



Universidad de Valladolid



**ESCUELA DE INGENIERÍAS
INDUSTRIALES**

UNIVERSIDAD DE VALLADOLID
ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Mecánica

Programa en Labview para el diagnóstico de la combustión en banco de ensayo.

Autor:

Mayado Rubio, Francisco Javier

Tutor:

Dr. Andrés Melgar Bachiller
Departamento de Ingeniería
Energética y Fluidomecánica

Valladolid, julio de 2017

Resumen

Debido a la gran importancia del motor de combustión interna alternativo en la actualidad y a la necesidad de la optimización de los procesos de análisis y control de ciertas variables para su estudio, trabajos como el que se va a presentar, ayudan a la modernización y estandarización de estos procesos.

Con la ayuda de elementos simples como un ordenador, un osciloscopio, un cable USB, y la programación gráfica de Labview se plantea un análisis de la presión en la cámara de combustión del motor en función de su ángulo de giro, y procesar esos datos en tiempo real para trabajar con ellos en el ordenador. Esto permitirá almacenar y tratar esos datos de muchas formas posibles y calcular otras variables de importancia del MCI que se calculan a partir de la presión de la cámara.

Palabras clave

LabView, MCI, diagnóstico de presión, Instrumentación, Instrumentación virtual.

Nomenclatura

MCIA: motores de combustión interna alternativos

LV: LabVIEW

P: presión

FQL: fracción de calor liberado

DFQL: derivada de FQL.

FQM: fracción de masa quemada.

DFQM: derivada de FQM.

α : ángulo del cigüeñal a lo largo del ciclo.

MEP: motores de encendido provocado

MEC: motores de encendido por compresión

T_{ca} : temperatura al cierre de la admisión.

P_{ca} : presión al cierre de la admisión

L/r : relación de longitud entre la longitud de la biela y el radio de la manivela.

r_c : relación de compresión del motor.

ϑ : volumen específico.

VI: virtual instrument

PC: ordenador personal

PMS: punto muerto superior

Índice

Resumen	3
Palabras clave.....	3
Nomenclatura	4
1. INTRODUCCIÓN	7
1.1. Justificación.....	7
1.2. Entorno y antecedentes	7
1.3. Objetivos.....	8
1.4. Estructura del proyecto.	9
2. FUNDAMENTOS TEÓRICOS APLICADOS	11
2.1. Medida de presión en motores.....	11
2.1.1. Proceso de medida	12
2.1.2. Herramientas de diagnóstico del proceso de combustión	13
2.1.3. Transductores piezoeléctricos	15
2.1.4. Amplificadores de carga.....	19
2.1.5. Adquisición de datos	23
2.2. Modelo de diagnóstico específico.	25
3. INSTRUMENTACIÓN	31
3.1. Funcionamiento del osciloscopio	31
3.1.1. Modelo y características.	31
3.1.2. Resumen de las funciones utilizadas.....	32
4. MODELO INFORMÁTICO.....	35
4.1. La instrumentación virtual en LV.....	35
4.2. Programación gráfica. LabVIEW y terminología general	36
4.3. Glosario de términos utilizados en el programa.....	39
4.3.1. Arrays y sus funciones.....	39
4.3.2. Clusters y sus funciones.....	41
4.3.3. Estructuras	42
4.3.4. Visualización de datos.....	45
4.3.5. Operadores matemáticos.....	45
4.3.6. Funciones VISA	47
4.3.7. Funciones String.....	50
4.4. Desarrollo de programa.....	55

4.4.1. Prueba.VI	56
4.4.2. Configurawaveform.VI	58
4.4.3. Leeronda.VI	60
4.4.4. Calculaangulos.VI.....	63
4.4.5. Cogepresionciclo.VI.....	67
4.4.6. Presioncorregida.VI.....	69
4.4.7. Diag.VI.....	71
4.5.Gráficas de la P- α y el calor liberado.....	75
5. CONCLUSIONES	79
6. BIBLIOGRAFÍA.....	81

1. INTRODUCCIÓN

1.1. Justificación

La medida de presión en cámara permite obtener datos valiosos para analizar las características de los procesos que tienen lugar en el motor. Teniendo los datos de la variable P a lo largo del ciclo, se puede llegar a obtener la evolución de otros parámetros como la FQL, y por lo tanto analizar la combustión del motor.

En estos tiempos, los ordenadores forman parte de la vida diaria, y su desarrollo permite realizar tareas de alta complejidad con cierta facilidad. Lenguajes de programación como LV permiten tratar los datos que proceden del osciloscopio y hacer todos los cálculos necesarios para analizar las características del motor en tiempo real y así, optimizar la realización de ensayos de la combustión.

1.2. Entorno y antecedentes

El proyecto se ha desarrollado en el laboratorio de motores de combustión de la escuela de ingenierías industriales de la Universidad de Valladolid. Allí se encuentra la instalación experimental, que junto con el osciloscopio y un ordenador han permitido desarrollar el trabajo.



Figura 1.1. Banco de ensayos del laboratorio de motores de la escuela de ingenierías industriales de la universidad de Valladolid.

Una alternativa al sistema utilizado en este proyecto, que se explicará posteriormente, son los sistemas Indimeter. Estos sistemas permiten adquirir y tratar los datos con la ayuda de un PC. Consisten en una tarjeta de adquisición instalada en un PC. Disponen de entradas digitales y analógicas con las que pueden recibir la señal del ángulo (ya acondicionada) así como la señal del amplificador de carga, que es el dato de la presión en cámara. Configurando estos dispositivos, junto con datos técnicos del motor se podrían realizar medidas de forma automática e incluso ver los resultados en diferentes gráficos (P-V, P- α , FQL...).

En esta ocasión se ha optado por utilizar un osciloscopio, en el que se recibe la señal de presión ya en voltios, y se conecta mediante USB a un ordenador para el tratamiento de los datos, con la ayuda del programa LV. Este planteamiento es mucho más general y económico, con la ventaja de poder analizar los datos de forma ilimitada mediante programación, al gusto del usuario. Además con la base realizada y este documento, otros alumnos en el futuro podrán seguir desarrollando el programa.

1.3. Objetivos

El objetivo principal es crear una herramienta de diagnóstico en tiempo real,

Otros objetivos parciales son:

- La medida de la variable P de la cámara de combustión a lo largo del ciclo.
- El control del osciloscopio desde un ordenador mediante conexión USB. Para ello hay que configurar la conexión desde el programa LV.
- La realización de un programa mediante LV que permita calcular variables como el ϑ , el FQL y otros a partir de la P. Así se podrán sacar conclusiones sobre las características del proceso de combustión del motor.
- Se puede destacar como objetivo final, la realización de un documento que sirva de referencia a las personas que utilicen la instalación, alumnos de prácticas y que sirva de base para futuros TFG, TFM (trabajo fin de Máster).

1.4. Estructura del proyecto.

Se ha estructurado el TFG en tres grupos o pasos:

-Primer grupo. Comprende los elementos del equipo experimental en el laboratorio que sirven para medir la P en la cámara de combustión y acondicionar ese dato para el osciloscopio. En el apartado 2.1. se explicará el proceso de medida con profundidad.

Está formado por:

- El banco de ensayos en el cual se encuentra el motor.
- Los captadores piezoeléctricos y amplificadores de carga.
- El codificador angular.

-Segundo grupo. Comprende la comunicación entre el osciloscopio Yokogawa DL 750 y cualquier PC mediante conexión USB. Para esta comunicación se requiere la descarga de controladores de la página de National Instruments que permitirán de manera más rápida comunicar ambos instrumentos. De los controladores descargados se han utilizado los siguientes:

-YKDL750 Config Waveform.VI

-YKDL750 Initialize.VI

-YKDL750 Close.VI

-YKDL750 Read Waveform Length.VI

Son la base para el desarrollo posterior del programa de comunicación con el osciloscopio.

También se deben tener en cuenta los comandos necesarios para comunicarse con el osciloscopio que se encuentran en su manual. Estos se implementarán en el programa LV. Los comandos usados se desarrollarán en el apartado 3.1.2. del documento.

-Tercer grupo. Modelo diagnóstico desarrollado en LV.

Desarrollo del programa informático para el cálculo, a partir del valor de la P en cámara, de muchos otros parámetros como el volumen específico y el calor liberado. También conviene crear una interfaz lo más sencilla y clara posible para que sea accesible controlar el osciloscopio desde el ordenador y también tener en cuenta un posible desarrollo posterior, según requerimientos futuros. Se puede ver en el apartado 4.4.

2. FUNDAMENTOS TEÓRICOS APLICADOS

2.1. Medida de presión en motores

[2] El diagnóstico se basa en la medida con altas frecuencias de adquisición de variables del motor que evolucionan a lo largo del ciclo. En este caso, a partir de la medida de la P en la cámara de combustión en función del ángulo de giro del cigüeñal, α , se irán obteniendo otros parámetros.

Un sistema de medida suele estar formado por:

- Sensor para medir la variable seleccionada.
- Acondicionador de la señal del transductor.
- Sistema de adquisición y software de control.
- Herramienta de postprocesamiento de los datos medidos.

La idea es obtener un diagrama como el de la figura 2.1:

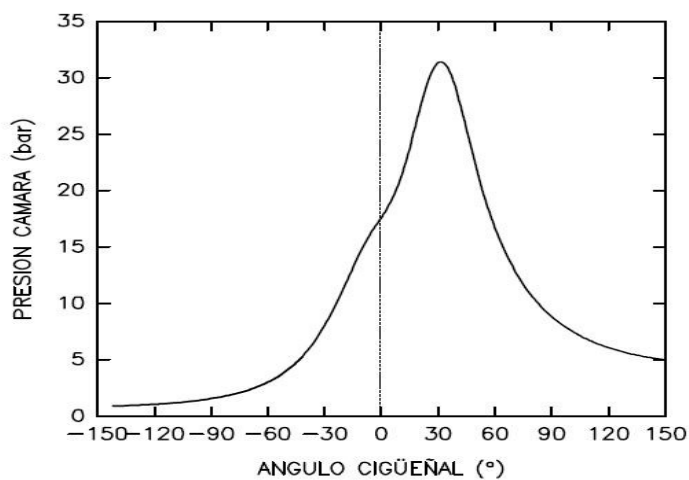


Figura 2.1. Diagrama presión-ángulo de cigüeñal.[4]

2.1.1. Proceso de medida

[5]El proceso de medida consiste en obtener una comparación cuantitativa entre un estándar predefinido y un parámetro físico que está ocurriendo y que está siendo cuantificado.

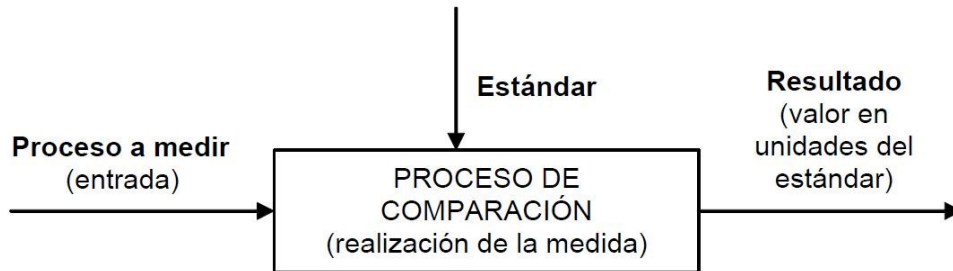


Figura 2.2. Proceso de medida.[5]

En la figura 2.2 se puede observar el proceso. La entrada es el proceso a medir, se compara con un estándar y se obtiene el resultado de esa medida.

Hay dos formas de realizar esa comparación:

- La primera es la comparación directa con patrón.
- La segunda consiste en utilizar un sistema de transducción que convierte la forma de entrada en una magnitud analógica la cual puede ser procesada y presentada. Es la más utilizada debido a que los sentidos humanos no son capaces de percibir con la suficiente resolución ciertas magnitudes, como en el caso de la medida de presión en la cámara de combustión.

Las tres partes más importantes en un proceso de medida son:

- Un sistema de detección-transducción que convierte la magnitud a medir en otra de más fácil detección, como el caso de los transductores piezoeléctricos que se han utilizado en el proyecto.
- Un sistema de acondicionamiento cuya misión es adaptar la salida del transductor a fin de que pueda ser reconocido por un tercer sistema. Esta tarea la realiza el amplificador de carga.
- Sistema de registro para la comprensión por parte del ser humano. Misión de la que se encarga el osciloscopio.

La elección de un sistema de medida u otro depende sobre todo de la evolución en el tiempo del proceso. Las particularidades del proceso son: su rápida variación a lo largo del tiempo, lo que le convierte en un sistema dinámico, su

periodicidad, ya que el proceso se repite cada cierto tiempo, y el interés de conocer la evolución de la variable a lo largo del ciclo.

2.1.2. Herramientas de diagnóstico del proceso de combustión

[2] Se llama fracción de calor liberado hasta un ángulo de cigüeñal dado **FQL(α)** a la parte del total de la energía admitida que el combustible ha liberado durante el proceso de combustión hasta α .

$$FQL(\alpha) = \frac{Q_L}{Q_{TOTAL}} \quad [ecuación 2.1]$$

La fracción de masa quemada **FMQ(α)** puede definirse como la parte de la masa con respecto a la masa total admitida, que ha sido quemada hasta el ángulo α .

$$FMQ(\alpha) = \frac{m_q}{m_{TOTAL}} \quad [ecuación 2.2]$$

Las dos magnitudes mencionadas son muy similares salvo por las imperfecciones del proceso de combustión, en concreto la incompleta conversión del combustible en CO_2 y H_2O .

$$FQL(\alpha) \text{ o } FMQ(\alpha) \left\{ \begin{array}{l} =0 \Rightarrow \text{Antes de combustión} \\ \in (0,1) \Rightarrow \text{Durante la combustión} \\ =1 \Rightarrow \text{Después de la combustión} \end{array} \right.$$

La derivada de $FQL(\alpha)$ representa la cantidad de energía liberada por el combustible por unidad de tiempo o ángulo:

$$DFQL(\alpha) = \frac{dFQL(\alpha)}{d\alpha} \quad [ecuación 2.3]$$

Por tanto la de FMQ representa la cantidad de masa que se quema por unidad de tiempo o ángulo.

$$DFMQ(\alpha) = \frac{dFMQ(\alpha)}{d\alpha} \quad [ecuación 2.4]$$

Como se dijo con anterioridad, a partir de la presión en la cámara de combustión es posible calcular tanto la $FQL(\alpha)$ como la $FMQ(\alpha)$. Esto es debido a que existe una relación biunívoca, entre la evolución de la presión en la cámara y la forma en la que se produce la combustión, para un motor dado y unas condiciones de funcionamiento determinadas. En el diagrama de la figura

2.3, se puede ver representada la evolución de las variables FMQ y DFQL a lo largo del ciclo.

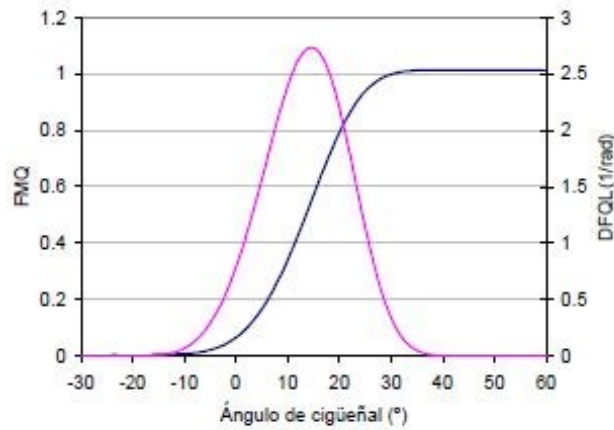


Figura 2.3. Diagramas FMQ Y DFQL según ángulo del cigüeñal [4]

Una particularidad en medida de presión en MEP es el fenómeno de la dispersión cíclica. Consiste en que la velocidad de combustión de ciclos consecutivos presenta una importante variabilidad que se denomina dispersión cíclica o aciclismo. Esta variabilidad en la velocidad de combustión provoca una variación en la presión. En la figura 2.4. se puede visualizar una imagen del fenómeno, en la medida de la variable P, en la que se representan varios ciclos consecutivos. Debido a que el proceso de combustión no es exactamente igual en ciclos consecutivos, tampoco lo es la evaluación de la P.

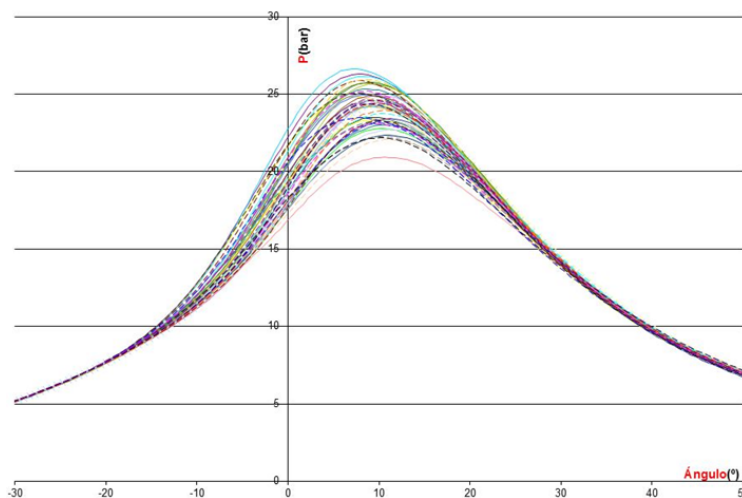


Figura 2.4. Dispersión cíclica experimental de la P en el motor utilizado en el ensayo.

Puede provocar que varíen aspectos del motor como las emisiones contaminantes, la tendencia a la autoinflamación, o las prestaciones del motor. Parámetros que se van a calcular como la $DFQL(\alpha)$ también se verán afectados, y esto se ve en la figura 2.5. Hay que puntualizar, que este fenómeno sobretodo afecta a los MEP. En los motores MEC no es un fenómeno tan importante.

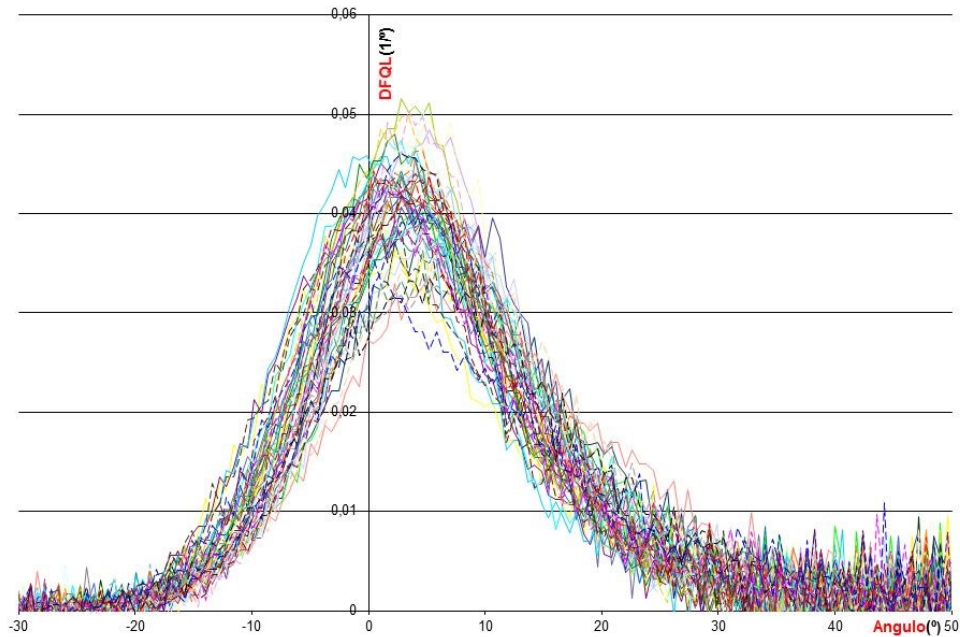


Figura 2.5. Gráfico experimental que representa el fenómeno de la dispersión cíclica en la $DFQL(\alpha)$.

2.1.3. Transductores piezoeléctricos

[2] Los transductores constituyen el elemento de la cadena de medida que entra en contacto con el medio físico que se va a medir. Se encargan de transformar la magnitud a medir en otra distinta. En este caso, los transductores utilizados son de tipo piezoeléctrico.

Los cristales piezoeléctricos proporcionan una señal eléctrica (carga eléctrica) proporcional a la fuerza sobre ellos aplicada, gracias al efecto piezoeléctrico. Este efecto consiste en la generación de carga eléctrica en un material cristalino por la aplicación de una tensión mecánica, que en este caso sería la fuerza ejercida por la presión. La transformación de esa presión en fuerza corre a cargo de los diafragmas. Los diafragmas son un tipo de sensores de fuerza, sensible a la diferencia de presión entre sus dos caras. El proceso de transformación de la P en voltios se puede observar en los diagramas de las

figuras 2.6, que se refiere al proceso de la señal, y la figura 2.7 cuyo diagrama se refiere a los dispositivos por los que pasa la señal en ese proceso.

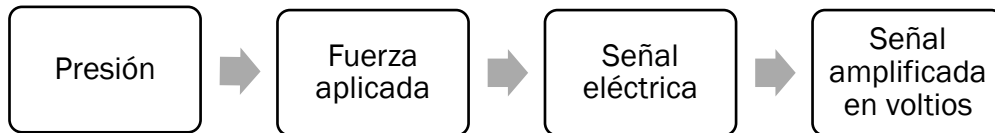


Figura 2.6. Proceso de la señal a lo largo del desarrollo

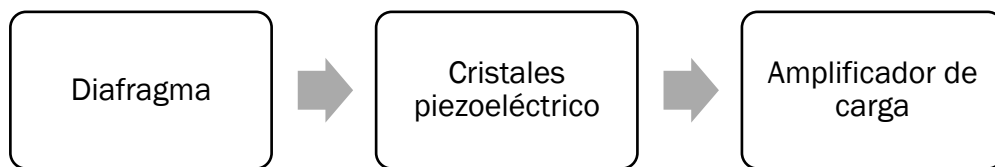


Figura 2.7. Dispositivos por los que pasa la señal durante el proceso.

Existen dos clases de efecto piezoeléctrico, como se puede ver en la figura 2.8:

Efecto piezoeléctrico longitudinal. Se caracteriza por presentar cargas eléctricas en las caras donde se aplica la fuerza, y en magnitud que sólo depende de dicha fuerza y no de las dimensiones del cristal. Es el más utilizado debido a que es una configuración más fácil de construir.

Se puede ver en la figura 2.8 que se disponen los cristales unos encima de otros para ser capaces de soportar una fuerza mayor.

Efecto piezoeléctrico transversal. Las cargas eléctricas aparecen en las caras perpendiculares al lugar dónde se aplica la fuerza, y dependen tanto de la magnitud de la fuerza aplicada como de las dimensiones del cristal.

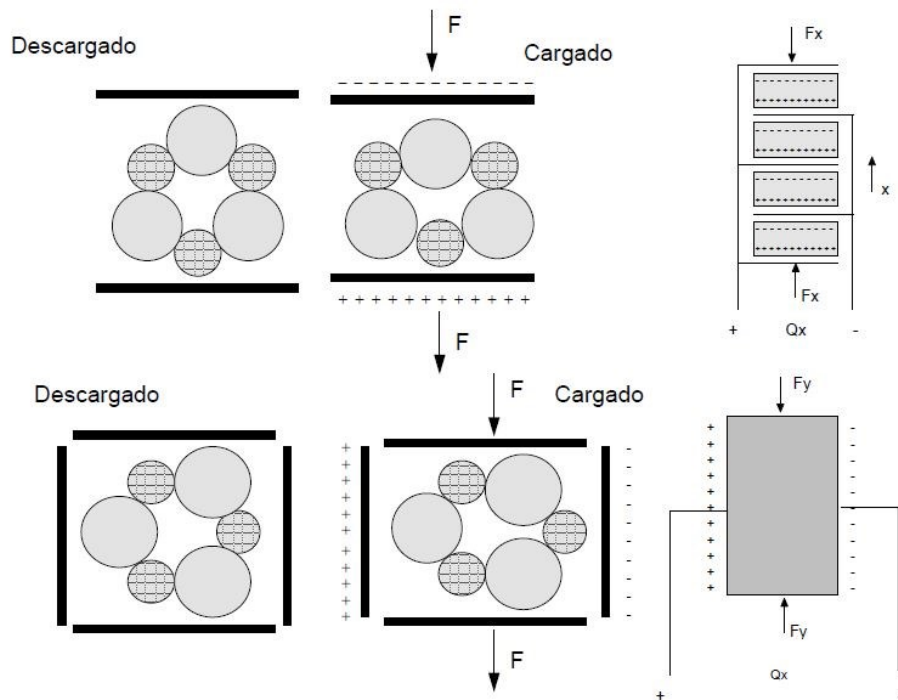


Figura 2.8. Tipos de efectos piezoeléctricos.[2]

Material del cristal

[2] A pesar de que el cuarzo proporciona menos carga eléctrica que otros materiales con propiedades piezoeléctricas, ofrece otras valiosas características que han contribuido a extender su uso de manera notable:

-Alta resistencia a la presión: (40.000 bar). Si bien el límite práctico, está muy por debajo de este valor.

-Temperatura de trabajo hasta 500°C. Si bien por encima de 200°C (punto de Curie) aparecen transformaciones en su estructura cristalina que provocan una pérdida de características piezoeléctricas parcialmente irreversible. A partir de este punto, la sensibilidad decrece fuertemente con la temperatura. Existen cristales de reciente desarrollo que permiten extender este límite hasta 400°C.

-Elevada resistencia de aislamiento. Se entiende por tal la resistencia a través del propio cristal, que dificulte la compensación interna de cargas en el mismo. Para el cuarzo, la resistividad es del orden de $7,5 \cdot 10^{17} [\Omega \cdot m]$

-Elevada linealidad, ausencia de histéresis.

La gráfica de la figura 2.9 es la que se obtiene al calibrar para calibrar un transductor, que contiene una serie de medidas en orden descendente y ascendente de la salida del transductor en función de la magnitud física a medir, que sería la presión en nuestro caso. Con su ayuda, se pueden definir varias características metroológicas de los captadores.

Características metroológicas de los captadores de presión piezoeléctricos.[5]

Ruido

El ruido, en una cadena de medida, está constituido por cualquier perturbación aleatoria, ajena a la magnitud a medir, que sea capaz de alterar la lectura de la medición.

Es importante destacar que es una perturbación aleatoria que puede proceder del propio equipo eléctrico, de los componentes utilizados en la medida, de fenómenos producidos por el propio transductor...etc.

Es un fenómeno totalmente indeseable y siempre se busca la forma de que se mantenga en unos niveles de varios órdenes de magnitud por debajo de la mínima señal a medir.

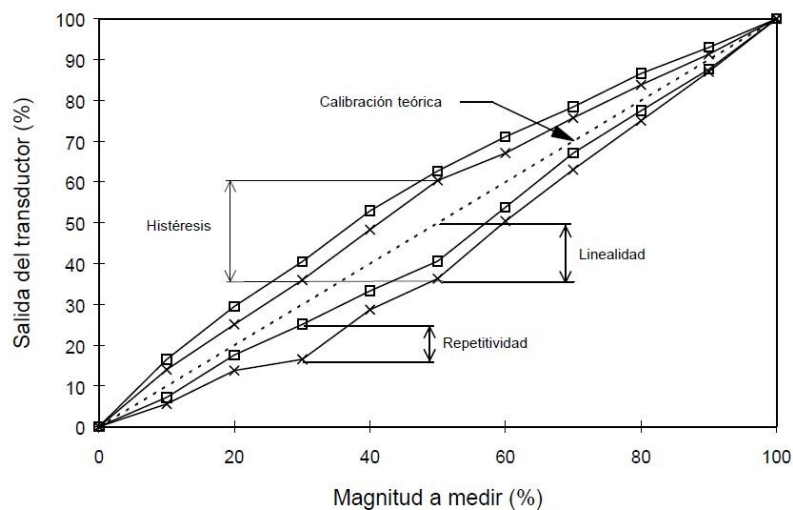


Figura 2.9. Imagen dónde se pueden ver las distintas características metroológicas.[5]

Linealidad

La linealidad se define como la máxima desviación de la curva de calibración respecto a una línea recta, que es la recta discontinua que se puede ver en el centro de la gráfica.

Histéresis

La histéresis por otro lado es la diferencia máxima entre las dos curvas que presenten mayor desviación.

Repetitividad

La repetitividad es la capacidad de un transductor para reproducir el mismo valor a la salida, tras aplicarle a la entrada la misma magnitud, en ciclos sucesivos.

Resolución

Otro concepto importante a definir es la resolución, que es la mínima diferencia en la magnitud a medir que puede ser detectada de manera fiable por el transductor, afectada por factores mencionados antes como la histéresis.

2.1.4. Amplificadores de carga

[2] Debido a que siempre existe un consumo de cargas al acondicionar la señal, la carga acumulada en los transductores piezoeléctricos debe ser medida intentando alterarla lo menos posible. Este fenómeno se intenta evitar utilizando un circuito con un amplificador operacional que convierte la carga acumulada en el transductor en una tensión proporcional.

La tensión de salida es filtrada y amplificada en un amplificador de tensión de ganancia variable, el cual suministra una salida adecuada a los instrumentos de medida de tensión.

Una particularidad de este tipo de transductores es que no son capaces de medir en estacionario. De hecho, el valor medio de la salida de estos transductores es cero, por lo que a posteriori es necesario asignar un valor de offset a la señal.

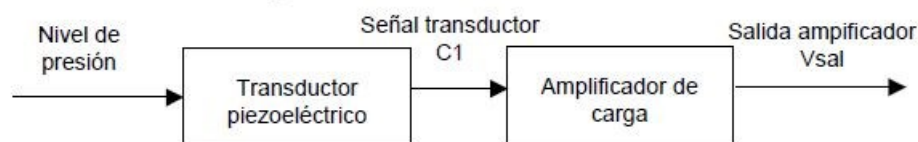


Figura 2.10. Proceso de la señal.[2]

En el diagrama de la figura 2.10 se pueden ver los pasos del proceso desde la variable P hasta el voltaje de salida del amplificador. A continuación, se puede ver el análisis del proceso de forma matemática:

$$C_1[pC] = S_{\text{transductor}} \left[\frac{pC}{bar} \right] \times P[bar] \quad [\text{ecuación 2.5}]$$

siendo $S_{\text{transductor}}$ la sensibilidad del transductor.

$$V_{\text{salida}}[V] = G_{\text{Amplificador}} \left[\frac{V}{pC} \right] \times C_1[pC] \quad [\text{ecuación 2.6}]$$

siendo $G_{\text{Amplificador}}$ la ganancia del amplificador.

Por lo tanto:

$$V_{\text{salida}} = G_{\text{Amplificador}} \times C_1 = G_{\text{Amplificador}} \times S_{\text{transductor}} \times P \quad [\text{ecuación 2.7}]$$

Los amplificadores suelen estar preparados para introducir directamente la sensibilidad del transductor y la ganancia total que se quiere obtener, es decir, qué incremento de tensión se quiere tener a la salida del amplificador para un determinado incremento de presión:

$$\frac{\Delta V_{\text{sal}}}{\Delta P} \left[\frac{V}{bar} \right] = G_{\text{Amplificador}} \left[\frac{V}{pC} \right] \times S_{\text{transductor}} \left[\frac{pC}{bar} \right] \quad [\text{ecuación 2.8}]$$

[5] Las dos características básicas de los amplificadores operacionales son:

- Misma tensión en las dos entradas.
- Ausencia de intensidad en las entradas.

El circuito de la figura 2.10 convierte la variación de la carga en el transductor en una variación de tensión proporcional, para obtener una señal adecuada en voltios.

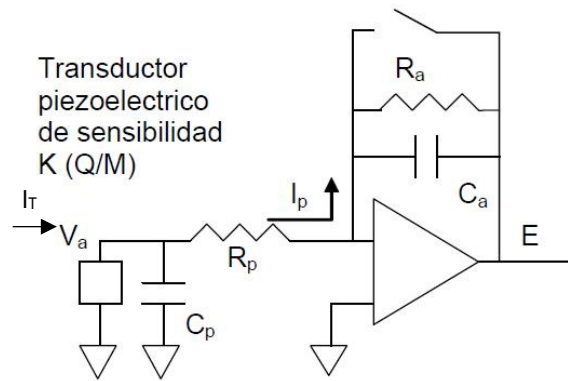


Figura 2.11. Circuito de conversión de la carga en una variación de tensión proporcional.[7]

Se plantea el análisis matemático del circuito teniendo en cuenta sus características mencionadas previamente, de las que se deduce que

La ecuación de la primera rama es la 2.9:

$$\frac{dM}{dt} K = \frac{dQ}{dt} = I_T = -\frac{V_p}{R_p} - \frac{dV_p}{dt} C_p \quad \text{[ecuación 2.9]}$$

Se aplica la transformada de Laplace para pasar al dominio de la frecuencia y así obtener la ecuación 2.10:

$$sKM(w) = -V_p(w) \left(\frac{1}{R_p} + sC_p \right) \quad \text{[ecuación 2.10]}$$

La ecuación 2.11 pertenece a la segunda rama:

$$I_p = \frac{V_p}{R_p} = -\frac{E}{R_a} - \frac{dE}{dt} C_a \quad \text{[ecuación 2.11]}$$

Se aplica nuevamente la transformada de Laplace a la ecuación 2.11 y así pasar al dominio de la frecuencia:

$$V_p(w) \frac{1}{R_p} = -E(w) \left(\frac{1}{R_a} + sC_a \right) \quad [\text{ecuación 2.12}]$$

Se iguala $V_p(w)$ en ambas ecuaciones y se despeja la tensión de salida, E , entre la magnitud M .

$$\frac{E(w)}{M(w)} = \frac{sKR_a}{(1+sR_pC_p)(1+sC_aR_a)} \quad [\text{ecuación 2.13}]$$

Sustituyendo s por jw , se puede obtener la respuesta en frecuencia del sistema, representada en la figura 2.12:

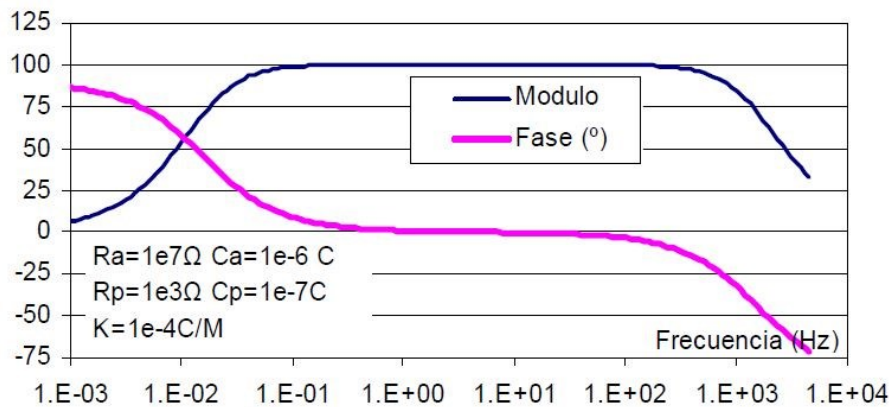


Figura 2.12. Diagrama de módulo y fase. [7]

Teniendo en cuenta que R_p es la resistencia del cable y que C_p es la capacidad del cable, y son valores mucho más bajos que R_a y C_a , se podría considerar que:

$$\left[\begin{array}{l} 1 + jwR_pC_p \approx 1 \\ 1 + wjC_aR_a \approx wjC_aR_a \end{array} \right] \longrightarrow \frac{E(w)}{M(w)} \approx \frac{K}{C_a} \equiv \text{Ganancia}$$

Por lo tanto, existe un rango de frecuencias en el que la ganancia no depende de la frecuencia. Es esta zona en la que puede utilizarse el circuito como amplificador de carga. Es esta zona, la que remarcada en la figura 2.13 con el cuadro verde.

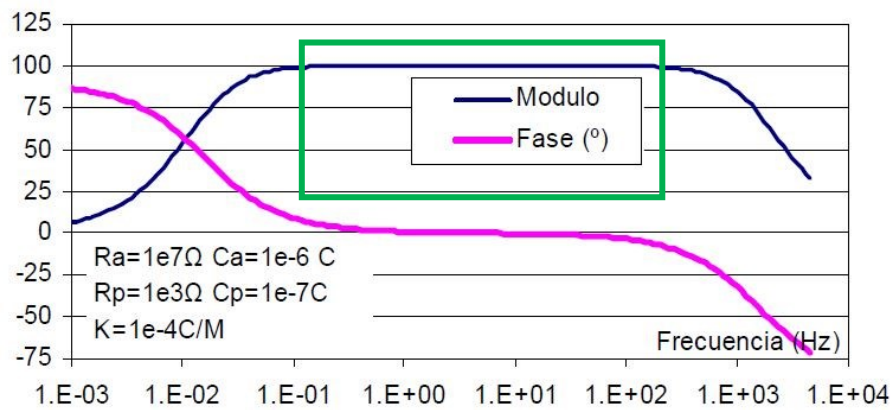


Figura 2.13. Zona de frecuencias en la que la ganancia no depende de la frecuencia.

La conclusión que se ha obtenido de todo esto es que el tipo de transductores utilizado no es capaz de medir en estacionario, de hecho, el valor medio de la salida de estos transductores es cero, por lo que a posteriori es necesario asignar un valor de offset a la señal. Esto se verá implementado en el modelo informático desarrollado en LV, en concreto, en el apartado 4.4.3.

2.1.5. Adquisición de datos

[2] En la siguiente figura se puede ver el proceso completo de medida.

Hay dos partes diferenciadas:

- La primera es la que se refiere al proceso captador-amplificador-osciloscopio.
- La segunda es la debida al codificador angular y al acondicionamiento de señales, en la cual se quiere conocer el ángulo de giro del cigüeñal.

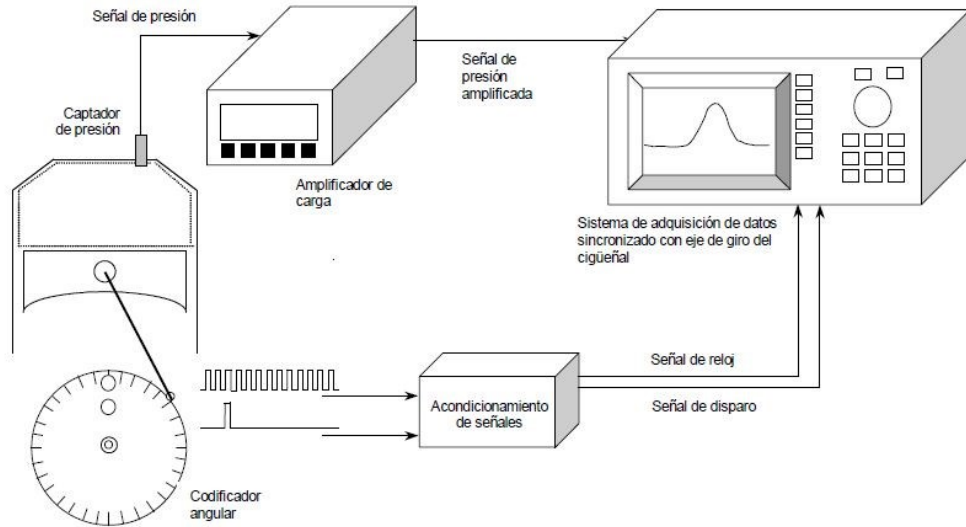


Figura 2.14. Imagen del proceso de adquisición de datos. [2]

Para sincronizar la adquisición de datos con el ángulo de giro del cigüeñal, se coloca en el eje del cigüeñal un codificador angular que genera un tren de pulsos, que se utiliza como señal del reloj del equipo de adquisición (señal NxVuelta) y una señal de disparo de la adquisición (señal 1xVuelta).

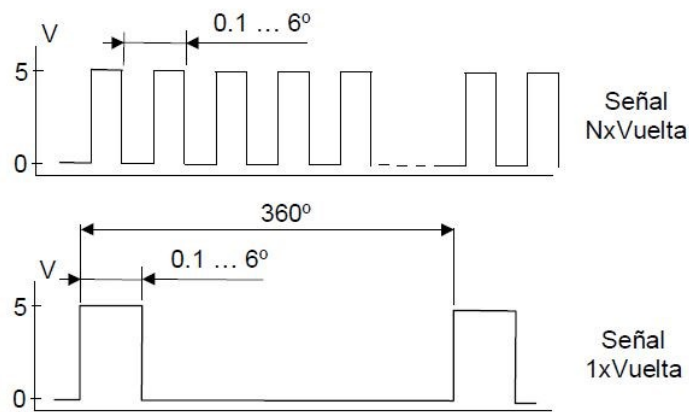


Figura 2.15. Imagen de la sincronización de la señal con el ángulo del cigüeñal. [2]

La señal de disparo inicia la adquisición de manera que el primer dato corresponde a una posición angular fija, a un punto del ciclo termodinámico. Por lo que es necesario saber en qué posición angular está esa referencia respecto del PMS.

La señal de reloj dispara cada una de las adquisiciones individuales de datos, con lo que sabiendo el incremento angular de la señal de reloj se sabe la distancia angular entre cada dato.

De esta forma, sólo es necesario el registro de una única señal, la señal de presión, sabiendo a qué posición angular corresponde cada dato y consecuentemente se conoce el volumen en el interior del cilindro.

2.2. Modelo de diagnóstico específico.

[7] A continuación se detallará el proceso de cálculo del FQL.

Para empezar, se aplica el balance de energía a un sistema cerrado (cámara de combustión) que evoluciona desde el estado 1 al estado 2 en un proceso en el que se intercambia calor y trabajo con el entorno.

$$U_2 - U_1 = Q_{1-2} - W_{1-2} \quad [\text{ecuación 2.14}]$$

- U es la energía interna de los gases contenidos en la cámara.
- Q se refiere al calor neto transferido al sistema durante la transición entre los dos estados, que es el correspondiente al liberado durante el proceso de combustión menos el flujo que sale por las paredes.
- W es el trabajo intercambiado con el pistón en su desplazamiento.

Expresando las magnitudes de forma específica, es decir, respecto de la masa que hay en el cilindro, la cuál no varía, se obtienen las ecuaciones 2.15 y 2.16:

$$m(u_2 - u_1) = Q_{1-2} - W_{1-2} \quad [\text{ecuación 2.15}]$$

$$u_2 - u_1 = \frac{Q_{1-2}}{m} - \frac{W_{1-2}}{m} = q_{1-2} - w_{1-2} \quad [\text{ecuación 2.16}]$$

Una característica importante a nivel termodinámico de este planteamiento, es la ausencia de masa en las ecuaciones que se van a presentar, por lo que no es necesario conocer la masa que hay dentro del cilindro.

El trabajo total realizado cuando el sistema pasa del estado 1 al estado 2 es:

$$w = \int_1^2 P d\theta \quad [\text{ecuación 2.17}]$$

Se sustituye en la ecuación 2.13 del balance de energía y se obtiene la ecuación 2.18:

$$u_2 - u_1 = q_{1-2} - \int_1^2 P d\theta \quad [\text{ecuación 2.18}]$$

Para el caso de un gas ideal puede demostrarse que la energía interna depende exclusivamente de la temperatura, ya que en un gas ideal se desprecia toda interacción entre las moléculas o átomos que lo constituyen, por lo que la energía interna es sólo energía cinética, que depende sólo de la temperatura.

Sabiendo que c_v es la capacidad calorífica molar a volumen constante, y que las temperaturas se expresan en Kelvin para poder aplicar posteriormente la ecuación de estado, que es la 2.25, se obtiene la ecuación 2.19:

$$c_v(T_2-T_1)=q_{1-2}-\int_1^2 P d\vartheta \quad [\text{ecuación 2.19}]$$

Asumiendo que la presión evoluciona linealmente entre los dos estados, se puede aplicar la regla del trapecio para resolver la integral y despejando la variable q_{1-2} obtener la ecuación 2.20 :

$$q_{1-2}=c_v(T_2-T_1) + \frac{P_1+P_2}{2}(\vartheta_2 - \vartheta_1) \quad [\text{ecuación 2.20}]$$

Llamando dq_l al calor liberado en el intervalo, el incremento q_{1-2} corresponde a Δdq_l .

A continuación se calcula la fracción de calor liberado en cada intervalo:

$$q_l [0]=0 \quad [\text{ecuación 2.21}]$$

$$q_l [1]=c_v(T_1-T_0) + \frac{P_1+P_2}{2}(\vartheta_1 - \vartheta_0) + q_l [0] \quad [\text{ecuación 2.22}]$$

$$q_l [2]=c_v(T_2-T_1) + \frac{P_2+P_1}{2}(\vartheta_2 - \vartheta_1) + q_l [1] \quad [\text{ecuación 2.23}]$$

$$q_l [i]=c_v(T_i-T_{i-1}) + \frac{P_i+P_{i-1}}{2}(\vartheta_i - \vartheta_{i-1}) + q_l [i - 1] \quad [\text{ecuación 2.24}]$$

Por lo tanto, es necesario conocer tanto el volumen específico como la temperatura, para poder hallar el calor liberado.

Cálculo de la temperatura

Para lo cuál, se utiliza la ecuación de estado:

$$T = \frac{P \cdot \vartheta}{R} \quad [\text{ecuación 2.25}]$$

siendo P la variable medida en el laboratorio, ϑ el volumen específico que se hallará a continuación, y R la constante universal de los gases ideales, que tomará el valor de 287 J/Kg·K

Cálculo del volumen específico

Asumiendo que la masa no varía durante toda la evolución por los diferentes estados (1,2...n) :

$$m = \frac{V(\alpha)}{\vartheta(\alpha)} = \frac{V(\alpha_0)}{\vartheta(\alpha_0)} \rightarrow \vartheta(\alpha) = \vartheta(\alpha_0) \cdot \frac{V(\alpha)}{V(\alpha_0)} ; \quad [\text{ecuación 2.26}]$$

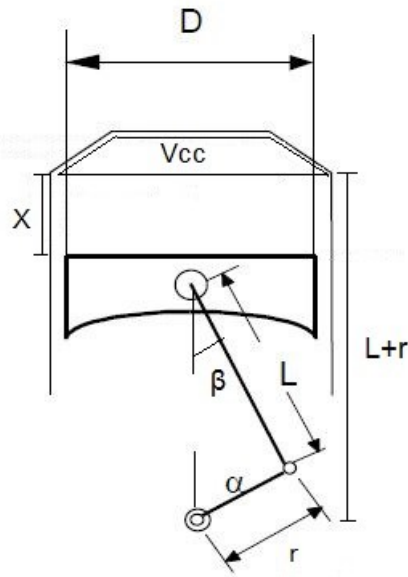


Figura 2.16. Datos geométricos del motor. [4]

Gracias a la figura 2.16, se calcula el volumen total (V) que es igual al V_{cc} , que es el volumen de la cámara de combustión, más el volumen desplazado por el pistón (V_D) que será $V_D = A_p \cdot x$

$$V = V_{cc} + x \cdot A_p \quad [\text{ecuación 2.27}]$$

Se despeja la variable $\frac{V}{V_{cc}}$ de la ecuación 2.27 porque será de interés posteriormente.

$$\frac{V}{V_{cc}} = 1 + \frac{x \cdot A_p}{V_{cc}} \quad [\text{ecuación 2.28}]$$

Se calcula el área del pistón (A_p):

$$A_p = \frac{V_D}{2 \cdot r} \quad [\text{ecuación 2.29}]$$

A continuación se obtiene una relación trigonométrica a partir de la figura 2.16, basándose en los ángulos α , β y en L , longitud de la biela, y r que es el radio de la manivela del cigüeñal.

$$L \cdot \text{sen}\beta = r \cdot \text{sen}\alpha \quad \rightarrow \quad \text{sen}\beta = \frac{r \cdot \text{sen}\alpha}{L} \quad [\text{ecuación 2.30}]$$

También con la ayuda de la figura 2.16 se obtiene la relación de longitudes entre L, r y x.

$$x = L + r - L \cdot \text{cos}\beta - r \cdot \text{cos}\alpha \quad [\text{ecuación 2.31}]$$

Sabiendo la conocida identidad trigonométrica llamada “relación pitagórica”:

$$\text{cos}^2\beta + \text{sen}^2\beta = 1 \quad [\text{ecuación 2.32}]$$

junto con la ecuación 2.30 hallada anteriormente, se puede obtener:

$$\text{cos}\beta = \sqrt{1 - \frac{r^2}{L^2} \text{sen}^2\alpha} \quad [\text{ecuación 2.33}]$$

Se sustituye este valor, en la ecuación 2.31 y se obtiene:

$$x = L + r - L \sqrt{1 - \frac{r^2}{L^2} \text{sen}^2\alpha} - r \text{cos}\alpha \quad [\text{ecuación 2.34}]$$

que a su vez, se sustituye en la ecuación 2.28 para obtener la 2.35:

$$\frac{V}{V_{cc}} = 1 + \frac{V_D}{2rV_{cc}} x = 1 + \frac{V_D}{2V_{cc}} \left[\frac{L}{r} + 1 - \frac{L}{r} \sqrt{1 - \frac{r^2}{L^2} \text{sen}^2\alpha} - \text{cos}\alpha \right] \quad [\text{ecuación 2.35}]$$

Incluyendo un nuevo parámetro, la relación de compresión, r_c .

$$r_c = \frac{V_{\max}}{V_{\min}} = \frac{V_D + V_{cc}}{V_{cc}} = \frac{V_D}{V_{cc}} + 1 \quad \rightarrow \quad r_c - 1 = \frac{V_D}{V_{cc}} \quad [\text{ecuación 2.36}]$$

y sustituyendo esta relación en la ecuación 2.35:

$$\frac{V}{V_{cc}} = 1 + \frac{r_c - 1}{2} \left[\frac{L}{r} + 1 - \sqrt{\frac{L^2}{r^2} - \text{sen}^2 \alpha} - \cos \alpha \right] \quad [\text{ecuación 2.37}]$$

A continuación se dividen entre V_{cc} los dos términos de la ecuación 2.26 para obtener la ecuación del volumen específico, ϑ , en función del ángulo, α , y de otros parámetros conocidos. Se ha cumplido uno de los objetivos del cálculo y se ha obtenido una ecuación adimensionalizada ya que el parámetro r_c es adimensional. Por lo tanto, se puede afirmar que el modelo diagnóstico utilizado es independiente del tamaño del motor.

$$\vartheta(\alpha) = \vartheta(\alpha_0) \cdot \frac{V(\alpha)/V_{cc}}{V_0/V_{cc}} = \vartheta(\alpha_0) \cdot \frac{1 + \frac{r_c - 1}{2} \left[\frac{L}{r} + 1 - \sqrt{\frac{L^2}{r^2} \text{sen}^2 \alpha} - \cos \alpha \right]}{1 + \frac{r_c - 1}{2} \left[\frac{L}{r} + 1 - \sqrt{\frac{L^2}{r^2} \text{sen}^2 \alpha_0} - \cos \alpha_0 \right]} \quad [\text{ecuación 2.38}]$$

Es esencial señalar que el punto inicial, 0, es el momento del cierre de la admisión, y que a parte de los datos de presión, P_{ca} , que se miden en el laboratorio, también hay que medir la temperatura en ese momento, T_{ca} . Con estos, es posible calcular el $\vartheta(\alpha_0)$.

No obstante, en realidad es imposible medir T_{ca} . Se debe tener en cuenta que las variaciones de temperatura son enormes y que se dan en periodos de tiempo muy cortos. La obtención de P_{ca} viene dada por la gráfica $P-\alpha$ que se obtiene, pero contando con un offset. En consecuencia, en el laboratorio los tres parámetros que se modificarán para conseguir unos datos coherentes serán: P_{ca} , T_{ca} y α_0 que es el ángulo inicial que se mide respecto al PMS.

Los calculos del apartado 2.2. se pueden ver en el apartado 4.4.7. implementados en el programa LV.

3. INSTRUMENTACIÓN

3.1. Funcionamiento del osciloscopio

3.1.1. Modelo y características.

[1] El modelo utilizado es el YOKOGAWA DL 750. Para la conexión con el PC se utilizará un cable usb (bus universal en serie).

El osciloscopio es básicamente un dispositivo de visualización gráfica que muestra señales eléctricas variables en el tiempo. El eje vertical representa el voltaje, mientras que el eje horizontal representa el tiempo.

Los osciloscopios pueden ser analógicos ó digitales. Los primeros, prácticamente en desuso, trabajan directamente con la señal aplicada, que una vez amplificada desvia un haz de electrones en sentido vertical proporcionalmente a su valor. En contraste los osciloscopios digitales utilizan previamente un conversor analógico-digital (A/D) para almacenar digitalmente la señal de entrada, reconstruyendo posteriormente esta información en la pantalla. El modelo que se ha utilizado es de tipo digital.[9]

El DL750 ScopeCorder combina las funciones de un osciloscopio para capturar fenómenos instantáneos y un registrador de datos para el monitoreo de tendencias a largo plazo. Es un grabador de 8 ranuras con hasta 16 canales analógicos y 16 entradas lógicas dependiendo de los módulos de entrada seleccionados. La funcionalidad de captura dual realiza la adquisición de datos en la misma forma de onda a dos velocidades de muestreo diferentes y la funcionalidad GigaZoom permite la visualización instantánea de datos de longitud completa.

En la siguiente tabla se pueden ver el tiempo de comunicación con el ordenador en función de los tipos de datos por el puerto USB:

Volumen de datos	Formato Byte	Formato Word	Formato ASCII
1000	Aprox.29 ms	Aprox.31 ms	Aprox.479 ms
10000	Aprox. 47ms	Aprox. 52 ms	Aprox. 4.5 s
100000	Aprox. 210	Aprox. 330 ms	Aprox. 45.1 s
1000000	Aprox.2 s	Aprox. 3.2 s	Aprox. 453 s

Tabla 3.1. Tiempos de comunicación con el PC para varios tipos de datos del modelo YKDL 750 Yokogawa, por el puerto USB. [1]

Los datos formato Byte y Word son binarios y por ello tardan mucho menos en transmitirse. El formato ASCII en cambio es mucho más lento. Se trabajará con transmisión de datos de formato Word.

3.1.2. Resumen de las funciones utilizadas.

El primer paso es la configuración del osciloscopio para la conexión mediante usb.

Para ello, se debe presionar la tecla MISC marcada en rojo en la siguiente imagen y cuyo botón se encuentra en el panel frontal.

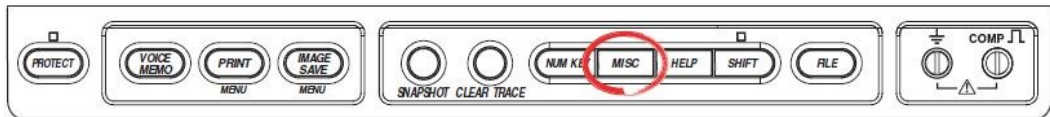


Figura 3.1. Imagen del botón MISC en el panel frontal del osciloscopio. [1]

Posteriormente, se selecciona Remote Control, Device USB utilizando el botón giratorio que se encuentra también en el panel frontal.

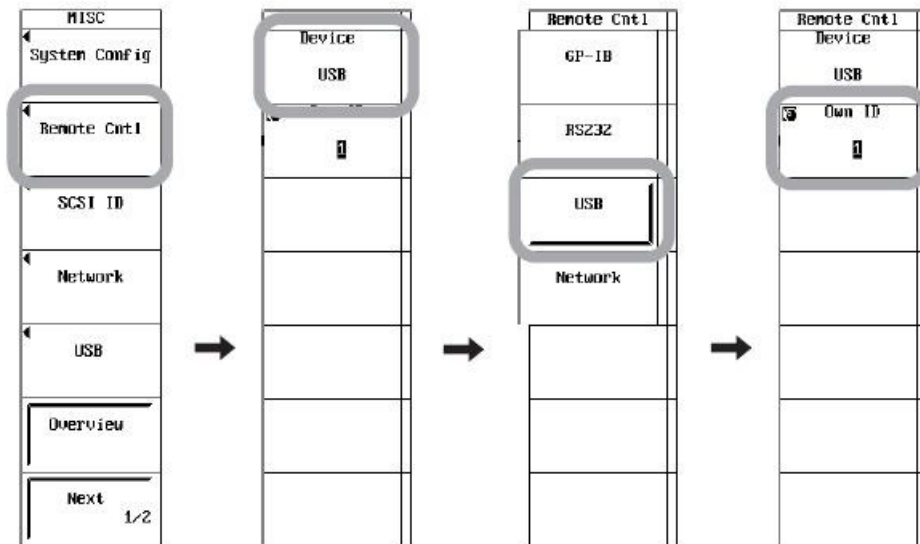


Figura 3.2. Configuración del osciloscopio para la entrada USB. [1]

A continuación se enumeran los grupos de comandos de comunicación con el osciloscopio más utilizados en el desarrollo de la programación en LV:

RECOOrder Group

Los comandos de este grupo se ocupan de la grabación en tiempo real de los datos recibidos. Se puede hacer lo mismo pulsando la tecla RECORDER que se encuentra en el panel frontal.

STARt Group

Este grupo es usado para comenzar la adquisición de la forma de onda. Se puede ejecutar la misma función pulsando el botón START/STOP que se encuentra en el panel frontal.

STATus Group

Este grupo de comandos se utilizan para hacer ajustes y configuraciones relacionados con el informe del estado del dispositivo.

En concreto, se han usado:

- :STATus?

Pregunta la configuración relacionada con el estado de comunicación de esta función.

- :STATus:CONDition?

Pregunta los contenidos del registro de condiciones.

WAVEform Group

Este otro grupo de comandos trata con la adquisición de los datos en forma de onda. No existe en el panel frontal ningún botón que sustituya esta función.

- WAVEform:SEND?

Pregunta la forma de la onda. En el caso de que los tipos de datos están configurados en tipo word, los datos pueden ser convertidos a voltaje usando la siguiente ecuación:

$$\text{Voltaje} = \frac{\text{Rango-datos} \cdot 10}{\text{División}} + \text{Offset} \quad [\text{ecuación 3.1}]$$

Siendo el valor de Division = 24000,

Range es el valor que devuelve al preguntar: WAVEform:RANGe?

Offset es el valor que devuelve al preguntar :WAVEform:OFFSet?

- *WAVEform:START*

Establece el primer punto de los datos en forma de onda especificado por el *WAVEform:TRACe*.

- *WAVEform:TRACe*

Establece el objetivo de los datos en forma de onda o pregunta la configuración actual.

4. MODELO INFORMÁTICO

4.1. La instrumentación virtual en LV

[6] Un instrumento virtual es un módulo de software que simula el panel frontal de un instrumento de medida, como por ejemplo la carcasa delantera llena de botones de un osciloscopio. En la figura 3.4, se puede ver el panel frontal del osciloscopio usado:

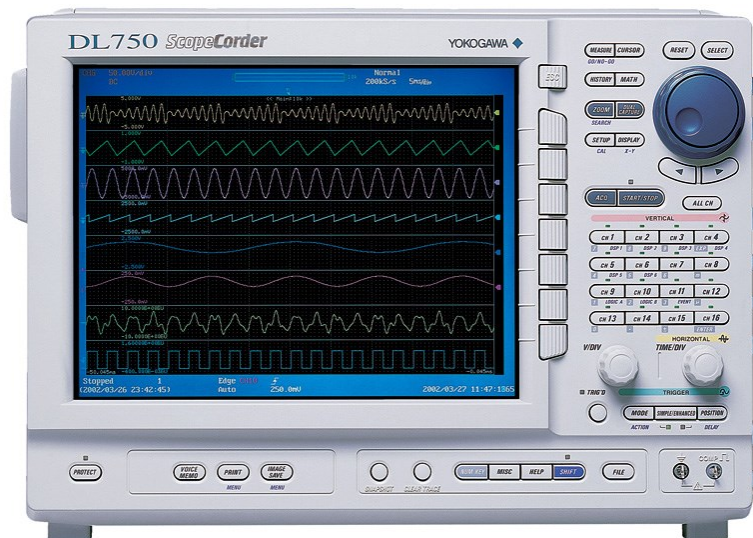


Figura 4.1. Imagen del osciloscopio utilizado.

Apoyándose en elementos software compatibles con el ordenador mediante muchas posibles vías de comunicación como GPIB, RS-232, USB, Ethernet, realiza una serie de medidas como si se tratase de un instrumento real.

De esta manera, cuando se ejecuta un programa que funciona como instrumento virtual o VI (Virtual Instrument), el usuario ve en la pantalla de su ordenador un panel cuya función es idéntica a la de un instrumento físico facilitando la visualización y el control del aparato. Partiendo de los datos reflejados en el panel frontal, el VI debe actuar recogiendo o generando señales como lo haría su homólogo físico.

La utilización del ordenador como control de instrumentación no es algo novedoso de la actualidad. Ya en la década de los 70 se usaba mediante la interface de bus IEEE 488 o GPIB (General Purpose Interface Bus). De hecho, en un primer momento en este proyecto se planteó la posibilidad de utilizar el osciloscopio Yokogawa DL708E y comunicarlo mediante GPIB. Pero finalmente se buscó otra opción más moderna y se eligió el dispositivo DL750 para poder realizar la conexión mediante USB, que es más rápida y que está muy estandarizada en la actualidad.

4.2. Programación gráfica. LabVIEW y terminología general

[6],[3] Hasta hace un tiempo, para realizar la construcción de un VI se utilizaban softwares que pese a que ofrecían una serie de facilidades como el uso de algunas funciones gráficas, la mayor parte del programa estaba basado en texto. Esto implicaba una inversión de tiempo muy alta.

Hasta que llegó el momento en el que nació LabVIEW (“Laboratory Virtual Instrumentation Engineering Workbench”) en el año 1986, que utiliza programación de código G siendo la G de gráfico. Se puede utilizar en muchos sistemas como Windows, Mac OS, Linux...

LV es un entorno de programación gráfico que puede utilizarse para crear aplicaciones de forma rápida y eficiente con interfaces de usuario profesionales. Los programas de LV son llamados instrumentos virtuales o VIs. Están formados por panel frontal, que es la interfaz de usuario y diagrama de bloques, el programa que está bajo esa interfaz. En las figuras 4.2 y 4.3 se pueden observar el panel frontal y el diagrama de bloques del programa principal respectivamente.

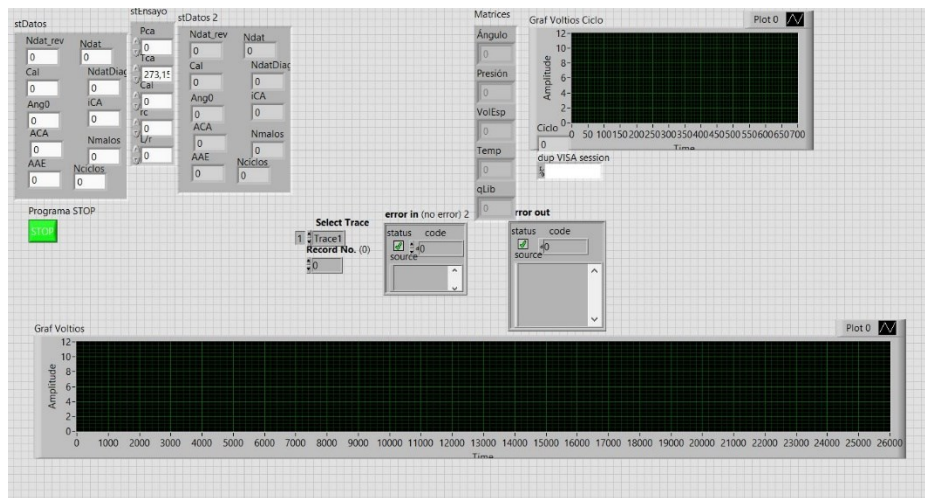


Figura 4.2. Panel frontal del programa principal de LV.

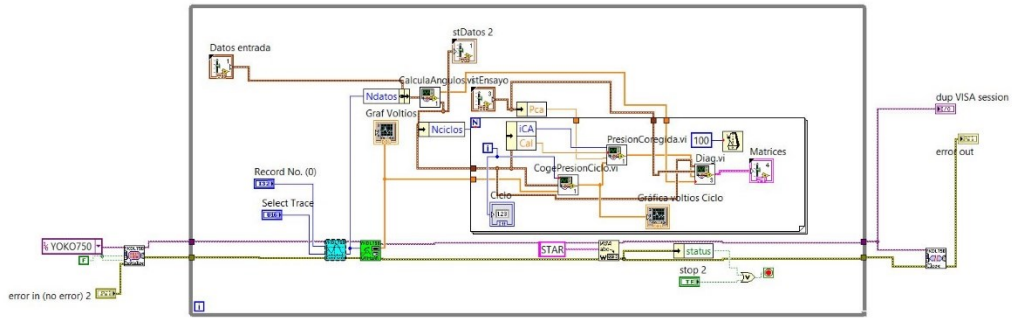


Figura 4.3. Diagrama de bloques del programa principal de LV.

Tras crear la ventana del panel frontal, se puede añadir código usando representaciones gráficas de funciones para controlar los objetos del panel frontal. En lenguajes de programación basados en texto, las instrucciones determinan el orden de ejecución, en cambio, LV tiene la peculiaridad de usar flujo de datos gráfico, es decir, los datos fluyen a través de nodos en el diagrama de bloques y eso es lo que determina su orden de ejecución.

Entre los objetos del diagrama de bloques se incluyen los terminales, subVIs, funciones, estructuras y cables.

Terminales

Los objetos de la ventana del panel frontal aparecen como terminales en el diagrama de bloques. Los terminales son puertos de entrada y de salida que intercambian información entre el panel frontal y el diagrama de bloques.

-Terminales de control/indicador: están asociados a los controles e indicadores del panel frontal. En un terminal de control se puede modificar y controlar el valor, como el de la figura 4.4. En la figura 4.5 se puede ver un terminal indicador, que indica el valor de una variable determinada.

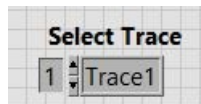


Figura 4.4. Selección de pista.

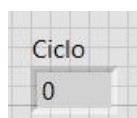


Figura 4.5. Imagen del indicador de ciclo.

Funciones

Son elementos operativos fundamentales de LabVIEW. No tienen ventanas de panel frontal ni de diagrama de bloques pero sí paneles de conectores. Son aquellas que tienen un fondo amarillo pálido, como la función Add de la figura 4.6:

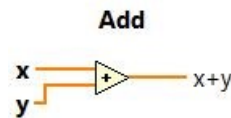


Figura 4.6. Función suma.

SubVIs

Los subVIs son VIs que se insertan en el diagrama de bloques pero cuyo código se encuentra en otro fichero.

Cables

Con los cables se transfieren datos entre objetos del diagrama de bloques. Los cables tienen distintos colores, estilos y grosores, en función de sus tipos de datos. Los cables se pueden romper por varias razones, como al intentar cablear dos objetos con tipos de datos incompatibles.

En la imagen de la figura 4.7 se pueden ver los diferentes colores, grosores y formas de los cables:

	Scalar	1D Array	2D Array	
Numeric				Orange (floating point)
				Blue (integer)
Boolean				Green
String				Pink

Figura 4.7. Tipos de datos y colores de cables.[6]

Los cables que intercambian datos de tipo numérico son de color naranja para los datos en coma flotante y azules para enteros.

En el caso de los datos booleanos, son de color verde pero de tipo discontinuo. Finalmente los string tienen una forma particular y son de color rosa.

El grosor y la forma cambian tanto en el caso de arrays de una y dos dimensiones.

4.3. Glosario de términos utilizados en el programa

[3] La idea de escribir un pequeño glosario de las funciones utilizadas, que son las más comunes, es hacer una guía al lector del trabajo realizado en la parte de programación del TFG, y servir de base para futuros trabajos. Debido a que en el apartado 4.4, no se ha usado mucha terminología específica del programa, y se han intentado explicar los pasos dados con la mayor sencillez posible, aquí se explicarán con detalle las funciones utilizadas en LV.

4.3.1. Arrays y sus funciones

Un array es un conjunto de datos del mismo tipo. Puede tener una o más dimensiones y hasta 2^{31} elementos por dimensión. Puede ser de cualquier tipo excepto de otros arrays, graphs o charts. Se accede a cada elemento de un array mediante un índice, el cual es cero-base, es decir, va de 0 a N-1, donde N es el número de elementos.

ARRAY SIZE (tamaño del array).

Devuelve el número de elementos del array.



Figura 4.8. Función Array Size.

BUILD ARRAY (construir array).

Concatena arrays o añade elementos extras a un array. Se puede redimensionar para incrementar el número de entradas.

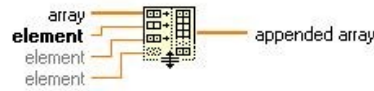


Figura 4.9. Función Build Array.

ARRAY SUBSET (subarray de un array).

Devuelve una parte de un array a partir de un índice y longitud determinados.

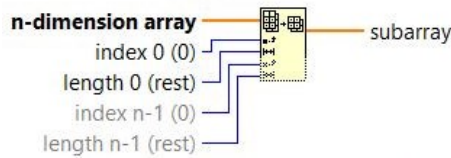


Figura 4.10. Función Array Subset.

INDEX ARRAY (indexar array).

Devuelve un elemento de un array de n-dimensión desde un índice.

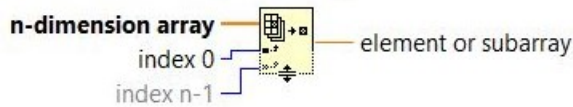


Figura 4.11. Función Index Array.

SPLIT 1D ARRAY.

Divide un array de una dimensión, desde un índice, en dos arrays.

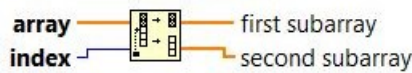


Figura 4.12. Función Split 1D Array.

4.3.2. Clusters y sus funciones

Un cluster es una colección ordenada de uno o más elementos. A diferencia de los arrays, estos elementos pueden ser de cualquier combinación de tipos de datos. Tienen un tamaño fijo y no pueden tener combinación de controles e indicadores.

UNBUNDLE (separar).

Descompone un cluster en sus elementos individuales.

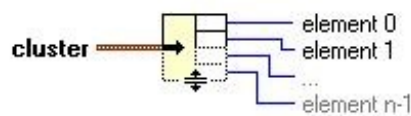


Figura 4.13. Función Unbundle.

BUNDLE (unir).

Une todas las entradas individuales en un único cluster o cambia los valores de los componentes conectados.

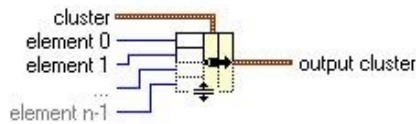


Figura 4.14. Función Bundle.

UNBUNDLE BY NAME (separar por nombre).

Devuelve los elementos del cluster cuyos nombres se especifican.

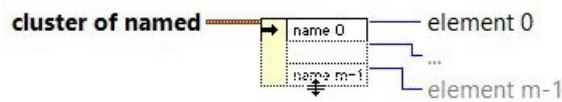


Figura 4.15. Función Unbundle by Name.

BUNDLE BY NAME (unir por nombre).

Reemplaza componentes en un cluster ya existente, especificando su nombre.

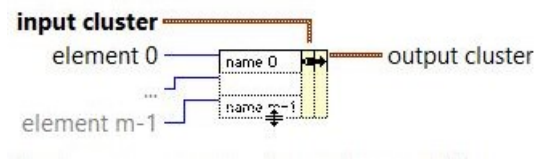


Figura 4.16. Bundle by Name.

INDEX AND BUNDLE CLUSTER ARRAY.

Indexa un conjunto de arrays y devuelve un cluster de arrays.

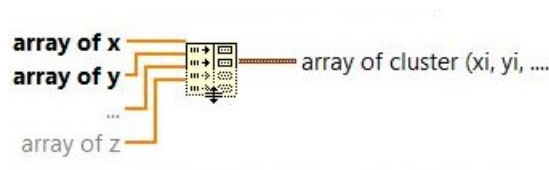


Figura 4.17. Función Index and Bundle Cluster Array.

4.3.3. Estructuras

BUCLE FOR

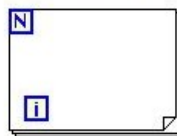


Figura 4.18. Bucle For.

Se usan este tipo de bucles cuando se quiere repetir una operación un número determinado de veces. Se ejecuta desde $i=0$ hasta $N-1$ veces.

-El terminal N es el llamado terminal contador. Contiene el número de veces que se ejecutará el subdiagrama creado en el interior de la estructura. Hay varias formas de fijar ese valor y se hará desde el exterior.

-El terminal i nos indica el número de veces que se ha ejecutado la estructura.

BUCLE WHILE

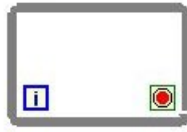


Figura 4.19. Bucle While.

Se usan este tipo de estructura cuando se quiere repetir una operación mientras una determinada condición sea cierta o falsa.

-El terminal condicional es el círculo rojo. A él se conecta la condición que hará que se ejecute el subdiagrama. Entonces cada vez que se ejecute una iteración, LV comprueba el estado del terminal.

Hay dos opciones:

-Stop if true: parará cuando la condición sea cierta.

-Continue if true: parará cuando la condición sea falsa.

-El terminal i es el de iteración. Indica el número de veces que se ha ejecutado el bucle y que, como mínimo, siempre será una $i=0$.

WAIT UNTIL NEXT ms MULTIPLE



Figura 4.20. Función de temporización.

No es una estructura pero se usa en ellas. Es una función de temporización en la ejecución del código. Se añade en los bucles iterativos (While y For). Sirve para esperar una cantidad determinada en milisegundos entre una iteración y la siguiente.

CASE ESTRUCTURE

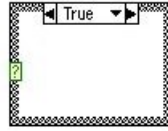


Figura 4.21. Estructura Case.

Esta estructura permite tener dos o más alternativas para un mismo paso.

El terminal selector (se puede ver en la imagen con un signo de interrogación verde). Por ejemplo, al añadir un selector booleano, en el caso de que este selector resulte true, se pondrán en marcha el caso TRUE, y en el caso de que sea false, se pondrá en marcha el caso FALSE.

Si al terminal de pregunta se le añade un selector numérico, pues hará lo mismo pero con las opciones 1,2,3...

FORMULA NODE

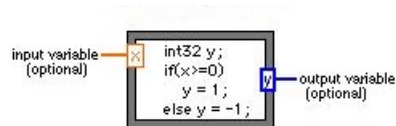


Figura 4.22. Función fórmula.

Es una función de características similares a las estructuras vistas, pero en lugar de contener un subdiagrama, contiene una o más fórmulas separadas por un punto y coma. Esta función se utiliza para ejecutar fórmulas matemáticas que serían complicadas de crear utilizando las diferentes herramientas matemáticas que LV incorpora.

Una vez escrita la fórmula en el interior del rectángulo, sólo hay que añadir los terminales que harán la función de variables de entrada o de salida.

4.3.4. Visualización de datos

WAVEFORM GRAPH

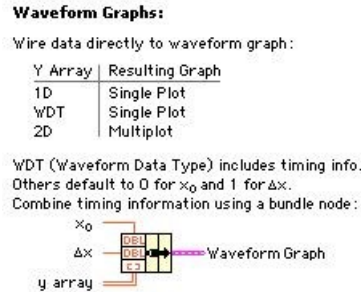


Figura 4.23. Función Wavephorm Graph.

Representa una serie de valores Y equiespaciados dada siempre una distancia delta de X (ΔX) comenzando a partir de un valor inicial X_0 . A un mismo punto X_1 sólo le puede corresponder un valor de Y_1 .

Cuando se representa una nueva serie de datos, estos datos reemplazan a los ya existentes en lugar de añadirse al lado, y pierden los valores representados con anterioridad.

Existen dos posibilidades a la hora de representar una única gráfica en una waveform graph. La primera consiste en unir un array de valores numéricos directamente a la graph de forma que esta interpreta cada valor como un nuevo punto comenzando en $x=0$ e incrementando x en 1 para cada punto.

La segunda consiste en crear un cluster en el cual, junto con el array de valores, se indica el valor inicial X_0 , y el incremento ΔX .

4.3.5. Operadores matemáticos

TO LONG INTEGER

number — **I32** — 32bit integer

Figura 4.24. Función conversión a tipo I32.

Convierte un número a uno de tipo I32. Estos números son enteros, tienen 32 bits de almacenamiento, con o sin signo, y un intervalo de -2.147.483.648 a 2.147.483.647.

NOT



Figura 4.25. Función NOT.

Si el valor que entra es falso, la función devuelve TRUE. Si es cierto, la función devuelve FALSE.

ROUND TOWARD INFINITY



Figura 4.26. Función de aproximación al entero próximo más bajo.

Sirve para aproximar el número de entrada al próximo entero más bajo.

TO SINGLE PRECISION FLOAT



Figura 4.27. Función conversión a número en coma flotante.

Los SGL son un tipo de números en coma flotante, es decir, fraccionales. Además son de precisión simple, con un formato de 32 bits. Son los más sencillos y los que menos sobrecargan la memoria.

TO UNSIGNED WORD INTEGER

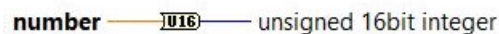
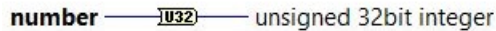


Figura 4.28. Función conversión a tipo Word U16.

Convierte un número en uno de tipo Word U16. Este es un número entero sin signo con 16 bits de almacenamiento y un intervalo de 0 a 65.535.

TO UNSIGNED LONG INTEGER



number — U32 — unsigned 32bit integer

Figura 4.29. Función conversión a entero Long U32.

Convierte un número en uno de tipo Long U32. Este tipo de número presenta las siguientes características: es entero, no tienen signo, tienen 32 bits de almacenamiento y un intervalo de 0 a 4.294.967.295.

4.3.6. Funciones VISA

Para realizar una comunicación con un instrumento se pueden utilizar diferentes buses de comunicación siempre y cuando el instrumento disponga de ellos. Algunos de los buses más utilizados en instrumentación son GPIB, de serie, VXI... Para acceder al bus desde un ordenador personal la mayoría de las veces es necesaria una tarjeta controladora del bus, ya sea GPIB, VXI, o actualmente adaptadores de bus serie puesto que muchos ordenadores ya prescindieron de esta característica. Para acceder a cada una de estas tarjetas controladoras de bus se puede utilizar las funciones o Vis propios del bus.

La utilización de estas funciones propias del bus fuerzan a que las aplicaciones desarrolladas para un determinado instrumento trabajan sobre un bus específico: por ejemplo, si se utilizan las funciones de GPIB para controlar un multímetro vía GPIB no se pueden utilizar dicho programa para controlar el mismo instrumento utilizando un bus diferente. Para solucionar este y otros problemas semejantes de interoperabilidad entre instrumentos de diferentes fabricantes, en 1993 National Instruments junto con otras empresas como GenRad, Racal Instruments, Tektronik y Wavetek formaron el consorcio llamado VXIplug&play Systems Alliance. Uno de los estándares desarrollados por este grupo fue VISA (Virtual Instrument Software Architecture) que es una API (Application Programming Interface) de alto nivel que se encarga de hacer transparentes los recursos softwares que se utilizan.

Utilizando VISA, con el mismo código del programa se puede controlar cualquier instrumento independientemente de la vía de comunicación (GPIB, USB...)

Por lo tanto, las características más importantes de una comunicación que utiliza VISA son:

- Independencia de la plataforma utilizada.

-Independencia de la interface utilizada.

VISA Session: debe abrir una sesión VISA con un recurso para comunicarse con él, al igual que un canal de comunicación.

Las funciones de comunicación VISA más utilizadas son las funciones VISA Write y VISA Read. En la mayoría de los instrumentos debe enviar información en forma de comando o consulta antes de poder volver a leer información del instrumento. Por lo tanto, la función VISA Write suele ir seguida de una función VISA Read. Éstas funcionan con cualquier tipo de comunicación de instrumento y son las mismas independientemente del tipo de comunicación.

VISA WRITE

Es el VI que LV proporciona para enviar datos por el puerto serie. La representación del icono se puede ver en la siguiente imagen:

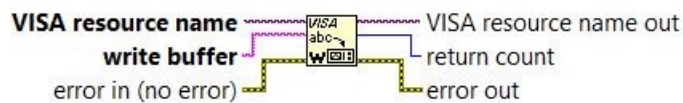








Figura 4.30. VISA Write.

-  **VISA resource name** indica el puerto serie donde se quiere enviar la cadena de caracteres que previamente se escribirá en el buffer mediante esta función. En este caso, el YOKO750.
-  **write buffer** aquí se han de introducir , en forma de cadena de caracteres, los datos que se desean enviar por el puerto serie.
-  **error in** describe las condiciones del error que ocurren antes de que este nodo se ejecute.
-  **VISA resource name out** es igual que la entrada.
-  **return count** contiene el número actual de bytes escritos.
-  **error out** contiene información del error.

Lectura del puerto serie

Una vez configurado el puerto serie y establecida la conexión, es posible que el periférico haya transferido datos al ordenador. Para el programador, este proceso de recepción de datos es transparente, es decir, es el propio puerto serie quien se encargará de gestionar la comunicación con el periférico si este desea enviar información. Una vez finalizada la transferencia, los datos recibidos quedan almacenados en el buffer de recepción a la espera de que sean leídos. Hasta ese punto, todos los pasos son realizados de forma automática, sin intervención del programador.

Para acceder a esa información, es necesario programar un acceso de lectura al puerto. Para ello, se usará la función VISA Read.

Hay que tener en cuenta que una vez se haya leído cierta información del buffer de recepción, esta dejará de estar almacenada en él, dejándose espacio en el buffer para nuevos datos que puedan llegar en el futuro.

VISA READ

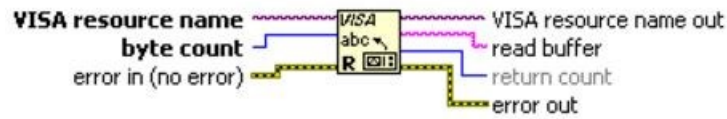









Figura 4.31. VISA Read.

-  **VISA resource name** especifica el recurso que va a ser abierto. En este caso, el YOKO750.
-  **byte count** es el número de bytes que se desean leer del puerto serie.
-  **error in** describe las condiciones del error que ocurren antes de que este nodo se ejecute.
-  **VISA resource name out** es igual que la entrada.
-  **read buffer**: una vez se ha ejecutado el icono, esta salida devuelve en forma de cadena de caracteres (string), los datos leídos del buffer de recepción.
-  **return count** contiene el número de bytes realmente leídos.
-  **error out** contiene información del error.

4.3.7. Funciones String

CONCATENATE STRINGS

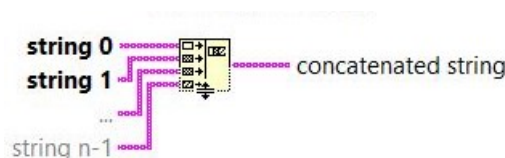


Figura 4.32. Función concatenación de Strings.

Sirve para unir strings en el orden en el que entran en la función.

DECIMAL STRING TO NUMBER

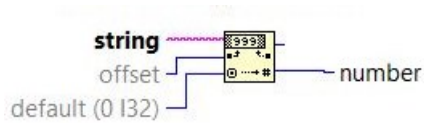


Figura 4.33. Función conversión string en número.

Convierte los caracteres numéricos de la cadena string, empezando por el que se indica en el offset, a un número entero decimal y lo devuelve en la salida “number”.

- abc **string** puede ser un string, un cluster de strings, un array de strings o un array de clusters de strings.
- I32 **offset** : es un número que especifica la posición de comienzo. Para el primer carácter del string sería 0. Si ponemos un número <0 o bien no ponemos nada, por defecto vale 0.
- I32 **default** sirve para indicar la representación numérica de la salida number. Por defecto viene un entero de 32 bit de valor 0.
- I32 **number** puede ser un número, un cluster, un array de números...dependiendo de la estructura del string.

FORMAT INTO STRING

Función que sirve para convertir datos de carácter binario, string, en datos ASCII para sacarlos por pantalla y que se puedan leer.

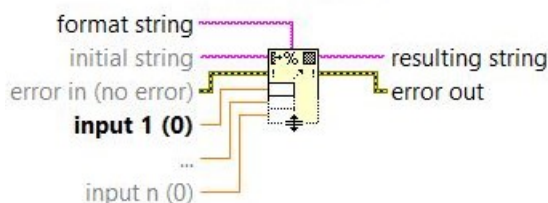





Figura 4.34. Función Format into String.

- abc **format string** especifica cómo convertir los argumentos de entrada en el string resultante. Por ejemplo, si se pone %d , se redondea el argumento a un entero

 **initial string** especifica el string base al cuál se puede añadir cualquier argumento para formar el string resultante.

 **error in**

 **input 1..n** especifica los parámetros de entrada que la función convierte.

 **resulting string** contiene la concatenación del string inicial más la salida formateada.

 **error out**

PICK LINE

Se utiliza para elegir la pista del osciloscopio. En la entrada multi-line se incluye un string de varias filas, en el que están las diferentes opciones de canal. En el comando line index, se elige la final y por tanto, se elige el canal. En el string resultante (output string), se encuentra el string completo formado por el string de entrada más la línea elegida del multi-line string.

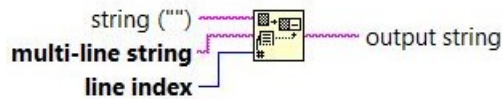



Figura 4.35. Función Pick Line.

 **string** es el string al cual LV añade el string elegido del multi-line string. Por defecto es un string vacío.

 **multi-line string** contiene varios strings separados en filas.

 **line index**

El índice de línea selecciona la línea del multi-line string. Debe ser numérico. Un índice de línea de 0 selecciona la primera línea

 **output string** es el string resultante.

FRACT/EXP STRING TO NUMBER

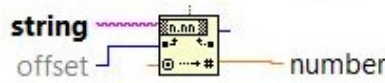


Figura 4.36. Función Fract/Exp String to Number.

Interpreta los caracteres desde 0 hasta 9, +, -, e, E, y el punto decimal en string, empezando en offset, como un número en coma flotante en notación de ingeniería o formato exponencial o fraccional que devuelve en la salida number.

- abc **string** puede ser un string, un cluster de strings, un array de strings o un array de clusters de strings.
- I32 **offset** es un número que especifica el punto donde comienza a actuar la función, siendo 0 el primero.
- DBL **number** puede ser un string, un cluster de strings, un array de strings o un array de clusters de strings, dependiendo de la estructura del string.

SEARCH/SPLIT STRING

Función usada en leeronda.vi para quitar los 10 primeros datos que llegan del osciloscopio, que será el offset, siendo string la cadena que llega del osciloscopio. En la salida tenemos el string, que contiene el resto de datos, quitando los 10 primeros.

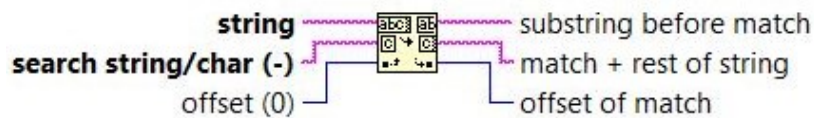


Figura 4.36. Función Search/Split String.

Se utiliza para dividir un único string en dos substrings.

- abc **string** es el string de entrada.
- I32 **offset** es un número que especifica donde hago la separación.
- abc **substring before match** devuelve la porción previa al offset.
- abc **match + rest of string** devuelve la otra porción del string.

UNFLATTEN FROM STRING

Función que permite convertir los datos en string en datos numéricos.

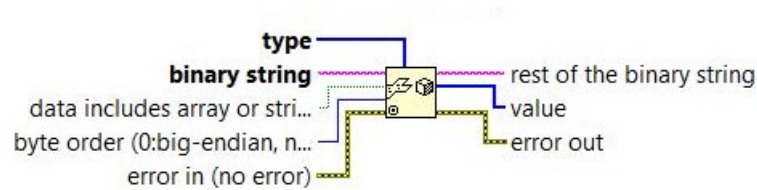





Figura 4.38. Función Unflatten from String.


 **type** es el tipo de datos al cual se quiere convertir el string.


 **binary string** es el string de entrada.


 **byte order** indica el orden de transmisión de los bytes.

0	big-endian, network order (por defecto).El más significativo se manda el último.
1	native, host order . Usa el formato de transmisión que tenga el ordenador con el que se establece la comunicación.
2	little-endian . El byte menos significativo se transmite el último.

 **error in** error de entrada.

 **rest of the binary string** contiene los bytes que quedan sin transmitir de la función string.

 **value** devuelve el string en el tipo de datos indicado.

 **error out** error de salida.

4.4. Desarrollo de programa

En este apartado se van a explicar los distintos programas realizados en LV.

En el apartado 4.4.1 se encuentra el programa principal, llamado **Prueba**. De él, derivan el resto de programas: **Configurawaveform**, **Leeronda**, **Calculaangulos**, **Cogepresiónciclo**, **Presióncorregida**, **Diag**.

El VI **Configurawaveform**, que se encontrará en el apartado 4.4.2 permite configurar el osciloscopio y cuantificar el número de datos que se reciben del osciloscopio para trabajar con ellos posteriormente.

El programa **Leeronda.VI** se ha creado con el objetivo de transformar los datos recibidos de la P en voltios.

El VI **CalculaÁngulos** permite, partiendo del nº de datos, el ángulo inicial, y el nº de datos por revolución, calcular el número de datos “malos” (que son unos datos no válidos que se tendrán que eliminar porque son de la primera revolución del motor desde que comienza la adquisición), el número de ciclos, el AAE, el ACA, el iCA y el ángulo del ciclo.

En el apartado 4.4.5 se encuentra el programa **Cogepresiónciclo.VI** que permite obtener la porción de datos de P en voltios con la que se quiere trabajar, que será la porción de datos del ciclo.

El programa **Presióncorregida** sirve para convertir el valor de la P en voltios a P en bares y corregir su valor con la ayuda del Pca.

En el apartado 4.4.7 el VI denominado **Diag** me permite calcular ϑ , T^a y FQL para todos los puntos del ciclo.

Cada uno de los programas viene explicado de dos formas:

-La primera, con un diagrama de bloques. Se ha tratado de ser muy escueto para lograr informar al lector de lo que se ha hecho de manera clara y muy breve.

-La segunda, es una explicación más detallada en un texto. No se van a explicar todos los pasos y funciones, ya que hay algunas que no aportan información relevante para explicar el proceso.

4.4.1. Prueba.VI

La figura 4.39 es el diagrama de bloques del programa principal. Se puede ver la estructura básica del programa y el orden en el cuál se producen las acciones. Debido a que los otros VI's se van a explicar más detenidamente, se ha preferido explicar el funcionamiento básico del programa con dos diagramas de bloques:

-El diagrama descriptivo de la figura 4.40 en el cuál se puede ver el orden de ejecución de los VI's.

-En el diagrama de la figura 4.41 se explica la función principal de cada programa y por lo tanto, la del programa Prueba completo.

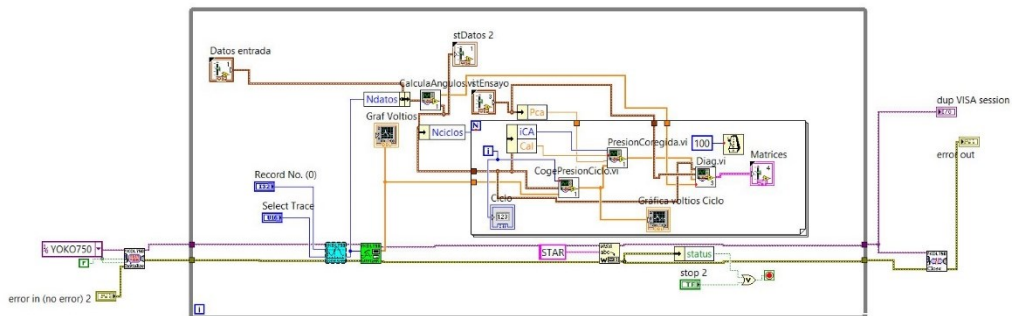


Figura 4.39. Diagrama de bloques de Prueba.VI.

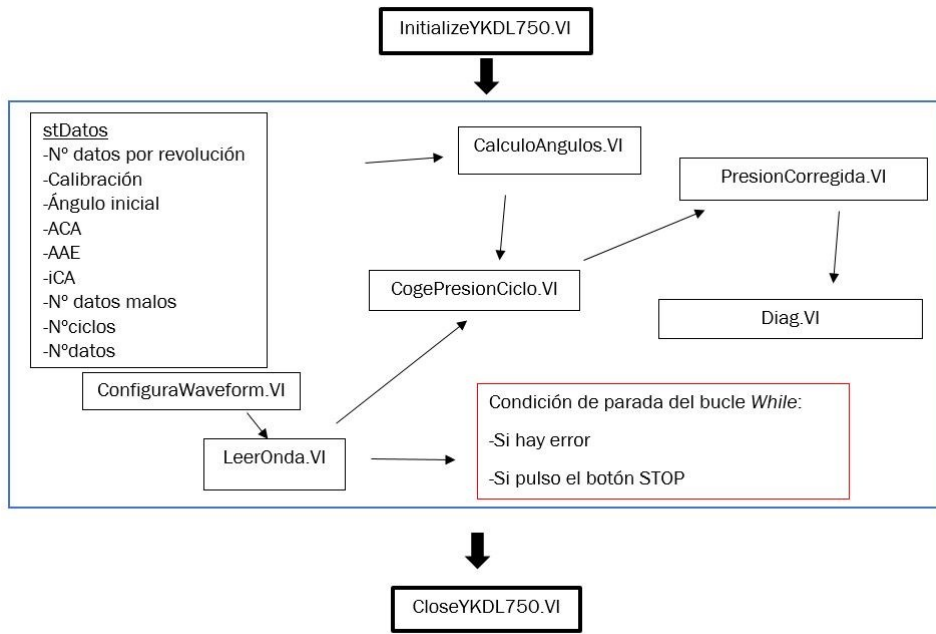


Figura 4.40. Diagrama de VIs del programa Prueba.

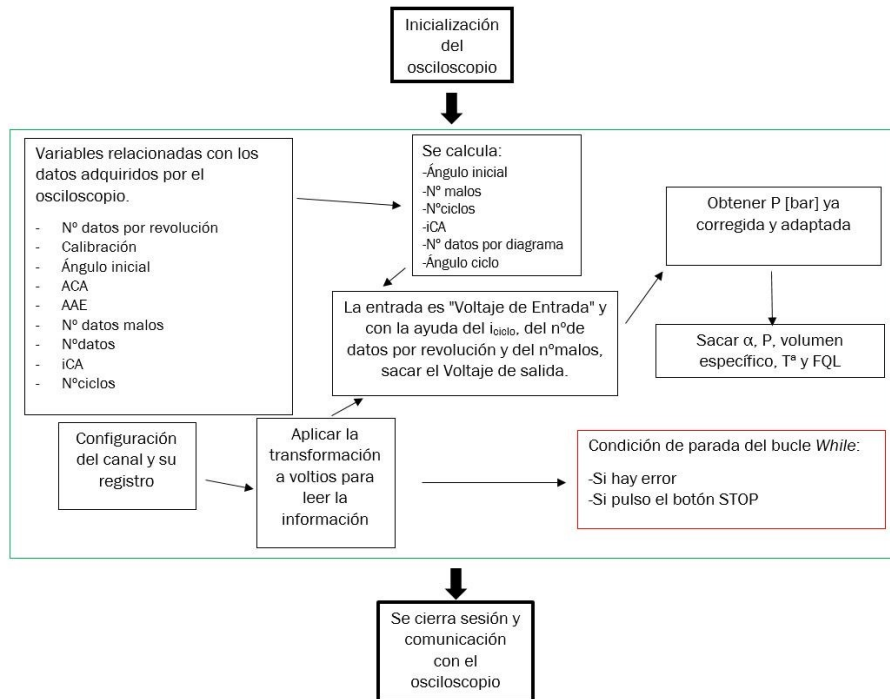


Figura 4.41. Diagrama descriptivo de Prueba.VI.

4.4.2. Configurawaveform.VI

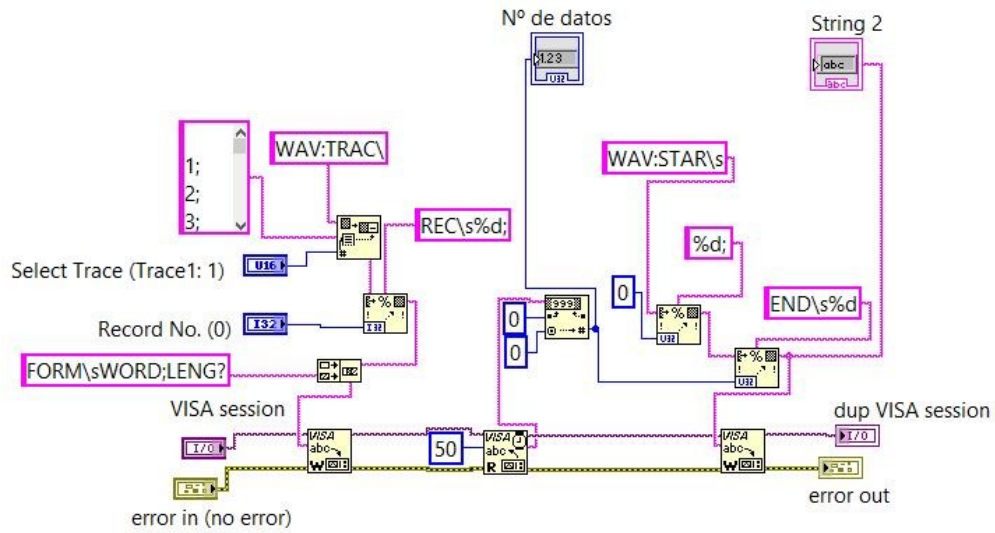


Figura 4.42. Diagrama de bloques de Configurawaveform.VI

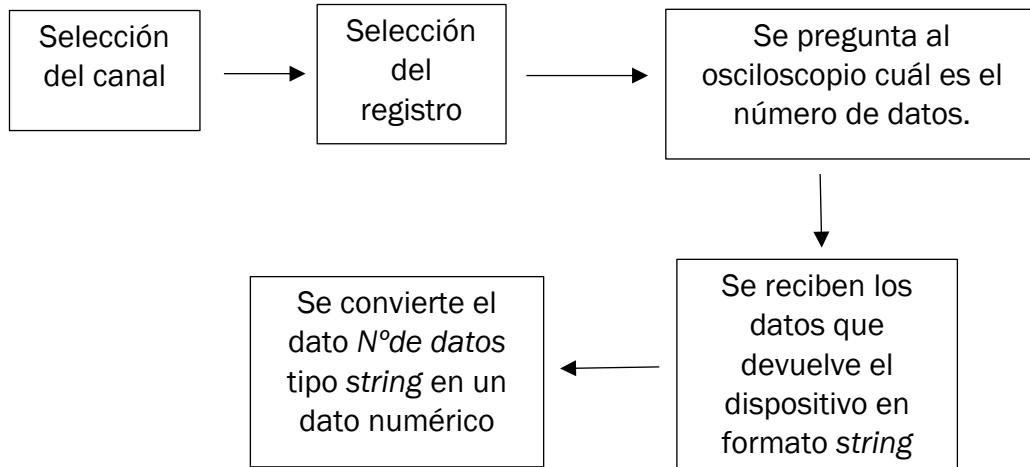


Figura 4.43. Diagrama de bloques de Configurawaveform.VI

El programa *ConfiguraWaverform* sirve para configurar el osciloscopio de forma remota. Lo primero, es configurar el canal por el cuál se va a transmitir la información al osciloscopio, utilizando para ello la función *pickline*, que permite elegir el canal entre muchas opciones desde 1 hasta 16. Se ha elegido el canal 1 para este trabajo. Posteriormente, se pasa a elegir el registro donde se utilizará el comando *REC*, del grupo de comandos *RECOder Group* que sirven para grabar los datos recibidos en tiempo real.

A posteriori, tras inicializar la sesión *Visa*, se envía al osciloscopio la pregunta sobre la longitud, es decir, la cantidad de datos de la cadena *string* que son el número de datos que se reciben. Para ello, se utiliza una *Visa Write*. Y siempre, después de una *Visa Write*, habrá una *Visa Read* (se ha hablado de esto detenidamente en el apartado 4.3.6) para poder leer la respuesta del osciloscopio.

El siguiente paso es la función *Decimal String to Number*, que permite convertir el número de datos en formato *string*, en datos numéricos. Ya es posible sacar un indicador en el panel frontal que permite visualizar el número de datos. El panel frontal se puede ver en la figura 4.44:

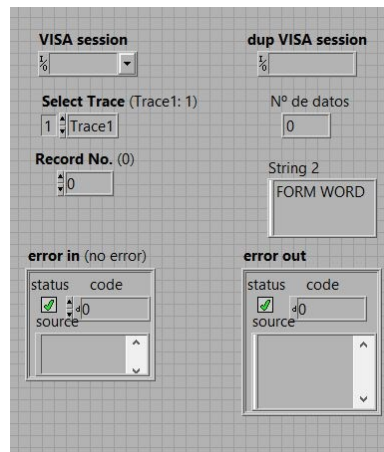


Figura 4.44. Panel frontal de ConfiguraWaveform.

Finalmente, se cierra la sesión *Visa*.

4.4.3. Leeronda.VI

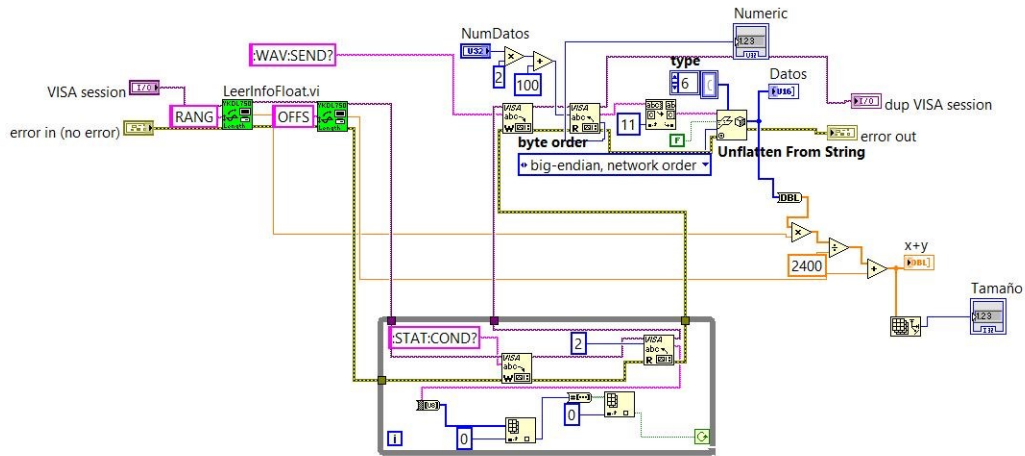


Figura 4.45. Diagrama de bloques de Leeronda.VI

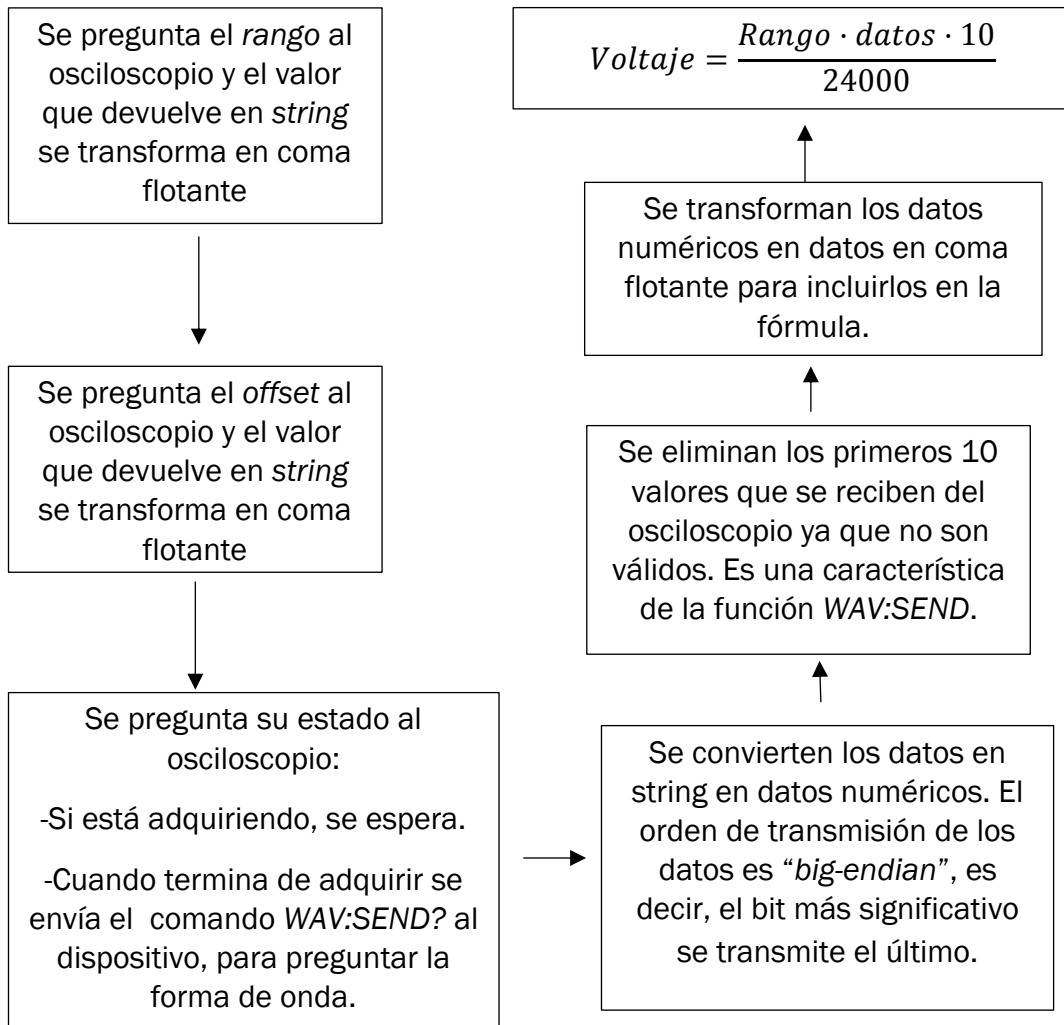


Figura 4.46. Diagrama descriptivo de Leeronda.VI

El programa *Leeronda.VI* comienza inicializando la sesión *Visa*. Después se avanza hasta el *LeerintoFloat.VI* que es un pequeño programa que se puede ver en la figura 4.47.

Este programa consiste en preguntar el rango con el comando *WAVeformRANGe*. Por supuesto, como el resto de los programas, necesita inicializar previamente una sesión *Visa*, y también un *Visa Write* y *Visa Read*. El dato que devuelve el osciloscopio es una cadena string. Se utiliza la función *Fract/Exp String to Number* que permite transformar los valores de esa cadena string en datos numéricos de tipo coma flotante, es decir, fraccionales.

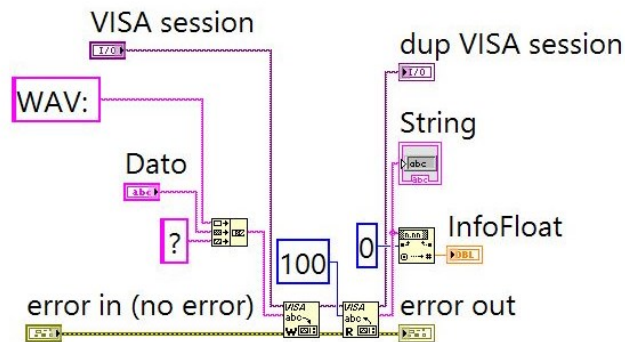


Figura 4.47. LeerIntoFloat.VI

Posteriormente se repite la acción, pero en vez de preguntar por el rango, se pregunta por el offset, utilizando para ello el mismo programa de la figura 4.47, pero en este caso, con el comando *WAVeformOFFSet*.

A continuación, se abre un bucle *While*. En él, se pregunta al dispositivo por su estado, es decir, se quiere saber si está adquiriendo datos en ese momento. En el caso de que esté adquiriendo, se espera. Pero en el caso de que haya terminado de adquirir, se avanza con el programa. Es importante destacar que el bucle *While* tiene un terminal condicional configurado como *Continue if true*, es decir, parará sólo si la condición es falsa, si no, continúa. Se puede ver más explicado en el apartado 4.3.3.

Por otro lado, se utilizan los datos numéricos obtenidos en el programa *Configurawaveform* que es el número de datos con el que se va a trabajar.

Se aplica el comando *WAVeform:SEND?*, comentado en el apartado 3.1.2, que sirve para preguntar la forma de onda y se hace con un byte count igual al número de datos por 2 debido a que cada dato tiene 2 bytes ya que se está trabajando con datos de carácter binario tipo Word. La respuesta del dispositivo es de tipo string, por lo tanto, se debe transformar ese tipo de datos en numéricos. Antes de eso, se deben quitar los 10 primeros datos, ya que no son válidos, y comenzar en el dato número 11. Esto se realiza con la función *Search/Split String* que sirve para, partiendo de un string, dividirlo en dos substrings desde un offset que será 11. Por lo tanto, se tiene un string con los

10 primeros datos y otro string a partir del dato número 11. El motivo por el cuál se deben quitar esos diez primeros datos, es porque el manual de utilización del osciloscopio lo pide.

Tras este paso, con el substring que interesa, se llega a la función *Unflatten from String*, que efectivamente sirve para transformar esos datos tipo string en datos numéricos. Eligiendo para ello, el orden de transmisión de datos tipo *big-endian*, es decir, en el cuál el más significativo se manda el último.

A continuación se aplicará la ecuación 3.1. (apartado 3.1.2) que permite transformar en voltaje los datos. Como se puede ver se deben utilizar el rango, el número de datos y el offset, que se han obtenido previamente:

$$\text{Voltaje} = \frac{\text{Rango} \cdot \text{datos} \cdot 10}{24000} + \text{Offset} \quad [\text{ecuación 3.1}]$$

Esta fórmula que se aplica viene en el manual del osciloscopio y es una fórmula específica para datos binarios, bien de tipo Word o de tipo Byte. Si fuera el caso de tener establecido el osciloscopio para datos de formato ASCII, habría otros requerimientos para transformar ese dato en voltios.

Ya se puede decir que se tienen los datos de presión recibidos del osciloscio en voltios, lo cuál es un paso muy importante para posteriormente aplicar los cálculos del calor liberado y el volumen específico.

4.4.4. Calculaangulos.VI

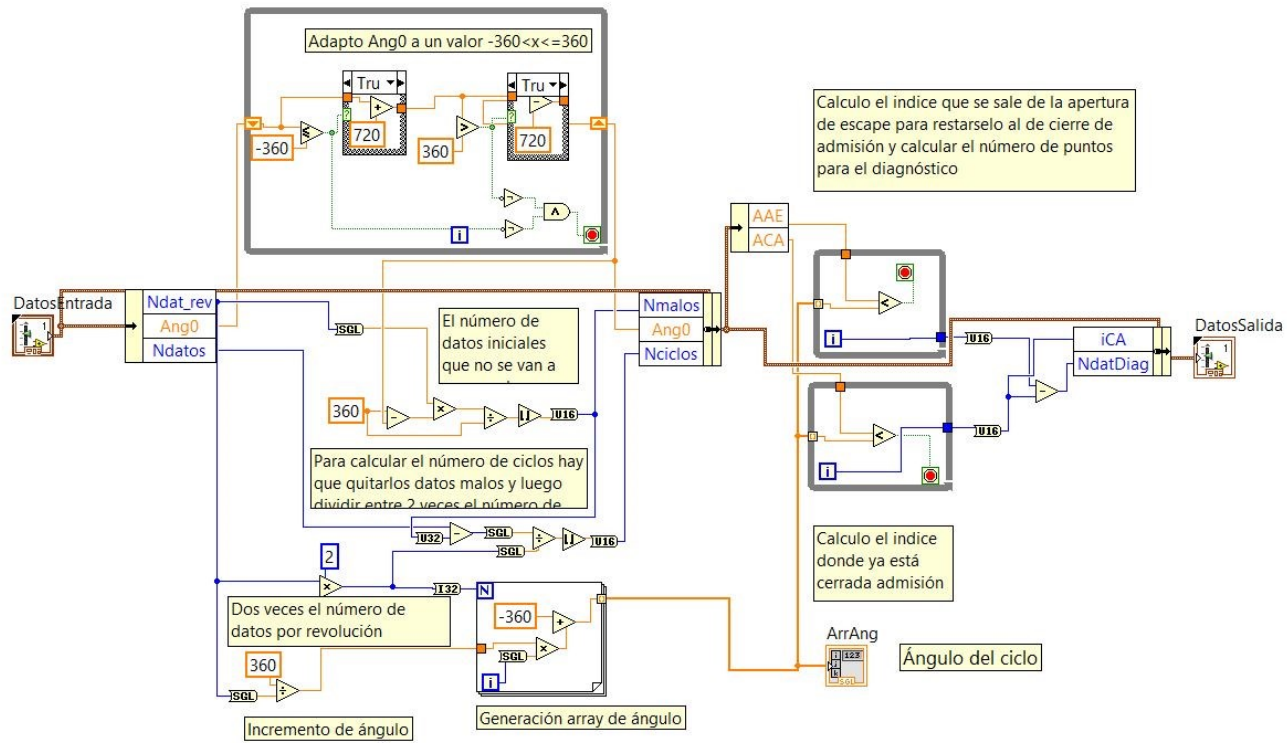


Figura 4.48. Diagrama de bloques de Calculaangulos.VI

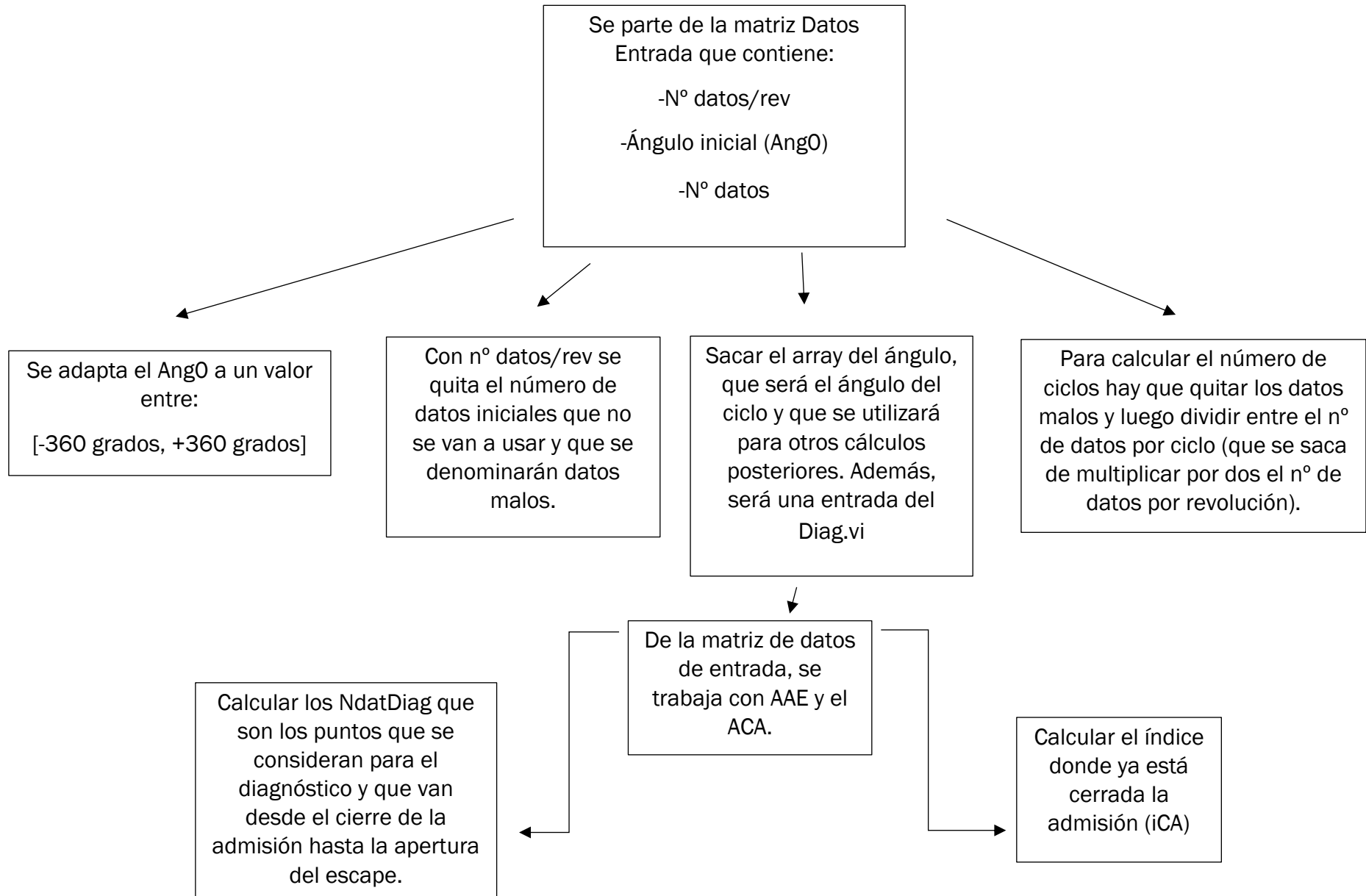


Figura 4.49. Diagrama descriptivo de Calculaángulos.VI.

Primero, hay que definir la matriz de datos que se llama *DatosEntrada*, que es un cluster de diez elementos que contiene variables relacionadas con los datos adquiridos por el osciloscopio:

-Ndata_rev: número de datos por cada revolución.

-Cal: calibración del transductor (viene en el manual del fabricante).

-Ang0: ángulo inicial referido respecto al ángulo dónde se da el PMS.

-ACA: ángulo en el cierre de la admisión.

-AAE: ángulo en la apertura del escape

-Ndatos: número de datos obtenidos. **Es el dato que ya se ha calculado.**

-NdatDiag: número de datos del diagnóstico.

-iCA: índice a partir del cuál se cierra la admisión.

-Nmalos : número de datos no válidos.

-Nciclos: número de ciclos.

El objetivo de este programa es ir calculando datos del cluster *DatosEntrada*. Para empezar a trabajar con ellos, y ayudándose de la función *Unbundle by Name*, se cogen, de todo ese cluster, tres datos que serán: *Ndata_rev*, *Ang0*, *Ndatos*. Esa función permite sacar elementos de un cluster por su nombre.

El *ang0* se va a adaptar a un valor que esté en el intervalo: $[-360 \leq x \leq 360]$ grados. La idea es que si el valor del ángulo es ≤ -360 grados se le suman 720 grados para incluirlo en ese intervalo. Pero si el valor es > -360 grados, puede estar en el intervalo o puede pasar por el límite superior y ser > 360 grados. Por lo tanto, se comprueba si es > 360 grados, y si fuese así, se restarían 720 grados, para quedar el valor dentro del intervalo comentado arriba. Una vez, hecho este cálculo, se vuelve a incluir con la función *Bundle by Name*, este elemento en el cluster *Datos* de entrada.

Usando el *Ndat_rev* se eliminan los datos que van desde el ángulo inicial (*Ang0*) hasta el ángulo 360 grados. Esos datos se denominarán datos malos.

Por otra parte se calcula el *Nciclos* usando los ciclos por revolución y el número de datos por ciclo, que será el número de datos por revolución multiplicado por dos. Al *Ndatos* hay que eliminarle esos datos malos calculados previamente.

Por otro lado, se genera un indicador, que es un array que será el ángulo del ciclo, con la ayuda de un bucle *While*.

Por último, se trabaja con *AAE* y *ACA*. Se resta *iAE* (que sería el índice de apertura del escape) menos el *iCA*, para así calcular el *NdatDiag* que es el número de datos del diagrama con el que se trabajará. Realmente lo que se hace es coger los datos de diagnóstico (*NdatDiag*), que serán aquellos que van desde el cierre de la admisión hasta la apertura del escape. Hay que recordar

que los cálculos se hacen en un sistema cerrado. No se quieren coger los datos con las válvulas abiertas. Como recordatorio se explica ligeramente el significado del avance en la apertura del escape y el retraso en el cierre de la admisión en la figura 4.50:

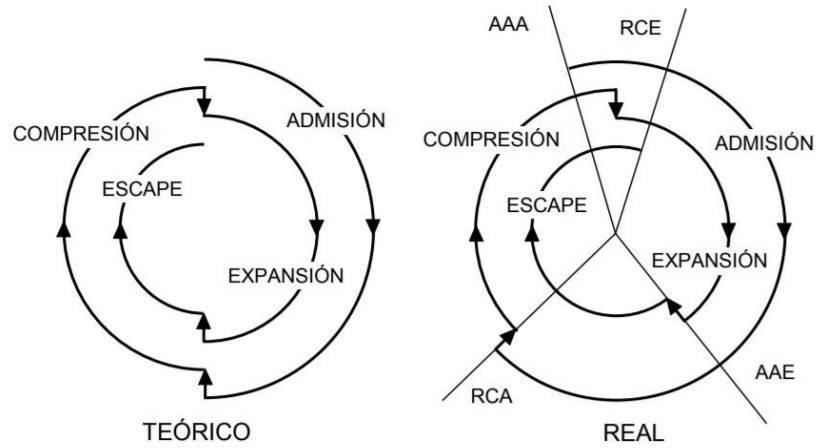


Figura 4.50. Diagrama de distribución en un MCI.

El motivo por el cuál se busca abrir antes y cerrar las válvulas después, respecto de los puntos muertos, es porque así se mejora el llenado del cilindro y las pérdidas por bombeo disminuyen.

4.4.5. Cogeapresionciclo.VI

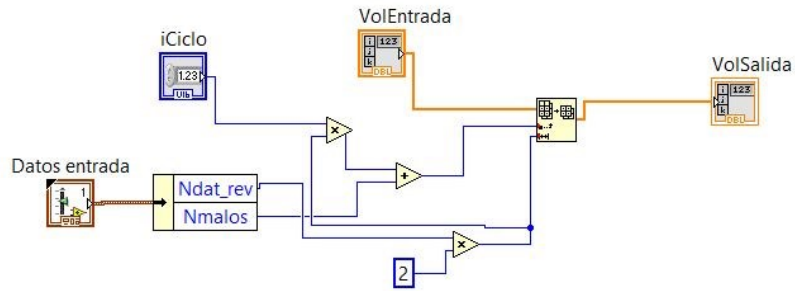


Figura 4.51. Diagrama de bloques de cogeapresionciclo.VI

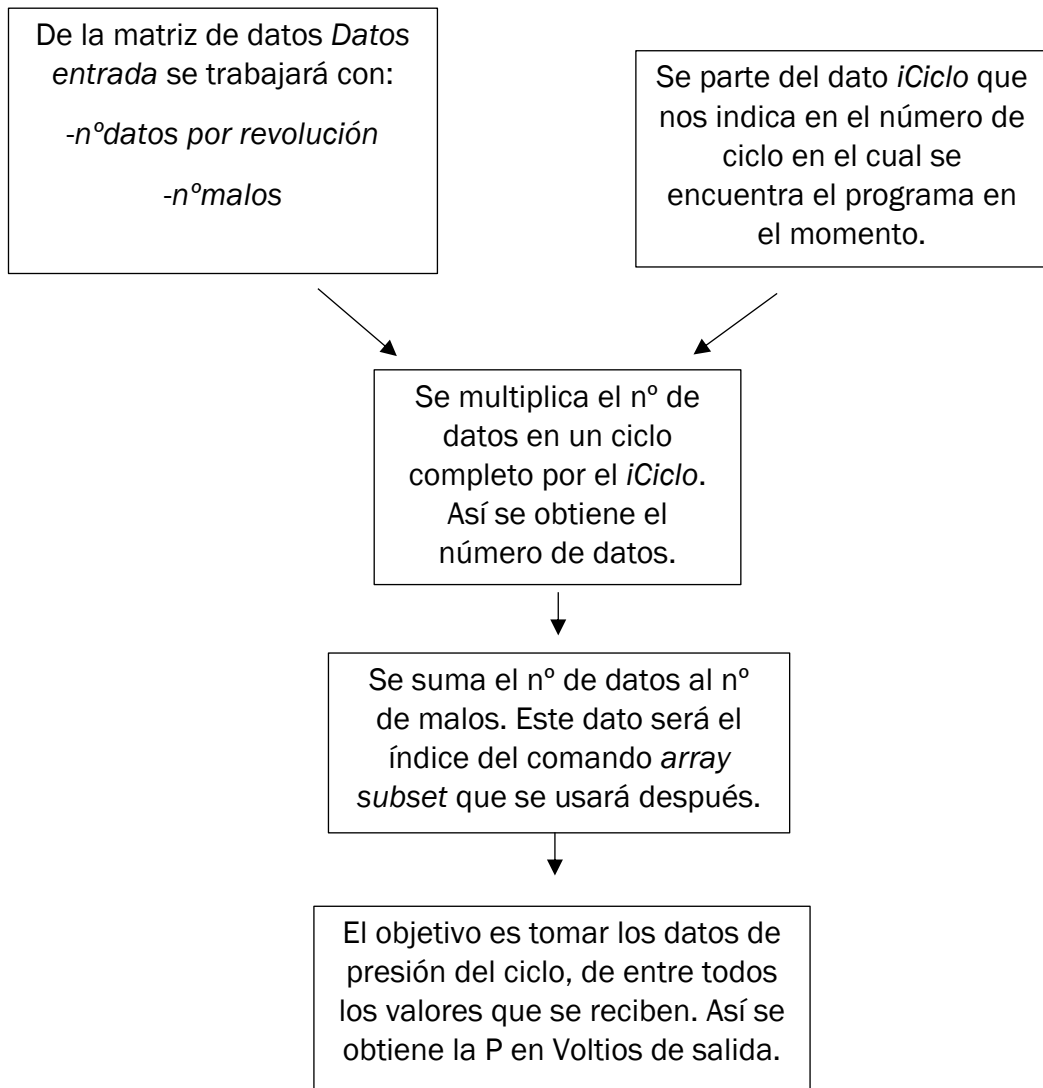


Figura 4.52. Diagrama descriptivo de Cogeapresionciclo.VI.

Este programa sirve para obtener la porción de datos del array de P en voltios obtenido en el Leeronda.VI del apartado 4.4.3.

Consiste en multiplicar el iCiclo que es el índice del ciclo en el que se encuentra el programa en cada momento por el número de datos de cada ciclo que será a su vez el número de datos en cada revolución(Ndat_rev) por dos. Así se obtiene el número de datos total en cada ciclo. A este dato, se le suma el número de datos malos.

Se utiliza la función Array Subset, en la cuál la entrada es la P en voltios ya calculada, el índice el dato calculado en el párrado anterior y la longitud es el número de datos en cada ciclo. Así se acotan, de la enorme cantidad de datos que viene del osciloscopio, los datos de presión del ciclo.

4.4.6. Presioncorregida.VI

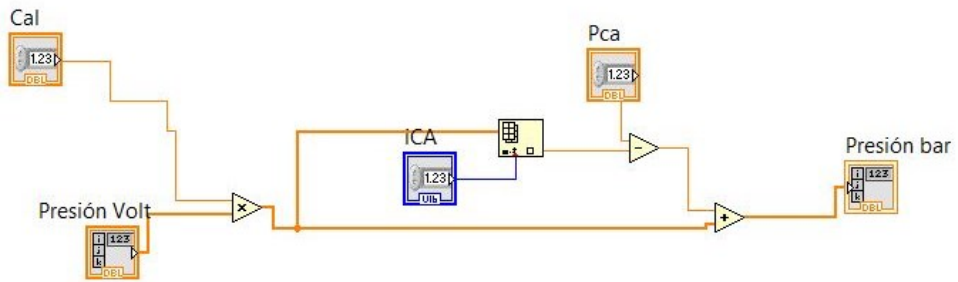


Figura 4.53. Diagrama de bloques de Presioncorregida.VI

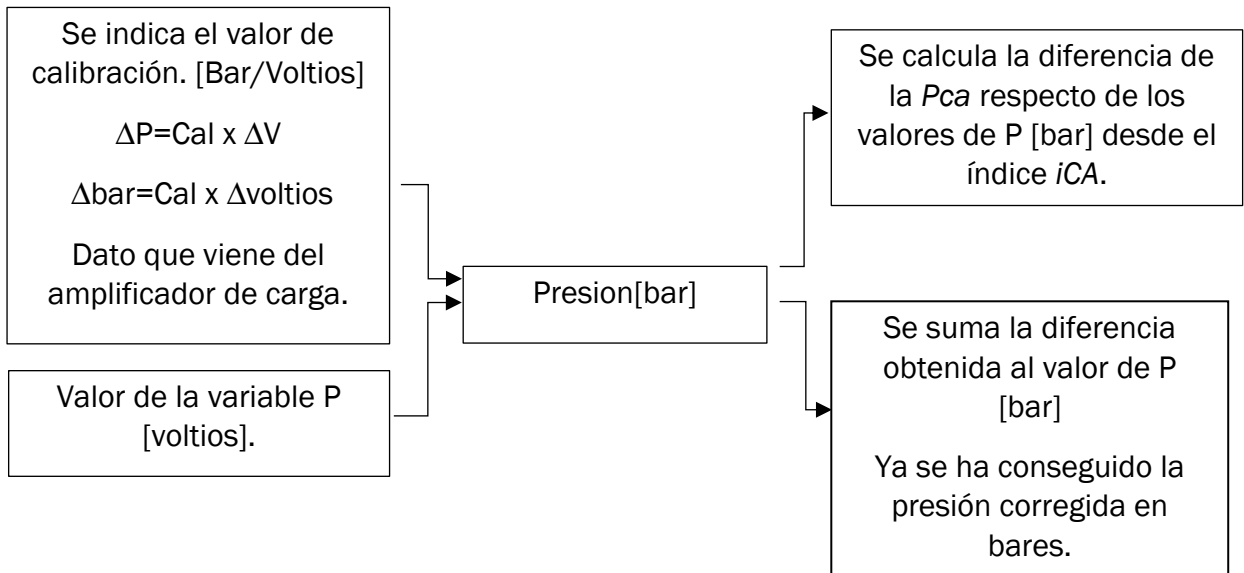


Figura 4.54. Diagrama descriptivo de Presioncorregida.VI.

El programa *Presioncorregida* permite corregir el valor de la presión en voltios que se ha obtenido en apartados previos.

La fórmula utiliza es la ecuación 4.1:

$$\Delta P = Cal \times \Delta V \quad [ecuación 4.1]$$

ΔP es el incremento de presión en bares.

ΔV es el incremento de presión en voltios.

Cal se refiere a la calibración del transductor, que es un dato del amplificador de carga y que viene en el manual del fabricante.

A continuación, usando la función *IndexArray* se toma una porción del array que es ese ΔP , tomando como índice *iCA*, es decir, se toman los datos de presión a partir del cierre de la admisión. La idea es que con la calibración, se pueda conseguir un valor como el *Pca* que como se dijo con anterioridad es un valor difícil de medir en el laboratorio. Por lo tanto, se calcula la diferencia entre *Pca* y los valores de presión, y esa diferencia es la que se suma posteriormente.

4.4.7. Diag.VI

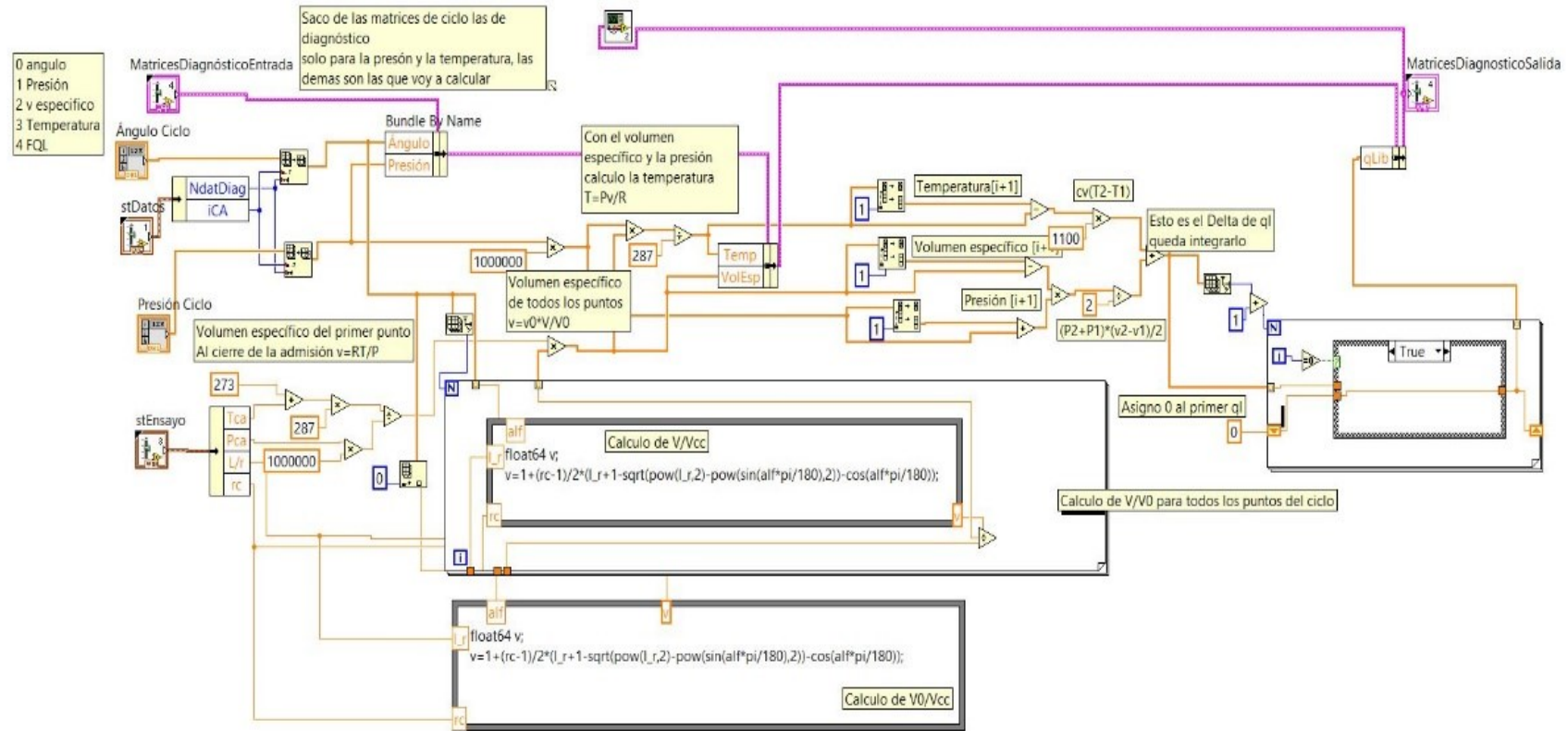


Figura 4.55. Diagrama de bloques de Diag.VI

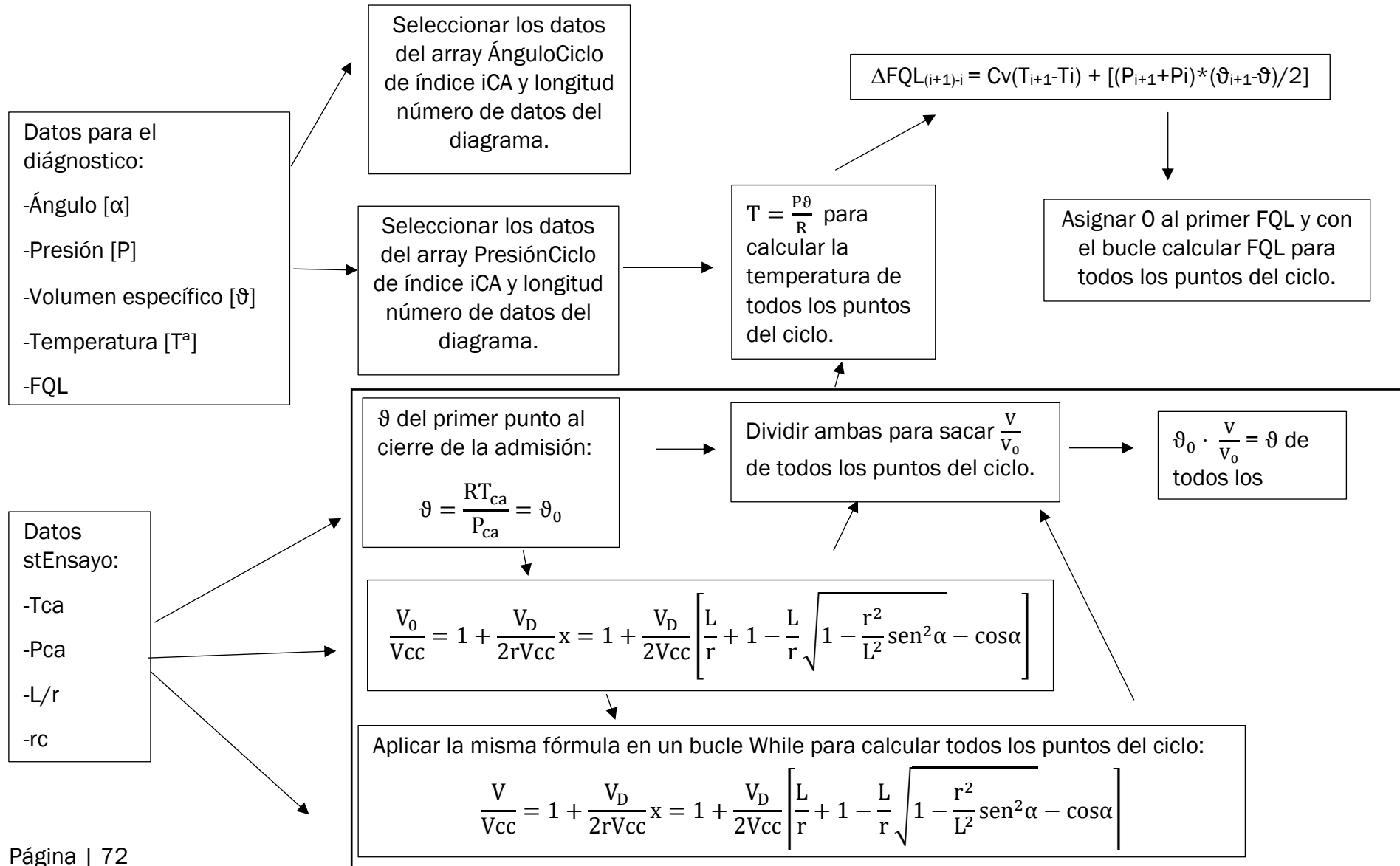


Figura 4.56. Diagrama descriptivo de Diag.VI.

El programa *Diag.VI* es en el que se van a realizar los cálculos del volumen específico y el calor liberado.

Las matrices diagnóstico de entrada contienen los siguientes datos:

- Ángulo (0)
- Presión (1)
- Volumen específico= ϑ (2)
- Temperatura (3)
- FQL (4)

Estos son los datos que se van a calcular

Primeramente, se coge la porción del array ángulo del ciclo que interesa, que será aquella que comienza en *iCA* y cuya longitud es igual al número de datos del diagrama. De esta forma ya tengo calculado los datos ángulo (0) de las matrices de diagnóstico de entrada. Se hace exactamente lo mismo con la *P* del ciclo.

Por lo tanto, falta calcular la temperatura, el volumen específico y la FQL. Para ello, se necesitará la matriz de datos *StEnsayo* que contiene:

- Tca*: temperatura en el cierre de la admisión.
- Pca*: presión en el cierre de la admisión.
- L/r* : esta relación viene explicada en el modelo de diagnóstico 2.2.
- r_c*: relación de compresión.

Se calcula el volumen específico del primer punto, del cierre de la admisión.

$$\vartheta = \frac{RT}{P} \quad [\text{ecuación 4.2}]$$

A continuación, con la ayuda de la función *Formula Node* se calculará la ecuación 2.37 hallada en el apartado de diagnóstico, que decía:

$$\vartheta(\alpha) = \vartheta(\alpha_0) \cdot \frac{V(\alpha)/V_{CC}}{V_0/V_{CC}} = \vartheta(\alpha_0) \cdot \frac{1 + \frac{r_c - 1}{2} \left[\frac{L}{r} + 1 - \sqrt{\frac{L^2}{r^2} \sin^2 \alpha - \cos \alpha} \right]}{1 + \frac{r_c - 1}{2} \left[\frac{L}{r} + 1 - \sqrt{\frac{L^2}{r^2} \sin^2 \alpha_0 - \cos \alpha_0} \right]} \quad [\text{ecuación 2.37}]$$

Tras aplicar esta fórmula, ya se ha obtenido el volumen específico en todos los puntos. Con ϑ , es fácil calcular *T* para todos los puntos con la ecuación de estado:

$$T = \frac{P \cdot \vartheta}{R} \quad [\text{ecuación 4.3}]$$

Por lo tanto, sólo queda calcular la FQL.

$$\Delta FQL_{(i+1)-i} = Cv(T_{i+1}-T_i) + [(P_{i+1}+P_i)*(\vartheta_{i+1}-\vartheta)/2] \quad [\text{ecuación 4.4}]$$

Hay que tener en cuenta que para calcular los puntos $i+1$ lo que se hace es utilizar la función *Split1D Array* que permite dividir un array en dos y coger ambas partes. Lo que se hace es dividirla en el índice 1. Así este array siempre va a ir un i por delante que el array sin modificar. Esto se hace tanto para ϑ , como para P como para T .

Ya está calculado ΔFQL . Se integra, asignando como valor inicial que $FQL=0$. Para ello basta con usar una *estructura Case* que permite asignar ese 0 al valor inicial.

Ya se ha conseguido el objetivo y se ha calculado tanto la FQL, como la P y el volumen específico para todos los puntos.

4.5. Gráficas de la P- α y el calor liberado

En la figura 4.57, que corresponde a la gráfica P- α , se puede ver el drástico aumento de la P al producirse la combustión del motor. La campana que se produce es debida a la compresión, combustión y expansión. En el instante en el que se ha realizado la captura, alcanza un valor aproximado de 25 bares.

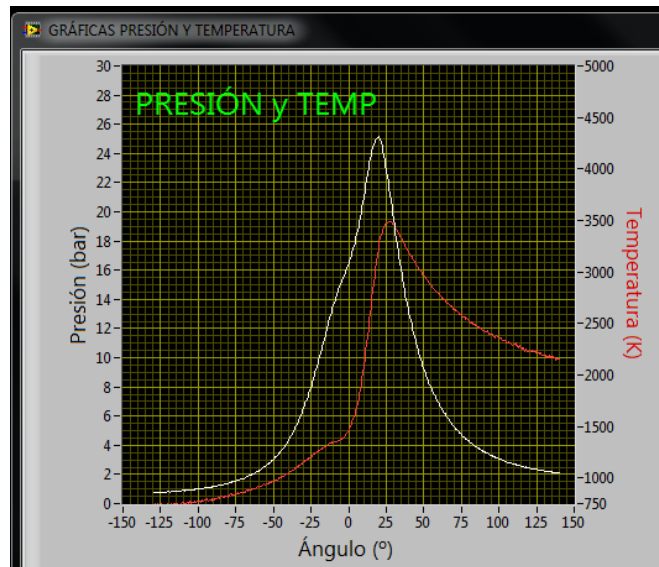


Figura 4.57. Gráfica experimental P- α

La gráfica 4.58 muestra el calor liberado en función del ángulo de giro del cigüeñal (en blanco). Hay que diferenciar la fracción de calor liberado, FQL, que es el calor liberado entre el calor total, respecto al q_l , que es únicamente la cantidad de calor liberado. Es el q_l lo que se puede ver representado en la gráfica. Logicamente el calor comienza a liberarse a medida que se produce la combustión.

La derivada del calor liberado (en rojo) da una idea de la velocidad a la que se desarrolla la combustión y durante la compresión y la expansión vale prácticamente cero. Por lo tanto, la campana representa la zona donde se produce la combustión.

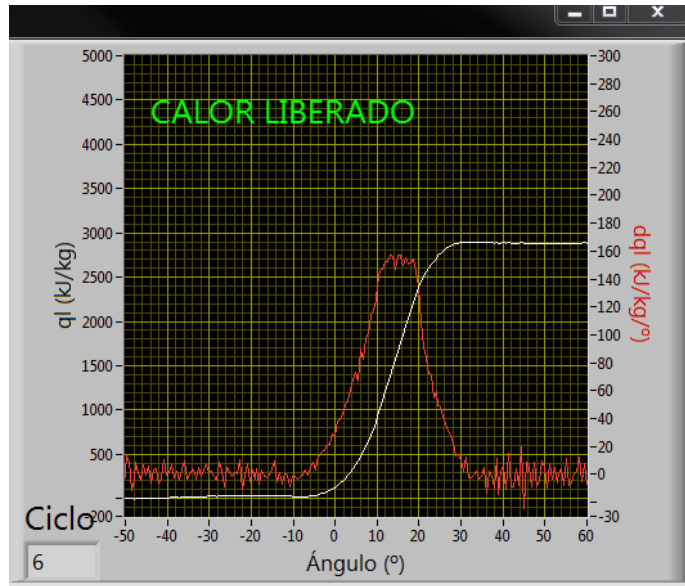


Figura 4.58. Gráfica experimental obtenida

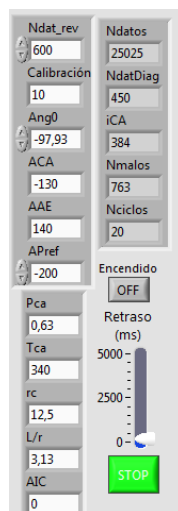


Figura 4.59. Controles del programa.

En la figura 4.59 se pueden ver los controles del programa. Ya se ha mencionado a lo largo del apartado 2.2. que las variables P_{ca} , T_{ca} se modifican a gusto del usuario para crear unos datos de ensayo coherentes. Tras varios intentos, se ha llegado a la conclusión de que los valores de la figura son los idóneos para el ensayo realizado.

En la figura 4.60 se puede ver la pantalla del programa completa. Se puede observar que tiene más gráficas y funciones de las mencionadas y eso es debido a que el programa se ha ido actualizando y mejorando con el paso del tiempo durante este curso. En este documento no se van a tratar.

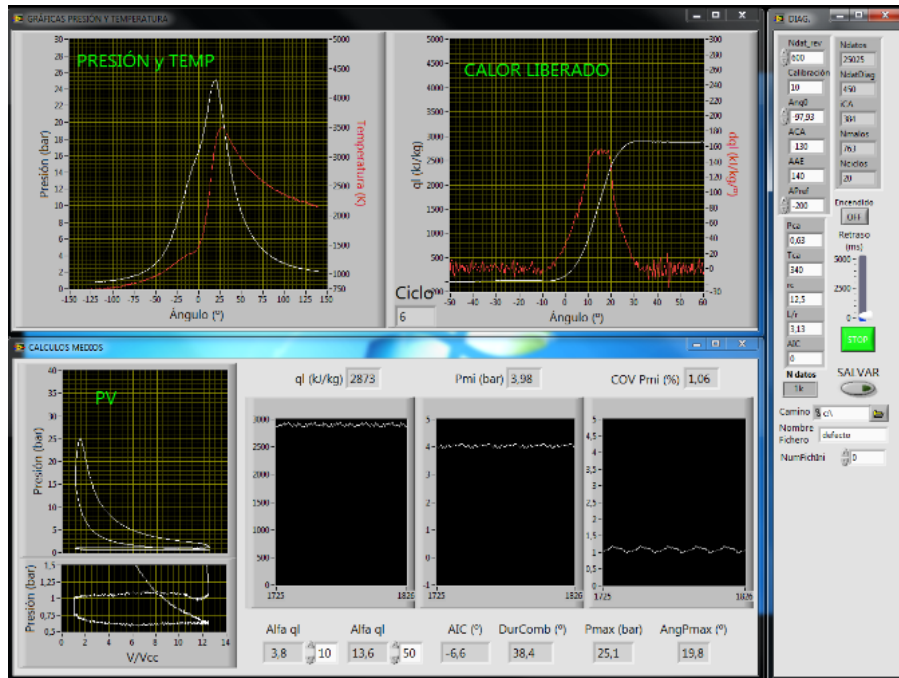


Figura 4.60. Pantalla completa del programa.

5. CONCLUSIONES

Para finalizar, se presentan las conclusiones extraídas de la realización del proyecto.

A nivel profesional, ha sido una experiencia enriquecedora ya que el trabajo ha permitido un desarrollo en diferentes áreas, como los motores de combustión interna alternativos, la comunicación y las posibilidades de trabajar con osciloscopios y por supuesto, el desarrollo en la programación gráfica de LV.

El mayor reto personal ha sido familiarizarse con un lenguaje de programación partiendo de cero y aprender acerca sobre las comunicaciones con el osciloscopio y su funcionamiento. Aprender algo de cero siempre supone un mayor esfuerzo, aunque también aporta una mayor satisfacción.

En cuanto a las conclusiones que se pueden extraer del trabajo:

-Se ha logrado cumplir el objetivo principal y se ha creado una herramienta de diagnóstico que permite medir la variable P a lo largo del ciclo en tiempo real. Ha sido un éxito ya que se puede ver la evolución instantánea de esos datos en las gráficas finales del osciloscopio.

-Se ha conseguido crear un sistema capaz de medir esa P de forma fiable. Un banco de ensayos, los sistemas de acondicionamiento de la señal, un osciloscopio y la programación en LV han permitido transmitir esa información al usuario.

-Se ha logrado con éxito controlar el osciloscopio Yokogawa DL750 de forma remota a través del ordenador y con la ayuda de una conexión USB. Esto permite dejar el osciloscopio conectado en la instalación sin moverlo, y poder seguir desarrollando el trabajo desde cualquier lugar con un PC.

-También hay que destacar haber conseguido con éxito el desarrollo del programa de LV. Tal y como hemos visto en el apartado 4.4, se ha conseguido llegar al objetivo y calcular otras variables a raíz de la P, como el volumen específico y el calor liberado.

-El objetivo final era conseguir que este documento sirviese de referencia a otros alumnos en el futuro. Se ha logrado un documento que permitirá a alumnos en el futuro tenerlo como base para realizar otros TFG o TFM. Una vez entendidos los pasos tomados en este trabajo, se podrá seguir desarrollando el programa en LV y así poder calcular muchas otras variables en el futuro para hacer un estudio en profundidad de las características del motor del laboratorio, en concreto, de su combustión. También permitirá a alumnos de la universidad que estudien asignaturas relacionadas en un futuro, ver de una forma real como puede ser el análisis del motor y así poder aplicar los estudios teóricos en un caso práctico.

6. BIBLIOGRAFÍA

- [1] Manual osciloscopio Yokogawa DL 750 (versión inglesa). 3ª edición. Diciembre 2005
- [2] Curso: Diagnóstico a partir de la medida de presión en cámara. *Cidaut*.
- [3] Curso: LabVIEW Core 1. Versión agosto 2012. National Instruments.
- [4] Apuntes de la asignatura de motores de combustión interna alternativos (versión 15-16). Universidad de Valladolid. Blanca Giménez Olavarria, Andrés Melgar Bachiller.
- [5] Métodos experimentales en Ingeniería Térmica. Andrés Melgar, Jose Manuel Villafruela. Universidad de Valladolid.
- [6] LabVIEW Programación para Sistemas de Instrumentación. Joaquín del Río Fernández. Grupo editorial Garceta. 1ª edición 2011. ISBN 978-84-92812-48-4
- [7] Motores de combustión interna alternativos. Profesor F. Payri y Prof. J.M. Desantes. Editorial Reverte. 2014.
- [8] Práctica de máquinas y motores térmicos, Indicador de ciclo en un motor alternativo de combustión interna, A. Lecuona, P. Rodríguez, C. Vereda. Universidad Carlos III de Madrid.
- [9] Presentación sobre el funcionamiento del osciloscopio de la Fundación San Valero

