UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

TEMA DE TESIS

"DISEÑO E INTEGRACIÓN DE UN SISTEMA PARA EVALUAR EMISIONES AUTOMOTRICES CON INTERFAZ VIRTUAL, EMPLEANDO BANCAS ANALIZADORAS DE GASES COMERCIALES"

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

P R E S E N T A N

SÁNCHEZ ALFARO MARÍA GUADALUPE

VÁZQUEZ ZURITA BEATRIZ GABRIELA

DIRECTOR DE TESIS: M. en I. PEDRO IGNACIO RINCÓN GÓMEZ

CIUDAD UNIVERSITARIA MÉXICO, D.F. 2008

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

AGRADECIMIENTOS

A la Universidad Nacional Autónoma de México, por permitirme pertenecer a ella, estoy orgullosa de mi universidad.

A la Facultad de Ingeniería, por brindarme una educación de calidad y por proporcionarme las instalaciones necesarias a lo largo de toda mi carrera.

Al ingeniero Pedro Rincón, por ofrecernos su ayuda y comprensión en el transcurso del desarrollo de esta tesis.

A Bety, por ser mi compañera de tesis y gran amiga.

A mis padres y hermanos que siempre me han dado su apoyo incondicional durante toda mi vida. Papás esta tesis es para ustedes.

Y por último, pero igual de importante, a mi novio Humberto, porque desde que lo conozco siempre ha estado conmigo y me ha apoyado bajo cualquier circunstancia.

Gracias.

Atte. Lupita.

AGRADECIMIENTOS

A MI MADRE:

A quien le agradezco infinitamente todo lo que soy, gracias por apoyarme siempre e impulsarme día a día para ser una mejor persona, cada uno de mis logros son gracias a ti, TE AMO mamá.

A MIS HERMANOS:

Irina y Maximiliano, gracias por todo su cariño y entusiasmo, LOS AMO.

A MI PADRE:

Por sus palabras, que me han ayudado a buscar nuevos retos y nuevas metas.

A MI UNIVERSIDAD:

Mil gracias por todo lo que me has dado, nunca olvidare cada momento que me permitiste vivir y aprender dentro de tus instalaciones, fuiste y seguirás siendo mi segunda casa, orgullosamente PUMA, orgullosamente UNIVERSITARIA.

A MI AMIGA LUPITA:

Por todo el apoyo y paciencia que me dio, para poder realizar este trabajo de titulación.

AL INGENIERO PEDRO RINCON:

Por habernos dado la oportunidad de desarrollar esta tesis, y ayudarnos a la elaboración de la misma, muchas gracias por el apoyo y conocimientos que nos brindo.

A MIS AMIGOS:

Un millón de gracias por todos los consejos sabios que me han dado, y por todo el apoyo, cariño y amistad que me han ofrecido, mis mejores recuerdos son gracias a ustedes, los quiero mucho.

A EDUARDO:

Por hacer que volviera a creer, TA.

Gracias.

Atte. Beatriz Vázquez.

CONTENIDO

- **CAPÍTULO VI**
  DISEÑO DEL SOFTWARE

- **CAPÍTULO VII**
  INFORME DE PRUEBAS Y CONCLUSIONES

SINÓPSIS

El Laboratorio de Control de Emisiones (LCE) de la Facultad de Ingeniería (FI) de la Universidad Nacional Autónoma de México (UNAM) fue creado con el fin de estudiar las emisiones de los vehículos que poseen motores de combustión interna. Para poder hacer esto, se vale de diferentes instrumentos de medición, analizadores de gases, dinamómetros, escáners automotrices, etcétera. Algunos instrumentos o dispositivos que sólo pueden adquirirse en el extranjero son diseñados en este laboratorio con la asesoría de la coordinación de instrumentación del Instituto de Ingeniería. El desarrollo de estos dispositivos permite que el laboratorio cuente cada vez con más equipo útil adecuado a sus necesidades.

En la presente tesis se muestra el desarrollo de un proyecto para el Laboratorio de Control de Emisiones; se trata de la instrumentación de una banca analizadora de gases, la cual no tenía ningún uso. En el desarrollo de este proyecto se utilizó una computadora personal con el entorno de programación LabVIEW, versión 8.0 de *National Instruments*®. El software que se utiliza para controlar la banca analizadora fue, precisamente, desarrollado con este entorno de programación, lo que permitió generar un instrumento virtual muy versátil y de fácil manejo.

El analizador de gases opera como un instrumento más del laboratorio y se piensa que en el futuro pueda ser parte de un sistema integral de análisis de gases para vehículos automotores ligeros.

PRÓLOGO

La presente tesis tiene por objeto ilustrar el desarrollo completo de un analizador de gases para el uso del Laboratorio de Control de Emisiones de la Facultad de Ingeniería de la UNAM. Este trabajo se compone de siete capítulos, los cuales de distribuyen de la siguiente manera.

El primer capítulo es una breve semblanza del laboratorio, que sirve de panorama general para que el lector comprenda de forma clara y objetiva el porque de la existencia de este laboratorio en una institución docente y también para que se familiarice con los diferentes dispositivos e instrumentos básicos que forman parte de su equipo. Además, en este capítulo se describe el tipo de actividades particulares que se llevan a cabo en el mismo.

Posteriormente, en el capítulo II el lector tendrá una visión más clara de la relación que existe entre los analizadores de gases y los dinamómetros automotrices, para esto se describen diferentes tipos de dinamómetros automotrices que pueden existir en un laboratorio de emisiones, así como también de sus principales características, su principio de operación y su aspecto físico.

El capítulo III está enfocado en describir el funcionamiento del analizador de gases desarrollado en esta tesis. Este capítulo permite que el lector conozca el contexto particular en que se encuentra este dispositivo.

El capítulo IV consiste en la descripción del hardware propuesto como solución para instrumentar el analizador de gases. El lector podrá distinguir tanto los diferentes módulos que componen el equipo completo del analizador, como también el principio de operación de los mismos.

El capítulo V permite que el lector se familiarice con la utilización de un paquete de desarrollo de software denominado LabVIEW; que le será de mucha utilidad para comprender el diseño completo propuesto para instrumentar el analizador.

El capítulo VI permite que el lector comprenda el funcionamiento del software que gobierna al sistema completo y que le da una identidad propia como un instrumento virtual independiente.

Las pruebas que se hicieron al sistema completo así como los resultados de las mismas se describen en el capítulo VII, junto con las conclusiones a las que se llegaron.

Por último, se incluyen en la tesis la bibliografía consultada y dos apéndices: uno que ilustra al lector los instrumentos virtuales (VI's) y funciones de la biblioteca estándar del paquete de desarrollo LabVIEW; y otro en el que se describe el proceso que se sigue para la medición de las emisiones producidas por los vehículos automotores.

INTRODUCCIÓN

La causa principal de contaminación en nuestros días se debe al humo que emana de los automóviles. La contaminación del aire es uno de los problemas ambientales más importantes, y generalmente es resultado de las actividades del hombre.

El aire puro es una mezcla gaseosa compuesta por un 78% de nitrógeno, un 21% de oxígeno y un 1% de diferentes compuestos, tales como el argón, el dióxido de carbono y el ozono. Se puede entender por contaminación atmosférica cualquier cambio en el equilibrio de estos componentes, lo cual altera las propiedades físicas y químicas del aire.

En la República Mexicana, grandes urbes como la ciudad de México, Guadalajara y Monterrey son de las más afectadas por las emisiones vehiculares.

A pesar de lo mencionado anteriormente, es la ciudad de México la que sufre más de la contaminación atmosférica, dado que su situación geográfica no le permite un flujo de aire como en el resto del país. Esto provoca que los gases se acumulen y se concentren en la atmósfera.

Según los datos del Instituto Nacional de Estadística, Geografía e Informática (INEGI), en 2004 las emisiones anuales de contaminantes en la Zona Metropolitana del Valle de México son superiores a 3.5 millones de toneladas[1].

No es extraño que en ocasiones se tenga que aplicar el “plan de contingencia ambiental” ante los altos índices de contaminación en el aire de la ciudad de México, si se está consciente de que existe un parque vehicular superior a los 4 millones de vehículos[2] y que a pesar del programa "Hoy no circula", el tránsito de vehículos en la ciudad es continuo. Es obvio que tal cantidad de motores dejen un impacto considerable en el ambiente semicerrado de la ciudad aunado a otros factores como la industria y los incendios forestales.

Para regular la contaminación en el Valle de México, el gobierno del DF ha implantado medidas para el control de emisiones de gases, producto de la combustión de los vehículos automotores, mediante la creación de programas como "ProAmbiental Vehicular",

[1] INEGI, http://www.inegi.gob.mx/inegi/fnivel.aspx?s=est&c=6159&e=&i=

[2] UGAM-Transporte y Vialidad, http://www.union.org.mx/guia/actividadesyagravios/transporte.htm

"Hoy no circula" y la red de monitoreo del denominado Índice Metropolitano de Contaminación Ambiental (IMECA).

ProAmbiental Vehicular es el programa destinado a regular la flota vehicular del Distrito Federal y del Estado de México, para que circule cumpliendo con los Niveles Permisibles de Emisión de Gases Contaminantes, contribuyendo así a mantener más limpia la atmósfera. La verificación de cada automóvil debe realizarse cada 6 meses (dos veces por año) en un "Verificentro" (centros autorizados para verificar la emisión de gases para vehículos con placas del Distrito Federal o del Estado de México).

Además, los problemas ambientales que enfrenta la población en la actualidad han llevado a la formulación de normas para la protección del medio ambiente. Para generar dichas normas ha sido necesario utilizar equipos sofisticados que proporcionen información útil para establecer ciertos índices de contaminantes.

El incremento en la cantidad de vehículos automotores que circulan en las ciudades ha provocado la contaminación del aire, es por esto que surgió la necesidad de cuantificar el nivel de contaminantes que despiden estos vehículos para mantener la calidad del aire en ciertos índices.

Para determinar qué tanto contamina un vehículo se desarrollaron los analizadores de gases, los cuales permiten medir la cantidad de gases que se generan como subproductos de la combustión en un motor, los cuales pueden provocar efectos adversos en los seres vivos o alterar el medio ambiente.

La norma que rige en México la cantidad de emisiones permitidas y las características de los vehículos automotores es la NOM-041-ECOL-1999[3]. La NOM-047-ECOL-1993 proporciona las características de los equipos y el procedimiento para estimar los límites de emisión de contaminantes de vehículos que utilizan combustibles como la gasolina, gas licuado de petróleo, gas natural u otros combustibles alternos.

Debido a la gran importancia que tiene este problema, la Facultad de Ingeniería de la Universidad Nacional Autónoma de México, creó el Laboratorio de Control de Emisiones, para investigar diferentes métodos para reducir los contaminantes de las emisiones vehiculares. De esta forma, con ayuda del Instituto de Ingeniería, el laboratorio también

[3] Instituto Nacional de Ecología, http://www.ine.gob.mx/ueajei/publicaciones/gacetas/275/nom41.html

genera la tecnología necesaria para poder estudiar el fenómeno atmosférico con el fin de encontrar soluciones al problema que aqueja a la ciudad. Algunos de estos estudios además sirven para hacer reformas a las normas vigentes o generar nuevas, en cuanto a emisiones de vehículos se refiere.

El desarrollo de la presente tesis trata precisamente acerca del desarrollo de un analizador que permita obtener los índices de los gases contaminantes arrojados por los automóviles. Con esto se pretende que el laboratorio pueda contar con una herramienta útil para fines de investigación y docencia que contribuya de alguna manera para generar soluciones al problema ambiental que aqueja principalmente a la ciudad de México.

Por todo lo anterior los objetivos precisos de esta tesis son los siguientes:

Objetivos:

- Integrar el hardware necesario para el desarrollo de un analizador de gases.

- Diseñar la interfaz virtual entre la computadora y el analizador de gases por medio del entorno gráfico de programación LabVIEW 8.0 de National Instruments®.

CAPÍTULO I

EL LABORATORIO DE CONTROL DE EMISIONES DE LA FACULTAD DE INGENIERÍA DE LA UNAM

El objetivo de este primer capítulo es darle al lector una idea más clara del equipo con el que se cuenta y las actividades que se realizan en el Laboratorio de Control de Emisiones de la Facultad de Ingeniería de la UNAM, para entender posteriormente el porque se requiere del sistema diseñado en esta tesis.

I.1. ANTECEDENTES

El Laboratorio de Control de Emisiones se ubica en el ala oriente del edificio anexo de la Facultad de Ingeniería de la UNAM, junto a las oficinas administrativas y al centro de Diseño y Manufactura de la División de Ingeniería Mecánica e Industrial (DIMEI); fue inaugurado en Octubre de 1996, ocupando un área de 1,379.67 [m^2], que comparte con los laboratorios de Termofluidos en la planta baja y con cubículos y oficinas académico-administrativas en el primer piso. Este laboratorio forma parte del departamento de Termofluidos de la DIMEI y actualmente sigue prestando atención a más de 400 estudiantes (figura I.1).

Figura I.1. Laboratorio de Control de Emisiones.

A la fecha se han desarrollado más de 14 proyectos subvencionados, que han sido apoyados por la UNAM, mediante el "Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica" (PAPIIT), por el gobierno del Distrito Federal, por el gobierno de Michoacán y por el fideicomiso SEMARNAT-CONACYT, además del apoyo de diferentes industrias, y de pequeños servicios que algunos particulares solicitan.

En el LCE se han desarrollado procesos, los cuales son:

1. Investigación en el control de emisiones por fuentes móviles.
2. Investigación y desarrollo en Motores de Combustión Interna Alternativos (MCIA).
3. Servicios de evaluación de sistemas, dispositivos y aditivos que reducen el consumo de combustible y las emisiones contaminantes.

4. Servicios de asesoría en Motores de Combustión Interna Alternativos.
5. Desarrollo de ciclos de manejo.
6. Desarrollo de equipo de medición.

Y también innovaciones tecnológicas como son:

1. Dinamómetro de chasis para vehículos ligeros.
2. Banco para ensayos de motocicletas
3. Sala de ensayos para motores de combustión interna alternativos
4. Sistema de adquisición de datos vehiculares a bordo
5. Banco de flujo estacionario para aplicaciones automotrices.

I.2. INFRAESTRUCTURA DEL LABORATORIO DE CONTROL DE EMISIONES

El Laboratorio de Control de Emisiones cuenta con el equipo necesario para realizar estudios específicos de emisión en vehículos tanto livianos como medianos, tal es el caso de las emisiones del sistema de escape y emisiones evaporativas, en condiciones ambientales normalizadas y de acuerdo a los procedimientos y normativas vigentes.

Entre los equipos con los que se cuenta en el LCE están:

- Banco de Flujo
- Banco de motocicletas (dinamómetro para motocicletas)
- Dinamómetro de chasis
- Dinamómetro de banco
- Escáner automotriz
- Medidor de gasto másico de combustible
- Sistema AVL® (Computador central con analizadores de C0, $C0_2$, HC, $N0_x$)
- Sistema Retrofit

I.3. OBJETIVOS DEL LCE

Los objetivos de LCE se agrupan en tres rubros principales: docencia, investigación y servicios.

1. Docencia: Apoyar la docencia a través de cursos con contenidos prácticos y actuales en el ámbito de MCIA, tanto para alumnos de carreras afines así como para profesionistas y técnicos que los requieran.

2. Investigación: Desarrollar métodos, técnicas, tecnologías y herramientas para el control de las emisiones contaminantes y reducción del consumo de combustible de los MCIA.

3. Servicios: Ofrecer servicios y asesoría para evaluar sistemas, dispositivos, combustibles y aditivos, que reduzcan las emisiones contaminantes y el consumo de combustibles en fuentes móviles.

Además también se tiene por objetivo desarrollar equipo de medición para ensayos en motores y vehículos, y desarrollar ciclos de manejo.

I.4. FUNCIONES QUE SE LLEVAN A CABO EN EL LCE

Dentro de las principales funciones que tiene el Laboratorio de Control de Emisiones de la Facultad de Ingeniería de la UNAM se encuentran las siguientes:

- ***Enseñanza***

El L.C.E. desempeña un papel fundamental como centro docente, ya que presta todos los servicios necesarios a los alumnos de ingeniería mecánica para desarrollar prácticas que les ayuden a una mejor comprensión de los conocimientos teóricos, que se dan en asignaturas como termodinámica o mecánica de fluidos.

Además este laboratorio promueve la prestación de servicio social a estudiantes de las carreras de Ingeniería Mecánica e Ingeniería Electrónica, puesto que dentro de sus instalaciones se desarrolla equipo mecánico que requiere de instrumentación electrónica.

- ***Homologación y certificación vehicular***

Otra de las funciones que tiene este laboratorio es la de realizar la homologación a prototipos y/o modelos de vehículos de producción que pretendan comercializarse en el país.

Así mismo se encarga de verificar el cumplimiento de normas constructivas en vehículos destinados al transporte público de pasajeros.

- ***Control de la certificación vehicular***

El LCE también controla el cumplimiento de las normas tanto constructivas como de emisión en vehículos, los cuales son seleccionados aleatoriamente de las partidas de vehículos que son comercializados, igualmente se encarga de verificar las emisiones de los vehículos.

- ***Desarrollo e investigación***

Para el LCE, realizar investigación y desarrollo de normas e implementar programas de investigación y capacitación es una de sus actividades más importantes.

El laboratorio ha trabajado en varios proyectos que involucran el desarrollo de normas de emisiones conjuntamente con el gobierno del Distrito Federal, con el fin de combatir el grave problema de emisiones vehiculares en el valle de México.

En el siguiente capítulo se explicará como es que están relacionados los sistemas analizadores de gases con los dinamómetros, dentro de un laboratorio como el LCE de la Facultad de Ingeniería.

CAPÍTULO II

EL PAPEL DE LOS ANALIZADORES DE GASES EN SISTEMAS DINAMOMÉTRICOS EN UN LABORATORIO DE EMISIONES

En el presente capítulo el lector podrá conocer el papel que desempeñan los analizadores de gases en sistemas dinamométricos y para esto se dará una explicación de que es un dinamómetro automotriz, como está compuesto, para qué se utiliza y cómo se clasifican según su tecnología.

II.1. FUNCIÓN DE LOS ANALIZADORES EN SISTEMAS DINAMOMÉTRICOS

La relación que existe entre los analizadores de gases y los sistemas dinamométricos es muy importante.

Dado que no siempre es posible evaluar la concentración de contaminantes de los vehículos cuando están circulando libremente en las calles de la ciudad o en las carreteras, se vuelve necesario el uso de los sistemas dinamométricos.

Como se verá más adelante, los dinamómetros se encargan de reproducir o imitar las condiciones que se presentan cuando un motor de un vehículo o un vehículo como tal se encuentra circulando libremente. La ventaja es que con los dinamómetros esto se puede hacer en un lugar estacionario, sin que el vehículo se traslade hacia algún lugar. De esa manera se toman muestras de los gases contaminantes con la ayuda de los analizadores de gases para, posteriormente, poder determinar su comportamiento como si se encontraran circulando libremente.

Los dinamómetros, cuentan con Unidades de Absorción de Potencia (PAU: Power Absortion Unit). Estos dispositivos frenan a un motor con diferentes intensidades a los que se les denomina "niveles de carga".

La intención de frenar los motores de los vehículos dentro de un laboratorio de emisiones, como el de Ciudad Universitaria (CU), es reproducir el esfuerzo que los motores hacen cuando los vehículos suben pendientes. Y es que entre más esfuerzo haga un motor de combustión interna, más contamina.

El procedimiento realizado para la medición de las emisiones producidas por los vehículos automotores se explicará en el Apéndice B.

II.2. EL DINAMÓMETRO COMO INSTRUMENTO DE MEDICIÓN

El dinamómetro es un instrumento que se utiliza para medir la fuerza o potencia mecánica de un motor. Comúnmente se utiliza en los laboratorios de análisis de emisiones de gases de vehículos que poseen motores de combustión interna, para analizar los componentes contaminantes de las emisiones bajo determinado ciclo de trabajo; sin embargo, puesto que se trata de un instrumento que sirve para medir la potencia de un motor

cualesquiera, también se pueden utilizar para hacer estas mediciones a motores eléctricos o de otra índole. Un dinamómetro se compone de tres partes fundamentales (figura II.1):

- Estructura de acoplamiento (eje)
- Unidad de absorción de potencia (rotor y estator)
- Celda de carga

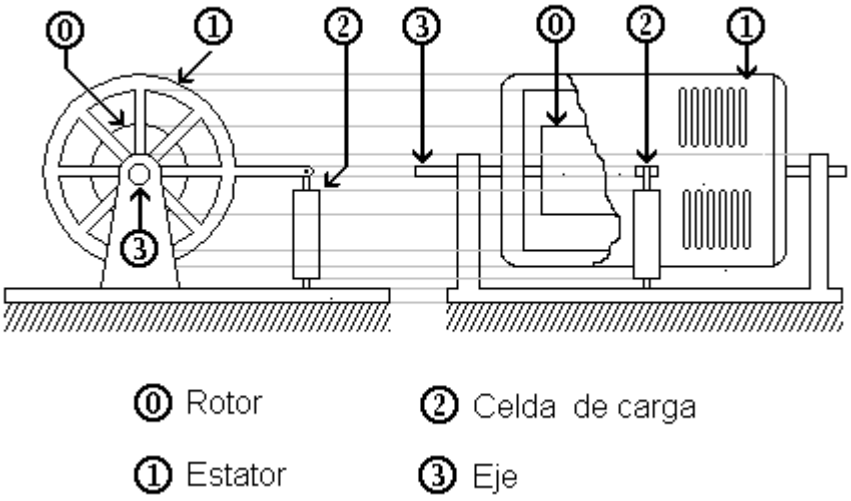


Figura II.1. Elementos básicos de un dinamómetro.

La estructura de acoplamiento o eje es la que se encarga de conectar al motor o al automóvil que se desea evaluar con la unidad de absorción de potencia, para que ésta pueda frenarlo.

La Unidad de Absorción de Potencia es un freno que se acopla al motor que se desea evaluar para poder forzarlo. Mediante una *PAU* es posible evaluar el desempeño de un motor aplicándole diferentes intensidades de frenado para observar su comportamiento en cuanto al consumo de combustible (rendimiento), nivel de ruido, potencia, velocidad, temperatura, etc.

La *PAU* está compuesta de dos partes fundamentales: el rotor y el estator. El motor a evaluar siempre se acopla al rotor de la *PAU,* para que sea frenado por medio del estator. La forma en que éste logra frenar al rotor depende de la tecnología que se utilice (puede ser mecánica, hidráulica o magnéticamente). El estator de una *PAU* es un elemento flotante, porque en lugar de que se encuentre fijo a una referencia estática, se sujeta mediante rodamientos a la flecha del rotor. Para evitar que éste gire junto con el rotor, se recarga en un dispositivo denominado celda de carga por medio de un brazo que los interconecta.

La celda de carga es un instrumento que se encarga de resistir la tensión que el motor ejerce sobre la unidad de absorción de potencia. Cuando el estator frena al rotor, éste tiende a girar en el mismo sentido que el rotor porque, como se mencionó anteriormente, es una estructura flotante; este movimiento se ve contrarestado por la conexión mecánica de la celda de carga. A pesar de ello, el estator logra girar ligeramente sobre el eje del rotor deformando la celda de carga comprimiéndola (figura II.2).

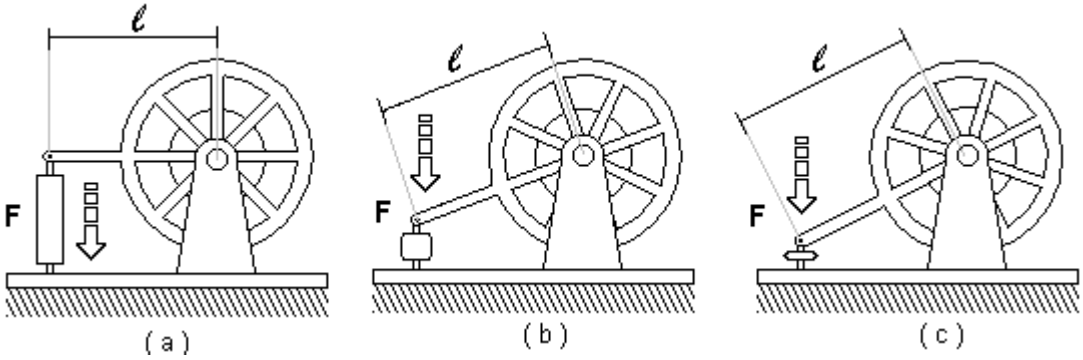


Figura II.2. Dinamómetro en funcionamiento. (nótese la deformación de la celda de carga)

La deformación que sufre la celda de carga es longitudinalmente proporcional a la fuerza que el motor ejerce sobre ella. Generalmente la celda de carga consiste en un resorte robusto o en alguna estructura metálica deformable que sea capaz de recobrar por sí misma su forma original. En la celda de carga se instala algún transductor que permita hacer mediciones longitudinales de las deformaciones que experimenta la celda cuando se frena a un motor. De esta forma se puede conocer el par del motor que se está evaluando.

II.3. TIPOS DE DINAMÓMETROS

Existen diferentes tipos de dinamómetros dependiendo de su tecnología. La variedad que se presenta a continuación permite comprender que estos dispositivos se diseñan para hacer mediciones muy específicas bajo diferentes circunstancias. Así, la existencia de varios tipos de dinamómetros no significa que alguno sea mejor que otro, por el hecho de haber sido diseñado bajo una u otra tecnología, ya que el diseño de estos dispositivos obedece principalmente a las necesidades que en un momento dado se requieran.

II.3.1. El dinamómetro de chasis

Los *dinamómetros de chasis* son instrumentos que se utilizan para medir la potencia del motor instalado en un vehículo. Aunque existen diferentes tipos de dinamómetros de

chasis, generalmente éstos se componen por un par de rodillos encima de los cuales se hace correr un vehículo, como se muestra en la figura II.3. Estos rodillos se encuentran acoplados a algún tipo de absorbente de potencia (freno) mediante el cual se le aplica una carga al vehículo.

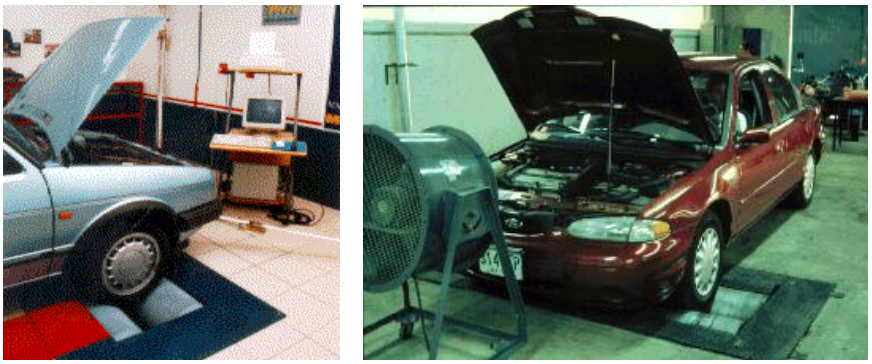


Figura II.3. Aspecto parcial de dos dinamómetros de chasis típicos.

La ventaja más importante del dinamómetro de chasis es la posibilidad de evaluar el motor de un vehículo sin necesidad de desmontarlo. Esto facilita el procedimiento de evaluación pero dificulta considerablemente la tarea de medir potencia, debido al gran número de factores que influyen en el sistema (hay que recordar que existen pérdidas en la transmisión, pérdidas por fricción en los neumáticos, presión atmosférica, temperatura ambiental, etc.). Además la instalación de sensores en el motor también está limitada cuando éste se encuentra montado en un automóvil. Existen otros tipos de dinamómetros de chasis, que no cuentan con ninguna clase de rodillos; únicamente se acoplan a los tambores de las flechas motrices del automóvil (figura II.4).

(a) (b)

Figura II.4. (a) Dinamómetro de chasis carente de rodillos; (b) detalle del acoplamiento directo al tambor de la flecha motriz.

Cuando se utilizan dinamómetros carentes de rodillos, se pueden despreciar las pérdidas por fricción en los neumáticos, pero con ello se complica el procedimiento de la prueba porque es necesario desmontar las ruedas de tracción del automóvil.

De igual forma, entre los dinamómetros de chasis existen algunos modelos que son muy complejos y que generalmente sólo se encuentran en algunas fábricas de automóviles (figura II.5).

Figura II.5. Dinamómetro de chasis en fábricas de automóviles.

El hecho de que exista gran cantidad de factores que influyen en el desempeño de un motor, no significa que los dinamómetros de chasis sean dispositivos ineficientes, por el contrario permiten conocer la potencia efectiva de un motor instalado en un vehículo particular.

Los dinamómetros de chasis se usan típicamente para:

- Evaluar rápidamente el desempeño de un automóvil.
- Medir las pérdidas en la transmisión de un vehículo.
- Medir exactamente el rendimiento de un vehículo.
- Hacer evaluaciones de consumo de combustible, ruido o emisiones

II.3.2. El dinamómetro de banco

Es posible medir la potencia de un motor instalado en un vehículo pero es más conveniente y exacto hacer estas mediciones con el motor instalado en un sistema de prueba denominado *dinamómetro de banco,* como el que se aprecia en la figura II.6.

Figura II.6. Aspecto del acoplo de un dinamómetro de banco.

Este dispositivo, a diferencia del dinamómetro de chasis, mide la potencia de un motor directamente de su flecha motriz.

Las mediciones hechas con este instrumento poseen una exactitud más alta que las que se hacen con un dinamómetro de chasis, debido a que no existen pérdidas provocadas por transmisión ni por fricción de neumáticos que influyan en los resultados. Además de que se puede tener un mejor control de todos los parámetros y de las condiciones de la prueba debido a la facilidad para la instalación de sensores y para el ajuste del motor. Sin embargo, evaluar el funcionamiento de un motor con un dinamómetro de banco implica desmontar el motor del automóvil; esto significa que es necesario agregar todos los sistemas auxiliares que apoyen el funcionamiento del mismo, como el suministro del combustible y la energía eléctrica, la extracción de la descarga, el flujo aéreo para refrescar al motor y para la favorecer la combustión, el mando de temperatura aéreo, refrigerante, actuador del acelerador, etc. Por esta razón un dinamómetro de banco resulta ser un sistema muy complejo.

Los dinamómetros de banco se usan típicamente para:

- La investigación, el desarrollo y la afinación de un motor
- Hacer mediciones exactas de rendimiento del motor
- Normalizar un prototipo

II.3.3. El dinamómetro de remolque

El *dinamómetro de remolque* es un tercer tipo de dinamómetro que no es muy común; se instala a manera de remolque en un vehículo cualquiera, pero en lugar de hacer correr el vehículo en un juego de rodillos de un ambiente estático (como en el caso del dinamómetro de chasis), el vehículo se hace correr en un camino libre y plano.

Los dinamómetros de remolque cargan el motor aplicando una fuerza al vehículo (figura II.7). Este tirón es utilizado para simular un ciclo de paseo montañés mientras se maneja en un lugar plano que cuente con trayectorias rectas y curvas. Los dinamómetros de remolque casi siempre son usados exclusivamente por fabricantes de vehículos durante el proceso de desarrollo de prototipos, por esta razón el vehículo bajo prueba normalmente se instrumenta con sensores.

Figura II.7. Aspecto de un dinamómetro de remolque.

Como se observó, existen varios tipos de dinamómetros que con la ayuda de un analizador de gases nos pueden proporcionar las mediciones deseadas, es por esto que en el siguiente capítulo se explicará de forma general en que consisten los analizadores de gases.

CAPÍTULO III

ANALIZADOR DE GASES

El objetivo de este tercer capítulo es darle al lector una idea más clara de lo que es un analizador de gases, para qué sirve, y cuál es su funcionamiento.

III.1. INTRODUCCIÓN

El analizador de gases sirve para medir los diversos contaminantes del aire en el gas del escape de un automóvil. El análisis de estas mediciones indicará si éste gas contiene contaminantes en exceso o también si el motor está mecánicamente correcto y operando de manera adecuada. Se emplea en pruebas de medición de los gases que emanan del motor de combustión interna. Estas pruebas pueden ser aplicadas tanto a motores colocados en un automóvil como en motores que se prueban por separado. Las mediciones en el tubo de escape pueden ayudar a determinar el comportamiento del motor, el rendimiento, el encendido, el sistema de combustible y el control de emisiones.

Los analizadores de gases automotrices miden generalmente las concentraciones de 2 a 5 gases. El analizador de dos gases mide únicamente HC y CO. El analizador de cuatro gases mide HC, CO, O_2 y CO_2. Un analizador de cinco gases incluye en sus mediciones los cuatro anteriores y el NO_x.

En nuestro caso, el analizador de gases con el que se cuenta en el LCE puede medir los cinco gases: oxígeno, O_2 [%]; monóxido de carbono, CO [%]; dióxido de carbono, CO_2 [%]; hidrocarburos, HC [ppm] y óxidos de nitrógeno, NO_x [ppm].

III.2. PRINCIPIO DE FUNCIONAMIENTO

El analizador de gases utiliza para las mediciones una técnica de análisis denominada “Infrarrojo No Dispersivo” (NDIR: Non Dispersive Infrared) que le permite determinar la concentración de CO, HC y CO_2 como se explica más adelante. Para las mediciones como el O_2 y los NO_x se deben utilizar sensores electroquímicos (figura III.1). Estos sensores presentan una señal eléctrica que es proporcional a la concentración de estos gases en la muestra. Es necesario agregar dichos sensores externamente.

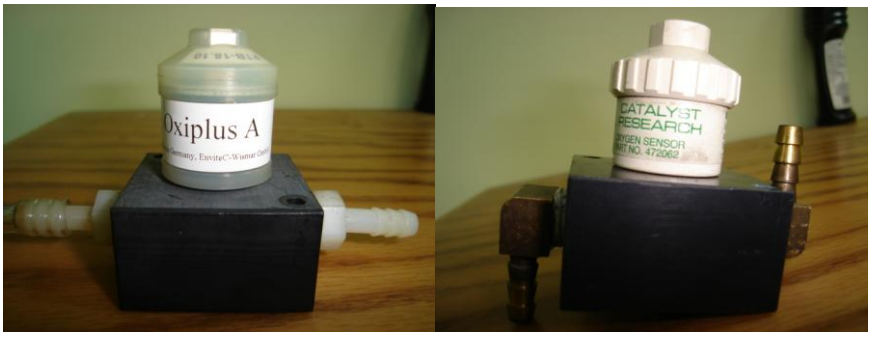


Figura III.1. Sensores para la medición de O_2 y NO_x.

III.2.1. Infrarrojo no dispersivo

Como ya se comentó, el analizador está basado en una tecnología NDIR. Mediante una celda infrarroja se pueden detectar las concentraciones de gases dentro de la muestra, utilizando la técnica de absorción de ondas infrarrojas. El principio de operación del método NDIR es el siguiente: se evalúa la cantidad de luz que absorbe un gas al ser expuesto a una fuente luminosa infrarroja. La concentración de un gas se encuentra en función de la cantidad de moléculas que tiene por unidad de volumen, cuando se hace pasar un haz de luz infrarroja a través de este gas, la absorción de luz infrarroja aumenta considerablemente cuando la concentración de la muestra de gas aumenta por lo que la intensidad de luz medida por un detector disminuye.

En algunos casos, los analizadores de gases poseen dos cámaras que funcionan por separado. En una de las cámaras se encuentra un gas de referencia, y en la otra se hace circular el gas proveniente de la muestra, permitiendo hacer una comparación de información utilizando la misma fuente luminosa. El sistema tiene que realizar varios muestreos para obtener valores diferenciales en las mediciones. Por lo tanto, se puede decir que a diferentes frecuencias de la fuente luminosa es posible obtener una respuesta para cada tipo de gas, permitiendo conocer la concentración del mismo, e identificarlo[1].

Un sólo ancho de banda dentro del espectro infrarrojo es seleccionado para cada gas a medir (figura III.2), donde su absorción es conocida sustancialmente y donde ningún otro gas se absorbe significantemente[2].

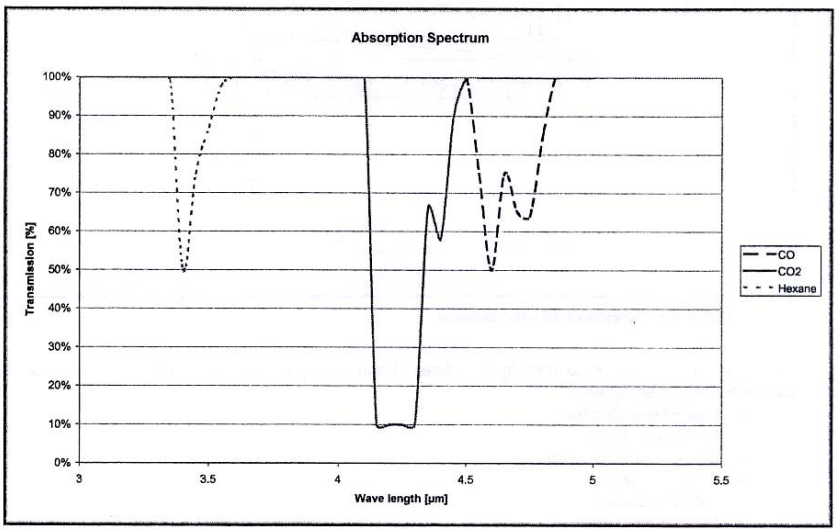


Figura III.2. Espectro de Absorción.

[1] "Instrumento virtual para análisis de gases de combustión", Santiago Cruz Lauro y Rincón Gómez Pedro Ignacio, Laboratorio de Control de Emisiones, Facultad de Ingeniería, UNAM.
[2] Manual Sensors Hardware Interface AMB II 9270-054.

El sistema de medición NDIR incluye los siguientes elementos:

- *Fuente infrarroja*: Esta fuente produce un amplio rango de luces con frecuencias que abarcan desde el espectro infrarrojo hasta el visible.

- *Celda de muestras*: Es el medio en el que el gas va a ser medido, la celda permite el paso de la luz infrarroja a través de la muestra.

- *Filtro infrarrojo*: Se usa para seleccionar una cierta longitud de onda de luz infrarroja. El filtro de longitud de onda está basado en el gas que va a medirse, es por esto que para medir la concentración del CO, HC y CO2 se usan filtros por separado.

- *Detector infrarrojo*: La luz infrarroja que no es absorbida por la muestra del gas se transmite a un detector infrarrojo. En este sensor se produce un voltaje de salida que es proporcional a la intensidad de luz medida.

III.3. RUTINAS DE CALIBRACIÓN Y ZERO

El analizador de gases es capaz de ejecutar las rutinas de calibración y zero (o también llamado autozero o zeroing).

El proceso de calibración se refiere a ajustar, con la mayor exactitud posible, las indicaciones de un instrumento de medida con los valores de la magnitud que ha de medir, para este caso, ajustar la medida de cada gas con una muestra patrón.

En general, cuando un equipo proporciona datos erróneos o exagerados en la medición de los gases, resulta necesario realizar la calibración del mismo.

En el caso de los analizadores de gases, la muestra patrón se obtiene de un tanque especial (figura III.3a), en el cual se indica el porcentaje y número de partículas que contiene de CO, CO_2, HC y NO_x (figura III.3b).

(a) *(b)*

Figura III.3. Tanque de Calibración.

El proceso de zero o zeroing es una calibración pero con el aire del medio ambiente. Al realizar el zero se calibra el O_2. Siempre que se enciende el sistema se recomienda realizar un zeroing para que las mediciones que se obtengan posteriormente sean confiables.

III.4. DESCRIPCIÓN DEL PROYECTO

El analizador propuesto se diseñó e integró de tal forma que pudiera cubrir con las necesidades requeridas en el Laboratorio de Control de Emisiones y con el fin de poder ser utilizado conjuntamente con los demás sistemas empleados en el laboratorio.

El diseño del analizador de gases se divide en dos partes fundamentales: el hardware y el software.

Para lo que se refiere al hardware, el analizador de gases fue implementado alrededor de una banca analizadora de gases comercial denominada microbanca, que junto con otros elementos le permiten obtener, tratar y transportar la muestra para su análisis. El hardware se explica a detalle en el capítulo IV.

Con lo que respecta al software, éste es el que controla al analizador de gases y fue creado utilizando el entorno de programación LabVIEW 8.0, que puede ser adaptado a las exigencias que se requieran. Con este entorno, como se verá más adelante, se crearán lo que se llaman instrumentos virtuales, que en su conjunto darán lugar al instrumento virtual analizador de gases y a la interfaz que consta de un panel de control virtual con el que tiene interacción el usuario final. El desarrollo del software se describe a detalle en el capítulo VI.

El analizador de gases es controlado desde la computadora a través de un puerto serial asíncrono. La PC se encarga de procesar los datos leídos, provenientes de la microbanca y los interpreta como concentraciones de gases. El protocolo de comunicación es simple y amplio permitiendo que la PC controle la actividad de la microbanca. El control del analizador se realiza por medio de comandos.

En el siguiente capítulo se abordará el tema del hardware propuesto para el analizador de gases, en donde se podrán observar cada uno de sus componentes.

CAPÍTULO IV

DESCRIPCIÓN DEL HARDWARE

Con el fin de que el lector entienda de una forma más clara el diseño y la estructura del analizador de gases, en el presente capítulo se describe con detalle cada uno de los módulos que conforman el hardware del mismo.

IV.1. DESCRIPCIÓN DEL HARDWARE POR MÓDULOS

Para poder realizar una descripción más fácil del hardware del analizador de gases, se ha dividido por módulos (figura IV.1), siendo estos los siguientes:

- Módulo de obtención de la muestra
- Módulo de transporte de la muestra (sonda de muestreo)
- Módulo de eliminación de partículas de agua
- Módulo de filtrado de partículas sólidas suspendidas
- Módulo de desvío de gases
- Módulo de bombeo de gases muestra
- Módulo anti-retorno de muestra
- Módulo de regulación de flujo
- Módulo de análisis de concentraciones de gases contaminantes
- Módulo de suministro eléctrico

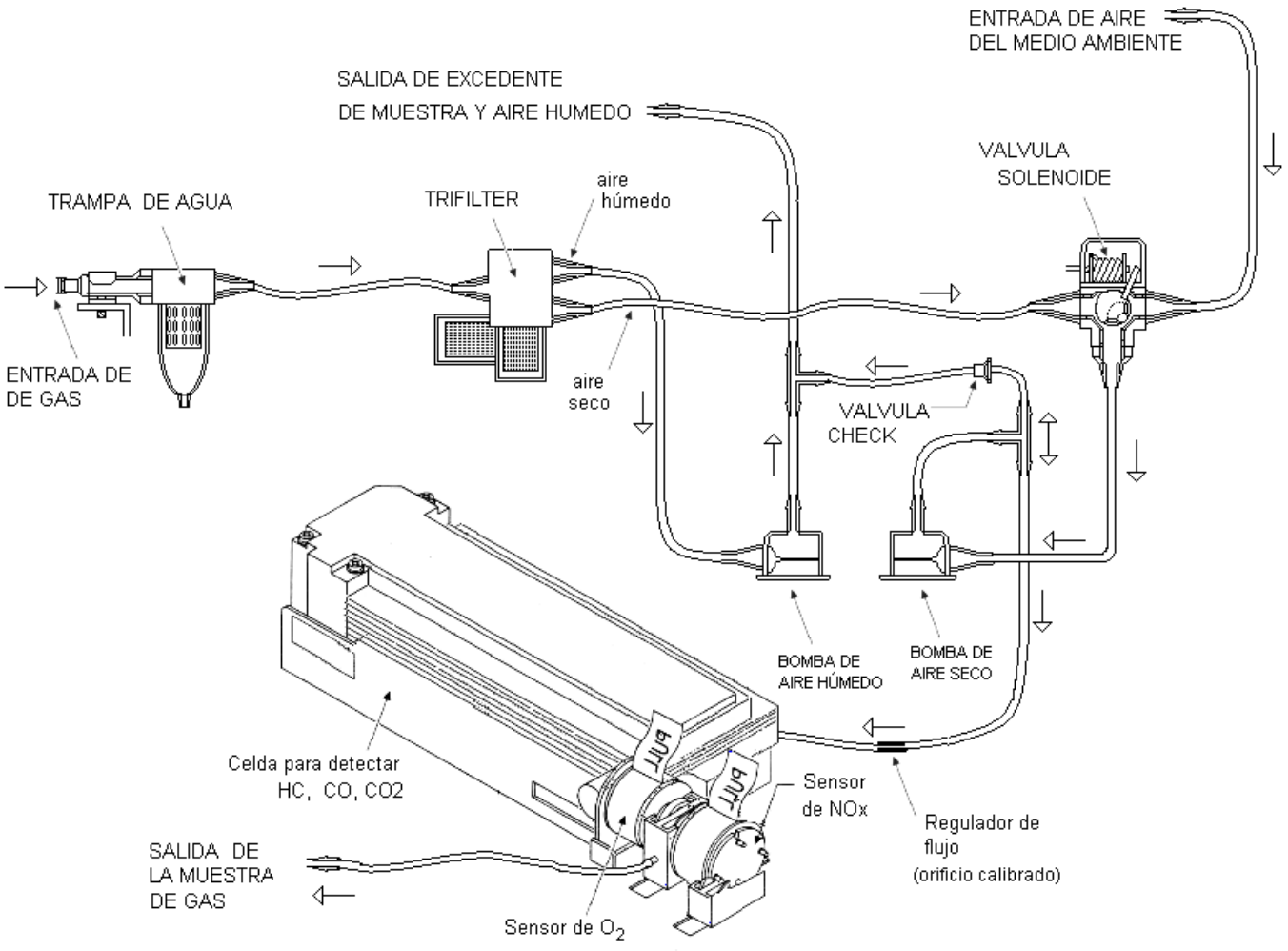


Figura IV.1. Módulos que conforman el sistema analizador de gases.

En los siguientes subtemas serán descritos los módulos antes mencionados, así como su función dentro del analizador de gases.

IV.1.1. Módulo de obtención de la muestra

El módulo de obtención de la muestra, como su nombre lo dice, se encarga de obtener la muestra a analizar del escape del automóvil. Esto se realiza por medio de una "sonda". Una sonda es un objeto cuya misión es llegar a un objetivo prefijado y realizar algún tipo de acción o mandar información. En la figura IV.2 se observan dos tipos de sondas utilizadas para esta tarea. Un extremo de la sonda va dentro del escape del auto y el otro va a la entrada de la trampa de agua.

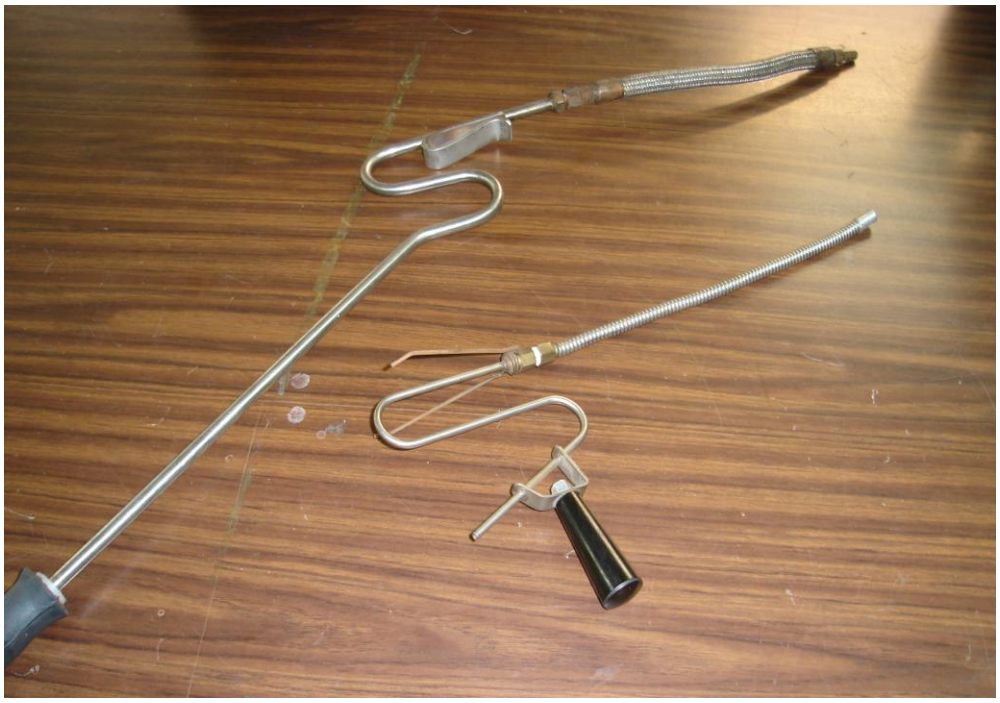


Figura IV.2. Sondas para obtención de muestras.

IV.1.2. Módulo de transporte de la muestra (sonda de muestreo)

La sonda de muestreo es una manguera especial y construida con plásticos que no despiden hidrocarburos. Estas mangueras son muy importantes ya que gracias a ella la muestra puede ser transportada de un módulo a otro, hasta llegar al módulo donde se encuentra la microbanca analizadora de gases. La foto de las sondas utilizadas se puede ver en la figura IV.3.

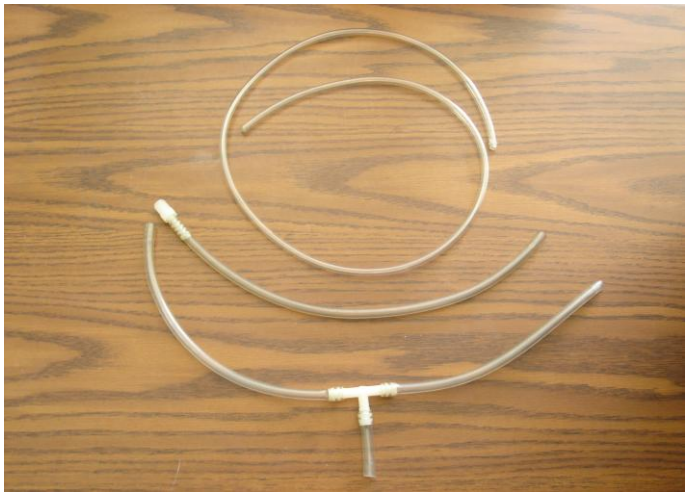
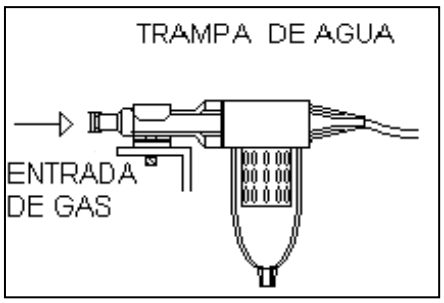


Figura IV.3. Sondas para la transportación de la muestra.

IV.1.3. Módulo de eliminación de partículas de agua

Como se aprecia en la figura IV.1, la muestra de gas entra directamente a la "Trampa de Agua", la cual forma parte de la etapa de filtrado. La trampa de agua (figura IV.4a) es un dispositivo encargado de separar todas aquellas moléculas de agua que se encuentran suspendidas dentro de la muestra de gas que se va a analizar. La foto de la trampa de agua utilizada es la mostrada en la figura IV.4b y IV.4c.



(a)

(b)

(c)

Figura IV.4. Trampa de agua.

Este elemento permite condensar las partículas de agua para separarla de la muestra de gas y es de suma importancia ya que la eliminación de dichas partículas es indispensable para que las lecturas provenientes de los sensores sean correctas. Las moléculas de agua que vienen dentro de la muestra pueden alterar de una forma considerable las lecturas requeridas.

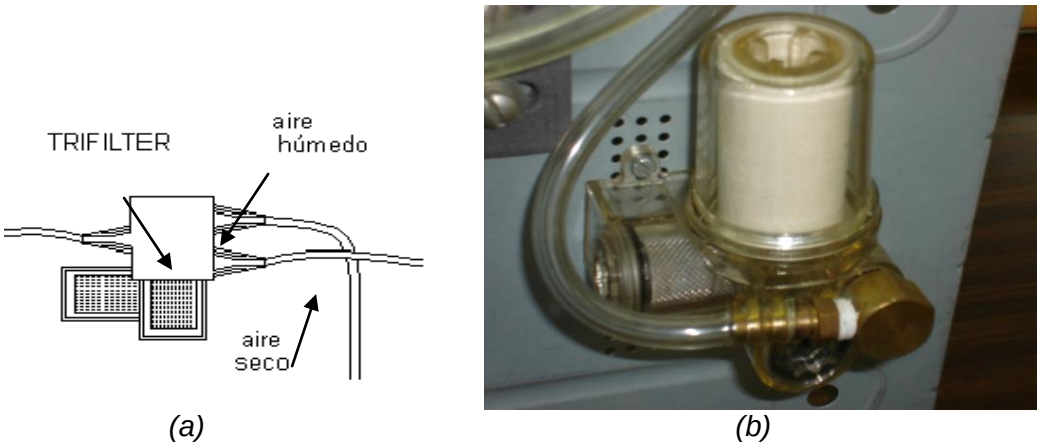
IV.1.4. Módulo de filtrado de partículas sólidas suspendidas

El módulo de filtrado de partículas sólidas suspendidas también forma parte del filtrado de la muestra de gas, y está compuesto de un dispositivo llamado “Trifilter”, el cual consta de un filtro de partículas sólidas suspendidas, que se encarga de eliminar todos aquellos residuos que lleva la muestra de gases y que no deben entrar en contacto con las celdas sensibles del analizador, puesto que son altamente corrosivas.

El trifilter está acoplado a dos condensadores, los cuales se encargan de eliminar cualquier molécula de agua que no pueda ser filtrada por la trampa de agua, que como se mencionó antes puede causar errores en las lecturas que se requieran.

Como se aprecia en la figura IV.5a, el módulo trifilter posee una entrada y dos salidas; por la entrada pasa la muestra de gas que proviene de la salida de la trampa de agua, en una de las salidas se obtiene gas seco, por lo que se conoce como “Salida de Gases Secos”, y en la otra salen gases húmedos, conocida como “Salida de Gases Húmedos”.

La muestra de la salida de gases secos es la que se utiliza para hacer el análisis de los gases, debido a que se trata de una muestra libre de partículas sólidas, y de moléculas de agua. La foto del trifilter utilizado en el proyecto es la mostrada en la figura IV.5b.



(a) *(b)*

Figura IV.5. Trifilter.

En la figura IV.6a y IV.6b se tiene la foto de algunos filtros de partículas sólidas suspendidas que van dentro del trifilter. Se puede observar claramente la diferencia de cuando el filtro está nuevo (el que está de color blanco) y cuando ya ha sido usado cierto tiempo (el que está negro). Obviamente el que está negro es por las partículas sólidas que quedan atrapadas.

(a) *(b)*

Figura IV.6. Filtros de partículas sólidas suspendidas para el trifilter.

Cabe mencionar que en el caso del proyecto, adicionalmente se le agregó otro filtro (figura IV.7), para asegurarse de que la muestra del gas está preparada para ser analizada.

Figura IV.7. Filtro adicional.

IV.1.5. Módulo de desvío de gases

Debido a que el sistema analizador de gases requiere ser calibrado para tener lecturas más confiables, se requiere una entrada directa de aire del medio ambiente. Es por esta razón que se implementó el módulo de desvío de gases que consiste en una válvula

solenoide. La válvula solenoide es un dispositivo de aplicaciones neumáticas, que se encarga de seleccionar el origen de la muestra de gases que será analizado.

En la figura IV.8a se puede observar que la muestra puede ser seleccionada de dos sitios posibles: cuando se requiere hacer el análisis de gases producto de una combustión, se selecciona la muestra proveniente del sistema de filtrado, en donde la muestra después de pasar por la trampa de agua y el trifilter, entra a la válvula solenoide para ser canalizado a la banca analizadora de gases; y el segundo caso, es cuando se desea calibrar el sistema, aquí se hace uso de una muestra proveniente de la entrada directa de aire del medio ambiente, que es seleccionada por la válvula solenoide. La válvula solenoide permite pasar sólo una entrada a la vez.

La foto de la válvula solenoide utilizada es la mostrada en la figura IV.8b.

(a) *(b)*

Figura IV.8. Válvula solenoide.

Dado que la válvula solenoide va a ser controlada por las salidas de la banca, las cuales a su vez serán dirigidas desde la computadora por medio del comando "Lectura/Escritura de Entradas/Salidas (Read/Write I/O's)", y como las salidas de la banca son valores TTL (0 a 5 [V]), y la válvula para funcionar requiere de 12 [V], entonces se tuvo que utilizar un circuito que logrará activar un dispositivo de 12 [V] cuando sólo se tienen 5 [V].

Para este circuito se hizo uso de un relevador RIM-ODC5 de NTE Electronics, Inc. (figura IV.9), el cual tiene como entrada de 3.0 a 8 VDC y una salida de hasta 60 VDC a 3 [A].

Figura IV.9. Relevador.

El circuito utilizado se muestra en la figura IV.10.

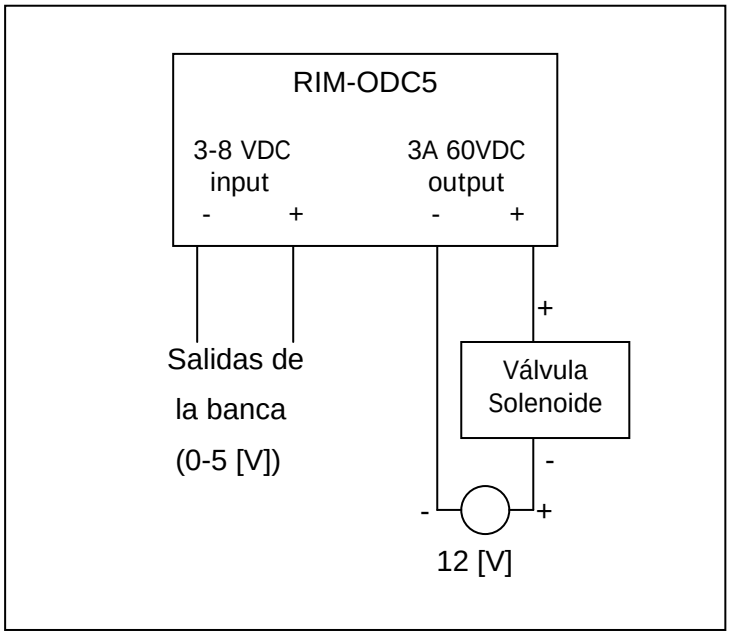
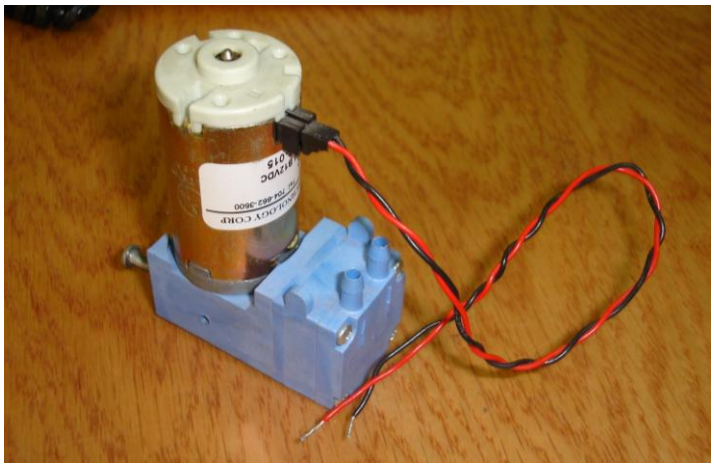


Figura IV.10. Circuito para el control de la válvula solenoide.

IV.1.6. Módulo de bombeo de gases muestra

El módulo de bombeo de gases muestra está conformado de dos “bombas neumáticas”. Una de ellas es la que se encarga de llevar la muestra de gases húmedos junto con excedentes de muestra, proveniente del trifilter hacia el exterior de todo el sistema analizador de gases (figura IV.11a); y la otra es la que transporta la muestra de gases secos o del medio ambiente, provenientes de la válvula solenoide, hacia la banca analizadora de gases, como se observa en la figura IV.11b. La foto de las bombas que se usaron es la indicada en la figura IV.11c.

(a) *(b)*

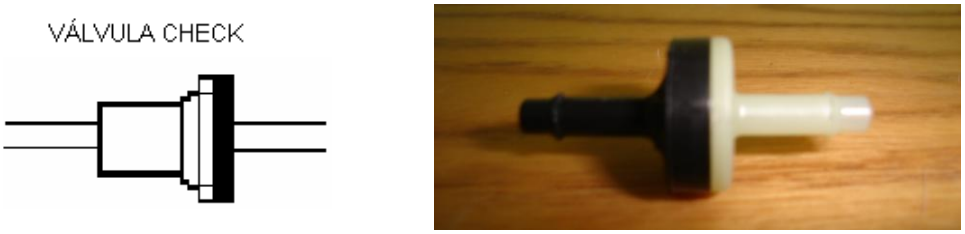


(c)

Figura IV.11. Bomba neumática.

IV.1.7. Módulo anti-retorno de muestra

El dispositivo que conforma el módulo anti-retorno de muestra es una “válvula check”, y dentro del diseño se encuentra a la salida de la bomba neumática que transporta los gases secos, conectándose con la salida de los gases húmedos. La válvula check (figura IV.12a) se utiliza para transportar en una sola dirección las muestras de gases, evitando que se regresen dentro de la banca analizadora de gases, y se garantiza una muestra de gases correcta para el análisis. La foto de la válvula check que se usó es la indicada en la figura IV.12b.



(a) (b)

Figura IV.12. Válvula check.

IV.1.8. Módulo de regulación de flujo

El módulo de regulación de flujo consta de un pequeño orificio calculado también llamado orificio calibrado (figura IV.13a), el cual se encarga de regular el flujo de la muestra de gases, ya que para la microbanca un flujo excesivo de la muestra de gases puede causar lecturas erróneas y por lo tanto un reporte del análisis falso. En la figura IV.13b se muestra la foto de dos orificios calibrados.

(a) *(b)*

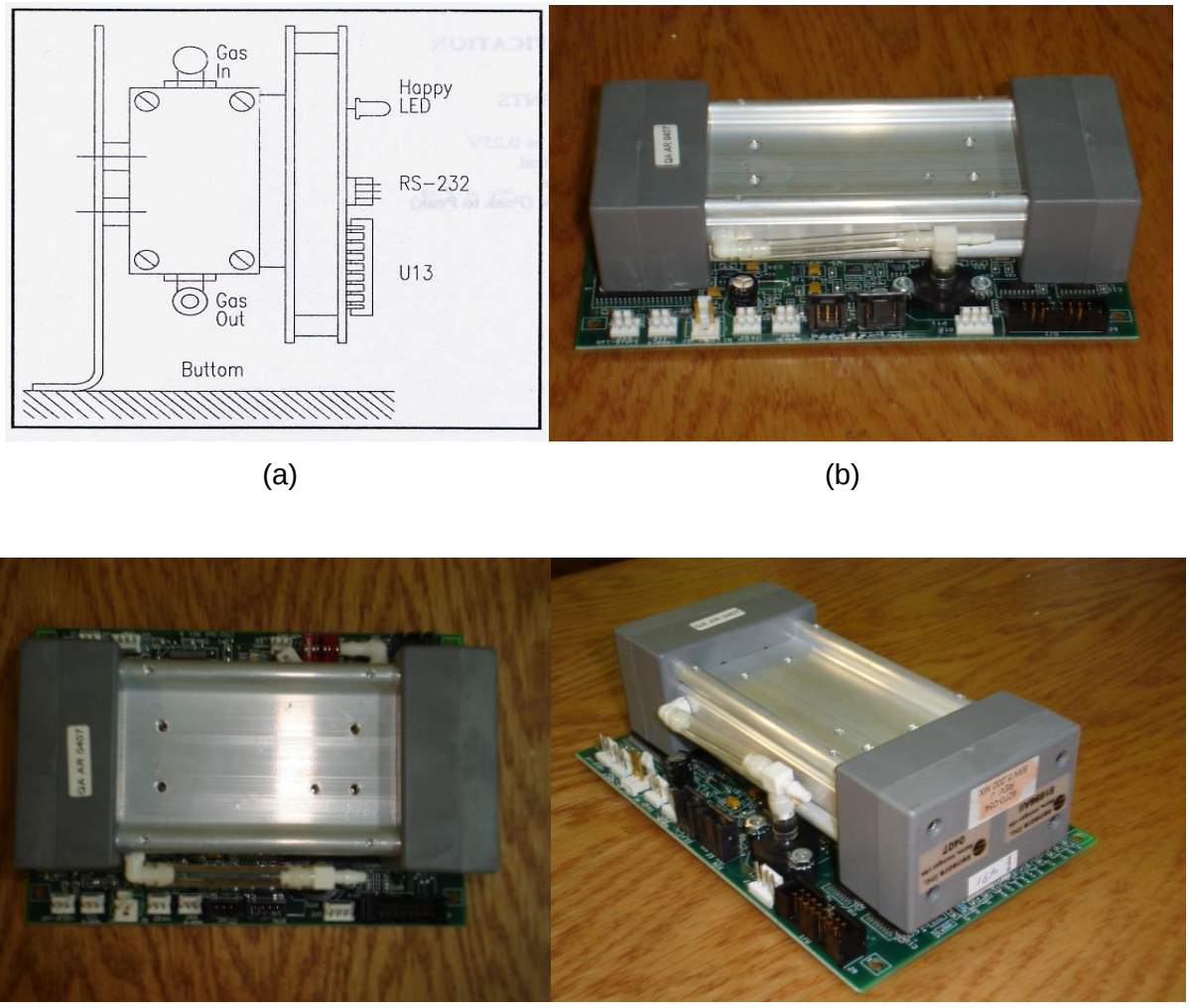
Figura IV.13. Regulación de flujo.

IV.1.9. Módulo de análisis de concentraciones de gases contaminantes

El dispositivo que conforma el módulo de análisis de concentraciones de gases contaminantes se conoce como “banca analizadora de gases o microbanca”, en este caso se hizo uso de una banca modelo AMB II 9270-054 de Sensors, inc. (figura IV.14a), la cual está conformada de un pequeño módulo de detección de gases, diseñado para una fácil incorporación en diferentes aplicaciones automotrices, y que reduce el consumo de baja energía de 10 [W] con una sola fuente de alimentación de 5 [V].

La microbanca puede medir específicamente Monóxido de Carbono (CO), Dióxido de Carbono (CO_2) e Hidrocarburos como el Propano (C_3H_8) o Hexano (C_6H_{14}), y si se le implementan sensores electroquímicos también puede medir concentraciones de Oxígeno (O_2) y Óxidos de Nitrógeno (NO_x).

Las fotos de la banca desde distintos ángulos se muestran en la figura IV.14b, IV.14c y IV.14d.



(a) (b)

(c) (d)

Figura IV.14. Banca Analizadora de Gases marca Sensors AMB II.

IV.1.9.1. Especificaciones de la banca analizadora de gases

La banca analizadora de gases Sensors AMB II 9270-054 cuenta con las especificaciones[1], mostradas en la tabla V.1.

[1] Manual Sensors Hardware Interface AMB II 9270-054.

Temperatura de Almacenaje	-50 [°C] a 70 [°C]
Temperatura Operacional	-12 [°C] a 48[°C]
Temperatura del Gas	T_{gas} aproximadamente igual a la $T_{ambiente}$
Gases Medidos	HC Hidrocarburos como C_6H_{14} Hexano CO Monóxido de Carbono CO_2 Dióxido de Carbono HC Hidrocarburos como C_3H_8 Propano O_2 Oxígeno (opcional a través de un sensor electroquímico) NO_X Óxido Nítrico (opcional a través de un sensor electroquímico)
Márgenes	HC = 0 a 2000 [ppm] Hexano (Resolución estándar) = 0 a 4000 [ppm] Propano (Resolución estándar) = 0 a 3276.7 [ppm] Hexano (Alta Resolución) = 0 a 3276.7 [ppm] Propano (Alta Resolución) CO = 0 a 15% CO_2 = 0 a 20% HC = 0 a 20000 [ppm] Hexano (Sólo resolución estándar) = 0 a 40000 [ppm] Propano (Sólo resolución estándar) O_2 = 0 a 250% NO_X = 0 a 5000 [ppm] (Como Óxido Nítrico)
Resolución	HC: 1 [ppm] vol. (Resolución estándar) 0.1 [ppm] vol. (Alta resolución) CO: 0.01% vol. (Resolución estándar) 0.001% vol. (Alta resolución) CO_2: 0.1% vol. (Resolución estándar) 0.01% vol. (Alta resolución) O_2: 0.01% vol. NO_X: 1 [ppm]
Tiempo de Respuesta	T_{90} = 3.5 [s]
Taza de Flujo	0.3... 6.0 [l/min]
Tamaño de Partícula	< 5 [µm]
Presión Operativa	750 [mbar] – 1100 [mbar] 1000 [mbar] nominal
Requerimientos de Energía	Voltaje de fuente: 5 [V] DC ± 0.25 [V]. Corriente: 2 [A] típico. Máximo 3 [A]. Potencia: Típica 10 [watts], máxima 15 [watts] Ondulación: 150 [mV] (pico a pico). Comunicación: Vía RS-232.

Tabla IV.1. Especificaciones de la banca AMBII.

A continuación se mencionan algunos aspectos importantes de la banca como son lo que es la advertencia interna, el calentamiento interno, los conectores con los que cuenta.

Advertencia interna

La bandera de advertencia interna indica si la banca está funcionando correctamente o no, para visualizar esta bandera se puede invocar el comando “Get data” explicado en el Capítulo VI en el apartado VI.1.4.2. Esto también se puede observar físicamente en la banca, la banca cuenta con un led llamado “Happy LED”, el cual parpadea cambiando de color verde a amarillo y viceversa, cuando la bandera de advertencia interna está activa, el “Happy LED” está parpadeando con una frecuencia de 5 [Hz], es decir que no esta trabajando correctamente. La frecuencia de parpadeo que indica que la banca está funcionando bien es de 1 [Hz].

Calentamiento interno

Cuando la banca analizadora se conecta a la energía eléctrica, inmediatamente comienza su calentamiento interno. La banca para poder realizar mediciones exactas, precisas y confiables requiere de un previo calentamiento interno. El tiempo que falta para que la banca ya se haya calentado se puede observar con el comando “Get Status” en el Capítulo VI apartado VI.1.4.3. Esto es posible dado que la banca tiene un contador de calentamiento, el cual tiene como propósito predecir el tiempo de calentamiento restante para la banca.

El tiempo de calentamiento se reporta en segundos y es continuamente contado. Después de que el contador ha empezado a contar, éste no puede ser restablecido – el contador puede ser sólo decrementado.

Conectores

A continuación se describen brevemente cada uno de los conectores que conforman la banca analizadora:

- ***Transductor NOX (J1):*** Es una entrada designada a conectar directamente un transductor electroquímico de NO a la banca.

- ***NOX (J3):*** Este conector es usado en conjunción con los múltiples sensores de NO_X y el transductor de NO_X.

- ***O2 (J4):*** Este conector es usado para conectar un sensor electroquímico de O_2 a la banca.

- ***RPM Analógico (JA4):*** Este conector es usado cuando se requiere la conversión Analógico/Digital por aplicaciones específicas del cliente.

- ***Temperatura del aceite (J5):*** Este conector es usado en conjunción con los sensores RPM/OIL-Temperature. Si los sensores no son usados, este conector A/D puede ser utilizada para aplicaciones específicas del cliente.

- ***Humedad Relativa (JA5):*** Este conector Analógico/Digital es usado para aplicaciones específicas del cliente.

- ***RS232 (J7):*** Este conector es usado para la comunicación entre la computadora huésped y la microbanca AMB II de Sensors.

- ***Encendido (J8):*** Este conector sirve para encender la microbanca, requiriendo una fuente 5[V] a 3 [A].

- ***Muestra I/O (J9, J10):*** Estos conectores leen las entradas digitales o escriben las salidas digitales y pueden ser usadas para controlar las bombas del sistema.

- ***Entrada Digital de RPM (J11):*** Esta entrada está designada para la lectura de señales en un intervalo de frecuencias entre 1 [Hz] y 1 [KHz]; además de que puede ser usada para leer RPM.

- ***Temperatura Externa (J12):*** Este conector sirve como entrada Analógica/Digital, puede ser usada para aplicaciones específicas que requiera el cliente.

IV.1.9.2. Protocolo de comunicación

La comunicación entre la banca analizadora y la computadora es por medio del puerto serial asíncrono RS-232, del cual sólo se utilizan 3 terminales, los correspondientes al receptor (RxD), al transmisor (TxD) y a la tierra (DGND).

La comunicación serial para el caso de la banca AMB II 9270-054 tiene los siguientes parámetros:

Velocidad de transferencia de datos:	9600 baudios
Bits de paro:	1
Bits de comienzo:	1
Bits de datos:	8
Paridad:	Ninguna

El tiempo que transcurre cada que se envía una cadena compuesta por el bit de comienzo, los bits de datos y el bit de paro es de 50 [ms].

IV.1.10. Módulo de suministro eléctrico

El módulo de suministro eléctrico es el que se encarga de suministrar corriente a dispositivos que se encuentran dentro analizador de gases. Cuenta con dos fuentes de corriente directa (figura IV.15).

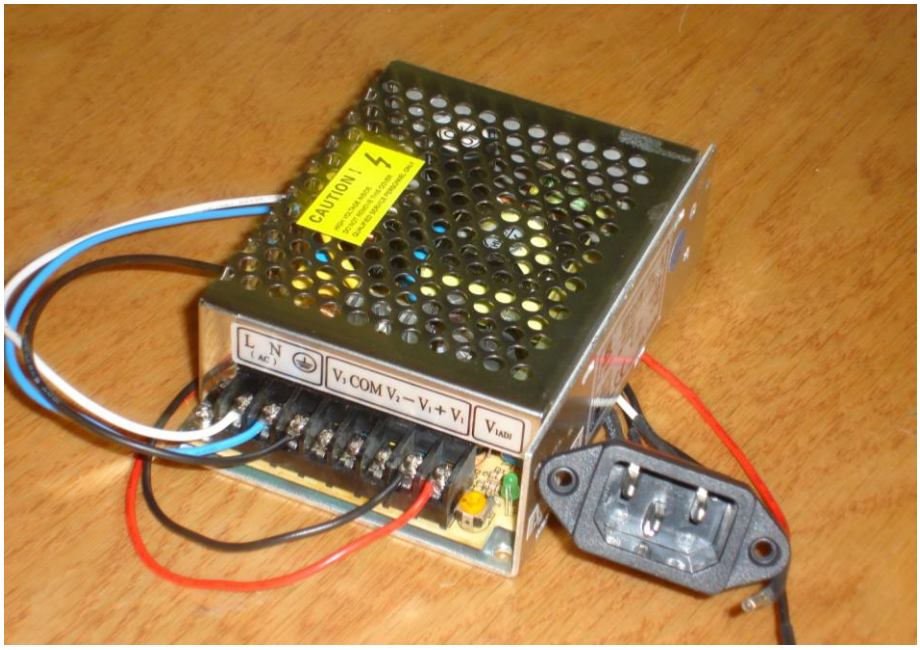


Figura IV.15. Fuente de corriente directa.

Una fuente da una tensión eléctrica con salida de voltaje de 5 [V] y de 12 [V] a 1 [A]; los 5 [V] se toman para alimentar la banca analizadora y los 12 [V] para los ventiladores. Se debe tener cuidado de que la tensión para alimentar la banca sea la más exacta posible, con

valor de 5 [V], porque si se le proporciona más tensión puede quemarse la banca. Esta tensión se ajusta con un pequeño tornillo que ya trae integrado la fuente.

La otra fuente es la que se encarga de suministrar corriente tanto a las bombas como a la válvula solenoide.

Se hizo uso de dos ventiladores (figura IV.16). Uno de ellos es usado para evitar el calentamiento excesivo de la banca, para mantener su buen funcionamiento y exactitud en las lecturas. El otro es también para evitar el calentamiento, pero en este caso de las fuentes de corriente.

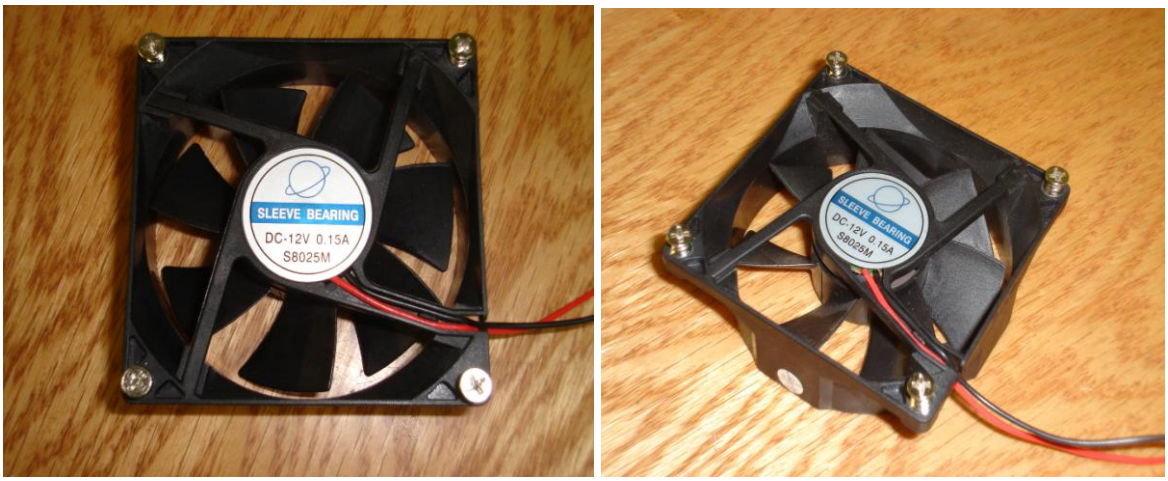
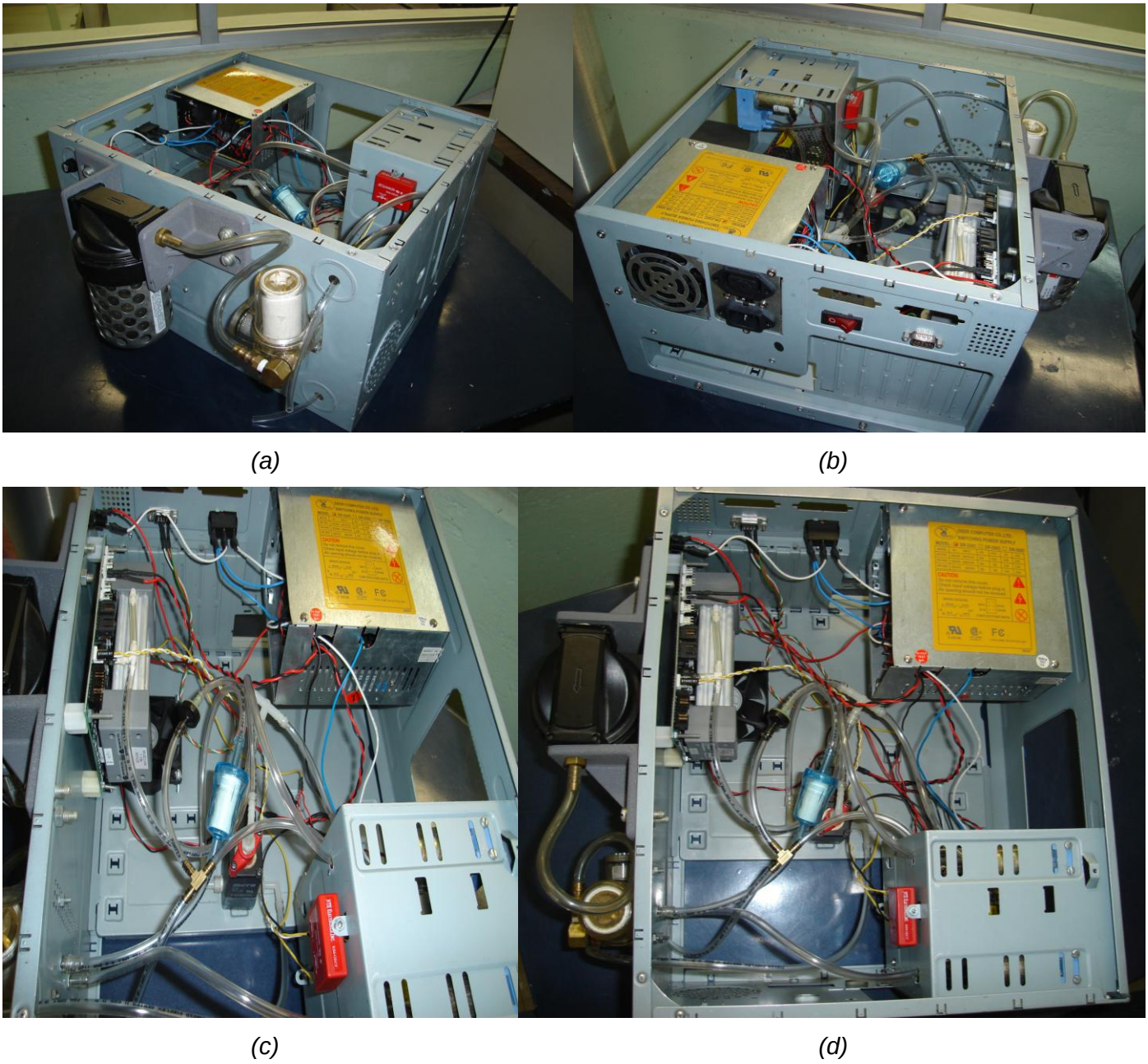


Figura IV.16. Ventiladores.

IV.2. HARDWARE COMPLETO

Ya teniendo todos los elementos por separado, y una vez probado su buen funcionamiento individualmente, se procedió a la integración de los mismos. Para esto se utilizó un gabinete de computadora vacío con el que ya contaba el LCE, al cual se le realizaron las perforaciones necesarias para la colocación de los elementos, y posteriormente se hicieron las conexiones requeridas para el funcionamiento general del sistema analizador de gases.

De un costado se colocaron la trampa de agua y el trifilter (figura IV.17a). En otro costado se puso el interruptor de encendido del equipo, el conector RS-232, y la conexión del cable de la fuente (figura IV.17b). Las fotos de las figuras IV.17c y IV.17d muestran la conexión interna de las mangueras y los cables de alimentación de las fuentes, bombas y válvula utilizadas.



(a) *(b)*

(c) *(d)*

Figura IV.17. Analizador de gases.

Antes de empezar a hablar acerca de los instrumentos virtuales programados para este trabajo de tesis, en el siguiente capítulo se hace la descripción del entorno de programación de LabVIEW 8.0 para que posteriormente, se entienda lo que es un instrumento virtual.

CAPÍTULO V

INSTRUMENTACIÓN VIRTUAL POR MEDIO DE COMPUTADORAS PERSONALES

El presente capítulo permitirá al lector familiarizarse con la terminología que se emplea en la instrumentación virtual. Principalmente se incluye información acerca de la estructura y funcionamiento del entorno de programación LabVIEW.

V.1. LAS COMPUTADORAS PERSONALES Y LA INSTRUMENTACIÓN VIRTUAL

Son muchas las aplicaciones donde se hace indispensable el tratamiento de señales que nos proporcionen información sobre fenómenos físicos. Generalmente es necesario hacer este tratamiento sobre cantidades de información que suelen ser muy grandes para procesarlas manualmente. Por esta razón es necesario utilizar sistemas de cómputo que ofrecen gran velocidad de procesado sobre cantidades elevadas de información.

El control de instrumentos por medio del uso de computadoras personales se remonta a los años 70's, cuando se utilizó por primera vez la *interfaz de bus* IEEE 488 o GPIB (General Pruporse Interface Bus). Sin embargo, es hasta los años 90's, cuando los procesadores de 16 y 32 bits se han incorporado a equipos asequibles, consiguiendo con ello altas velocidades de comunicación de datos y grandes capacidades de memoria. Esta popularización de las computadoras personales de altas prestaciones ha traído consigo un fuerte desarrollo de potentes paquetes de software que simplifican la creación de instrumentos virtuales.

Un ***instrumento virtual*** es un módulo programado que simula el panel frontal de un instrumento real, apoyándose en elementos de hardware accesibles por la computadora personal (tarjetas de adquisición de datos, tarjetas DSP (Digital Signal Processor, Procesador Digital de Señales), instrumentos accesibles vía GPIB, VXI, RS-232), que se comporta aparentemente como si se tratara de un instrumento real. De este modo, cuando se ejecuta un programa que funciona como instrumento virtual o VI (Virtual Instrument), el usuario puede ver en la pantalla de su computadora personal un panel cuya función es idéntica a la de un instrumento físico, facilitando la visualización y el control del aparato. A partir de los datos reflejados en el panel frontal, el VI debe actuar recogiendo o generando señales, como lo haría su homólogo físico.

V.2. EL ENTORNO DE DESARROLLO LABVIEW

Hasta hace poco, la tarea de construcción de un VI se llevaba a cabo con paquetes de software que ofrecían una serie de facilidades, como funciones de alto nivel y la incorporación de elementos gráficos que simplificaban la tarea de programación y elaboración del panel frontal. Sin embargo, el cuerpo del programa seguía basado en texto, lo que suponía mucho tiempo invertido en detalles de programación que en realidad nada tienen que ver con la funcionalidad del VI. Con la llegada de software de programación gráfica LabVIEW de

National Instruments®, Visual Designer de Burr Brown® o VEE de Hewlett Packard®, el proceso de creación de un VI se ha simplificado notablemente, minimizándose el tiempo de desarrollo de las aplicaciones.

LabVIEW es el acrónimo de **Lab**oratory **V**irtual **I**nstrument **E**ngineering **W**orkbech. Es un lenguaje y a la vez un entorno de programación gráfica en el que se pueden crear aplicaciones de una forma rápida y sencilla[1].

V.2.1. Programación modular

Debido al entorno gráfico que se maneja, la programación en LabVIEW se realiza utilizando una estructura *Modular*, en lugar de la estructura *Secuencial* tradicional, que implica una sucesión de órdenes mediante sentencias textuales. La programación modular es un método de diseño que tiende a dividir el problema total en aquellas partes que poseen una identidad propia.

La programación modular, que no es un método exclusivo de programación de LabVIEW, surgió porque frecuentemente deben repetirse una cierta secuencia de sentencias en varios lugares dentro de un programa. Para ahorrar al programador el tiempo y el esfuerzo necesarios para volver a copiar estas sentencias, muchos lenguajes de programación ofrecen una posibilidad denominada subrutina o subprograma. Este mecanismo permite asignar un nombre libremente elegido a una secuencia de sentencias, para poder utilizarlo como una abreviatura en cualquier parte en que aparezca esta secuencia de secuencias. De esa forma un *subprograma* es una parte autónoma del programa que realiza una misión o función bien definida, que puede ser invocada por otras partes del programa siempre que se necesite para desarrollar su función.

En LabVIEW, la utilización de subprogramas simplifica la programación, ya que no existe diferencia entre procedimientos y funciones. Así en LabVIEW, a las aplicaciones que se incluyen dentro de otra aplicación se les llama subVIs. Es importante tener en cuenta que, por sí mismos, los subVIs tienen exactamente las mismas propiedades y cualidades que un VI, ya que de hecho son un VI. Únicamente reciben ese nombre porque son llamados por otro

[1] "LabVIEW Entorno gráfico de programación". José Rafael Lajara Vizcaíno, José Pelegrí Sebastiá. Ed. Alfaomega marcombo.

VI a nivel superior, el cual se encarga de pasar datos al subVI para que los procese, y éste le devuelve al primero los resultados obtenidos.

Una de las grandes ventajas de LabVIEW es la de poder trabajar con aplicaciones totalmente ejecutables dentro de otras, de manera que el programador puede organizar el desarrollo de una aplicación compleja en partes independientes. De este modo resulta mucho más fácil localizar el motivo de un posible fallo.

Para poder utilizar un subVI es necesario crearle un icono y un conector. LabVIEW admite que haya varios subVI's con el mismo icono e incluso mismo conector, diferenciándolos únicamente por el nombre que el programador le haya puesto. Por lo tanto, para hacer más fácil la lectura e interpretación del diagrama de bloques conviene utilizar iconos diferentes para cada VI que se desarrolle con el fin de utilizarlo como subVI.

Hablar de subVI's es análogo a hablar de subrutinas en otros lenguajes de programación, pero con la enorme ventaja de que dichos subVI's son totalmente ejecutables. Es decir, podemos comprobar si funciona correctamente en cualquier momento del desarrollo de nuestra aplicación sin necesidad de tener completo el diagrama de bloques principal.

Un VI no puede usarse recursivamente, esto es, no puede ser su propio subVI, o un subVI de uno de sus subVI y así sucesivamente.

V.2.2. Ventanas panel y diagrama

Cuando se hace un programa en LabVIEW siempre utilizan dos ventanas: una en la que se implementará el "p*anel frontal",* es decir, el tablero de control y despliegue del instrumento que se desea simular; y otra ventana denominada "d*iagrama de bloques",* que soportará el nivel de programación para que el sistema completo tenga funcionalidad.

V.2.3. Controles e indicadores

Para la creación del *Panel Frontal* se dispone de una biblioteca de controles e indicadores de todo tipo (figura V.1), y con la posibilidad de crear más, diseñados por el propio programador.

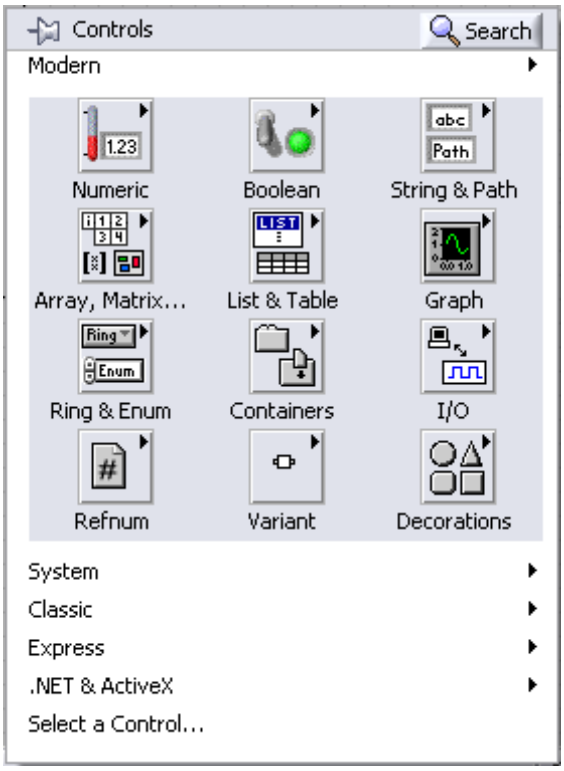


Figura V.1. Menú de controles de LabVIEW 8.0.

En la figura V.1, la flecha ▸ colocada en cada menú indica que éste a su vez tiene un submenú de controles, por ejemplo el menú “Numeric” tiene el submenú indicado en la figura V.2.

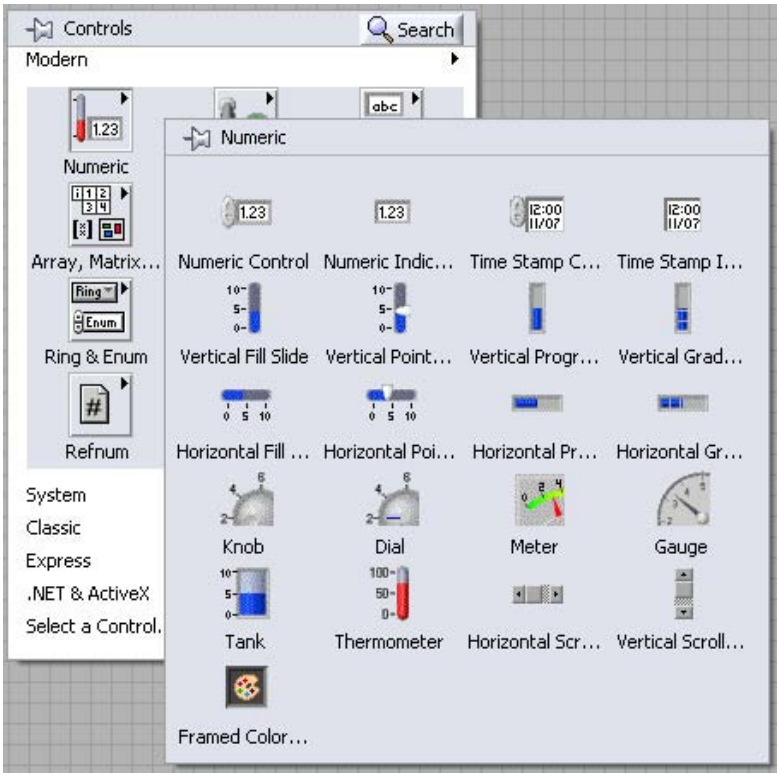


Figura V.2. Submenú de controles para el menú “Numeric” de LabVIEW 8.0.

Cuando un control es colocado desde la *biblioteca*, como parte de un panel virtual, se crea una variable cuyos valores y propiedades vendrán a ser determinados por el programador desde el panel.

V.2.4. Herramientas de LabVIEW

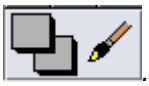
Una herramienta es un modo de funcionamiento especial de ratón. Se usan para llevar a cabo funciones específicas de edición o ejecución. Estas herramientas se encuentran situadas en la llamada “Tools Palette” (figura V.3).

Figura V.3. Herramientas de LabVIEW 8.0.

A continuación se describen cada una de estas herramientas:

- *Operate value (valor operativo)*  . Maneja los controles del panel frontal.
- *Position/Size/Select* (situación/tamaño/selección)  . Selecciona, mueve y redimensiona objetos.
- *Edit text (edición de texto)*  . Crea y edita textos.
- *Connect wire (conexión de cables)*  . Enlaza objetos del diagrama de bloques y asigna a las terminales del conector del VI los controles e indicadores del panel frontal.
- *Object shortcut menu (menú de objeto)*  . Despliega el menú asociado al objeto. Tiene el mismo efecto que si se pulsa el botón derecho del ratón sobre el objeto.
- *Scroll window (desplazamiento de la pantalla)*  . Desplaza la pantalla en la dirección que se desea para ver posibles zonas ocultas.
- *Set/Clear breakpoint (establecer/quitar puntos de ruptura)*  . Permite poner tantos puntos de ruptura como se desee a lo largo del diagrama de bloques. Cuando

durante la ejecución se llega a uno de ellos, LabVIEW conmuta automáticamente al diagrama de bloques.

- *Probe data (sonda de datos)*  . Se utiliza para comprobar los valores intermedios dentro de un VI que es ejecutable pero que genera resultados sospechosos o inesperados.
- *Get color (capturar color)*  . Permite saber de manera específica qué color tiene un objeto, textos u otros elementos.
- *Set color (colorear)*  . Colorea diversos objetos y los fondos.

V.2.5. Interconexión de bloques

El pase de parámetros es a través de las terminales de conexión del bloque donde aparecen un determinado número de terminales definidos en la creación del icono correspondiente al subVI.

En la ventana que soporta el nivel de programación (*diagrama de bloques*) se interconectan las terminales de cada variable involucrada con las herramientas o VI's provenientes de bibliotecas que el sistema tiene, o bien, otros VI's que el programador previamente haya creado.

Para hacer estas conexiones se utilizan cables virtuales y es necesario hacer uso de la herramienta *Connect Wire*  , que como ya se explicó anteriormente, no es más que un cursor con forma de carrete que se puede mover mediante el ratón de la PC y con el cual estos cables se pueden dibujar directamente en pantalla como se aprecia en la figura V.4.

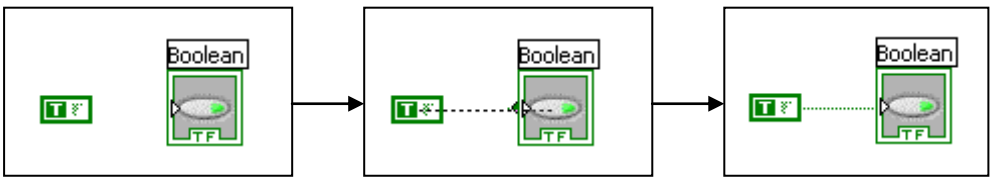


Figura V.4. Proceso de "cableado" virtual.

Se puede comparar la ventana de programación (*diagrama de bloques*) con una placa de circuito impreso, donde las terminales del panel se cablean a bloques funcionales (circuitos integrados) que se interconectan para generar los datos o señales que se desean

visualizar. A su vez, estos circuitos integrados contienen bloques circuitales conectados entre sí en su interior, al igual que un bloque creado en LabVIEW puede estar formado por la interconexión de otros bloques. De esta forma la programación gráfica permite diseñar un VI vertiendo las ideas de funcionalidad directamente a un diagrama de bloques, como se haría al dibujar un diagrama de bloques en un papel.

V.2.6. Ejecución de un programa

Cuando se está listo para probar un VI (es decir, compilar y correr el programa) sólo hay que hacer clic sobre la flecha de ejecución  . Si se requiere detener la ejecución del VI, sólo basta con hacer clic en el botón pause (pausa)  , con lo cual en el diagrama de bloques se queda parpadeando la siguiente secuencia que se ejecutará.

Un VI no se puede compilar o ejecutar si no está completo o si hay un error (flecha del botón Run rota ). Los VIs no están completos mientras se están construyendo hasta que se unen todos los íconos del diagrama de bloques. Si una vez hecho esto continuase la flecha del botón Run rota, quiere decir que algo falta o hay algún error. Para averiguar las razones por las que un VI permanece roto se pueden observar en la llamada Error List (figura V.5).

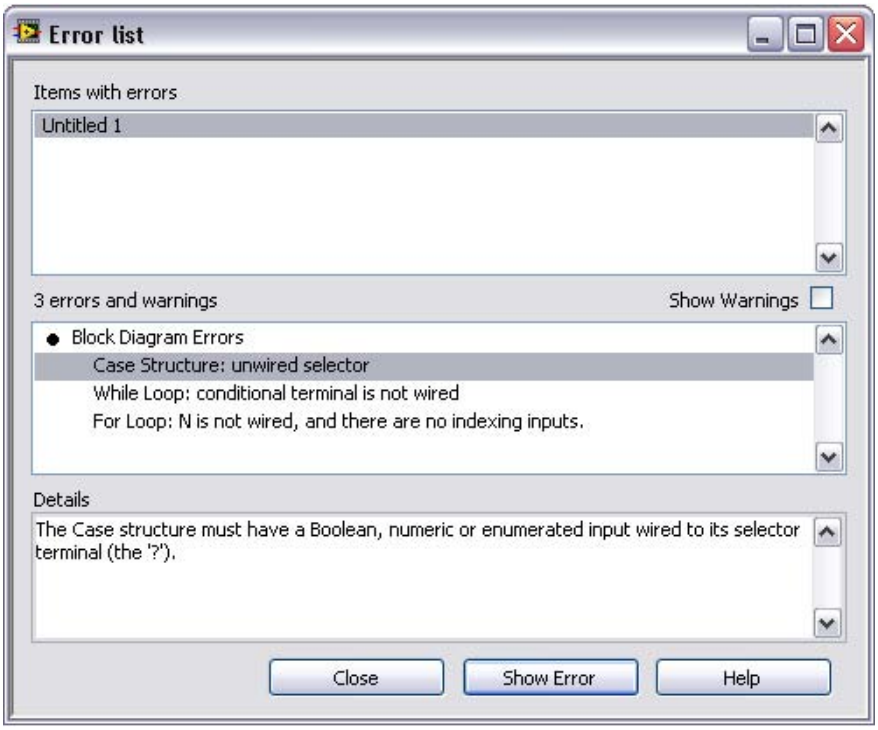


Figura V.5. Lista de Errores en LabVIEW 8.0.

V.2.7. Creación de un Icono

Para crear o modificar un icono se debe dar doble clic sobre el icono de la parte superior derecha de la ventana Panel y aparece la opción Edit Icon (editar icono) como se muestra en la figura V.6a.

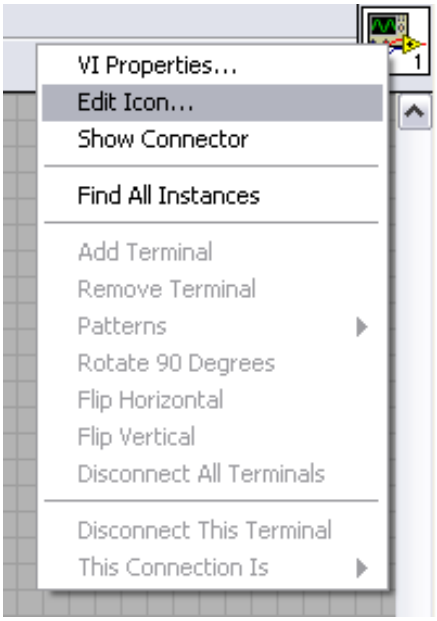


Figura V.6a. Edit Icon en LabVIEW 8.0.

Posteriormente al seleccionar esta opción aparece la ventana de la figura V.6b, en la cual se le puede dar la apariencia que el programador desee.

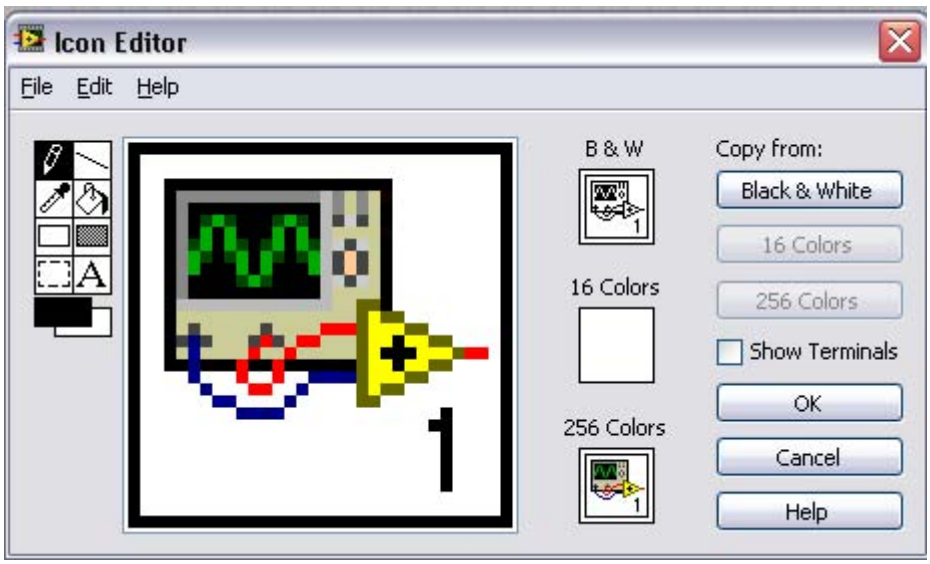
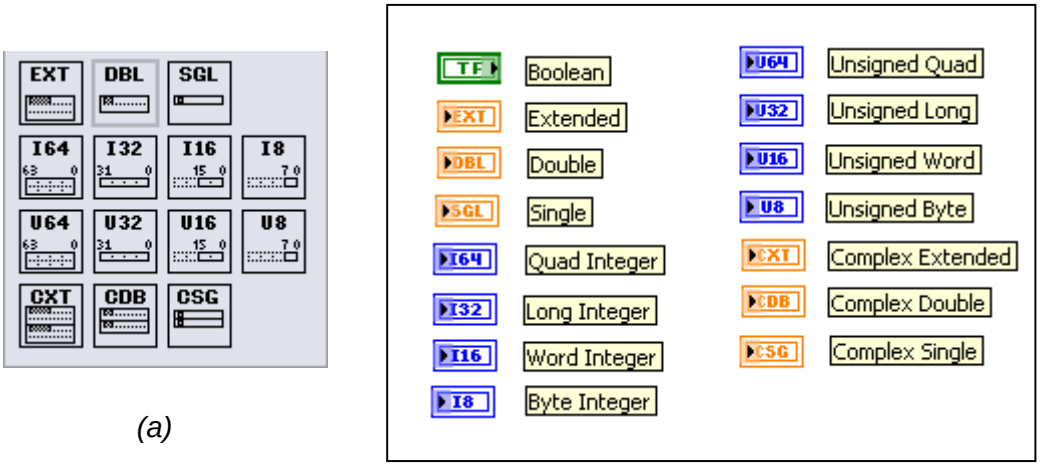


Figura V.6b. Edit Icon en LabVIEW 8.0.

V.2.8. Tipos de Datos

LabVIEW ofrece una gran variedad de tipos de datos. Para esto se basa en asignarle un color propio a cada uno de ellos en el diagrama de bloques. De esta manera, y como consecuencia de una memorización o asimilación práctica, el usuario se facilita la tarea de identificarlos y reconocer inmediatamente si se está trabajando con el tipo de dato adecuado. En la figura V.7a se tiene el cuadro para configurar el tipo de los controles o indicadores; y en la figura V.7b se muestra su correspondiente representación en el diagrama de bloques.

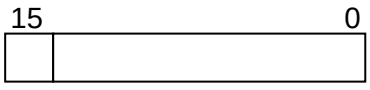


(a) *(b)*

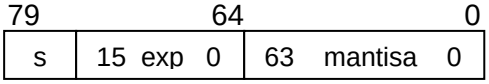
Figura V.7. Símbolos para diferentes tipos de datos.

A continuación se explica cada uno de ellos:

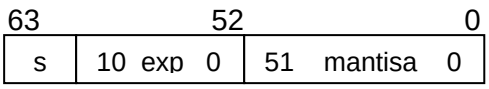
- ***Boolean:*** Se representan en color verde claro y se refiere a los tipos de datos booleanos que en realidad son números enteros de 16 bits. El *bit* más significativo contiene el valor booleano y puede tener valores de falso o verdadero si se encuentra en él un número cero o un uno respectivamente.



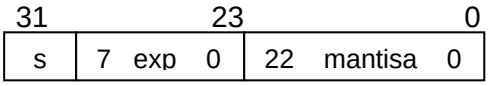
- **Numéricos:** Hay diferentes tipos:
  - **Extended*:*** Se representan en color naranja y presentan el siguiente formato:



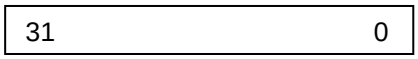
  - **Double:** Se representan en color naranja, cumplen con el formato de doble precisión IEEE de 64 bits. Es el valor por default de LabVIEW.



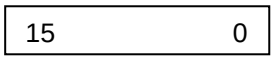
- **Single:** Se representan en color naranja y cumplen con el formato de precisión IEEE de 32 bits.



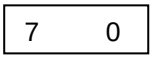
- **Long Integer:** Se representan en color azul y se refiere a los números enteros largos con un formato de 32 bits con o sin signo.



- **Word Integer:** Se representan en color azul y se refiere a los números que poseen un formato de 16 bits con o sin signo.



- **Byte Integer:** Se representan en color azul y se refiere a los números con formato de 8 bits con o sin signo.



- **Unsigned Long:** Se representan en color azul y se refiere a un entero largo de 32 bits sin signo.

- **Unsigned Word:** Se representa en color azul y se refiere a una palabra de 16 bits sin signo.

- **Unsigned Byte:** Se representa en color azul y se refiere a un byte de 8 bits sin signo.

- **Complex Extended:** Se representa en color naranja y se refiere a números complejos con precisión extendida de 32 bits.

- **Complex Double:** Se representa en color naranja y se refiere a números complejos con precisión doble de 16 bits.

- **Complex Simple:** Se representa en color naranja y se refiere a números complejos con precisión simple de 8 bits.

- **Arrays:** Los arreglos dependen del tipo de datos que contenga, así también el color en el representa depende del tipo de datos que contenga. LabVIEW almacena el tamaño de cada dimensión de un arreglo como Long Integer seguido por el dato. Los arreglos booleanos se almacenan de forma diferente, como un paquete de bits y la dimensión del mismo viene dado en bits en lugar de bytes. El bit 0 se guarda en la posición más alta de la memoria, y el bit 15 en la posición más baja.

- **Strings:** Se representan de color rosa y representan cadenas de caracteres. LabVIEW los almacena como si fueran un array uni-dimensional de bytes enteros.

- **Clusters:** Se representan de color marrón o rosa y sirve para almacenar diferentes tipos de datos. Un cluster se comporta como si fuera un arreglo, con la diferencia que en él se guardan las direcciones indirectas de los datos y no los datos como en un arreglo común.

LabVIEW cuenta con estructuras de datos como las que se muestran en la figura V.8.

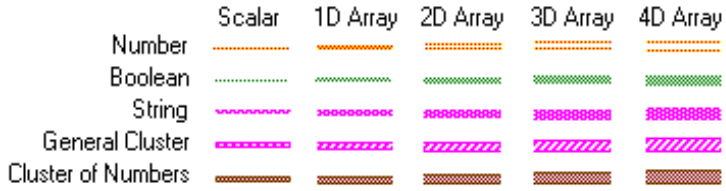


Figura V.8. Representación de estructuras de datos para diferentes tipos de datos.

V.2.9. Estructuras de Programación

Las estructuras de programación que soporta LabVIEW son las mismas que poseen otros lenguajes de programación, con la diferencia que éstas se despliegan en modo gráfico en la ventana de programación (figura V.9).

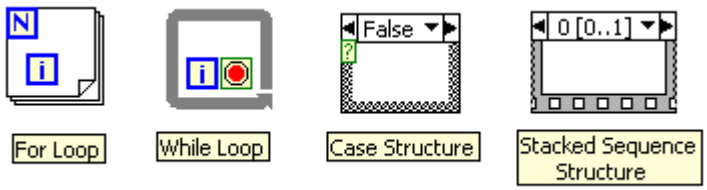


Figura V.9. Estructuras de programación en LabVIEW 8.0.

De esta forma, las estructuras que LabVIEW posee son:

- ***For Loop:*** Se utiliza cuando se desea que una operación se repita un número determinado de veces. Su equivalente en lenguaje convencional es:

```
For i = 0 to N-1
    Ejecutar instrucciones
```

- ***While Loop:*** Se utiliza cuando se desea que una operación se repita mientras una determinada condición sea cierta. Su equivalente en lenguaje convencional es:

Do ejecutar instrucciones
While condición **is TRUE**

- ***Case:*** Se utiliza en aquellas situaciones en las que el número de alternativas disponibles sean dos o más. La estructura *case* de LabVIEW permite que se realice una operación u otra diferente dependiendo del estado de una variable booleana o numérica. En caso de ser booleana, si la variable es verdadera, la estructura Case ejecuta el sub-diagrama *TRUE*, de lo contrario ejecuta el sub-diagrama *FALSE*.

- ***Sequence:*** Esta estructura no tiene su homóloga en los diferentes lenguajes convencionales de programación, ya que en éstos las sentencias se ejecutan en el orden de aparición en que están declaradas dentro de la estructura del programa. Sin embargo, como en LabVIEW esto no existe, las funciones se ejecutan de una forma aparentemente simultánea, de modo que muchas ocasiones el programador necesita crear un libreto en el cual decide qué funciones tienen que realizarse rigurosamente antes que otras, creándose un flujo de programa. Cada subdiagrama estará contenido en un "frame" o marco y estos se ejecutarán en orden de aparición.
    - *Sequence Local*. Se utiliza para pasar datos de un frame a otro.

- ***Formula Node:*** Es una estructura peculiar de LabVIEW que sirve para introducir fórmulas matemáticas de forma textual (figura V.10), que de otra forma podrían resultar muy complicadas de implementarse, utilizando las herramientas gráficas de las bibliotecas de funciones matemáticas. De esta forma el programador puede crear módulos gráficos personalizados que en su interior son de tipo texto. Además soporta funciones trigonométricas entre otras. No hay límite para el número de variables o de fórmulas y nunca podrá haber dos entradas o dos salidas con el mismo nombre, aunque una salida si podrá tener el mismo nombre que una entrada. Todas las variables de salida deberán estar asignadas a una fórmula por lo menos una vez.

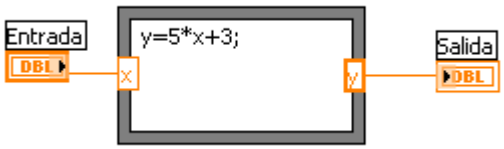


Figura V.10. Ejemplo del uso de una estructura Formula Node.

V.2.10. Variables locales y globales

Las variables son imprescindibles en cualquier tipo de problemas, ya que permiten almacenar la información necesaria para su resolución.

En LabVIEW todos los controles introducidos en el *Panel Frontal* que generan una terminal en la ventana *Diagrama* van a ser variables, identificadas por el nombre asignado a su etiqueta. Pero puede ocurrir que se desee utilizar el valor de cierta variable en otro subdiagrama o en otro VI o simplemente que se requiera guardar un resultado intermedio. La forma más sencilla de hacerlo es generando variables locales y/o globales, dependiendo de la aplicación.

En las variables locales los datos se almacenan en algunos de los controles o indicadores existentes en el *Panel Frontal* del VI creado; es por eso que estas variables no sirven para intercambiar datos entre VI's. La principal utilidad de estas variables radica en el hecho de que una vez creada la variable local no importa que proceda de un indicador o de un control se podrá utilizar en un *Diagrama* tanto de entrada como de salida.

Las variables globales son un tipo especial de VI, que únicamente dispone de *Panel Frontal*, en el cual se define el tipo de dato de la variable y el nombre de identificación imprescindible para después poder referirse a ella.

Attribute node: Los *atributos nodales* (figura V.11), se pueden considerar como variables que dependen únicamente de las terminales de control o de tipo indicador a partir de la cual se han creado y permiten que se pueda leer o modificar sus atributos en *Panel Frontal*. Con estas variables se puede cambiar un objeto de Control a Indicador y viceversa; también se pueden modificar atributos como el color, la posición la visibilidad y la activación de cualquier terminal. El programador puede identificarlos en cualquier momento por medio de una orden llamada *FIND ATTRIBUTE NODE* desde el *Panel Frontal* o *FIND TERMINAL* desde la ventana de *Diagrama.*

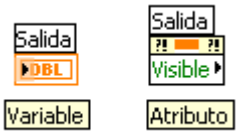


Figura V.11. Iconos representativos de una variable y uno de sus atributos (área de diagrama).

V.2.11. Utilización del puerto serie mediante LabVIEW

LabVIEW proporciona herramientas de gran utilidad para el manejo del puerto serie. Todas las funciones que son necesarias a la hora de realizar una comunicación serie entre el PC y un periférico se encuentran ya programadas en forma de Instrumentos Virtuales. De esta forma, la utilización del puerto serie es casi transparente al programador de LabVIEW.

La manera de actuar es la siguiente: cuando se desee realizar alguna operación con el puerto, se escogerá el icono necesario para dicha función, éste se cableará de forma adecuada y, al ser ejecutado, LabVIEW se ocupará de manejar el puerto convenientemente para obtener o entregar los datos requeridos. Los menús para el manejo del puerto serie se observan en la figura V.12.

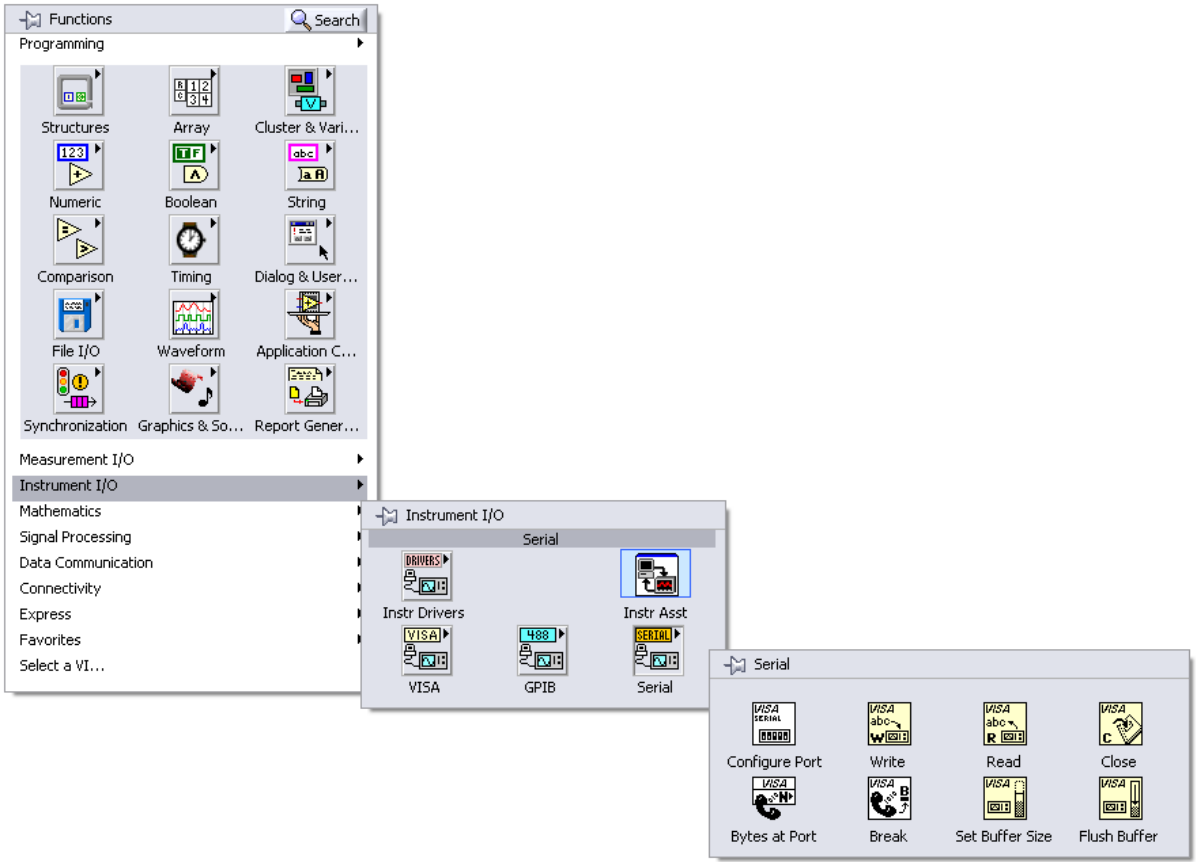


Figura V.12. Menús para el manejo del puerto serie.

Para utilizar el puerto serie hay que seguir los dos pasos siguientes:

- Realizar la configuración del puerto serie, inicializándolo según las características que se deseen para la comunicación. Ya no será necesario volver a configurar el puerto mientras no se varíenlas condiciones de la comunicación.

- Acceder al puerto serie para recibir o enviar datos tantas veces como se desee.

Antes de poder utilizar el puerto serie para transmitir y/o recibir, es imprescindible configurarlo. De esta manera se le indica a la PC cómo ha de actuar en las comunicaciones, es decir, qué puerto serie ha de utilizar, con qué velocidad de transmisión ha de emitir y recibir bytes, qué tipo de paridad ha de utilizar, etc.

Es importante tener en cuenta que la configuración que se le dé al puerto serie de la PC ha de ser exactamente la misma que utilice el dispositivo periférico. Si esto no fuera así, PC y periférico no podrían comunicarse con éxito, ya que estarían utilizando especificaciones de transmisión diferentes.

En el siguiente capítulo se explicará la programación de cada uno de los comandos que se necesitaron para poder diseñar la interfaz virtual entre el analizador de gases y la computadora.

CAPÍTULO VI

DISEÑO DEL SOFTWARE

El presente capítulo trata la forma en que se diseñó la interfaz virtual para el analizador de gases, así como cada uno de los instrumentos virtuales que lo conforman.

VI.1. DESCRIPCIÓN DE LOS INSTRUMENTOS VIRTUALES

VI.1.1. Estructura general de un comando

Como ya se mencionó, la banca se rige por comandos, por lo que en el presente capítulo se explicará cada uno de ellos.

La estructura general de un comando consta de varios campos, y estos campos cambian dependiendo de si se trata de una solicitud a la banca o de si es la respuesta de la banca a una solicitud hecha.

Para el caso de una solicitud los campos que la conforman de manera general son: *command* (comando), *command_mode* (modo_comando), *data_count* (contador_dato), *data* (dato) y *chksum* (checksum); y se envían en el siguiente orden:

Solicitud:

<*command, command_mode*><*data_count*>[…<*data*>…]<*chksum*>

Debe tenerse en cuenta que una palabra, en nuestro caso, son 2 bytes (16 bits). Cada palabra está determinada por los pico paréntesis <>. Por ejemplo, la primera palabra <*command, command mode*> la conforman dos campos, cada uno de 1 byte. En cambio el campo <*data count*> ya es de 2 bytes. Excepto el campo *data*, ya que es el único que puede abarcar más de 1 palabra, dependiendo del número de datos que se vayan enviar en dicho campo, que como se explicará más adelante, está dado por el *data_count*.

A continuación se describe cada uno de los campos:

El campo *command* es de 1 byte y contiene el número de comando del que se esté tratando. Este número ya está designado para cada uno de los comandos que se verán más adelante.

El campo *command_mode* también es de 1 byte y contiene un número que designa el modo en el que se va a trabajar. Sólo se pueden tener dos valores posibles:

0: modo normal
1: modo petición de datos para comandos largos

El campo *data_count* es de 2 bytes y contiene el número de palabras que van a venir en el campo data. Este campo también es designado nada más con la letra n (<n>).

El campo *data* es muy importante porque es el que va a contemplar todos los datos que se requieren enviar en la solicitud para obtener la respuesta deseada. El símbolo [...<*data*>...] quiere decir que los parámetros del campo data son opcionales dependiendo del comando que se trate, esto mismo hace que la longitud de este campo sea variable. El tamaño del campo *data* está dado por la multiplicación del *data_count* y 2 bytes (1 palabra), es decir:

data_count * 2 bytes = *data_count* palabras

El campo *data* tiene una forma integral dado que puede integrarse por varios elementos llamados d(1), d(2), ..., d(n). Cabe recordar que el número de elementos es determinado por el *data_count*:

data = <d(1)><d(2)>…<d(n)>

El campo de *chksum o checksum,* es utilizado más que nada para saber si la información llegó completa. El *checksum* se refiere a la sumatoria de 16 bits de todos los bytes de datos que se han transmitido.

Si se llaman c0, c1, ..., c7 a los bits del campo *command*. Y mc0, mc1, ..., mc7 a los bits del campo *command_mode*. Así también n0, n1, ..., n15 son los bits del campo *data_count*. Y para el campo *data*, que a su vez se subdivide en d(1),d(2), ..., d(n); entonces se tiene que los bits para d(1) son d(1)0, d(1)1, ..., d(1)15. Similarmente para d(2) son d(2)0, d(2)1, ..., d(2)15. Igual para d(n): d(n)0, d(n)1, ..., d(n)15. Entonces el *checksum* queda como la suma de todos los bits anteriores en el orden mostrado a continuación:

		mc0	mc1	mc2	...	mc6	mc7	c0	c1	c2	...	c6	c7
		n0	n1	n2	...	n6	n7	n8	n9	n10	...	n14	n15
		d(1)0	d(1)1	d(1)2	...	d(1)6	d(1)7	d(1)8	d(1)9	d(1)10	…	d(1)14	d(1)15
chksum =	+	d(2)0	d(2)1	d(2)2	...	d(2)6	d(2)7	d(2)8	d(2)9	d(2)10	…	d(2)14	d(2)15
		.	.	.	.	.	.	.	.	.		.	.
		.	.	.	.	.	.	.	.	.		.	.
		.	.	.	.	.	.	.	.	.		.	.
		d(n)0	d(n)1	d(n)2	...	d(n)6	d(n)7	d(n)8	d(n)9	d(n)10	…	d(n)14	d(n)15

Cuando se envía la solicitud con el *checksum* incluido, el receptor recibe la cadena de bits y, a su vez, debe realizar también la suma de los datos, es decir, hacer su propio *checksum*; después de esto se compara si son iguales los dos *checksum*, si sí lo son entonces la información se puede decir que llegó completa; si no son iguales, entonces la información llegó incompleta y requiere enviarse de nuevo la solicitud. Se debe aclarar que el *chksum* no detecta errores que indiquen si la información que se envía a la banca es la correcta, sólo indica si llegó completa la información.

Para el caso de la respuesta, básicamente se utilizan los mismos campos empleados en una solicitud, con excepción del campo *command_mode,* que es cambiado por el de *response_mode*, enviando una cadena como la que se indica a continuación.

Respuesta:

<*command, response mode*><*data_count*>[...<*data*>...]<*chksum*>

El campo *command* sólo es la repetición del comando requerido que se mandó en la solicitud hecha.

El campo *response_mode* es el que indica si el comando fue recibido exitosamente o si hubo algún error en la transmisión. También este campo es de 1 byte y puede tener 3 posibles valores:

- *ACK* (0x06): comando recibido OK y emitido completamente.
- *BUSY* (0x05): comando recibido OK pero el proceso está aún pendiente.
- *NACK* (0x15): error en la transmisión o comando de error.

El 0x al principio del número recibido significa que este número es hexadecimal, es decir, que en decimal para un *ACK* se recibe un 6, para un *BUSY* se recibe un 5 y para un *NACK* un 21.

VI.1.2. Comandos largos

Se dice que un comando es largo cuando la banca no puede terminar el comando solicitado dentro del tiempo entre mensajes, si esto sucede el *response_mode* es puesto a *busy* y no son transmitidos datos. La respuesta general sería como sigue:

Solicitud:

<*command*, 0><n>[<…*data*…>]<*chksum*> ‘0’ = modo normal

Respuesta:

<*command, BUSY*><0><*chksum*>

Para determinar cuando el comando está completado, se puede usar la función [*process_status*], el cual es por definición un comando inmediato. Si el comando está completado y el dato está disponible, el comando deberá ser llamado en un modo largo especial descrito como sigue:

Petición:

<*command*, 1><0><*chksum*> ‘1’ = Modo petición dato

Respuesta:

<*command, response_mode*><n>[<…*Data*…>]<*chksum*>

Data está disponible hasta que el comando pendiente sea emitido en el modo largo u otro comando sea emitido, con la excepción del comando *process_status*. Cualquier comando puede ser un comando largo, sin haberse indicado explícitamente en este capítulo. Mientras que haya un comando pendiente, sólo el comando *process_status* podrá ser procesado, los otros comandos recibirán un *NACK*.

VI.1.3. Procesamiento de datos

A continuación, se mencionan algunos aspectos importantes que son necesarios saber para el procesamiento de datos, que maneja la banca AMB II 9270-054:

- **Bits indefinidos:** Todos los bits indefinidos o no usados, en un comando solicitado, son puestos a cero internamente. Los bits indefinidos en la respuesta serán puestos a 0.

- **Ordenamiento de bytes:** Los campos de *data_count*, *data* y *checksum* están organizados en palabras y serán transmitidos en dos bytes: el byte menos significativo (LSB: *low significant byte*) y el byte más significativo (MSB: *most significant byte*). El orden de transmisión y recepción para una palabra es primero el LSB seguido por el MSB.

- **Condiciones NACK** (Negative ACKnowledgement): Las siguientes condiciones están dentro de una respuesta NACK por la banca:

1. El tiempo entre cadena es mayor que 50 [ms] mientras llega el dato.
2. No se transmiten datos suficientes.
3. Error de *checksum*.
4. Comando desconocido.
5. Un parámetro requerido está fuera del rango definido.

VI.1.4. Descripción de los comandos

El *software* que se diseño para este trabajo de tesis, y que sirve para obtener la comunicación entre el sistema analizador de gases y la computadora, se realizó con la integración de varios instrumentos virtuales, los cuales corresponden cada uno a un comando válido para la banca. Los comandos utilizados, se muestran en la tabla VI.1. Todas las tablas expuestas en este capítulo fueron tomadas del Manual Software Interface Specification de Sensors propio de la banca AMBII 9270-054.

No. de Comando	Descripción
2	Calibración (*Calibration*)
3	Obtener dato (*Get data*)
4	Obtener estado (*Get status*)
5	Lectura/Escritura de dato de calibración (R/W calibration data)
6	Programando flash (Flash programming)
7	Switches característicos de usuario (User features switches)
8	Lectura/ Escritura de entradas/salidas (*Read/Write I/O's*)
9	Calibrar RPM analógico (Calibrate analog RPM)
11	Estado del proceso (*Process status*)
12	Obtener versión (*Get version*)
13	Número serial (*Serial number*)
14	Usuario EEPROM (User EEPROM)
16	Calibrar transductor de presión (Calibrate pressure transducter)
18	Activar nivel de acceso (Set access level)

Tabla VI.1. Comandos para la banca AMB II 9270-054.

De los comandos mencionados en la tabla sólo se programaron los que se consideraron más importantes, de acuerdo a las necesidades del LCE, éstos son:

1. Calibración
2. Obtener Dato
3. Obtener Estado
4. Lectura/Escritura de Entradas/Salidas
5. Estado del Proceso
6. Obtener Versión
7. Número Serial

VI.1.4.1. Calibración (*calibration*)

El comando *calibration* es usado para calibrar cada canal de la banca. En el apartado VI.1.1 se vio la estructura general para hacer una solicitud, pero en este caso ya se trata de un comando en especial, así que sólo se hace la sustitución de ciertos valores, como por ejemplo el número correspondiente al *command* es el 2; n para este caso también es 2, dado que se van a enviar dos palabras en el campo *data* que son *mode* y *gas*. Para que quede más claro, la solicitud completa es la siguiente:

Solicitud:

<2, *command_mode*><2><*mode*><*gas*><*chksum*>

El campo *command_mode* es el mismo para todos los comandos, tiene el valor de cero, el cual indica que el modo en el que se va a trabajar es el normal.

El campo *mode* puede tomar 4 valores, dependiendo de estos valores se seleccionan los gases que se pueden calibrar y también el tipo de calibración que se desea realizar. Todo esto se ilustra en la tabla VI.2.

Mode	HC	CO	CO_2	O_2	NO_X	HiHC	Descripción
0	X	X	X	X	X		Zero
1	X	X	X	X	X	X	Calibración de Usuario de Punto Simple
2	X	X	X		X		Calibración # 1 de Dos Puntos
3	X	X	X		X		Calibración # 2 de Dos Puntos

Tabla VI.2. Valores del campo mode para el comando Calibration.

El campo *gas* es para seleccionar el gas que se va a calibrar. La palabra a enviar se divide de la manera mostrada en la tabla VI.3.

No. de bit

7	6	5	4	3	2	1	0
-	-	HiHC	NO_X	O_2	CO_2	CO	HC

Nombre del bit

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

Tabla VI.3. Bits del campo gas para el comando Calibration.

Por ejemplo, si el gas que se quiere calibrar es CO y HiHC, entonces la cadena de bits que se mandaría sería 00000000 00100010. Específicamente para este comando, la combinación HC y HiHC no es válida.

La respuesta a este comando sería:

Respuesta:

<2, *response_mode*><1><*mini_status*><*chksum*>

El campo *mini_status* regresa el código de estado de la calibración llevada a cabo. El campo de bits está formado como se indica en la tabla VI.4, en donde, *2Pt. Cal* corresponde a una calibración de dos puntos, y *1Pt.Cal* es una calibración de un punto o punto simple.

7	6	5	4	3	2	1	0
-	-	-	2Pt. Cal	-	-	1Pt.Cal	Zero

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

Tabla VI.4. Bits del campo mini_status para el comando Calibration.

VI.1.4.1.1. Instrumento Virtual Calibración

La programación con LabVIEW es muy sencilla, como se podrá ver a continuación. La ventaja del entorno de programación LabVIEW es que se pueden crear objetos o

instrumentos virtuales, los cuales pueden ser utilizados en otros programas más extensos o más completos.

Para la programación del instrumento virtual *Calibration* primero se generaron 2 instrumentos virtuales: *concatena* y *byte*; los cuales serán utilizados en el programa principal de cada comando o instrumento virtual.

El instrumento virtual concatena tiene el símbolo 

. Este instrumento se creó con el propósito de formar la palabra completa de 16 bits, es decir, juntar el byte más significativo MSB (*Most Significant Byte*) y el byte menos significativo LSB (*Least Significant Byte*) para obtener así el Número unificado.

El procedimiento de la programación del instrumento virtual concatena se ve en la figura VI.1, en donde, los números que aparecen en la imagen, corresponden a los números que se encuentran entre paréntesis () a lo largo de esta descripción. Primero se obtienen los dos bytes (1), el MSB y el LSB, cada uno de estos bytes se hace pasar por un convertidor de número a cadena de dígitos hexadecimales (2), con longitud de 2 dígitos (3), a su vez estas dos cadenas se hacen pasar por un concatenador de cadenas (4) para obtener una sola, la cual se vuelve a convertir de cadena de dígitos hexadecimales en número (5) para finalmente obtener el Número unificado (6), es decir, la palabra completa de 16 bits.

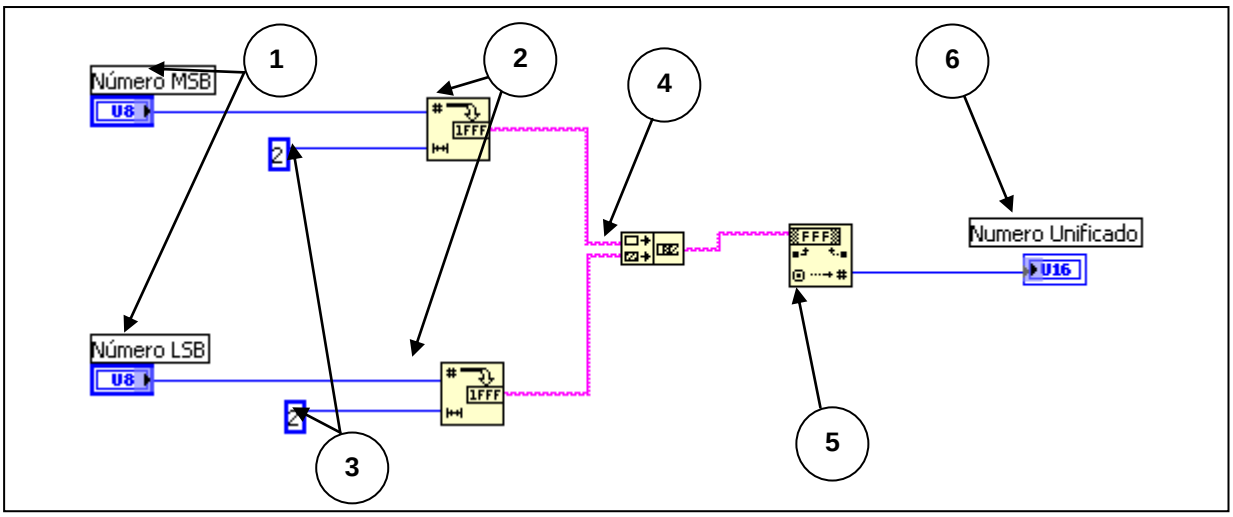


Figura VI.1. Instrumento virtual Concatena.

El instrumento virtual *byte*, representado por el símbolo 

, se encarga de separar un byte en sus ocho bits que lo conforman, esto es muy utilizado en los casos en que se

mencionan los campos de bits, por ejemplo, en el campo *gas* o *ministatus* de este mismo comando.

La programación del instrumento virtual *byte* se muestra en la figura VI.2. Primero se obtiene el byte que se va a dividir (1), se convierte de número a un arreglo booleano (2), del cual se van a ir obteniendo cada uno de sus elementos, del 0 al 7, con la ayuda del objeto *Index Array* (3) ya predeterminado de LabVIEW, el cual extrae el elemento que se le indica con el índice (4), consiguiendo así cada uno de los bits llamados Boolean 0, Boolean 1, ..., Boolean 7.

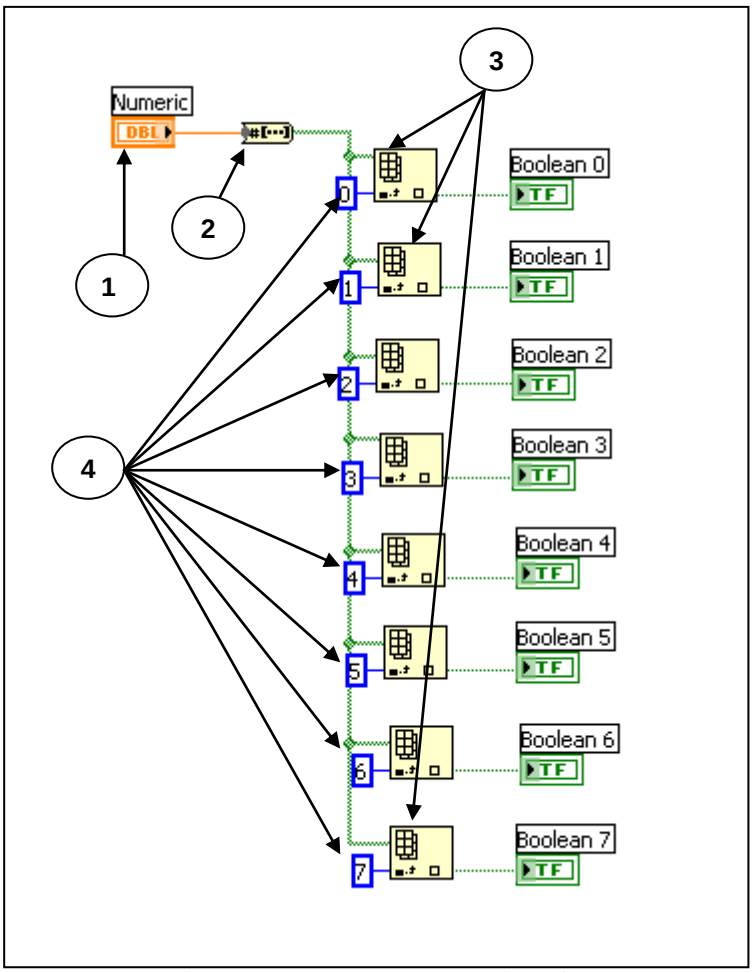


Figura VI.2. Instrumento virtual Byte.

Con los dos instrumentos virtuales explicados anteriormente, ahora se puede describir el instrumento virtual *Calibration*.

La programación del instrumento virtual *Calibration,* como la de todos los demás instrumentos virtuales que se expliquen más adelante, está dividida en dos partes, la primera consiste en recolectar los datos que se van a enviar para realizar la solicitud del comando y la

segunda parte para la decodificación de los datos obtenidos de la banca, es decir, la respuesta dada por la banca. Para la programación de las dos partes, se emplea una estructura de secuencia, la cual se puede ver en la figura VI.3 con el numero (1), en donde la solicitud se realiza en la secuencia 0 o frame 0 y la decodificación en el frame 1.

Para poder realizar la programación de la solicitud de este comando (figura VI.3), hay que reunir los datos que se van a mandar y juntarlos en un arreglo (2). Estos datos son los que ya se mencionaron anteriormente, es decir:

<2, *command_mode*><2><*mode*><*gas*><*chksum*>

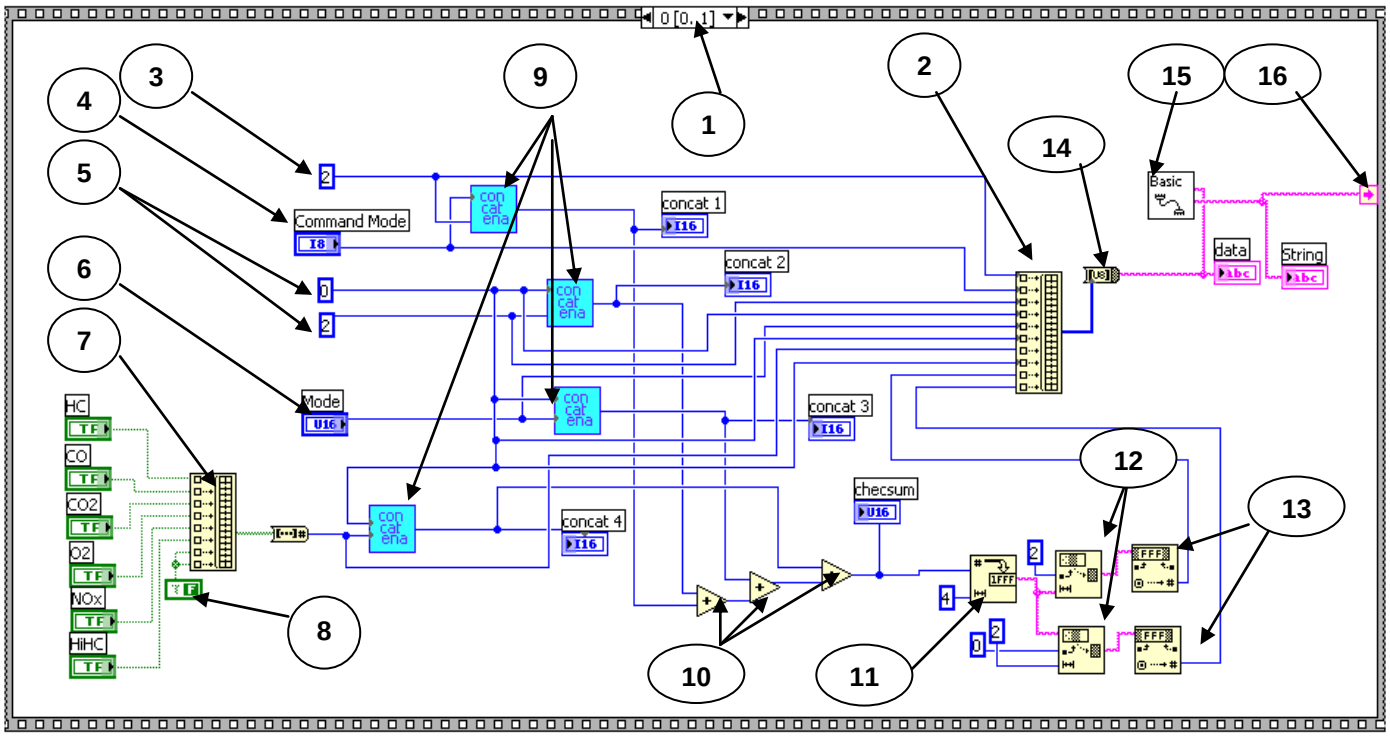


Figura VI.3. Instrumento Virtual Calibration (solicitud).

Lo primero que se debe de enviar, es un 2 (3) seguido del *command_mode* (4), el cual es un control en el que el usuario lo selecciona, cada uno de los elementos del control ya tiene designado el número de elemento que le corresponde, así el Normal es el elemento 0 y el Petición de datos el 1 (figura VI.4).

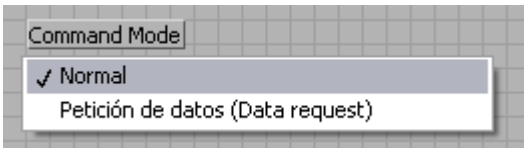


Figura VI.4. Control del campo Command_mode.

Posteriormente, se manda otro 2 correspondiente al *data_count* (5). Como el 2 decimal se puede representar en el byte menos significativo, entonces el byte más significativo vale 0.

Después sigue el campo *mode*, como se vio en la tabla VI.2, el campo *mode* puede tomar 4 valores, de los cuales, los correspondientes a la calibración de dos puntos no son tomados en cuenta por requerimientos del LCE. Por lo tanto para este caso *mode,* sólo se tomarán los dos primeros valores por medio de otro control (6) como el *command_mode* (figura VI.5).

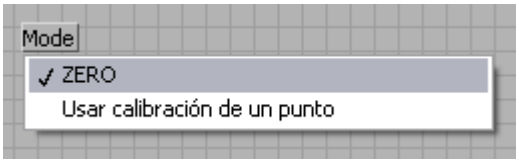


Figura VI.5. Control del campo Mode.

A continuación sigue el campo *gas*, como éste está dado por su campo de bits, entonces lo que se hizo fue un arreglo (7), el cual reúne cada uno de los bits. Estos bits también son dados por el usuario por medio de controles booleanos (figura VI.6), los cuales son activados o desactivados según se quiera para dar valor de 1 ó 0 al bit correspondiente. Para rellenar el byte, a los bits que valen 0 por default se le asignó una constante booleana que tiene valor falso (8).

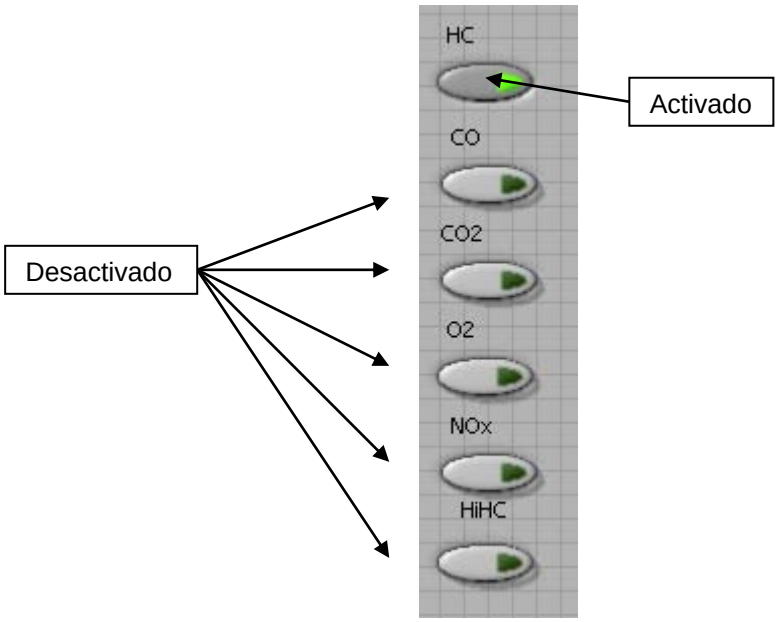


Figura VI.6. Control booleano del campo Gas.

Por último, se manda el *checksum*, que es la suma de todas las palabras enviadas. Para esto se hizo uso del instrumento virtual concatena (9), uniendo los 2 bytes de cada palabra, y realizando la suma con ayuda de sumadores (10).

Dado que son bytes los que están entrando al arreglo principal (2) y no la palabra completa, el *checksum* tiene que ser dividido en su MSB y LSB para poder entrar en el arreglo. Para realizar esto, una vez que se tiene la suma total se hace pasar por un convertidor de número a cadena de dígitos hexadecimales (11) con longitud de 4 dígitos; ya que se tiene esta cadena se divide en dos subcadenas, la primera se referirá al LSB y la segunda al MSB. El LSB se obtiene quitando los 2 dígitos correspondientes al MSB y dejando sólo los dos últimos dígitos, esto se hace utilizando el objeto *String Subset* (12) predeterminado de LabVIEW, poniéndole un *offset* de 2. Para obtener el MSB lo único que se hace es tomar los dos primeros dígitos de la cadena, estos se ejecuta similarmente con el mismo objeto *String Subset* (12) sólo que se le pone un *offset* de 0 y una longitud de 2. Después de la separación de los dos bytes, ambos pasan por otro convertidor (13), ahora de cadena de dígitos hexadecimales a número para llegar así al arreglo.

La explicación que se acaba de dar para la separación del LSB y MSB del *checksum*, ya no se explicará de nuevo en los siguientes comandos, puesto que el proceso de separación es exactamente igual.

El orden de los bytes al entrar al arreglo principal es como se menciona a continuación: Primero llega al arreglo el *command*, posteriormente va el *command_mode*. Seguidamente van los campos de *data_count*, *data* y *checksum* mandando primero el LSB y después el MSB. Este ordenamiento aplica para todos los instrumentos virtuales programados que se verán más adelante. El arreglo de bytes entra en un convertidor que los transforma en cadena (14).

Finalmente, la cadena que se obtiene del convertidor, ya contiene todos los datos que se van a mandar para solicitar el comando, y como esta cadena se tiene que enviar vía puerto serial, utilizamos el *Basic Serial Read and Write* predeterminado de LabVIEW 8.0 (15), en el cual, después de haber configurado los parámetros de bits de paro, velocidad de transferencia de datos, bits de datos y paridad, sólo se hace entrar la cadena que se va a transmitir y directamente nos devuelve la cadena leída del puerto, la cual se conecta a un *Sequence Local* (16) para su futura utilización en otras secuencias.

Para la parte de la decodificación, se hace referencia a la figura VI.7, la cual corresponde a la secuencia 1 (1) de la estructura de secuencia.

La cadena a decodificar concierne a la respuesta:

<2, *response_mode*><1><*mini_status*><*chksum*>

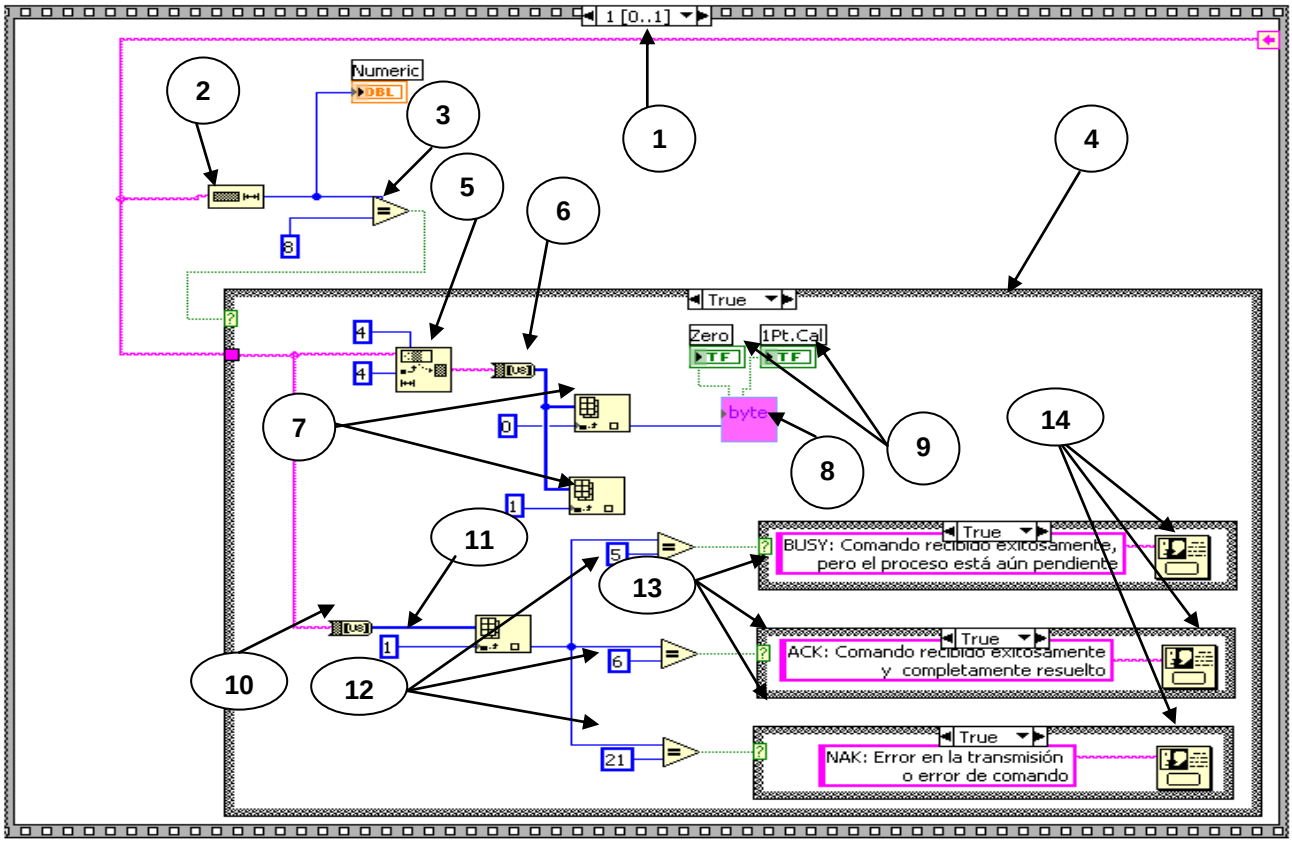


Figura VI.7. Instrumento Virtual Calibration (respuesta).

Primero, la cadena recibida tomada del *Sequence Local* se introduce en un *String Length* (2), éste obtiene la longitud de la cadena la cual se compara (3) con 8, ¿porqué 8?, porque es el número de bytes que esperamos recibir: 1 byte del 2 + 1 byte del *response_mode* + 2 bytes de n + 2 bytes del *mini_status* + 2 bytes del *chksum*. Este comparador da una salida binaria, regresa un *true* si es igual a 8 y regresa un *false* en cualquier otro caso; ésta salida entra en una estructura *case* (4). Si la salida del comparador es *true* entonces se procede a hacer la decodificación, si es *false* no se procede a hacer alguna acción. En el caso *true*, la cadena leída del puerto serial se introduce a un *String Subset* (5) para quitarle el *command*, el *response_mode*, el *data_count* y el *checksum* y obtener sólo los datos: *mini_status*. Para esto se le quitan los 4 primeros bytes quedando 4 bytes más de los cuales no se toman los dos últimos porque corresponden al *checksum*.

Posteriormente la cadena restante se transforma a un arreglo de bytes (6) para que con el *Index Array* (7) se seleccionen los elementos 0 y 1 correspondientes a los dos bytes del *mini_status*, que es el que nos interesa.

El elemento 0 es el LSB, y con ayuda del instrumento virtual programado *byte* (8) que se explicó anteriormente, se toman sólo los bits que son de utilidad, en este caso el bit 0 y el bit 1, concernientes al *zero* y 1Pt. Cal (calibración de 1 punto) respectivamente. Para la visualización de estos datos en el panel frontal del instrumento virtual *Calibration*, las salidas proporcionadas por el objeto byte son conectadas a dos *displays* o indicadores (9) ubicados en el panel frontal (figura VI.8).

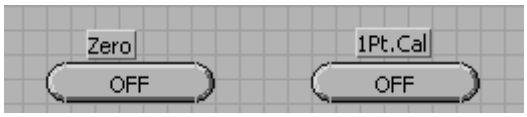


Figura VI.8. Indicadores en el panel frontal.

Como el contenido del elemento 1 son puros ceros, ya no se toma en cuenta y no se toma ningún bit de este byte.

Para verificar que el comando haya llegado correctamente a la banca, hay que checar el *response_mode*, que puede tomar los valores mencionados en el apartado VI.1.1 Estructura General de un Comando. Para hacer esto la cadena leída del puerto serial se convierte (10) directamente a un arreglo de bytes, del cual se toma el elemento 1 (11) que se relaciona con el *response_mode*. Como la única acción que se puede tomar es avisar al usuario el estado en que llegó el comando, entonces el *response_mode* se compara con los tres valores posibles que puede tomar (12), y dependiendo de la salida de cada comparador, éstas van a dar a estructuras case (13), que, en cada caso, si el valor de la salida del comparador es *true*, entonces se indica que se arroje un letrero (14) que indique al usuario la situación en que llegó el comando solicitado. En caso de que la salida sea *false* no se procede a realizar alguna acción. Si se da el suceso de que el *response_mode* recibido sea igual a un 21 decimal y, por lo tanto, haya habido un error en la transmisión, entonces deberá enviarse nuevamente la solicitud del comando hasta recibir una respuesta favorable, es decir, que el *response_mode* tenga el valor de 6.

Los símbolos *concat1*, ..., *concat4*, *checsum*, *data* y *string* de la figura VI.3, y *Numeric* de la figura VI.7 son *displays* colocados en el panel frontal los cuales son para uso del programador, para saber que lo que se está mandando y recibiendo es correcto.

Finalmente para este comando, el panel frontal con el que va a interactuar el usuario luce de la manera mostrada en la figura VI.9.

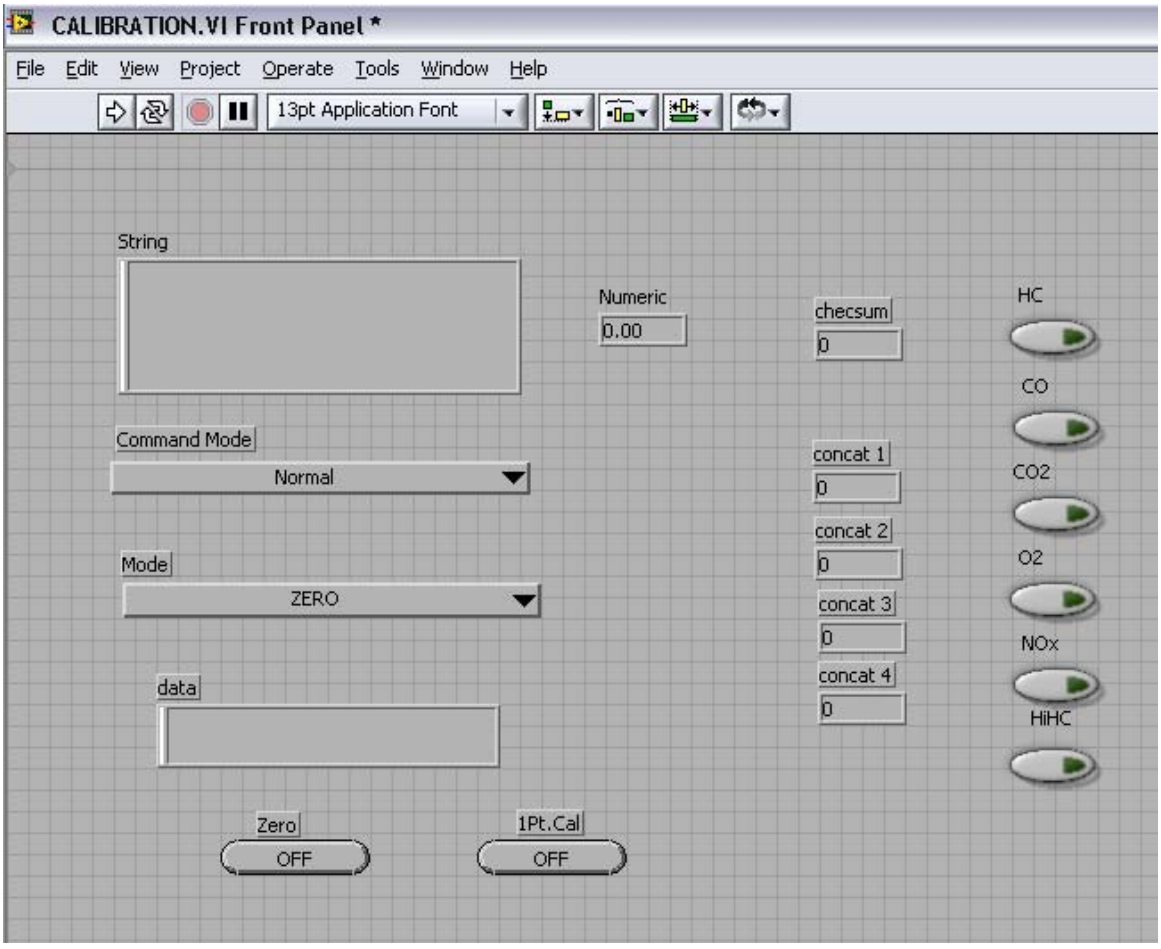


Figura VI.9. Panel Frontal del Instrumento Virtual Calibration.

Una vez terminado el instrumento virtual, se editó su icono 

correspondiente, para ser, posteriormente, llamado en el programa principal.

VI.1.4.2. Obtener dato (*get data*)

El comando *get data* reporta el dato del *gas* y canales auxiliares. La solicitud para este comando se conforma como sigue:

Solicitud:

<3, *command_mode*><2><*mode*><*units*><*chksum*>

Para este comando, el campo *command* toma el valor 3, que es el número de comando correspondiente al comando *get data*, y *data_count* o *n* es igual a 2, porque el

campo *data* abarca 2 subcampos: *mode* y *units*. El campo *mode* sirve para seleccionar la forma o el tipo en que se va reportar el dato del gas. Los 4 posibles valores que puede tomar se ilustran en la tabla VI.5.

Mode	Descripción
0	Dato de gas reportado en concentración
10	Dato de gas reportado en voltaje
11	Dato de gas reportado en cálculos A/D
12	Dato de gas reportado en Modulaciones

Tabla VI.5. Valores del campo mode para el comando get data.

El campo *units,* es utilizado para seleccionar las unidades para varios datos, por ejemplo para los gases, la temperatura, la presión, la humedad relativa, etc. La palabra completa dividida en bits se muestra en la tabla VI.6.

7	6	5	4	3	2	1	0
HC	pRes	-	-	O2res	Range	Temp	Press

15	14	13	12	11	10	9	8
tempRes	-	-	IrRes	exTemp	RH	RPM	OilTemp

Tabla VI.6. Bits del campo units para el comando get data.

Las unidades posibles así como sus valores correspondientes para cada dato se indican en la tabla VI.7.

Para la respuesta el formato que se sigue es:

Respuesta:

<3, *response_mode*><20><*status*><...[*data*]...><*chksum*>

En el caso de la respuesta de este comando, n corresponde a 20, es decir, que en los siguientes dos campos: *status* y *data*, se envían 20 palabras en total, 1 palabra es del *status* y 19 son del *data,* se puede observar que se maneja el símbolo <...[*data*]...>, esto es para no enumerar estas 19 palabras, aunque también podría expresarse como <*…data1…*><*…data2…*>…<*…data19…*>.

Units	***Descripción***
Press	0 : Presión en [mbar] 1 : Presión en [inHg]
Temp	0 : Temperatura en [°C] 1 : No válido (unidad de temperatura es siempre en °C)
Range	No válido
O2res	0 : Alta resolución en O_2 (xx.xx) 1 : Baja resolución en O_2 (xx.x)
pRes	Resolución de Presión / Press = 0 / Press = 1 0 / xxxx / xx.xx 1 / xxxx.x / xx.xxx
HC	0 : HC en Hexano 1 : HC en Propano
OilTemp	0 : [°C] 1 : [mV]
RPM	0 : RPM en [1/min] 1 : RPM en [mV]
RH	0 : Humedad relativa en % 1 : Humedad relativa en [mV]
exTemp	0 : [°C] 1 : [mV]
IrRes	0 : Resolución estándar para HC, CO, CO2 HC : 1 [ppm] CO : 0.01% CO_2 : 0.1% 1 : Resolución alta para HC, CO, CO2 HC : 0.1 [ppm] CO : 0.001% CO_2 : 0.01%
tempRes	0 : Resolución de temperatura – xx.x 1 : Resolución de temperatura – xx.xx

Tabla VI.7. Valores del campo units para el comando get data.

El campo *status* da resultado de varios datos, los cuales sirven para determinar el estado en el que se encuentra la banca. Cada bit indica uno de estos datos, como se muestra en la tabla VI.8.

7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Warmup	Cond	hiHC	Zero	notAcc	-	-	Lowflow

15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
IntWarn	-	-	NOXrange	O2range	CO2range	COrange	HCrange

Tabla VI.8. Bits del campo status para el comando get data.

En la tabla VI.9 se indica el significado de cada bit, así como las condiciones que se activan para dichos bits.

Bit	Significado	Condición Puesta	
LowFlow	Bandera baja detectada	0 : J10-Pin1 está en 5[V] (TTL) 1 : J10-Pin1 está en 0[V] (TTL)	
notAcc	Dato no puede ser correcto	notAcc=1 si cualquiera de lo siguiente es verdad: Get Status SW0: b0 o b8, Get Status SW6: b1,b13 o b14, Get Status SW7: b0, b1, b2, b3, b4, b7, b8, b10, b11 o b13	
Zero	Zero recomendado	La banca requiere un zero	
hiHC	Rango de HiHC	IRes=0 (resolución estándar IR) 0: HC<2000 ppm Hexano (<4000 Propano) 1: HC>2000 ppm Hexano (>4000 Propano) IRes=1 (alta resolución IR) 0 : HC < 200 ppm Hexano (< 400 Propano) 1 : HC > 200 ppm Hexano (> 400 Propano)	
Cond	Advertencia de condensación	Ver Bandera de Condensación	
Warmup	Calentamiento en progreso	La banca está en calentamiento	
HCrange	HC fuera de rango	<-26 ppm (Propano)	IRes=0 (resolución estándar IR): >40000 ppm Propano IRes=1 (alta resolución IR): >4000 ppm Propano
		< -13 ppm (Hexano)	IRes=0 (resolución estándar IR): >20000 ppm Hexano IRes=1 (alta resolución IR): >2000 ppm Hexano
COrange	CO fuera de rango	CO < -0.06%	CO > 15.0%
CO2range	CO_2 fuera de rango	CO_2 < -0.3%	CO_2 > 20.0%
O2range	O_2 fuera de rango	O_2 < -0.10%	O_2 > 21.7%
NOXrange	NO_X fuera de rango	NO_X < -32 ppm	NO_X > 5000 ppm
IntWarn	Advertencia interna	Cuando la bandera de advertencia interna está activa, el "happy LED" está parpadeando a 5 [Hz]. Frecuencia de parpadeo normal es 1 [Hz]	

Tabla VI.9. Valores del campo status para el comando get data.

Para el campo *data* se dijo que eran 19 palabras, es decir 19 datos, la tabla VI.10 señala a que se refiere cada *data*.

Data	Modo
1	Canal HC
2	Canal CO
3	Canal CO_2
4	Canal O_2
5	Canal NO_x
6	Entrada Analógica de RPM
7	Entrada Analógica de Temperatura Aceite
8	Temperatura Ambiente
9	Presión
10	Entrada Analógica de Humedad Relativa
11	Entrada Analógica de Temperatura Externa
12	Canal de Referencia
13	Temperatura Lámpara
14	Temperatura Detector
15	Spare (Pieza de Repuesto)
16-19	Reservado

Tabla VI.10. Valores del campo data para el comando get data.

VI.1.4.2.1. Instrumento Virtual Obtener Dato

Para la programación del instrumento virtual *get data*, se requiere de un instrumento virtual llamado *units*.

El instrumento virtual *units* tiene el símbolo 

y sirve para seleccionar las unidades y resolución de algunos datos, como son los gases, la temperatura, presión, humedad relativa, etc. Para esta explicación se toma como referencia la figura VI.10.

La programación de este instrumento virtual es muy sencilla y con ésta se obtienen los dos bytes de la palabra *units*. Para cada byte se utiliza un arreglo (1) para unir cada uno de los elementos o bits (2) que conforman la palabra, los cuales ya se indicaron en la tabla VI.5. Como se manejan variables booleanas, a los elementos que valen cero se les asocia una constante booleana con valor *false* (3). Por último la salida del arreglo se convierte (4) en número para su futura utilización.

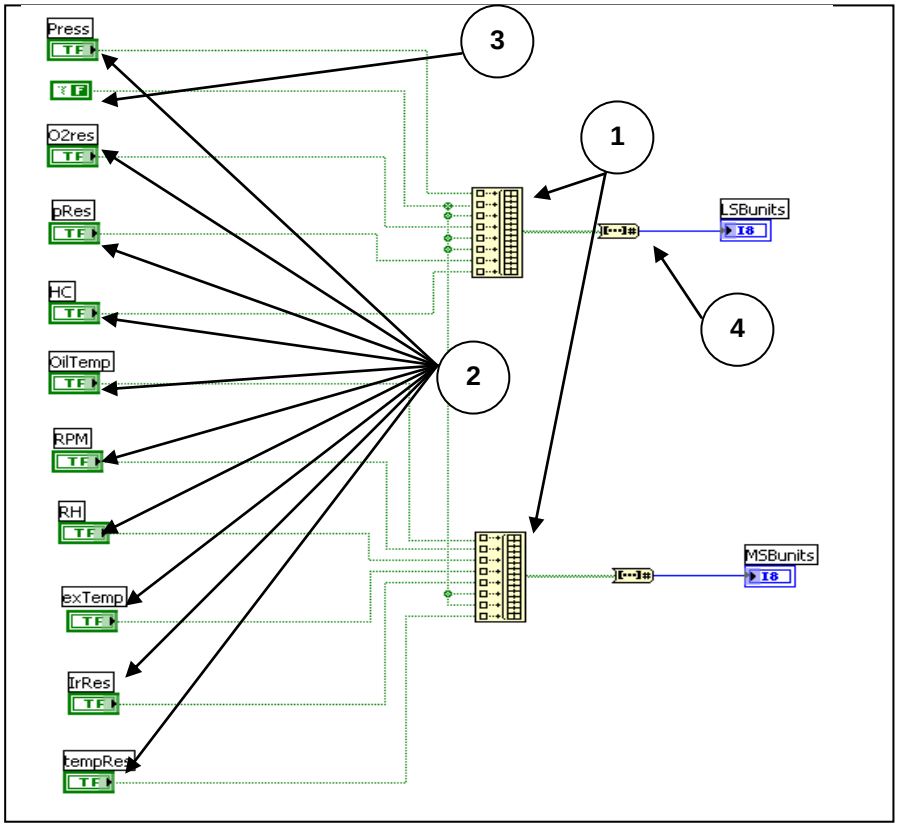


Figura VI.10. Instrumento virtual Units.

Una vez programado el instrumento virtual *units*, se programa el instrumento virtual *get data*. Para la siguiente explicación se hace referencia a la figura VI.11.

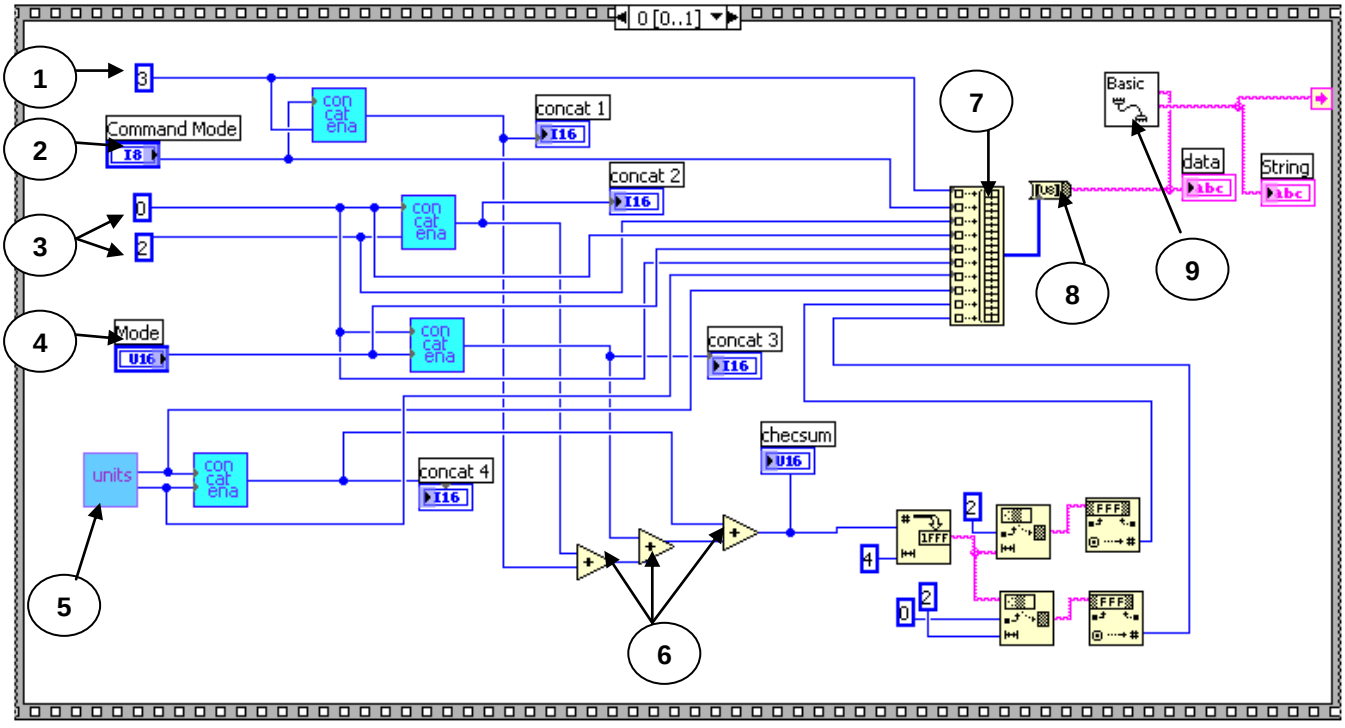


Figura VI.11. Instrumento virtual get data.

La programación de la solicitud para el instrumento virtual get data es muy parecida a la del comando anterior. Para la solicitud se hace uso de la secuencia cero de una estructura de secuencia, en la cual se requiere enviar la siguiente cadena:

<3, *command_mode*><2><*mode*><*units*><*chksum*>

Para eso, como en el comando anterior, primero se manda el 3 (1), contiguamente el *command_mode* (2), así también los dos bytes del *data_count*: 0 y 2 (3). Después sigue un control para seleccionar el *mode* (4) que se va a usar, éste aparece en el panel frontal con las opciones que se muestran en la figura VI.12.

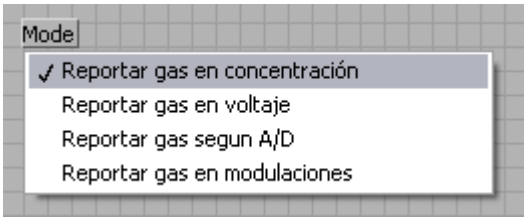


Figura VI.12. Control para seleccionar el mode.

Posteriormente se utiliza el instrumento virtual *units* (5) explicado anteriormente, y por último se realiza la sumatoria para obtener el *checksum*. El procedimiento para la separación y obtención de los dos bytes (LSB y MSB) del *checksum* ya se explicó en el primer comando.

Todos estos elementos se juntan en un arreglo principal (7), del cual la salida va a convertidor (8) el cual lo transforma en cadena, para después ser enviado por vía puerto serial (9).

Para la segunda parte o secuencia 1 se refiere a la figura VI.13. En ésta hay que decodificar la cadena:

<3, *response_mode*><20><*status*><...[*data*]...><*chksum*>

Para esta parte sólo se toma la cadena que fue leída del puerto serial (1), una vez que la banca haya contestado, y se obtiene su longitud por medio de un *String Length* (2). Una vez obtenida ésta, se compara con 46 (3), que son el número de bytes que se esperan recibir si la respuesta llega completa. El lector puede preguntarse ¿de dónde salen esos 46 bytes?, pues se tiene la siguiente suma: 1 byte del 3 + 1 byte del *response_mode* + 2 bytes del

data_count + 2 bytes del *status* + 38 bytes de data (son 19 *data* * 2 bytes = 38 bytes) + 2 bytes del *checksum*.

La salida del comparador va conectada a una estructura *case* (4). Si la salida es *false* no se realiza acción alguna. Si es *true*, la cadena entra a otro instrumento virtual que se programó llamado *deco* (5), el cual se describe a continuación.

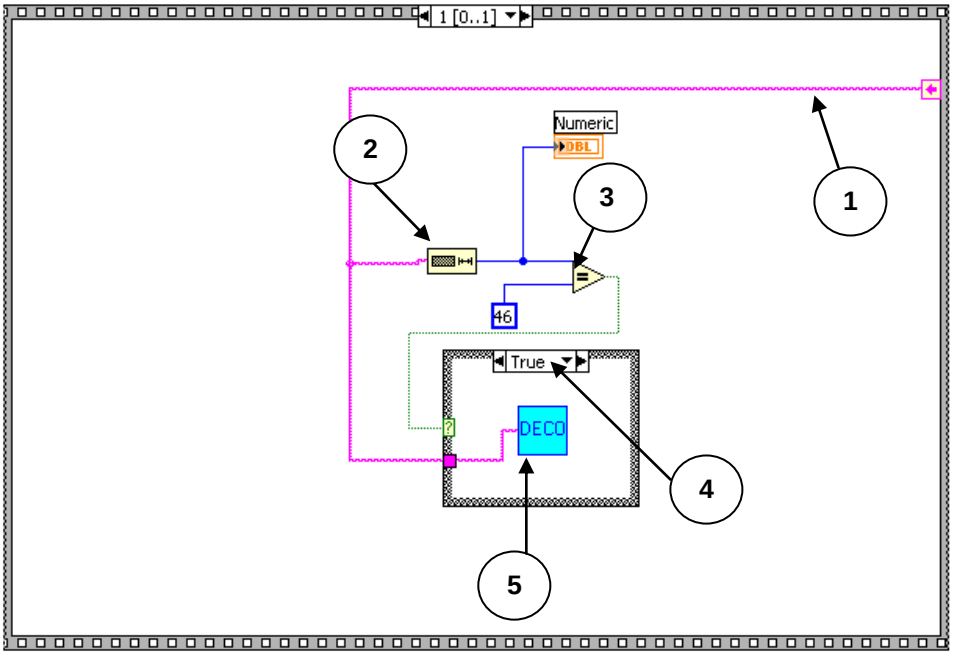


Figura VI.13. Instrumento virtual get data.

El instrumento virtual *deco*, con el símbolo 

, se programó aparte, ya que la decodificación de este comando es muy extensa, como se puede observar en la figura VI.14.

La programación comienza tomando la cadena a la cual se le quitan los 4 primeros bytes correspondientes al *response_mode* y al *data_count*, y quedan 42 bytes (1). A partir de aquí se va decodificando byte por byte usando el *Index Array* (2), por ejemplo, los dos primeros bytes corresponden al *status*, entonces con la ayuda del objeto byte (3) que se había programado anteriormente, estos bytes se descomponen en bits y las salidas van conectadas a *displays* (4) que se encuentran en el panel frontal de este instrumento virtual.

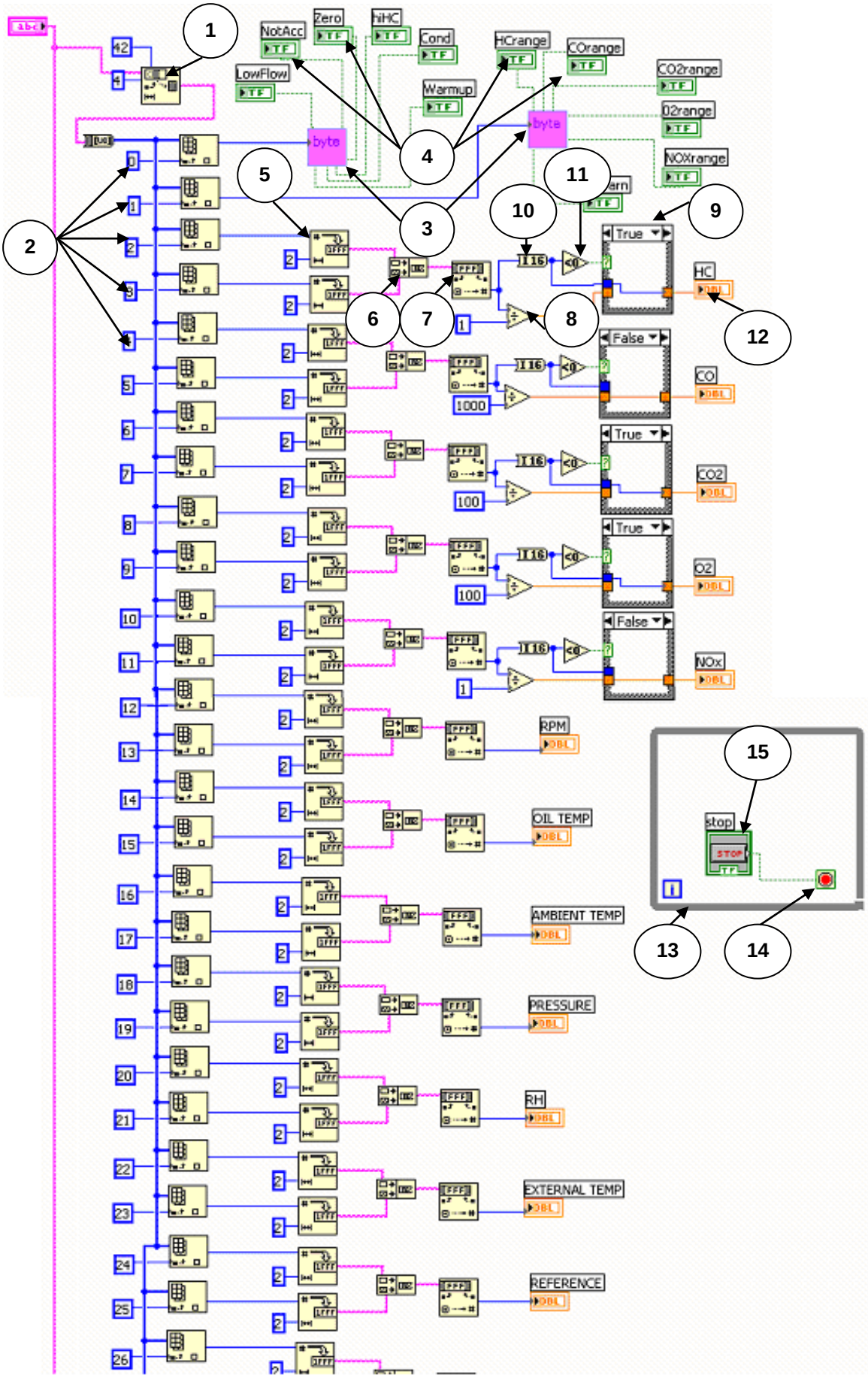


Figura VI.14. Instrumento Virtual Deco.

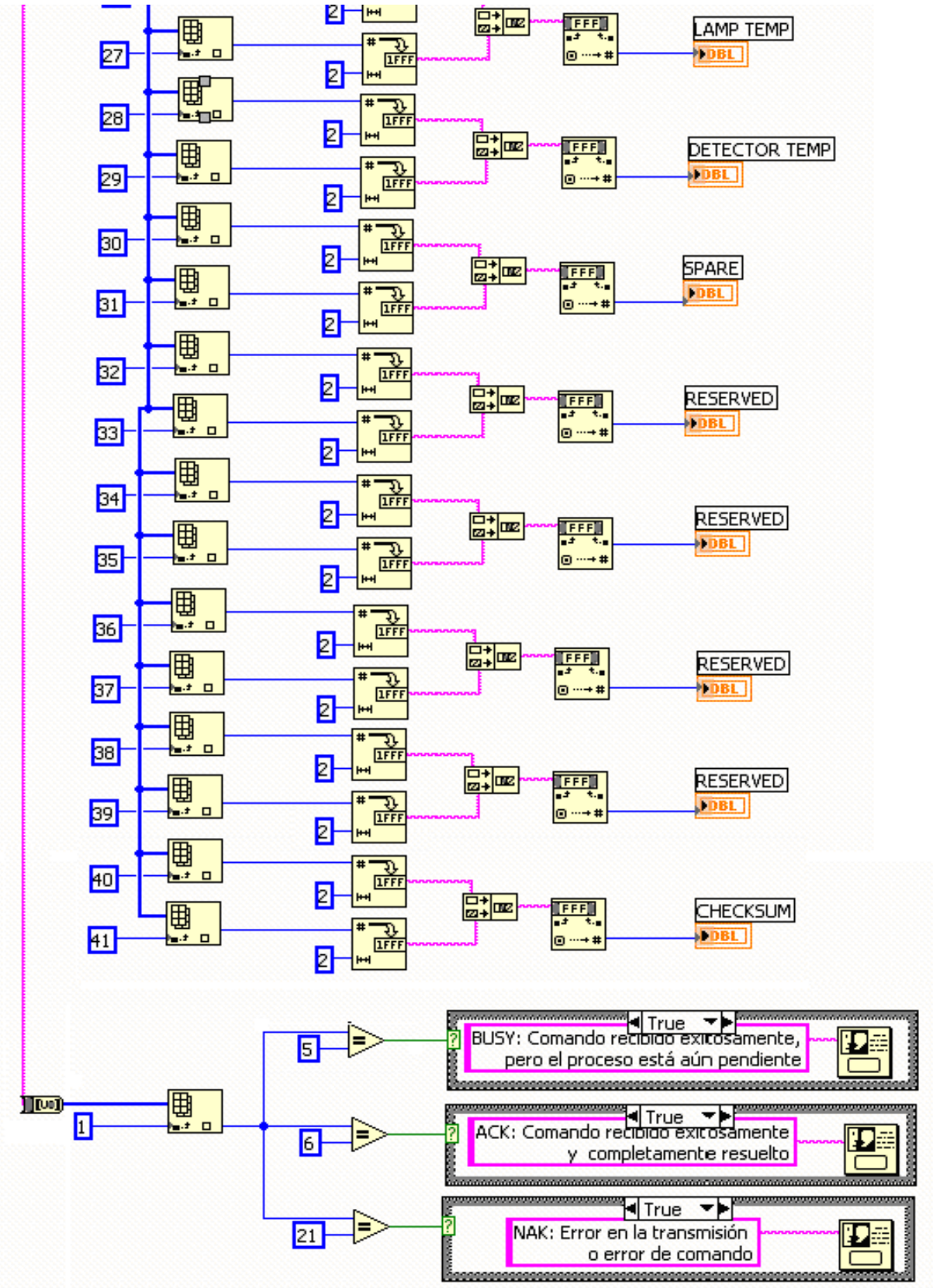


Figura VI.14. Instrumento Virtual Deco.

(Continuación)

Los siguientes 40 bytes se manejan en pares, es decir, se junta la palabra completa de cada *data*. Para esto, cada palabra se trata de la siguiente manera: primero el número de cada bytes se transforma a cadena (5), y ambas se conectan con el objeto *Concat String* (6), esta cadena a su vez se transforma nuevamente en número (7) para desplegarse en un *display,* que también se encuentra en el panel frontal. Para el caso de los índices 2, 3, ..., 11, este número, antes de llegar al *display*, debe dividirse (8) entre un factor, que depende del tipo de gas, para poder obtener la concentración del mismo en las unidades correctas, por ejemplo, para el HC y el NO_x el factor entre el que se tiene que dividir es 1, para el CO es 1000, y para CO_2 y O_2 es 100. Se utiliza una estructura case (9), en donde el selector viene de una comparación (10) del número obtenido antes de la división después de haber sido convertido (11) en un entero de 16 bits con <0. Si la salida de este comparador es *true*, entonces se deja pasar el entero de 16 bits al *display* (12). Si la salida es *false*, se despliega en el *display* el número resultante de la división.

Aunque si es muy extenso este objeto virtual y puede verse muy complicado, todo es repetición de lo mismo.

Por último, se agregó un botón en el panel frontal, el cual sirve para detener la aparición de la pantalla o panel frontal del objeto *deco (*figura VI.15*)*. Este botón se programó usando una estructura llamada *While Loop* (13), la cual repite la aparición de la pantalla hasta que reciba una señal lógica *true* en el botón (14) de esta estructura, y para poder darle ese valor lógico se le asocia un botón (15) que va al panel frontal, que será oprimido por el usuario cuando desee cerrar la pantalla.

Finalmente, el panel frontal del objeto *deco* se puede observar en la figura VI.15, y la que corresponde al panel frontal del comando *get data* se muestra en la figura VI.16.

Una vez terminado el instrumento virtual, se editó su icono 

correspondiente, para ser, posteriormente, llamado en el programa principal.

Este comando se diseñó de tal forma que, cuando se corra, la primera pantalla que aparezca sea la de la figura VI.16, y posteriormente la de la figura VI.15 como una subventana, la cual puede ser cerrada por el usuario al oprimir el botón *Stop*.

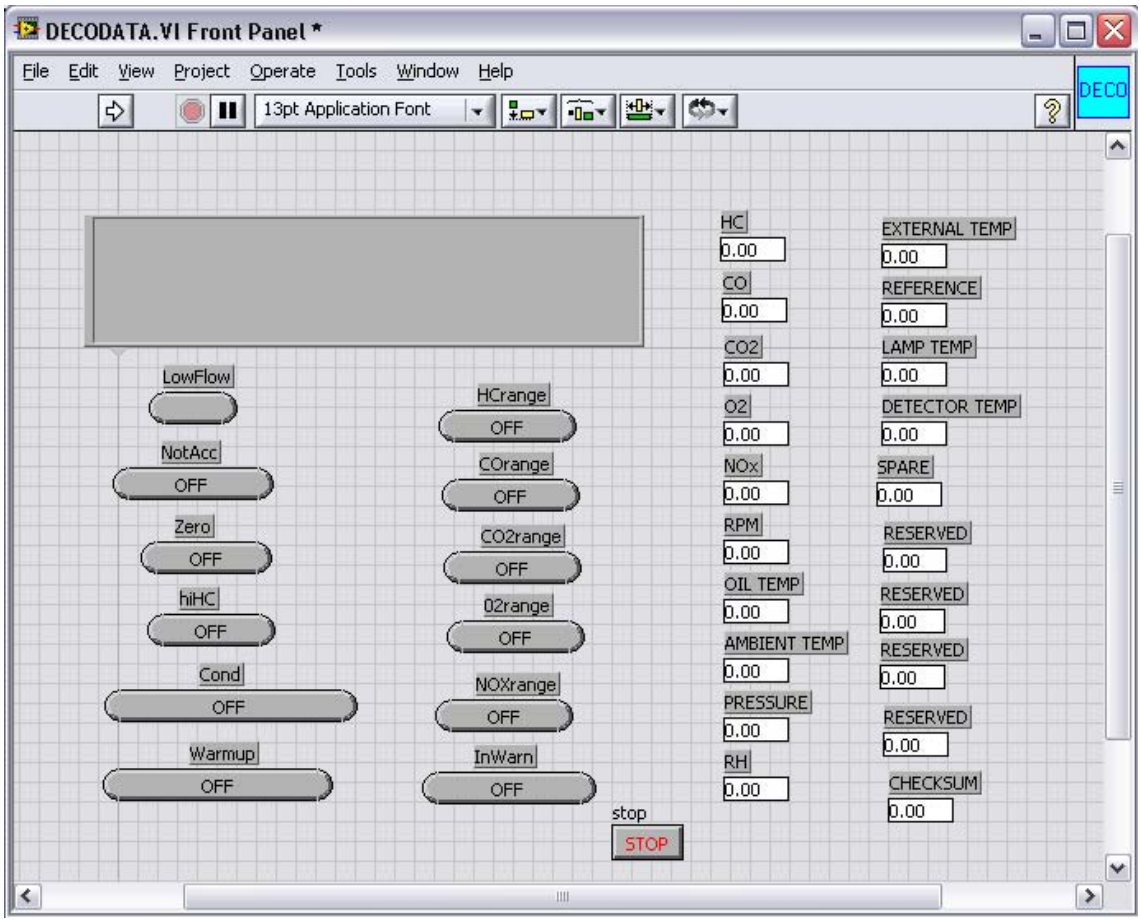


Figura VI.15. Panel frontal del instrumento virtual deco.

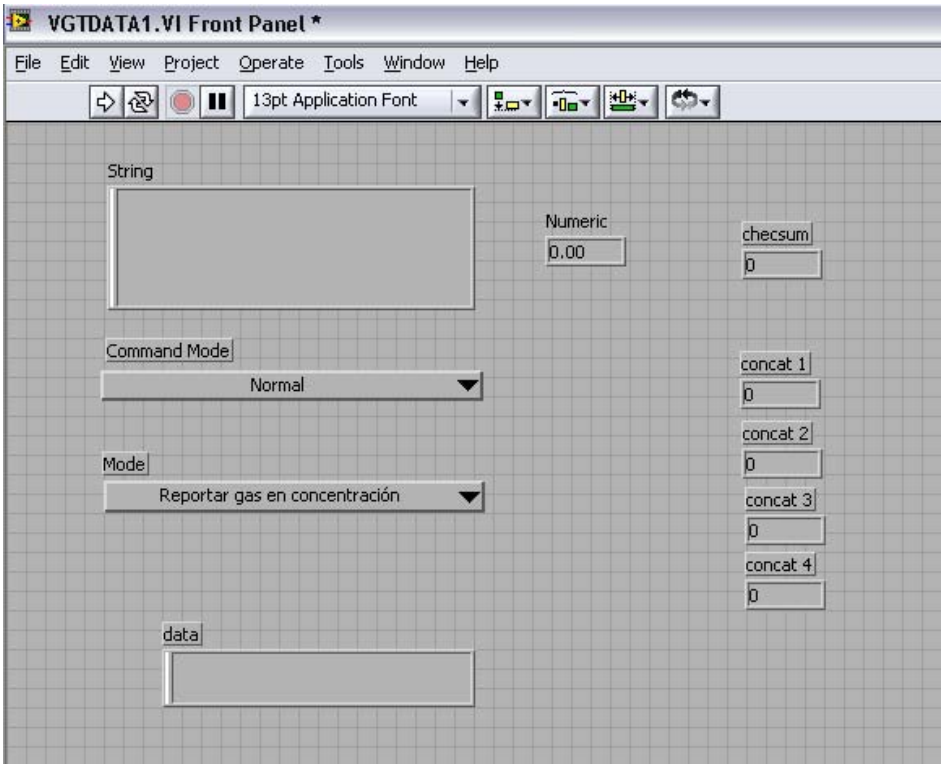


Figura VI.16. Panel frontal del instrumento virtual get data.

VI.1.4.3. Obtener estado (*get status*)

Este comando reporta el estado actual de la banca, engloba lo que es el calentamiento, el *zero*, si se conectó un nuevo transductor para medir oxígeno u óxidos de nitrógeno, advertencias de calibración, etcétera.

La cadena de bytes para la solicitud que se debe enviar en el caso de este comando es:

Solicitud:

<4, *command_mode*><0><*chksum*>

Se puede observar la solicitud a enviar es muy simple, el número de comando correspondiente es el 4, además de que n vale 0, es decir, este comando no requiere del campo *data*.

Para la respuesta se tiene:

Respuesta:

<4, *response_mode*><11>[<*palabra_estado#0*>...<*palabra_estado#10*>] <*chksum*>

La respuesta es un tanto más complicada, n vale 11 dado que se envían 11 palabras en el campo *data* llamadas *palabra_estado* (o también *status_word*): palabra_estado#0, ..., palabra_estado#10; éstas se explican a continuación.

La *palabra_estado#0* da información completa del estado. Su campo de bits es el de la tabla VI.11.

7	6	5	4	3	2	1	0
BadNOx	NewNOx	LowFlow	BadO2	CalZero	NewO2	Zero	Warmup

15	14	13	12	11	10	9	8
Internal	ADrailed	-	-	-	-	-	Cond

Tabla VI.11. Bits del campo palabra_estado#0 para el comando get status.

En la tabla VI.12 se da la descripción de cada bit.

Bit	Descripción
Warm-up	La banca está en calentamiento
Zero	La banca requiere un Zero
NewO2	Nuevo Transductor de O_2 instalado
CalZero	CalZero=1:(SW1==0)&(SW2==0)&(SW3==0)&(SW4==0)& (SW5==0)&(SW9==0)
BadO2	Advertencia de Calibración o Zero para O_2
LowFlow	Valor de tiempo real: 0:J10-Pin1 es 5[V] (TTL) 1:J10-Pin1 es 0[V] (TTL)
NewNOx	Nuevo Transductor de NO_x instalado
BadNOx	Advertencia de Calibración o Zero para NO_x
Cond	Advertencia de Condensación
ADrailed	Si SW7 != 0 entonces ADrailer=1 Si SW7 == 0 entonces ADrailer=0
Internal	Si SW6 != 0 entonces Internal=1 Si SW6 == 0 entonces Internal=0

Tabla VI.12. Valores del campo palabra_estado#0 para el comando get status.

La *palabra_estado#1* es exclusiva para el estado del *Zero*. Su campo de bits es el indicado en la tabla VI.13.

7	6	5	4	3	2	1	0
-	NOxzero	O2zero	REFzero	-	CO2zero	COzero	HCzero

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

Tabla VI.13. Bits del campo palabra_estado#1 para el comando get status.

La descripción de cada bit se proporciona en la tabla VI.14.

Bit	Descripción
HCzero	Advertencia de Zero para HC
COzero	Advertencia de Zero para CO
CO2zero	Advertencia de Zero para CO_2
REFzero	Advertencia de Zero para Ref
O2zero	Advertencia de Zero para O_2
NOxzero	Advertencia de Zero para NO_x

Tabla VI.14. Valores del campo palabra_estado#1 para el comando get status.

La *palabra_estado#2* indica el estado de la calibración de punto simple. Su campo de bits (tabla VI.15), así como su correspondiente tabla de descripción (tabla VI.16) se muestran a continuación.

7	6	5	4	3	2	1	0
-	NOxcal	O2cal	REFcal	HiHCcal	CO2cal	COcal	HCcal

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	Presscal

Tabla VI.15. Bits del campo palabra_estado#2 para el comando get status.

Bit	Descripción
HCcal	Advertencia de calibración de HC
COcal	Advertencia de calibración de CO
CO2cal	Advertencia de calibración de CO_2
HiHCcal	Advertencia de calibración de HiHC
REFcal	Advertencia de calibración de Ref
O2cal	Advertencia de calibración de O_2
NOxcal	Advertencia de calibración de NO_x
Presscal	Reservado para uso futuro (siempre ‘0’)

Tabla VI.16. Valores del campo palabra_estado#2 para el comando get status.

La *palabra_estado#3, a su vez,* indica el estado de la calibración de dos puntos. El campo de bits (tabla VI.17) correspondientes a esta palabra, así como su tabla de descripción (tabla VI.18), son las siguientes:

7	6	5	4	3	2	1	0
-	NOxcal2	-	-	-	CO2cal2	COcal2	HCcal2

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

Tabla VI.17. Bits del campo palabra_estado#3 para el comando get status.

Bit	Descripción
HCcal2	Advertencia de calibración 2 de HC
COcal2	Advertencia de calibración 2 de CO
CO2cal2	Advertencia de calibración 2 de CO_2
NOxcal	Advertencia de calibración 2 de NO_x

Tabla VI.18. Valores del campo palabra_estado#3 para el comando get status.

Tanto la *palabra_estado#4* como la *palabra_estado#5* están reservadas para su uso futuro. Su campo de bits (tabla VI.19), por lo tanto, está vacío:

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

Tabla VI.19. Bits del campo palabra_estado#4 y 5 para el comando get status.

La *palabra_estado#6* sirve para mostrar si hay advertencias internas, que son las mostradas en el campo de bits de la tabla VI.20, y su descripción a su vez en la tabla VI.21.

7	6	5	4	3	2	1	0
-	-	-	-	NOxoffset	O2offset	BlkHtr	-

15	14	13	12	11	10	9	8
EEPROM	Iteration	IRbeam	-	-	-	-	-

Tabla VI.20. Bits del campo palabra_estado#6 para el comando get status.

Bit	Descripción
BlkHtr	Bloque de calentamiento fuera de control
O2offset	Advertencia de offset de O_2
NOxoffset	Advertencia de offset de NO_x
IRbeam	Error de intensidad de emisión NDIR. Si la señal de tensión de HC o CO o CO_2 es menor que el 66% del valor inicial durante la producción
Iteration	Error de iteración
EEPROM	EEPROM incompatible o corrompida

Tabla VI.21. Valores del campo palabra_estado#6 para el comando get status.

La *palabra_estado#7* indica que canales de AD están no activos. A continuación se muestran su campo de bits (tabla VI.22), así como su tabla descriptiva (tabla VI.23).

7	6	5	4	3	2	1	0
Tambrailed	OILrailed	RPMrailed	NOxrailed	O2railed	CO2railed	COrailed	HCrailed

15	14	13	12	11	10	9	8
-	-	Tlmprailed	RHrailed	Tblkrailed	Texrailed	REFrailed	Prailed

Tabla VI.22. Bits del campo palabra_estado#7 para el comando get status.

Bit	Descripción
HCrailed	Canal de HC no activo
COrailed	Canal de CO no activo
CO2railed	Canal de CO_2 no activo
O2railed	Canal de O_2 no activo
NOxrailed	Canal de NO_x no activo
RPMrailed	Canal de RPM no activo
OILrailed	Canal de Aceite no activo
Tambrailed	Canal de Temperatura ambiente no activo
Prailed	Canal de Presión no activo
REFrailed	Canal de Referencia no activo
Texrailed	Canal de Temperatura Externa no activo
Tblkrailed	Canal de Temperatura del bloque de calentamiento no activo
RHrailed	Canal de Humedad relativa no activo
Tlmprailed	Canal de Lámpara no activo

Tabla VI.23. Valores del campo palabra_estado#7 para el comando get status.

La *palabra_estado#8* indica cuando se necesita una calibración en el caso de instalar un nuevo transductor para medir oxígeno u óxidos de nitrógeno. Para esto su campo de bits es el mostrado en la tabla VI.24.

7	6	5	4	3	2	1	0
O2zero	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	NOxcal1	NOxzero	O2cal

Tabla VI.24. Bits del campo palabra_estado#8 para el comando get status.

y su tabla de descripción es la tabla VI.25.

Bit	Descripción
O2zero	Siempre ‘0’
O2cal	Si un nuevo transductor de O_2 es instalado
NOxzero	Si un nuevo transductor de NO_x es instalado
NOxcal1	Si un nuevo transductor de NO_x es instalado

Tabla VI.25. Valores del campo palabra_estado#8 para el comando get status.

Para mostrar el estado del calentamiento de la banca, se hace uso de la *palabra_estado#9*. Su campo de bits corresponde a la tabla VI.26.

7	6	5	4	3	2	1	0
DAC	IRVoltage	REF	TLamp	ADC	Cond	Tblock	BlkRange

15	14	13	12	11	10	9	8
Unstable	-	-	-	-	-	-	-

Tabla VI.26. Bits del campo palabra_estado#9 para el comando get status.

Y su tabla descriptiva es la que se muestra en la tabla VI.27.

Bit	Descripción
BlkRange	Boque detector de temperatura fuera de rango
Tblock	Bloque detector de temperatura inestable
Cond	Temperatura de lámpara < temperatura ambiente
ADC	Convertidor analógico digital está no activo
TLamp	Temperatura de lámpara inestable
REF	Canal de referencia inestable
IRVoltage	Baja tensión en el canal IR
DAC	DAC inestable
Unstable	La banca es inestable

Tabla VI.27. Valores del campo palabra_estado#9 para el comando get status.

La información en esta palabra de estado puede ser usada para determinar la causa de porque la banca está aún en calentamiento.

Por último para este comando, la *palabra_estado#10* da un tiempo aproximado de calentamiento en segundos (tabla VI.28).

	Descripción
SW 10	Tiempo restante estimado de calentamiento en segundos

Tabla VI.28. Valores del campo palabra_estado#10 para el comando get status.

VI.1.4.3.1. Instrumento Virtual obtener estado

La programación del instrumento virtual *get status* es muy similar a la del *get data*, la metodología es la misma: se envía la solicitud, se recibe la respuesta y la decodificación se realiza en otro objeto virtual *deco* diferente al del instrumento Obtener Dato.

Para este instrumento virtual la parte de la solicitud hace referencia a la figura VI.17, y la cadena a enviar es la siguiente:

<4, *command_mode*><0><*chksum*>

De hecho, como ya se mencionó, esta parte es muy sencilla para este comando, primero se manda un 4 (1), seguido del *command_mode* (2), contiguamente van dos 0 correspondientes al LSB y MSB del *data_count* (3), y finalmente la suma para el *checksum* (4). Como ya se sabe, estos elementos se unen en un arreglo (5) para, posteriormente, ser convertido en cadena (6) y ser enviado por el puerto serial (7).

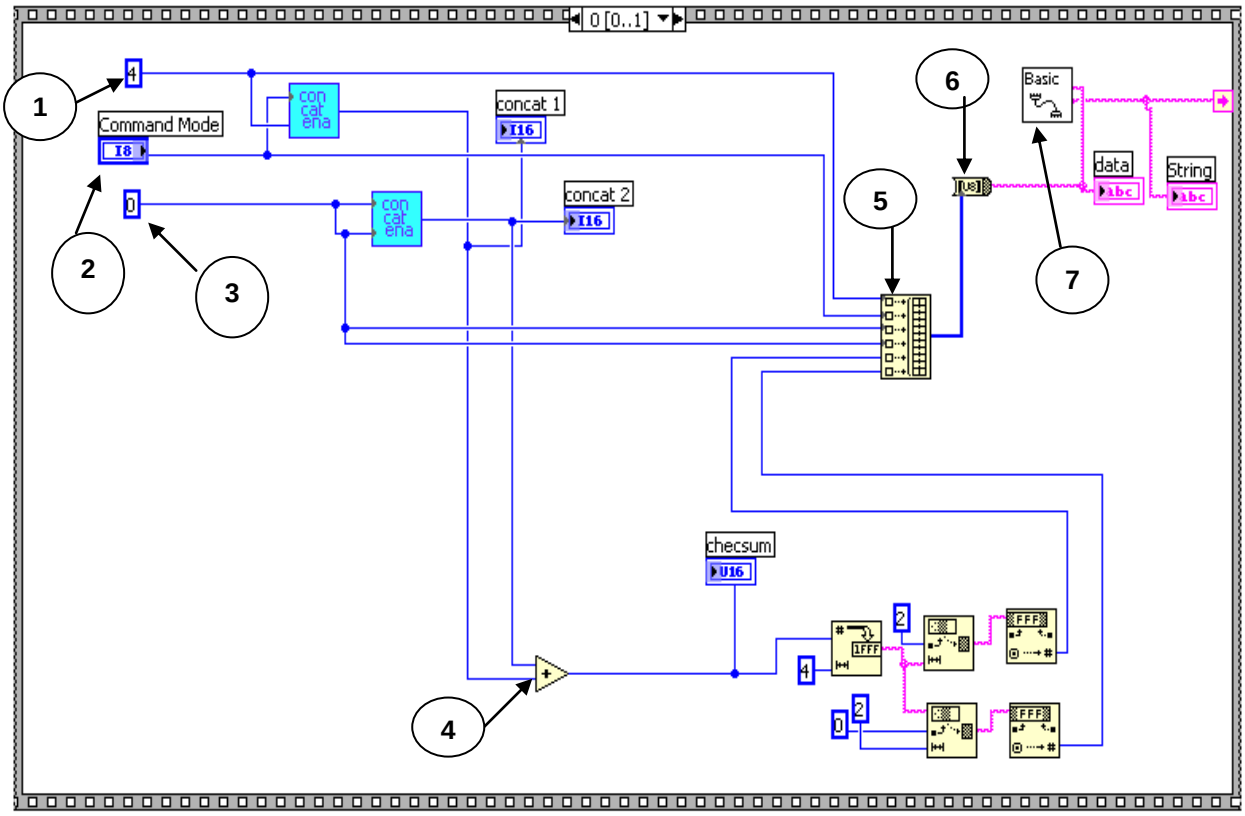


Figura VI.17. Instrumento virtual get status.

Una vez que se lee el puerto serial, hay que decodificar la respuesta:

<4, *response_mode*><11>[<*palabra_estado#0*>...<*palabra_estado#10*>]<*chksum*>

Para esta parte se refiere a la figura VI.18. En ésta primero se obtiene la longitud de la cadena (1), la cual se compara (2) con 28: 2 bytes del *command* y *response_mode* + 2 bytes del *data_count* + 22 bytes del *data* + 2 bytes del *checksum*. Si la salida de este comparador es *true* se procede a realizar la decodificación de los datos. Para esto se hace uso del objeto *deco* (3) que se explica a continuación.

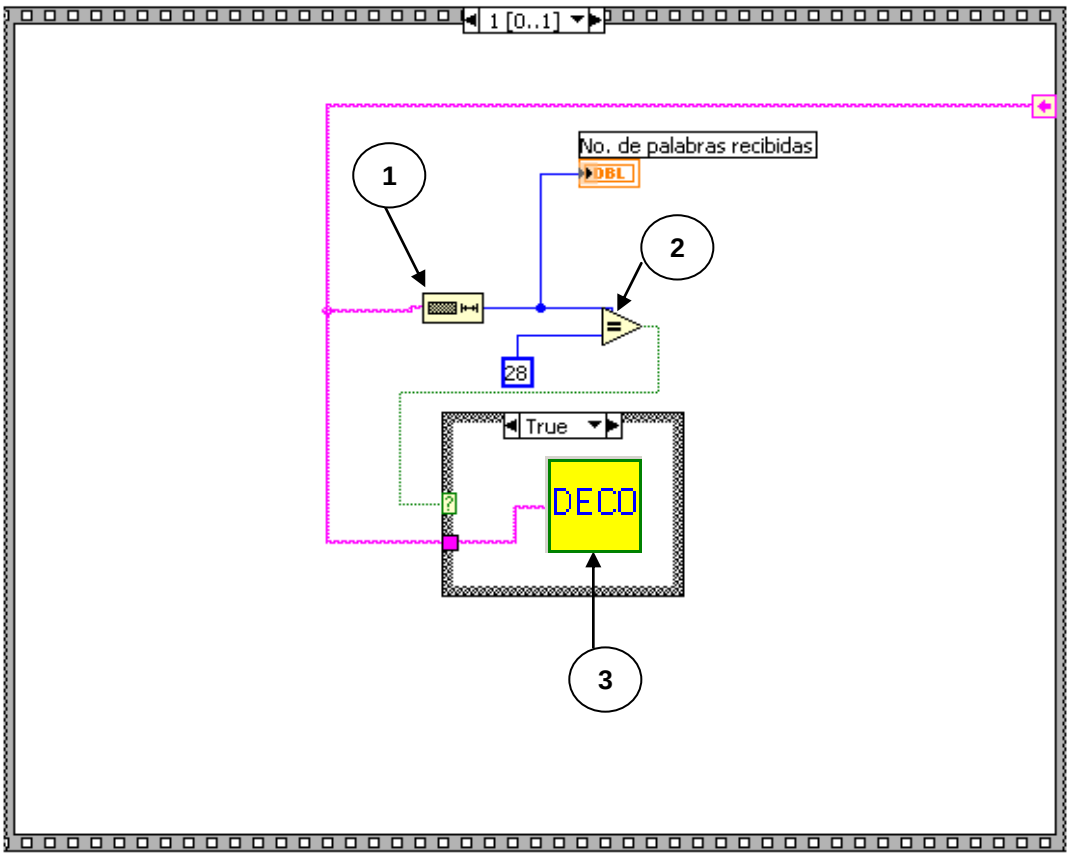


Figura VI.18 Instrumento virtual get status.

El objeto *deco* programado para este comando, con el símbolo 

y mostrado en la figura VI.19, es muy similar al ocupado en el comando anterior. Para este deco de este comando se toma la cadena leída y se le sustraen 4 bytes: el *command*, el *response_mode* y 2 del *data_count*, quedando así 24 bytes de los 28 recibidos (1). Después de esta última acción se van decodificando cada uno de los bytes, recordando que vienen en parejas, ya que una pareja forma una palabra. Por ejemplo, los bytes 0 y 1 (2) son los que corresponden a la *palabra_estado#0*, la que con ayuda del objeto byte (3), se subdivide en los bits que la conforman (ver Tabla VI.11), mostrándose éstos en *displays* (4) que dan al panel frontal.

Lo expuesto anteriormente aplica para cada pareja, excepto para los bytes 7, 8, 9 y 10, que corresponden a las palabras: *palabra_estado#4* y *palabra_estado#5* (5), las cuales están reservadas para un uso futuro, y por lo tanto, la salida de todos sus bits son cero y no tiene caso que aparezcan en un *display*.

Para la *palabra_estado#10*, que tiene los bytes 20 y 21, se unen estos últimos con un concatenador (6) y se muestra el valor de la palabra en un *display* (7).

Por último, aquí también se ocupa un botón stop (8), el cual fue programado de la misma manera que se explicó en el objeto *deco* para el instrumento virtual Obtener Dato.

Dado que las imágenes del programa realizado en LabVIEW son muy extensas, es difícil que se pueda obtener una sola imagen que contenga todos los elementos programados, por esto, la figura VI.19 aparece por partes, tratando de mantener el orden del programa original.

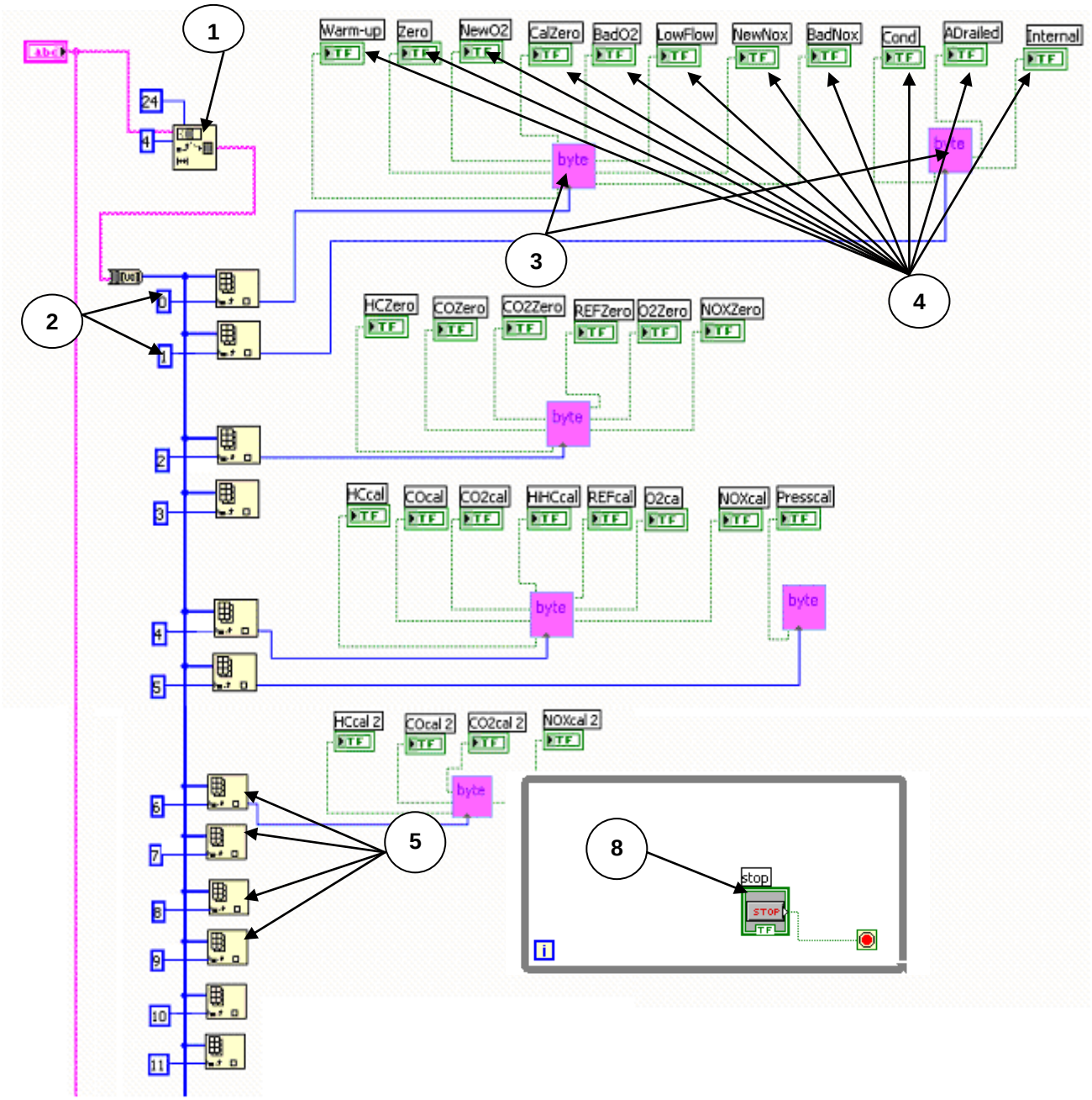


Figura VI.19. Instrumento virtual deco.

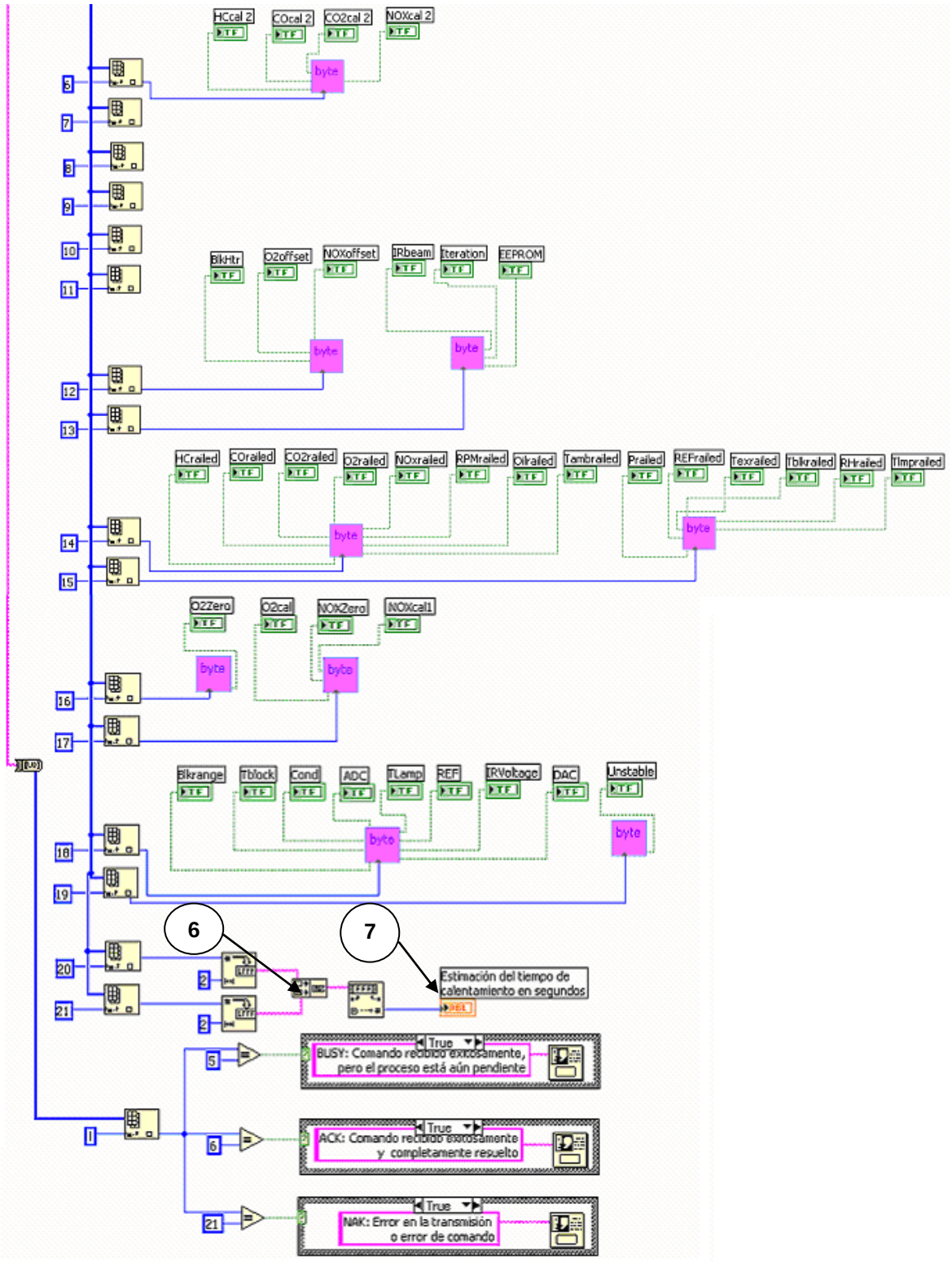


Figura VI.19. Instrumento virtual deco (continuación).

Finalmente, el panel frontal del objeto *deco* es el indicado en la figura VI.20, en el cual son mostrados todos los valores que son arrojados a los *displays*; la mayoría de estos son booleanos (sólo pueden tener dos valores: *on* u *off*).

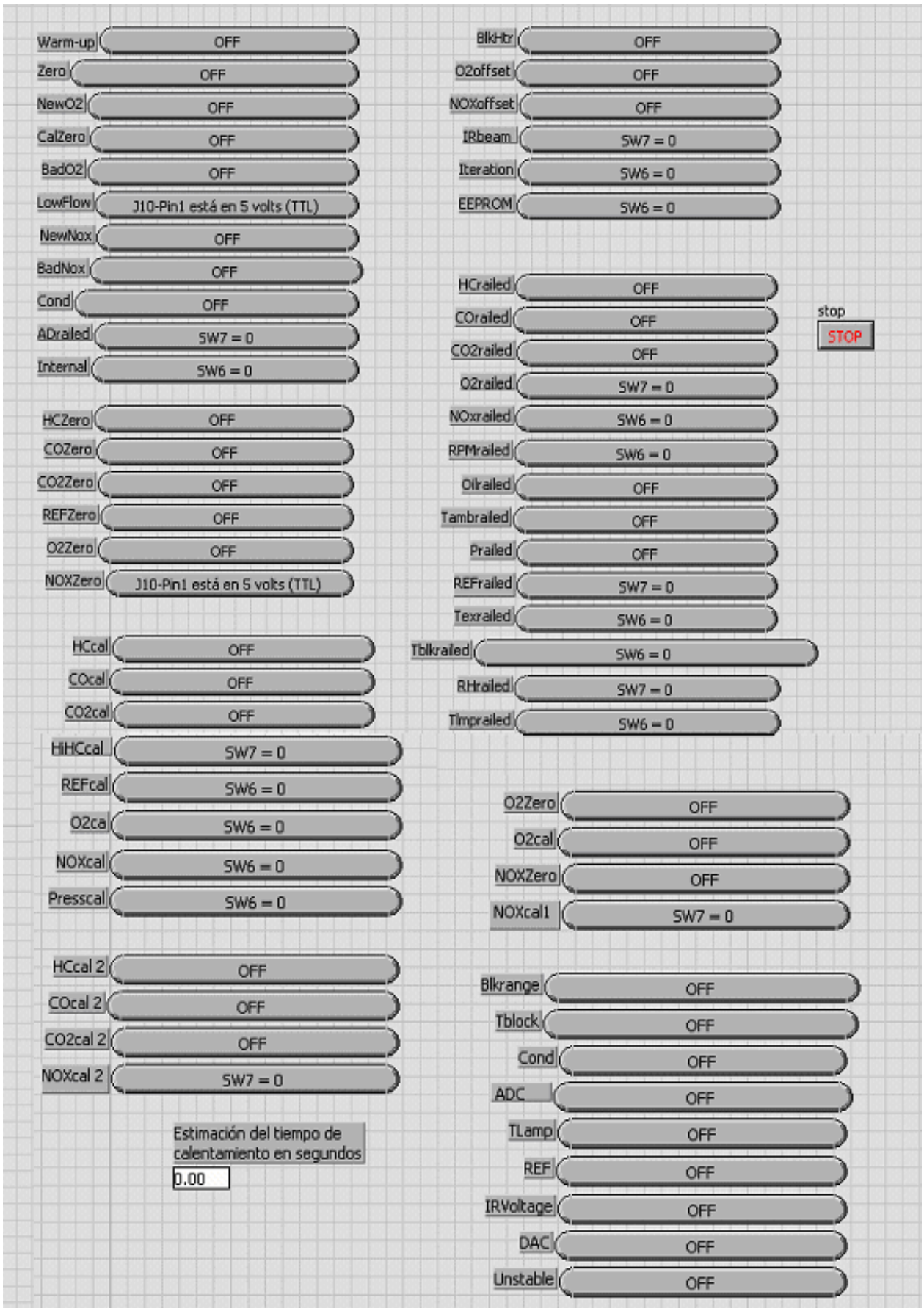


Figura VI.20. Panel frontal del instrumento virtual deco.

Una vez terminado el instrumento virtual, se editó su icono correspondiente 

, para ser, posteriormente, llamado en el programa principal.

VI.1.4.4. Lectura/Escritura de entradas/salidas (*Read/Write I/O's*)

Este comando sirve para leer lo que hay en alguna de las entradas de la banca, como también para escribir un 0 o 1 lógicos en alguna de las salidas de la misma. Las entradas y salidas de la banca se mencionaron en el capítulo IV. Descripción del hardware.

Los campos que hay que enviar para este comando son:

Solicitud:

<8, *command_mode*><n><*read/write*><*mode*>[<*lodata*>]<*chksum*>

Al igual que en el comando anterior, el valor de *n* no es fijo, en esta ocasión depende del campo *lodata*, el cual a su vez depende del modo (*mode*) que se haya escogido. El campo *read/write* pone en operación la lectura o escritura y, otra vez, sólo puede tener dos valores posibles (tabla VI.29).

	Descripción
0	Leer entradas/salidas
1	Escribir entradas/salidas

Tabla VI.29. Valores del campo read/write para el comando Read/Write I/O's

El campo *mode* selecciona el tipo de operación de entrada/salida. En la tabla VI.30 se muestran los valores que puede tomar este campo, así como sus respectivas descripciones y la indicación de si aplica en un modo de sólo lectura, de sólo escritura o de lectura y escritura.

Modo	Descripción	r/w
0	Seleccionar mapa solenoide	r/w
1	Seleccionar calibración solenoide 1	r/w
2	Seleccionar calibración solenoide 2	r/w
4	Seleccionar solenoide 1	r/w
8	Seleccionar solenoide 2	r/w
16	Seleccionar bomba simple	r/w
32	Seleccionar bomba de desagüe	r/w
64	Leer Sensor de Bajo Flujo	r/
128	Seleccionar mapa físico I/O	r/w

Tabla VI.30. Valores del campo mode para el comando Read/Write I/O's.

En el campo *lodata* es en donde se ponen las salidas en *on* u *off*. Este campo se puede dividir en 3 diferentes, dependiendo del *mode*.

El *lodata* para *mode*=0 presenta el campo de bits mostrado en la tabla VI.31.

7	6	5	4	3	2	1	0
-	-	Drain Pump	Sample Pump	Air Sol 2	Air Sol 1	Cal Sol 2	Cal Sol 1

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

Tabla VI.31. Bits del campo lodata cuando mode=0 para el comando Read/Write I/O's.

El significado de cada bit se muestra en la tabla VI.32.

Bit	Significado
Cal Sol 1	Calibración Solenoide 1
Cal Sol 2	Calibración Solenoide 2
Air Sol 1	Aire del Solenoide 1
Air Sol 2	Aire del Solenoide 2
Sample Pump	Bomba Simple
Drain Pump	Bomba de Desagüe

Tabla VI.32. Valores del campo lodata cuando mode=0 para el comando Read/Write I/O's.

El *lodata* para *mode*=1, 2, 4, 8, 16, y 32 se pone en 0 si el *read/write*=0 y se pone en 1 si *read/write*=1, es decir (tabla VI.33):

Read/write	Descripción
0	Salida OFF
1	Salida ON

Tabla VI.33. Valores del campo lodata para el mode=1, 2, 4, 8, 16, y 32.

El *lodata* para *mode*=128 tiene el campo de bits siguiente (tabla VI.34):

7	6	5	4	3	2	1	0
Air Sol 1	Drain Pump	Sample Pump	Air Sol 2	-	Cal Sol 2	Cal Sol 1	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

Tabla VI.34. Bits del campo lodata cuando mode=128 para el comando Read/Write I/O's.

El significado de los bits es el mismo que el de la tabla VI.32.

En la respuesta se tiene:

Respuesta:

<8, *response_mode*><n>[<*lodata*>]<*chksum*>

Para este comando, independientemente del modo, si se hace una lectura de una entrada, es decir, *read/write*=0, se reciben 8 palabras porque la respuesta nos la da en el campo *lodata*. Para el caso de que *read/write*=1, es decir, se está escribiendo en una salida, ya no esperamos que nos responda en el campo *lodata*, y por lo tanto sólo se reciben 6 palabras.

Este comando puede ser útil para controlar, desde la computadora, las salidas que tiene la banca, y con esto, poder activar alguna bomba, ventilador, válvula o cualquier otro dispositivo que se requiera. Obviamente la conexión directa del dispositivo a la banca no funcionaría, debido a que a la salida de la banca sólo podemos tener valores lógicos (TTL) de

0 y 5 [V], y si se trata de un motor, por ejemplo, que trabaja con 12 [V] es necesario utilizar un relevador el cual se active con un valor lógico y tenga una salida mínima de 12 [V] para que funcione la bomba.

Para el caso de esta tesis, se utilizó la salida *Air* para controlar la válvula solenoide que selecciona la muestra que se va a analizar, ya sea la obtenida de un automóvil o la que proviene del aire para realizar un *Zero*. Cabe mencionar que toda esta explicación de la conexión del relevador con la válvula está indicada en el capítulo anterior.

VI.1.4.4.1. Instrumento Virtual Lectura/Escritura de entradas/salidas.

La programación de este instrumento virtual es mucho más compleja que la de los anteriores, así que se tratará de ser lo más claro posible. La siguiente explicación toma como referencia la figura VI.21. La cadena a enviar para realizar la solicitud es:

<8, *command_mode*><n><*read/write*><*mode*>[<*lodata*>]<*chksum*>

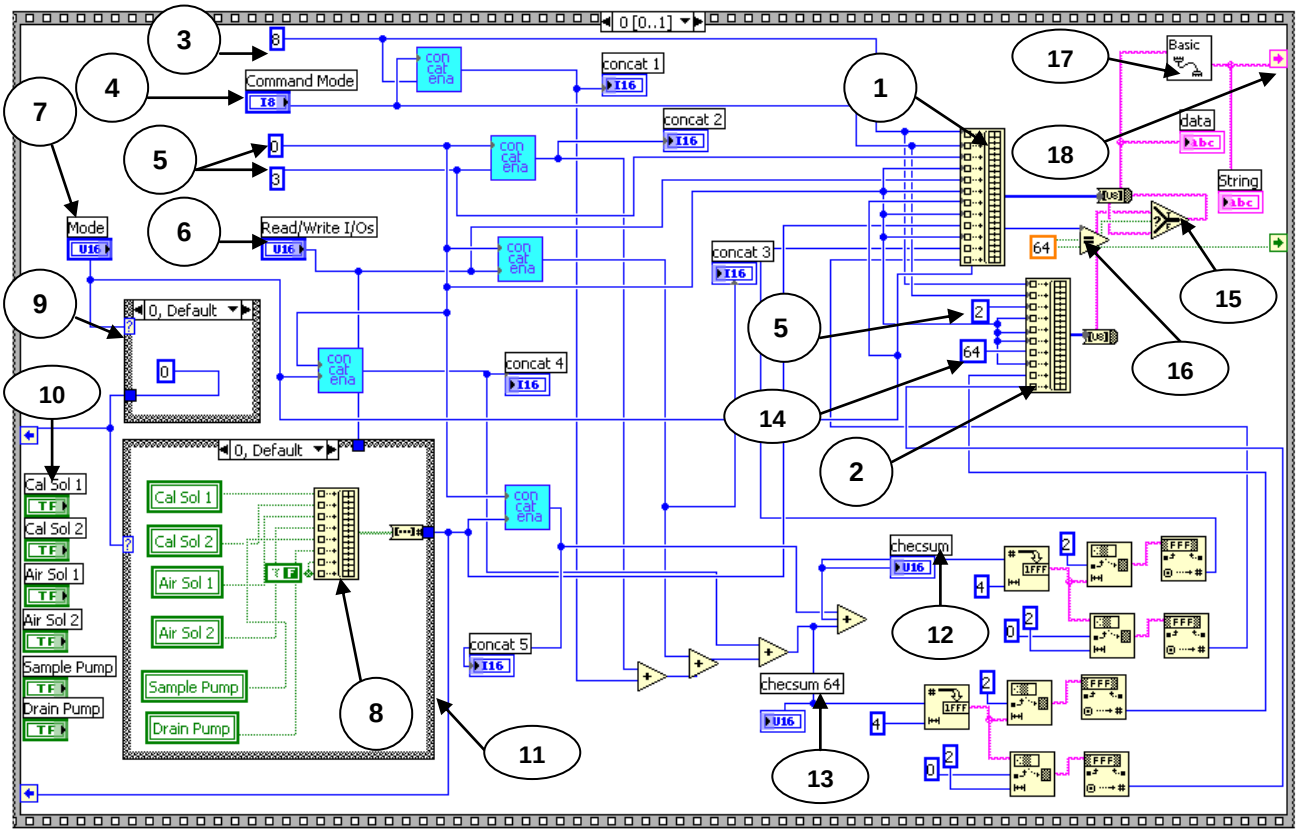


Figura VI.21. Instrumento Virtual Read/Write I/O's.

Como se pudo observar en las tablas que rigen este comando, hay 9 tipos de modos: 0, 1, 2, 4, 8, 16, 32, 64 y 128. De estos, todos son de lectura y escritura excepto el número 64, que es de sólo lectura, por esto este último recibirá un trató diferente a todos los demás. Entonces en este instrumento virtual, la solicitud se puede dividir en dos partes: la correspondiente al *mode* = 0, 1, 2, 4, 8, 16, 32, 128 y la que corresponde al *mode* = 64.

Para el caso de que *mode* = 0, 1, 2, 4, 8, 16, 32, 128; sí aparece el campo *lodata* y n toma entonces el valor de 3. Cuando *mode* = 64, no aparece el campo *lodata* y, por lo tanto, n = 2.

Para llevar a cabo lo descrito en los párrafos anteriores, se utilizaron arreglos (1 y 2). El arreglo (1) es para cuando *mode* = 0, 1, 2, 4, 8, 16, 32, 128 y el arreglo (2) es si *mode* = 64. Primero se manda un 8 (3), el *command_mode* (4) y los dos bytes de n (5), el cual, como ya se mencionó, puede ser 3 ó 2, según sea el caso. Después sigue el *read/write* (6), que es un control colocado en el panel frontal y tiene dos opciones (figura VI.22):

Figura VI.22. Control del read/write .

Posteriormente sigue el *mode* (7), que también es un control en el cual el usuario elige el modo en el que se va a trabajar. Este control tiene las opciones mostradas en la figura VI.23.

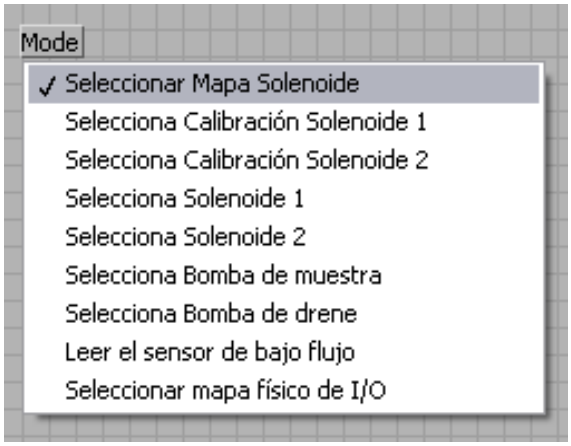


Figura VI.23. Control del mode.

Para el caso de que *mode* = 0, 1, 2, 4, 8, 16, 32, 128, el campo *lodata* se forma con otro arreglo, en el cual se unen cada uno de los controles, los cuales el usuario puede activar o desactivar en el panel frontal (figura VI.24).

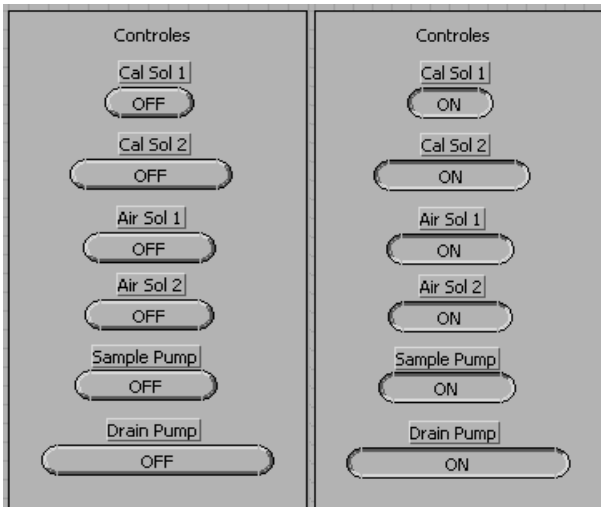


Figura VI.24. Controles para el lodata.

El orden en que cada uno de estos controles entra en el arreglo (8) depende del campo *mode*. Para solucionar este problema, se creó una estructura case (9), en donde cada caso toma el valor de *mode*. Se puede observar en la figura VI.25, que el comportamiento del campo *mode* se divide en tres partes: la primera para *mode* = 0, la segunda para *mode* = 1, 2, 4, 8, 16, y 32, y la tercera para *mode* = 128. Cada caso de la estructura *case* dará como salida un número: 0, 1 ó 2; correspondientes respectivamente a las partes 1ª, 2ª o 3ª.

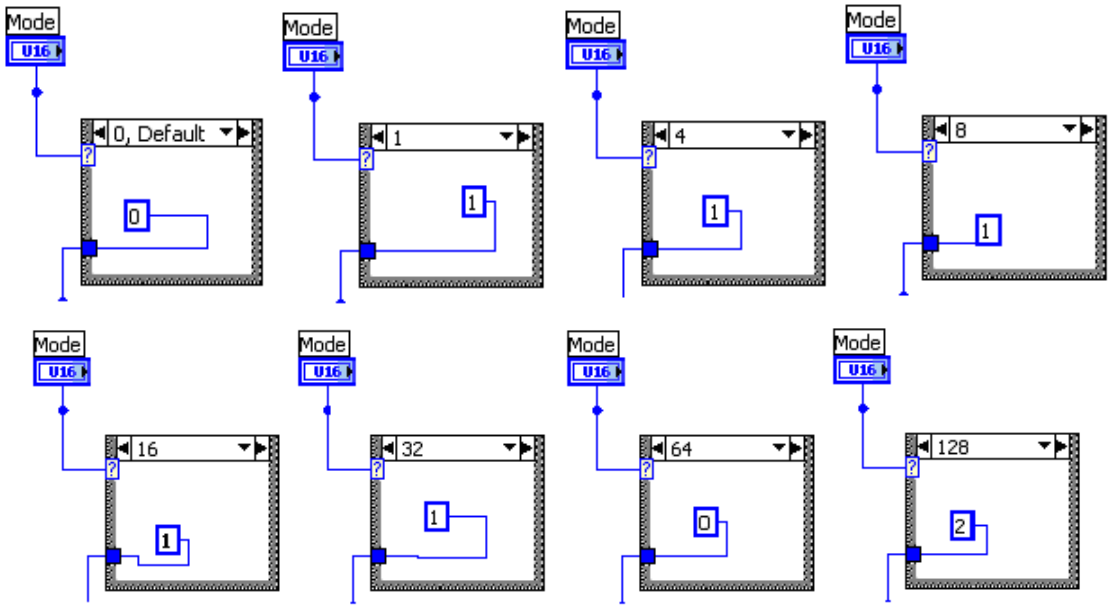


Figura VI.25. Estructura case para los valores del mode.

En los casos mostrados anteriormente se observar que se encuentra el número correspondiente al *mode* = 64, el usuario se ha de estar preguntando, ¿porqué si se supone que este caso no contempla el campo *lodata,* sigue el mismo trato que los demás? La verdad es que no importa el valor que se dé en este caso, sólo se pone porque como se toma directamente lo que se pone en el control *mode*, si este caso no se crea y en *mode* se selecciona la opción 64, entonces no se sabría que caso escoger aunque en realidad no se vaya a ocupar.

Una vez que se obtiene el número: 0, 1 ó 2, se procede a crear una segunda estructura case, en la cual se designará, el orden de los controles anteriormente mencionados, en donde ahora cada uno de los casos tomará los valores 0, 1 ó 2. Para esta parte se hizo uso de las llamadas variables locales (10). Como la salida de los controles son las mismas pero se van a utilizar en dos casos distintos en diferente orden, si se ponen sólo los controles así y se jalan a un caso, ya no se podrían poner en el otro caso, entonces para eso se ocupan las variables locales (figura VI.26), las cuales hacen referencia a esos controles y se pueden generar tantas veces sea necesario para ponerse en cada uno de los casos.

Figura VI.26. Variables locales utilizadas como controles.

Ahora sí, teniendo ya las variables locales, se ordenan en la estructura case (11) según corresponda el caso, dependiendo del número 0, 1 ó 2, como se muestra en la figura VI.27.

Hay que recordar que, como ya se dijo, el caso 0 es para cuando *mode* = 0, el caso 1 es para *mode*=1, 2, 4, 8, 16, y 32, y el caso 2 para *mode*=128. En el caso 1 se puede notar que sólo se hace pasar el valor que toma *read/write*, esto es porque, para este caso, simplemente si *read/write* = 0, la salida es 0, y si *read/write* = 1, la salida es 1.

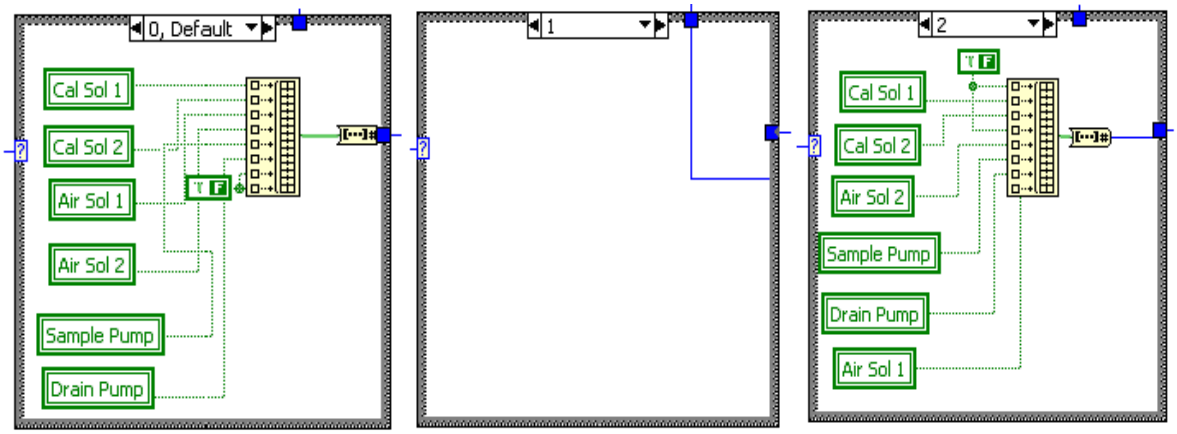


Figura VI.27. Estructuras case para los valores del mode.

Ya por último, se realiza el *checksum,* como se ha hecho en los comandos pasados, sólo que en esta ocasión serán dos *checksum* en lugar de uno, porque son dos arreglos diferentes (1 y 2) y cada uno lleva un *checksum* también diferente. El primer *checksum*, simplemente se llamó así *checksum* (12) y al segundo se le llamó *checksum* 64. El *checksum* 64 (13) sólo se toma antes de que se realice la suma con lo del campo *lodata*, la suma anterior a eso es la misma. Después de esto se juntan los datos en cada uno de los arreglos, poniendo un 64 (14) directamente en el *mode* del arreglo (2). Así entonces, se obtienen dos salidas que se convierten en cadena.

Posteriormente se debe de escoger cual de las dos salidas es la que se va a mandar por el puerto serial para hacer la solicitud, para esto se hace uso del elemento *Select* (15), éste sirve para escoger una de dos entradas, dependiendo de una variable lógica. Para este caso las dos entradas son las dos cadenas, y la variable booleana sale de la comparación (16) del *mode* con 64; si *mode* = 64 la salida del comparador será *true* y entonces se deja pasar la cadena de cuando *mode* es 64; en cualquier otro caso dejará pasar la otra cadena. Una vez que se tiene resuelto cual cadena se debe dejar pasar, ésta se manda por el puerto serial (17), y a la vez se lee lo que la banca haya respondido, lo cual se envía a la secuencia 2, por medio de un *sequence local* (18) para su próxima decodificación.

Para la parte de la decodificación, se sabe que la cadena recibida es la que se muestra a continuación, recordando que el campo *lodata* aparecerá sólo en los casos diferentes a *mode* = 64.

<8, *response_mode*><n>[<*lodata*>]<*chksum*>

Para empezar, observemos en la figura VI.28, aquí se obtiene la longitud (1) de la cadena recibida y ésta se va a comparar (2) con el número de palabras que se esperan recibir si la respuesta está completa, pero, nuevamente, esto depende del *mode*, porque si el campo *lodata* no aparece, entonces se tendrán dos palabras menos; es decir, si *mode* ≠ 64 el número de palabras a comparar será 8, si *mode* = 64 el número de palabras a comparar será 6. Para resolver esta cuestión se hizo uso, otra vez, del elemento *select* (3) en donde las dos entradas son 6 y 8 y la variable lógica es la misma que resultó de la comparación del *mode* con 64 (16 de la figura VI.21). Si la salida del comparador es *true* (*mode* = 64) entonces se dejará pasar el 6, en cualquier otro caso se dejará pasar el 8. Teniendo esta salida se realiza la comparación (2). Si la salida de este comparador es *true* se procede a realizar la decodificación en la primera estructura case (4).

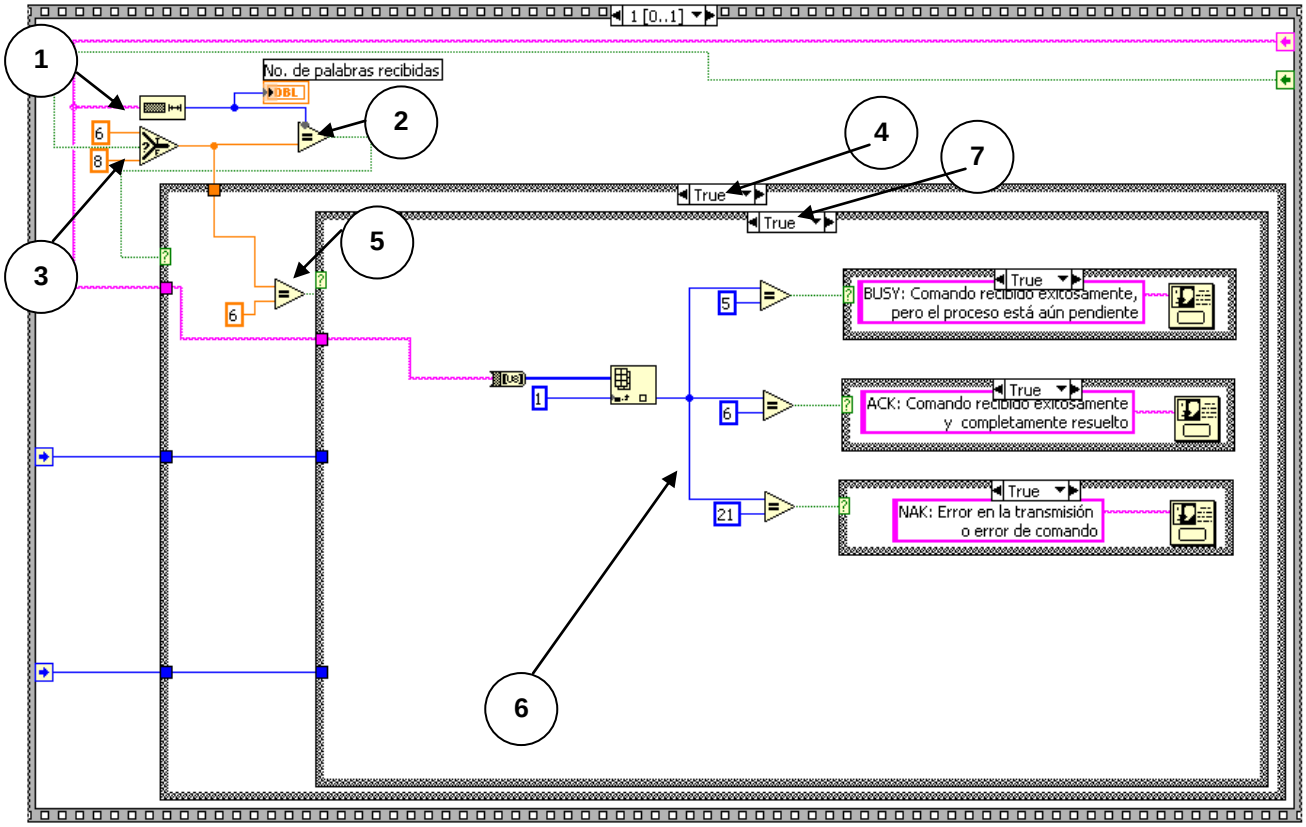


Figura VI.28. Instrumento virtual Read/Write I/O's.

Una vez estando en el primer *case*, hay que saber si se trata de un *mode* igual a 64 o es alguno de los otros. Así que, nuevamente se realiza otra comparación (5) con 6 tomando el valor a comparar de la salida del *select* (3). Si la salida del comparador es *true* cuando se comparó con 6, entonces significa que *mode* es 64 y, por lo tanto, sólo se decodifica lo correspondiente al *response_mode* (6), esto se realiza en una segunda estructura case (7), la cual el caso *true* es para *mode* = 64 y el caso *false* para *mode* = 0, 1, 2, 4, 8, 16, 32, 128.

Si la salida del comparador (5) es *false,* entonces se procede a decodificar lo correspondiente a cada *mode* (*mode* = 0, 1, 2, 4, 8, 16, 32, 128). Cabe recordar que estos se pueden dividir en tres grupos: la primera para *mode* = 0, la segunda para *mode* = 1, 2, 4, 8, 16, y 32, y la tercera para *mode* = 128; según el orden en que se reciben los bits. Para el análisis de estos tres grupos se creó una tercera estructura case (8 de la figura VI.29). Para saber el caso del que se trata, se toma la salida que se dio en el case (9 de la figura VI.21) por medio de un *sequence local* (9 de la figura VI.29), el cual ya determina el número del grupo para cada *mode*. Para cada caso se procede a decodificar el campo *lodata* y lo correspondiente al *response_mode*, aunque este último ya no se mencione.

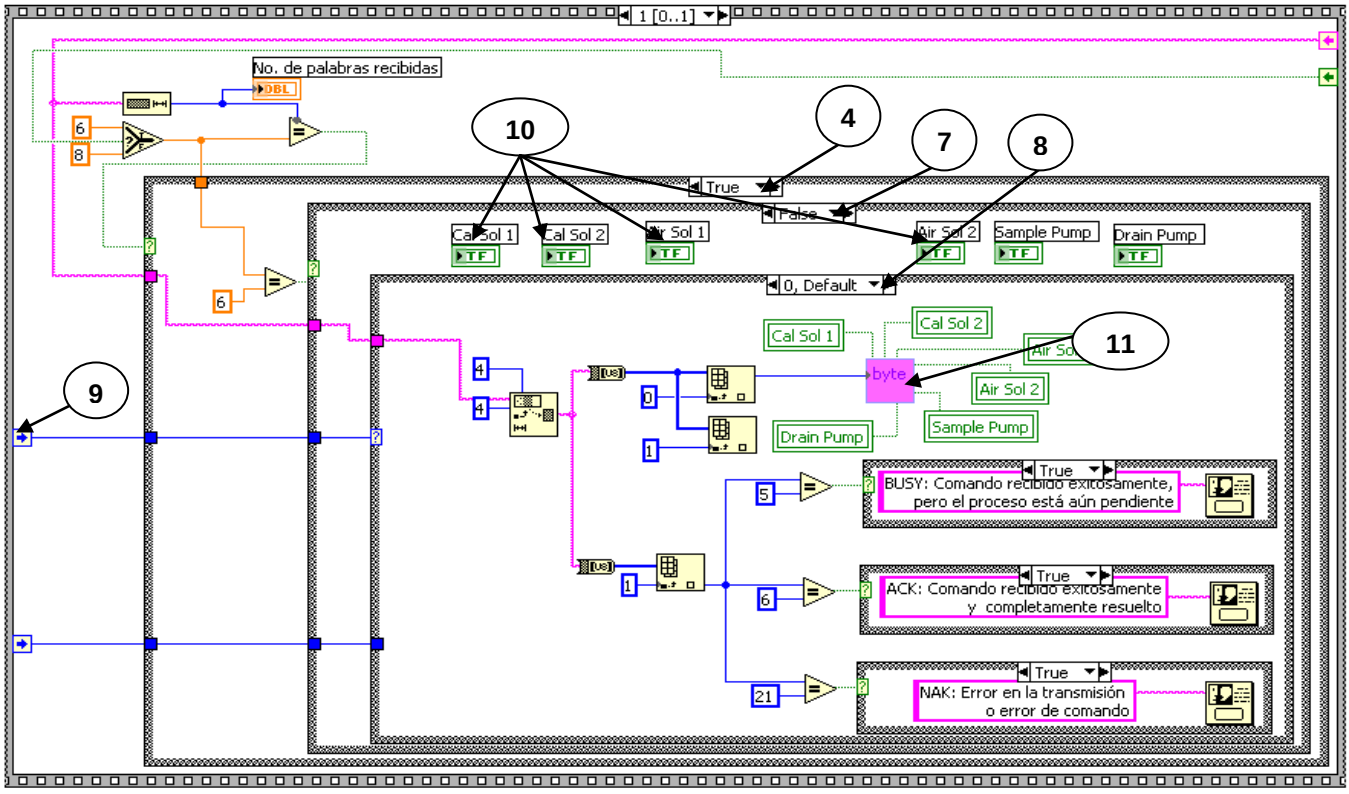


Figura VI.29. Instrumento virtual Read/Write I/O's.

Tanto en el caso 0 como en el caso 3, el campo *lodata* se divide en su campo de bits; ocurre lo mismo que en el case (11 de la figura VI.21), en los dos casos se utilizan los mismos indicadores pero en diferente orden y no se pueden poner en un caso porque entonces ya no se pueden poner en el otro. Por esto, se hace uso otra vez de las variables locales, mostradas en la figura VI.30:

Figura VI.30. Variables locales.

Las variables locales corresponden a los indicadores de la figura VI.31. Dichos indicadores se encuentran en el panel frontal, los que pueden estar activados o desactivados. El indicador *Output* es para el caso 2.

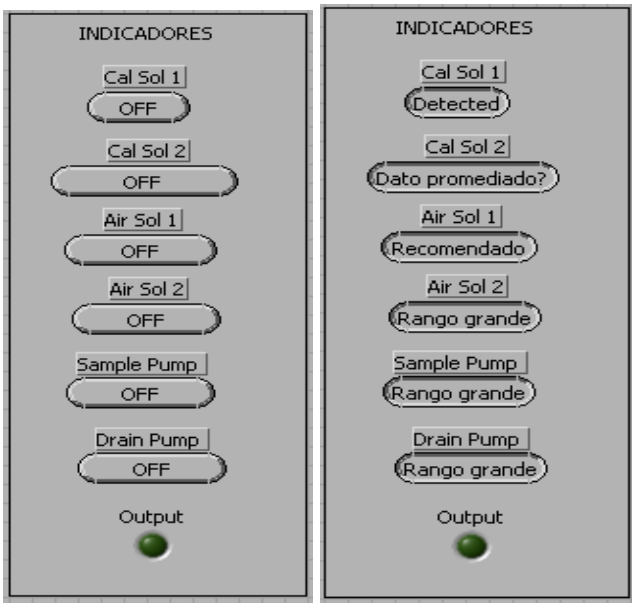


Figura VI.31. Indicadores en el Panel Frontal.

Para el caso 0 (figura VI.29), haciendo uso del objeto byte programado anteriormente (11), los bits van en el siguiente orden:

el bit 0 es Cal Sol 1
el bit 1 es Cal Sol 2
el bit 2 es Air Sol 1
el bit 3 es Air Sol 2
el bit 4 es Sample Pump
el bit 5 es Drain Pump

Para el caso 1 (figura VI.32), si *read/write* es 0 la salida es 0 y si *read/write* es 1 entonces la salida es 1. Para estos dos casos se genera otra estructura case (12), mostrada en la figura VI.33, en donde, en este *case,* nuevamente se utilizan las variables locales para *Output* 

.

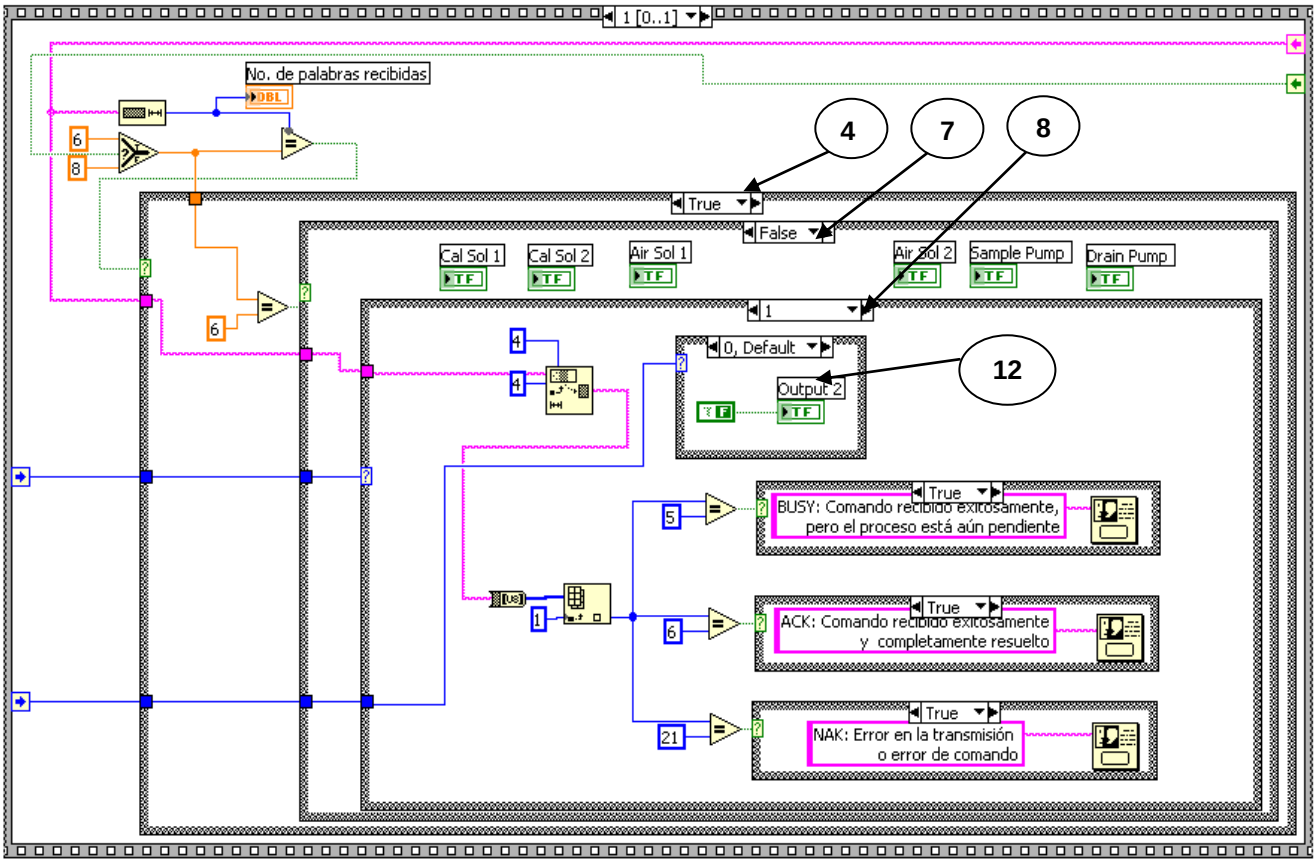


Figura VI.32. Instrumento virtual Read/Write I/O's.

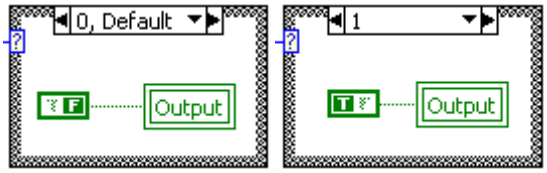


Figura VI.33. Estructura case para el caso 1.

Por último, para el caso 2 (figura VI.34), también haciendo uso del objeto byte (13), los bits siguen el orden siguiente:

el bit 1 es Cal Sol 1

el bit 2 es Cal Sol 2

el bit 4 es Air Sol 2

el bit 5 es Sample Pump

el bit 6 es Drain Pump

el bit 7 es Air Sol 1

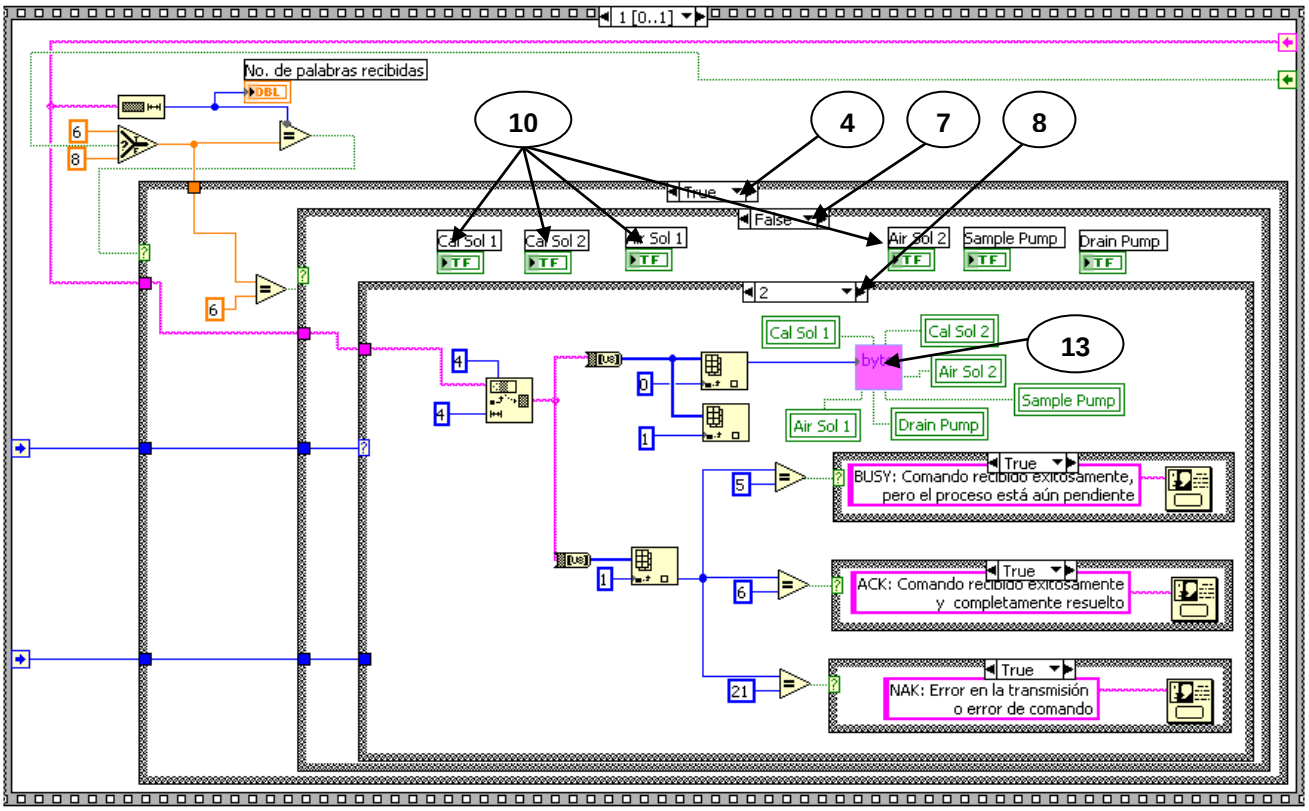


Figura VI.34. Instrumento virtual Read/Write I/O's.

Finalmente, el panel frontal para el instrumento virtual Lectura/Escritura de Entradas/Salidas es el mostrado en la figura VI.35.

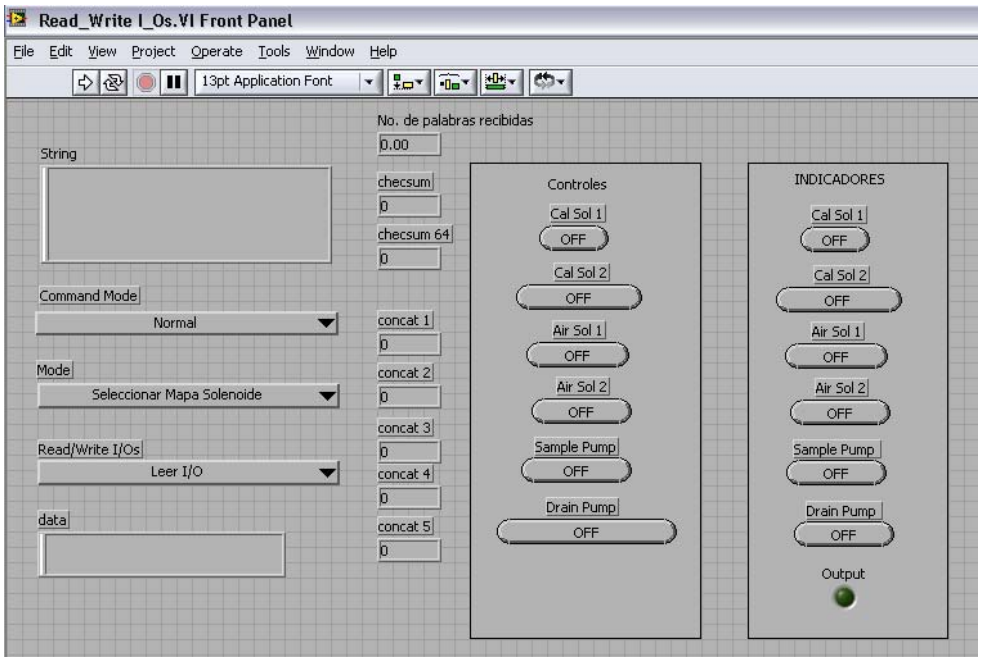


Figura VI.35. Panel frontal del instrumento Read/Write I/O's.

Una vez terminado el instrumento virtual, se editó su icono  correspondiente, para ser, posteriormente, llamado en el programa principal.

VI.1.4.5. Estado del proceso (*Process status*)

Este comando process status es por default un comando no muy largo. La cadena para hacer la solicitud de este comando es muy sencilla y se muestra a continuación.

Solicitud:

<11, *command_mode*><0><*chksum*>

En este caso el número de comando es 11, y el *data_count* vale cero; dado que no hay algún campo de datos, simplemente con el número del comando se sabe lo que se está solicitando.

La respuesta dada por la banca tiene la siguiente forma:

Respuesta:

<11, *response_mode*><2><*process_status*><*process*><*chksum*>

Como se puede observar, el *data_count* toma el valor de 2 ya que a continuación siguen dos campos de datos: *process_status* y *process*.

El campo *process_status* indica el estado de los comandos pendientes en la banca AMB y puede tomar los valores que se muestran en la tabla VI.35.

Process status	Descripción
0	No hay procesos pendientes
>0	Un proceso está pendiente (BUSY)

Tabla VI.35. Valores del campo process_status para el comando Process Status.

El campo *process* muestra el número de comando del proceso actualmente pendiente o el último proceso activo, sus valores se muestran en la tabla VI.36.

Process	Descripción
0	El último comando no fue muy largo
>0	Pendiente o último proceso pendiente (BUSY)

Tabla VI.36. Valores del campo process para el comando Process Status.

Process será puesto a '0' cuando el comando pendiente se haya completado y haya sido enviado en el modo muy largo, o cuando un nuevo comando sea emitido.

VI.1.4.5.1. Instrumento Virtual Estado del Proceso

Para la parte de la solicitud del comando se hará referencia a la figura VI.36. Como ya se vio, la cadena a enviar es muy sencilla:

<11, *command_mode*><0><*chksum*>

así que lo que se manda primero es un 11 (1), correspondiente al número de comando; posteriormente se envía el *command_mode* (2), después van dos ceros que toman del mismo punto, y estos corresponden al LSB y MSB del *data_count* (3), y por último se realiza el *checksum* (4), que en este caso es sólo la suma de dos palabras. Todo esto entra en un arreglo (5), que es convertido a cadena (6) y mandado por el puerto serial (7).

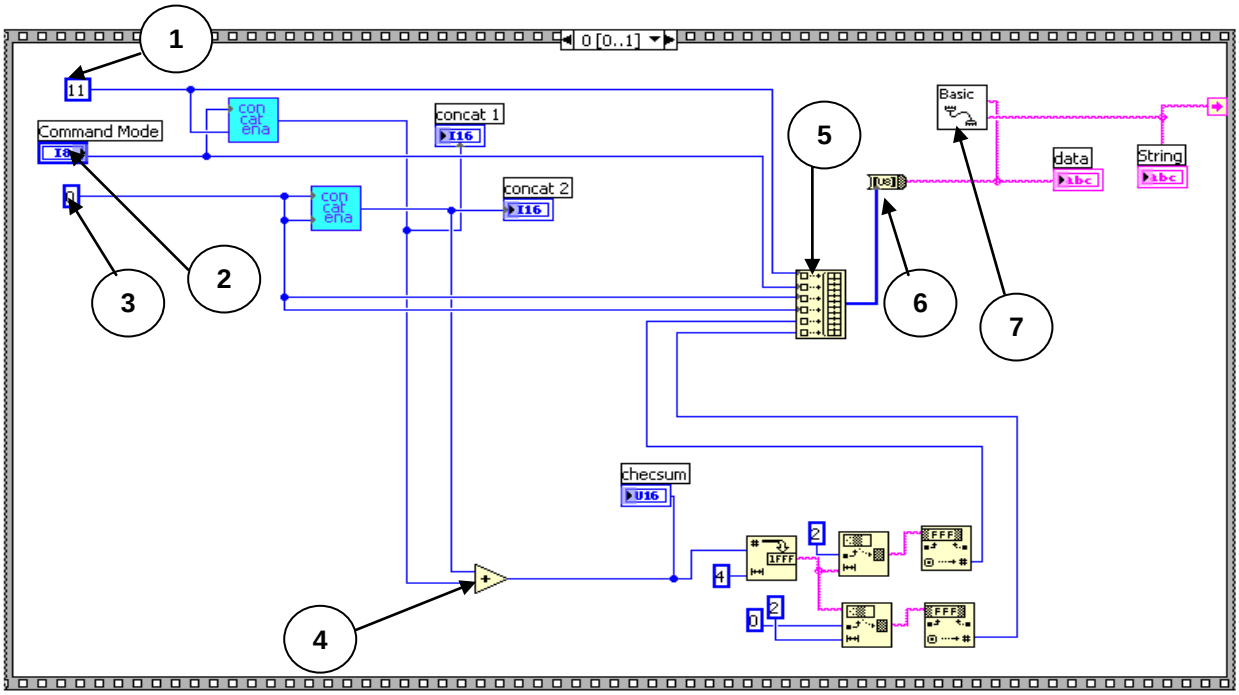


Figura VI.36. Instrumento virtual Process Status.

Para la decodificación se hace referencia a la figura VI.37. La cadena a decodificar es la siguiente:

<11, *response_mode*><2><*process_status*><*process*><*chksum*>

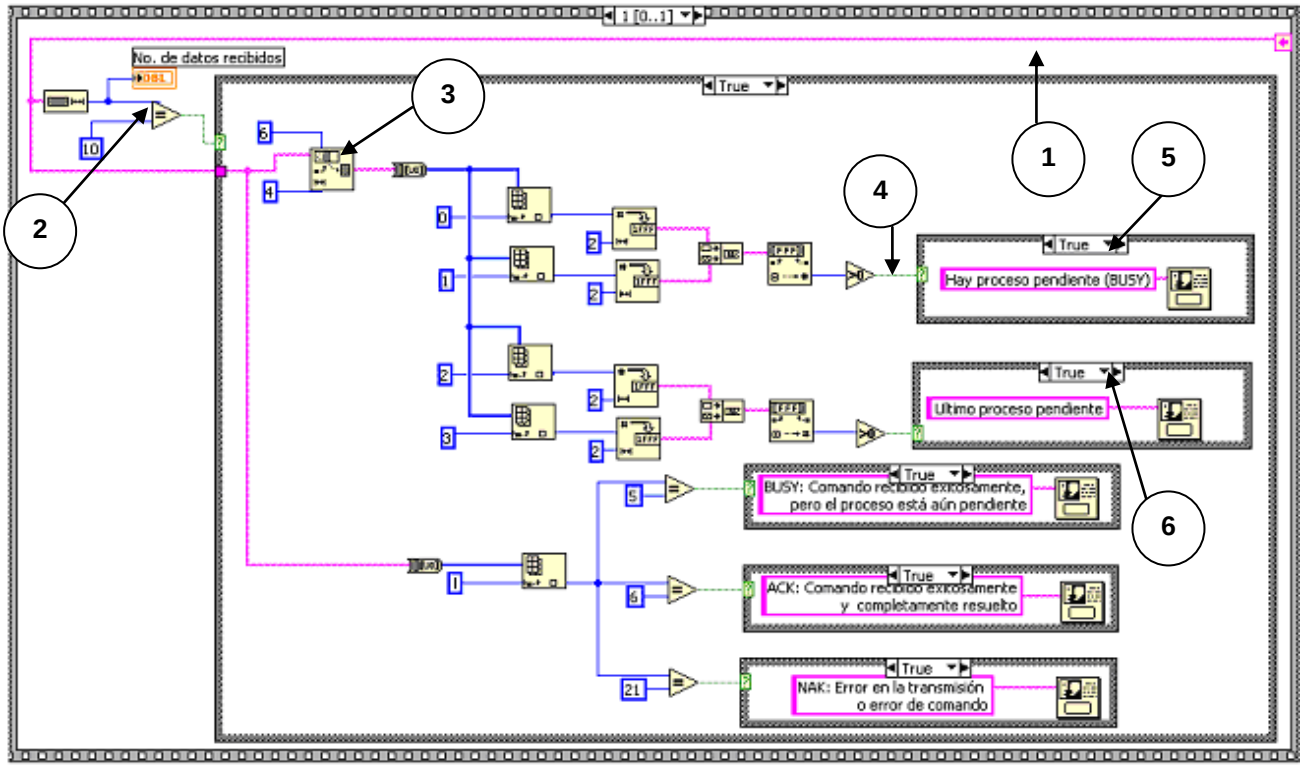


Figura VI.37. Instrumento virtual Process Status.

Una vez que se obtiene la cadena recibida del puerto serial (1), la longitud de ésta se compara con 10 (2), que es el número de bytes que se recibirían si la respuesta es correcta; la salida del comparador va a una estructura case, si la salida del comparador es *false*, no se realiza alguna acción; si fue *true*, entonces se le quitan los cuatro primeros bytes (3), quedando así los seis últimos, de los cuales los dos últimos corresponden al *checksum* y no se utilizan.

La primera palabra de la cadena que se obtiene en la acción anterior, es la correspondiente al *process_status*, que según su descripción indica que si es 0 entonces no hay procesos pendientes, y si es >0 hay un proceso pendiente; entonces el dato recibido se compara para saber si es >0 (4), y esta salida va a una estructura case que puede ser *false* o *true*. La salida del comparador va a esta estructura case, y dependiendo del valor de la salida, se le manda al usuario un mensaje (figura VI.38) para avisarle si hay o no procesos pendientes (5).

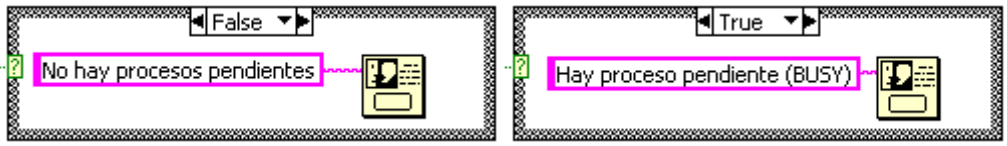


Figura VI.38. Case para el campo process_status.

La segunda palabra es del *process*, y éste tiene una descripción parecida a la del campo *process_status*, por lo que se hace algo similar (figura VI.39), pero ahora avisando al usuario si fue un comando no muy largo, o si está todavía pendiente el último proceso (6).

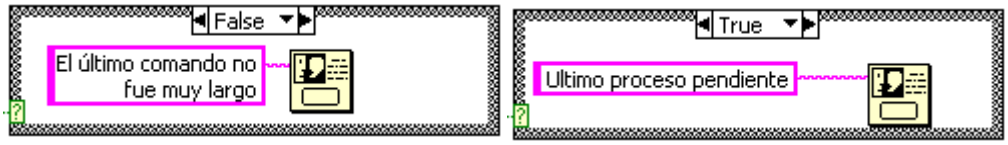


Figura VI.39. Case para el campo process.

Finalmente, el panel frontal para este comando se muestra en la figura VI.40.

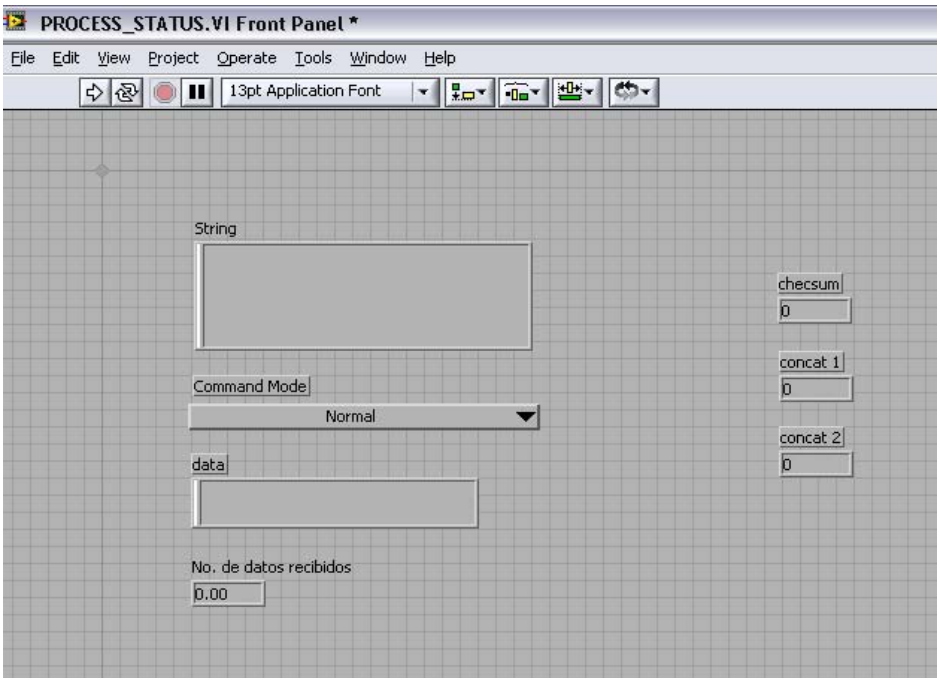


Figura VI.40. Panel Frontal del Instrumento Virtual Process Status.

Una vez terminado el instrumento virtual, se editó su icono 

correspondiente, para ser, posteriormente, llamado en el programa principal.

VI.1.4.6. Obtener versión (*get version*)

El comando *get version*, regresa información sobre la versión de la banca. El número de versión, tanto del *hardware* como del *software* es única de cada banca y siempre que se solicite este comando el número de versión será el mismo, para un mismo aparato.

Para hacer la solicitud de este comando se manda la siguiente cadena:

Solicitud:

<12, *command_mode*><2><*read/write*><*mode*><*chksum*>

Como hay dos campos de datos: *read/write* y *mode*, el *data_count* vale 2.

El campo *read/write* sólo puede tomar un valor (tabla VI.37).

Read/write	Descripción
0	Leer número de versión del hardware y software

Tabla VI.37. Valores del campo read/write para el comando get version.

De hecho, el campo pudo llamarse únicamente *read* y en la cadena que se envía en la solicitud también podía enviarse un cero directamente. No se puede escribir el número serial porque son datos que da el fabricante de la banca y, como ya se dijo, es único de cada banca.

El campo *mode* sirve para seleccionar el tipo del número de versión a ser leído, y puede tener dos valores posibles, que se muestran en la tabla VI.38.

Mode	Descripción
0	Obtener número de versión del software
1	Obtener número de versión del hardware

Tabla VI.38. Valores del campo mode para el comando get version.

Para el caso en cuestión, el número de versión del *software* para la banca AMB II es 20500 y el número de versión del *hardware* es 13824.

Para la respuesta se tiene:

Respuesta:

<12, *response_mode*><1><*version*><*chksum*>

En el campo *version* se da el número de versión solicitado, y a su vez también puede tomar dos valores posibles, mostrados en la tabla VI.39.

Mode	Descripción
0	Número de versión del software
1	Número de versión del hardware

Tabla VI.39. Valores del campo version para el comando get version.

VI.1.4.6.1. Instrumento virtual Obtener Versión

Para programar la solicitud de este comando, se hace referencia a la figura VI.41, y la cadena que se debe mandar es la siguiente:

<12, *command_mode*><2><*read/write*><*mode*><*chksum*>

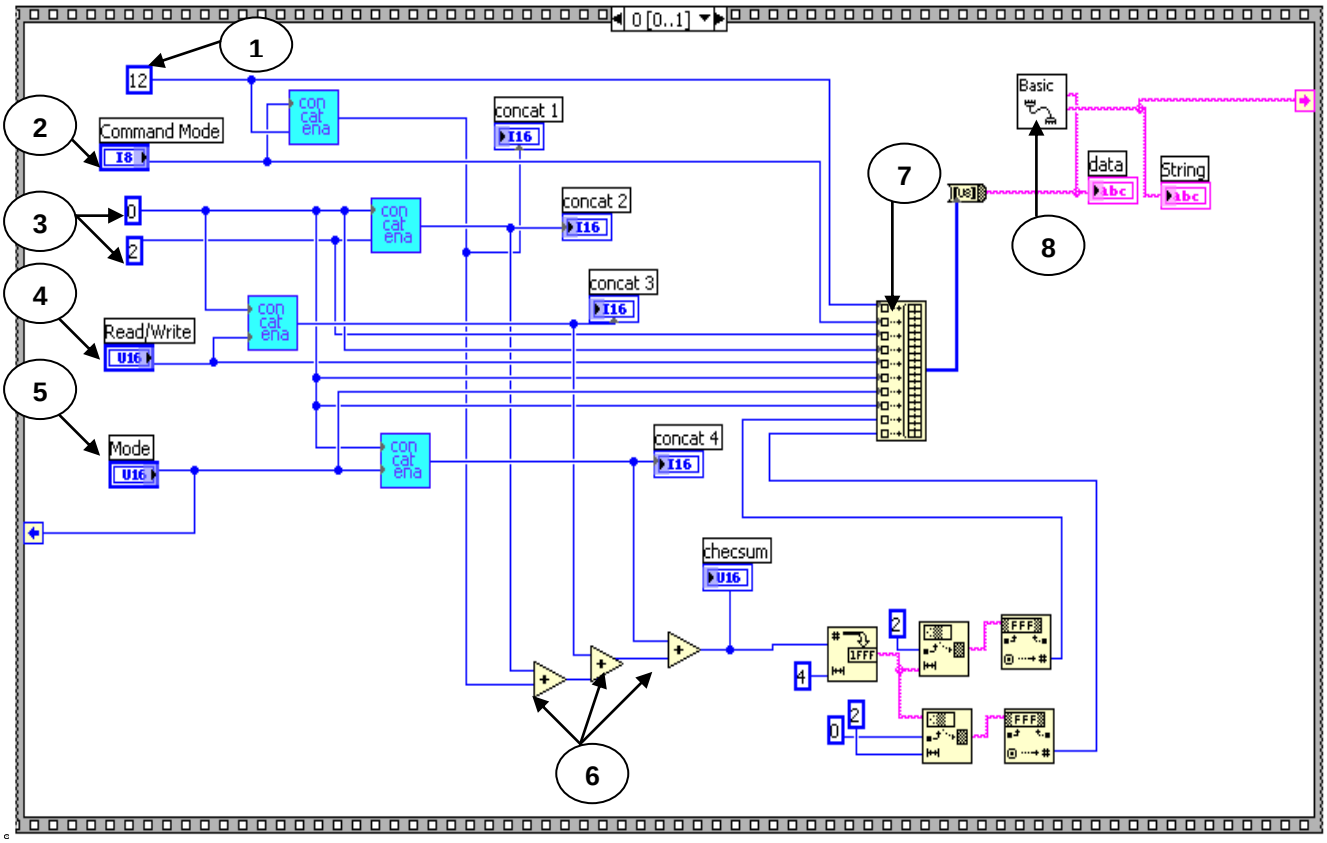


Figura VI.41. Instrumento virtual get version.

Primero se manda el 12 (1), después se manda el *command_mode* (2), seguidamente se mandan los bytes del *data_count*: un 0 y un 2 (3); posteriormente va el *read/write* (4) el cual es un control de una sola opción (figura VI.42), y después sigue el *mode* (5) que es un control de dos opciones (figura VI.43).

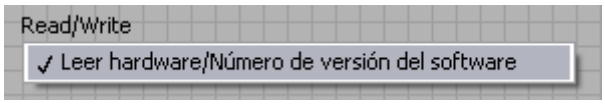


Figura VI.42. Control para seleccionar el read/write en el panel frontal.

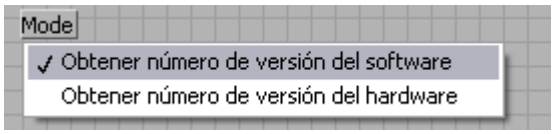


Figura VI.43. Control para seleccionar el mode en el panel frontal.

Ya por último se hace la suma para sacar el *checksum* (6) y juntar todo en un arreglo (7), la cadena resultante se manda vía puerto serial (8).

Para la decodificación de este comando se hará referencia a la figura VI.44. La cadena a decodificar es:

<12, *response_mode*><1><*version*><*chksum*>

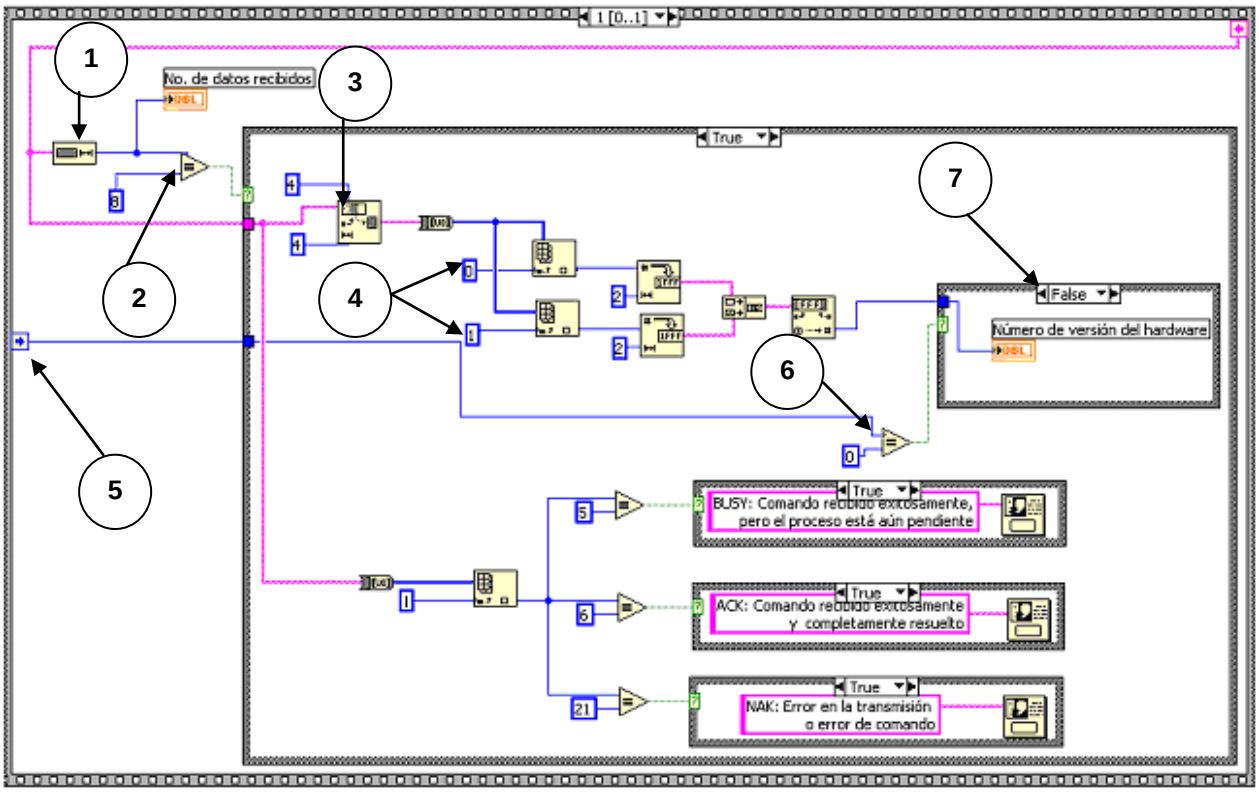


Figura VI.44. Instrumento virtual get version.

Para esto, la cadena leída del puerto serial se hace pasar por un *string length* para obtener su longitud (1), y ésta se compara con 8 (2), que son el número de bytes que se esperan recibir; si la salida del comparador es *true*, se procede a quitar los 4 primeros bytes correspondientes al *response_mode* y al *data_count*, y se dejan los 4 últimos (3) de los cuales, nuevamente sólo se toman los dos primeros (4) correspondientes al campo *version*.

Como se dijo anteriormente, *version* da el valor de la versión de la banca del *hardware* o del *software* según se le haya indicado en campo *mode*, es decir *version* depende de *mode*. Por esto, se toma el valor de *mode* con un *sequence local* (5) y se compara con cero (6), la salida del comparador determina la condición para una estructura case (figura VI.45), en la cual se va a mostrar lo que contiene el campo *version* en un *display* que da al panel frontal, ya sea el número de versión de *hardware* en caso de que sea false o el número de versión del *software* si fue *true* (7).

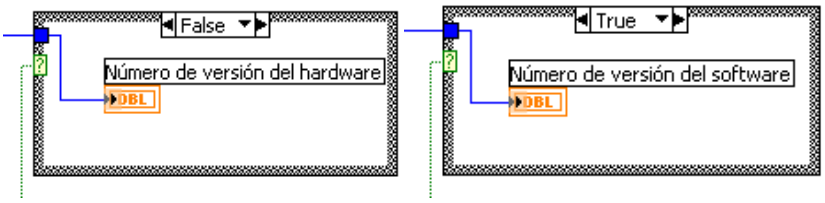


Figura VI.45. Case que muestra lo que contiene el campo versión.

El panel frontal para este instrumento virtual es el de la figura VI.46.

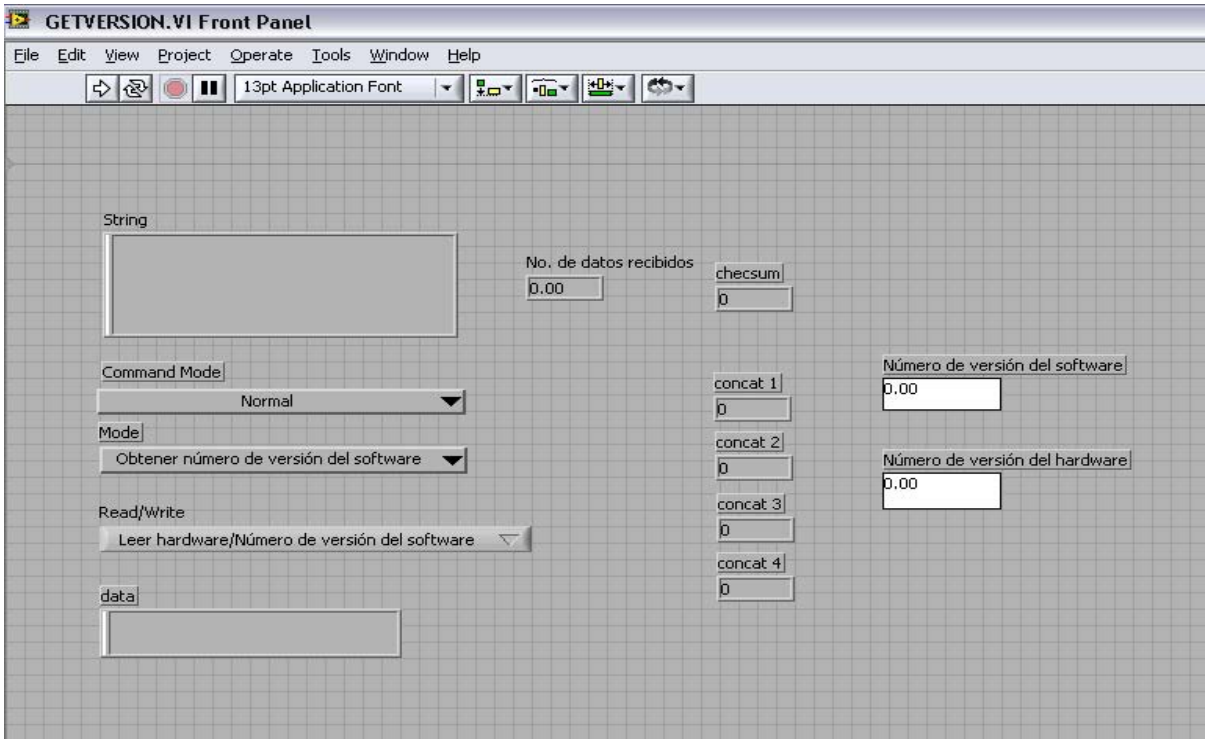


Figura VI.46. Panel frontal del instrumento virtual get version.

Una vez terminado el instrumento virtual, se editó su icono 

correspondiente, para, posteriormente, ser llamado en el programa principal.

VI.1.4.7. Número Serial (*Serial Number*)

El número serial, al igual que el número de versión, es único de cada banca y también ya está dado por el fabricante. Los 16 bits en el número serial contienen sólo la palabra baja del número serial. Internamente el número serial está almacenado en un formato de 32 bits. El número serial de 32 bits está construido por dos palabras de 16 bits.

$$SN32 = SerialNo32 * 2^{16} + SerialNo16$$

La cadena necesaria para la solicitud de este comando es la que se muestra a continuación:

Solicitud:

<13, *command_mode*><1><*read/write*><*chksum*>

El campo *read/write*, que nuevamente sólo podría llamarse *read*, sirve para seleccionar el modo de lectura para el número serial, y tiene dos posibles valores (tabla VI.40).

Read/write	Descripción
0	Leer número serial (16 bits)
2	Leer número serial (32 bits)

Tabla VI.40. Valores del campo read/write para el comando Serial Number.

La respuesta de este comando es como sigue:

Respuesta:

<13, *response_mode*><n>[…<*data*>…]<*chksum*>

Aquí se puede observar que *data_count* o *n* varía dependiendo del campo *data*. Éste a su vez depende del valor del campo *read/write* que se le haya dado.

Cuando *read/write* = 0 el dato regresado en el campo data es:

Data	Descripción
SerialNo16	Número serial (0..15)

Tabla VI.41 Valores de data cuando read/write= 0 para el comando Serial Number

Cuando *read/write* = 2 el dato regresado en el campo data es:

Data	Descripción
SerialNo16	Número serial (0..15)
SerialNo32	Número serial (16..31)

Tabla VI.42 Valores de data cuando read/write= 2 para el comando Serial Number

Para la banca AMB II, el número serial es 47306. Para *read/write* = 0, sólo aparece este número. Para *read/write* = 2, el SerialNo16 vale 47306 y el SerialNo32 vale 0.

VI.1.4.7.1. Instrumento Virtual Número Serial

Para explicación de la solicitud a enviar de este comando se referirá a la figura VI.47. La cadena que se va a mandar es:

<13, *command_mode*><1><*read/write*><*chksum*>

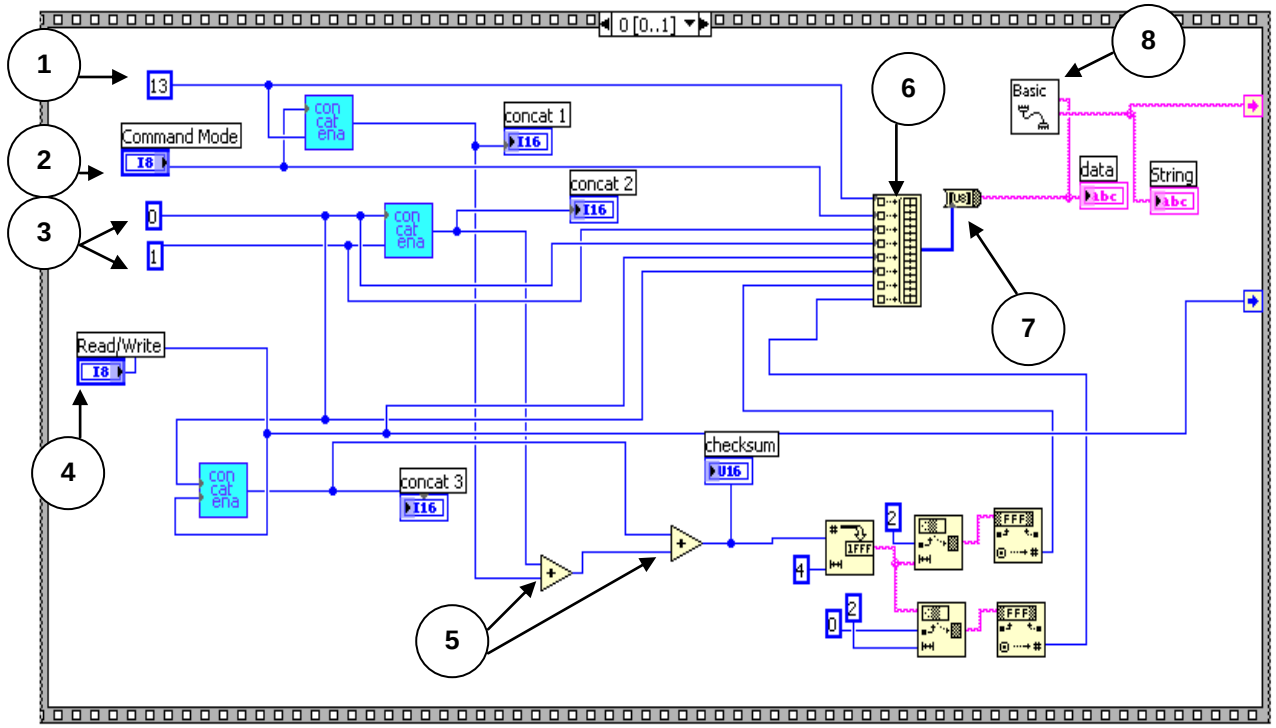


Figura VI.47. Instrumento virtual Serial Number.

Primero se expide el 13 (1), consecutivamente va el *command_mode* (2), después los dos bytes del *data_count*: 0 y 1 (3). Posteriormente sigue el *read/write* (4), éste es un control de dos opciones colocado en el panel frontal (figura VI.48).

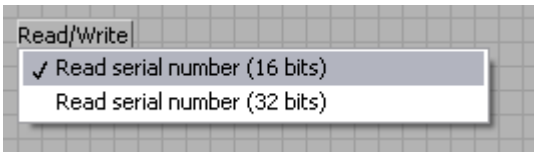


Figura VI.48. Control en el panel frontal para el campo read/write

Y por último se realiza la sumatoria de todas las palabras para el *checksum* (5). Juntando todas estas palabras en un arreglo (6) y convirtiéndolo en cadena (7) se manda por el puerto serie (8).

La decodificación es con respecto a la figura VI.49. La cadena a decodificar es la siguiente:

<13, *response_mode*><n>[…<*data*>…]<*chksum*>

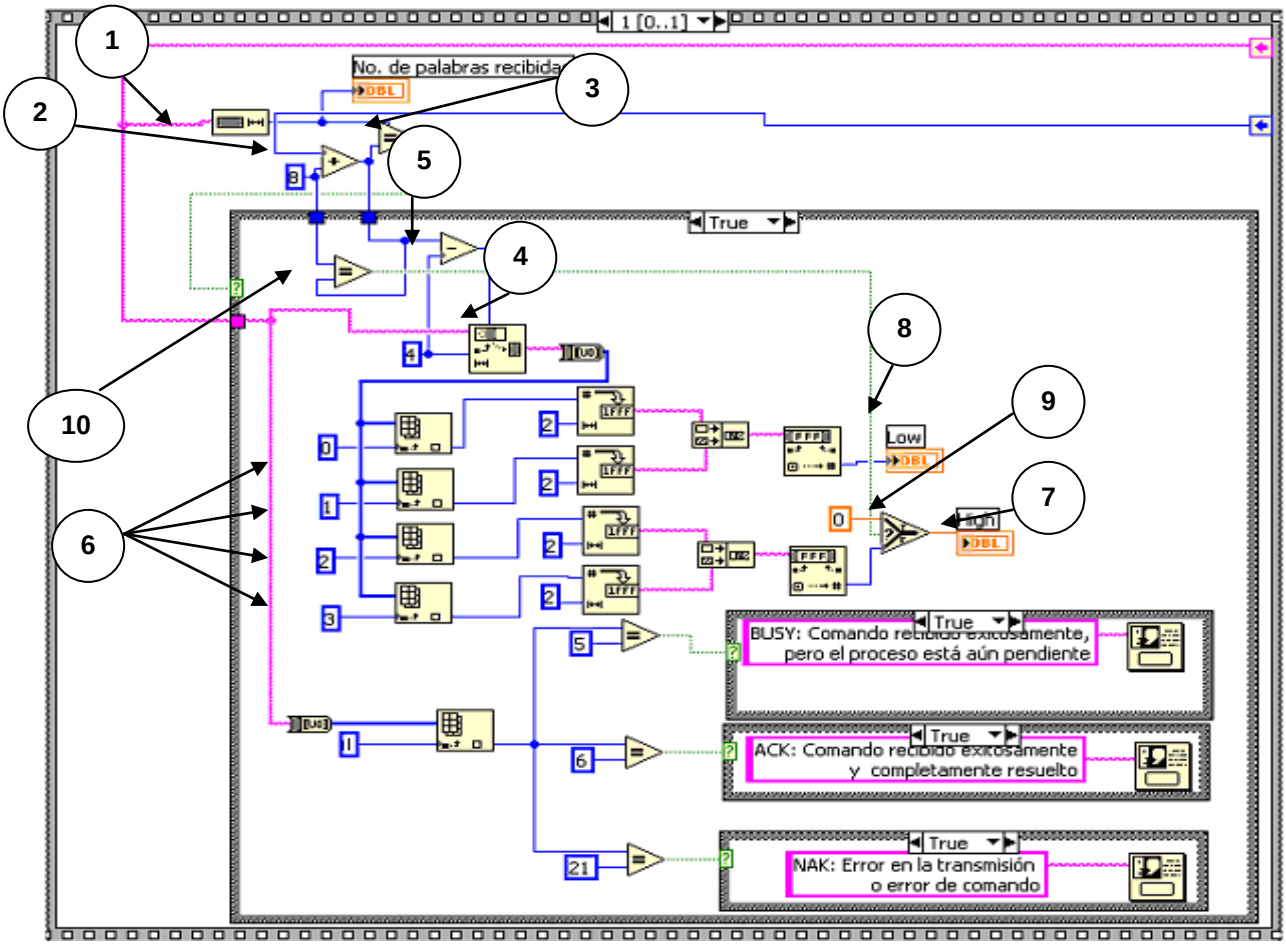


Figura VI.49. Instrumento virtual Serial Number.

Para empezar se obtiene la longitud (1) de la cadena recibida en el puerto serial, después, para la comparación tenemos dos opciones dependiendo del valor que haya al campo *read/write*. La primera es que si *read/write* = 0, entonces el campo *data* sólo contendrá una palabra, es decir, 2 bytes; por lo tanto, para este caso, la comparación sería con 8 (1 byte del 13 + 1 byte del *response_mode* + 2 bytes de n + 2 bytes de *data* + 2 bytes del *chksum*). La segunda es que *read/write* = 2 y para esto se tendrán dos palabras en el campo data, es decir, 4 bytes; por lo que, en este otro caso se tiene que hacer la comparación con 10 (1 byte del 13 + 1 byte del *response_mode* + 2 bytes de n + 4 bytes de *data* + 2 bytes del *chksum*).

Se puede observar en la figura VI.49, que al sumar el valor de *read/write* con 8 se tendrá el número de bytes con el que se desea comparar la longitud de la cadena, ya que si *read/write* = 0, entonces la suma 0 + 8 = 8 bytes esperados, y si *read/write* = 2, entonces la suma 2 + 8 = 10 bytes esperados. Con esto, en la programación se toma un sumador en donde las entradas son el *read/write* y 8 (2) y la salida de éste va al comparador junto con la longitud de la cadena (3). Si la salida del comparador es *false* no se procede a realizar alguna acción, si fue *true* entonces inicia la decodificación dentro de la estructura *case*.

A la cadena recibida se le quitan los 4 primeros bytes correspondientes al *response_mode* y al *data_count*, como esto se hace con el objeto predeterminado *String Subset* (4), como ya se ha visto, se le indica el *offset* (número de bytes que se le quitan) y el número de bytes de la cadena restante, para obtener éste último se toma la salida del sumador (2), que contiene el número correcto de bytes que tiene la cadena y le restan los 4 bytes que se le van a quitar (5).

Después se procede a obtener los cuatro primeros bytes de la subcadena (6); si *read/wrtite* = 0, entonces los dos primeros corresponden al data y los dos últimos son del *chksum* y la salida que va al *display* High (7) se obliga a que sea cero; y si *read/write* = 2, entonces los cuatro bytes corresponden al campo data. El *display* Low (8) siempre muestra el valor proporcionado por la banca. Para el *display* High, se utiliza el objeto predeterminado *Select* (9), el cual elegirá entre 0 (en caso de que *read/write* = 0) o el valor de la segunda palabra proporcionado por la banca (en caso de que *read/write* = 2), esta elección depende de un valor lógico, si éste es *true* se elegirá el 0, si es *false* se elegirá el contenido de la segunda palabra. El valor lógico se obtiene de una comparación (10) en donde una de las entradas es el 8 que entra en el sumador (2), y la otra es la salida del mismo sumador; si *read/write* = 0, entonces la salida del sumador es 8 y al ser comparada con el 8 que entra al sumador, el valor lógico dado por el comparador será *true* y se elegirá el 0. Si *read/write* = 2, entonces la salida del sumador es 10 y al ser comparada con el 8 que entra al sumador, como

sus valores son diferentes, el comparador arrojará un *false* y se escogerá entonces lo que está contenido en la segunda palabra de la subcadena.

Finalmente el panel frontal para el comando *Serial Number* es el mostrado en la figura VI.50.

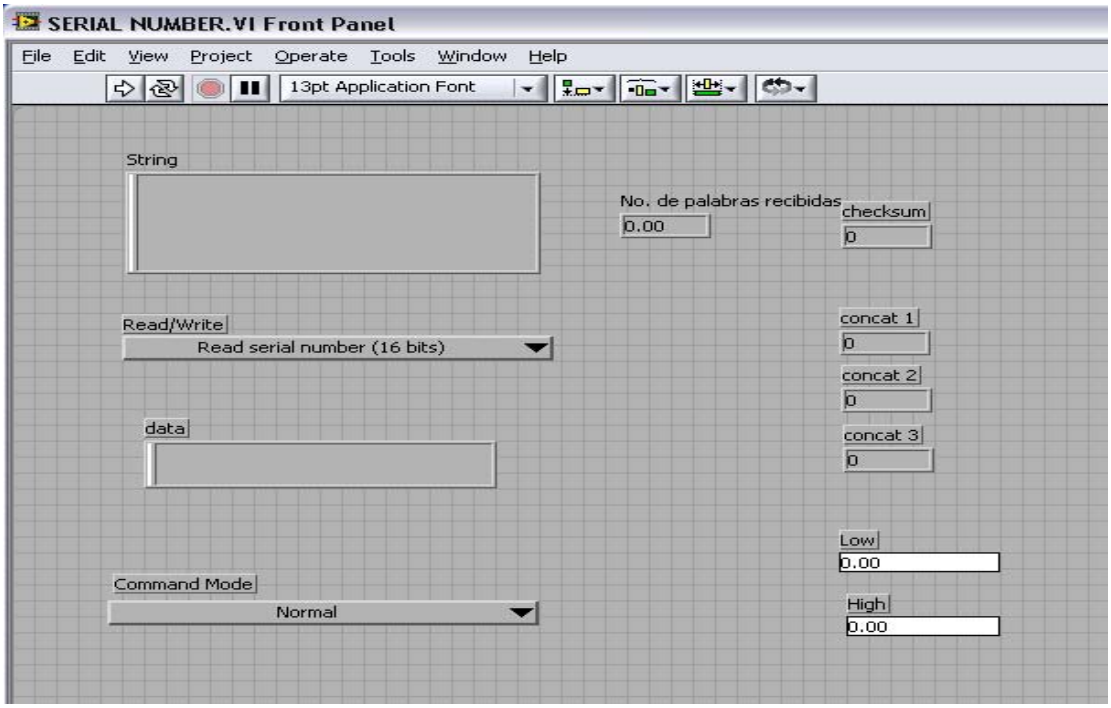


Figura VI.50. Panel Frontal del Instrumento Virtual Serial Number.

Una vez finalizada la programación del instrumento virtual se creó su icono correspondiente 

.

Al final, a todos los instrumentos virtuales se les agregó el botón de *stop* (figura VI.51) para que, al ser llamados en el programa principal se pueda detener su ejecución sin necesidad de cerrar el programa principal.

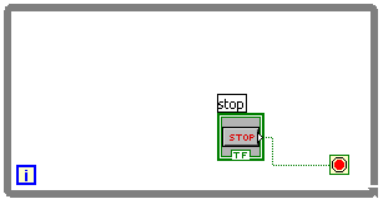


Figura VI.51. Botón de stop para los instrumentos virtuales.

VI.2. DESCRIPCIÓN DEL PROGRAMA PRINCIPAL

Una vez programados cada uno de los instrumentos virtuales se creó un programa principal, en el cual se hace uso de esos instrumentos virtuales, llamándolos como subVI's.

El programa principal muestra el panel frontal de la figura VI.52, con el cual, finalmente, el usuario podrá tener una interacción directa, y podrá fácilmente comunicarse con el sistema analizador de gases por medio de esta interfaz.

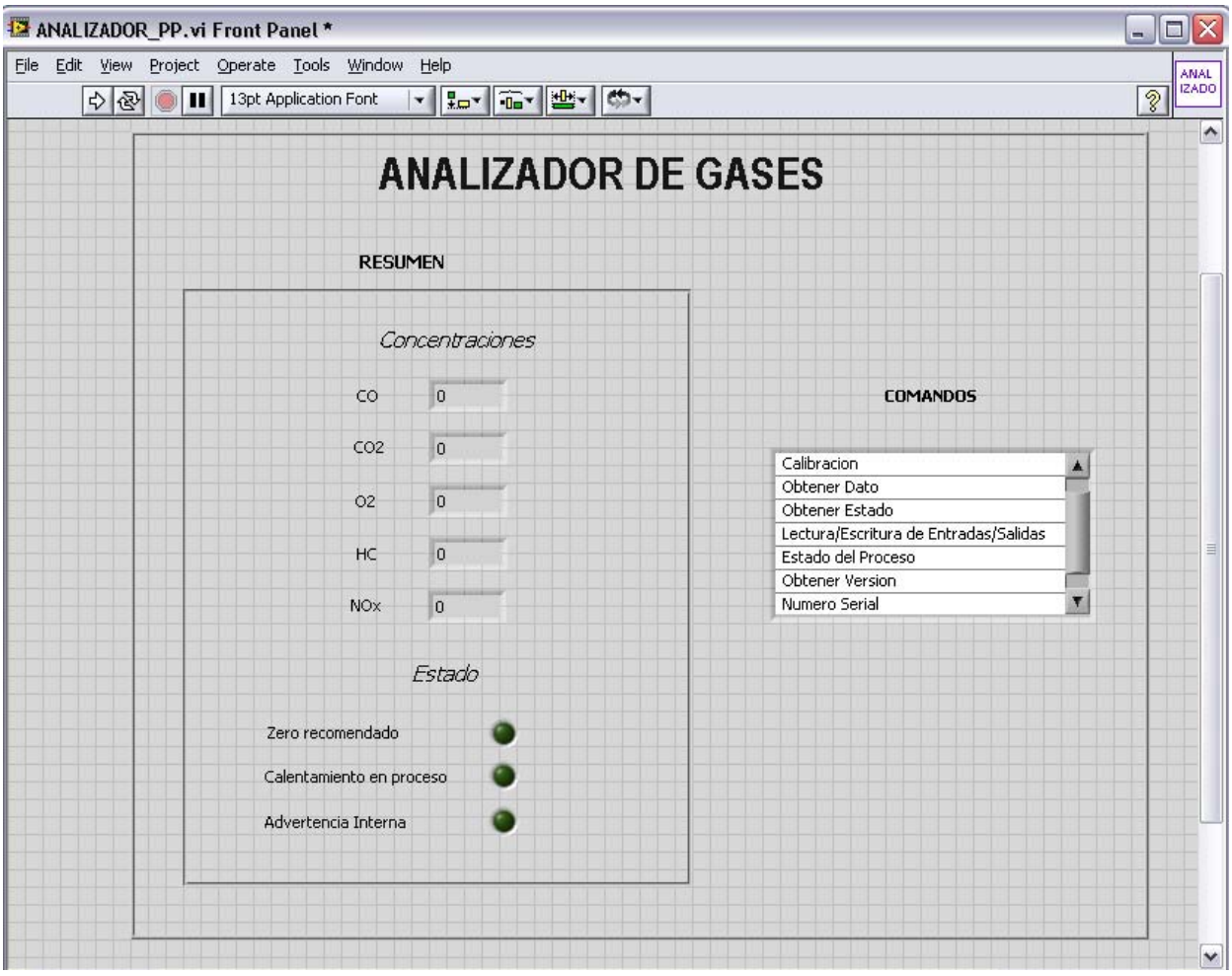


Figura VI.52. Panel Frontal del Programa Principal.

En el programa principal se indican los principales aspectos que en un cierto caso se puedan necesitar o que son de mayor relevancia para el usuario, como son las concentraciones de cada uno de los gases medidos por la banca, si se recomienda o no la realización del proceso de zeroing, si la banca está aún en calentamiento, o si existe la advertencia interna, es decir, un mal funcionamiento de la banca.

Cuando el usuario requiere de un mayor detalle de alguno de los comandos, o algún dato que no esté proporcionado en la pantalla principal, entonces se tiene un menú en el que el propio usuario puede seleccionar el comando deseado y automáticamente será corrido apareciendo así los paneles frontales, mostrados en cada uno de los comandos anteriormente descritos.

Ya por último, se creó el icono del programa principal 

llamado analizador, para que posteriormente, en el Laboratorio de Control de Emisiones se utilice como uno más de sus módulos para medición y verificación de emisiones de automóviles de combustión interna.

Una vez que ya se implementó el analizador de gases, en el siguiente capítulo se muestran las pruebas realizadas al sistema tanto individual como conjuntamente.

CAPÍTULO VII

INFORME DE PRUEBAS Y CONCLUSIONES

El presente capítulo da a conocer al lector las diferentes pruebas realizadas a cada uno de los módulos desarrollados, que forman parte del analizador de gases, así como las pruebas hechas al integrar todos estos módulos. El lector conocerá las conclusiones finales que se obtuvieron del desarrollo de la presente tesis.

VII.1. PRUEBAS REALIZADAS AL HARDWARE

Debido a que el analizador de gases está conformado por diferentes módulos, es decir, se diseñó de forma modular, conforme se fueron implementando los diferentes módulos, se tuvieron que realizar pruebas correspondientes a cada uno de ellos, para comprobar su correcto funcionamiento, así como también su interacción con los demás módulos. A continuación se describen las diferentes pruebas realizadas para algunos módulos.

VII.1.1. Prueba del módulo de desvío de gases

Este módulo, formado por la válvula solenoide, que es un dispositivo de aplicaciones neumáticas, fue probado con una fuente del LCE, comprobando su buen funcionamiento, es decir que, en efecto, sí seleccionara y cambiara el origen de la muestra, porque como se recordará son dos los casos en los que la válvula puede escoger el origen de la muestra: puede seleccionar una muestra que es producto de una combustión o una muestra proveniente del aire del medio ambiente.

Una prueba complementaria a este módulo fue la que se hizo al relevador RIM-ODC5 de NTE Electronics, aquí se corroboró que al suministrarle a dicho módulo una tensión entre 3.0 y 8 VDC, la salida que proporcionara fuera de hasta 60 VDC a 3[A]. Esta prueba fue necesaria ya que el relevador es el que va a proporcionarle al módulo final la tensión que la válvula solenoide necesita.

VII.1.2. Prueba del módulo de bombeo de gases muestra

La prueba al módulo de bombeo de gases muestra se realizó de la siguiente manera: el módulo está conformado por dos bombas neumáticas de aspiración, se probó que ambas bombas aspiraran correctamente, así como también la polaridad con la que éstas trabajan. Se necesitó para esta prueba una fuente que nos suministrara una tensión de 5 [V].

VII.1.3. Prueba del módulo de análisis de concentraciones de gases contaminantes

Para probar el funcionamiento de la banca analizadora de gases primeramente se revisó que arrancara al conectarla a una fuente con una tensión de 5 [V]. Esto se puede ver con el llamado “happy led”, explicado en el capítulo IV apartado IV.1.9.1. Especificaciones de

la banca analizadora de gases; se verificó que este led parpadeara a su frecuencia normal. La fuente fue previamente calibrada con un tornillo que trae integrado, verificando que la tensión fuera lo más exacta posible a los 5 [V] con un multímetro, dado que una tensión mayor provocaría que se quemara la banca.

Una vez revisada la banca analizadora se hizo uso del software desarrollado para comprobar que la banca estuviera enviando y recibiendo datos, es decir, en esta etapa se comprobó que la banca y la PC tuvieran una comunicación exitosa mediante un puerto serial asíncrono. La prueba que se realizó consistió en que desde una PC y usando la interfaz virtual se mandara una instrucción a la banca analizadora y ésta respondiera a la instrucción con una secuencia de bits. Lo anterior se realizó para cada uno de los comandos programados.

VII.1.4. Prueba del módulo de suministro eléctrico

Debido a que este módulo consta de dos fuentes de corriente directa de 5 y 12[V] a 1 [A], que alimentan diferentes dispositivos del analizador de gases, las pruebas fueron realizadas con la ayuda de algunos de estos dispositivos.

En cada una de las fuentes se probó que las tensiones que manejaba estuvieran dentro de los rangos necesarios para activar diferentes dispositivos. Por ejemplo, para una de ellas, que es para uso de nuestro analizador, se comprobó que la tensión suministrada fuera exacta de 5[V], y que, en efecto, alimentara correctamente a la banca analizadora, y que la tensión de 12[V] prendiera los dos ventiladores que tiene el sistema. Para la otra fuente se hizo una prueba similar, se observó que los 12 [V] que suministra fueran capaces de alimentar a las bombas y a la válvula solenoide al mismo tiempo.

Todos los demás módulos no requerían de una prueba previa, así que fueron probados junto con todo el sistema.

VII.2. PRUEBAS REALIZADAS AL SOFTWARE

VII.2.1. Prueba de la interfaz virtual

Como se puede observar en el capítulo VI, la programación se hizo también por módulos llamados instrumentos virtuales o comandos. Una vez terminados estos

instrumentos virtuales, se creó el instrumento virtual principal en el cual se mandan llamar todos los anteriores.

Una vez finalizada toda la programación de la interfaz virtual haciendo uso del entorno de programación LabVIEW 8.0 de National Instruments®, se llevaron a cabo las pruebas correspondientes para ver el buen funcionamiento de la misma.

Para poder examinar el funcionamiento de la interfaz virtual se hizo uso de la banca analizadora de gases. Primero se comprobó, como ya se mencionó anteriormente, que entre la banca analizadora y la PC existiera comunicación, es decir, que la banca respondiera a alguna solicitud hecha por el usuario desde la computadora y que la respuesta arrojada por la banca se viera reflejada en la pantalla de la PC. La comunicación también fue comprobada al ver que, por ejemplo, mientras la banca analizadora estuviera realizando su calentamiento interno, el software desarrollado reportaba que la banca se encontraba aún en proceso de calentamiento, y cuando la banca alcanzaba su temperatura de trabajo el software indicaba que había completado dicho calentamiento. Después, se verificó que la respuesta a cada solicitud dada fuera la correcta, aquí se fueron probando cada uno de los instrumentos virtuales que conforman la interfaz virtual para ver si funcionaban bien.

Esta prueba fue muy importante ya que si el software desarrollado no funciona bien, la solicitud a la banca será errónea, y por lo tanto la respuesta de la banca también, llevando a un análisis falso, o con errores. Así que con esto se verificó el correcto funcionamiento de la interfaz virtual.

VII.3. PRUEBAS GENERALES REALIZADAS AL SISTEMA COMPLETO

Las pruebas generales se realizaron al tener ya implementado todo el analizador de gases, con el fin de comprobar la correcta interacción y el funcionamiento entre todos los módulos que conforman al sistema, así como la buena comunicación entre el software desarrollado y el sistema analizador. Para poder tener una adecuada comprobación del sistema, la prueba general se dividió en dos partes:

En la primera prueba se probó el adecuado suministro de energía en todo el sistema, es decir, que al encender el analizador de gases, las fuentes suministraran las tensiones correspondientes a cada uno de los dispositivos que lo conforman, y que estos trabajaran en

conjunto. Se comprobó que la válvula solenoide, la banca analizadora, los ventiladores, y las bombas funcionaran correctamente.

La segunda prueba consistió en comprobar la comunicación entre la banca analizadora de gases y la PC mediante el puerto serial asíncrono. Haciendo uso de la interfaz virtual diseñada para el analizador de gases, se probó que al mandar una instrucción desde la PC a la banca analizadora, ésta la recibiera y dependiendo de la solicitud respondiera. Se probaron varios comandos para ver si la solicitud del software era correcta, y si la respuesta de la banca correspondía a la solicitud. Con esto, por ejemplo, se comprobó que la válvula solenoide estuviera siendo bien controlada desde la PC, es decir, que dependiendo de la solicitud enviada, la válvula solenoide seleccionara el origen de la muestra con que se tenga que trabajar.

A continuación se presentan algunas pantallas tomadas de cuando se ejecutan algunos de los instrumentos virtuales.

En la pantalla de la figura VII.1, por ejemplo se observa que el comando *get data* fue recibido exitosamente, además se tiene que la banca está aún en calentamiento y se pueden observar las mediciones en cada uno de los displays.

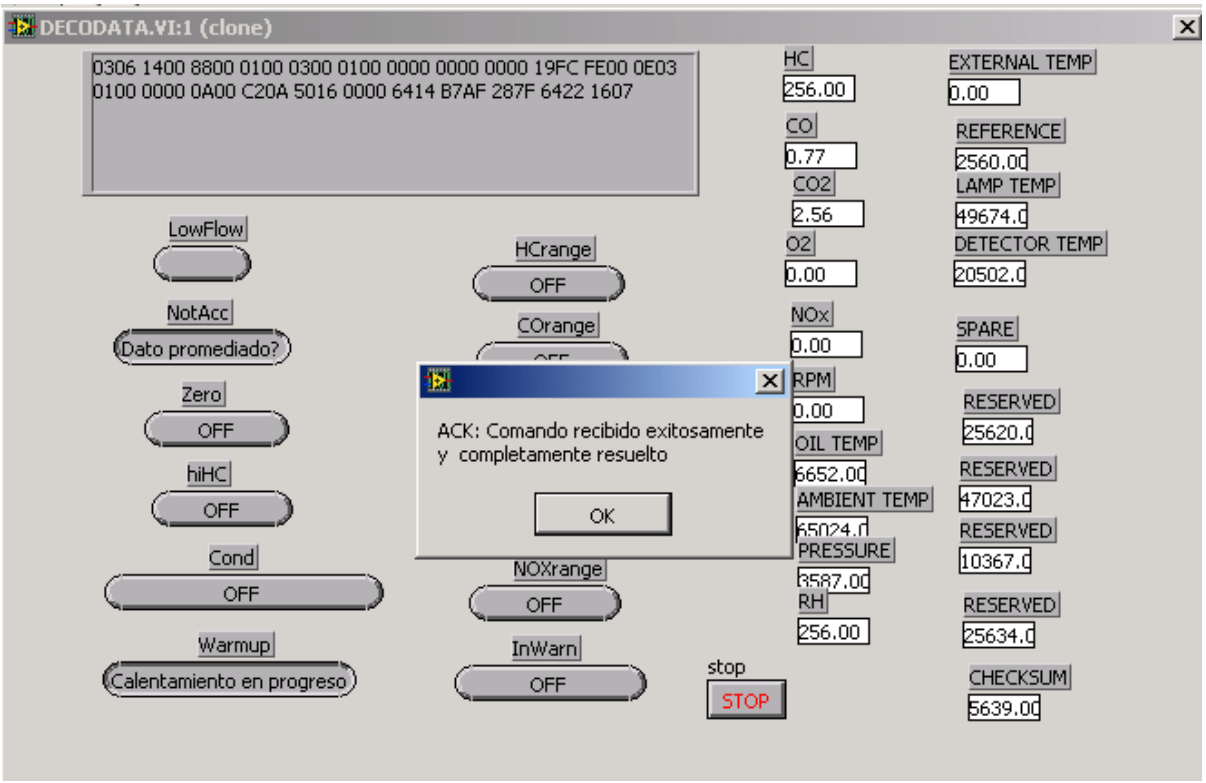


Figura VII.1. Instrumento virtual get data ejecutándose.

En la pantalla de la figura VII.2 se indica que se recomienda hacer un zero.

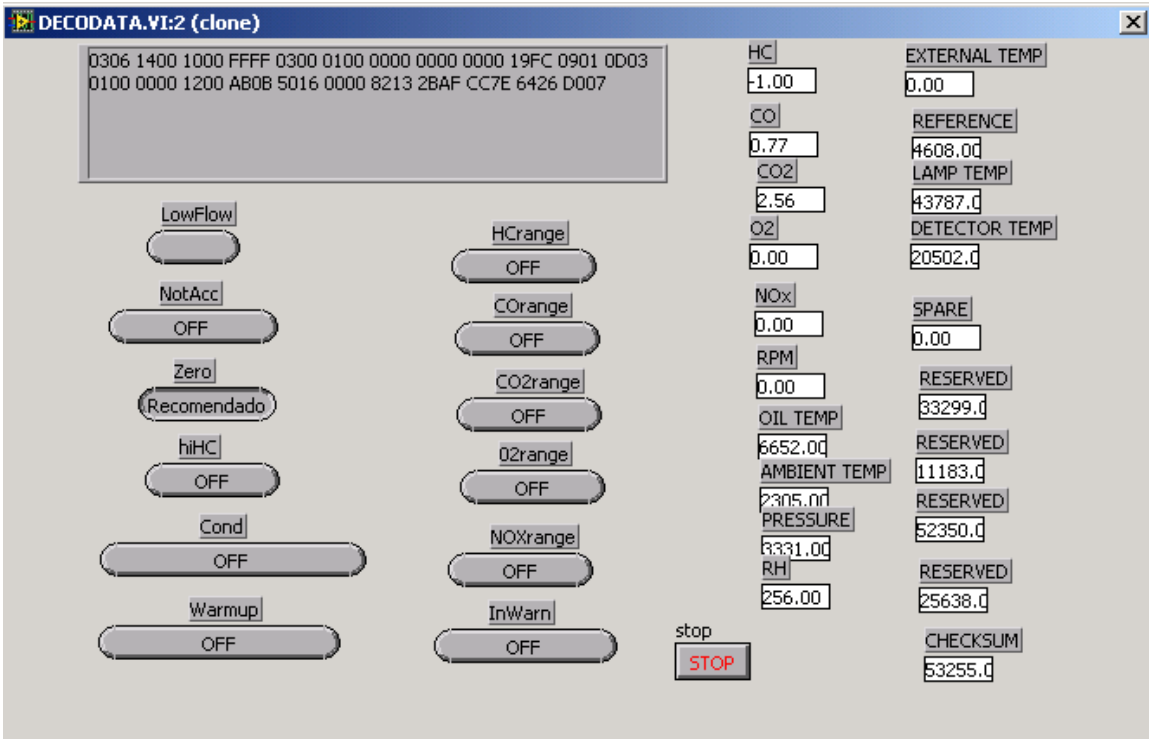


Figura VII.2. Instrumento virtual get data ejecutándose.

En la siguiente pantalla (figura VII.3), se muestra la ejecución del instrumento virtual *read/write I/O's* para el control de la válvula solenoide.

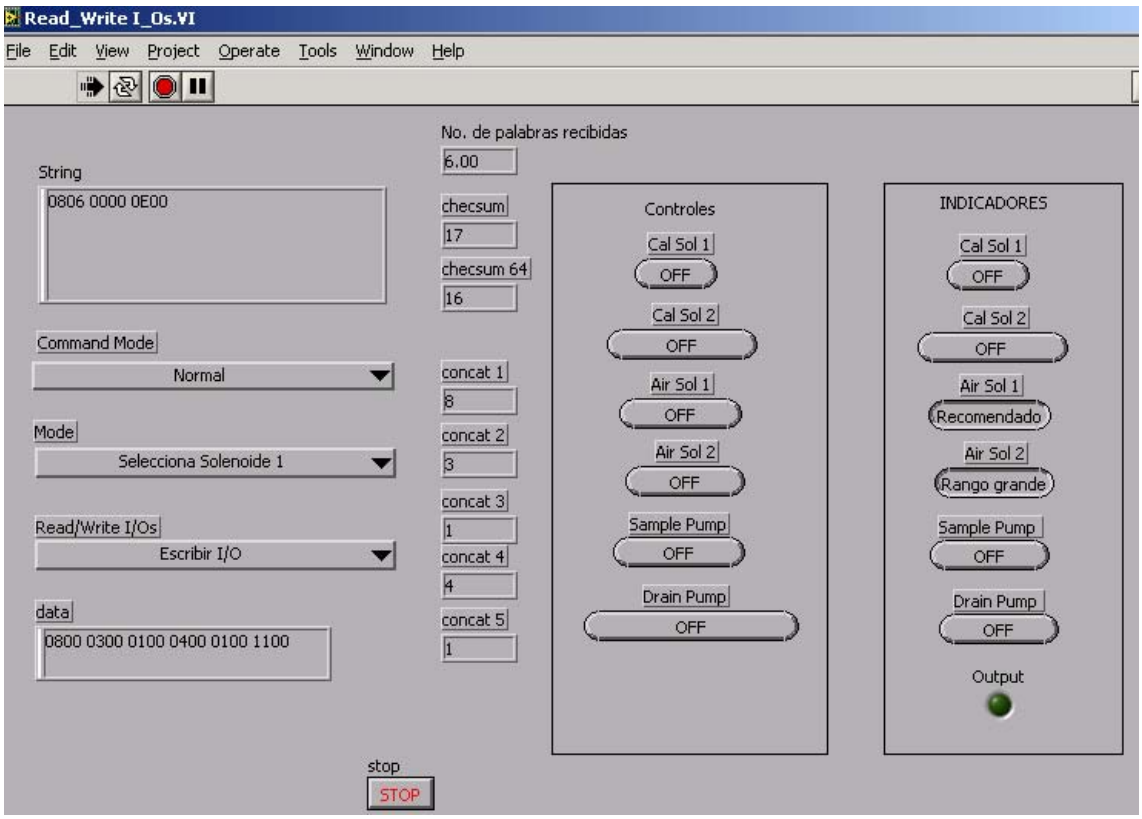
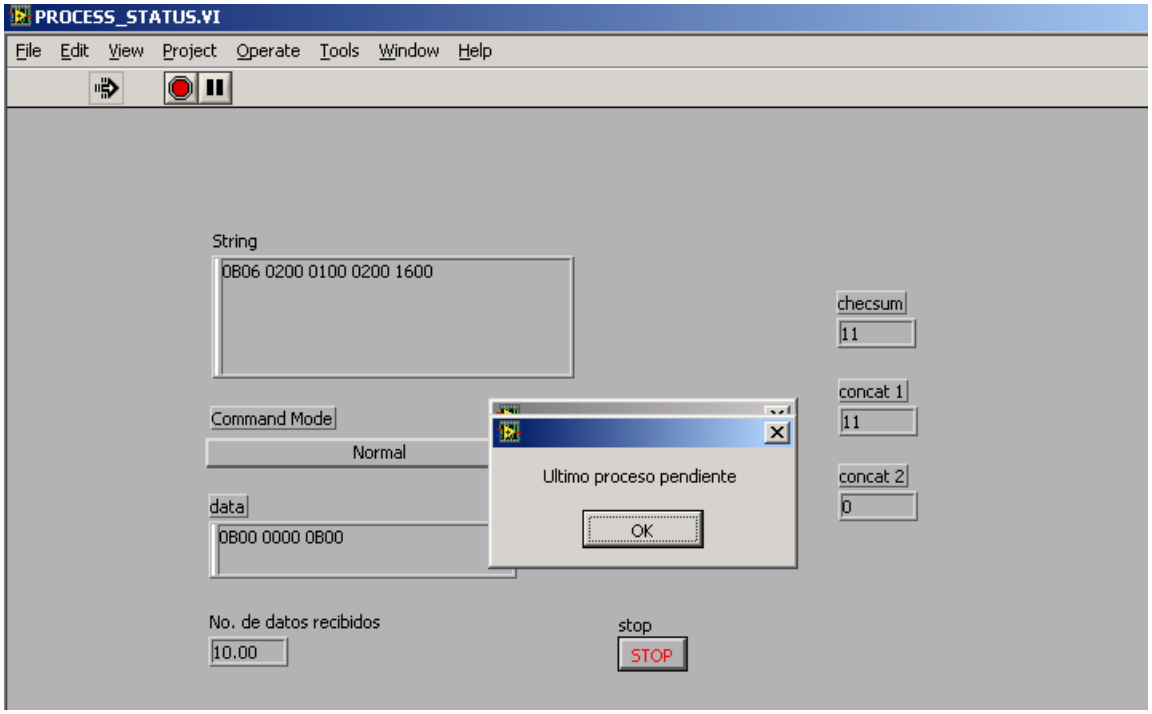
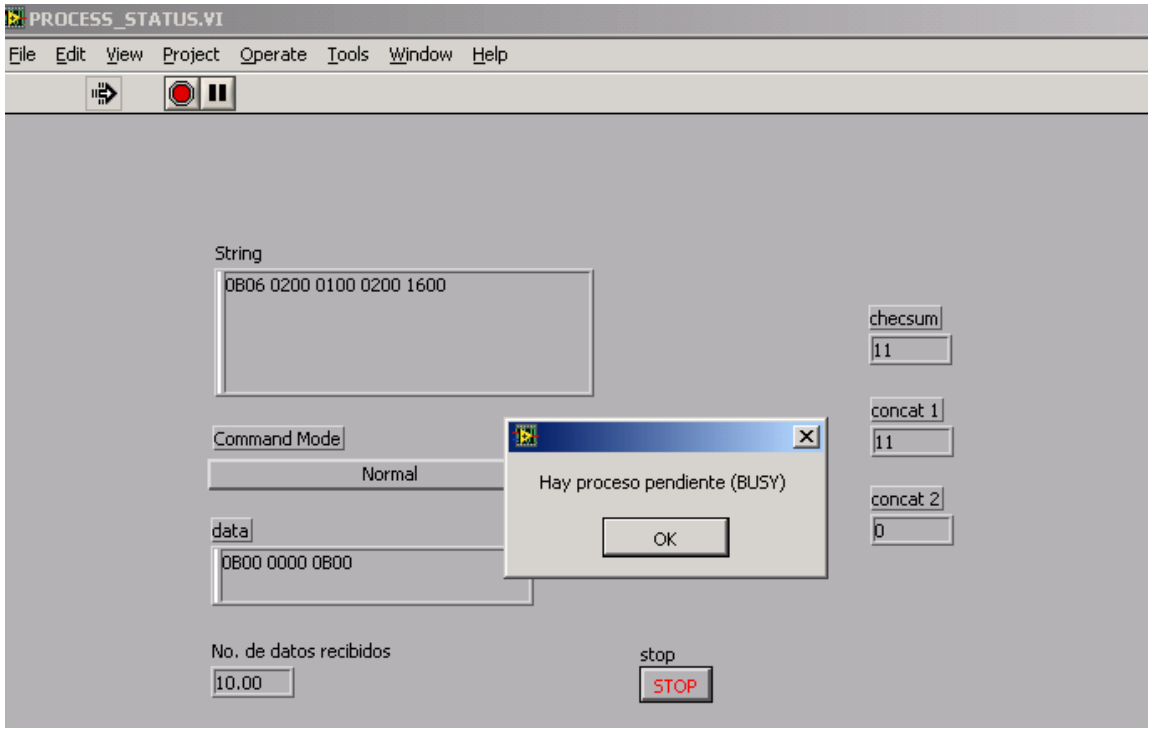


Figura VII.3. Instrumento virtual read/write I/O's ejecutándose.

En las pantallas de las figuras VII.4a y VII.4b, se tiene la ejecución del instrumento virtual *process status*.



(a)



(b)

Figura VII.4. Instrumento virtual process status ejecutándose.

En la pantalla de la figura VII.5, se observa el número de versión tanto del hardware como del software de la banca analizadora.

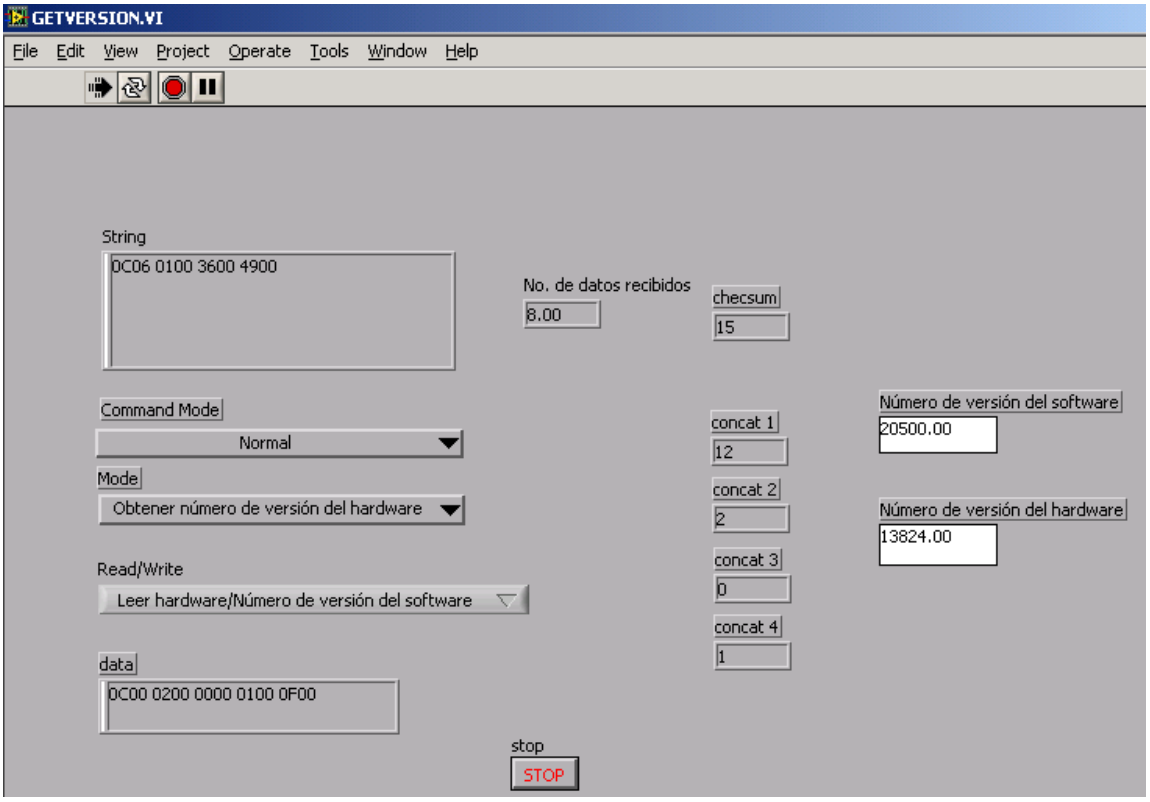


Figura VII.5. Instrumento virtual get version ejecutándose.

Por último, la pantalla de la figura VII.6 ilustra el número serial de la banca analizadora.

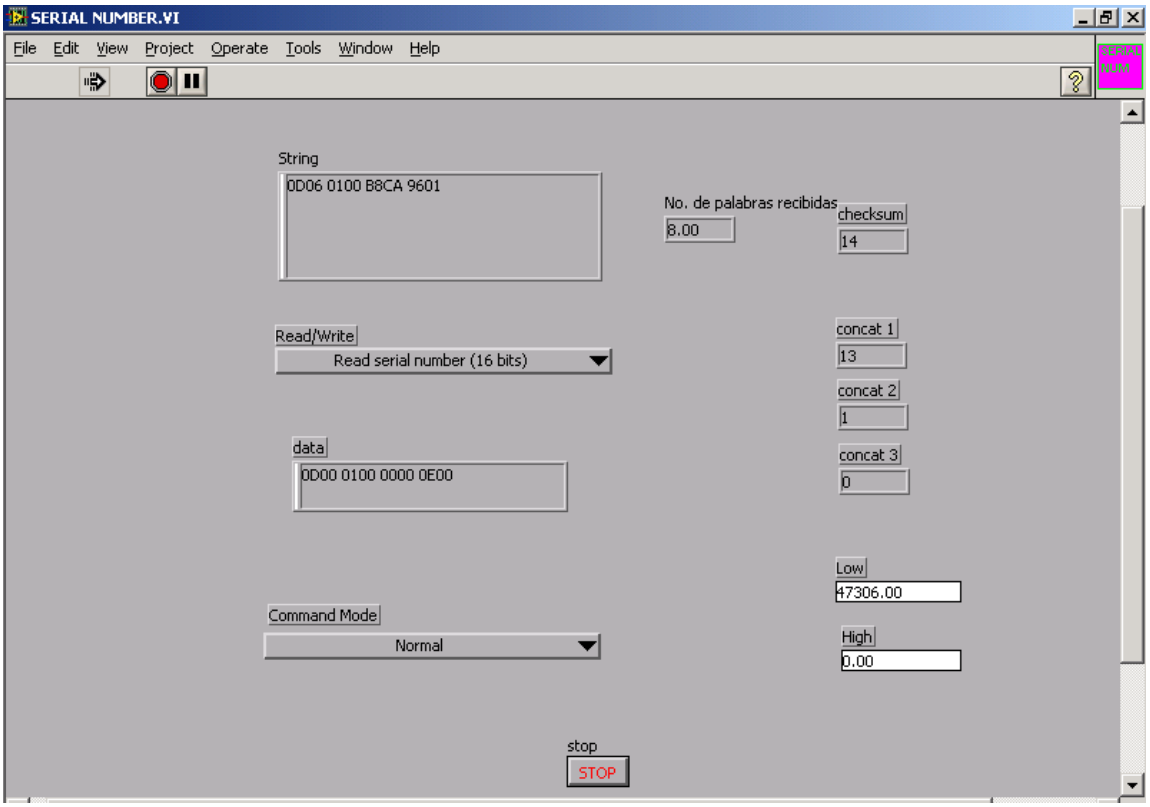


Figura VII.6. Instrumento virtual serial number ejecutándose.

VII.4. CONCLUSIONES

Se logró integrar el hardware necesario para el desarrollo de un analizador de gases, que pudiera llevar a cabo el análisis y monitoreo de gases contaminantes automotrices, tal es el caso del CO, CO_2, HC, NO_x y O_2, que son gases emitidos por motores de combustión interna a gasolina.

La interfaz virtual diseñada, que comunica al analizador de gases y la computadora, y controla a este sistema, cumplió con lo requerido que era desplegar la información obtenida haciendo uso de una computadora personal.

El uso del entorno de programación LabVIEW 8.0 de National Instruments® permitió que el proceso de instrumentación se facilitara considerablemente, porque además de ser gráfico y no a base de código, el programa desarrollado se simplifica de forma modular, permitiendo dividir un problema complejo en otros más sencillos.

Al diseñar e integrar este sistema y al crear el instrumento virtual analizador, se creó un elemento versátil el cual cubre con las necesidades requeridas por el LCE y puede ser utilizado conjuntamente con los demás sistemas empleados en este laboratorio, constituyendo así un módulo más.

Lo anterior permite que el instrumento virtual analizador pueda ser llamado en otro programa, es decir, ser usado para formar parte de un sistema más complejo.

Finalmente, gracias a la implementación y diseño de estos analizadores de gases se pueden reducir los niveles de contaminación en el medio ambiente, debido a que estas proporcionan datos importantes sobre las emisiones de gases automotrices que permiten detectar que tanto contamina un automóvil.

BIBLIOGRAFÍA

ARTÍCULOS

1. Isaac Schifter, Esteban López Salinas.
   Usos y Abusos de las Gasolinas.

2. Jeff Anderson.
   Tipos de dinamómetros y de unidades de absorción de potencia.
   Propiedad literaria de Dyne Sistemas Cía. LLC, 1997.

3. Santiago Cruz Lauro, Rincón Gómez Pedro Ignacio.
   Instrumento virtual para análisis de gases de combustión.
   Laboratorio de Control de Emisiones, Facultad de Ingeniería, UNAM.

DIRECCIONES DE INTERNET

4. Instituto Nacional de Estadística, Geografía e Informática (INEGI).
   http://www.inegi.gob.mx/inegi/fnivel.aspx?s=est&c=6159&e=&i=

5. Unión de Grupos Ambientalistas UGAM - Transporte y Vialidad
   http://www.union.org.mx/guia/actividadesyagravios/transporte.htm

6. Instituto Nacional de Ecología
   http://www.ine.gob.mx/ueajei/publicaciones/gacetas/275/nom41.html

LIBROS

7. Antonio Manuel Lázaro.
   LabVIEW Programación gráfica para el control de Instrumentación.
   Ed. Paraninfo / Thomson editores,
   1era edición, Madrid España 1997.

8. José Rafael Lajara Vizcaíno, José Pelegrí.
   LabVIEW Entorno gráfico de programación. LabVIEW 8.20 y versiones anteriores.
   Ed. Alfaomega/ Marcombo,
   1era edición, México 2007.

MANUALES

9. Manual Técnico de LabVIEW®.
Propiedad editorial de National Instruments ® USA 1990.

10. Manual Sensors, inc.
AMBII 9270-054,
Hardware Interface SPECIFICATION Version 1.00.

11. Manual Sensors, inc.
AMBII 9270-054 Firmware Version 5.116,
Software Interface SPECIFICATION Version 1.00.

TESIS

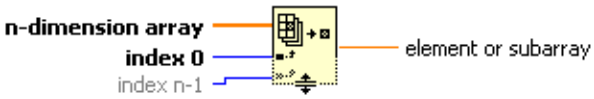
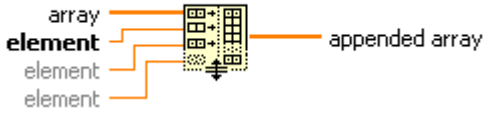
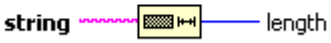
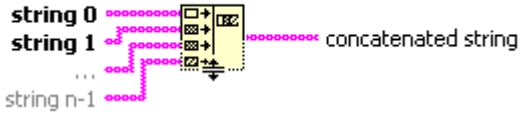
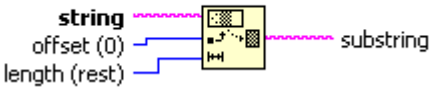
12. Rincón Gómez Pedro Ignacio.
Tesis Licenciatura "Instrumentación del dinamómetro de Chasis del Laboratorio de Control de Emisiones, Facultad de Ingeniería, UNAM".
Diciembre 1999.

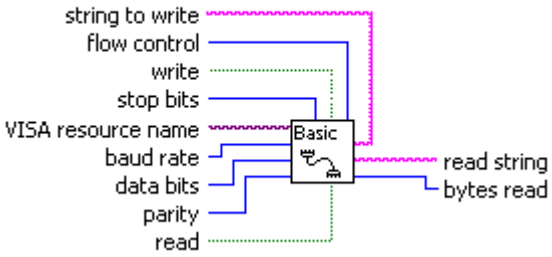
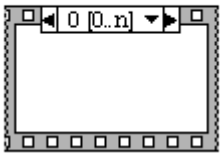
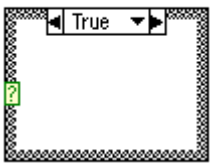
APÉNDICE A

INSTRUMENTOS VIRTUALES DE LABVIEW

A continuación se presenta una lista de los VI's y algunas funciones que se utilizaron en el diseño del programa, todos pertenecen a la biblioteca estándar de LabVIEW 8.0. En la mayoría de las aplicaciones únicamente es necesario hacer las conexiones de las entradas que se despliegan en negrita, las demás no son tan indispensables.

ICONO REPRESENTATIVO	DESCRIPCIÓN
Add **x** **y** x+y	Realiza la suma de las dos entradas.
Subtract **x** **y** x-y	Realiza la diferencia de las entradas.
Equal? **x** **y** x = y?	Regresa un valor TRUE si **x** es igual a **y**. En otro caso esta función regresa un valor FALSE.
Greater Than 0? **x** x > 0?	Regresa un valor TRUE si **x** es mayor que 0. En otro caso, esta función regresa un valor FALSE.
Select **t** **s** **f** s? t:f	Regresa el valor de la entrada **t** o **f**, dependiendo del valor de **s**. Si s es TRUE, esta función regresa el valor de la entrada **t**. Si **s** es FALSE, esta función regresa el valor de la entrada **f**.
Byte Array To String **unsigned byte array** string	Convierte un arreglo de bytes sin signo representando caracteres ASCII en una cadena.
String To Byte Array **string** unsigned byte array	Convierte una cadena en un arreglo de bytes sin signo.

Number To Boolean Array 	Convierte un entero a un arreglo booleano de 8, 16, o 32 elementos, dependiendo del número de bits en el entero. El elemento 0 del arreglo corresponde al bit menos significativo del complemento a 2 de la representación binaria del entero.
Boolean Array To Number 	Convierte un arreglo booleano a un entero sin signo de 32 bits interpretando el arreglo como el complemento a 2 de la representación binaria de un entero, con el primer elemento del arreglo comenzando con el bit menos significativo.
Index Array 	Regresa el elemento o subarreglo (**element or subarray**) del arreglo de dimensión n (**n-dimension array**) indicado en el índice (**index**).
Build Array 	Concatena múltiples arreglos o agrega elementos a un arreglo n-dimensional.
String Length 	Regresa en **length** el número de caracteres (bytes) en la cadena (**string**).
Concatenate Strings 	Concatena dos o más cadenas de caracteres en el orden en que se introduzcan. Como salida se tiene una cadena resultante de dichas uniones.
String Subset 	Regresa la subcadena (**substring**) de la cadena de entrada empezando en el **offset** y conteniendo **length** número de caracteres.
Hexadecimal String To Number 	Interpreta la cadena como un número hexadecimal y regresa un número.
Number To Hexadecimal String 	Convierte un número a cadena de dígitos hexadecimales.

One Button Dialog 	Despliega una caja de diálogo que contiene un mensaje.
Basic Serial Write and Read.vi 	Lleva a cabo una lectura del puerto serial, una escritura del puerto serial, o una combinación de estas acciones. Si se realizan las dos acciones primero se escribe el dato, y después se lee lo que se haya recibido. Este VI esperará hasta el número de bytes especificado que recibirá el puerto. Sólo el número de bytes especificados serán leídos.
Stacked Sequence Structure **[Stacked Sequence]** 	Consiste de uno o más diagramas, o frames, que ejecutará secuencialmente.
Case Structure 	Tiene uno o más subdiagramas, o casos, exactamente uno de los cuales ejecuta cuando la estructura se ejecuta. El valor conectado en el selector terminal determina cual caso va a ejecutarse y puede ser booleano, cadena, entero.
While Loop 	Repite el subdiagrama que tiene dentro hasta la condición terminal, una entrada terminal de valor booleano determina cuando se detiene el ciclo.

APENDICE B
MEDICIÓN DE LAS EMISIONES PRODUCIDAS POR VEHÍCULOS AUTOMOTORES

Con el fin de evaluar en forma cuantitativa las emisiones producidas por los vehículos automotores, se han desarrollado procedimientos que tratan de reproducir las condiciones reales de operación en el laboratorio.

Las emisiones gaseosas reglamentadas en los automotores son: hidrocarburos no quemados, monóxido de carbono y óxidos de nitrógeno. La prueba DGN-AA-II-1980 es un procedimiento federal de prueba empleado para certificar los automóviles nuevos a partir de los modelos de 1975. Así, los fabricantes de automóviles deben entregar al Gobierno Federal una prueba certificada de la cantidad de contaminantes que emiten sus autos. Esto no quiere decir que todos los automóviles nuevos pasan la prueba, sino que cada año deben hacerlo sobre todo si se trata de modelos nuevos o modificaciones a los existentes.

La prueba proporciona la caracterización más representativa disponible de emisiones de escape y economía urbana de combustible. Se realiza en una celda de ambiente controlado donde la temperatura y otras condiciones pueden mantenerse dentro de límites específicos.

Durante este proceso el vehículo se conduce en un dinamómetro de chasis con un programa de manejo de paro y marcha a una velocidad de aproximadamente 35 [km/h]. Mediante el uso de volantes de inercia y un freno de agua, se reproducen las cargas que el vehículo experimentaría en el camino. Los gases de escape del vehículo se recolectan, diluyen y se mezclan completamente con el aire filtrado circundante a un flujo de volumen constante conocido.

Las emisiones recolectadas incluyen un arranque en frío del motor y uno en caliente una vez que el auto ha recorrido 12 [km] y descansado 10 minutos. El dinamómetro de chasis reproduce la inercia del vehículo con volantes, y la carga del camino con un freno de agua. Para cada clase de peso de inercia se especifica una carga de camino que toma en consideración la resistencia aerodinámica promedio del vehículo. Un día antes del arranque en frío programado, el vehículo debe permanecer en reposo cuando menos 12 horas a temperatura entre 20 y 30°C.

En el momento de la prueba el auto se empuja sobre el dinamómetro sin arrancar el motor y se conecta el sistema de correlación de emisiones al tubo de escape, un ventilador de enfriamiento funciona de acuerdo con el motor abierto.

El sistema de muestreo de emisiones y el vehículo de prueba arrancan simultáneamente, de modo que las emisiones se recolecten durante el arranque del motor, el conductor sigue un programa de manejo controlado, el Programa Urbano de Manejo en Dinamómetro, creado para representar el manejo urbano en promedio.

El ciclo de manejo dura 1 374 segundos y cubre una distancia de 12 [km]. Las emisiones de escape que se miden cubren tres regímenes de operación del motor.

Las de escape, en los primeros 505 segundos de la prueba, son las emisiones transitorias frías, cuando el vehículo se calienta gradualmente a medida que se maneja en el ciclo.

Las emisiones mostrarán entonces los efectos de operación de arranque en frío y las características del calentamiento del vehículo. Cuando pasan los 869 segundos restantes del ciclo, se considera que ya se ha "calentado" el auto y las emisiones son las llamadas "estabilizadas".

El régimen final de la prueba tras la saturación en caliente es la sección "no transitoria" y muestra los efectos del arranque en caliente. Las emisiones de cada una de las pruebas se recogen en bolsas separadas, para analizar cuantitativamente su composición.

Finalmente se cuenta la masa emitida de cada contaminante en gramos por kilómetro recorrido. La economía del combustible se mide en un dinamómetro de chasis que reproduce las velocidades y cargas típicas del manejo urbano y en carretera. La economía de combustible se calcula a partir de los datos de las emisiones de descarga, en el caso de carretera la velocidad promedio es de 78 [km/h].

En la Figura B.1 se muestra un esquema general del equipo de prueba de emisiones en automóviles.

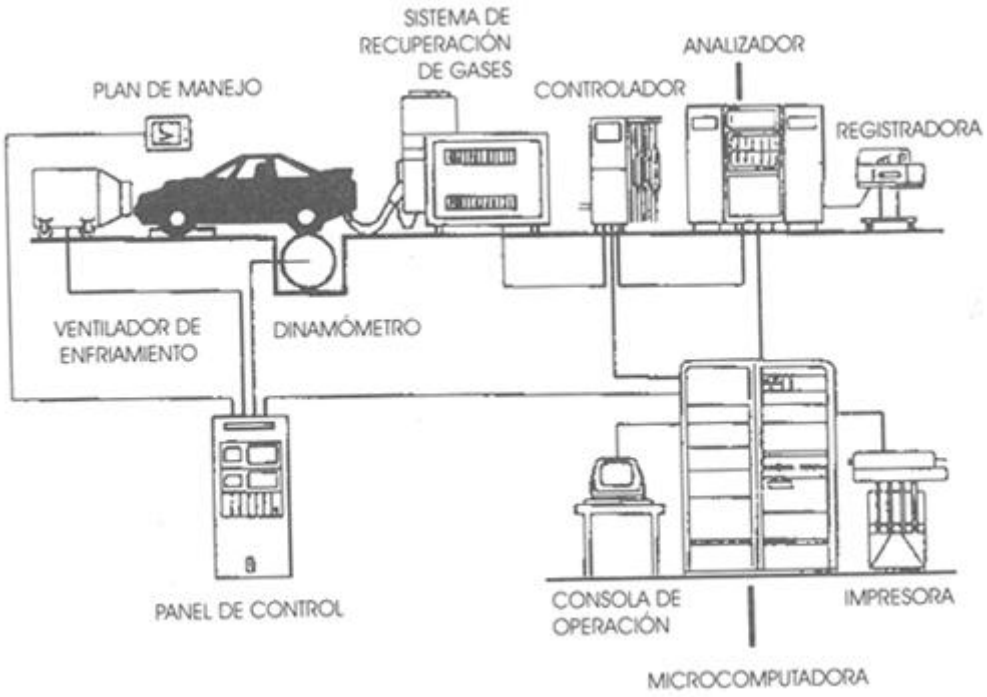


Figura B.1 Esquema de un sistema de evaluación de las emisiones de automóviles.