciones fijadas por el usuario, el algoritmo general del programa definirá si el sentido es positivo o negativo y no es necesario un algoritmo para "saber" en qué sentido está girando el motor.

A pesar de que no se detecta como tal el sentido de giro del motor, sí existe una medida para definir dicho sentido. Se ha mencionado anteriormente que a diferencia de otros motores, un motor a pasos por sus características no es generalmente usado en un control de lazo cerrado. Otros motores hacen uso de variación de diferencia de potencial para controlar su velocidad y demás estrategias para controlar su par (torque). Un motor a pasos es controlado por pulsos definidos que incluso pueden ser contabilizados bastando únicamente con un control de lazo abierto. No obstante para este proyecto se ha decido utilizar lo más cercano a un control de lazo cerrado a modo de verificación.

El algoritmo que permite el giro de los motores, sin importar la etapa en la que se encuentre el posicionador, es relativamente simple. Considerando el hecho que el punto terminal del posicionador parte de un origen A (X_0, Y_0, Z_0) y se desea realizar un desplazamiento hasta las siguientes coordenadas B (X_1, Y_1, Z_1), se realiza una simple comparación dentro de la estructura *Formula Node* del diagrama de bloques. De este modo se puede definir que, mientras las coordenadas de A sean diferentes a las coordenadas de B, los motores habrán de moverse en el sentido que corresponda

indicado por el algoritmo en cuestión a través de las variables *sentido X*, *sentido Y* y *sentido Z*. El tipo de movimiento y diferentes algoritmos de selección y decisión serán explicados más adelante.

El bloque *Formula Node* permite todas las estructuras de lenguaje C, sin embargo en la IDE de Arduino se pueden realizar segundas lecturas dentro de un ciclo *while* que permiten la exclusión del bucle en determinado momento. En LabVIEW no es posible realizar dobles lecturas, entonces sólo se trabaja con el dato de entrada, por ello si se usara un ciclo *while* en LabVIEW, se repetiría de manera infinita pues no existe retroalimentación y se inutilizaría el programa. Así pues ha sido necesario sólo trabajar con ciclos *if* y varias banderas buscando que el resultado de ejecución fuera idéntico a un ciclo *while*.

5.2.7 Lectura y despliegue de sensores de acotación al origen

Así como existen tres sensores para el monitoreo del giro de los motores a pasos, también existen sensores para definir el origen del sistema. Se han utilizado tres *switch bumper* para actuar como sensores de acotación. Debido a que estos sensores están normalmente abiertos, al momento de ser presionados es cuando se presenta el cambio de señal.

Figura 5.22 *Switch bumper.*

En el diagrama de bloques estos *switch bumper* reciben el valor de una variable booleana (0 o 1) y su forma en el Panel Frontal es por medio de los indicadores booleanos tipo LED “Llegada a 0 X” para el eje X, “Llegada a 0 Y” para el eje Y, “Llegada a 0 Z” para el eje Z. De este modo cuando uno de los indicadores se ilumina significa que en ese eje el posicionador se ha movido hasta el punto donde se presiona el sensor *switch bumper* de ese eje.

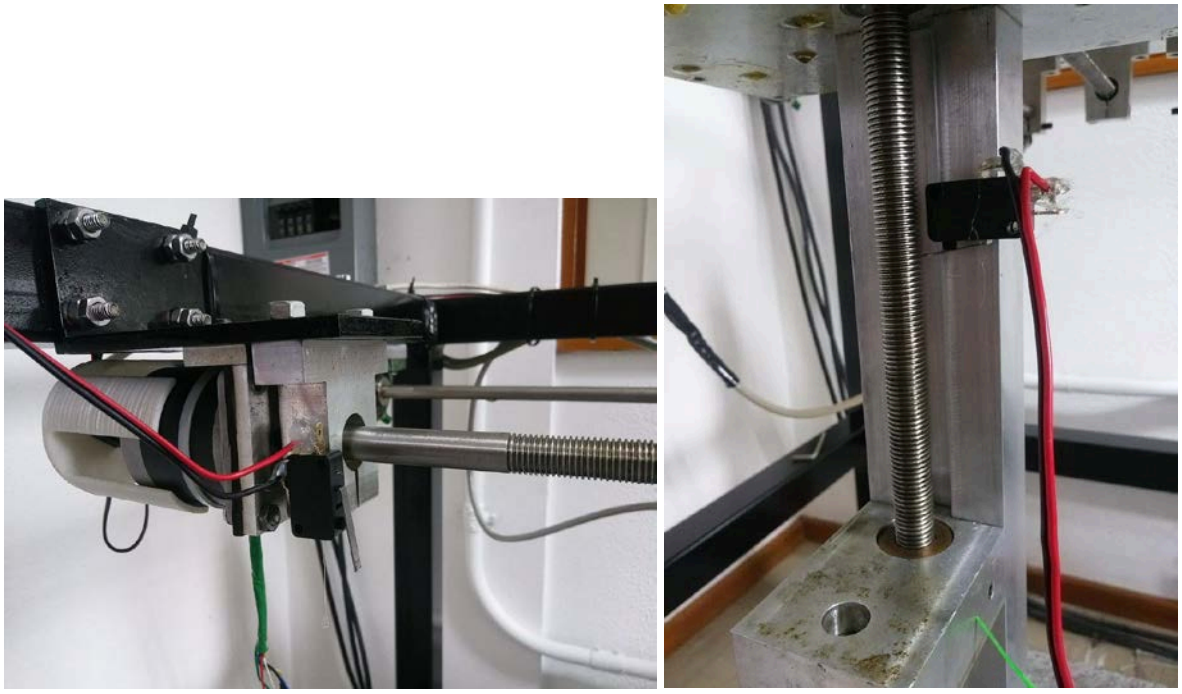


Figura 5.23 *Acotación de ejes con switch bumper.*

5.3 Primer programa: Ubicación en el segundo origen y retorno

Como se ha comentado previamente el procedimiento total que realiza el posicionador requiere de tres partes: la ubicación en un segundo origen, el desplazamiento de barrido y el retorno al origen absoluto. Sin embargo por condiciones de procesamiento del computador fue necesario dividir este procedimiento en dos partes. Existe un primer programa que desplaza el actuador del posicionador de forma positiva en un segundo origen; este programa también tiene una función de retorno. El segundo programa consiste en el desplazamiento de barrido, captura de datos y almacenamiento.

Para el primer programa, en su modalidad de ubicación en segundo origen, se ha considerado que el actuador del posicionador se mueve desde una esquina de la estructura cúbica que lo sostiene (un origen absoluto) hasta llegar a las coordenadas que se solicitan. En la modalidad de retorno se considera que el actuador se encuentra en un punto distinto al origen absoluto, así sean unas pocas micras hasta varios centímetros, y su única función es regresar el actuador a su posición original.

5.3.1 Principio básico de funcionamiento

El primer programa contiene una interfaz en la cual se puede elegir el tipo de modalidad que se desea a través de dos *Radio buttons* con los nombres “IDA” y “REGRESO”. Dependiendo de la opción que se elija, se ejecutará el algoritmo pertinente. En el caso de que la opción sea “IDA”, las casillas de control numérico se activarán permitiendo la

escritura de las coordenadas donde se desee fijar el segundo origen, en caso de que la opción sea “REGRESO” estas casillas serán deshabilitadas.

Una vez elegido el tipo de desplazamiento, si este es “IDA” se solicitará un avance en cuando menos una variable, en el caso de “REGRESO” no se exige ningún dato. Enseguida se habrá de oprimir el botón “COMENZAR” que permitirá la ejecución del algoritmo seleccionado. Una vez que haya finalizado dicho desplazamiento el programa mandará una alerta de término y en seguida terminará su ejecución de manera automática.

5.3.3 Selección de variables de desplazamiento

En el diagrama de bloques se puede observar la variable referida al par de *Radio butons*. La salida de esta variable lleva a una estructura *switch case*, que se encarga de definir según la opción seleccionada, el algoritmo a utilizar. Con el programa en funcionamiento es necesario introducir los datos de coordenadas de segundo origen para el tipo de movimiento “IDA”, en el caso de la modalidad de “REGRESO” no se requiere introducir dato alguno. Cabe mencionar que ninguno de los dos programas entrará en funcionamiento hasta que se pulse el botón “COMENZAR” del Panel Frontal.

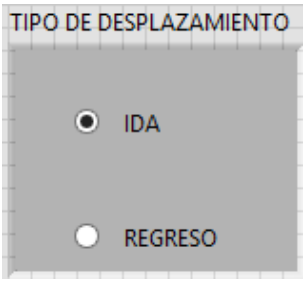


Figura 5.24 *Selección de movimiento.*

Dentro de cada caso del par de *Radio buttons* se activan o desactivan dos indicadores del Panel Frontal, uno para IDA y otra para REGRESO, dependiendo de qué tipo de movimiento sea, se habrá de iluminar dicho indicador según el caso. Además a través de las propiedades de control de variables que maneja LabVIEW se puede habilitar y deshabilitar una variable; en el Panel Frontal cuando se trabaja con el regreso, se desactivan las variables de introducción de coordenadas. Este fenómeno es posible ya que, dentro del diagrama de bloques, las terminales de propiedad de variable “*Disabled*” para las variables *cmX, mmX, cmY, mmY, cmZ* y *mmZ*, que son donde se ha de escribir las coordenadas del segundo origen, son desactivadas por un control “*Disabled and grayed out*” que inhabilita dicha variable y además la vuelve más tenue visualmente, indicando que su uso en ese momento no está permitido. El control “*Disabled and grayed out*” es colocado dentro del caso de “REGRESO”, para el caso de “IDA” este control es cambiado por otro control llamado “Enabled” el cual permite el uso y visualización de las variables.

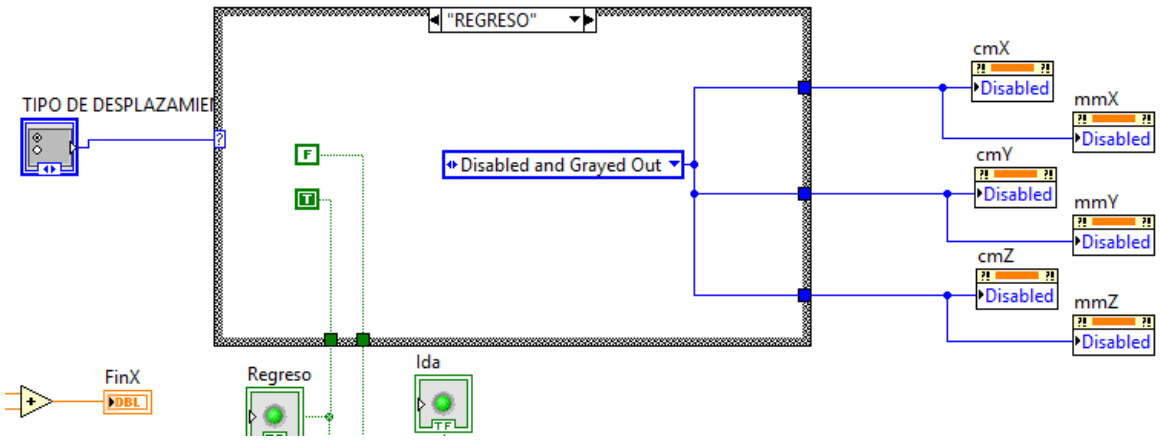


Figura 5.25 *Deshabilitación de variables de movimiento.*

5.3.3 Algoritmo de desplazamiento al segundo origen

Se ha dicho que cuando se trabaja con el tipo de movimiento “IDA” las variables de coordenadas del segundo origen son activadas. Estas variables son *cmX, mmX, cmY, mmY, cmZ* y *mmZ* para los ejes X, Y y Z respectivamente. En el Panel Frontal se puede observar que se solicitan los centímetros y los milímetros a los cuales se desea desplazar el posicionador, las variables *FinX, FinY* y *FinZ* son la suma de estos centímetros y milímetros para cada eje.

Se considera que cada paso del encoder representa un avance en micras por medio del motor a pasos, por tal motivo se ha tenido que multiplicar las variables “centímetros” y “milímetros” por una constante diferente cada una de modo que, al multiplicar estas variables por su respectiva constante resulte en un total de pasos que, al ser sumados, den la totalidad de pasos que el encoder necesita para recorrer la distancia deseada.

El funcionamiento principal recae en una serie de condiciones en las cuales, dependiendo de si se cumplen o no el motor se moverá en sentido dextrógiro. Al colocar un valor positivo en al menos una de las coordenadas del segundo origen y presionar el botón “COMENZAR”, el motor avanzará y con él el conteo de pasos del encoder para ese eje. Mientras el conteo de pasos del encoder sea diferente al total de la suma de pasos dados por las variables “centímetros” y “milímetros” de ese eje, el motor avanzará en sentido positivo. De no escribir un número en alguna casilla de las coordenadas de segundo origen, el valor de “centímetros” o “milímetros” para ese eje será cero.

Figura 5.26 *Botones de inicio y pausa.*

Cuando el conteo de pasos haya llegado a una igualdad con la coordenada, ese motor se detendrá, en el caso de que el motor rebase el total de pasos deseados, el motor girará en sentido negativo el número de pasos que haya rebasado. Este proceso se repite para cada eje del posicionador y cada uno trabaja de manera independiente, es decir, si sólo se desea avanzar en dos coordenadas, estas funcionarán de manera independiente entre sí y el movimiento para el tercer eje jamás habrá de activarse. En el momento en que se logre una igualdad de pasos deseados con los medidos en cada uno de los ejes se desplegará un mensaje indicando que se ha llegado a la posición deseada y el programa se terminará automáticamente.

Cabe mencionar que el algoritmo del movimiento “REGRESO” hace volver al posicionador a su punto de origen absoluto, esto es, al presionar los *switch bumper* que acotan el origen. Sin embargo en función de que se trata de un tipo de botón que presenta un cierto sesgo cuando se presiona, es decir, existe una pequeña distancia entre su punto muerto superior y el punto donde se completa el circuito y cambia el valor de la señal, se ha optado por que el posicionador no empiece a contar pasos del encoder hasta que el circuito de *switch bumper* se encuentre abierto, es decir, el sensor no indique que el actuador se encuentra en su punto de acotamiento de origen para ese eje. Este proceso se realiza para cada uno de los tres ejes.

5.3.4 Algoritmo de retorno

Para el caso de movimiento “REGRESO” se ha supuesto que el posicionador se ha movido en al menos uno de sus ejes fuera del punto de acotamiento de origen u origen absoluto. Dado este hecho al momento de seleccionar esta opción de movimiento, el algoritmo de retorno hará que el posicionador vuelva a su posición original.

El funcionamiento del algoritmo de “REGRESO” recae en que cada motor del posicionador se moverá en sentido negativo si el switch bumper de ese eje se encuentra abierto. Dicho de otra manera, si el sensor que se encarga definir el acotamiento de origen no se encuentra presionado, el motor habrá de moverse hasta que lo esté. Una vez presionado, se mostrará un LED encendido en el Panel Frontal indicando que se ha llegado al origen en ese eje.

Debido a que el algoritmo de “REGRESO” sólo exige el retorno al origen absoluto, no ha sido necesario agregar un conteo de pasos del encoder puesto que no interesa en qué posición se encuentra el punto terminal, sólo importa si ha llegado o no a su origen. El programa terminará de forma automática en el momento en que los tres sensores se encuentren presionados, cuando esto suceda, se desplegará un mensaje indicando que se ha llegado al origen absoluto.

5.4 Segundo programa: Desplazamiento lineal, superficial y volumétrico

Si el primer programa se encarga del posicionamiento en un segundo origen y del retorno, el segundo programa se encarga de todo lo demás. Las acciones comprendidas dentro de este programa son el desplazamiento de barrido, según sea el caso deseado, la lectura del osciloscopio y el guardado de los datos registrados.

Para el segundo programa se considera que el actuador del posicionador ya ha sido movido hasta las coordenadas donde se desea que comience su movimiento de barrido. El algoritmo de este programa realizará tres tipos de movimiento: un desplazamiento lineal sobre alguno de los ejes X, Y o Z, un desplazamiento en forma de plano sobre los planos XY, XZ o YZ, o bien un desplazamiento en un volumen. Dependiendo de las opciones dadas por el usuario, se hará uso del algoritmo pertinente.

5.4.1 Principio básico de funcionamiento

Para este programa se utilizó un único código para los distintos tipos de movimiento; en este caso fue el código de desplazamiento volumétrico. A pesar de que se usa el desplazamiento volumétrico, la forma en que se introducen las variables define cómo se habrá de mover aún si se trata de una línea recta.

El usuario debe introducir los valores de la distancia que desee cubrir, también debe introducir qué tan seguido desea realizar una medición. Por medio de las variables de centímetros y milímetros facilitadas en la interfaz, al igual que con el primer programa, se definirá la distancia a desplazarse. Por el lado de las mediciones, estas pueden ser realizadas en micrómetros puesto que cada paso del encoder avanza una muy pequeña distancia. La forma en la cual se desplaza el posicionador es en zigzag.

El programa tiene una función de lectura de osciloscopio de un máximo de cuatro canales. Por medio de programación en LabVIEW se puede conocer los datos de las propiedades de la onda que se registre. Por el lado del almacenamiento de información se realiza una lectura cada vez que el encoder contabilice los pasos de separación que coinciden con los deseados por el usuario. Cuando se haya recorrido la totalidad de distancia en cada eje de movimiento, el programa enviará una alerta de proceso realizado y automáticamente se terminará.

5.4.2 Selección de tipo de desplazamiento

Al igual que ocurre con el primer programa, en el segundo se encuentran en el Panel Frontal varias cajas que contienen *Radio buttons* que se expresan en el diagrama de bloques como una selección ligada a una estructura *Switch case*. El primer grupo de *Radio buttons* es el que define qué tipo de desplazamiento se desea realizar: Línea recta, Superficie o Volumen.

El segundo grupo de Radio buttons depende de la selección que se haya tomado con el primer grupo. En el caso de que se eligiera “Línea recta” el programa dirigirá sus opciones a las que van ligadas a un movimiento sobre un eje: Eje X, Eje Y o Eje Z. En el caso que se hubiera elegido la opción “Superficie” en el primer grupo de *Radio buttons*, se dirigirá a las opciones ligadas con un plano: Plano XY, Plano XZ o Plano YZ. Si se elige en el primer grupo de *Radio buttons* la opción “Volumen” ninguna segunda selección habrá de hacerse puesto que se ocupará la totalidad de sus elementos de escritura.

Al igual que sucedía con el primer programa, las opciones de selección escritura se habilitarán o deshabilitaran dependiendo de qué opción se había elegido previamente. Por ejemplo, si se elige la opción de Línea recta, los *Radio buttons* de selección de plano se deshabilitarán así como dos opciones de escritura. Si dentro del menú de Línea recta se elige el Eje Y, las opciones de escritura para Eje X y Eje Z se deshabilitarán igualmente.

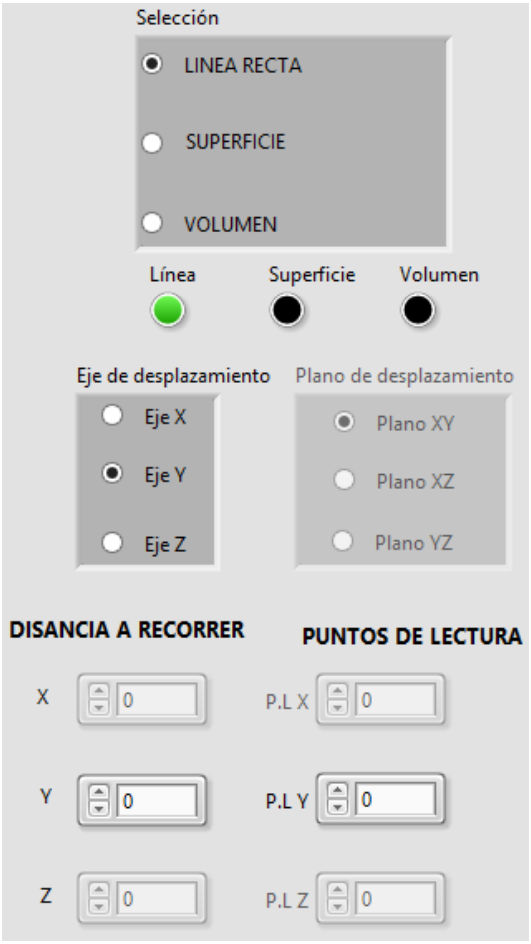


Figura 5.27 *Selección de tipo de desplazamiento.*

Por medio de las variables de distancia asignadas por el usuario se define qué tanto debe avanzar en cada uno de los ejes. Se muestran en el programa las variables de centímetros y milímetros para cada eje. Dentro del diagrama de bloques estas variables

son un control numérico de tipo double. Las opciones que no se desean utilizar se deshabilitan a partir de la propiedad de variable *Disabled an Grayed out*, según sea la combinación que se haya introducido se habrán de habilitar y deshabilitar diferentes opciones y controles según sea el caso. Existe un total de siete combinaciones, tres para línea recta, tres para plano y una para volumen; la variable de nombre *control* es la encargada de indicarle al programa cómo se moverá.

5.4.3 Algoritmo de desplazamiento

Al igual que en el primer programa, el desplazamiento se hace a partir de una serie de condiciones. La diferencia con el primer programa es que, en el primero todos los motores funcionan simultáneamente, en el segundo programa dos motores no pueden funcionar al mismo tiempo y además estos se mueven bajo un orden específico. No obstante la consigna elemental sigue siendo la misma: Mientras las coordenadas de un punto inicial A(X_0, Y_0, Z_0) sean diferentes a las coordenadas de un punto B(X_1, Y_1, Z_1) el motor girará en el sentido indicado hasta alcanzar la igualdad.

El segundo programa debe contar un número *n* de pasos a recorrer para cada eje. Este número de pasos deseados está dado por la suma de las variables de centímetros y milímetros para cada eje. Las variables *VAR1*, *VAR2* y *VAR3* son las sumas de los centímetros y milímetros para los ejes X, para Y y para Z respectivamente.

Para efectos explicativos se le asignará el nombre de “cota inferior” al punto donde inicia el desplazamiento en un eje y “cota superior” al punto máximo de distancia que se mueve el posicionador en ese eje, definido por el usuario. Dicho de otra manera, si el usuario desea recorrer una distancia de 10 centímetros en el eje X y 12.5 centímetros en el eje Y, la cota inferior en X será 0 y la cota superior en X será 10 centímetros, la cota inferior en Y sería 0 y la cota superior de Y sería 12.5.

Se ha dicho que se ocupa el algoritmo de movimiento de volumen para todos los otros tipos de movimiento, la diferencia es cómo se introducen las variables iniciales. Para explicar el algoritmo será necesario definir que los ejes X, Y y Z son los ejes físicos del posicionador y los ejes A, B y C son una representación de una estructura cúbica o de una matriz de tres dimensiones que es la estructura base sobre la que se fundamenta el algoritmo de desplazamiento. Este algoritmo supone que se inicia el movimiento en un eje A hasta llegar a su cota superior o inferior donde inicia el movimiento de un eje B, cuando el eje B llega a su cota inferior o superior se inicia el movimiento del eje C. Este razonamiento supone una matriz como base en la cual se toma como prioridad de movimiento al eje A, en segundo lugar al eje B y en tercero al eje C. Teniendo en cuenta esto se puede jugar con los datos de entrada.

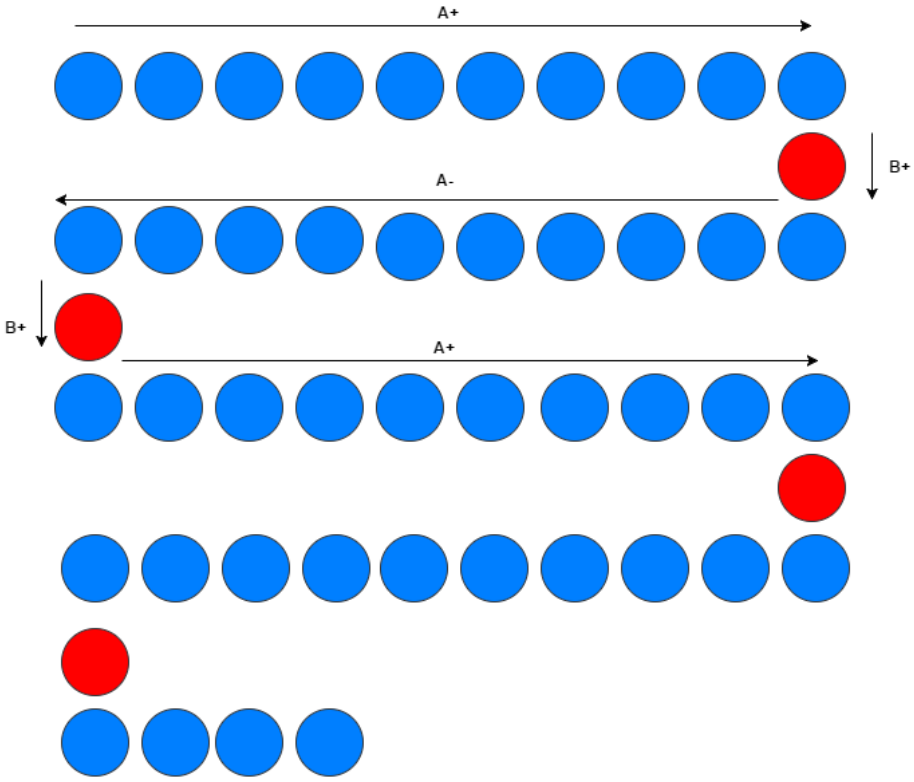


Figura 5.28 *Avance sobre ejes B y C.*

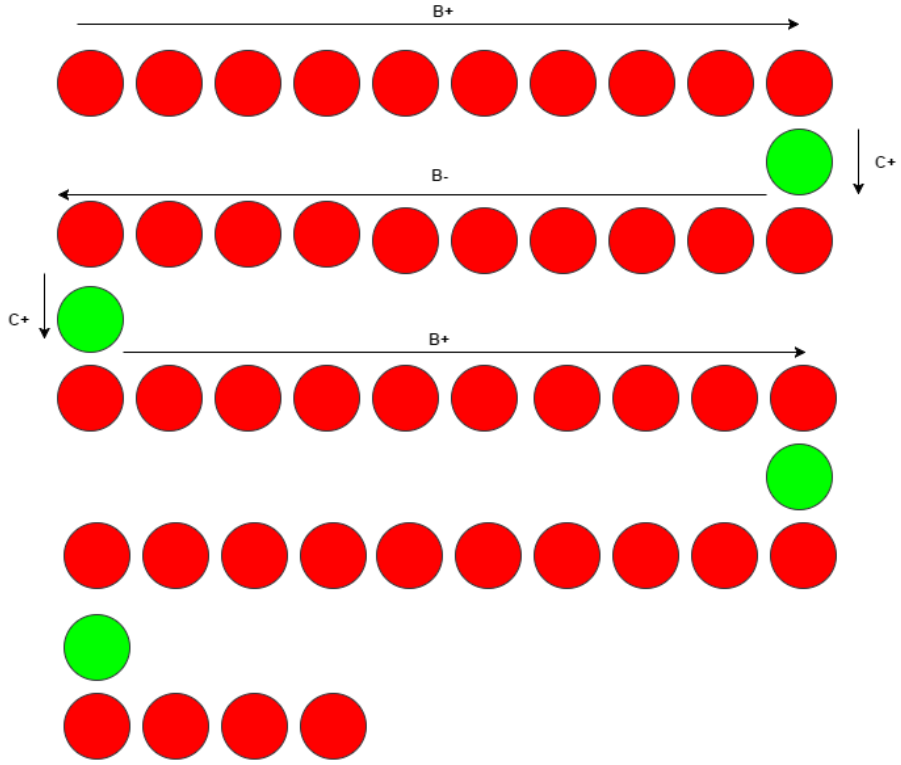


Figura 5.29 *Avance sobre ejes A y B.*

Como se ha dicho con anterioridad, el posicionador se mueve en zigzag. Para fines prácticos se supondrá que el usuario ha elegido la opción de un desplazamiento volumétrico. Para este caso el movimiento comenzará con el motor de X (eje A para el algoritmo) girando de forma positiva yendo desde su cota inferior hasta llegar a su cota superior; una vez alcanzado este punto se desactivará el movimiento en X y se activará el motor de Y (eje B para el algoritmo) de forma positiva, cuando se detecte un pulso del encoder de Y, se desactivará el movimiento de Y, y se reactivará el movimiento del motor de X pero ahora de forma negativa hasta que llegue a su cota inferior, donde se volverá a desactivar X y se reactivará Y. Este proceso se repetirá hasta que se llegue a la cota superior de Y donde habrá de desactivarse su movimiento y Z (eje C para el algoritmo) comenzará a moverse hasta registrar un pulso; el paso siguiente será que el movimiento en X se active de manera normal pero ahora en cada cota de X, superior o inferior, el movimiento de Y será negativo. El procedimiento se repetirá hasta que se llegue a la cota superior de Z donde el programa terminará.

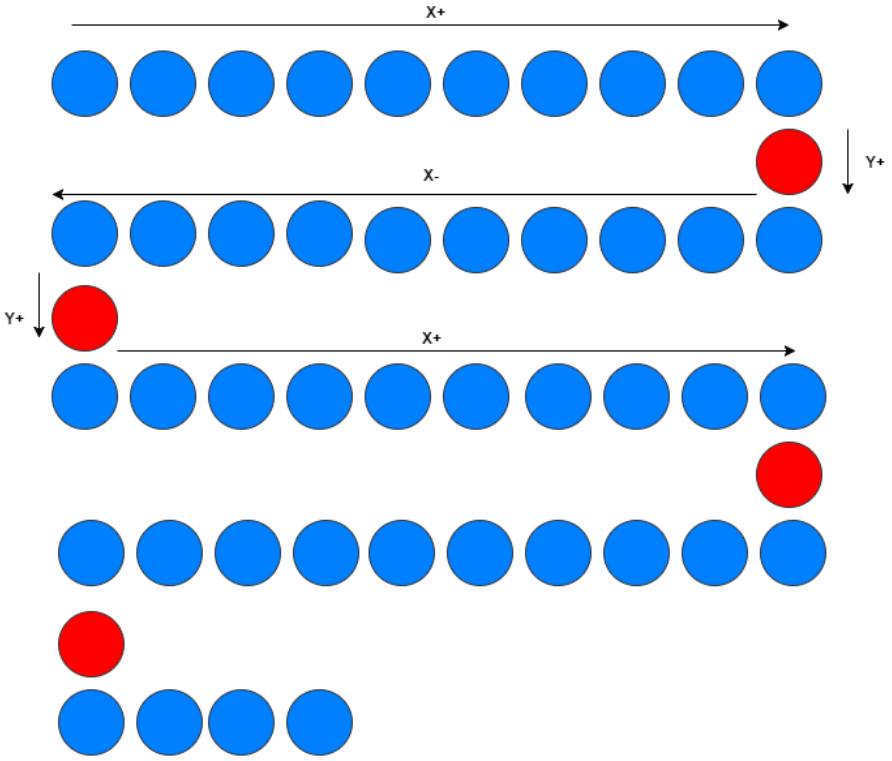


Figura 5.30 *Ejemplo de avance sobre X y Y basado en ejes A y B de la matriz.*

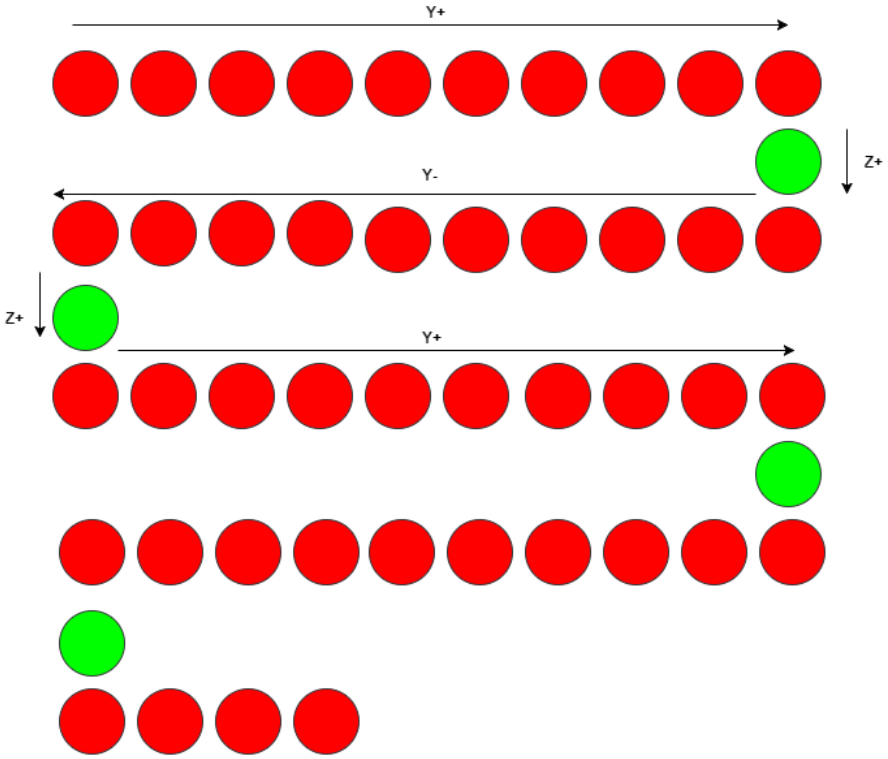


Figura 5.31 *Ejemplo de avance sobre Y y Z basado en ejes B y C de la matriz.*

Existe una sección del código en la que una estructura *Switch case* es manipulada a través de la variable *control*. Dentro de cada caso del *Switch case* se define a qué ejes se les debe asignar un número de pulsos a leer para que funcione como línea, plano o volumen. Para el caso de una línea (*control* = 1, 2, ó 3), sólo debe existir una cantidad de pulsos a leer en sólo uno de los ejes. Si se elige por ejemplo el eje Y, los pulsos a leer en X serán cero, en Y serán los solicitados por el usuario (variable *VAR2*) y en Z será igualmente cero.

A pesar de la obviedad de los datos solicitados por cada tipo de movimiento, es la forma en la que son introducidos al código lo que define el movimiento. Para el caso de línea recta, si se desea moverse sobre el eje X, es necesario introducir en el código para el eje A el valor de la variable *VAR1*, en el eje B cero pulsos y en el eje C cero también. Si se desea mover sobre el eje Z, se debe introducir al código el valor de la variable *VAR3* en el eje A, cero en B y cero en C. Se entiende que es el mismo razonamiento para ambos casos, en los dos se utiliza como entrada al código el eje A. Sin embargo la variable *control* no sólo define la forma en que se introducen los datos, sino también la manera en que salen.

Para el caso de la salida, la variable *control* define qué motor se utilizará y en qué sentido. A través de otra estructura *Switch case* de siete casos se define qué motores se utilizan. Por ejemplo, si se desea un movimiento en el plano XZ (*control* = 5), en la entrada de datos habrán de introducirse los valores de *VAR1* para el eje A y *VAR3* para el eje B y cero para el eje C; por el lado de la salida para el eje A se activará el motor X y para el eje

B se activará el motor Z, para el eje C se moverá el motor de Y, cuando el eje B llegue a su cota superior, sólo un pulso para que se detecte un avance y complete la estructura de matriz para dar por terminado el programa.

5.5 Lectura de señales del osciloscopio

Un osciloscopio es un instrumento de medición para captura de señales variantes en el tiempo. Para este proyecto se ha usado el osciloscopio Tektronix modelo DPO 3014. Este es un modelo de osciloscopio electrónico que permite el muestreo de cuatro canales. Al igual que un osciloscopio convencional, el DPO 3014 cuenta con comandos para el desplazamiento de la señal, escalamiento y selección de canal. La ventaja que presenta Tectronix con sus osciloscopios electrónicos es la facultad de guardar imágenes, datos de muestreo y en particular para este proyecto, la comunicación vía serial con una computadora a través de un puerto COM.

Para hacer uso del osciloscopio ha sido necesaria la instalación de los drivers necesarios para su puesta en marcha. Igualmente ha sido necesaria una instalación de drivers para su comunicación con LabVIEW. Al igual que con las tarjetas de Arduino, estos drivers de LabVIEW permiten la apertura y cierre de comunicación, recepción de datos y análisis de información.

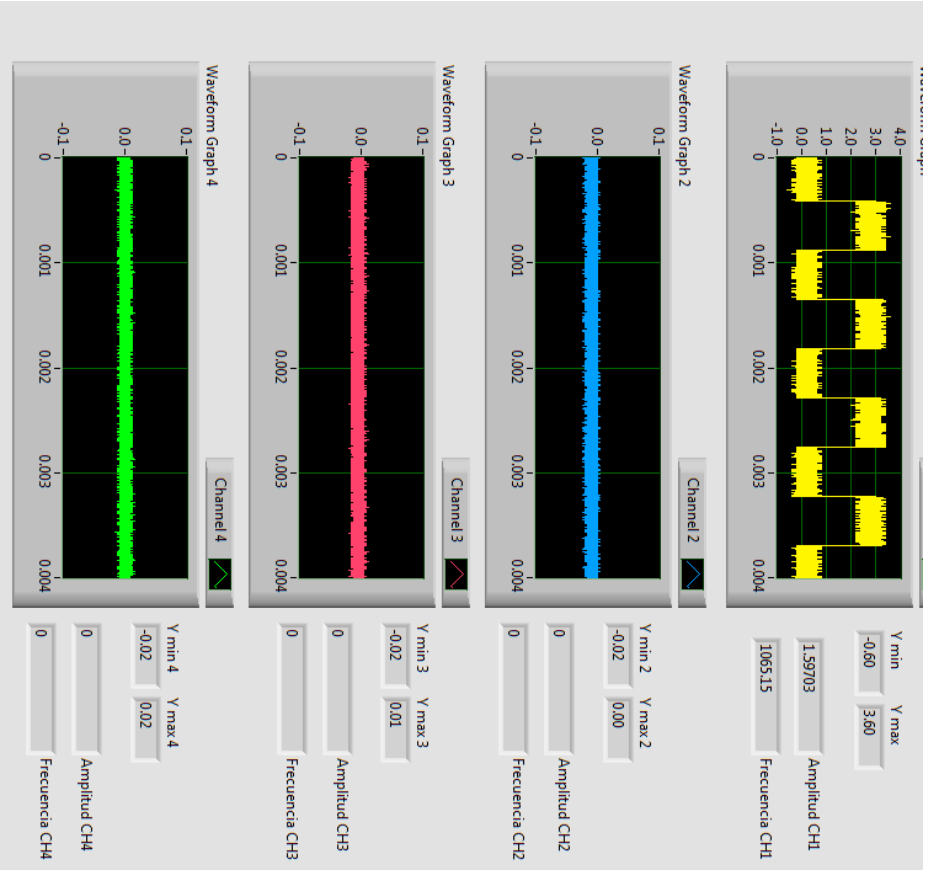


Figura 5.32 *Muestreo virtual de señales de osciloscopio.*

En el uso del osciloscopio Tektronix DPO 3014 es necesario primero establecer el canal de comunicación a través de los bloques *Initialize* y *Close* dentro del diagrama de bloques de LABVIEW en el menú de Tektronix. Para el bloque *Initialize* es necesario asignar un puerto COM a través del control *VISA Resource Name*, un puerto de la computadora estará designado a la comunicación exclusiva del osciloscopio. Es importante aclarar que si no se inicializa y cierra el canal de comunicación no será posible compilar el programa. Cabe resaltar que, debido a que se trabaja con cuatro señales, los bloques necesarios para mostrar y analizar cada lectura se escribirán cuatro veces.

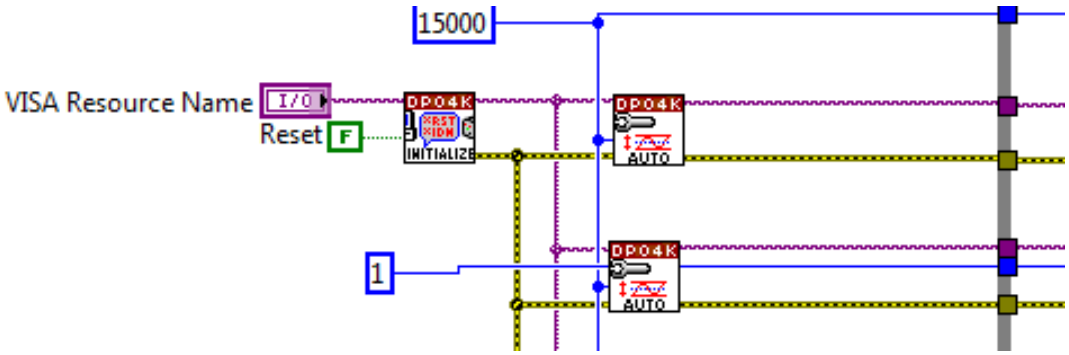


Figura 5.33 *Inicialización de osciloscopio en el diagrama de bloques.*

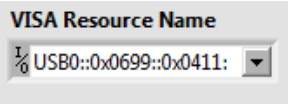


Figura 5.34 *VISA Resource Name en Panel Forntal.*

Para la lectura de la señal se ha utilizado el bloque *Auto Setup*. Este bloque permite el escalamiento y lectura de un canal de manera automática al iniciarse el programa. Debido a que se desea tener la imagen de la señal de manera independiente, se ha ocupado el bloque *Single Waveform* que permite a LabVIEW obtener la información de la señal en tiempo real de un solo canal. El bloque *Waveform Graph* muestra en el Panel Frontal la imagen de la gráfica con un autoescalamiento en su amplitud y frecuencia.

El software LabVIEW cuenta con una serie de bloques destinados al análisis y medición de las señales. A través del bloque *Tone Measurements* es posible determinar ciertas propiedades de la onda que se analice. Para este proyecto sólo se ha solicitado la medición de la frecuencia y la amplitud. Como medida extra se ha anexado a este programa los valores de las crestas positivas y negativas de las ondas en los cuatro canales.

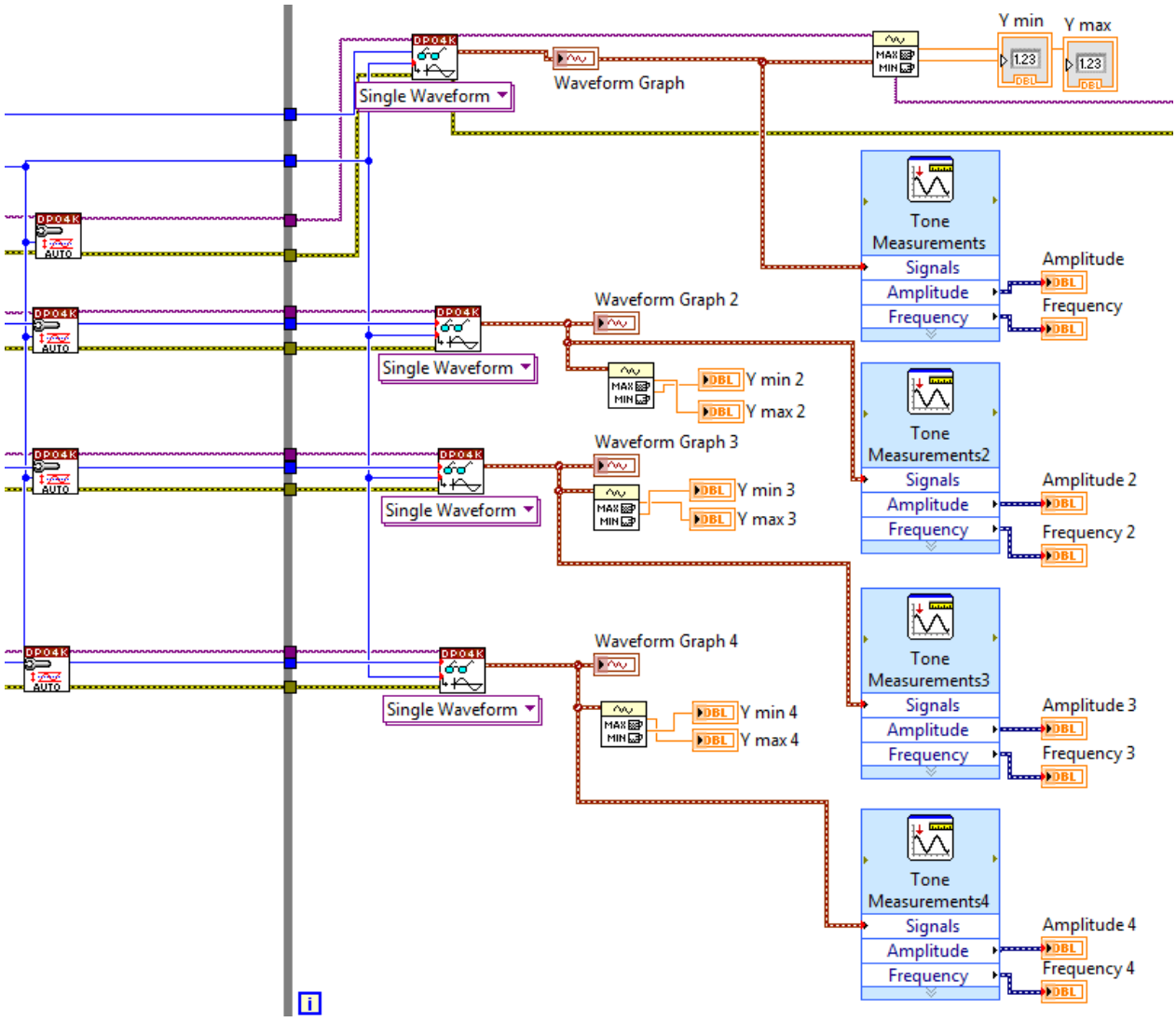


Figura 5.35 *Lectura y análisis de señal en el Diagrama de Bloques.*

Las pruebas realizadas para determinar el buen funcionamiento del programa de LabVIEW con el osciloscopio Tektronix DPO 3014 fueron hechas con ayuda de un generador de señales. A pesar de que la medición obtenida en LabVIEW es idéntica a las recibidas por el osciloscopio, existe la presencia de ruido y estática en las terminales de los cables BNC cuyo canal no está en uso. Como lecturas se obtienen amplitudes bajas no mayores a 0.0005 voltios y las frecuencias muy altas sobrepasando los 20000 Hertz. A través de la programación se han separado estas lecturas del resto para evitar que existan datos no deseados en el archivo de texto generado. Para solucionar el problema se optó por eliminar las mediciones cuya amplitud fuera menor a 0.001 voltios; en el caso de que una amplitud tenga un valor menor, el programa convertirá el dato tanto de amplitud como de frecuencia en cero, ese número será el mostrado y registrado en la base de datos.

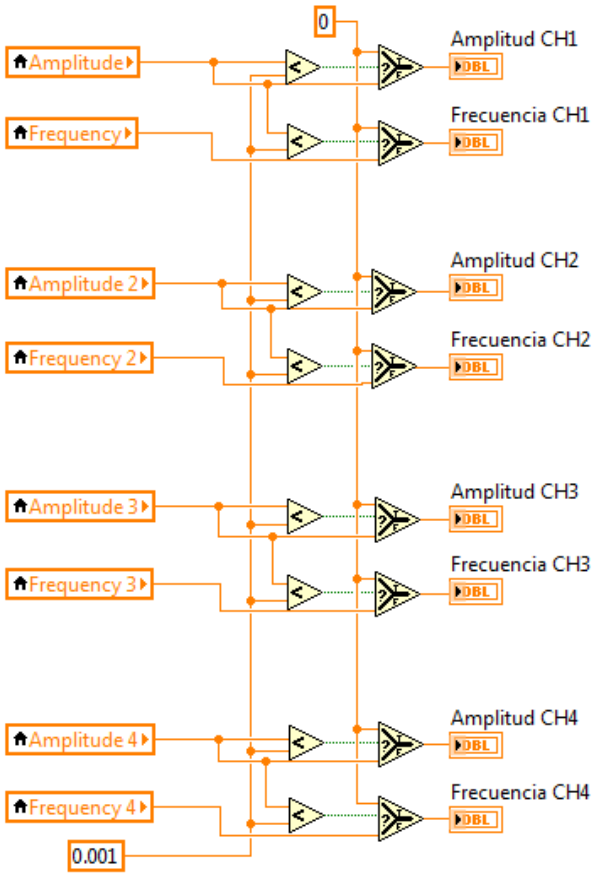


Figura 5.36 *Discriminación de señales no deseadas.*

5.6 Almacenamiento de información de lecturas

Las lecturas registradas por el osciloscopio se almacenan en un archivo de texto de manera organizada. Dentro del archivo están contenidos los datos de número de lectura, fecha y hora de lectura, coordenadas donde fue tomada la lectura, el canal del osciloscopio, amplitud y frecuencia de la onda. Se debe tomar en cuenta que existen cuatro canales conectados por lo que habrá un total de 18 datos por línea dentro del archivo de texto.

El algoritmo de desplazamiento controla de manera indirecta la toma de lecturas utilizando como referencia las variables de punto de lectura en X, Y y Z (*PLX*, *PLY* y *PLZ* respetivamente). Cada vez que el módulo de las lecturas del encoder y el valor de las variables de punto de control sea igual a cero, la sección del programa destinada al desplazamiento se detendrá por un total de 1500 milisegundos. Al detenerse también se habrá de disparar la señal para poner en marcha el algoritmo de almacenamiento; cuando acabe este tiempo los motores se volverán a poner en marcha. Supongamos que se elige un desplazamiento en el plano XY, y un valor de Puntos de lectura PLX=5 y PLY=4 (PLZ al estar desactivado tendrá un valor de cero y todo módulo de un número *n* con cero es cero). Cada vez que el valor de lectura del encoder del eje X sea igual a PLX o uno de sus múltiplos, y además al mismo tiempo, el valor de lectura del encoder del eje Y sea igual a

PLY o uno de sus múltiplos, se activará la variable *detención* que habilita el algoritmo de almacenamiento.

En el Panel Frontal existen ciertos elementos referidos al almacenamiento de las lecturas. El primero es un cuadro de diálogo para elegir el destino donde será creado y guardado el archivo dentro del explorador de Windows, en este cuadro se debe especificar la ruta y el nombre del archivo así como su extensión. El segundo elemento apreciable es una tabla que muestra de manera ordenada los datos de las mediciones conforme estas son tomadas. Finalmente se aprecia un indicador LED que señaliza la detención de los motores y el almacenamiento de una lectura.

Figura 5.37 *Selección de ruta en el explorador de Windows.*

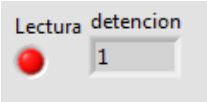


Figura 5.38 *Indicadores de detención y lectura.*

Dentro del diagrama de bloques, el registro de datos se hizo en dos etapas: la generación de una tabla con cadenas de caracteres y el registro de esta tabla mediante la escritura en un archivo. Para la generación de la estructura de una tabla se ha hecho uso del bloque *Build Array* para mostrar en el Panel Frontal y *Concatenate Strings* para el archivo de texto; estos bloques permiten crear un arreglo de cadenas de caracteres de *n* elementos. El único problema que presenta este bloque es que es necesario que todos sus elementos sean cadenas de caracteres, por ello ha sido necesario transformar cada dato numérico que se desee escribir en la tabla a un valor tipo string. Esta transformación se ha realizado por medio del bloque *Format Into String.*

Para darle formato al archivo de texto se han utilizado los elementos de escritura *Tab constant* y *Carriage Return Constant* como entradas del bloque *Concatente Strings*. El primero genera un espacio de tabulación entre la cadena anterior y la posterior, el segundo se encarga de generar un salto de línea y volver al primer elemento de la línea siguiente. De esta manera se obtiene un archivo de texto con los datos separados en columnas.

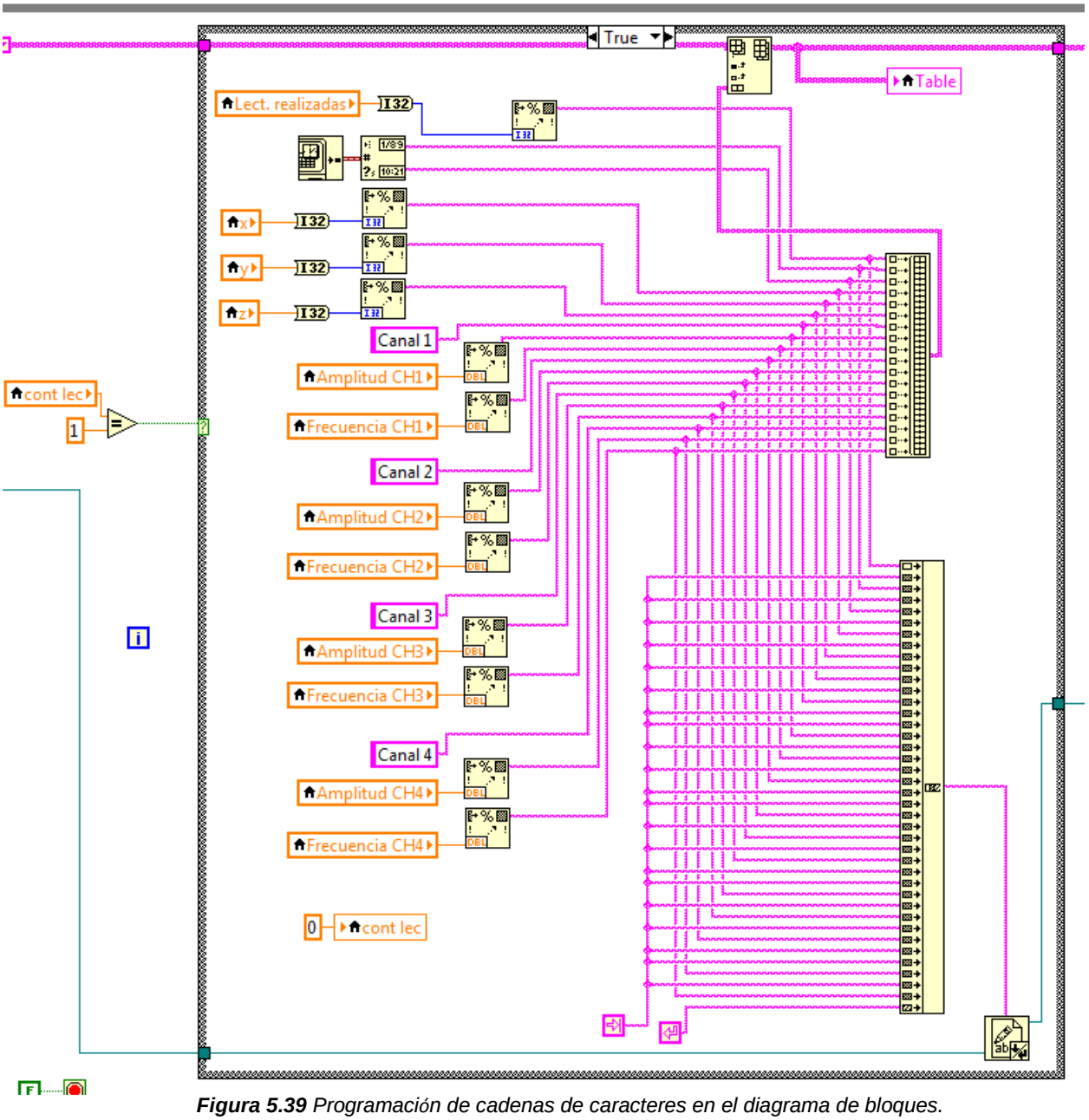


Figura 5.39 *Programación de cadenas de caracteres en el diagrama de bloques.*

Si bien se ha generado una tabla a través de los mecanismos mencionados, es necesario crear un archivo de texto con dicho arreglo de cadenas de caracteres. Al igual que pasa con los bloques de Arduino y el osciloscopio, la escritura de un archivo requiere de una inicialización (y creación) y de un cierre. Mediante el bloque *Set File Position* se asigna la posición donde comenzará a escribirse el archivo, para este caso el archivo comenzará a escribirse al final del último carácter registrado. Con el bloque *Write To Text File* se permite la escritura como tal de la cadena dentro del archivo creado en la posición del marcador indicado. Debido a que esta es una estructura cíclica (*For Loop*) y además cuenta con un *Shift Register*, en cada iteración se escribirá una nueva línea dentro del

mismo archivo de texto resultando en una serie de cadenas ordenadas dentro de un solo archivo.

3	11/01/2017	12:59 p.m.	10	0	0	Canal 1	1.603157	1066.463167	Canal 2	0.0
4	11/01/2017	12:59 p.m.	15	0	0	Canal 1	1.603157	1066.463167	Canal 2	0.0
5	11/01/2017	12:59 p.m.	20	0	0	Canal 1	1.603157	1066.463167	Canal 2	0.0
6	11/01/2017	12:59 p.m.	25	0	0	Canal 1	1.603157	1066.463167	Canal 2	0.0
7	11/01/2017	12:59 p.m.	30	0	0	Canal 1	1.603157	1066.463167	Canal 2	0.0
8	11/01/2017	12:59 p.m.	35	0	0	Canal 1	1.603157	1066.463167	Canal 2	0.0
9	11/01/2017	12:59 p.m.	40	0	0	Canal 1	1.603157	1066.463167	Canal 2	0.0
10	11/01/2017	12:59 p.m.	45	0	0	Canal 1	1.603157	1066.463167	Canal 2	0.0
11	11/01/2017	12:59 p.m.	50	0	0	Canal 1	1.603157	1066.463167	Canal 2	0.0
12	11/01/2017	12:59 p.m.	55	0	0	Canal 1	1.603157	1066.463167	Canal 2	0.0
13	11/01/2017	12:59 p.m.	60	0	0	Canal 1	1.603157	1066.463167	Canal 2	0.0
14	11/01/2017	12:59 p.m.	65	0	0	Canal 1	1.603157	1066.463167	Canal 2	0.0
15	11/01/2017	12:59 p.m.	70	0	0	Canal 1	1.603157	1066.463167	Canal 2	0.0
16	11/01/2017	12:59 p.m.	75	0	0	Canal 1	1.603157	1066.463167	Canal 2	0.0
17	11/01/2017	12:59 p.m.	80	0	0	Canal 1	1.603157	1066.463167	Canal 2	0.0
18	11/01/2017	12:59 p.m.	86	0	0	Canal 1	1.603157	1066.463167	Canal 2	0.0
19	11/01/2017	12:59 p.m.	90	0	0	Canal 1	1.603157	1066.463167	Canal 2	0.0
20	11/01/2017	01:00 p.m.	95	0	0	Canal 1	1.603157	1066.463167	Canal 2	0.0
1	11/01/2017	01:04 p.m.	0	0	0	Canal 1	0.158851	1065.824988	Canal 2	0.0
2	11/01/2017	01:04 p.m.	5	0	0	Canal 1	0.158851	1065.824988	Canal 2	0.0
3	11/01/2017	01:04 p.m.	10	0	0	Canal 1	0.158851	1065.824988	Canal 2	0.0
4	11/01/2017	01:04 p.m.	15	0	0	Canal 1	0.158851	1065.824988	Canal 2	0.0
5	11/01/2017	01:04 p.m.	20	0	0	Canal 1	0.158851	1065.824988	Canal 2	0.0
6	11/01/2017	01:04 p.m.	25	0	0	Canal 1	0.158851	1065.824988	Canal 2	0.0
7	11/01/2017	01:04 p.m.	30	0	0	Canal 1	1.600932	1065.614597	Canal 2	0.0
8	11/01/2017	01:04 p.m.	35	0	0	Canal 1	1.599789	1065.627109	Canal 2	0.0

Figura 5.40 *Ejemplo de toma de lectura en el archivo TXT.*

5.7 Alertas contra errores

El primer programa cuenta con una serie de alertas al oprimir el botón “COMENZAR” en el caso que existan errores al momento de introducir la información. Dentro de estas alertas está definir si el dato que introduce el usuario es un número, en caso de serlo, demandará al usuario que escriba un número positivo para esa variable. El segundo caso se suscita cuando se introduzcan números negativos; si se considera que se parte desde el origen absoluto, no puede existir un número negativo ya que eso haría mover el motor del eje en cuestión hacia un espacio que no existe o forzar el avance quemando los motores y dañando las piezas del posicionador.

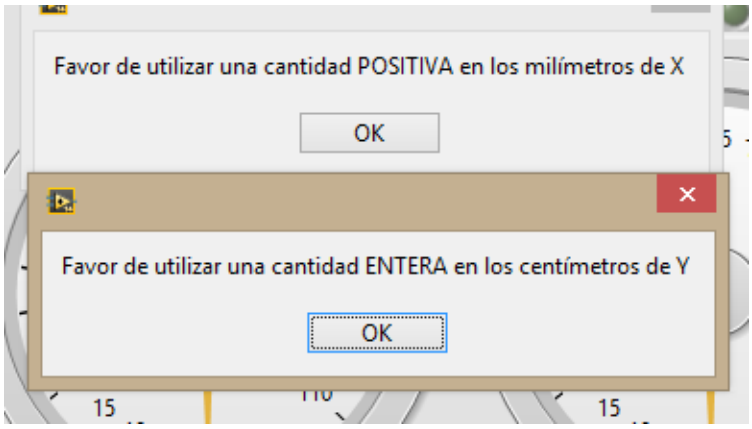


Figura 5.41 *Ejemplo de alertas desplegadas.*

Al momento que se detecta un error y se despliega la alerta, el usuario deberá oprimir el botón “Aceptar” de la ventana emergente y automáticamente el valor de la variable con el error se convertirá en cero hasta que se dé un número correcto. El programa ha sido hecho de manera que no avancen los motores o los sensores inicien su proceso de conteo hasta que el botón “COMENZAR” sea oprimido sin que se presente un error, es decir, si y sólo si no hay presencia de errores (números negativos o elementos que no sean números), al oprimir el botón “COMENZAR” el algoritmo seleccionado se ejecutará; en caso de que hayan errores se pedirá reescribir el dato de la variable con el error y volver a presionar el botón “COMENZAR”.

En el diagrama de bloques se hizo uso de una serie de condiciones booleanas ligadas a los posibles casos en que se cometería el error. Si estas condiciones resultan verdaderas, una estructura *switch case* se ejecutará en su caso TRUE. Dentro de la estructura se cuenta con el bloque *One Button Dialog* que permite a LabVIEW crear ventanas emergentes y conectado a este último, una cadena de caracteres que contiene el mensaje que se desea desplegar. Dentro de la misma estructura *switch case* se encuentra un control que cambia el valor de la variable con error a cero. Para este propósito se han utilizado seis estructuras switch case: los centímetros y milímetros de cada uno de los tres ejes.

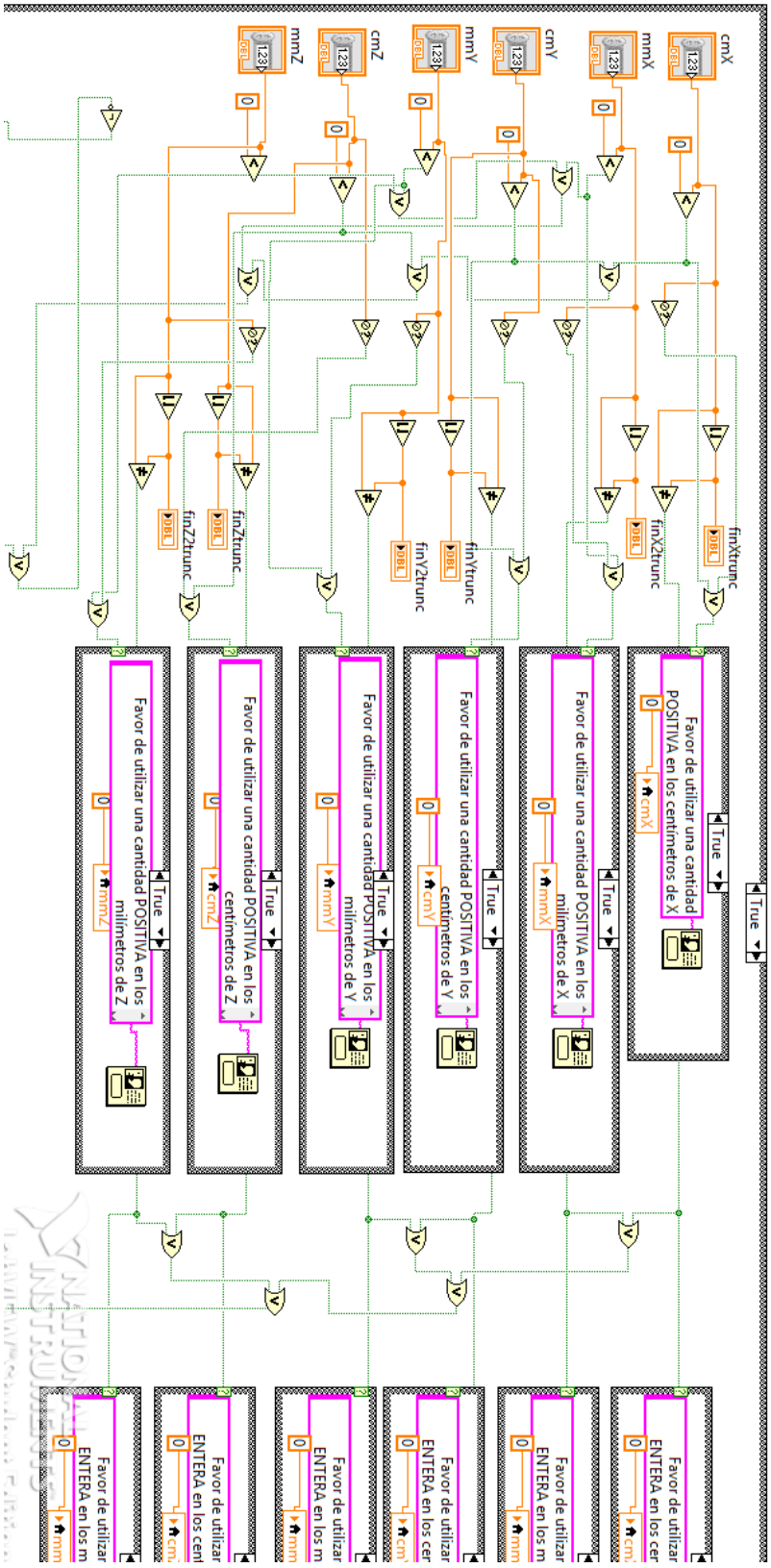


Figura 5.42 *Programación de alertas de error.*

5.8 Pruebas de funcionamiento

Las pruebas de funcionamiento de cada programa se han hecho de manera paulatina y por partes, Desde el funcionamiento del programa para mover sólo un motor hasta la integración de todos los elementos del programa. Cabe destacar que las secciones de código que involucran una estructura *Formula node* fueron creadas con la IDE de Aduino y fue necesaria su modificación para que fuesen ejecutables en LabVIEW.

Las primeras pruebas realizadas fueron las que involucraban la marcha de los tres motores. Al mismo tiempo se realizó un VI que fuera capaz de medir los pulsos de cada encoder y los sensores switch bumper. Cuando se confirmó la funcionalidad de ambos se procedió a combinar ambos VI's y hacer las modificaciones necesarias para su correcto funcionamiento.

El algoritmo para el movimiento de barrido se realizó en Arduino y posteriormente se hizo su equivalente en LabVIEW, vinculándolo con las secciones del programa mencionadas en el párrafo anterior. Para una apreciación más clara sin la necesidad de estar todo el tiempo en contacto con el posicionador, se creó una reproducción del circuito en una menor escala para realizar las pruebas de funcionamiento básico y transportarlo de manera más cómoda. Se realizaron pruebas para la lectura del osciloscopio de manera independiente así como para el almacenamiento de lecturas. Una vez que fue garantizado su funcionamiento se anexaron dichas secciones al programa principal y se hicieron las correcciones pertinentes para su adecuado funcionamiento.

Las pruebas de cada programa en su completo funcionamiento se realizaron cuidando que el conteo de pasos fuera correcto para cada revolución, es decir que 128 pasos fueran una revolución y 1280 fueran diez revoluciones, se cuidó que se conservara la posición del eje garantizando que los pasos son constantes. Finalmente se han realizado pruebas con el movimiento de barrido considerando la repetitividad en cada prueba buscando que siempre se obtuviera el mismo resultado si las condiciones iniciales son las mismas.

Para mostrar la lectura de los sensores se ha utilizado el osciloscopio Tektronix DPO 3014. Las pruebas realizadas se hicieron primeramente con un canal para evaluar la apreciación de imagen así como para programar el análisis de la frecuencia y la amplitud. Se ha utilizado igualmente un generador de señales y se han probado diferentes amplitudes de entre 0.5 hasta 15 voltios y frecuencias desde 100 hasta 15000 Hertz; se introdujeron para estas pruebas diferentes tipos de señal de entrada, desde una sinusoidal hasta señales cuadradas. En cada caso de evaluación se ha conseguido el muestreo de todas las señales a diferentes amplitudes y frecuencias.

Finalmente se consideró la realización de pruebas para el almacenamiento de datos. Al igual que con las demás partes del código general, la construcción del programa ha sido paulatina; se ha buscado primeramente crear el archivo y posteriormente escribir dentro de él, finalmente asignar múltiples caracteres y darle formato al archivo.

Las pruebas del programa completo se han realizado tanto para el programa uno como para el programa dos. Para el primero se ha buscado hacer múltiples pruebas asignando diferentes números de pasos a recorrer y regresar al posicionador a su posición de origen. En el caso del segundo programa se han igualmente hecho pruebas asignando distintos pasos a recorrer y registrando varias lecturas con ayuda del generador de señales, dichas pruebas han resultado en la generación de un archivo de texto TXT que contiene los datos de coordenadas y valores de frecuencia y amplitud de cada canal. Si bien el programa funciona de manera adecuada, es necesario puntualizar que la computadora consume demasiados recursos pudiendo, en ocasiones, presentar un error en la comunicación del osciloscopio; este error se habrá de solucionar pulsando “Aceptar” en la alerta que emerja y el programa volverá a funcionar de manera normal.

CAPÍTULO VI
Pruebas, Resultados y Conclusiones

6.1 Introducción

El proyecto presente cuenta con diferentes sistemas que trabajan en conjunto para crear una máquina capaz de posicionar un elemento en el espacio. Más específicamente este proyecto comprende de un sistema eléctrico que alimenta los motores y sensores del posicionador, un sistema mecánico capaz de desplazarse en tres ejes y una programación que permite el control de todo el posicionador en su conjunto; cada sistema ha de realizar su correcta función para que se cumpla con el objetivo planteado.

Es por tanto necesario entender que cada parte debe ser puesta a funcionar bajo distintos esquemas de evaluación. Considerando esta idea se han realizado pruebas particulares para cada sistema buscando que cada aspecto funcione adecuadamente y que en su conjunto cumplan con la labor propuesta en los objetivos. Para este propósito se han realizado pruebas de funcionalidad. Cabe mencionar que para cada prueba realizada se hizo un mínimo de cinco iteraciones, es decir, al menos cinco pruebas para cada parte que forma el sistema del posicionador ya que, en ocasiones, algún problema puede no presentarse en una única prueba pero si en subsecuentes.

6.2 Pruebas eléctricas

Las pruebas eléctricas comprenden todo lo relacionado al funcionamiento del circuito: su alimentación, rectificación y regulación; la distribución de los elementos al igual que su correcto funcionamiento, la interacción con la placa Arduino así como la construcción de la placa de circuito impreso.

Una vez que se consideró la arquitectura del circuito para alimentar los motores y se realizó el circuito en una tarjeta de conexiones protoboard, se hicieron las pruebas para la marcha y el sentido de giro del motor con la ayuda de una placa Arduino empleando el ejemplo de la IDE de Arduino, “MotorKnob”; en esta prueba se comprobó que la lógica seguida para el diseño del circuito era la correcta. Se hicieron las pruebas con tres motores a pasos modelo SM-42BYG011 de la marca Mercury Motor, que equiparan en su principio de funcionamiento a los que se encuentran montados en el posicionador; para estas pruebas se usó como fuente un cargador de laptop con una salida de 12 voltios y una corriente de 2 amperes. Las pruebas con el circuito fueron exitosas.

Para el caso de los sensores las pruebas fueron realizadas con la ayuda de la tarjeta Arduino UNO junto con una placa de conexiones protoboard. La conexión de cada sensor se realizó conforme a las especificaciones del fabricante en la hoja de datos o data sheet. Al demostrarse que la funcionalidad era correcta se contempló su inclusión dentro de la placa PCB. Una vez finalizada la placa de circuitos impresos se procedió a utilizar los mismos motores previamente mencionados y la misma fuente de energía. Esta vez con el apoyo de la placa Arduino MEGA se hicieron pruebas bajo la IDE de Arduino y el programa ejemplo “StepperOneRevolution” obteniendo un correcto funcionamiento.

Además de las pruebas de continuidad con multímetro, se midieron en la placa PCB las salidas de voltaje destinadas a los sensores garantizando que fueran de cinco voltios. La etapa posterior fue la conexión de los sensores y evaluar su correcto funcionamiento dentro de la placa PCB con la ayuda del Arduino MEGA y un programa de lecturas digitales creado para la evaluación de funcionamiento de los sensores. En las pruebas referidas a los sensores existieron ciertos errores tanto de continuidad como problemas físicos de conexión en las terminales; las fallas fueron atendidas hasta que se obtuvo una funcionalidad adecuada. El circuito encargado de alimentar y comunicar los sensores funciona de manera correcta.

Los circuitos de la etapa de potencia, el regulador y rectificador, no fueron realizados. Debido a que estos elementos formaban ya parte del proyecto anterior, no fue necesaria la creación de algún circuito de esta índole, únicamente se confirmó que en su salida se obtenía un voltaje de 12 voltios.

Por el lado de la seguridad se obtuvo en la placa PCB un circuito funcional que permite el aislamiento entre las terminales lógicas y de las terminales de la etapa de potencia. La placa PCB fue asegurada dentro de su caja contenedora y la comunicación con los encoders de cuadratura se hace a través de los conectores RJ45, estas medidas aseguran un buen funcionamiento aún en presencia de movimiento de los cables de conexión.

Finalmente se puede considerar que el circuito funciona de manera correcta. Los motores son alimentados con un voltaje medido con un multímetro, de 12.04 voltios y una corriente pico de 1.4 amperios. Los sensores por su cuenta, con cinco voltios y una corriente no superior a 25 miliamperios. Se tiene un circuito seguro para la manipulación externa y protegido de descargas, suciedad superficial y golpes ligeros debido a su contención dentro de una caja de plástico.

6.3. Pruebas mecánicas

Las pruebas mecánicas son aquellas que forman parte del movimiento del posicionador sin tomar en cuenta aspectos eléctricos o de programación. Dentro de estas pruebas se incluyen el correcto desplazamiento, el libre corrimiento de los husillos, las cotas de origen y límites en el posicionador así como errores que provocan atascamiento o retrasos. Se consideró primeramente probar el giro de los husillos de manera manual y posteriormente con la ayuda de un motor, limpiar y aceitar los ejes y evaluar su facilidad de movimiento con un motor a pasos. De esta manera se confirmó el giro de cada husillo, la facilidad que este tenía para girar con un motor a pasos y qué tanto avance era permitido por husillo antes de que este se trabara o atorara.

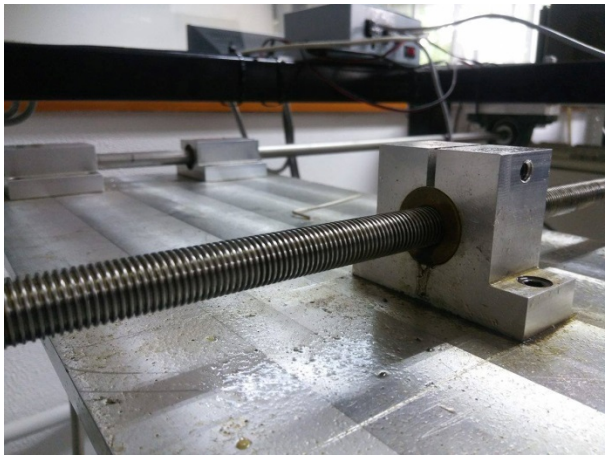


Figura 6.1 *Husillo de eje del posicionador.*

Primeramente debe entenderse que el posicionador fue diseñado, manufacturado y ensamblado en el IIMAS en el año 2010. El desplazamiento se realiza a través de un husillo de 12 hilos por pulgada, cada eje tiene su propio husillo. Conectado a la terminal trasera del eje del motor se ha incluido un cople que transmite el movimiento rotativo del motor hacia el encoder, igualmente se anexó el uso de una carcasa, previamente diseñada, que mantiene al encoder estático y alineado axialmente.

Se han realizado pruebas de corrimiento en el posicionador, es decir, pruebas en las que se ha hecho girar el husillo de cada eje de manera independiente para verificar que no existiesen imperfecciones, suciedad o algún elemento que no permita un libre avance.

La manera para verificar qué tanto podía desplazarse cada eje fue poner en marcha el motor de cada uno en sentido positivo hasta llegar al punto en que se comenzaban a suscitar problemas de avance. Del punto máximo que avanzó cada eje, se retrocedieron cinco revoluciones, considerando que no es adecuado llevar cualquier máquina hasta su límite físico y acotar su movimiento a un tanto antes de dicho límite. De estos puntos se fijaron marcadores y se tomó la distancia del origen a cada uno de ellos, truncada al entero inferior más próximo. Se encontró que para el eje X el desplazamiento máximo es de 43 centímetros, para el eje Y es de 40 centímetros y para el eje Z de 18 centímetros.

6.3.1 Caracterización de los ejes

A pesar de ser hechos con mucha precaución, los husillos fueron realizados a mano. Teniendo esta idea presente es natural pensar que no se trabaja con ejes idénticos; aunque tengan el mismo número de hilos es altamente probable que no sean iguales en cada uno de sus aspectos.

Se observó que existe una diferencia entre los pasos del motor y los pasos del encoder acoplado a ese motor. Cada motor tiene un total de 200 pasos por revolución mientras que cada encoder tiene un total de 128 pasos por revolución. El husillo de cada eje tiene 12 hilos por pulgada lineal, para dar una revolución, cada motor debe dar un total de 200 pasos. Esto hace que, para recorrer una pulgada, cada motor debe dar un total de

2400 pasos. Utilizando la misma lógica, si el encoder tiene 128 pasos, una pulgada lineal se habría de recorrer al dar un total de 1536 pasos del encoder. Sin embargo esto no sucede así, con esta cantidad de pasos, el encoder sólo consigue un avance de 21.65 milímetros de los 25.40 que comprenden una pulgada. Se considera que las razones pueden ser la falta de precisión en los husillos, errores de lectura (poco probable) o "saltos" no deseados que no son registrados al girar el encoder.

A pesar de que no existe una relación matemática entre los pasos necesarios para una pulgada entre el motor y el encoder, sí se encuentra una relación en las distancias avanzadas a partir de un *n* número de pasos. Existe una repetitividad en el avance lineal en relación a pasos del encoder. Por ejemplo, si se le solicita al programa que el encoder registre un total de 1000 pasos, el posicionador en ese eje recorre una distancia *d*, si se le vuelve a solicitar esa misma cantidad de pasos a registrar, la distancia *d* volverá a ser la misma. Existe un patrón que relaciona una cantidad de pasos con una distancia recorrida. No obstante fue necesario encontrar la distancia que ofrece cada paso del encoder y, considerando que cada husillo tiene pequeñas diferencias con sus hermanos, encontrar la medida de la distancia recorrida por paso en cada uno de los tres ejes.

Para determinar qué tanto avanza cada paso de cada encoder se optó por buscar una medida estándar. Se definió como objetivo recorrer una pulgada (25.40 mm). Dentro del programa se ingresaron números al azar buscando la cercanía más próxima a una pulgada, dependiendo del avance obtenido se agregaron o redujeron pasos en la siguiente iteración. Se encontró bajo este método que la cantidad de pasos más adecuada a 25.40 milímetros es: para el eje X un total de 2048 pasos, para el eje Y 2060 y para el eje Z 2045 pasos. Las distintas pruebas fueron realizadas con un vernier electrónico con una resolución de centésimas de milímetro.

Considerando que cada eje tiene un número de pasos de encoder necesarios para avanzar una pulgada, al dividir esta distancia entre el número de pasos necesarios se obtuvo el avance por paso de encoder en cada eje. Se resolvió que para el eje X cada paso tiene un avance de 12.4023 micrómetros, para el eje Y 12.3300 micrómetros y para el eje Z 12.4205 micrómetros por paso. De esta manera esta información pudo ser agregada al programa principal como factores para las operaciones de desplazamiento. Al realizarse pruebas definiendo las distancias por paso de cada encoder, se encontró que físicamente el desplazamiento es el mismo al indicado en la interfaz, es decir, se comprobó que si se le solicita al programa avanzar 20.5 centímetros, físicamente esta distancia será la recorrida.

6.4 Pruebas de interfaz

Por el lado de la interfaz se han realizado distintas pruebas de funcionamiento. A través de los elementos de control que muestra el Panel Frontal de LabVIEW se ha permitido el desplazamiento del posicionador por medio de órdenes dadas por el usuario. La forma en que estas órdenes son recibidas por LabVIEW es también un campo a evaluar. Se realizaron pruebas en tres aspectos: Manipulación correcta de información, velocidad de respuesta y protección contra errores de usuario.

La recepción de información de LabVIEW es fácil de evaluar, si bien por medio de un teclado entran los datos de escritura al programa, la información pertinente a los

sensores y lecturas de instrumentos periféricos debe realizarse a través de la comunicación serial del Arduino. Se probó comunicar LabVIEW con los encoders de cuadratura y realizar giros (pulsaciones) a distintas velocidades encontrando que con velocidades con pausas menores a 150 milisegundos puede presentar problemas de lectura; al realizarse más pruebas se encontró que lo más adecuado es que el programa tenga pausas de 200 o 250 milisegundos entre iteraciones. En el caso de los sensores Switch bumper se realizó un mínimo de cinco ciclos de diez iteraciones de pulsos, obteniendo que en cada una se registraba la lectura de manera adecuada. Por el lado de la lectura del osciloscopio se encontraron problemas inherentes a la comunicación serial con LabVIEW, agregando el hecho de que este software pierde velocidad de procesamiento al trabajar con el osciloscopio. Se encontró la presencia de errores de conexión al exceder los 15 segundos de espera de conexión entre LabVIEW y el osciloscopio al arrancar el programa. No obstante las lecturas se realizan de manera correcta en un tiempo aceptable.

En el caso de la velocidad de respuesta, por el lado de los sensores se tiene un resultado gratificante, los sensores responden en un tiempo sumamente bajo. Los motores igualmente tienen una respuesta rápida ante los pulsos de Arduino significando que la velocidad de comunicación entre la tarjeta y la computadora es alta. La respuesta por parte del osciloscopio resulta ser un tanto lenta en comparación con las demás, al consumir muchos recursos de la computadora la velocidad de procesamiento se ve mermada afectando incluso la velocidad de los demás procesos. No obstante se considera que el programa completo, trabajando en conjunto tiene una velocidad aceptable puesto que la lectura de sensores y osciloscopio se realiza a la velocidad de la máquina y el movimiento de motores presenta un retraso no mayor a 200 milisegundos.

En la interfaz se hicieron pruebas para posibles errores del usuario. Dentro de los cuadros de diálogo que solicitan la escritura de un dato, al colocar un elemento no esperado se desplegará una señal de alarma indicando el problema y borrando el elemento erróneo. Sin importar qué tipo de desplazamiento se desee realizar, no se permite iniciar el movimiento ni lecturas antes de que todos los datos solicitados en los cuadros de diálogo sean semánticamente correctos.

Finalmente se ha obtenido un programa de control y una interfaz interactiva de fácil uso, de una programación robusta y a prueba de errores del usuario. Ambos programas son de fácil comprensión y con un diseñado en medida que, sin importar quién lo use, el objetivo de la tarea sea cumplido. Se tiene una interfaz y programa cuya velocidad de procesamiento es adecuado, los elementos visuales son suficientes para los intereses generales de la labor a realizar y que, además, es capaz de registrar las lecturas en un archivo de texto. El funcionamiento es adecuado y aunque se ha probado cada VI para funcionar en una computadora con un rendimiento medio-alto, no se recomienda el uso simultáneo con otros programas.

6.5 Resultados finales

Si bien cada aspecto del posicionador (parte eléctrica, interfaz y programación) fue probada de manera independiente, también fue necesaria la prueba de todos los elementos en conjunto que conforman el presente proyecto. Es decir, basándose en el objetivo del proyecto se han de realizar las pruebas que evalúen el funcionamiento

general de todo el posicionador, incluidos la parte eléctrica, los algoritmos de desplazamiento, la interfaz y el muestreo de datos.

Se han realizado pruebas junto con el osciloscopio Tektronix DPO 3014 y un sensor ultrasónico (fabricado en el IIMAS) como punto terminal; las pruebas se realizaron para los cuatro canales del osciloscopio. Se ha colocado una distancia de 1, 5.5, 10, 12.7 y 17.5 centímetros para pruebas lineales, de superficie y volumétricas a una resolución de 12, 24, 48 y 96 micrómetros para los puntos de lectura. En cada experimento pueden variar estos valores de desplazamiento y puntos de lectura según el deseo del usuario. Las pruebas con las mencionadas configuraciones resultaron exitosas generando un archivo de texto.

Cada experimento genera un archivo de texto que contiene la información de la hora y día así como el número de iteración y la lectura de cada canal del osciloscopio con la posición en coordenadas donde fue tomada. La información de coordenadas se ofrece en centímetros con más de dos cifras significativas lo que permite observar el valor de los micrómetros, y los valores de amplitud y frecuencia por canal con cuatro cifras significativas.

Por el lado de la interfaz se han obtenido dos programas, uno para posicionar el punto terminal en las coordenadas deseadas, a partir de las cuales comenzará el barrido de lectura, con una resolución de milímetros y la opción de retorno al origen. El segundo programa, encargado del barrido y las lecturas del osciloscopio, con una resolución de desplazamiento de milímetros y de lectura de 12 micrómetros. Ambos programas forman parte de la interfaz hombre-máquina, de comprensión sencilla y fácil uso.

	Eje X	*Eje Y*	*Eje Z*
Avance máximo	43 cm	40 cm	18 cm
Resolución	12.402 µm	12.33 µm	12.42 µm

Finalmente se tiene un posicionador en tres dimensiones capaz de moverse en diferentes tipos de desplazamiento con alta precisión, a diferentes distancias y con retroalimentación; capaz de realizar lecturas provenientes de un osciloscopio Tektronix DPO, y registrar dichas lecturas en las coordenadas donde fueron tomadas, generando como resultado de operación un archivo de texto que contiene las lecturas de manera ordenada y estructurada. Y que, una vez terminado su proceso de lectura, es capaz de regresar a su posición de origen en espera del próximo experimento a realizar.

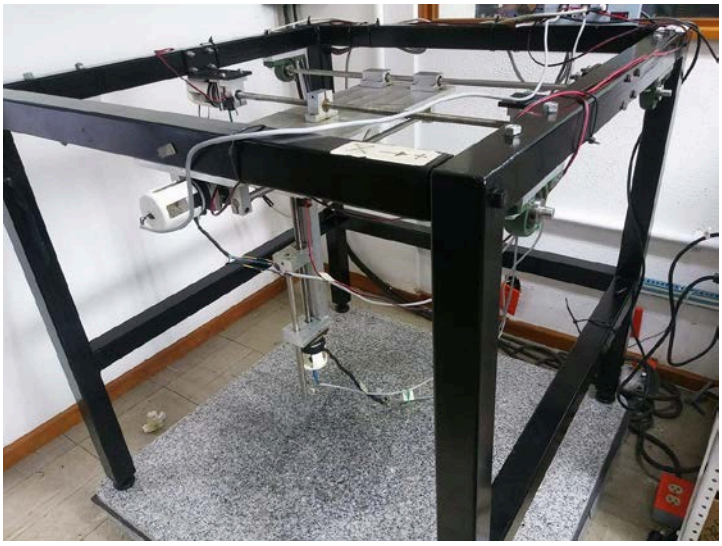


Figura 6.2 *Posicionador XYZ.*

Conclusiones

Se diseñó y construyó el circuito eléctrico para accionar los motores del posicionador, así también fue programada la interfaz para manipularlo. Se obtuvo como resultado un sistema cuya resolución se encuentra en poco más de 12 micrómetros, dependiendo del eje, capaz de moverse de manera uniforme en la mayor parte de su extensión, en diferentes formas de desplazamiento mientras se registran y almacenan datos provenientes de un osciloscopio electrónico.

La vinculación de LabVIEW con sistemas de arquitectura abierta como Arduino se cumplió de manera satisfactoria en cuanto a su funcionamiento y operación. Si bien el posicionador fue programado y es potenciado en LabVIEW, la dependencia al uso de tarjetas de adquisición de datos de esta (u otra) compañía fue eliminada con la creación de la placa de circuitos impresos y el uso de la placa Arduino MEGA.

A partir de las pruebas realizadas con los algoritmos para el uso de este posicionador, se han alcanzado los objetivos de selección de movimiento y distancias de desplazamiento. Igualmente la tarea de vinculación con el osciloscopio Tektronix DPO 3014 se considera exitosa, así como el almacenamiento de datos del mismo. Los objetivos referidos a la programación fueron satisfactoriamente alcanzados.

Los resultados obtenidos con las pruebas realizadas demuestran que se han logrado los objetivos propuestos para este trabajo de tesis. Se consiguió hacer de este sistema de posicionamiento, una herramienta de fácil manipulación capaz de adquirir y almacenar información útil en el proceso de caracterización de sensores ultrasónicos.

Trabajos a futuro

Se considera que si bien el posicionador tiene un funcionamiento adecuado, uno de los principales conflictos que se presentan es el uso excesivo de recursos de la computadora cuando esta trabaja en conjunto con el osciloscopio. De esta manera es considerablemente importante que, si se desea seguir trabajando con la interfaz creada en este proyecto, se cuente con una computadora de mayores recursos de procesamiento y memoria. En caso contrario se propone el uso de otro programa cuya interfaz sea más ligera para la computadora evitando el uso de LabVIEW para esta labor.

Teniendo en consideración que se ha optado por el uso de arquitectura abierta, se busca depender menos de licencias y productos de los cuales se necesite su compra como, por ejemplo, tarjetas de adquisición de datos y/o licencias de programa LabVIEW. Por lo tanto, si se ha usado una tarjeta Arduino en el trabajo presente, se propone que el entorno en el cual se diseñe y programe la interfaz sea igualmente de libre uso al público. Para este fin se propone el uso de Processing o Java como entornos para vincular al ser humano con el posicionador.

Finalmente dentro de los aspectos técnicos del posicionador se espera que, para el siguiente trabajo, se eliminen las fallas físicas de los husillos, es decir, buscar husillos cuyas imperfecciones sean diminutas volviéndolos (casi) iguales en su desplazamiento. También se espera la modificación y el correcto ajuste de las piezas de ensambles de la estructura del posicionador puesto que actualmente presentan imperfecciones que dificultan trabajar con la estructura interrumpiendo el giro de los motores y la falta de aprovechamiento de rango máximo de desplazamiento.

Referencias

[1] MENA Rivas, Alex Francisco. *Propuesta de diseño de manipulador cartesiano*. Proyecto de Grado, Universidad EAN 2010

[2] GARCÍA, Marisol de Jesús. *Diseño de una interfaz para un manipulador tipo scara que pueda usarse para ensamble electrónico*. Tesis de licenciatura. Escuela superior de Ingeniería Mecánica y Eléctrica, IPN, 2007

[3] ENGEL. *Robot cartesiano AV100*. Robot Industry. (Disponible en: http://www.directindustry.es/prod/engel/product-20405-1044693.html, 13 de mayo de 2016)

[4] PEPPERL. FUCHS*. Lectura de placas de circuitos impresos*. (Disponible en: http://www.pepperl-fuchs, el 13 de mayo de 2016)

[5] Avedaño M, Luis Enrique. (2005). *Fundamentos de Instrumentación*. Pereira, Colombia: Universidad Tecnológica de Pereira.

[6] Dzul May, Yarely. Olmedo Garcia, Mario. (2007). *Circuitos electrónicos para propósito de control y monitoreo de temperatura vía PC* (Tesis de pregrado). Facultad de Ingeniería en Electrónica y Comunicaciones, Universidad Veracruzana, PozaRica, Ver.

[7] Cooper, William. Helfrick, Alberrt. (1991). *Instrumentación electrónica moderna y técnicas de medición*. Editorial Pearson.

[8] Ogata, Katsuhiko. (2010). *Ingeniería de control moderna*. Madrid, España: Editorial Pearson.

[9] National Instruments Corporation (2016). *LabVIEW*. Ustin, TX. Recuperado de http://www.ni.com/labview/esa/

[10] García, Ricardo. *Fundamentos de Mecatrónica, Ch3: Sensores y transductores* [diapositiva]. Arica, Chile. Universidad de Tarapacá, 2016, 34 diapositivas, recuperado de Sitio web:

http://www.eudim.uta.cl/rmendozag/courses/2012/fundamentos_de_mecatronica/fundamentos_mecatronica_03.pdf

[11] National Instruments. (2016). *Encoder Measurements: HowTo Guide National Instruments*. 2016, de Ni.com Sitio web: http://www.ni.com/tutorial/7109/en/

[12] Hibder, William. *Diseño e implementación de una interfaz máquina–usuario en LabVIEW, para el control de un sistema de posicionamiento automatizado*. Tesis de Licenciatura. México, Ciudad de México. Universidad Nacional Autónoma de México, Facultad de Ingeniería, 2010.

[13] Xyoos Les cours d'informatique gratuits. (2008-2016). *Rj45*. Francia: Xyoos Publishing. Recuperado de http://cours-informatique-gratuit.fr/dictionnaire/rj45/

[14] Duran Ortega, Joel A. (2005). *Diseño y construcción de un sistema de posicionamiento automatizado ultrasónico*. Tesis de licenciatura. México. Facultad de Ingeniería, UNAM.

[15] National Instruments. (2016). *What Is Data Acquisition? - National Instruments*. 12 agosto 2016, de Ni.com Sitio web: http://www.ni.com/data-acquisition/what-is/esa.

[16] Amazon. (1996-2016*). Belkin Hi-Speed USB Cable*. Amazon.com, Inc. Recuperado de https://www.amazon.com/Belkin-Hi-Speed-USB-Cable-Feet/dp/B00004Z5M1.

[17] Zetina, Ángel. (2004). Electrónica básica. México, DF: Limusa.

[18] Arduino. (2016). Arduino Mega. Arduino products. Recuperado de https://www.arduino.cc/en/Main/ArduinoBoardMega2560.

[19] Angulo Usategui, José. (2009). Microcontroladores PIC". Editorial Paraninfo, 1º Edición.

[20] Zbar, Malvino, Miller. (2006). Prácticas de electrónica. México: Glencoe/McGraw Hill.

ANEXO 1: Manual de usuario

MANUAL DE USUARIO

El presente manual tiene como objetivo informar sobre la manipulación del posicionador 3D del laboratorio de Ultrasonido del DISCA. Se explican las funcionalidades básicas y se busca un correcto funcionamiento explicando paso a paso las instrucciones de uso. El conjunto de programas realizados para manipular el posicionador se han realizado en la versión 15 de LabVIEW, con la anexión de un osciloscopio Tektronix DPO 14XX series.

¿Cómo funcionan los programas del posicionador?

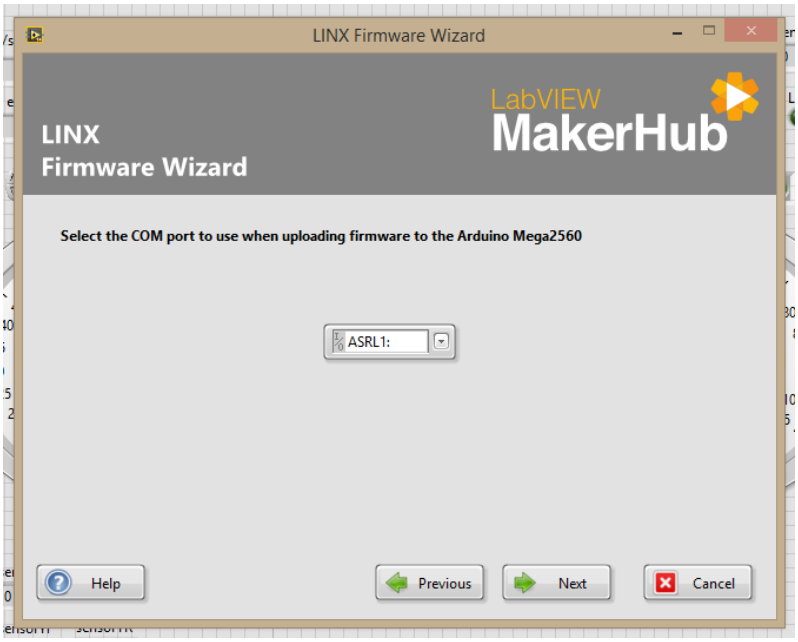
El posicionador cuenta con dos programas. El primer programa es un VI generado en LabVIEW llamado IDA_REGRESO.vi que permite la ubicación espacial del punto terminal del posicionador en las coordenadas indicadas por el usuario de un segundo origen, en el caso de que se parta del origen absoluto. La otra función de este programa es el retorno al origen para cuando el punto terminal se encuentra en una posición que no sea la de origen. El segundo programa de nombre LECTURA.vi es el encargado de realizar el desplazamiento de mapeo, lectura del osciloscopio digital y almacenamiento de lecturas.

Configuración inicial

Para que cualquiera de los dos programas funcione es necesario configurar la conexión entre LabVIEW y la tarjeta Arduino. Para este fin se hace uso del Firmware LINX de MakerHub. Para comenzar a configurarlo es necesario presionar en el menú superior la ventana Tools y seleccionar la opción MakerHub, LINX y luego la opción LINX Firmware Wizard. Automáticamente se deplegará una ventana para selección de opciones

En este menú de selección se puede elegir la familia de dispositivos, tipo de tarjeta y tipo de interfaz. Para el uso del posicionador presente se ha utilizado el Arduino MEGA por lo que se requiere seleccionar “Arduino” en la opción “Device Family” y posteriormente Arduino MEGA en el submenú “Device Type”, para la selección “Firmware Program Interface” se usará Serial/USB. Una vez finalizada la selección de estas opciones es necesario presionar el botón Next.

La siguiente ventana requiere de una selección de puerto COM. Es necesario buscar en la casilla de selección el puerto COM donde se encuentre conectado el Arduino MEGA (esta información puede buscarse en el Panel de Control de Windows en Administración de dispositivos). En caso de que no aparecer el puerto COM del Arduino existe la opción “Refresh”, por su traducción, permite refrescar la lista de dispositivos conectados. Si aun así la opción no aparece, entonces se debe revisar la conexión de la tarjeta Arduino.

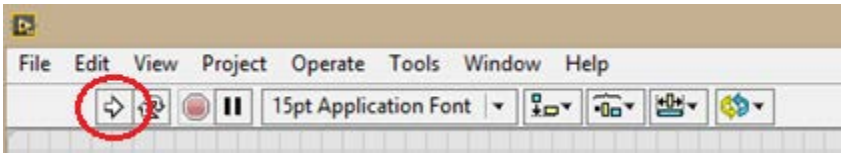


Después de seleccionar el puerto COM, el paso siguiente es dar click en Next y nuevamente aparecerá una ventana nueva, a esta nueva ventana se le dará Next nuevamente. Inmediatamente se mostrará una ventana que muestra la conexión con la tarjeta y finalizará la configuración

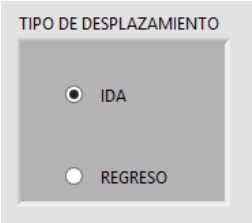
Nota Importante: Cada vez que un programa se termine, ya sea porque ha acabado su función o por un error, se necesitará reiniciar LabVIEW desde su origen, esto es, cerrar por completo el programa hasta que no forme parte de los procesos de la computadora. Entonces se necesitará volver a abrirlo y configurarlo otra vez.

Posicionar el punto terminal en coordenadas de segundo origen XYZ

Para posicionar el punto terminal en un segundo origen se requiere el uso del programa IDA_REGRESO.vi. Como todo programa de LabVIEW se requiere inicializar el programa con el botón “Run” en la parte superior izquierda de la ventana del panel frontal.



Una vez inicializado el programa la rejilla del panel frontal cambiará a un tono gris sólido que indicará que el programa ha comenzado a ejecutarse, Se debe esperar igualmente a que los indicadores de los motores cambien a un color verde. Después de este paso será posible seleccionar el tipo de movimiento, para ubicar el actuador en el segundo origen es necesario seleccionar la casilla que dice “IDA”.



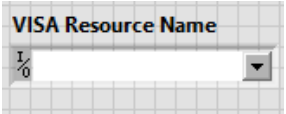
Ya seleccionada la opción IDA se debe colocar dentro de las casillas de desplazamiento por eje, la distancia a recorrer. Estas distancias vienen ya configuradas en opciones de centímetros y milímetros. En caso de colocar cero en alguna de ellas o en su defecto no colocar número alguno (cero también), el actuador no recorrerá ninguna distancia, es decir, permanecerá inactivo.

Una vez agregadas las distancias deseadas a recorrer se presionará el botón COMENZAR para iniciar el movimiento deseado. Se observará cómo los indicadores de avance de cada motor se mueven señalando que estos giran. Una vez que se haya llegado a la posición deseada, surgirá una alarma indicando que el posicionamiento ha finalizado y el programa se cerrará automáticamente.

Lectura de muestras

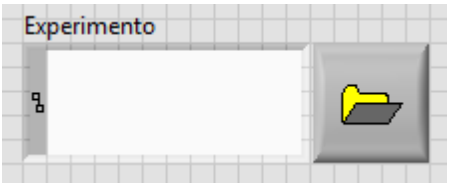
Para la lectura, desplazamiento de barrido y almacenamiento de muestreos se utilizará el programa de nombre LECTURA.vi. Es importante no inicializar el programa en LabVIEW. El primer paso a realizar es configurar el osciloscopio, este se debe encender y conectar vía USB a la computadora. Posteriormente se seleccionará en el puerto VISA Resource Name el osciloscopio con el que se trabajará, de no aparecer se debe revisar la conexión con la computadora.

Nota Importante: para que el programa funcione se necesita que los cuatro canales del osciloscopio estén conectados a un cable BNC y en la interfaz del osciloscopio se habilite la imagen de cada canal.



Después deseleccionar el puerto VISA se debe seleccionar el destino donde serán guardados los archivos de texto. Se pulsará el botón “Experimento” que abrirá inmediatamente el

Explorador de Windows, aquí se debe buscar el destino donde se desee guardar el experimento a realizar, es necesario igualmente escribir el nombre del experimento a realizar.



Al igual que con el programa IDA_REGRESO.vi, el programa de lectura necesita inicializarse. Una vez pulsado el botón Run comenzará a operar el programa. El programa estará habilitado para su uso hasta que los indicadores de funcionamiento del motor estén de color verde y las imágenes de señales del osciloscopio sean visibles.

Una vez habilitado el programa se habrán de escribir en las casillas de centímetros y milímetros las distancias a recorrer así como el tipo de desplazamiento (Línea recta, Superficie o Volumen).

Ya seleccionado el tipo de movimiento y distancias a recorrer se habrá de escribir las distancias (en micrómetros) a la que se tomarán las lecturas. Existe un indicador que muestra qué tantos pasos se desean y a qué distancia equivale, de este modo se tomara una lectura cada vez que un múltiplo de esa distancia se recorra.

Cuando se encuentren listas todas las selecciones se tendrá que pulsar el botón COMENZAR, de haber algún error se desplegará una alerta para corregirlo. Si todo es correcto iniciará el movimiento y el proceso de lectura, cuando este termine se desplegará una alerta de finalización y se cerrará el programa.

Retorno al origen

Cuando se haya finalizado el proceso de lectura se necesita regresar al origen el posicionador. Para este fin se requiere el uso del programa IDA_REGRESO.vi. Al igual que para el proceso de ubicación en el segundo origen, se requiere inicializar el programa y esperar a que cambien de color los indicadores de los motores.

Para seleccionar el retorno sólo basta dar click en la opción Regreso y posteriormente COMENZAR. El punto terminal del posicionador automáticamente iniciará su retorno en cada uno de sus ejes hasta llegar a su inicio de carrera. Al finalizar dicho movimiento se mostrará una alerta indicando que el actuador ha llegado a su posición original y el programa se terminará.

ANEXO 2: Diagramas de flujo

Modelo general de funcionamiento:

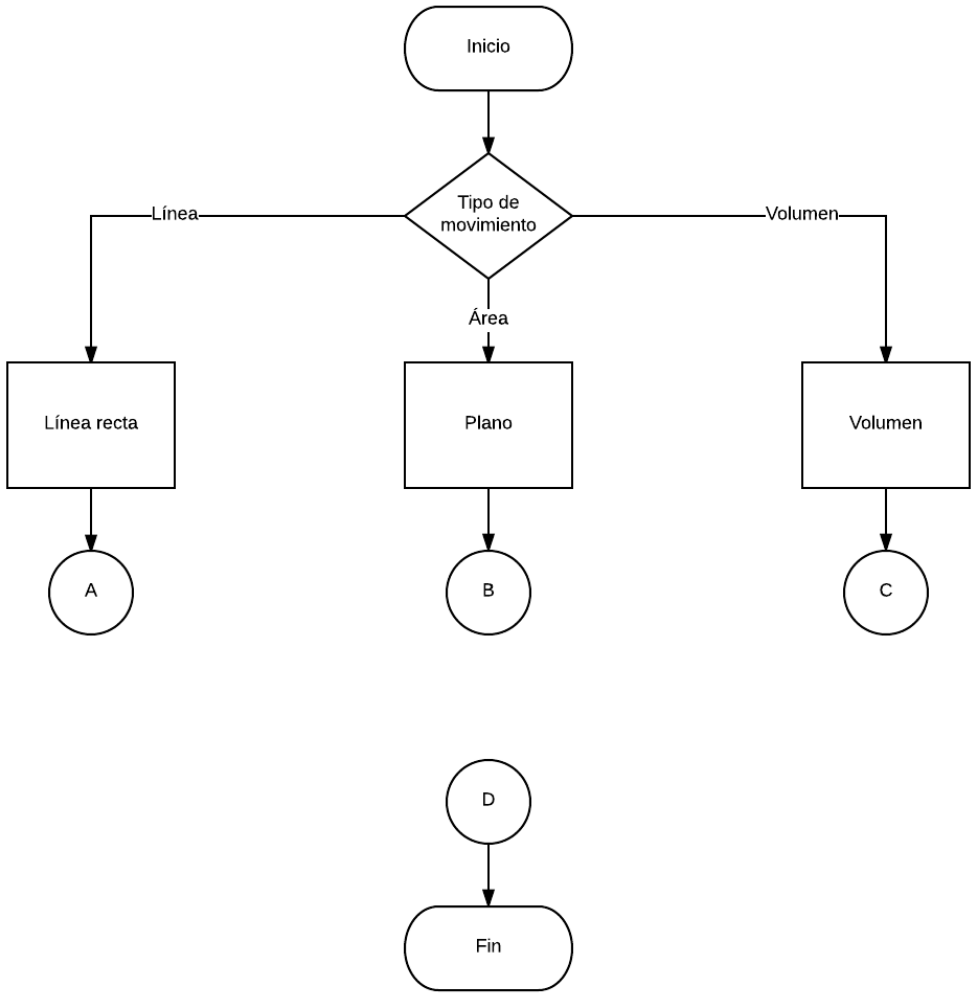


Diagrama del algoritmo de ida y regreso

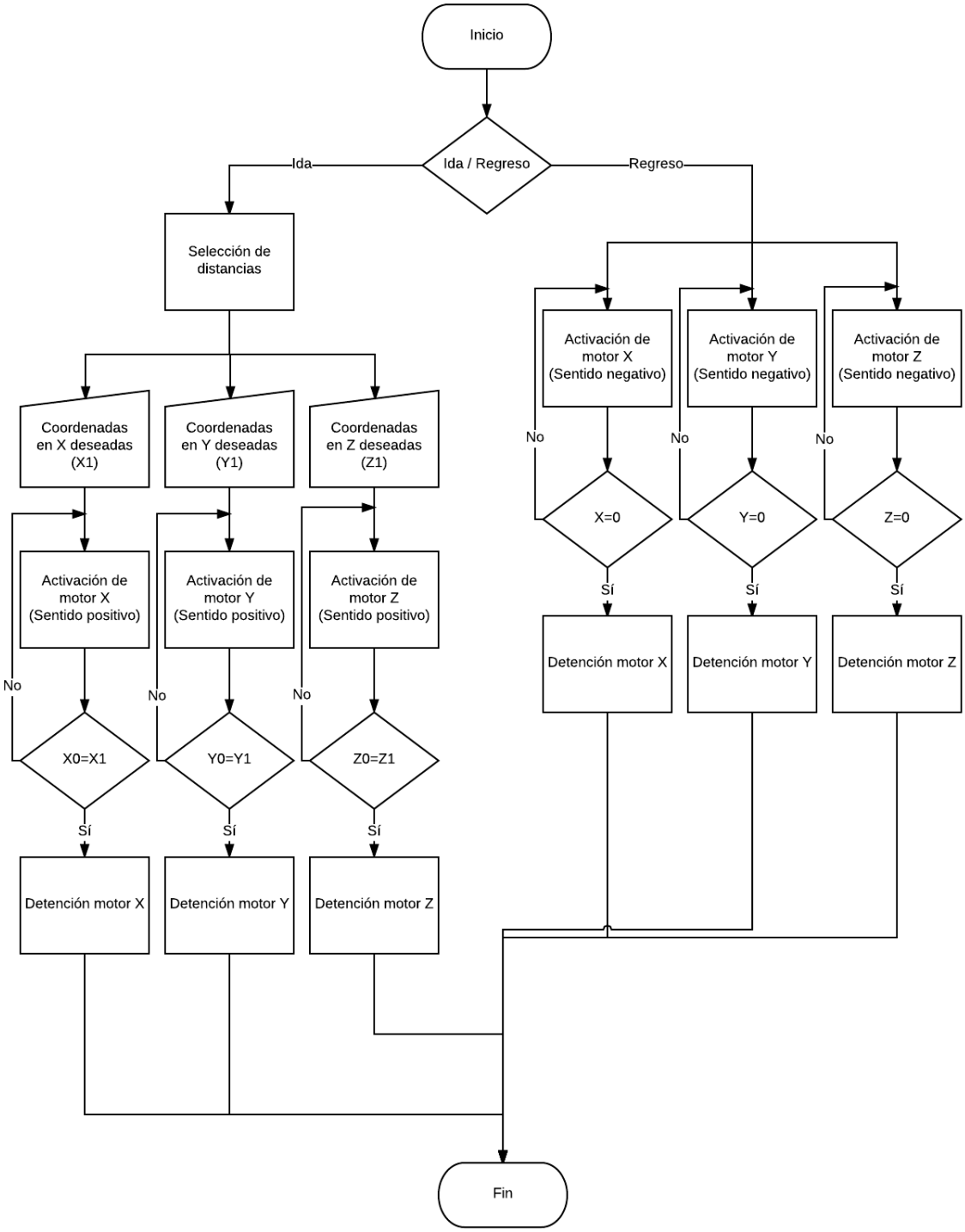


Diagrama del algoritmo de desplazamiento en línea recta:

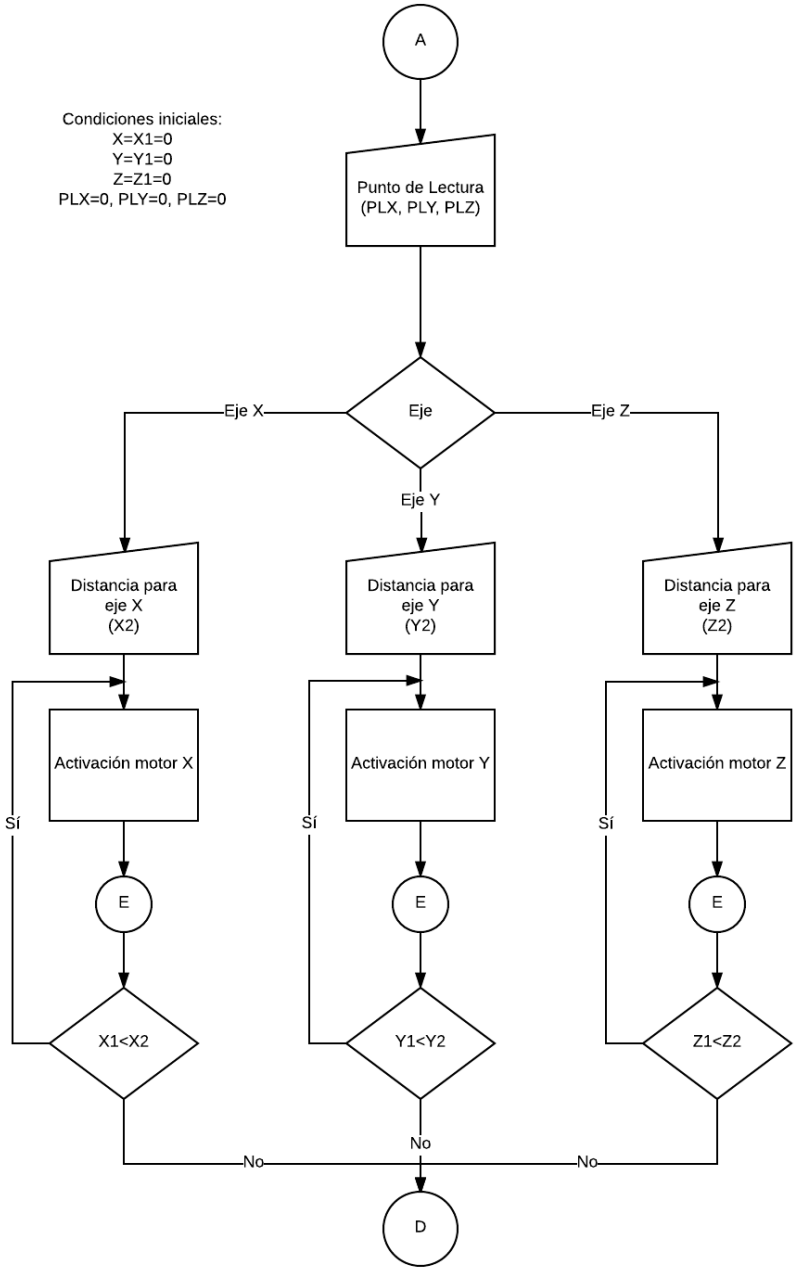
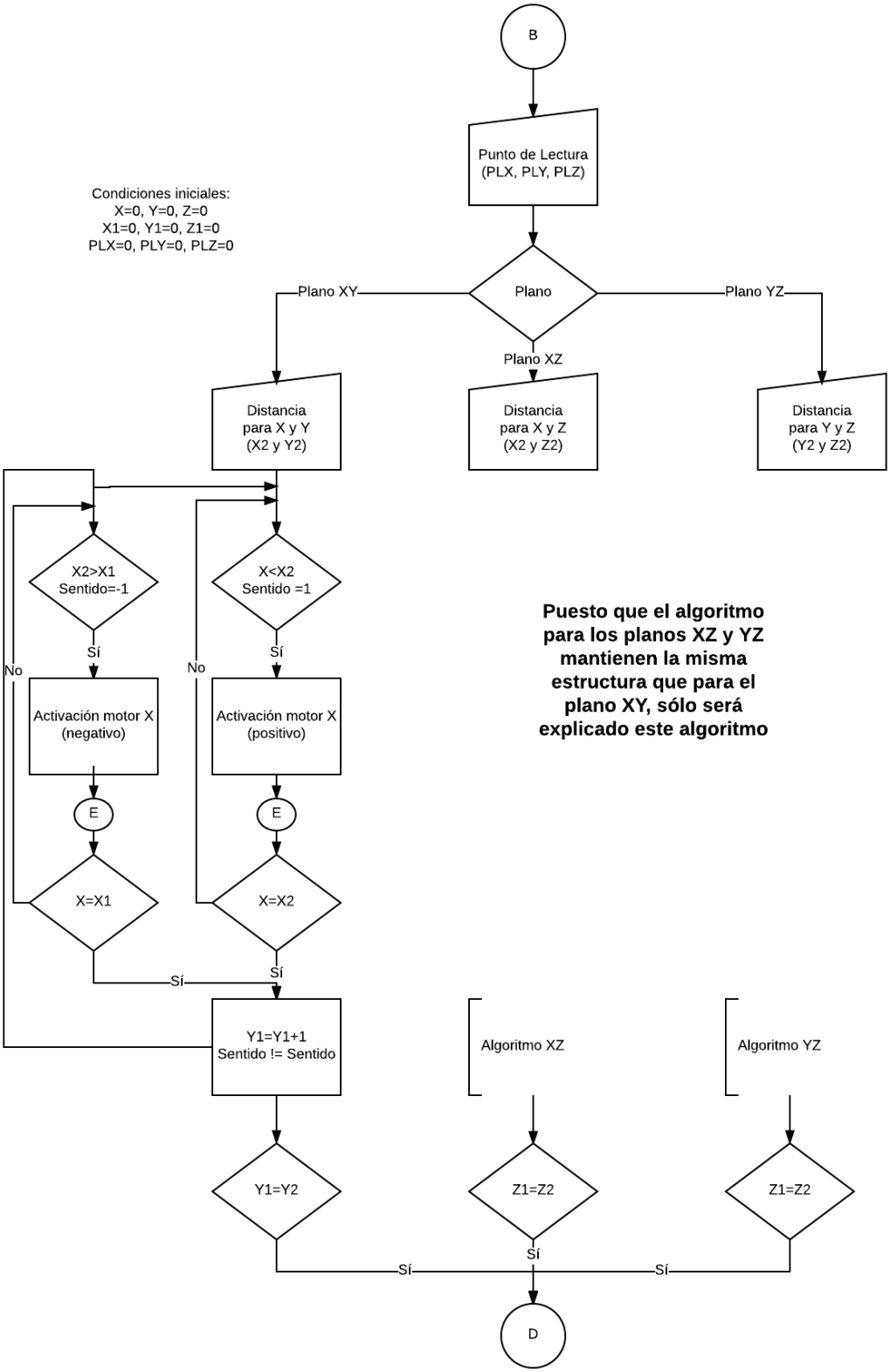


Diagrama del algoritmo de desplazamiento en planos:



Puesto que el algoritmo para los planos XZ y YZ mantienen la misma estructura que para el plano XY, sólo será explicado este algoritmo

Diagrama de algoritmo de desplazamiento en volumen:

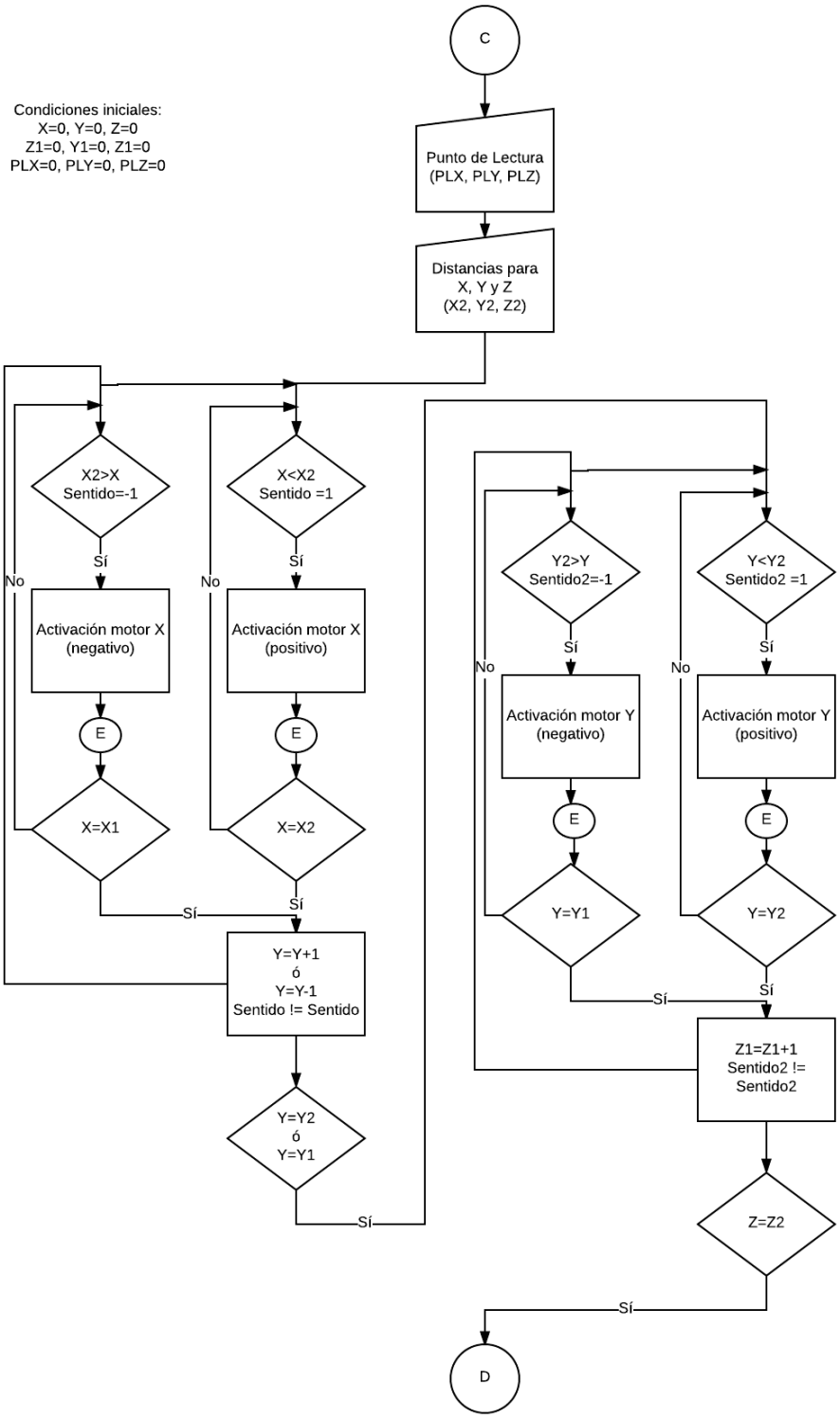
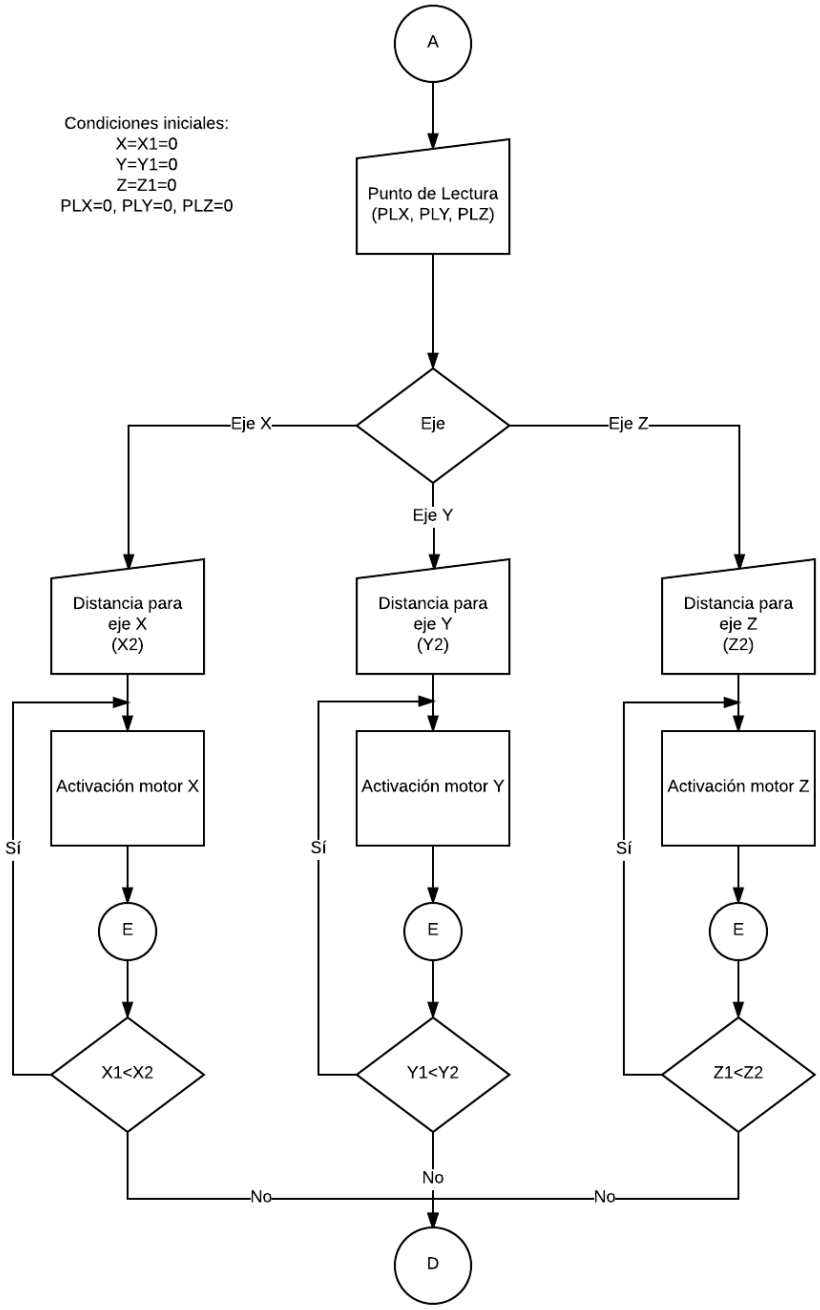


Diagrama del algoritmo de lectura de señales:



ANEXO 3: Código C del programa IDA_REGRESO.vi

```
if(inicio==1)
{

if(a_1>127)
a_1=0;

if(a_1<0)
a_1=127;


 if((b1-a)>0)
g1=1;

 if((b1-a)<0)
g1=-1;

if ((b1-a)==0)
g1=0;

if(ida==1&&sZR==0)
{

  xini=S2;


if(g1==1)
  {
    if(b==0&&xini!=x)
    {
      b=1;
      a++;
      a_1++;
      x=S2;
    }
    if(b==1&&xini!=x)
    {
      b=0;
      a++;
      a_1++;
      x=S2;
    }

  }

  if(g1==-1)
  {
    if(b==0&&xini!=x)
    {
      b=1;
      a--;
      a_1--;
      x=S2;
    }


    if(b==1&&xini!=x)
    {
      b=0;
      a--;
      a_1--;
      x=S2;
    }
  }
}

if(regreso==1)
{
if(S1==0)
{
g1=-1;
}
else
{
g1=0;
}
}

 }
```

Nota: Este código fue utilizado en las tres estructuras Formula Node dentro del diagrama de bloques de LabVIEW

ANEXO 3: Código C del programa LECTURA.vi

```
if(arranque==1)
{
if(control==1)
{
       if (x % puntoDeLecturaX ==
0&&bandLec==0)
       {
        gx=0;  gy=0;  gz=0;  ejeX1 = 0; ejeX2 =
0; ejeY1 = 0; ejeY2 = 0; ejeZ = 0; detencion=1;
bandLec = 1;  Lec=Lec+1;
      }
       else
       {
        detencion=0;
       }
}

if(control==2)
{
       if (x % puntoDeLecturaY ==
0&&bandLec==0)
       {
        gx=0;  gy=0;  gz=0;  ejeX1 = 0; ejeX2 =
0; ejeY1 = 0; ejeY2 = 0; ejeZ = 0; detencion=1;
bandLec = 1;  Lec=Lec+1;
      }
       else
       {
        detencion=0;
       }
}

if(control==3)
{
       if (x % puntoDeLecturaZ ==
0&&bandLec==0)
       {
        gx=0;  gy=0;  gz=0;  ejeX1 = 0; ejeX2 =
0; ejeY1 = 0; ejeY2 = 0; ejeZ = 0; detencion=1;
bandLec = 1;  Lec=Lec+1;
      }
       else
       {
        detencion=0;
       }
}

if(control==4)
{
       if (x % puntoDeLecturaX ==
0&&y%puntoDeLecturaY==0&&bandLec==0)
       {
        gx=0;  gy=0;  gz=0;  ejeX1 = 0; ejeX2 =
0; ejeY1 = 0; ejeY2 = 0; ejeZ = 0; detencion=1;
bandLec = 1;  Lec=Lec+1;
       }
       else
       {
        detencion=0;
       }
}

if(control==5)
{
       if (x % puntoDeLecturaX ==
0&&y%puntoDeLecturaZ==0&&bandLec==0)
       {
        gx=0;  gy=0;  gz=0;  ejeX1 = 0; ejeX2 =
0; ejeY1 = 0; ejeY2 = 0; ejeZ = 0; detencion=1;
bandLec = 1;  Lec=Lec+1;
       }
       else
       {
        detencion=0;
       }
}

if(control==6)
{
       if (x%puntoDeLecturaY==0&&y%
puntoDeLecturaZ== 0&&bandLec==0)
       {
        gx=0;  gy=0;  gz=0;  ejeX1 = 0; ejeX2 =
0; ejeY1 = 0; ejeY2 = 0; ejeZ = 0; detencion=1;
bandLec = 1;  Lec=Lec+1;
      }
       else
       {
        detencion=0;
       }
}
```

```
if(control==7)
{
        if
(x%puntoDeLecturaX==0&&y%puntoDeLectu
raY==0&&z % puntoDeLecturaZ==
0&&bandLec==0)
        {
         gx=0;  gy=0;  gz=0;  ejeX1 = 0; ejeX2 =
0; ejeY1 = 0; ejeY2 = 0; ejeZ = 0; detencion=1;
bandLec = 1;  Lec=Lec+1;
       }
        else
         {
          detencion=0;
         }
}

if (z<z2 + 1)
{

  xbis = sen1;
  ybis = sen2;
  zbis = sen3;

  sen1 = S1;
  sen2 = S2;
  sen3 = S3;

  if (y < y1 || y > y2)
  {
    auxY = 1;
  }
  else
  {
    auxY = 0;
  }

  ////////////////////////////////////
SUBEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
  if (x < x2 && condicionX == 0)
  {
    estado = 1;
    ejeX1 = 1;
    ejeX2 = 0;
    ejeY1 = 0;
    ejeY2 = 0;
    ejeZ = 0;
    gx=-1;
    gy=0;
    gz=0;

  if (sen1 == 0)
  {
    if (xbis != sen1)
    {
      x = x + 1;

      bandLec = 0;
    }
  }
  if (sen1 == 1)
  {
    if (xbis != sen1)
    {
      x = x + 1;

      bandLec = 0;
    }
  }

  }

  ////////////////////////////////   COTA
SUPERIOR

  if (x >= x2 && condicionX == 0)
  {
    ///subeeeee y
//////////////////////////////////////////////
//////////
    if (y < y2 && condicionY == 0)
    {
      ejeX1 = 0;
      ejeX2 = 0;
      ejeY1 = 1;
      ejeY2 = 0;
      ejeZ = 0;

      gx=0;
      gy=-1;
      gz=0;

    estado = 21;
```

```
    if (sen2 == 0)
    {
      if (ybis != sen2)
      {
        y = y + 1;
        condicionX = 1;

      }
    }
    if (sen2 == 1)
    {
      if (ybis != sen2)
      {
        y = y + 1;
        condicionX = 1;

      }
    }

  }

  ////////////////////////////////// cota
sup Y
  if (y >= y2 && condicionY == 0)
  {
    ejeX1 = 0;
    ejeX2 = 0;
    ejeY1 = 0;
    ejeY2 = 0;
    ejeZ = 1;

     gx=0;
     gy=0;
     gz=1;

    estado = 22;

    if (sen3 == 0)
    {
      if (zbis != sen3)
      {
        z = z + 1;
        condicionY = 1;

      }
    }
    if (sen3 == 1)
    {
      if (zbis != sen3)
      {
        z = z + 1;
        condicionY = 1;

      }
    }

  }

  ///bajaaaaaaaaaaaaaaaaaaa y
//////////////////////////////////////////////
//////////
  if (y > y1 && condicionY == 1)
  {
    ejeX1 = 0;
    ejeX2 = 0;
    ejeY1 = 0;
    ejeY2 = 1;
    ejeZ = 0;

     gx=0;
     gy=11;
     gz=0;

    estado = 23;
    if (sen2 == 0)
    {
      if (ybis != sen2)
      {
        y = y - 1;
        condicionX = 1;

      }
    }
    if (sen2 == 1)
    {
      if (ybis != sen2)
      {
        y = y - 1;
        condicionX = 1;

      }
    }

  }
```

```
   ////////////////////////////////// cota inf
Y
   if (y <= y1 && condicionY == 1)
   {
    ejeX1 = 0;
    ejeX2 = 0;
    ejeY1 = 0;
    ejeY2 = 0;

     gx=0;
     gy=0;
     gz=1;

    ejeZ = 1;
    estado = 24;

    if (sen3 == 0)
    {
     if (zbis != sen3)
     {
      z = z + 1;
      condicionY = 0;

     }
    }
    if (sen3 == 1)
    {
     if (zbis != sen3)
     {
      z = z + 1;
      condicionY = 0;

     }
    }

   }

 }

 ////////////////////////////////
BAJAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAA
 if (x > x1 && condicionX == 1)
 {
  estado = 3;
  ejeX1 = 0;
  ejeX2 = 1;
  ejeY1 = 0;
  ejeY2 = 0;
  ejeZ = 0;

    gx=1;
    gy=0;
    gz=0;

  if (sen1 == 0)
  {
   if (xbis != sen1)
   {
    x = x - 1;

   }
  }
  if (sen1 == 1)
  {
   if (xbis != sen1)
   {
    x = x - 1;

   }
  }

 }

 ////////////////////////////  COTA
INFERIOR

 if (x <=x1 && condicionX == 1)
 {
  //estado=4;

  ///subeeeee y
////////////////////////////////////////////
//////////
   if (y <y2 && condicionY == 0)
   {
    ejeX1 = 0;
    ejeX2 = 0;
    ejeY1 = 1;
    ejeY2 = 0;
    ejeZ = 0;

     gx=0;
     gy=-1;
     gz=0;
```

```
    estado = 41;
    if (sen2 == 0)
    {
      if (ybis != sen2)
      {
        y = y + 1;
        condicionX = 0;

      }
    }
    if (sen2 == 1)
    {
      if (ybis != sen2)
      {
        y = y + 1;
        condicionX = 0;

      }
    }

  }

  ////////////////////////////////// cota
sup Y
  if (y >= y2 && condicionY == 0)
  {
    condicionX = 1;

    ejeX1 = 0;
    ejeX2 = 0;
    ejeY1 = 0;
    ejeY2 = 0;
    ejeZ = 1;

     gx=0;
     gy=0;
     gz=1;

    estado = 42;

    if (sen3 == 0)
    {
      if (zbis != sen3)
      {
        z = z + 1;
        condicionY = 1;
        condicionX = 0;
      }
    }
    if (sen3 == 1)
    {
      if (zbis != sen3)
      {
        z = z + 1;
        condicionY = 1;
        condicionX = 0;
      }
    }

  }


  ///baajaaaaaaa y
////////////////////////////////////////////
//////////
  if (y > y1 && condicionY == 1)
  {
    condicionX = 1;
    ejeX1 = 0;
    ejeX2 = 0;
    ejeY1 = 0;
    ejeY2 = 1;
    ejeZ = 0;

     gx=0;
     gy=11;
     gz=0;

    estado = 43;

    if (sen2 == 0)
    {
      if (ybis != sen2)
      {
        y = y - 1;
        condicionX = 0;

      }
    }
    if (sen2 == 1)
    {
      if (ybis != sen2)
      {
        y = y - 1;
        condicionX = 0;
```

```
      }
    }

  }

  ////////////////////////////////// cota inf
Y
  if (y <= y1 && condicionY == 1)
  {
    ejeX1 = 0;
    ejeX2 = 0;
    ejeY1 = 0;
    ejeY2 = 0;
    ejeZ = 1;

     gx=0;
     gy=0;
     gz=1;

    estado = 44;

    if (sen3 == 0)
    {
      if (zbis != sen3)
      {
        z = z + 1;
        condicionY = 0;

      }
    }
    if (sen3 == 1)
    {
      if (zbis != sen3)
      {
        z = z + 1;
        condicionY = 0;

      }
    }

  }

 }

fin = 0;
}
else
{
    fin=1;
    ejeX1 = 0;
    ejeX2 = 0;
    ejeY1 = 0;
    ejeY2 = 0;
    ejeZ = 0;

     gx=0;
     gy=0;
     gz=0;

}

}
```