Universidad Carlos III de Madrid
Escuela Politécnica Superior
Departamento de Ingeniería de Sistemas y Automática

Proyecto final de carrera de Ingeniería Técnica Industrial:
Electrónica Industrial

Realización de drivers para LabVIEW

Autor: Jose Rosado Medina
Tutor: Arturo de la Escalera Hueso

Leganés, Julio 2011

Este proyecto supone para mí el fin de una etapa. Un orgullo.

Espero que dentro de unos años cuando eche la vista atrás y vea todo lo que me ha traído sea tan feliz como lo soy ahora mismo.

Para Marta, mi compañera, mi amiga.

Para el colegio San Viator, los Salesianos de Atocha y la Universidad Carlos III, por todo lo que aprendí y los buenos momentos que pasé.

Para Antonio, por volcar en mí una parcelita de tus conocimientos.

Para mis amigos y familiares por vuestro apoyo y confianza.

Gracias a todos.

Índice General

Índice de imágenes

Índice de tablas

INTRODUCCIÓN

Una de principales necesidades de hoy es el control total o parcial de las actividades industriales, para ello es necesario un mejor rendimiento tanto del equipamiento de la industria así como de los mismos trabajadores que son parte del proceso.

Es por ello que para mi proyecto final de carrera he realizado el diseño de un instrumento virtual (VI) que permita la configuración y control de un dispositivo láser mediante el puerto serie. El diseño de dicho instrumento se traduce en una interfaz en la que el usuario es capaz de acceder a todas las posibilidades que ofrece dicho instrumento de una manera sencilla e intuitiva.

Pero para lograr dicho instrumento utilizo un programa llamado LabVIEW el cual es proporcionado por la empresa internacional National Instruments. Dicha herramienta es un poderoso software con una plataforma gráfica de diseño de sistemas para el desarrollo de pruebas, control y diseño entre otros.

LabVIEW, software propiedad de National Instruments fue creado en el año 1987 teniendo como objetivo entregar al operador la posibilidad de realizar diversos análisis en varias áreas siendo unas de las más fuertes la automatización, siendo una de sus principales características la facilidad de manejo debido a que incorporaba un nuevo sistema de programación llamado G en alusión a su interfaz grafica. Al cabo de un tiempo muchos fueron los interesados en trabajar con él y por consiguiente muchas las aplicaciones realizadas lo que ha transformado a LabVIEW de una herramienta de análisis a un software capaz de resolver grandes problemas en la industria ya sea en el área de automatización entregando la posibilidad de visualizar diversos procesos o bien realización de SCADA, así como en comunicaciones, etc.

Finalmente, se diseña la interfaz que permita la utilización del programa al usuario y permita acceder a todas las posibilidades que ofrece.

OBJETIVOS

El objetivo fundamental de este proyecto es la comunicación RS232 con el láser SICK LMS 291-s05 utilizando como herramienta fundamental el LabVIEW, pudiendo controlar y visualizar los diferentes parámetros del láser.

Estoy comunicando el LabVIEW con un laser (SICK LMS 291-s05). Dicho laser envía tramas de tamaño variable, la longitud depende de la configuración del laser. Por ejemplo, para una lectura del laser de 0 a 100 grados con resolución 1 grado el numero de bytes es 202 (101 datos de 2 bytes cada dato, uno para cada grado en este caso).

Dentro de esta trama hay bytes que me interesan (que son los correspondientes a los datos y a la longitud) y bytes que no me interesan tener en cuenta en la medida como son los de STX, ADR, CMD, CRC... Además de cada dos bytes que recibo, no me interesan los bits 13, 14, 15. Yo sé con qué bytes comienza cada trama y con los bytes de "lenght" y su configuración, su longitud. El objetivo de este proyecto es saber es de qué manera puedo manejar las tramas en LabVIEW para poder presentar los bytes de datos.

Desde el primer momento me atrajo la idea de realizar mi proyecto final de carrera con una herramienta tan potente como LabVIEW. Ya había utilizado con anterioridad este programa pero quería seguir aprendiendo y trabajando con él y el poder utilizarlo en una comunicación RS232 es la forma perfecta.

Pero mi trabajo no se reduce al simple análisis sino también al control, en su mayor parte, gracias a la comunicación con el propio láser. Pudiendo mandar tramas de datos para cambiar los parámetros de configuración como los de encendido y apagado.

Es por ello que la principal ventaja que supone para mí este proyecto es la posibilidad de comunicar cualquier dispositivo RS232 con LabVIEW, puesto que al fin y al cabo todos los dispositivos comunican de forma similar, aunque cambie las tramas o la forma que tiene de enviarlas.

RESUMEN

Antes de realizar dicho instrumento virtual y debido a que es un software poco utilizado, he querido realizar en la primera parte del PFC una introducción teórica para que una persona que no lo haya utilizado nunca pueda entender en rasgos generales el funcionamiento del LabVIEW y su posterior utilización, de una manera, a mi parecer, amena y sencilla.

Una parte fundamental del proyecto, además del software, es el láser LMS 291-S05 con lo que he querido introducir un breve resumen de las principales tramas que permiten su utilización, en principio las de configuración y seguidamente las tramas para el control.

Para la comunicación con el láser he utilizado mi PC, el cual no tiene puerto RS232 con lo que he tenido que utilizar un conversor USB-RS232 con lo que también he incluido un breve resumen teórico.

La segunda mitad del proyecto se centra en el análisis experimental, en el que explico paso a paso las diferentes etapas que he seguido para llevar a cabo la implementación en LabVIEW, desde los VI utilizados y su conexión como el diseño que tiene la interfaz para la utilización del programa.

También he incluido un apartado en el que propongo las posibles mejoras propuestas para llevar a cabo en relación con este proyecto.

Por último he querido dejar constancia de mis conclusiones y bibliografía utilizada.

I. ANÁLISIS TEÓRICO

1.1 SCADAS

1.1.1 Historia

Los primeros SCADA (Supervisory Control And Data Adquisition) eran simplemente sistemas de telemetría que proporcionaban informes periódicos de las condiciones de campo vigilando las señales que representaban medidas y/o condiciones de estado en ubicaciones de campo remotas. Estos sistemas ofrecían capacidades muy simples de monitorización y control. La visión del operador en el proceso estaba basada en los contadores y las lámparas detrás de paneles llenos de indicadores. Mientras la tecnología se desarrollaba, los ordenadores asumieron el papel de manejar la adquisición de datos, disponiendo de comandos de control, y presentando la información sobre una pantalla de CRT. Los ordenadores agregaron la capacidad de programar el sistema para realizar funciones de control más complejas.

Los primeros sistemas automatizados SCADA fueron modificados con programas de aplicación específicos para atender a requisitos de algún proyecto particular. Hoy, los proveedores de SCADA están diseñando sistemas pensados para resolver las necesidades de distintas industrias con módulos de software específicos. Se puede encontrar software SCADA comercialmente disponible adaptado para procesamiento de papel y celulosa, oleoductos y gaseoductos, hidroeléctricas, distribución de agua, etc.

Los sistemas SCADA se están convirtiendo en una parte integral de la gestión corporativa. Estos sistemas ya no son vistos por la gerencia simplemente como herramientas de operación, sino como un recurso importante de información. Continúan sirviendo como centro de responsabilidad en la operación, pero también proporcionan datos a los sistemas y usuarios fuera del ambiente del centro de control que dependen de la información oportuna en la cual basan sus decisiones económicas cotidianas.

1.1.2 Conceptos básicos de los sistemas Scada

Los sistemas SCADA (Supervisory Control And Data Adquisition) son aplicaciones de software, diseñadas con la finalidad de controlar y supervisar procesos a distancia. Permiten a un usuario recoger datos de una o más instalaciones geográficamente distribuidas y/o enviar comandos y órdenes a estas instalaciones. Además, envían la información generada en el proceso productivo a diversos usuarios, tanto del mismo nivel como hacia otros supervisores dentro de la empresa, es decir, que permite la participación de otras áreas como por ejemplo: control de calidad, supervisión, mantenimiento, etc.

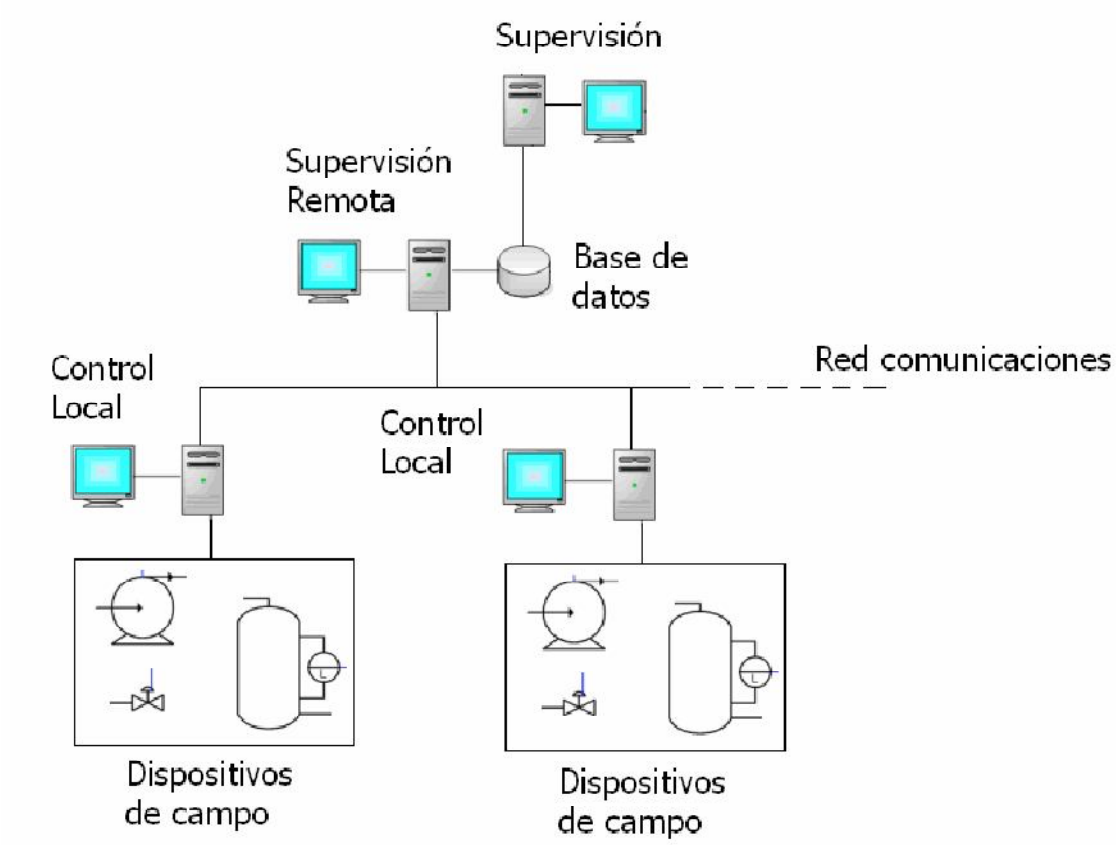


Fig. 1: *Estructura de un sistema SCADA*

El sistema está compuesto por numerosas unidades (controles locales), geográficamente distribuidas, recogiendo medidas y enviando órdenes a los dispositivos de campo, estas se encuentran conectadas a una o varias estaciones de supervisión remotas por medio de un sistema de comunicaciones. La estación de supervisión remota presenta la información adquirida en pantallas y permite al operador realizar tareas de control a distancia. Todo sistema SCADA tiene que cumplir una serie de requisitos básicos, que son:

⇒ Tener arquitectura abierta, es decir, debe permitir su crecimiento y expansión, así como tener capacidad de adecuarse a las necesidades futuras del proceso y de la planta.

⇒ La programación e instalación no debe presentar mayor dificultad, debe contar con interfaces gráficas que muestren un esquema básico y real del proceso.

⇒ Deben permitir la adquisición de datos de todo tipo de equipos, así como la comunicación a nivel interno y externo (redes locales y de gestión).

⇒ Deben ser programas sencillos de instalar, sin excesivas exigencias de hardware, y fáciles de utilizar, con interfaces amigables para el usuario.

1.1.3 Funciones de un Scada

Las funciones principales son:

- **Supervisión remota de instalaciones y equipos**: Permite al operador conocer el estado de los distintos dispositivos que componen la instalación.

- **Control remoto de equipos:** Mediante el sistema se pueden enviar órdenes a los controles locales para activar o desactivar los equipos remotamente (por ejemplo abrir válvulas, activar interruptores, etc.), de manera automática y también manual. Además es posible ajustar parámetros, valores de referencia, algoritmos de control, etc.

- **Visualización gráfica:** El sistema es capaz de ofrecer imágenes en movimiento que representen el comportamiento del proceso, dando al operador la impresión de estar presente dentro de una planta real. También pueden mostrar gráficos de las señales registradas en el tiempo.

- **Representación de señales de alarma:** A través de las señales de alarma se logra alertar al operador cuando tiene lugar una condición perjudicial o fuera de lo aceptable. Estas señales pueden ser tanto visuales como sonoras y se pueden realizar registros de incidencias.

- **Históricos:** Se cuenta con la opción de almacenar los datos adquiridos, esta información puede analizarse posteriormente, el tiempo de almacenamiento dependerá del operador o del autor del programa.

1.1.4 Transmisión de la información

Los sistemas SCADA necesitan comunicarse vía red, puertos GPIB, telefónica o satélite, es necesario contar con ordenadores que realicen el envió de datos hacia un ordenador remoto, este a su vez será parte de un centro de control y gestión de información.

Cada fabricante de equipos para sistemas SCADA emplean diferentes protocolos de comunicación y no existe un estándar para la estructura de los mensajes, sin embargo existen estándares internacionales que regulan el diseño de las interfaces de comunicación entre los equipos del sistema SCADA y equipos de transmisión de datos.

Un protocolo de comunicación es un conjunto de reglas y procedimientos que permite a las unidades remotas y central, el intercambio de información. Los sistemas SCADA hacen uso de los protocolos de las redes industriales.

1.1.5 Elementos del sistema

Un sistema SCADA está conformado por:

- **Interfaz Hombre-Máquina:** Permite la interacción del ser humano con los medios tecnológicos implementados. En el caso de un sistema SCADA la interfaz debe ofrecer al operador una recreación de lo que ocurre en la planta.

- **Supervisión Remota:** Se encarga de recibir los datos de la planta y enviar órdenes a la misma. También se encarga del almacenamiento y procesado ordenado de los datos, de forma que otra aplicación o dispositivo pueda tener acceso a ellos.

- **Control Local:** Lo constituye todo elemento que envía algún tipo de información a la aplicación de supervisión remota. Ejecuta las acciones del mando (programadas) en base a los valores actuales de las variables. Es parte del proceso productivo y necesariamente se encuentra ubicada en la planta.

- **Sistema de Comunicaciones:** Se encarga de la transferencia de información desde el punto donde se realizan las operaciones hasta el punto donde se supervisa y controla el proceso. Lo conforman los transmisores, receptores y medios de comunicación.

- **Transductores:** Son los elementos que permiten la conversión de una señal física en una señal eléctrica (y viceversa).

1.1.6 El mercado de los Scadas

En la industria hay una multitud de SCADA funcionando tanto para la supervisión y el control de un pequeño proceso de fabricación como para plantas enteras. Hay muchas empresas distribuidoras de sistemas SCADA, a menudo son proveedores de PLC, que realizan el SCADA que pueda comunicar con sus productos. Para elegir un sistema SCADA hay que tener en cuenta su compatibilidad con el entorno en el cual va a implementarse. Las diferencias que se encuentran entre distintos SCADA, se refieren principalmente al número de variables que el programa puede leer en tiempo real. Así, un SCADA debe elegirse según las características y limitaciones del proceso a controlar. Por ejemplo, la empresa Schneider Electric provee el software VIJEO LOOK para pequeñas y medianas aplicaciones (128, 512, 1024 I/O) y MONITOR PRO para aplicaciones grandes y complejas (256, 1024, o ilimitadas I/O). Otros de los paquetes SCADA más implantados en el mercado son:

CIRNET, de CIRCUTOR S.A.
SCADA InTouch, de LOGITEK.
WinCC, de Siemens.
1 Introducción 9
Coros LS-B/Win, de Siemens.
SYSMAC SCS, de Omron.
FIXDMACS, de Omron-Intellution.
RS-VIEW32 de Rockwell
GENESIS32 de Iconics

1.2 LABVIEW

1.2.1 LabVIEW

LabVIEW ("Laboratory Virtual Instrument Engineering Workbench") es un lenguaje de programación G con funciones integradas en el que se pueden generar rápidas y sencillas para el diseño de sistemas de adquisición de datos, instrumentación y control. LabVIEW es una creación de National Instruments. A diferencia de los lenguajes de programación de propósito general, LabVIEW tiene funciones específicas para acelerar el desarrollo de aplicaciones de medida, control y automatización. LabVIEW nos ofrece la posibilidad de conectar con otras aplicaciones y compartir datos a través de ActiveX, web, DLLS, librerías compartidas... En este caso nos centraremos en los datos compartidos mediante ActiveX ya que son los utilizados para desarrollar parte de este proyecto.

1.2.2 Programación G

G es como nos referimos al lenguaje gráfico de programación que utilizamos en LabVIEW, el cual usa una ejecución basado en el flujo de datos (dataflow). Al diseñar un programa de forma gráfica, se hace visible una programación orientada al flujo de datos, donde se obtiene una representación de los datos también de forma gráfica. Una función solo podrá ejecutarse cuando tenga disponibles todos los datos que le sirven como entradas. En conclusión, el flujo de datos va de izquierda a derecha en el diagrama de bloques y está determinado por las operaciones o funciones que procesan los datos.

1.2.3 Aplicaciones LabVIEW

LabVIEW tiene su mayor aplicación en sistemas de medición, como por ejemplo, visualización de procesos aplicaciones de control. LabVIEW se ha convertido en una herramienta de desarrollo estándar de la industria para aplicaciones de test en pruebas de producción. También puede emplearse para analizar y registrar resultados reales para aplicaciones en sectores como la automoción. Para las aplicaciones que requieren sonido y vibración, procesos de imagen, análisis de tiempo y frecuencia, wavelets diseño de filtros digitales, LabVIEW ofrece software extra especialmente diseñado para mejorar la velocidad del desarrollo del sistema.
Es posible realizar numerosos procesos de control y automatización y la excelente monitorización de maquinaria y trabajos de mantenimiento industrial.

1.2.4 National Instruments

National Instruments se fundó en 1976 es Austin, Texas y hoy en día, es una empresa referente líder en sistemas de medidas y automatización basados en PC.

1.3 PROGRAMACIÓN GRÁFICA CON LABVIEW

Cuando diseñamos programas con LabVIEW trabajamos bajo un VI, es decir, con un instrumento virtual. Los VI se caracterizan por ser un recuadro con su respectivo símbolo normalmente relacionado con su funcionalidad, tener entradas con su color de identificación de dato, tener una o varias salidas y por ser reutilizables.

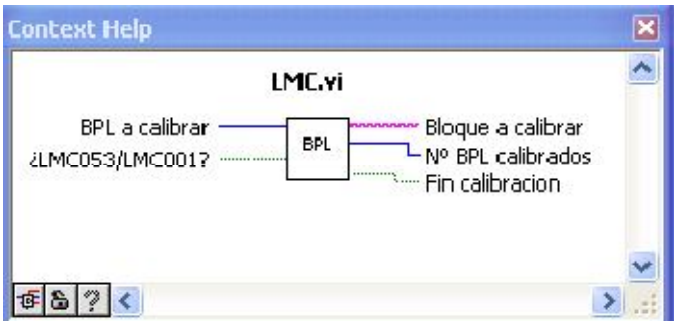


Fig. 2: *Ejemplo de un VI*

Los VI están estructurados de la siguiente manera:

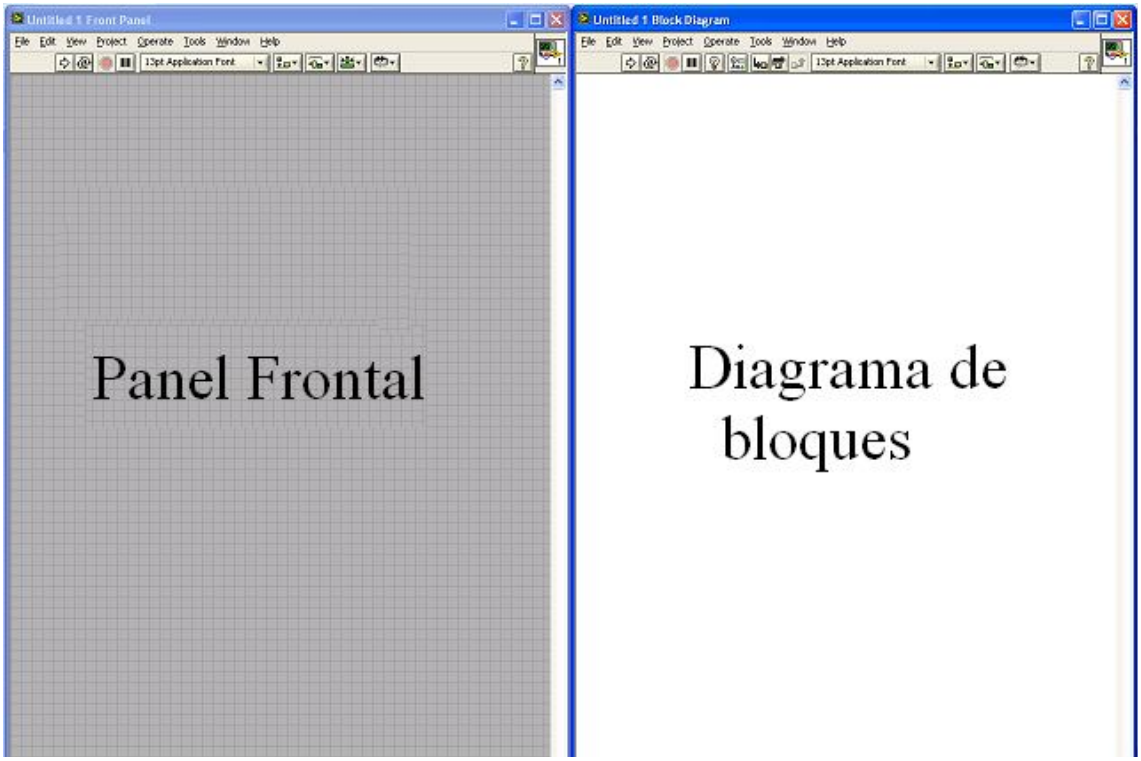


Fig. 3: *Estructura VI*

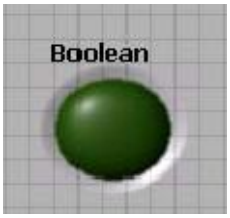
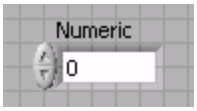
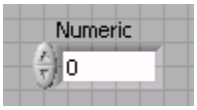
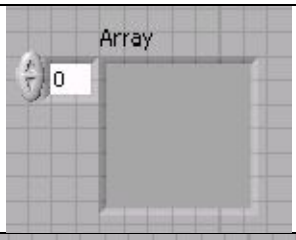
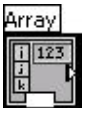
1.3.1 Panel Frontal

Lugar donde diseñamos la *interfaz* con el usuario, es decir, panel que interactúa con el usuario. Simula el panel del instrumento físico. El panel frontal puede contener botones, pulsadores o gráficos entre otros controles e indicadores, como veremos más adelante.

1.3.2 Diagrama de bloques

Lugar donde se relacionan los elementos utilizados en la interfaz mediante operaciones que determinan en sí como funciona el programa o el sistema. Aquí es donde se realizan las especificaciones funcionales. En conclusión, el diagrama de bloques es la solución gráfica al problema o necesidad que se plantea inicialmente. Los VI son jerárquicos y modulares, es decir, LabVIEW ofrece la posibilidad de dividir una aplicación en diversas tareas, las cuales se pueden separar de nuevo hasta conseguir que una aplicación compleja se convierta en una serie de subtareas mucho más sencillas. Esta aplicación permite comprobar la ejecución por separado de cada tarea y así es más asequible detectar los posibles errores de compilación de una forma más rápida, cómoda i eficaz para el usuario.

1.4 TIPOS DE DATOS

DATO		PANEL FRONTAL	DIAGRAMA DE BLOQUES
Booleanos	Enteros de 16 bits. El bit más significativo contiene el valor del booleano, es decir si el bit 15 se pone a 1 (TRUE), si se pone a 0 (FALSE)		
Numéricos racionales	Siguen el *estándar IEEE*[6]. El tamaño es de 32 bits para los de precisión simple, 64 bits para los de doble precisión y el tamaño de los extendidos depende de la plataforma con la que trabajemos.		
Numéricos enteros	Puede elegirse su tamaño (8, 16,32 o 64 bits), si se emplea un bit de signo y su representación.		
Arrays	Conjunto de datos ordenados y de un tipo determinado.		
Strings	Conjunto de datos ordenados y de un tipo		

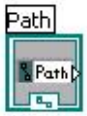
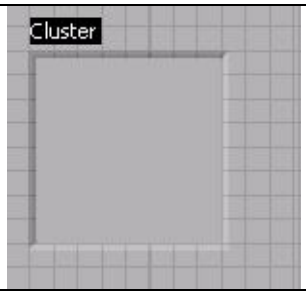
	determinado.		
Paths	Sirven para indicar rutas relativas o absolutas a directorios o fiches tanto de la máquina local como de otra red.		
Clusters	Conjunto de datos ordenados que pueden contener datos de varios tipos en su interior.		

Tabla 1: *Tipos de datos en LabVIEW*

1.4.1 Estructuras básicas

1.4.1.1 Sequence

Las estructuras tipos Sequence sirven para asignar el orden de ejecución del código que está en su interior. Se forma a través de fotogramas o, y en cada frame se sitúa la sección de código a desarrollar. La ejecución comenzará por el primer frame, cuando termine continuará con el segundo y así sucesivamente.

Existen dos tipos de estructuras Sequence: Stacked Sequence y Flat Sequence.

1.4.1.1.1 *Stacked Sequence*

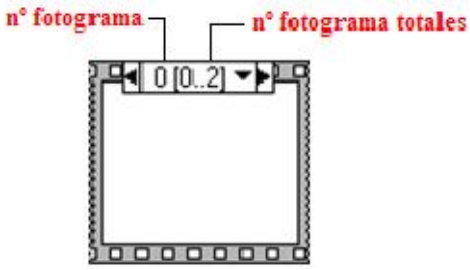


Fig. 4: *Estructura Stacked Sequence*

Dispone de un menú en la parte superior donde se indica la numeración del frame que se muestra y el número total de frames que dispone.

1.4.1.1.2 *Flat Sequence*

Funciona de igual forma que el anterior, pero es algo más visual ya que los frames se ven uno a continuación del siguiente, y el orden de ejecución va de izquierda a derecha.

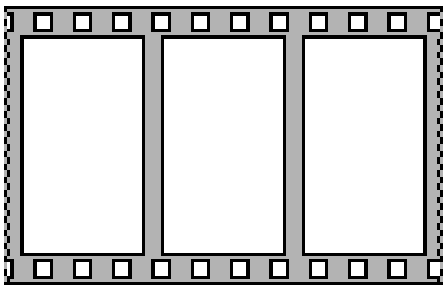


Fig. 5: *Estructura Flat Sequence*

1.4.1.2 **Case**

La estructura Case ejecuta un código u otro dependiendo de una condición. Dispone de un menú en la parte superior donde se muestra la condición para ejecutar el código del subdiagrama correspondiente.

El terminal que aparece en el lado izquierdo marcada con el símbolo "'?" es denominado selector. El valor que llega a este selector es la condición que se evalúa para seleccionar el subdiagrama a ejecutar.

Para un selector booleano solo se tendrás dos casos: verdadero o falso. Para numéricos la condición será que el dato del selector sea igual al mostrado en el menú del Case.

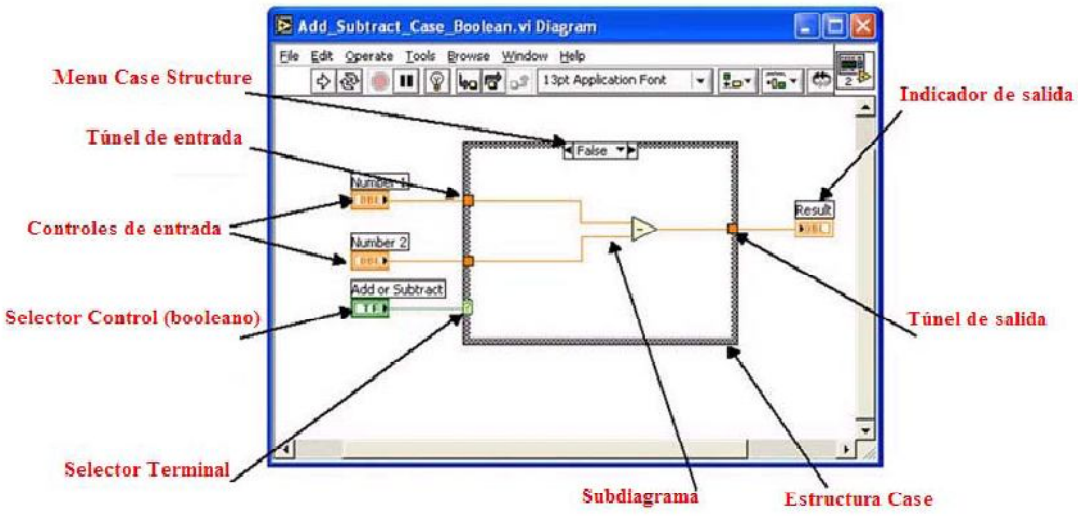


Fig. 6: *Estructura Case*

1.4.1.3 While

El bucle While repetirá el código de su interior hasta que se cumpla una condición, la cual es evaluada en cada iteración.

Aparece un terminal de iteración en forma de cuadrado azul con el símbolo “i”. El valor de este terminal es un número entero que irá aumentando una unidad por cada iteración del bucle, empezando a contar desde cero.

La condición de parada es el terminal verde de la esquina inferior derecha de la estructura. A este terminal se podrá conectar un valor booleano o un clúster de error.

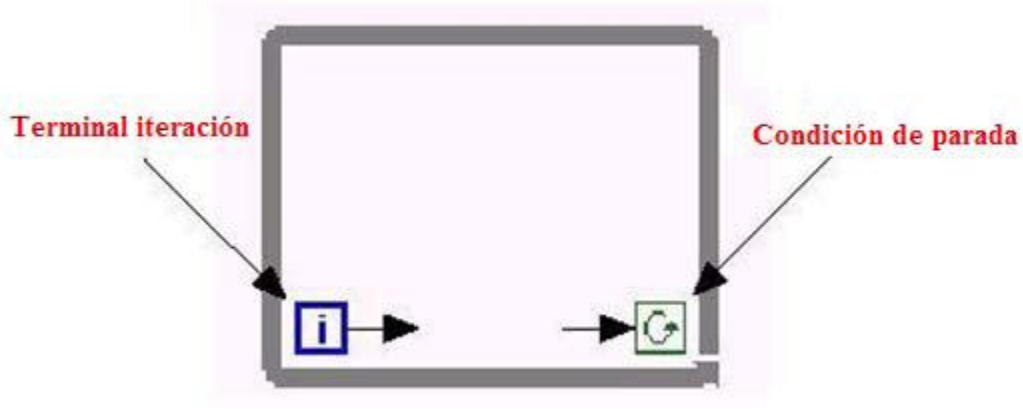


Fig. 7: *Estructura While*

1.4.1.4 Shift Register

Esta herramienta añade dos terminales a cada lado de las estructuras. Sirven para transferir un valor desde una iteración del bucle a la siguiente. Los valores se pasarán a la siguiente iteración en el terminal de la derecha y se leerán en el terminal de la izquierda.

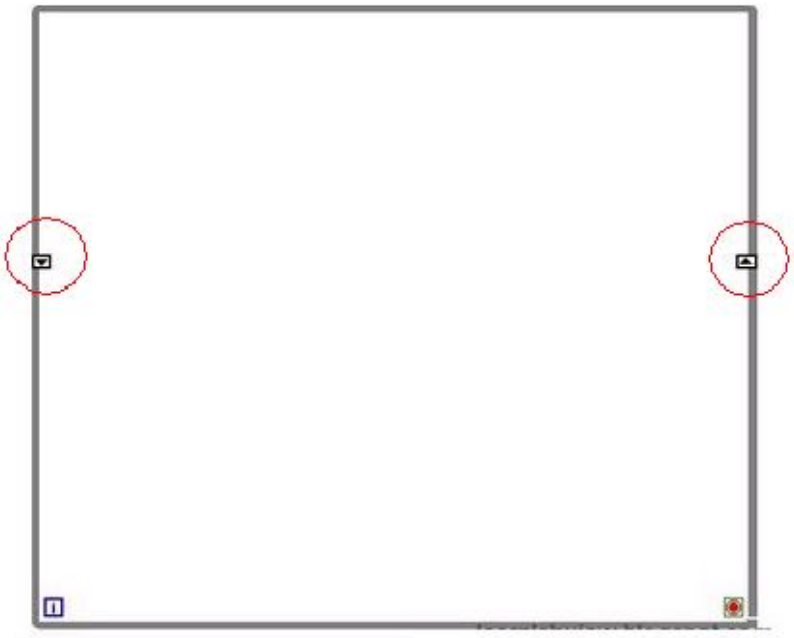


Fig. 8: *Shift Register*

1.4.1.5 **For**

El bucle For repite el código de su interior un número de veces determinado, el cual no se puede cambiar una vez que haya comenzado la ejecución del programa.
Consta de dos terminales numéricos:

- Terminal de iteración situado en el interior de la estructura y se va incrementando en una unidad por cada iteración empezando desde cero.
- Terminal de cuenta está colocado en la esquina superior izquierda de la estructura y está simbolizado con una N. En él se conectará un valor numérico que será el que fije el número de repeticiones del bucle.

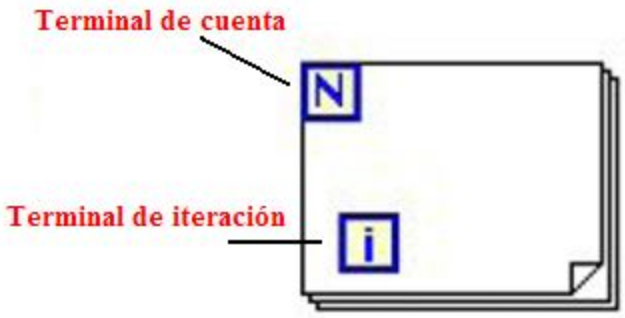


Fig. 9: *Estructura For*

1.4.1.6 **Formula node**

La estructura Formula Node no controla el flujo de ejecución, sino que evalúa una expresión matemática escrita como texto con una sintaxis parecida al lenguaje C.

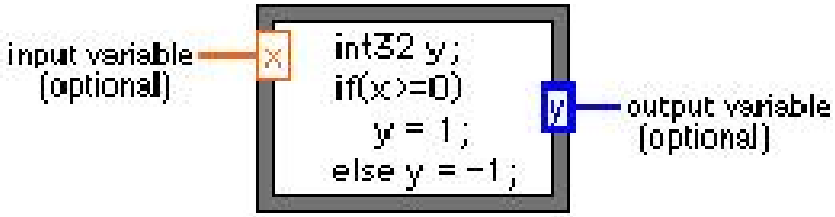


Fig. 10: *Fórmula Node*

1.5 PUERTO SERIE RS232

1.5.1 ¿Qué es?

La comunicación serie se emplea para transmitir datos entre un ordenador y un dispositivo periférico como un instrumento programable o incluso otro ordenador.

Consiste en un conector tipo DB25 de 25 pines, aunque es normal encontrar la versión de 9 pines DB9 ya que esta versión está mucha más extendida en diversos tipos de periféricos y es más económica.

Típicamente, la comunicación serial se utiliza para transmitir datos en formato ASCII. Para realizar la comunicación se utilizan 3 líneas de transmisión: (1) Tierra (o referencia), (2) Transmitir, (3) Recibir. Debido a que la *transmisión es asincrónica7*, es posible enviar datos por una línea mientras se reciben datos por otra.

1.5.2 ¿Cómo funciona?

Las señales con las que trabaja son señales digitales: +12V (0 lógico) y -12V (1 lógico) para la entrada y salida de datos y a la inversa en la señales de control. Cada pin puede ser de entrada o de salida teniendo una función específica cada uno de ellos.

Las características más importantes de la comunicación serial son la velocidad de transmisión, los bits de datos, los bits de parada, y la paridad.

TXD	Transmitir datos
RXD	Recibir datos
RTS	Solicitud de envío
DTR	Terminal de datos listo
CTS	Libre para envió
DSR	Equipo de datos listo
DCD	Detección de portadora
SG	Tierra
RI	Indicar de llamada

Tabla 2: *Pines RS232*

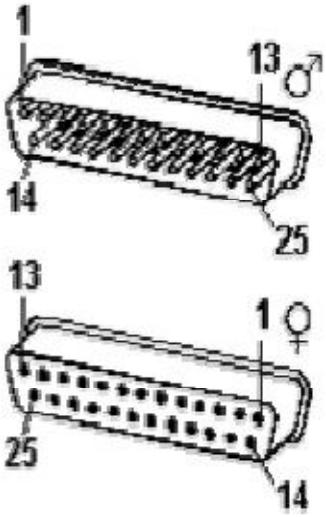
#	Pin	E/S	Función	Conector DB 25
1			Tierra de Chasis	
2	TXD	S	Transmitir Datos	
3	RXD	E	Recibir Datos	
4	RTS	S	Solicitud de Envío	
5	CTS	E	Libre para Envío	
6	DSR	E	Equipo de Datos Listo	
7	SG		Tierra de señal	
8	CD/DCD	E	Detector de Portadora	
15	TxC	S	Transmitir Reloj	
17	RxC	E	Recibir reloj	
20	DTR	S	Terminal de Datos Listo	
22	RI	S	Timbre Telefónico	
24	RTxC	S/E	Transmitir/Recibir Reloj	

Tabla 3: *RS232 de 25 pines*

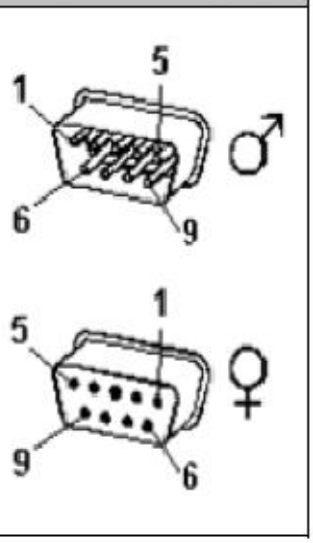
#	Pin	E/S	Función	Conector DB 9
1			Tierra de Chasis	
2	RXD	E	Recibir Datos	
3	TXD	S	Transmitir Datos	
4	DTR	S	Terminal de Datos Listo	
5	SG		Tierra de señal	
6	DSR	E	Equipo de Datos Listo	
7	RTS	S	Solicitud de Envío	
8	CTS	E	Libre para Envío	
9	RI	S	Timbre Telefónico	

Tabla 4: *RS232 de 9 pines*

Los pines que portan los datos son RXD y TXD los demás se encargan de otros trabajos: el DTR indica que el ordenador está encendido, DSR que el dispositivo conectado al puerto

está encendido, RTS que el ordenador al no estar ocupado puede recibir datos, al revés que CTS que lo que informa es que es el dispositivo el que puede recibir datos, DCD detecta que existen presencia de datos.

Fig. 11: *Asignación de pines en RS232 DB9*

1.5.3 Configuración antes de iniciar la comunicación

Antes de iniciar cualquier comunicación con el puerto serie RS232 se debe determinar el protocolo a seguir (que debe de ser el mismo para el aparato a conectar y el PC). Este paso depende del usuario ya que es él quien decide los valores de los parámetros necesarios y quien realiza la configuración en los dispositivos antes de la transmisión de datos.

1.5.3.1 Parámetros a configurar

- Nº bits: número de bits necesarios para comenzar la comunicación serie. Su valor está comprendido, normalmente entre 5 y 8. El valor automático de este parámetro que asigna LabVIEW es 8 bits.
- Paridad (dato opcional): Es una forma sencilla de verificar si hay errores en la transmisión serial. La opción de no usar paridad alguna también está disponible. Para paridad par e impar, el puerto serial fijará el bit de paridad (el último bit después de los bits de datos) a un valor para asegurarse que la transmisión tenga un número par o impar de bits en estado alto lógico (ver tabla 4)

0	Sin paridad (valor de referencia)
1	Paridad impar
2	Paridad par
3	Marcada
4	especificada

Tabla 5: *Tipos de paridad*

- Bits stop: Usado para indicar el fin de la comunicación de un solo paquete. Los valores típicos son 1, 1.5 o 2 bits (Ver tabla 13)

10	1 bits de stop
15	1.5 bits de stop
20	2 bits de stop

Tabla 6: *Bits de stop*

- Velocidad de transmisión puerto serie: Indica el número de bits por segundo que se transfieren, y se mide en baudios (*bauds*). Por ejemplo, 300 baudios representa 300 bits por segundo. Cuando se hace referencia a los ciclos de reloj se está hablando de la velocidad de transmisión.

- Protocolo de control de flujo: establece el tipo de control utilizado para transferir datos. Existen dos posibilidades de control de flujo de datos con la RS232: Una hardware mediante las líneas RTS/CTS y otro software XON/XOFF (tabla 7).

0	Ninguno	La transmisión no utiliza flujo de datos
1	XON/XOFF	(Transmisor preparado/transmisor ocupado). Protocolo de control de transmisión asíncrona para ajustar el flujo de información entre el dispositivo emisor y el receptor. Cuando el receptor no puede recibir más datos envía una señal (xoff) de corte de transmisión al emisor para que cese el envío. Cuando el receptor está listo envía una señal de continuación (xon) para solicitar más datos.
2	RTS/CTS	(Request To Send/Clear To Send) son las señales a través de las que se ejecuta el control del flujo de Hardware. Colocando la línea RTS en "ON" el ordenador señaliza al módem que está listo para recibir datos. La función de estas líneas puede cambiarse o desconectarse con el Software del ordenador o con las órdenes de mando de módem correspondientes.

Tabla 7: *Control de flujo*

Dependiendo de las características de los equipos a conectar se puede hacer un control de flujo RTS/CTS, XON/XOFF, ambos o ninguno.

1.5.4 Envío de datos

El ordenador controla el puerto serie mediante un circuito integrado específico llamado UART (Transmisor- Receptor – Asíncrono Universal).

Para controlar el puerto serie la CPU emplea *direcciones de puertos de Entradas / Salidas8* (In/Out) y *líneas de interrupción9* (IRQ). Cada usuario debe elegirlas de acuerdo a las que tenga libres o el uso que vaya a hacer de los puertos serie.

La comunicación serie siempre envía datos siguiendo una misma secuencia, en pequeños paquetes de 5, 6,7 u 8 bits y a una velocidad determinada. Una vez comenzada la transmisión de un dato los bits tienen que llegar uno detrás de otro a una velocidad constante. Por tanto, esta comunicación es idónea cuando el flujo de datos a enviar es pequeño o cuando debemos transferir datos entre largas distancias.

La mayor limitación que presenta la comunicación serie es que únicamente puede comunicar con un único dispositivo a la vez.

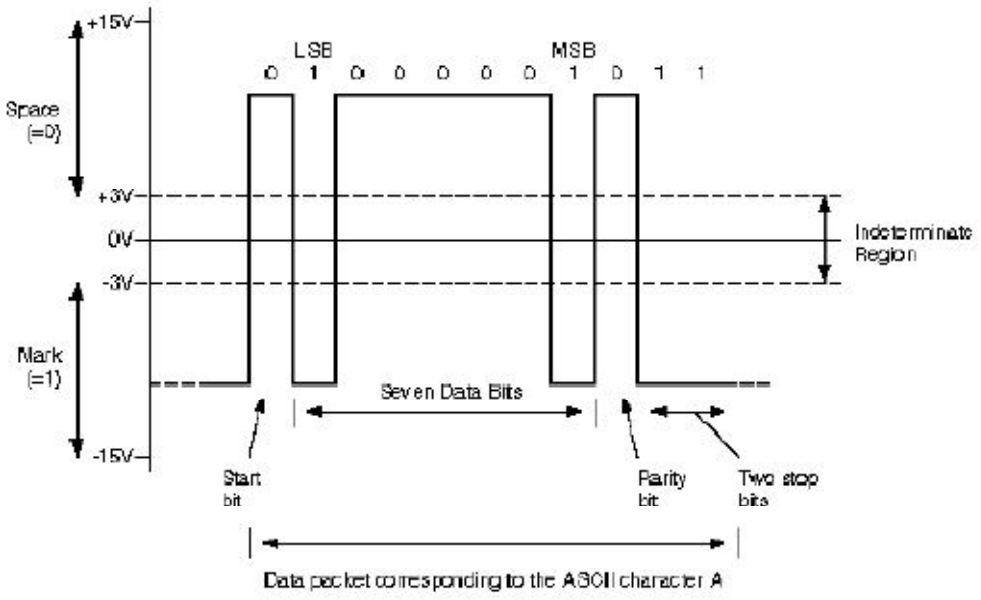


Fig. 12: *Comunicación serie RS232*

1.5.5 Comunicaciones serie en LabVIEW

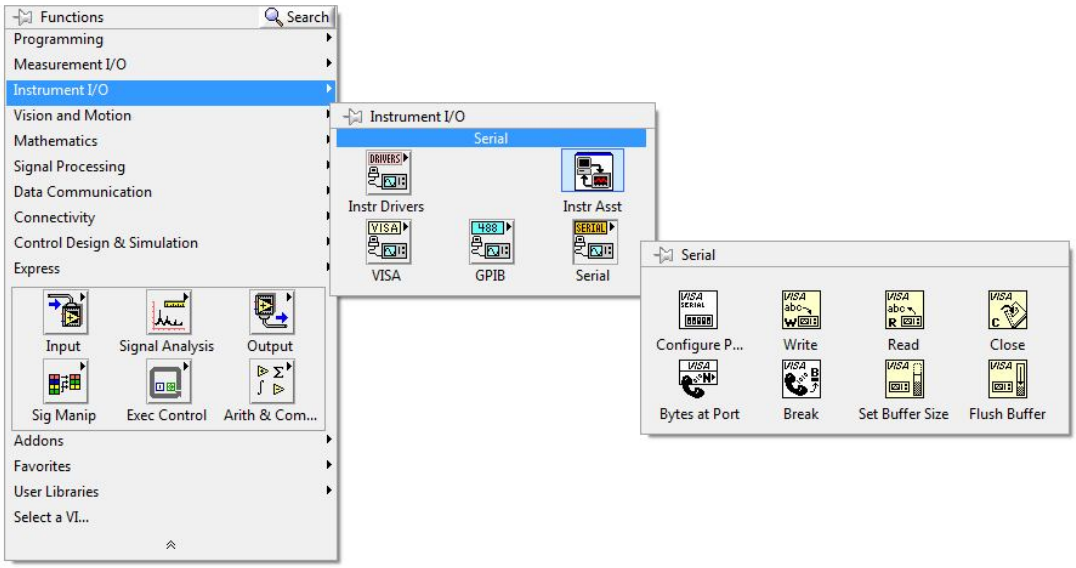


Fig. 13: *Paleta serial de LabVIEW*

1.5.5.1 VISA Configure Serial Port

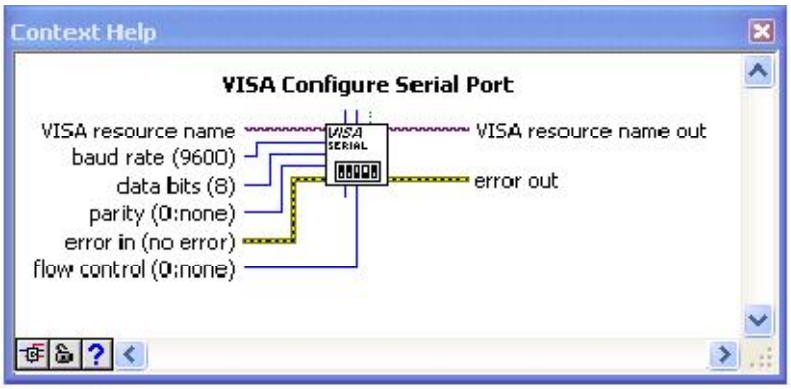


Fig. 14: *VISA configure port*

Configura el puerto serie con todos los parámetros que se han visto en el punto 1.5.3.1.

1.5.5.2 VISA write

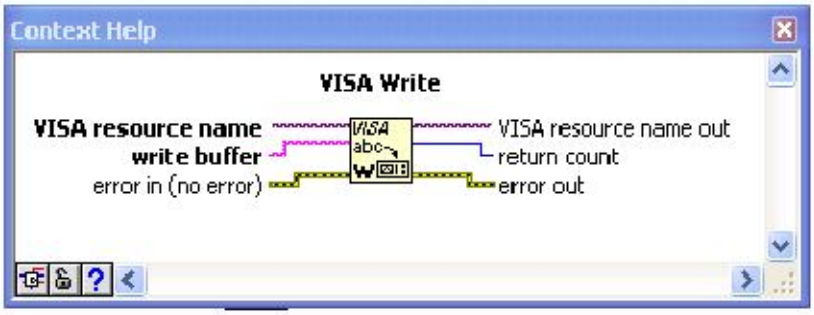


Fig. 15: *VISA Write*

Escribe datos desde el buffer hasta el otro dispositivo que participe en la comunicación.

1.5.5.3 VISA read

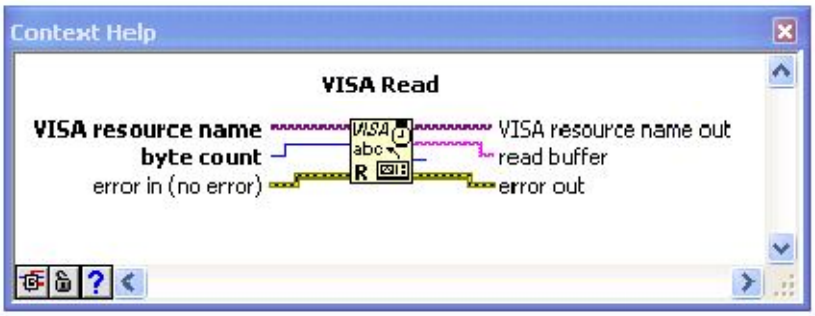


Fig. 16: *VISA Read*

Lee los datos que le envía el otro dispositivo que participa en la comunicación. Para que su funcionamiento sea correcto hay que indicarle el número de bytes que debe leer del subuffer (byte count), para darle un valor a este parámetro suele usarse la salida de la propiedad VISA Bytes at Serial Port (punto 1.5.5.4)

1.5.5.4 VISA Bytes at Serial Port

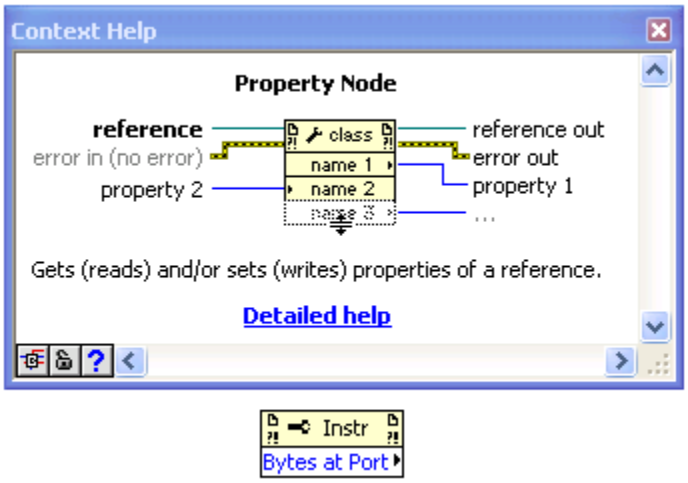


Fig. 17: *VISA Bytes at Serial Port*

Obtiene los bytes que hay en el buffer del puerto esperando a ser leídos.

1.5.6 VISA

VISA (Virtual Instrument software Architecture) es una librería desarrollada por varios fabricantes de equipos que proporcionan un estándar software para las operaciones de lectura y escritura en instrumentación.

1.6 LÁSER SICK LMS 291-S05

1.6.1 Introducción

El láser utilizado para la realización del proyecto es el láser Sick LMS 291-s05. El sistema de medida está basado en el principio de medida de "tiempo de vuelo".

Cada pulso del láser que se envía refleja en la superficie del objeto que se encuentre dentro del rango de lectura (para ver configuración del rango de lectura consultar apartado 1.6.2.1.). El tiempo que tarda el pulso desde que se emite hasta que se recibe es el que utiliza para calcular la distancia entre el objeto y el propio láser.

Los principales beneficios de este sistema de medida son:

- Los objetos son detectados independientemente del color o la superficie que tengan.
- Detección fiable de la presencia de objetos.

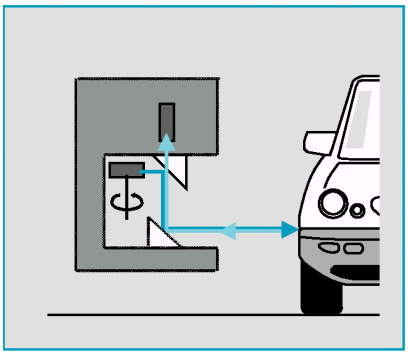


Fig. 18: *Principio de medida LMS*

1.6.2 Configuración para establecer la comunicación

Los pasos que hay que seguir referentes a la configuración para la comunicación con el láser son los siguientes.

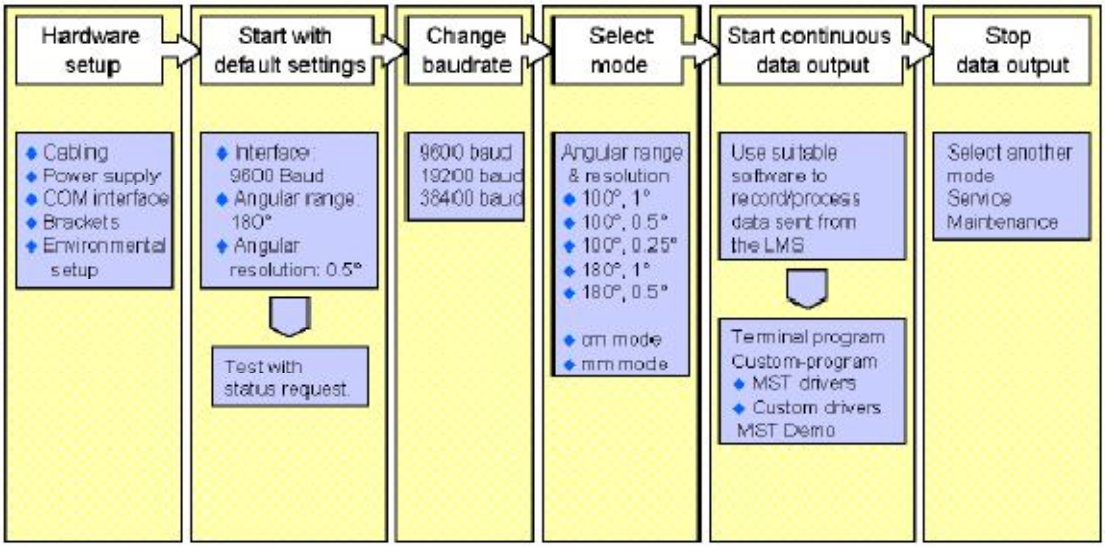


Fig. 19: *Overview Schematic for LMS communication setup*

La configuración por defecto del láser es la siguiente:

Parameter	Value		Remark
	LMS200/LMS220	LMS211/LMS221/LMS291	
Serial interface	9600 baud 8 data bits no parity 1 stop bit no flow control		Selection of RS232 or RS422 is done via a jumper in the LMS cable connector (refer to section B.3.) Commands see section 4.
Angular range	0° .. 180°	0° .. 100° (LMS211) 0° .. 180° (LMS221/291)	Commands see section 5.
Angular resolution	0.5°	0.5°	Commands see section 5.
Distance measurement mode	mm	cm	Commands see section 6.

Tabla 8: *LMS default settings*

1.6.2.1 Principales tramas de configuración

- **Trama de inicio:** Después de realizar la conexión del láser y que los indicadores se hayan apagado el láser envía una trama para indicar que está listo para empezar a trabajar. Se envía desde el láser al PC a 9600 baudios

LMS start-up message in hex LMS → PC
02 80 17 00 90 4C 4D 53 32 30 30 3B 33 30 31 30 36 33 3B 56 30 32 2E 31 30 20 10 72 D0

Tabla 9: *LMS start-up message in hex LMS-PC*

- **Cambiar el baudrate:** Por defecto tras arrancar, el láser se comunica a 9600 baudios y cada vez que se apaga se resetea a este valor. Para cambiar el sistema de medida a un nivel mayor se usan los siguientes comandos.

LMS baudrate setting	Telegram code in hex PC → LMS	Reply telegram in hex LMS → PC
9600 baud	02 00 02 00 20 42 52 08	06 02 81 03 00 A0 00 10 36 1A
19200 baud	02 00 02 00 20 41 51 08	06 02 81 03 00 A0 00 10 36 1A
38400 baud	02 00 02 00 20 40 50 08	06 02 81 03 00 A0 00 10 36 1A

Tabla 10: *Changing the baudrate*

- **Cambiar formatos de resolución:** Este aspecto de la configuración es quizá el más importante puesto que dependiendo en que formato estemos variara el número de datos que envíe el láser. Estos parámetros son los que usaré más adelante en la implementación de LabVIEW para saber cuántos bytes de datos me enviará el láser. Por ejemplo, para un rango angular de 0^0 a 100^0 con resolución de 1^0 el láser enviara 101 valores, uno por cada grado. Si la resolución es 0.5^0 mandará el doble de datos, exactamente dos por cada grado.
  Podemos ver los diferentes formatos de resolución y el número de datos que envía el láser en la Tabla 11.

Angular range	Angular resolution	Number of data values
0° .. 100°	1°	101
0° .. 100°	0.5°	201
0° .. 100°	0.25°	401
0° .. 180°	1°	181
0° .. 180°	0.5°	361

Tabla 11: *Setting different LMS resolution modes*

Cada vez que se envía la trama desde el PC para el cambio de formato el láser envía un telegrama de confirmación *ACK* al PC. En el momento que el PC reciba el telegrama correspondiente el cambio de formato se habrá realizado correctamente. Los telegramas correspondientes a cada rango son los siguientes:

LMS mode		Telegram code in hex PC → LMS	Reply telegram in hex LMS → PC
Angular range	Angular resolution		
0°..100°	1°	02 00 05 00 3B 64 00 64 00 1D 0F	06 02 81 07 00 BB 01 64 00 64 00 10 4A 3F
0°..100°	0.5°	02 00 05 00 3B 64 00 32 00 B1 59	06 02 81 07 00 BB 01 64 00 32 00 10 12 92
0°..100°	0.25°	02 00 05 00 3B 64 00 19 00 E7 72	06 02 81 07 00 BB 01 64 00 19 00 10 BE C4
0°..180°	1°	02 00 05 00 3B B4 00 64 00 97 49	06 02 81 07 00 BB 01 B4 00 64 00 10 5E B2
0°..180°	0.5°	02 00 05 00 3B B4 00 32 00 3B 1F	06 02 81 07 00 BB 01 B4 00 32 00 10 06 1F

Tabla 12: *Confirmation telegram format change*

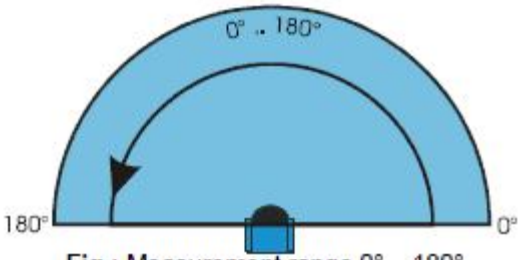


Fig.: Measurement range 0° .. 180°
(View from above, scan from right to left)

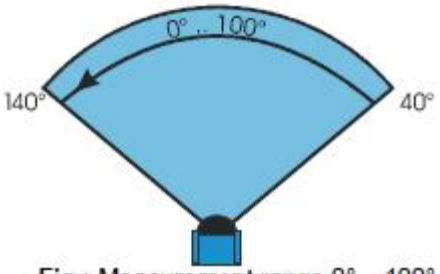


Fig.: Measurement range 0° .. 100°
(View from above, scan from right to left)

Fig. 20: *Rango angular de escaneo*

– **Configurar la unidad de medida:** Dependiendo de la aplicación, el LMS puede medir la distancia en dos unidades métricas diferentes.

Mode	Measurement/ detection range
mm mode	0 .. 8191 mm = 8.191 meters
cm mode	0 .. 8191 cm = 81.91 meters

Tabla 13: *Setting the LMS measurement mode*

Para cambiar el método de medida es necesario habilitar el LMS "settings mode"

o Cambio a mm.

LMS distance mode	Telegram code in hex PC → LMS	Reply telegram in hex LMS → PC
1. Settings mode	02 00 0A 00 20 00 53 49 43 4B 5F 4C 4D 53 BE C5	06 02 81 03 00 A0 00 10 36 1A
2. Switch to mm mode	02 00 21 00 77 00 00 00 00 00 00 00 01 00 00 02 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FC 7E	06 02 81 23 00 F7 00 00 00 46 00 00 00 01 00 00 02 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10 FA EA

Tabla 14: *Switch to mm mode*

o Cambio a cm.

LMS distance mode	Telegram code in hex PC → LMS	Reply telegram in hex LMS → PC
1. Settings mode	02 00 0A 00 20 00 53 49 43 4B 5F 4C 4D 53 BE C5	06 02 81 03 00 A0 00 10 36 1A
2. Switch to cm mode	02 00 21 00 77 00 00 00 00 00 00 00 00 00 02 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 E8 72	06 02 81 23 00 F7 00 00 00 46 00 00 00 00 00 00 02 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10 D2 F2

Tabla 15: *Switch to cm mode*

1.6.3 Control del envío de datos

Una vez que la configuración este completa, el LMS necesita un comando para empezar a enviar los datos de medida al PC. El LMS envía una cadena continua de datos al puerto serie.

LMS continuous data output	Telegram code in hex PC → LMS	Data stream in hex LMS → PC
Start	02 00 02 00 20 24 34 08	06 02 81 03 00 A0 00 10 36 1A <output string header> < LMS data >

Tabla 16: *Starting continuous data output from the LMS*

Antes de enviar algún comando para cambiar cualquier parámetro de la configuración es necesario detener el envío de datos del LMS.

LMS continuous data output	Telegram code in hex PC → LMS	Reply telegram in hex LMS → PC
Stop	02 00 02 00 20 25 35 08	06 02 81 03 00 A0 00 10 36 1A

Tabla 17: *Stopping the continuous data output*

1.6.4 Formato de la trama de datos

Las siguientes tablas muestran los detalles sobre el formato de la cadena de datos que envía el LMS.

STX	ADR	LenL Low byte	LenH High byte	CMD	Data LenL	DataL enH	Data 0° Low byte	Data 0° High byte	Data 1° Low byte	Data 1° High byte	..... (more data)	Status	CRC Low byte	CRC High byte

Tabla 18: *Output data string in byte units*

Designation	Data size (number of bits)	Remarks
STX	8	Start byte (STX = 02 hex)
ADR	8	Address of the subscriber (in this case the PC) addressed. Typically, the value is (81 hex).
Len	16	Length of the total LMS output data string. Number of following output data string bytes excluding the checksum (CRC = 2 bytes)
CMD	8	Command byte, in this case (B0 hex), which is the command for continuous data output.
DataLen	16	Number of measurement data bytes (depending on measurement mode, refer to section C.8.)
Data ...	n x 16	Measurement data values (2 bytes each) according to the measurement mode settings. Please refer to the note below)*
Status	8	Status byte Indication of system error, pollution etc. Refer to telegram listing for exact information.
CRC	16	CRC Checksum

Tabla 19: *Designation and description of output data string elements*

Para conseguir una medición de distancia correcta es necesario utilizar una máscara para eliminar los tres últimos bits de cada dato como muestra la tabla 20. Cada dato tiene un tamaño de dos bytes.

LMS measurement data value: Bits [0..12] max. value 8191															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Tabla 20: *Formato de dato en bits*

1.6.4.1 Información del byte de Status

Durante el envío continuo de datos, el LMS envía un byte de status al final de cada trama de datos (3er último byte de la trama, véase tabla 19). La información contenida en dicho byte es la siguiente:

Bit 0	Bit 1	Bit 2	
0	0	0	No error (this is the OK state)
1	0	0	Info
0	1	0	Warning
1	1	0	Error
0	0	1	Fatal error

Bit 6	Implausible measured value
Bit 7	Pollution

Tabla 21: *Status Byte Information*

1.7 ADAPTADOR REDCOM USB-COM ADAPTER

1.7.1 Introducción

El adaptador USB-RS232 está diseñado para servir de nexo entre el dispositivo con conexión RS232 y el puerto USB del PC de una manera fácil y sencilla.

Lo he tenido que utilizar puesto que el ordenador con el que he realizado el proyecto no cuenta con puerto RS232. Es mi propio portátil y las ventajas de utilizarlo para realizar el proyecto son considerables.

Fig. 21: *Gama adaptadores Red COM USB-RS232*

1.7.2 Modo de empleo

El sistema operativo que he utilizado es Windows 7, con lo que no he tenido que instalar ningún software adicional para utilizarlo, sólo con conectarlo al puerto USB Windows se encarga de la instalación automática de todos sus componentes con la posibilidad de utilizarlo al momento.

El adaptador se alimenta automáticamente al conectarlo al puerto USB con lo que no es necesaria ninguna fuente de alimentación externa.

II. ANÁLISIS EXPERIMENTAL

2.1 HARDWARE NECESARIO

- PC con puerto USB. Para llevar a cabo este proyecto he utilizado mi propio portátil (Acer Aspire 5920G) por la comodidad de poder trabajar en cualquier lugar y tener acceso siempre al programa en el que tengo realizada la implementación, además de manuales, datos y demás información requerida para la realización del proyecto. Gracias a ello puedo estar en estos momentos redactando en Holanda sin necesidad de ir a la universidad para hacerlo.
- Red COM USB-COM Adapter: El principal inconveniente de utilizar mi propio PC es que no cuenta con puerto serial del tipo RS232. Con lo que he tenido que utilizar un adaptador RS232-USB del tipo Meilhaus Electronic. Aunque haya utilizado dicho conversor la comunicación entre el PC y el láser sigue los patrones de la comunicación RS232, el adaptador sólo sirve de nexo entre PC y láser (apartado 1.7.)
- Láser Sick LMS 291-s05, véase apartado 1.6 para más información.

2.2 IMPLEMENTACIÓN

En el siguiente capítulo se hace referencia a la forma en que se diseñó la configuración, control y la muestra de los datos recibidos del láser Sick LMS 291-s05 utilizando el software LabVIEW. Se explica paso a paso las etapas realizadas.

2.2.1 Configuración

En el apartado de configuración se describen los pasos para comunicar el LMS desde el puerto serial de la computadora (COMX) utilizando los drivers de NI VISA para LabVIEW. El LMS contiene una serie de parámetros que pueden ser controlados a través de su entrada serial, (apartado 1.6)

2.2.1.1 Lectura del puerto serial

Para tener acceso al puerto serial usando LabVIEW se debe iniciar una sesión VISA. La configuración del tipo de comunicación serial se hace con “VISA configure serial port”, que se puede encontrar en Functions >> Instrument I/O >> Serial >> VISA configure serial port.

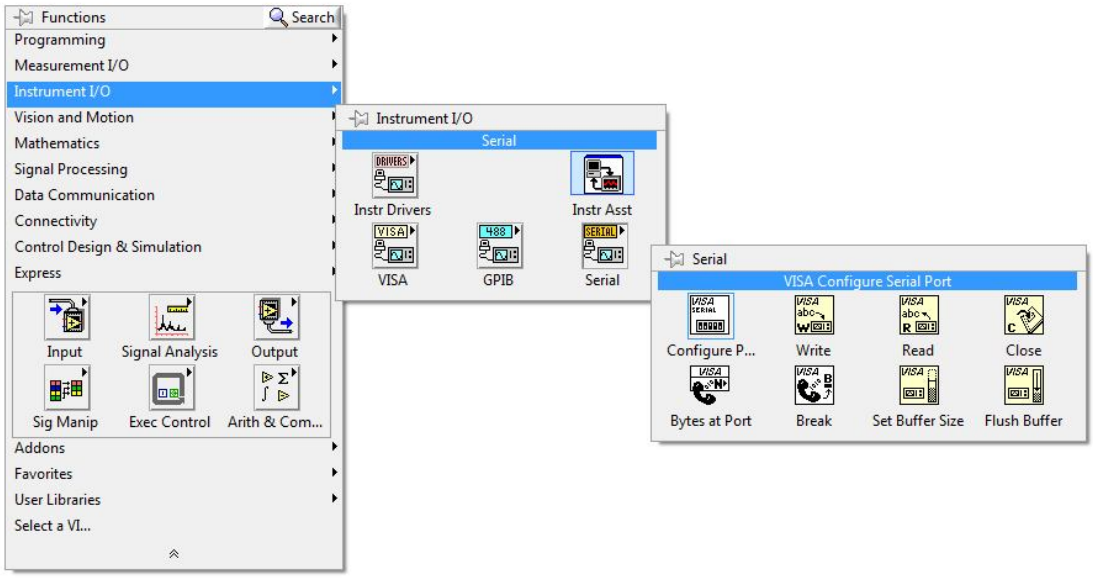


Fig. 22: *Paleta de funciones de VISA*

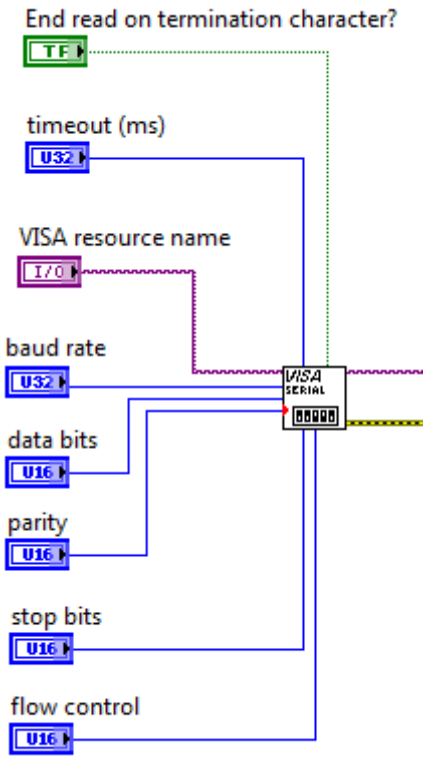


Fig. 23: *VISA Configure Serial Port*

Una vez inicializada la sesión VISA, se procede a configurar la lectura. Para lo cual se utiliza "VISA Read"

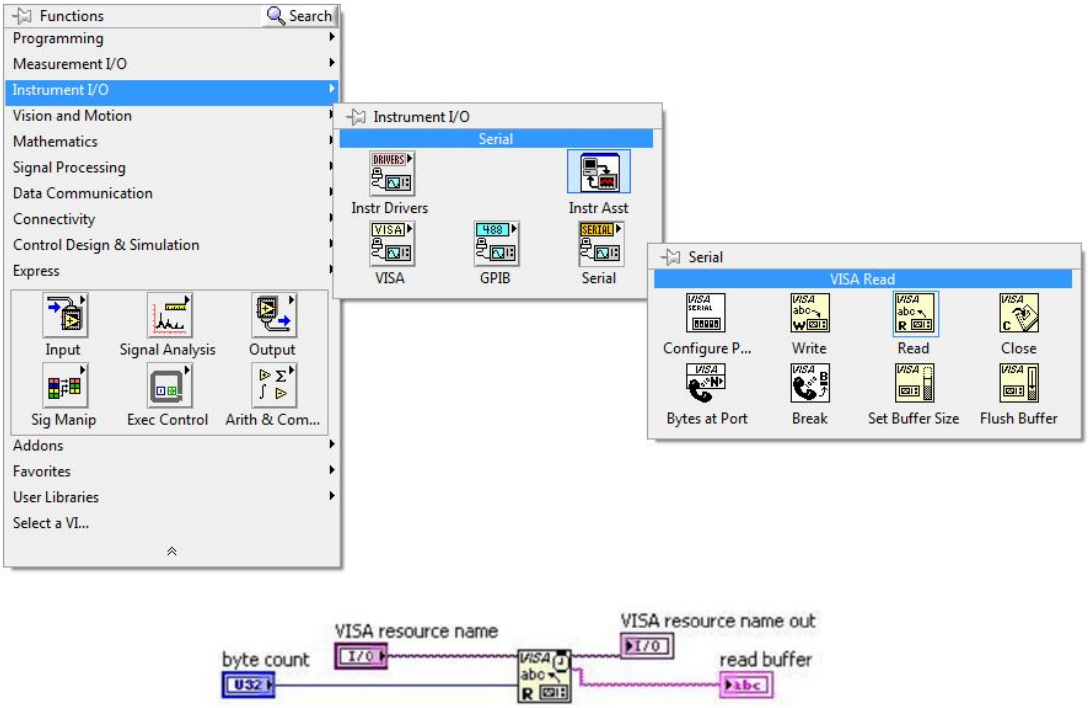


Fig. 24: *VISA Read*

El nodo de "byte count" recibe el tamaño de buffer que se escribió en el puerto. Para identificarlo, se coloca un Property Node ubicado en Functions >> Programming >> Property Node. Su nodo de referencia se conecta a la sesión VISA creada y luego, en property node se da un clic para seleccionar Serial Settings >> Number of bytes at serial port.

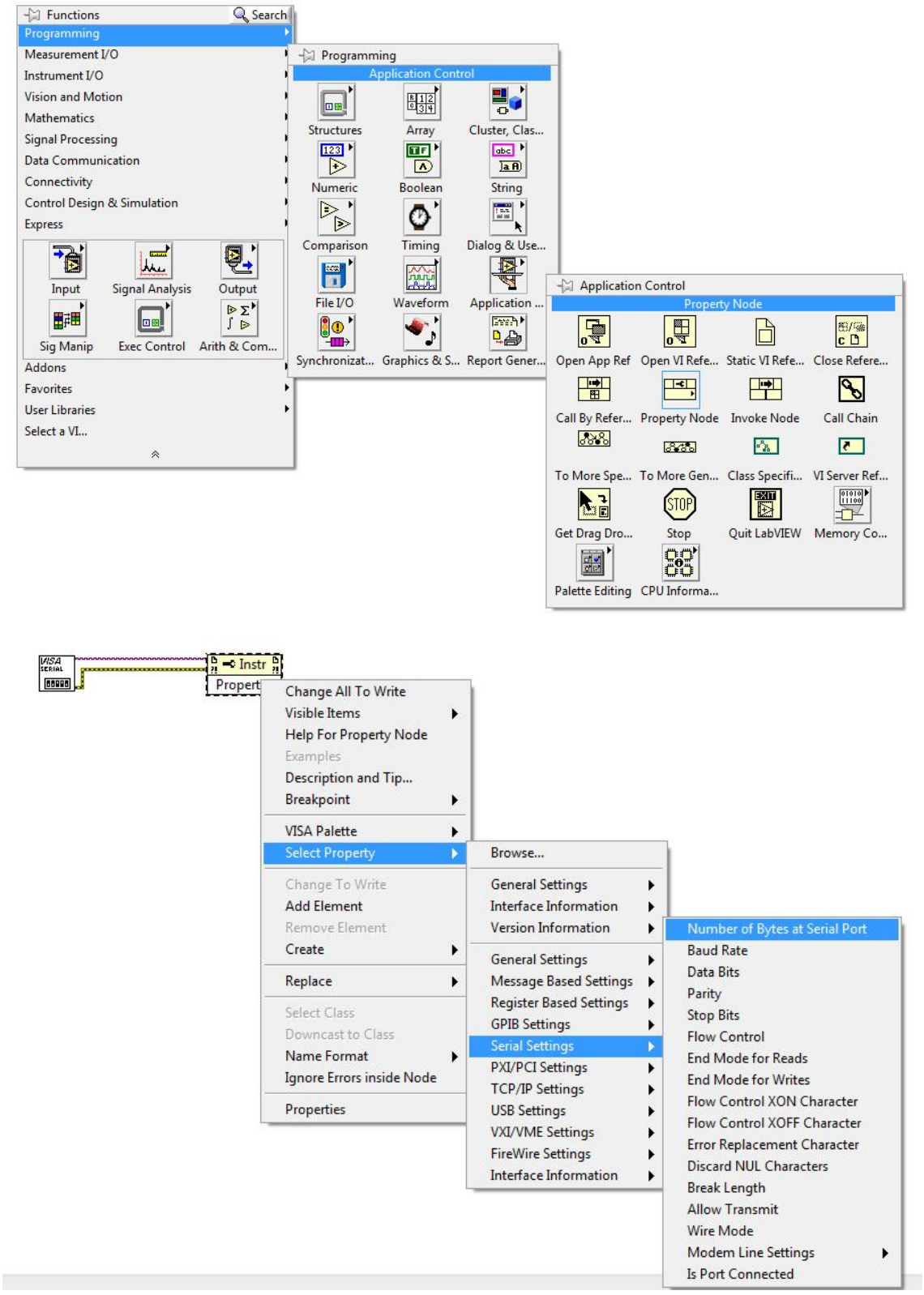


Fig. 25: *Configurando el Property Node para contar el número de bytes recibidos*

Por último se debe cerrar la sesión VISA para liberar el puerto y poderle dar otra función. Esto se logra con “VISA close” en Functions >> Instrument I/O >> Serial >> Visa Close, y, como buena costumbre de programación se coloca un controlador de errores.

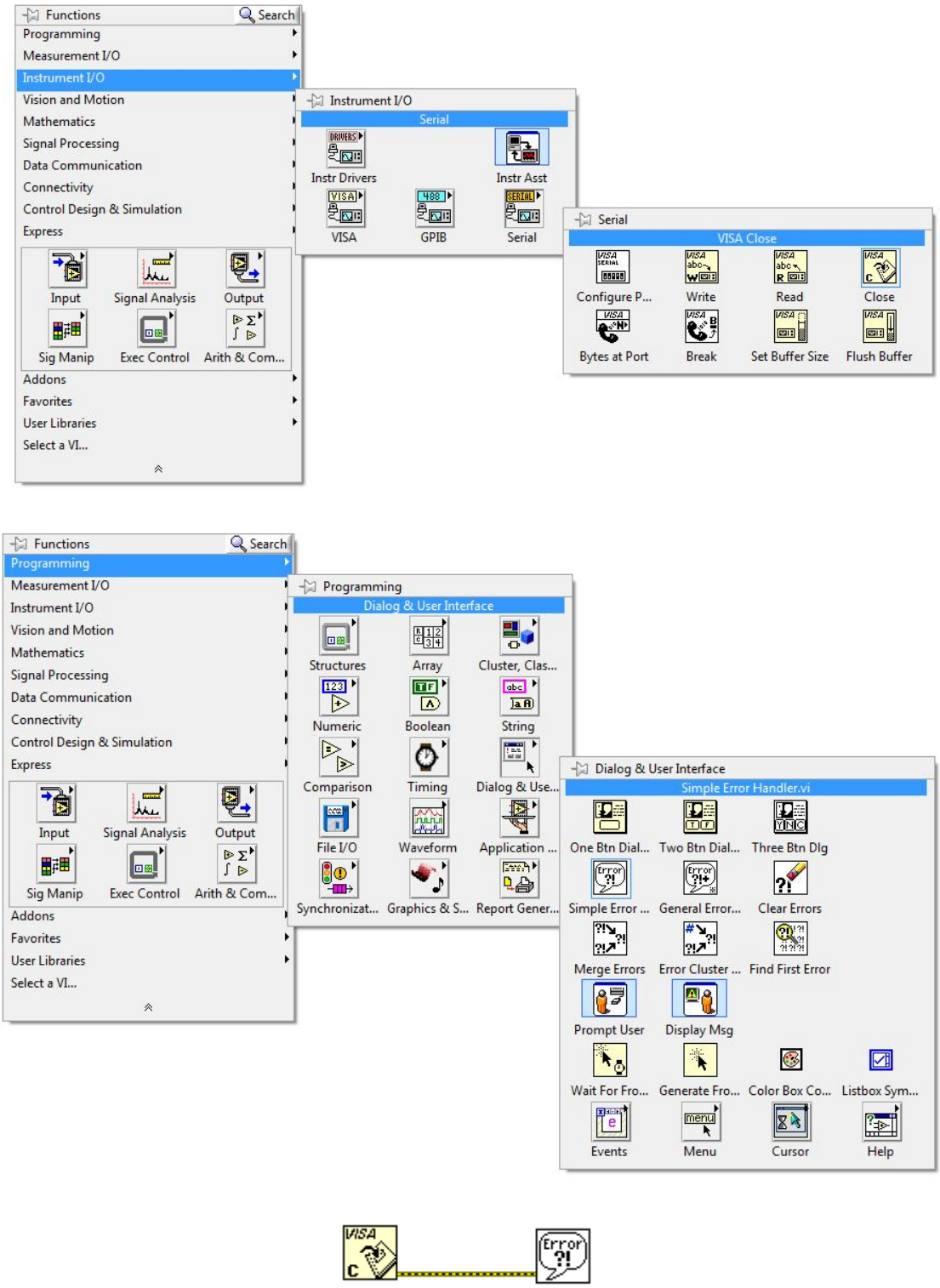


Fig. 26: *Simple Error Handler*

El VI para leer puede ser el siguiente:

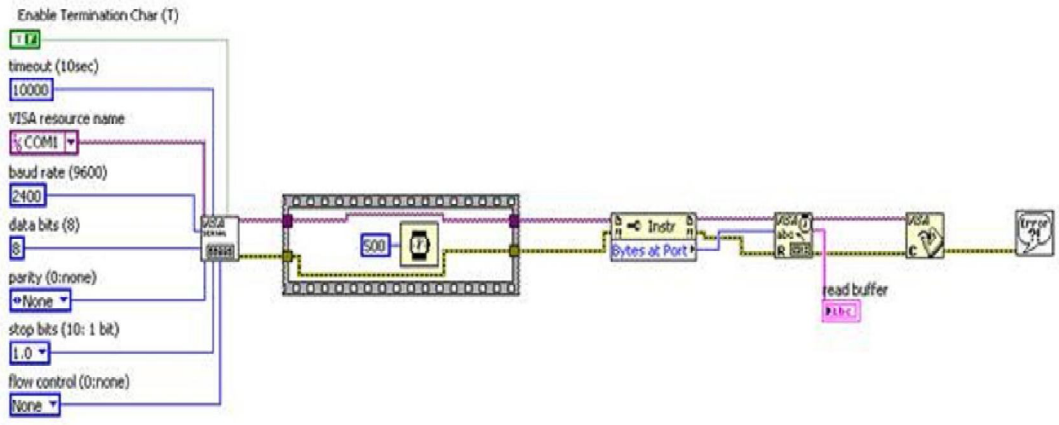


Fig. 27: *Leyendo el Puerto serial con una sesión VISA de LabVIEW*

Notas sobre la lectura en puerto serial:

Es importante sincronizar la velocidad de transferencia de datos del láser con el terminal de "Baud rate" del Visa Configure serial port, de tal manera que sea la misma para ambos (por defecto son 9600 baudios). La estructura "stacked sequence" con la función wait es tan sólo una espera programada para la lectura. Lo único presente en el Panel de control es el indicador "read buffer" donde se escriben los datos adquiridos. También es importante seleccionar el puerto COM adecuado. Al conectar el adaptador al puerto USB del ordenador lo reconoce automáticamente.

2.2.1.2 **Escritura en el puerto serial**

Escribir en el puerto serial usando LabVIEW es más sencillo, y los pasos se enuncian a continuación.

Primero, se inicializa una sesión VISA de la misma manera que se hizo al leer el puerto, con un "VISA Configure Serial Port". Luego, se coloca un "VISA Write" que se puede encontrar en Functions >> Programming >> Instrument I/O >> VISA Write.

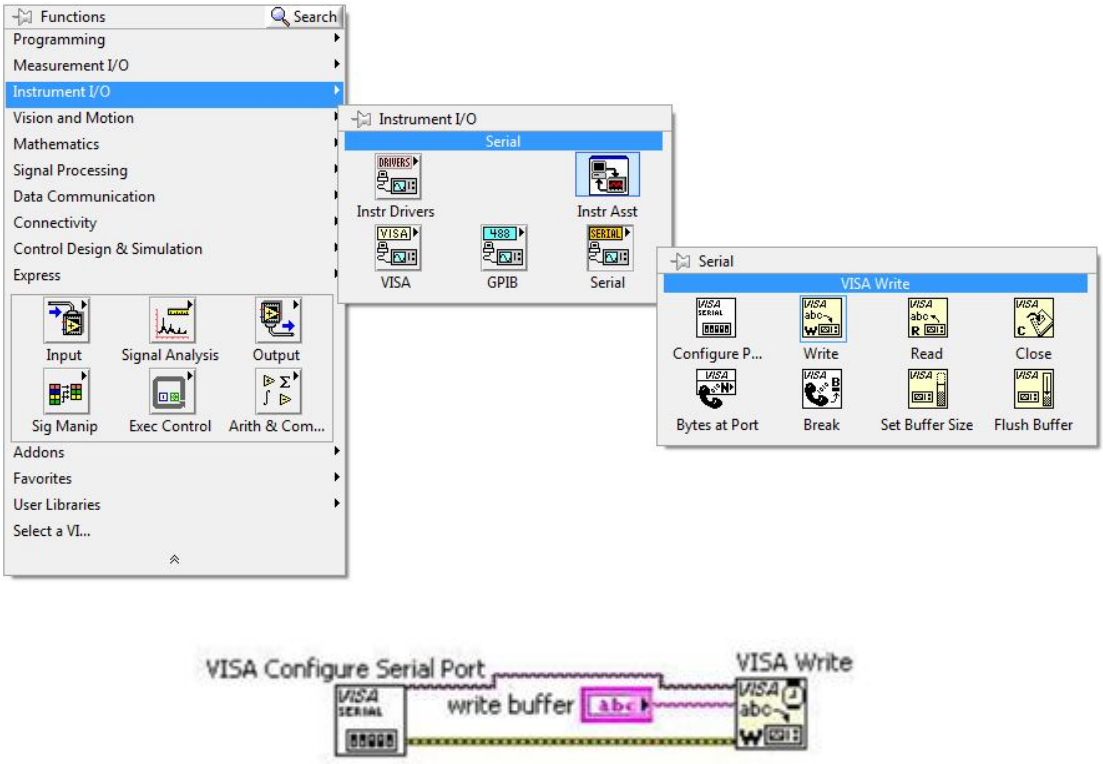


Fig. 28: *VISA Write*

Por último se cierra la sesión VISA con un “VISA close” y un “Simple Error Handler. El VI de escritura en puerto serial puede quedar como se muestra a continuación.

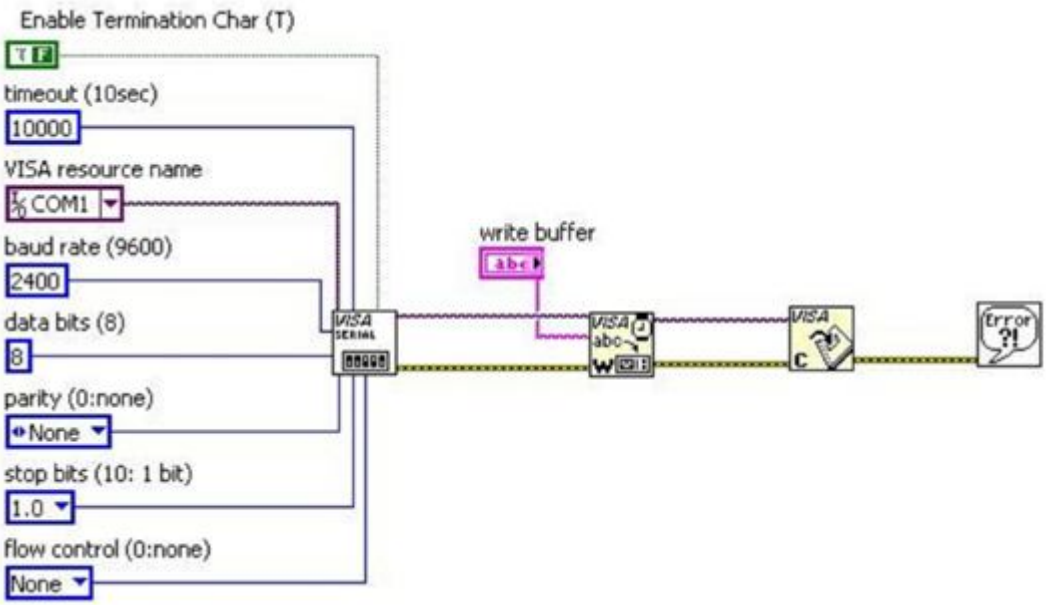


Fig. 29: *Escribir en el Puerto serial con una sesión VISA en LabVIEW.*

Al igual que en la lectura del puerto serial, la velocidad de transferencia es primordial.

En ambos casos, es importante sincronizar las tareas de escritura y lectura respectivamente, de tal manera que el láser o la computadora estén listos para enviar o recibir un dato en el puerto serial. Para ello he programado un loop mediante el cual puedo controlar la lectura o escritura del puerto serie.

2.2.1.3 Determinando si la conexión COM seleccionada es la correcta

Aunque el sistema detecta automáticamente si se ha conectado un nuevo dispositivo al puerto USB del ordenador, conviene comprobar que el puerto USB seleccionado es el adecuado.

La mejor manera de comprobarlo que encontré es la siguiente:

En la pantalla de inicio de LabVIEW abrí el buscador de ejemplos "Find examples". Luego busqué la carpeta "Hardware and Input and Output" y posteriormente seleccioné la de "serial". Abrí el VI "Basic Serial Write and Read.vi". Hay que asegurarse que el láser esté conectado al puerto USB y se escribe texto en la pantalla de *write* del VI (la superior). Cuando se ejecuta el VI, el mismo texto debe aparecer escrito en la ventana *Read*. Si no es así, hay que seleccionar otro puerto COM.

2.2.1.4 Conclusión

Escribir o leer en puerto serial utilizando LabVIEW se logra con una sesión VISA. Se configuran las características de la comunicación con un "VISA Serial Port Configuration" para que concuerden el láser y la computadora. La lectura se hace con "VISA Read" y un "Property Node" para leer la cantidad de datos recibidos. La escritura se hace con "VISA Write" únicamente. En ambos casos se cierra la sesión con "VISA Close" y "Simple Error Handler".

2.2.2 Control

En este apartado quiero explicar los parámetros que he creado para poder controlar el momento de lectura, escritura o la posibilidad de parada suave de la ejecución del programa.

2.2.2.1 Lectura

Para controlar la lectura del puerto serie he introducido todo el código referente a la lectura dentro de un bucle de tipo Case Structure en el que la condición para que se ejecute sea que el botón Leer del Front Panel esté activo.

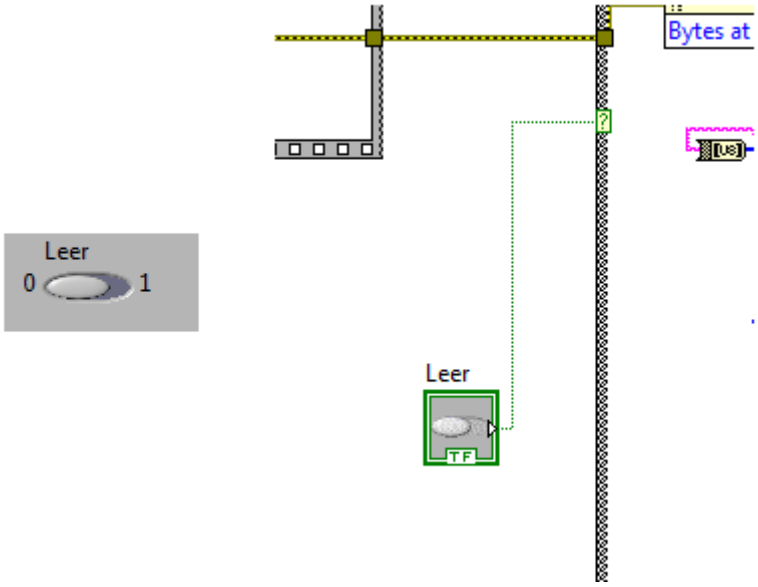


Fig. 30: *Control de lectura*

2.2.2.2 **Escritura**

De igual forma para controlar el proceso de escritura en el puerto serie he metido todos los componentes referentes a la escritura dentro de un bucle Case Structure controlado por el Switch Escribir.

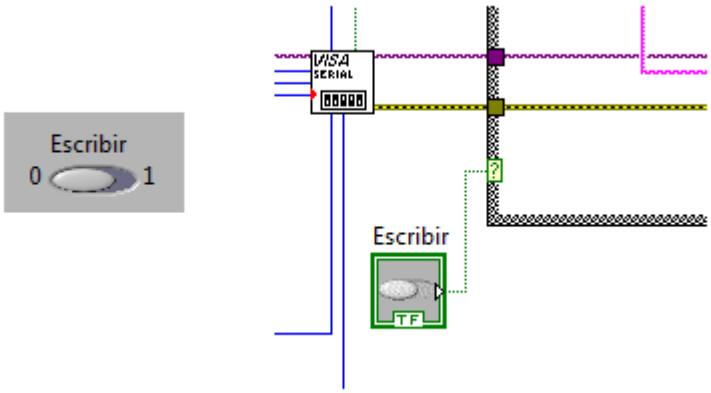


Fig. 31: *Control de escritura*

2.2.2.3 **Parada suave**

La mejor solución para evitar usar el botón de Abort Execution, que provocaría una parada brusca en la ejecución del programa, es introducir todo el código del programa dentro de un bucle tipo while, en el que se esté ejecutando hasta que se pulse el botón de Stop como se muestra en la figura 32.

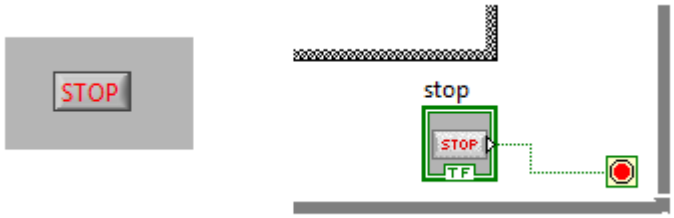


Fig. 32: *Control de parada de ejecución*

2.2.3 Datos

2.2.3.1 **Write**

En este apartado se muestran los objetos necesarios para la escritura en el puerto serie y su configuración para su correcta visualización. La selección del String Control debe realizarse desde el Panel Frontal.

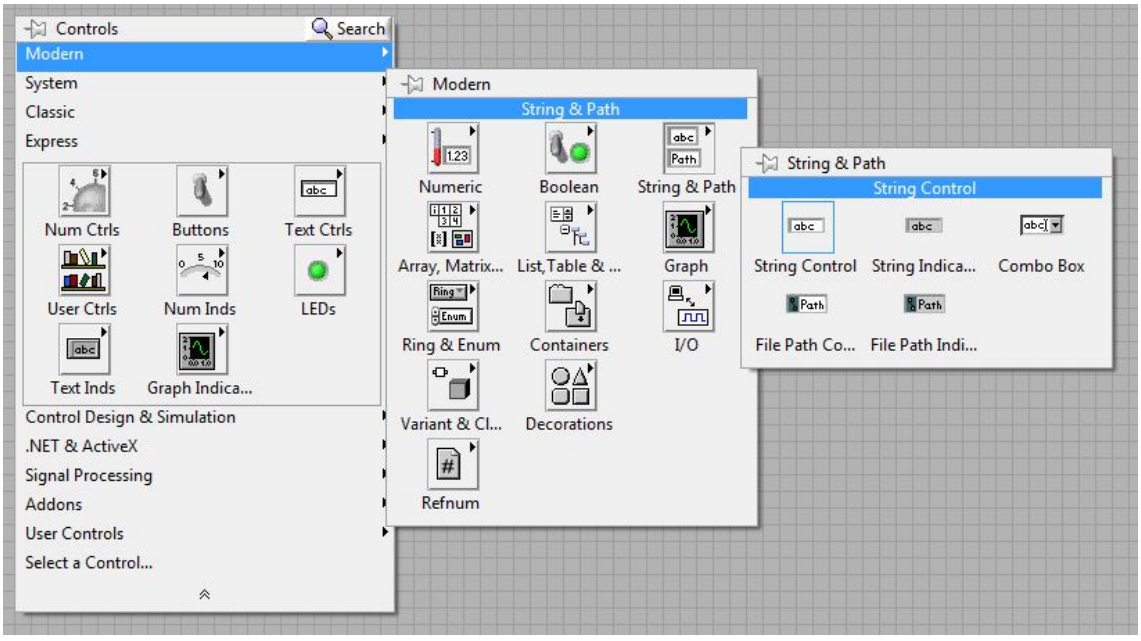


Fig. 33: *String Control*

La comunicación con el láser por el puerto serie se realiza en código ASCII, con lo que es necesario cambiar el modo de visualización a hexadecimal.

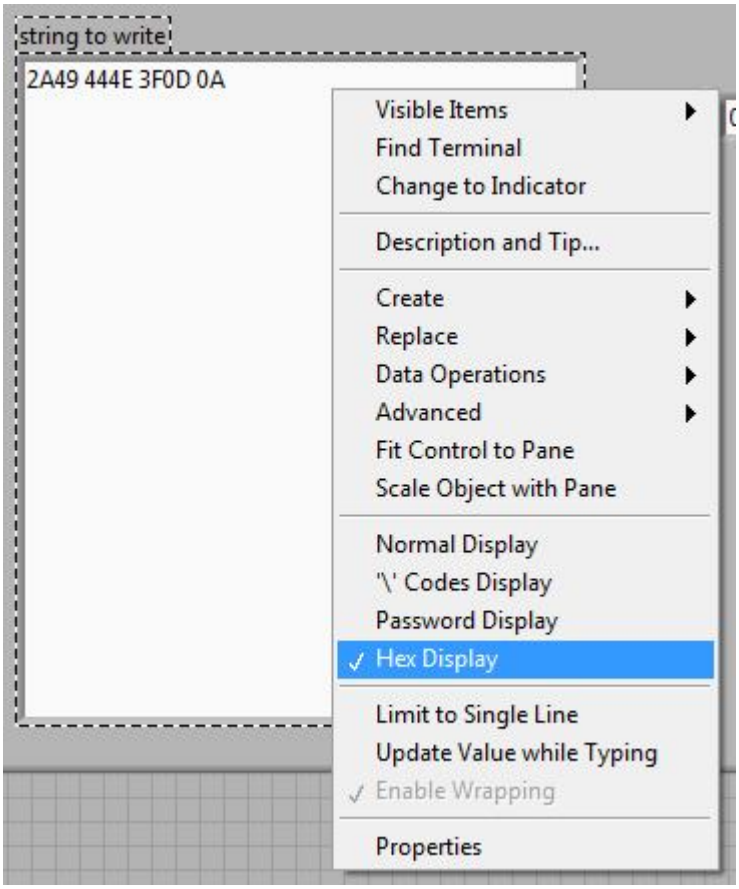


Fig. 34: *Visualización hexadecimal del String Control*

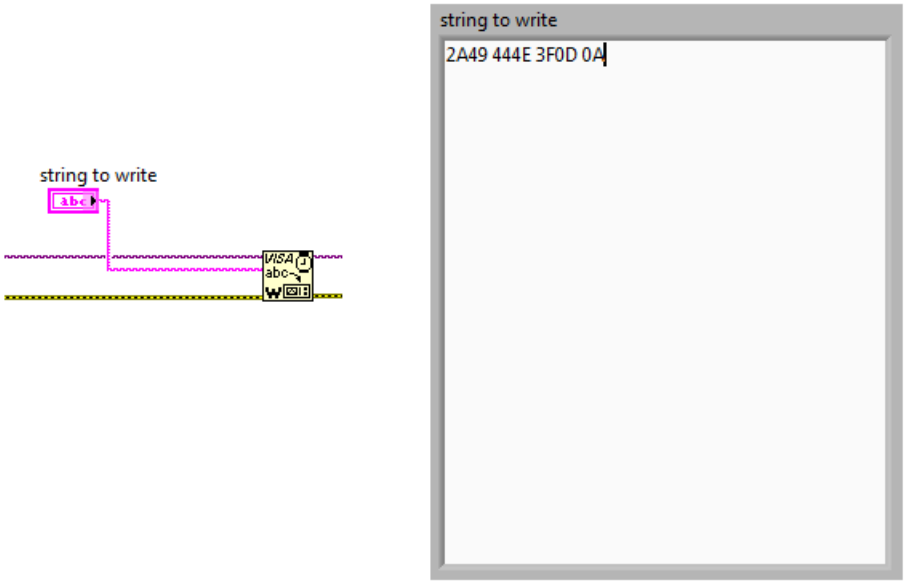


Fig. 35: *String de escritura en el puerto serie*

En el caso de no estar seleccionado el switch de escritura no entra dentro del structure de escritura.

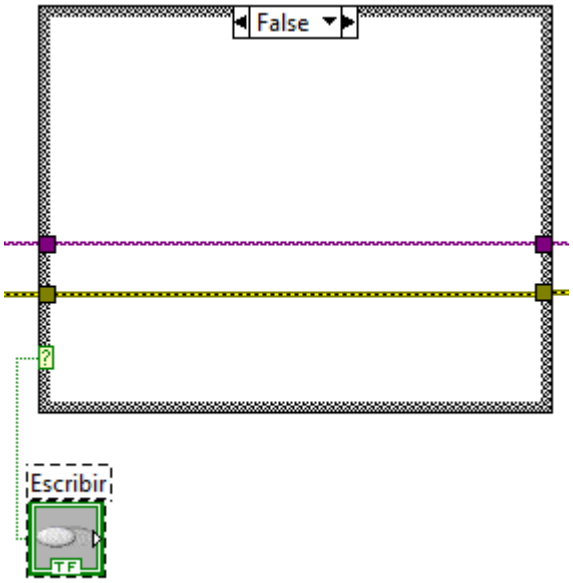


Fig. 36: *Caso de no estar seleccionado el botón Escribir.*

Nota sobre escritura en el puerto serie:

Aunque se cambie la configuración de visualización a hexadecimal, la comunicación entre PC y LMS sigue siendo en ASCII, sólo cambia la forma en el que el programa muestra la información y hace la conversión de manera automática. Para solucionar este problema he metido cada dato del string a un array en el que realmente los datos se guardan como hexadecimal.

2.2.3.2 **Read**

El proceso mediante el cual se leen los datos del puerto serie y se procesan para mostrar el resultado de las distancias en pantalla es el siguiente.

Nota sobre escritura y lectura en el puerto serie:

Aunque se cambie la configuración de visualización a hexadecimal, la comunicación entre PC y LMS sigue siendo en ASCII, sólo cambia la forma en el que el programa muestra la información y hace la conversión de manera automática. Para solucionar este problema he

metido cada dato del string a un array en el que realmente los datos se guardan como hexadecimal.

Por lo tanto, inicialmente convierto la cadena de datos a un array llamado Read Array, mediante el objeto String to Byte Array.

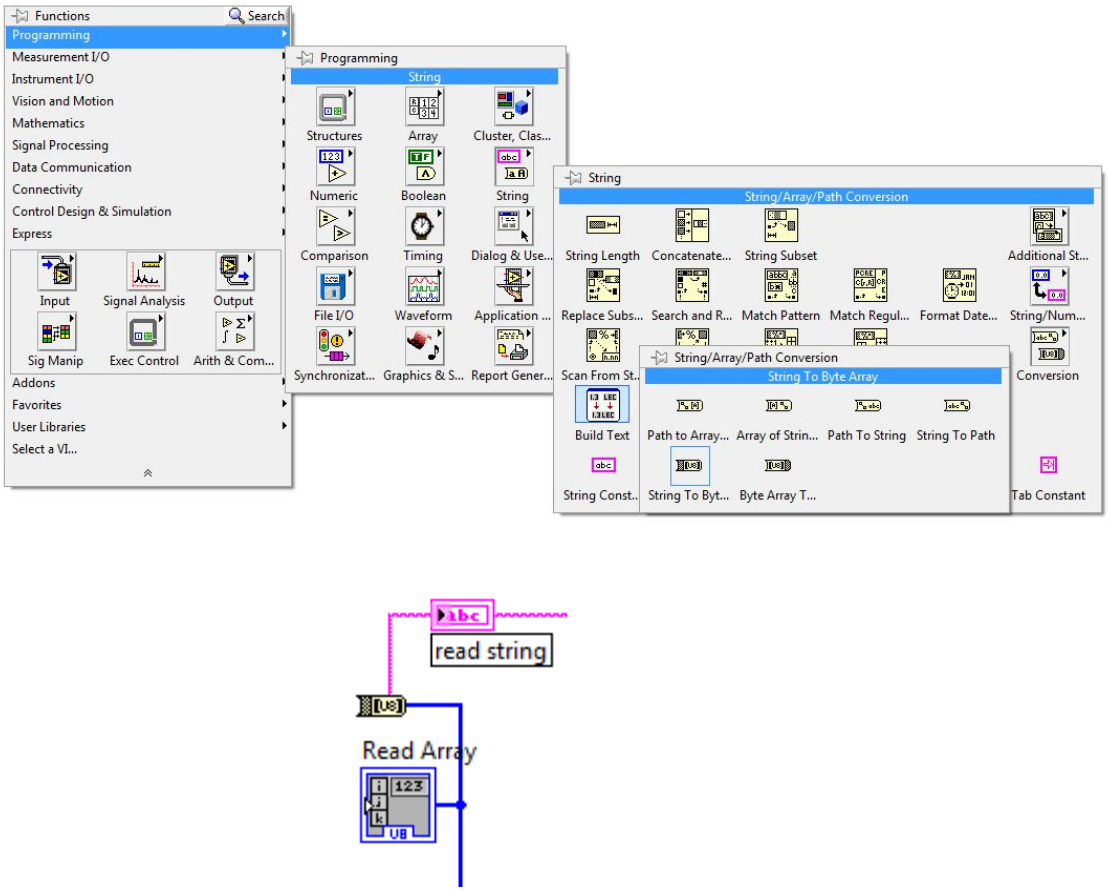


Fig. 37: *String to Byte Array*

Lo que me permite tratar cada dato recibido individualmente para poder aislar los bytes que me interesan de los que no, y dentro de los que interesan separar los referentes a la configuración de los datos. El array debe estar compuesto por unsigned bytes, para ello cuando se crea el dato numérico, dando botón derecho sobre él, dentro del desplegable representation seleccionamos unsigned byte.

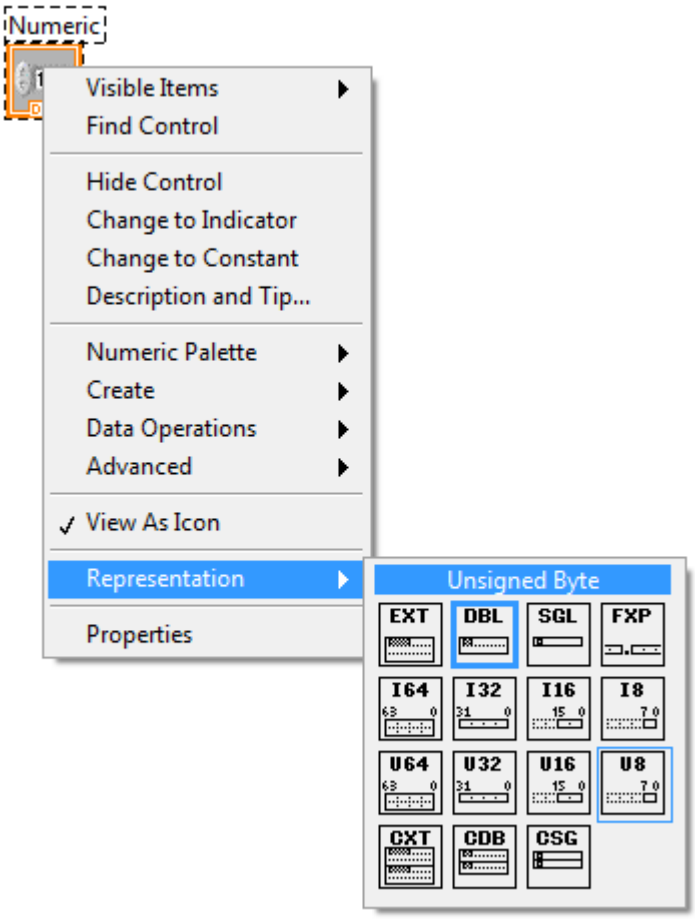


Fig. 38: *Cambio para representación Unsigned Byte*

Sólo queda cambiar el formato a hexadecimal para que muestre los datos como nos interesa verlos. Para ello, dar botón derecho y seleccionar properties. Dentro del cuadro de texto que aparece en la ficha Display Format seleccionar hexadecimal.

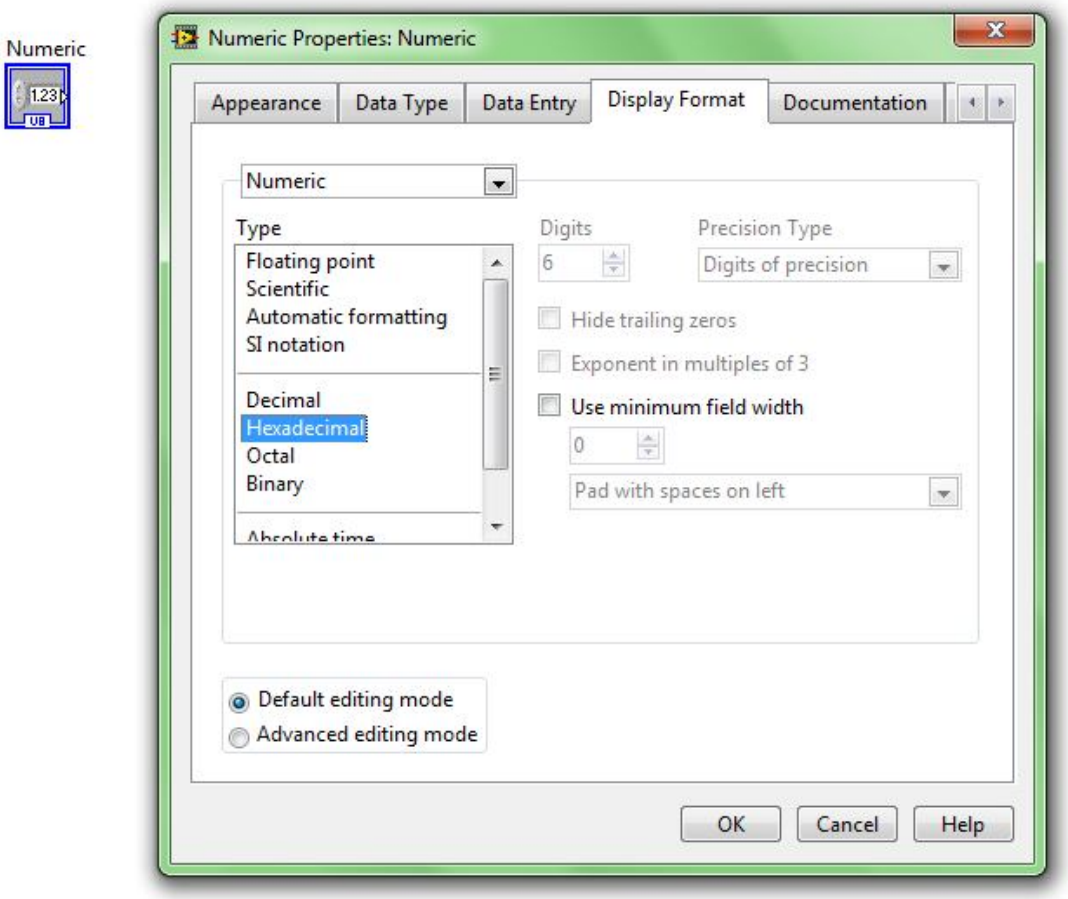


Fig. 39: *Cambio de formato a hexadecimal*

Ya sólo queda crear el array de datos arrastrando el objeto de unsigned byte dentro del objeto array en el panel frontal, para que el resultado sea un array de unsigned bytes con representación hexadecimal.

Fig. 40: *Creación array de datos deseado*

El siguiente paso es hacer un bucle for, mediante el cual podamos acceder a cada dato del array para su posterior tratamiento y que se ejecute tantas veces como bytes hayamos recibido en el puerto serie.

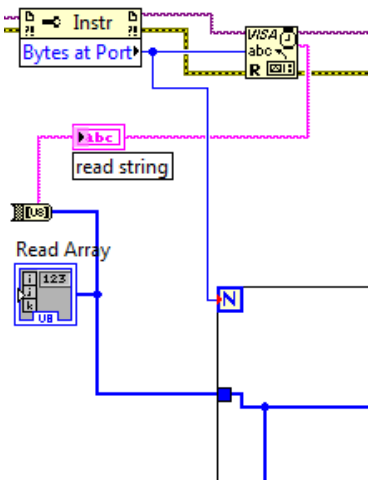


Fig. 41: *Bucle for Read*

El siguiente paso es coger sub-arrays de seis elementos, mediante el objeto Array Subset (fig. 42).

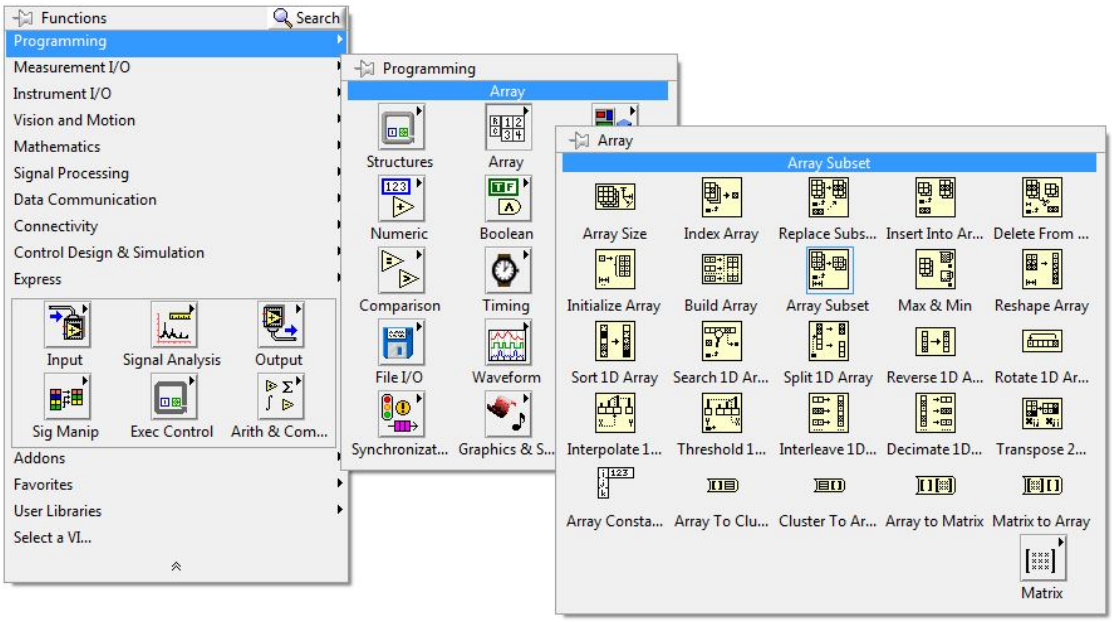


Fig. 42: *Array Subset*

De esta manera podemos compararlos con el comienzo de trama de cada configuración del LMS, he realizado cinco comparaciones diferentes una para cada tipo de configuración, en la imagen sólo se muestra para el caso de configuración de 0^o a 100^o con una resolución de 1^o. Para más información sobre principales tramas de configuración y número de datos enviados en función de la configuración véase apartado 1.6.2.1.

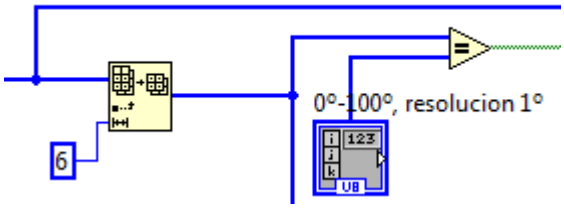


Fig. 43: *Comparación del comienzo de trama*

Cuando coincidan podremos saber en qué tipo de configuración se encuentra el láser y de esa manera saber cuántos bytes de datos va a enviar. Para comprobar que todos los bytes son iguales he utilizado el objeto And Array Elements.

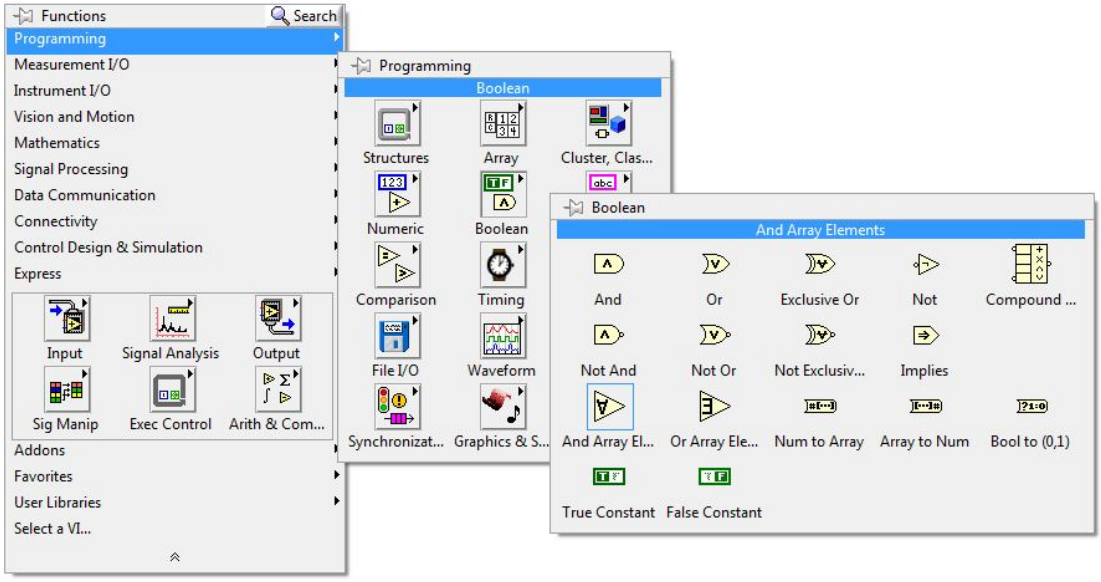


Fig. 44: *And Array Elements*

De esta manera, en el caso de ser verdadero se puede conectar como condición para un Case Structure para proceder en consecuencia.

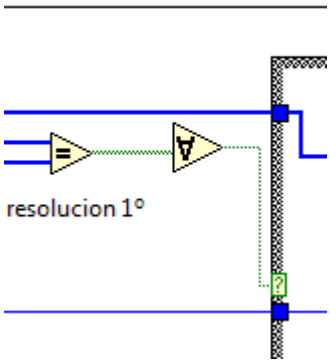


Fig. 45: *Conexión And Array Elements*

Una vez dentro del Case Structure, para el caso de ejemplo de encontrarse el láser en la configuración de 0^o a 100^o con una resolución de 1^o, se cogen los siguientes 202 bytes correspondientes a los bytes de datos que contienen las medidas que calcula el láser, una por cada grado al ser resolución 1^o. En el caso de ser la configuración de 0^o a 100^o con una resolución de 0.5^o serían 402 bytes de datos, dos por cada grado al ser resolución 0.5^o, y así sucesivamente. El objeto utilizado para coger los bytes de datos del array Read es otra vez Array Subset.

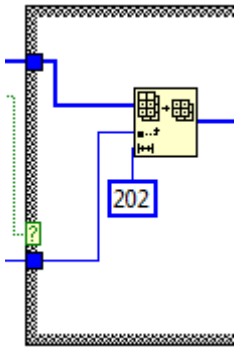


Fig. 46: *Array Subset para adquirir bytes de datos*

De esta manera a la salida del Array Subset obtendríamos un array que contendría los bytes de datos, pero la medida no sería del todo correcta. Como se puede observar en la tabla 18 la forma en la que el láser envía los datos son en paquetes de dos bytes primeramente el byte menos significativo y seguidamente el más significativo. En mi presentación de datos he querido cambiar este aspecto y colocar primeramente los high bytes y luego los low bytes puesto que creo que facilitaría su comprensión. Para ello he utilizado el VI Decimate 1D Array.

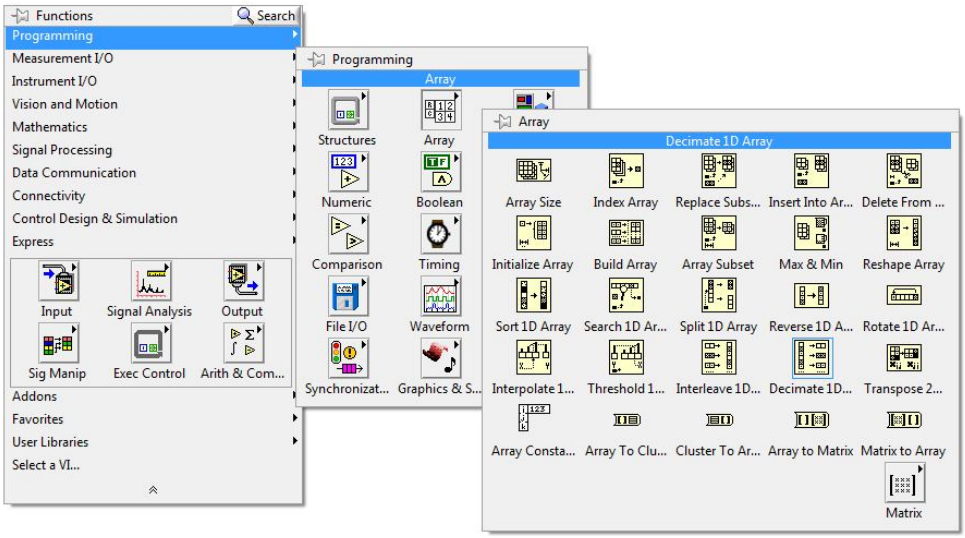


Fig. 47: *Decimate 1D Array*

Este VI además de permitirme separar los bytes impares de los pares para colocarlos correctamente, me facilita el poder actuar directamente sobre los bytes más significativos.

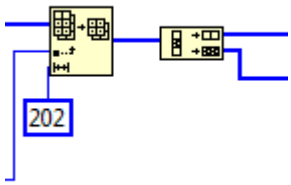


Fig. 48: *Separación bytes pares e impares de datos*

La razón por la cual me interesa actuar sobre los bytes más significativos es la siguiente:

Como ya expliqué en el apartado 1.6.4. y más exactamente en la tabla 20 dentro de cada dos bytes de datos los bits 13, 14 y 15 (equivalentes a los bits 5, 6 y 7 del byte más significativo) no hay que tenerlos en cuenta en la medida por lo que hay que aplicar un filtro a todos los high bytes que contiene el array para eliminar dichos bits. Esto se consigue haciendo una AND entre cada high byte y el valor 1F hexadecimal (00011111 en binario), lo que me permite mantener intactos los bits del 0 al 4 y no tener en cuenta los bits 5, 6 y 7.

Para demostrarlo pongamos un ejemplo en el que el byte más significativo del dato 0 sea DA hexadecimal (11011010 en binario) al aplicarle la función lógica AND con el valor 1F hexadecimal se eliminarían los bits deseados como muestra la figura 49.

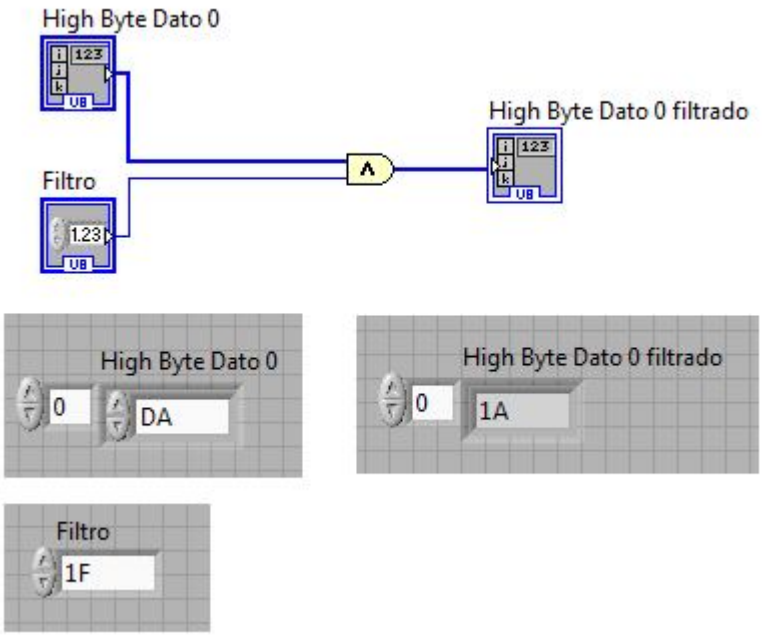


Fig. 49: *Ejemplo filtrado High Byte*

Por lo que el conjunto completo de VI's que forman el Case Structure de cada tipo de configuración sería en siguiente.

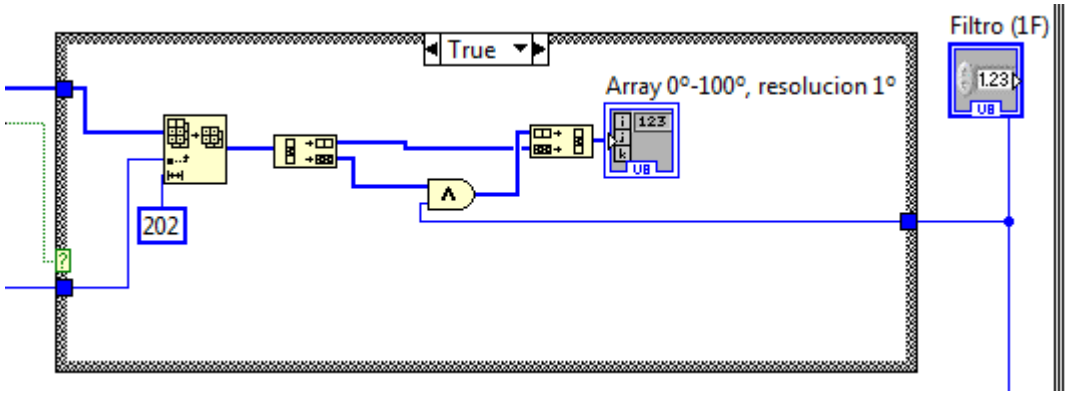


Fig. 50: *Case Structure configuración 0º a 100º con una resolución de 1º*

2.2.4 Presentación en el Panel Frontal

El Panel Frontal es el lugar desde donde el usuario debe controlar el programa. He querido separar cada nivel en diferentes carpetas para hacer su uso más sencillo e intuitivo.

2.2.4.1 Configuración

En el apartado configuración se pueden encontrar los objetos referentes a la configuración del puerto serie como baud rate, bit de paridad, bit de stop...

Fig. 51: *Panel Configuración*

2.2.4.2 **Control**

La carpeta control es la encargada de albergar los switch correspondientes a la lectura, la escritura y el botón de parada suave.

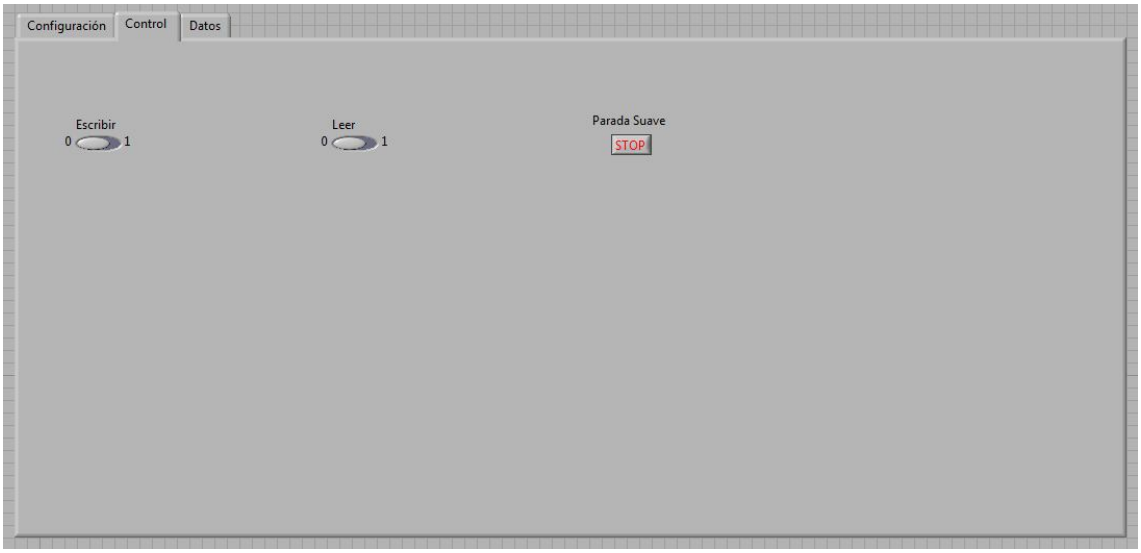


Fig. 52: *Panel control*

2.2.4.3 **Datos**

En la sección de datos se muestran las string para escribir y leer en el puerto serie como el array donde vuelco todos los datos del Read String para guardarlos como valores hexadecimales.

También se encuentran los arrays de cada tipo de configuración en el que guardo los datos correspondientes a las distancias que mide el láser que es el objetivo fundamental de este proyecto.

Las cabeceras fijas de las tramas de cada tipo de configuración que utilizo para compararlos con la trama de datos y saber en qué configuración está el láser y el filtro de valor constante 1F hexadecimal que uso para filtrar los bits 5, 6 y 7 de cada high byte de datos.

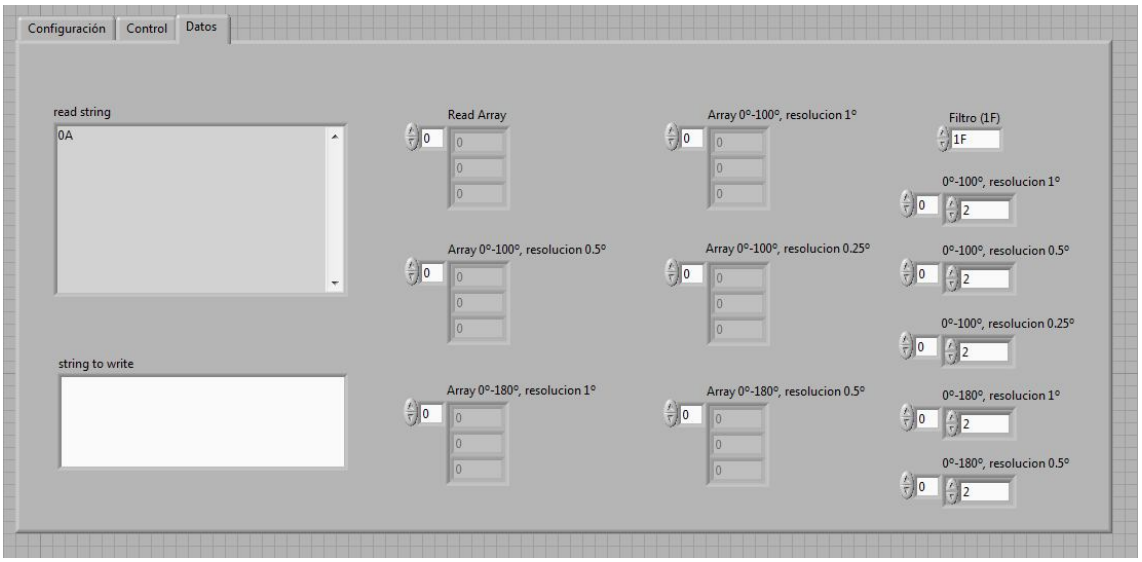


Fig. 53: *Panel Datos*

III. MEJORAS PROPUESTAS

Como principal mejora posible para realizar en el proyecto está la comprobación que el CRC que calcula el LMS y que envía en la trama de datos sea igual que el que calculemos en LabVIEW. Es algo que no pude realizar puesto que no sabía exactamente de qué manera lo calculaba el láser y tras probar varios métodos no conseguí que ninguno coincidiera. Por ejemplo, no sé si el láser calcula el CRC cambiando en la trama los low bytes por los high bytes o no. De todas maneras probé calcularlo de ambas formas y ninguno salía igual.

Un aspecto importante a tener en cuenta en el cálculo del CRC es saber exactamente con que bytes lo hace el LMS. El láser lo calcula comenzando en el byte STX y terminando en el byte de Status como muestra la siguiente tabla. Para más información sobre la estructura de los telegramas véase tabla 19.

DESCRIPTION OF THE TELEGRAM COMPONENTS	DATA LENGTH IN BITS / DATA LENGTH IN BYTES / DATA CLASS	EXPLANATION
Checksum	16/2/ WORD	CRC checksum of the entire data package, starting with STX and up to and including the status byte.

Tabla 22: *Estructura del telegrama Checksum*

3.1 INTRODUCCIÓN

La comprobación de redundancia cíclica (CRC) es un tipo de función que recibe un flujo de datos de cualquier longitud como entrada y devuelve un valor de longitud fija como salida. El término suele ser usado para designar tanto a la función como a su resultado. Pueden ser usadas como suma de verificación para detectar la alteración de datos durante su transmisión o almacenamiento.

Es útil para detección de errores, pero, en condiciones de seguridad, no podemos confiar en que el CRC puede verificar plenamente que los datos son los correctos en caso de que se hayan producido cambios deliberados y no aleatorios.

A menudo se piensa que si, cuando llega un mensaje, éste y su CRC coinciden, quiere decir que el mensaje no ha podido ser alterado durante su transmisión, aunque se haya transmitido por un canal abierto.

Esta suposición es falsa porque CRC es un mal método de cifrado de datos. De hecho, el CRC no se trata realmente de un método de cifrado, lo que realmente hace es utilizarse para el control de integridad de datos, pero en algunos casos se supone que se utilizarán para el cifrado.

Cuando un CRC se calcula, el mensaje se conserva (no cifrado) y la constante de tamaño CRC se sitúa hacia el final (es decir, el mensaje puede ser tan fácil como leer antes de la posición que ocupa el CRC).

Por supuesto, estos códigos están diseñados para ser lo suficientemente diferentes como para variar (y por lo general sólo en uno o dos bits). Pequeños cambios en la palabra clave producirían una gran diferencia entre un CRC y otro; por ese motivo es posible detectar el error.

3.2 ESTRUCTURA DE LOS CHECKSUMS

En el manual del LMS muestran parte del código correspondiente al cálculo del CRC.

```
/************************************************************
FUNCTION: Signature formation via CRC16 polynomial generator unsigned int build_crc16
(unsigned char * CommData, unsigned int len)

DESCRIPTION
Forms the checksum by means of CRC16_GEN_POL.
The following algorithm is used for a 16-bit checksum via the BYTE-oriented
buffer:
CRC_sum[High BYTE] = CRC_sum[Low BYTE]
CRC_sum[Low BYTE] = new data BYTE
Formation of the 16-bit CRC via CRC_sum
The following polynomial generator is used: x16 + x15 + x2 + 1
CRC16_GEN_POL EQU 8005H;
This constant is equal to x15 + x2 + 1, x16 is in the CARRY flag
Implementation in assembler for INTEL 80C196
************************************************************/

unsigned int build_crc16 (unsigned char *CommData, unsigned int uLen)
{
unsigned int uCrc16 = 0; /* Signature register */
unsigned int crc_data = 0; /* Current date */
static register unsigned int reg_len = uLen;
unsigned char *reg_data_ptr; /* Pointer to transferred data */
reg_data_ptr = CommData; /* Load transfer values from stack to register RAM */
/* Calculate CRC16 checksum */
CONT_CRC16:
asm SHL crc_data, #8; /*Shift low byte to high byte */
asm LDB crc_data, [reg_data_ptr]+; /*Load next byte and auto-increment */
asm SHL uCrc16, #1; /* Shift signature register one place to the left */
asm BNC NO_CARRY_SET; /* Interrogate the set CARRY flag */
asm XOR uCrc16, #CRC16_GEN_POL; /* If CARRY is set, XOR with polynomial gen. */
NO_CARRY_SET:
asm XOR uCrc16, crc_data; /* XOR the current date with signature reg. */
asm DEC reg_len; /* Continue loop until all data processed */
asm BNE CONT_CRC16;
END_CRC16:
return (uCrc16); /*return value is CRC16 checksum of data flow. */

}
```

Independientemente de la aplicación, esta función ANSI C aparece como sigue:

```
#define CRC16_GEN_POL 0x8005
#define MKSHORT(a,b) ((unsigned short) (a) | ((unsigned short)(b) << 8))

/* ::-------------------------------------------------------------------
:: FN: CreateCRC; CRC in ANSI - C
:: Synopsis: static void CreateCRC(BYTE *CommData,WORD uLen)
:: Function: formation of the CRC16 checksum.
-------------------------------------------------------------------*/
```

```
static WORD CreateCRC(unsigned char *CommData, unsigned int uLen )
{
unsigned short uCrc16:
unsigned char abData[2];
uCrc16 = 0;
abData[0] = 0;
while (uLen-- )
{
abData[1] = abData[0];
abData[0] = *CommData++;
if(uCrc16 & 0x8000)
{
uCrc16 = (uCrc16 & 0x7fff) << 1;
uCrc16 ^= CRC16_GEN_POL;
}
else
{
uCrc16 <<= 1;
}
uCrc16 ^= MKSHORT (abData[0] , abData[1]);
}
return(uCrc16);
        }
```

Como se puede observar viendo el código es que no está haciendo una suma del byte entero o de los 2 bytes, solo suma ciertos bits y con condiciones:

Los bits que suma son x16 + x15 + x2 + 1 (este último no es el bit 1 sino que le suma 1) donde el bit x16 es el bit de acarreo.

Es decir, al final utiliza 17 bits, 16 para el checksum y uno para el bit de acarreo, dependiendo de lo que este bit valga realizará una acción u otra.

- **if(uCrc16 & 0x8000)** : Quiere decir que si el bit 16, que será el nuevo bit 17 con la instrucción de abajo(aplicando máscara uCrc16 = (uCrc16 & 0x7fff) << 1; o sin aplicarla uCrc16 <<= 1.
- **(uCrc16 & 0x7fff):** Máscara para quedarse con todo el dato excepto el bit 16, es igual a uno hace una cosa (if) y si no la otra (else).
- **<<1**: Desplaza el byte a la izquierda 1 posición.
- **uCrc16 ^= CRC16_GEN_POL:** OR exclusiva con el dato definido 8005 (bit a bit) (8005 es nuestro polinomio, es decir bit0 bit2 y bit15 a 1 (x15, x2 y x0 que es 1) es 8005 en hexadecimal.

En este caso concreto el tipo de CRC es CRC 16-IBM como se puede apreciar en la siguiente tabla.

NAME	POLYNOMIAL	REPRESENTATIONS: NORMAL / REVERSED / REVERSE OF RECIPROCAL
CRC-16-IBM	$x^{16} + x^{15} + x^2 + 1$ (Bisync, Modbus, USB, ANSI X3.28, many others; also known as *CRC-16* and *CRC-16-ANSI*)	0x8005 / 0xA001 / 0xC002

Tabla 23: *CRC 16-IBM*

Para demostrar todo lo explicado anteriormente pongamos un ejemplo de una trama recibida para una configuración de 0^o a 100^o con una resolución de 1^o.

0280 CE00 B065 40FF 1FFF 1FFF 1FFF 1FFF 1FFF 1FFF 1FFF 1FFF 1FFF 1FFF 1FFF 1FFF 1FFF 1FFF 1FFF 1FFF 1FFF 1FFF 1FFF 1FFF 1FFF 1FFF 1FFF 1FFF 1F2B 052B 0513 05F0 04F5 04F6 04DD 0485 057B 05A0 05BC 05AF 05D0 10B9 10A8 1082 1014 0200 02DB 0 1CF 01CD 01C8 01CD 01C1 01B6 01BE 01C7 01D6 01D0 01D0 01D6 01D2 01D9 01DB 01DF 01DF 01E5 01E5 01EB 01F1 01FD 0102 020B 0210 021E 0224 021D 0204 0EC6 0D79 0DD9 0D95 0E8D 0FD1 0FEE 0FB8 0F5F 0F2F 0F91 0FD4 0FD4 1213 1343 1387 13A7 134C 13D7 1216 134A 139B 0BAE 0B87 0B2C 0B81 0B49 0B10 A244

Con lo que para este caso concreto el CRC calculado debería ser 44A2 que son los dos últimos bytes de la trama, mostrando primero el high byte seguido del low byte.

3.3 IMPLEMENTACIÓN EN LABVIEW

Para intentar calcular el CRC de la manera que lo calcula el LMS realicé la siguiente implementación en LabVIEW aunque no conseguí que el resultado fuera el mismo. He utilizado la misma trama de ejemplo de antes pero sólo cogiendo los datos desde el STX al Status Byte.

Como se puede observar el resultado no es 44A2.

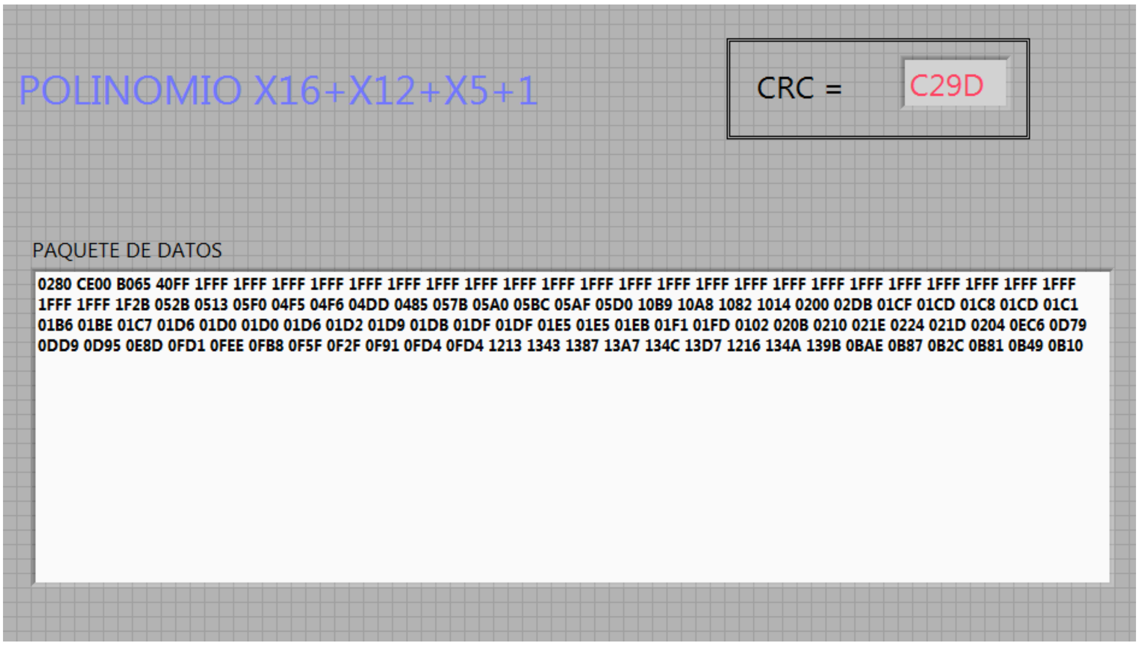


Fig. 54: *Panel Frontal cálculo CRC en LabVIEW*

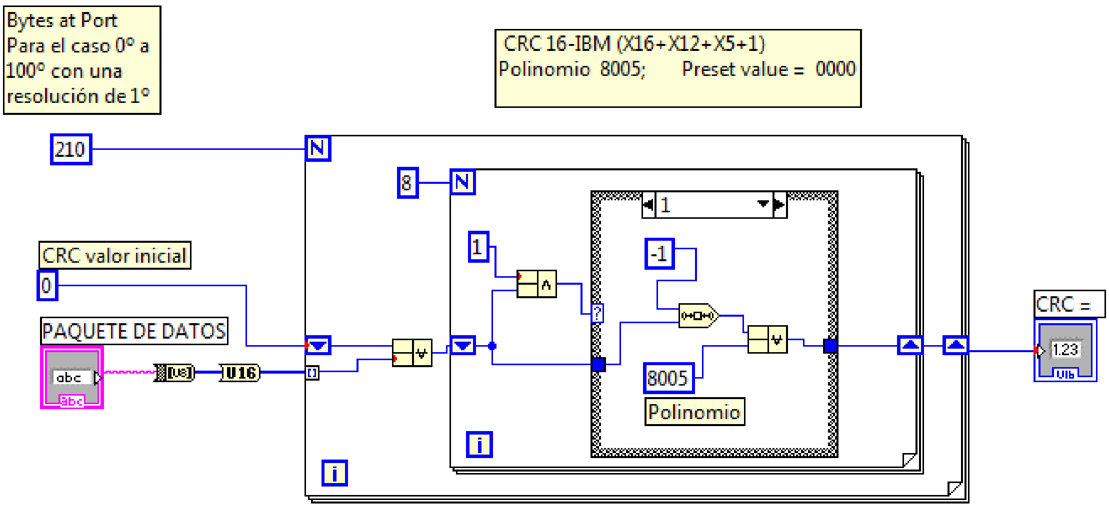


Fig. 55: *Diagrama de bloques cálculo CRC en LabVIEW*

IV. CONCLUSIÓN

En el ámbito personal, la realización de este proyecto ha supuesto para mí la mejora en mis conocimientos de LabVIEW. Como ya mencioné anteriormente utilicé LabVIEW hace años en una asignatura del Módulo de grado superior Regulación y Control Automático que hice antes de comenzar con la carrera. En aquel caso lo utilizamos para controlar desde el puerto serie una placa que habíamos realizado previamente. Pudiendo controlar por ejemplo el encendido y apagado de los LED's y su visualización en la interfaz de LabVIEW.

La dificultad de este proyecto es mucho mayor y gracias a ello he podido ver de qué manera comunica un dispositivo por el puerto serie, la forma y organización de las tramas de datos. Sé que cada dispositivo tiene su método de comunicación, pero creo que sabiendo uno sólo queda ver las diferencias y hacer una programación acorde al tipo de dispositivo. Con lo que espero poder utilizar esta herramienta más veces a lo largo de mi carrera profesional.

Gracias a la amplia variedad de herramientas presentes y el excelente funcionamiento de la interfaz gráfica otorgada por el lenguaje G en el control y configuración de dispositivos por el puerto serie se tradujo en una aplicación en la que es posible la comunicación siguiendo unos patrones establecidos con el láser LMS 291-S05. Todo esto realizado mediante una suma de instrumentos virtuales los cuales recopilan y entregan la información necesaria para su visualización.

En un ámbito técnico cabe destacar que la facilidad de interconexión entre el software y el hardware (DAQ) que LabVIEW proporciona, logra disminuir los tiempos y con ello los costos de realización de cualquier proyecto.

El avance en los sistemas desarrollados mediante LabVIEW se ha vuelto cada vez más completo entregando soluciones a todo nivel sin embargo la posibilidad de seguir explotando cada una de las herramientas presentes vuelve a este atractivo software en una herramienta quizás hasta necesaria tanto para alumnos como docentes todo esto avalado por la tendencia de la empresa actual en donde cada vez más se unen en una solo línea elementos un tanto distantes como neumática, control automático, electrónica y todo esto mediante a sistemas desarrollados para el visualización o bien el control de los procesos lo cual se traduce en una centralización de la información así como un control más cercano de los procesos que se realizan en el área de trabajo.

Aunque yo no he dado uso de ello en la realización de mi proyecto, es remarcable la posibilidad y facilidad de uso de otros lenguajes de programación como puede ser C++ o Python o el intercambio de datos con hojas de datos como Excel. Lo que facilitaría en gran medida la organización y tratamiento en la gestión de datos.

En cuanto al lenguaje de programación G se observa la facilidad de manejo que entrega para el usuario lo cual deriva en un fácil aprendizaje del sistema con una constante evolución e incorporación de nuevas herramientas.

A su vez, no solo el lenguaje evoluciona sino que además el software en si lo hace de forma paralela debido a que National Instruments incorpora constantemente nuevos elementos de desarrollo y análisis permitiendo realizar y abordar temas tan cotidianos como adquisición de datos, diseño y análisis de circuitos electrónicos con Electronics Workbench como empresa subsidiaria con la producción de su software MultiSIM, interconexión mediante protocolos RS-232, RS-485, TCP-IP hasta sus últimos módulos como por ejemplo MODBUS o bien CAN-USB es aquí donde NI ha marcado la diferencia y lo ha llevado a ser uno de los lideres en mercados tradicionales como son los campos de adquisición de datos, control de instrumentos e instrumentación virtual, e incorporarse en nuevos mercados como los sistemas de comunicaciones y sistemas embebidos.

También he podido observar que LabVIEW no solo se basa en control y análisis si no que también puede ser parte del uso cotidiano, como por ejemplo la línea desarrollada en conjunto con LEGO que puede llegar a ser sin duda un gran aporte dirigido a alumnos de niveles de educación mas inferiores y que supone un claro incentivo en el área de la robótica y sus derivados.

V. BIBLIOGRAFÍA

- http://www.ni.com
- http://www.measurementcomputing.com
- http://es.wikipedia.org/wiki/LabVIEW
- http://es.wikipedia.org/wiki/Comprobaci%C3%B3n_de_redundancia_c%C3%ADclica
- http://forums.ni.com/
- MÁNUEL LÁZARO, A., RÍO FERNÁDEZ, J.,"Programación Gráfica para el Control e Instrumentación", Madrid, España.
- Getting Started with LabVIEW, April 2003 Edition.
- Getting Started with the LabVIEW Datalogging and Supervisory Control Module 7.1.