UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

DESARROLLO DE UN EMULADOR EN TIEMPO REAL DE UN MOTOR DIÉSEL SOBREALIMENTADO DE AUTOMOCIÓN

AUTORA: AMANDA IGLESIAS MORENO

TUTOR: CARLOS GUARDIOLA GARCÍA

COTUTOR: PAU BARES MORENO

Curso Académico: 2013-14

A mis padres las personas
que más quiero

Quisiera agradecer a todas las personas que me han ayudado en el proyecto.
Gracias a mi tutor, Carlos Guardiola, por darme la oportunidad de trabajar en este proyecto.
A todos los chicos del departamento, por conseguir hacer el trabajo más ameno.
A Pau por la paciencia que ha demostrado y por su ayuda a lo largo de todo el proyecto.
Pero sobretodo agradecer a mis padres, no solo en estos meses sino durante toda la carrera, todo el apoyo que me han dado, por compartir mis alegrías, y por su ayuda en todos los momentos malos.

ÍNDICE

Índice de figuras

Índice de tablas

Capítulo.1 Introducción y Objetivos.

En el pasado los ingenieros se dedicaban a estudiar y realizar pruebas con el objetivo de obtener datos y conclusiones. Las líneas de investigación en la industria automovilística, antaño principalmente experimentales, han cambiado en la actualidad, ganando terreno el campo teórico y con él los modelos matemáticos que simulan el comportamiento de los motores.

El presente Trabajo Fin de Grado tiene por objeto el desarrollo de un sistema de simulación de un motor basado en distintos lenguajes y programados en un mismo entorno en tiempo real: se utiliza un modelo Simulink de valores medios que calcula presiones, temperaturas y caudales másicos en los diferentes puntos del motor y un modelo labVIEW encargado del cálculo de la presión en la cámara de combustión. Finalmente se generarán señales analógicas, disponiendo así de señales similares a las que se obtendrían mediante sensores situados en el motor.

El prototipado virtual generado, ejecutado en tiempo real, sustituiría al motor en los ensayos experimentales, pudiendo interactuar con un hardware bajo test por ejemplo la Unidad de Control Electrónica del automóvil. Conectar el dispositivo a un sistema simulado es en ocasiones la única alternativa al no disponer del motor real. En otras ocasiones este tipo de simulación se utiliza para acelerar el desarrollo del producto y supone una rentabilidad económica considerable respecto a la experimentación con un motor real.

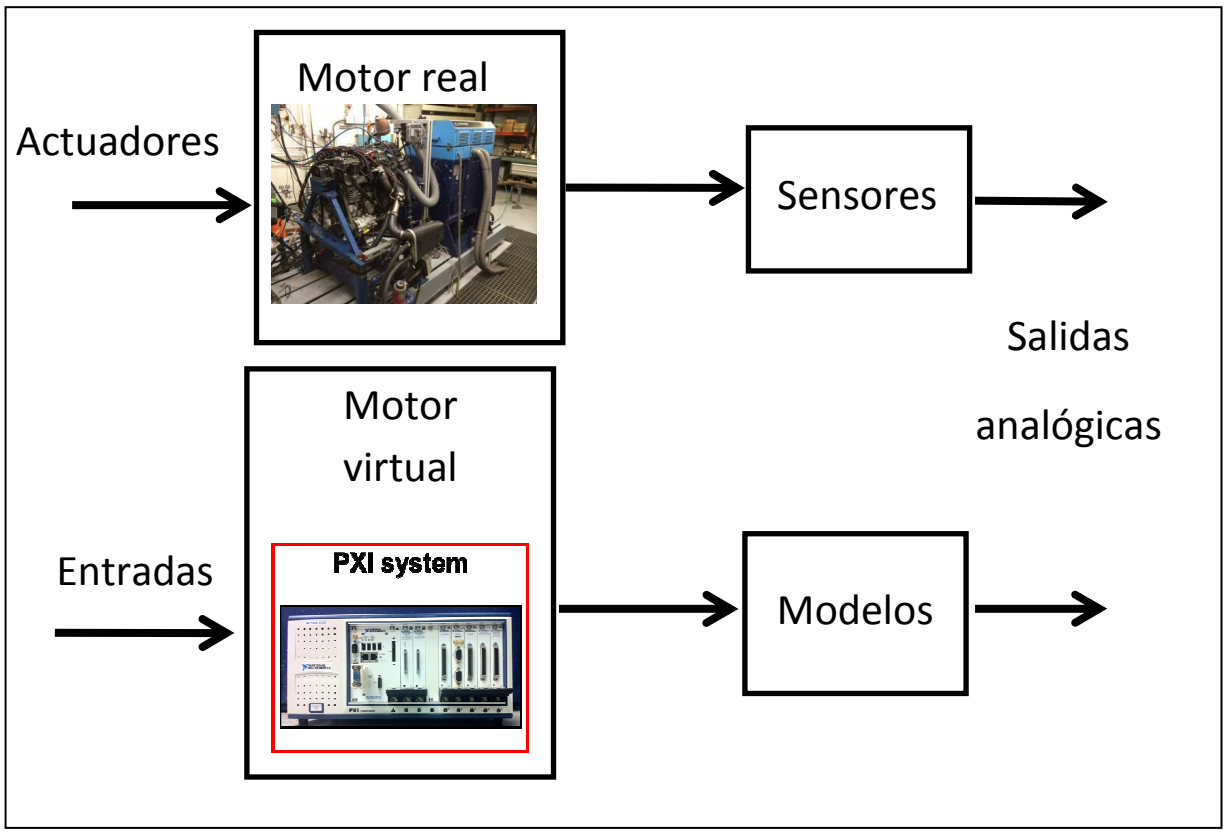


Figura 1: Sustitución de un motor real por el motor virtual

1.1 Antecedentes

El Instituto Universitario CMT-Motores Térmicos situado en el campus de Vera de la Universitat Politècnica de València, combina a diferencia de otros centros universitarios, la docencia con la investigación industrial y la colaboración con numerosas empresas multinacionales: BMW, Ford, Citroen, Renault, Peugeot, Aprilia, Fiat, entre otras. Más de 750 proyectos de final de carrera se han realizado, complementando el estudio de las líneas de investigación del centro, reducción del consumo de combustible, del ruido en los motores, optimización de la renovación de la carga, aerodinámica del vehículo, sobrealimentación, control de los contaminantes, optimización de la inyección. El objetivo del departamento es mejorar el comportamiento de los motores utilizando la simulación y los ensayos experimentales. Para ello el departamento dispone de más de 5500 m2 de laboratorios equipados con numerosos bancos de ensayos y equipos, entre ellos los destinados al control en tiempo real, PXI, dSpace y a la medida de emisiones.

Figura 2: Instituto Universitario CMT-Motores Térmicos

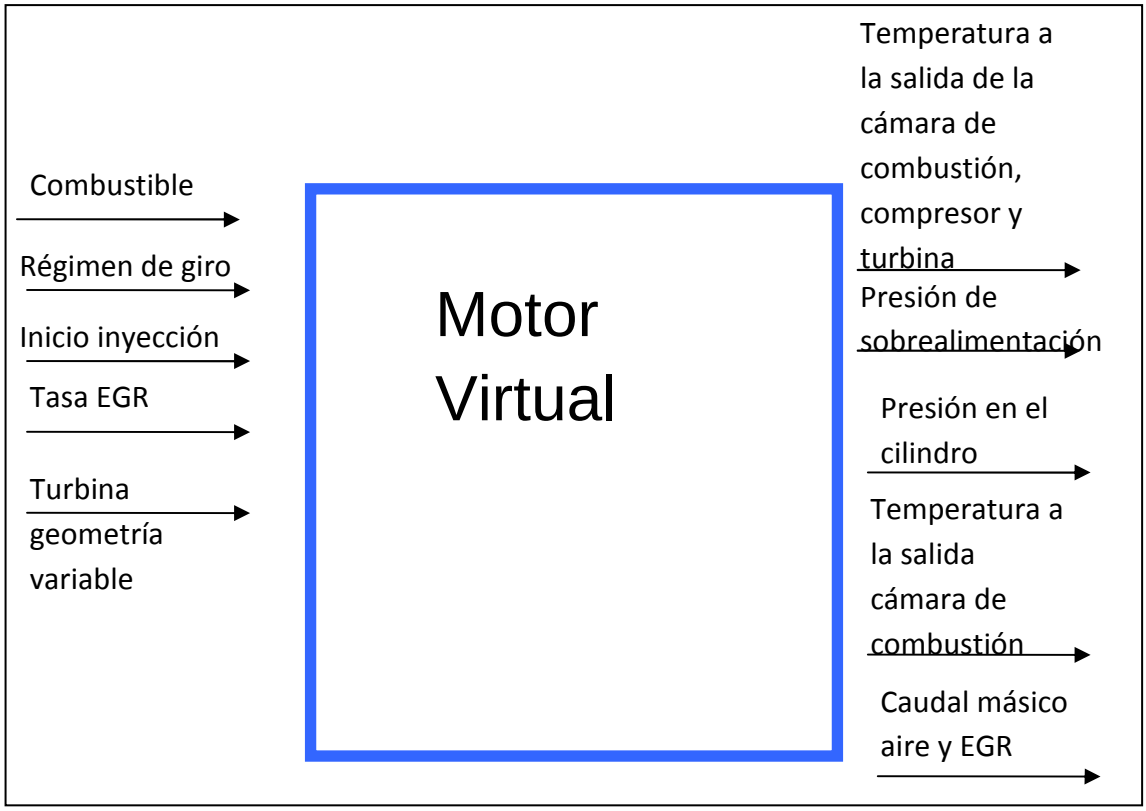
La idea del proyecto fue formada siguiendo las líneas de la industria automovilística actual, en la cual la simulación cada vez tiene más fuerza como etapa previa al diseño de motores: realizar un sistema de simulación de un motor integrando varios modelos matemáticos. La idea surge en el departamento de control del Instituto Universitario CMT-Motores Térmicos, buscando crear un motor virtual basado en la integración de modelos procedentes de entornos de programación diferentes. Las principales motivaciones eran la flexibilidad que proporciona el poder ejecutar distintos lenguajes de programación conjuntamente, facilitando futuras ampliaciones y el ahorro económico que supone el uso de un simulador en tiempo real frente a los costes de la experimentación con motores reales. La integración de los modelos se ha llevado a cabo con el uso de un software, Ni VeriStand, no empleado anteriormente en el departamento. Ese era el principal reto del proyecto, trabajando con un Software nuevo, explorar las posibilidades que este ofrece y lograr un sistema de simulación que pueda sustituir al motor real, y que en el futuro pueda ser fácilmente ampliado, así como utilizado en el centro para realizar investigaciones, alternativamente o paralelamente a la simulación en banco de ensayo de motores.

1.2 Objetivo

El objetivo del proyecto es realizar un motor virtual mediante la utilización conjunta de dos modelos efectuados en diferentes entornos de programación y basados en diferentes principios de funcionamiento. En primer lugar se simulara un modelo Simulink de valores medios, encargado del cálculo de presiones temperaturas caudales en diferentes partes del motor, este modelo se implementara conjuntamente con un modelo labVIEW de calculo de la presión en cámara de combustión. La simulación conjunta de ambos modelos se conseguirá gracias al uso de un software para pruebas en tiempo real, NI VeriStand.

Los modelos se simularan en tiempo real proporcionando salidas analógicas de las variables simuladas. Estas salidas analógicas podrían ser enviadas a la ECU de un automóvil para programarla sin necesidad de tener que recurrir al ensayo experimental.

El siguiente esquema muestra las salidas y entradas del sistema de simulación generado.



1.3 Estructura

La memoria está dividida en cinco partes para facilitar la comprensión al lector

- En la primera parte se realiza una breve introducción del proyecto, se explican los objetivos, las motivaciones y el ambiente de trabajo en el que se ha desarrollado.
- En el segundo capítulo está dedicado a los motores virtuales, resaltando las ventajas, inconvenientes, aplicaciones y requerimientos de los mismos.
- El tercer capítulo está dedicado al software y hardware utilizados, los programas empleados son complejos y con infinidad de posibilidades, el capítulo se centra en todos los aspectos que proporcionan los programas utilizados que han sido útiles en el desarrollo del proyecto y que es necesario dominar para entender la ejecución del mismo.
- En el cuarto capítulo se explican los dos modelos de simulación implementados, la comunicación de datos entre ambos y la generación de las señales analógicas de voltaje.
- El quinto capitulo está dedicado a los resultados del proyecto.

Finalmente se proporcionan una serie de anexos, algunos de ellos son simples documentos teóricos de compatibilidades de software, otros pequeñas guías de usuario, y por último aquellos más útiles, documentos que muestran al lector posibilidades que brindan el software y hardware empleados, que si bien han sido exploradas durante la realización del proyecto no han sido finalmente implementadas en la solución final.

Capítulo 2. Motor Virtual

2.1 Introducción a los motores virtuales

Los Motores virtuales se basan en modelos matemáticos que simulan el comportamiento de las diferentes partes del motor. Su implementación, surgió paralelamente al desarrollo de las tecnologías de computación, instrumentación y software de control. Las primeras empresas que apostaron por el desarrollo de modelos surgieron aproximadamente hace 20 años, en la actualidad, numerosas empresas se dedican a la implementación de modelos basados en motores, el software que desarrollan es comercializado a las principales compañías encargadas del desarrollo, diseño y fabricación de motores. Wirth research, OPAL-RT, Gamma Technologies, AVL son pioneras en el desarrollo de software basado en modelos, lideres en prototipado rápido y en hardware-in-the-loop test. La industria se encuentra actualmente en auge gracias a las ventajas que proporciona un motor virtual, como alternativa y complemento a la experimentación con motores reales.

2.1.1 Ventajas y desventajas

El elevado coste económico de las plataformas experimentales requiere alternativas para que el estudio de los motores resulte rentable. El coste no solo incluye la adquisición de los motores sino también el mantenimiento y ejecución de los ensayos. La simulación requiere una inversión inicial en hardware y software pero no un gasto económico continuado a medida que se desarrolla el producto o la investigación.

A la rentabilidad económica habría que añadir flexibilidad que supone trabajar con motores virtuales. La simulación permite acelerar el comportamiento, permitiendo trabajar con fenómenos cuya dinámica es muy lenta, pero también permite ralentizar la simulación para estudiar los sistemas con mayor detalle. Además posibilita la obtención de resultados que no son medibles experimentalmente con el nivel actual de tecnología. Respecto al tiempo de ejecución, el motor virtual puede ser también empleado en simulaciones estrictamente en tiempo real, ejecutándose los modelos a la misma velocidad que el sistema físico real.

Otra ventaja de los motores virtuales es la seguridad en el trabajo dado los riesgos que supone para los investigadores trabajar en ciertos entornos experimentales. Los riesgos son los derivados del uso de combustibles, de las temperaturas y presiones elevadas que se alcanzan durante la combustión, como de las sustancias contaminantes producidas.

En cuanto a las desventajas de la simulación, cabe destacar la necesidad de conocimiento y comprensión de los factores implicados en el proceso, así como del software a emplear. Por otra parte se ha de recurrir a la experimentación para validar los resultados, y comprobar que no ha habido errores en la realización del modelo.

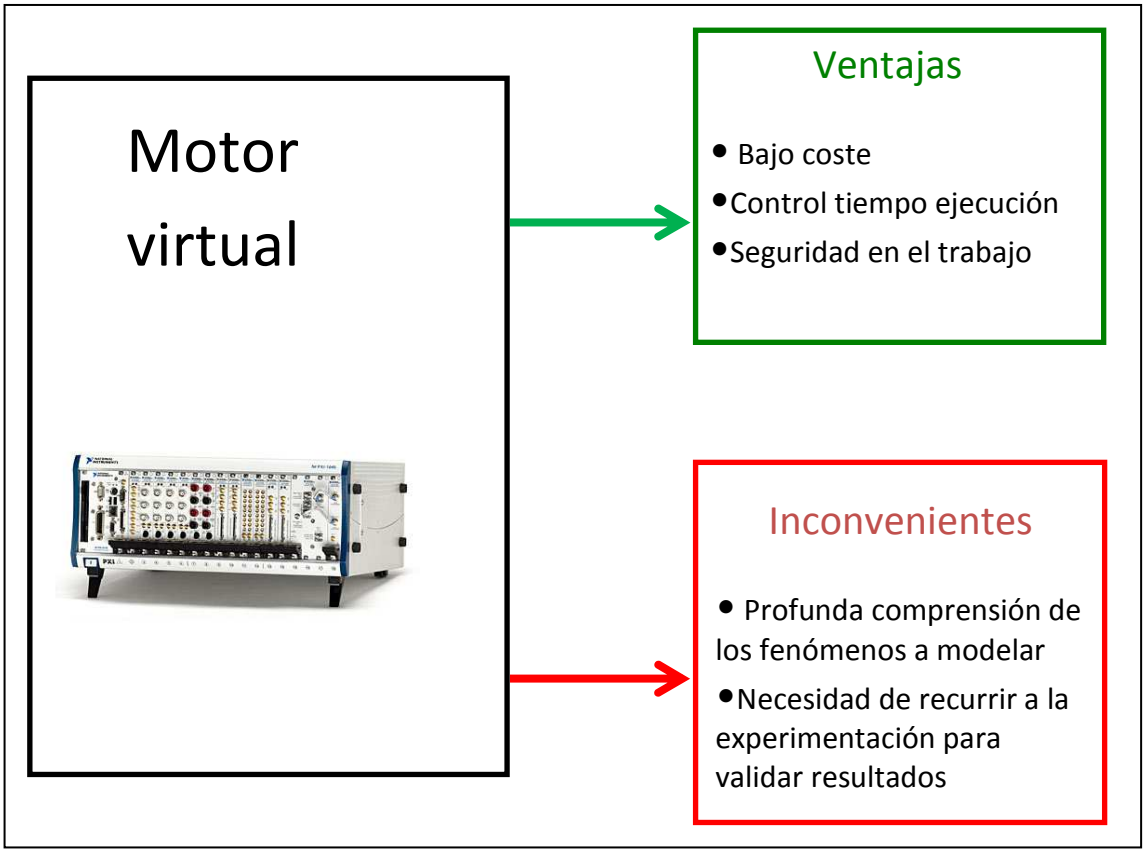


Figura3. Ventajas y desventajas motor virtual

2.1.2 Aplicaciones

Las aplicaciones del motor virtual creado son numerosas: estudios con el objetivo de reducir emisiones contaminantes, verificar que un motor cumplirá con los ciclos de certificación antes de disponer del prototipo del vehículo, evaluación del consumo del

motor y de la eficiencia, estudios de los estados estáticos como dinámicos de multitud de variables en cualquier entorno de conducción.

Además gracias a las señales analógicas generadas la principal aplicación del motor sería la simulación hardware-in-the-loop: los estímulos eléctricos generados pueden ser las señales para que un dispositivo bajo test, por ejemplo la unidad de control electrónica del automóvil, funcione de la misma manera que si estuviera conectada a un entorno de experimentación. Se simulan diversas señales por ejemplo la temperatura de los gases de escape de un motor y se prueba si la ECU puede detectarla e informar en el caso de que se produzca un valor excesivo. Un automóvil tiene numerosas ECU, Hardware-in-the-loop permite comprobar el correcto funcionamiento de las mismas.

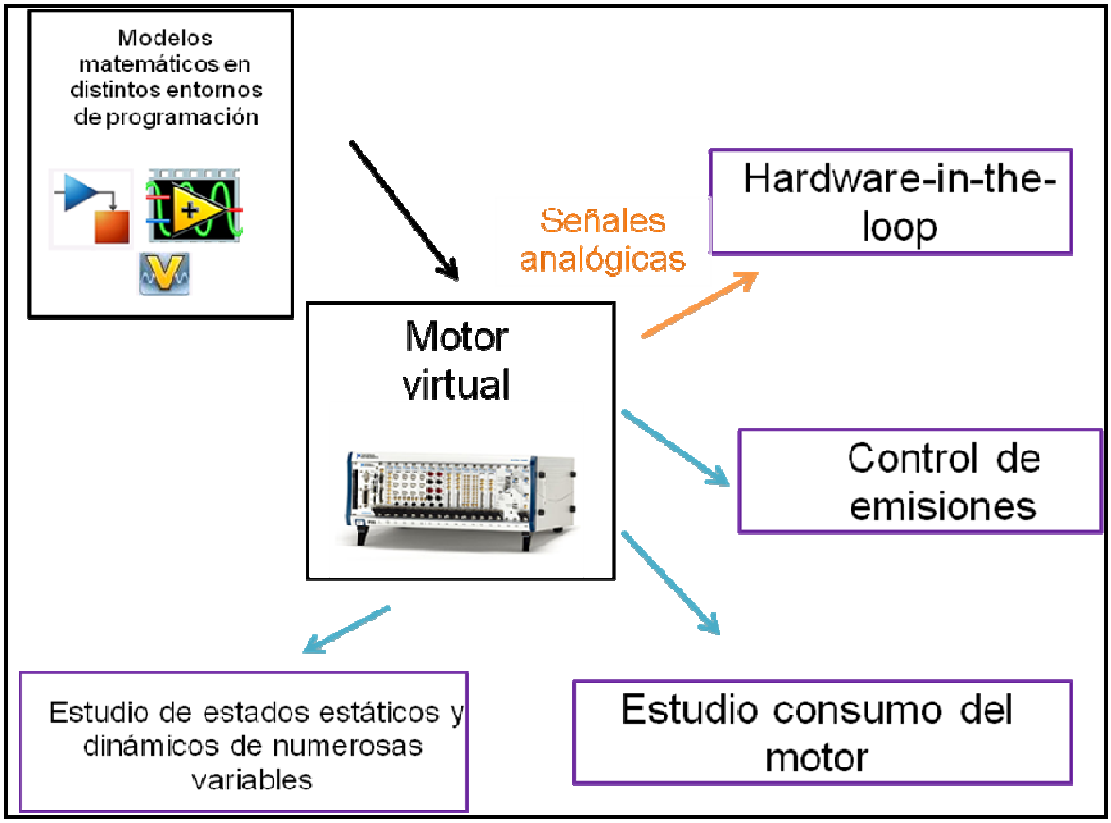


Figura4. Aplicaciones motor virtual

2.1.3 Tiempo Real

En la simulación en tiempo real el procesador en el que se ejecutan los modelos funciona a la misma velocidad que el sistema físico que simula. Las aplicaciones de este tipo de simulación, son entre otras, el diseño de sistemas, el estudio del comportamiento en regímenes dinámicos y la simulación Hardware-in-the-loop, está se basa en engañar al sistema embebido, como si estuviera operando con entradas del mundo real. Es vital que el simulador genere las señales a la misma velocidad que el sistema físico que simula, garantizando que la respuesta del dispositivo bajo test sea la misma que si estuviera conectado al entorno real.

Distinto software permite la ejecución de programas en tiempo real: LabVIEW, Simulink, código C. El código generado se implementará en diferentes plataformas NI-PXI, dSPACE, xPC Target, CompactRIO. Cada procesador RT debe ser programado en un lenguaje distinto, sin embargo entornos como VeriStand permiten la comunicación entre distintos lenguajes de programación (Simulink, labVIEW, C) para ser compilados en un mismo entorno RT.

2.2 Señales Simuladas en el motor virtual creado

En el motor virtual creado se simularán múltiples señales mediante el uso de modelos Simulink y labVIEW generando posteriormente salidas analógicas proporcionales a las magnitudes simuladas. Las señales implementadas son la temperatura en cuatro puntos del motor: entrada y salida de la cámara de combustión, salida del compresor y turbina, la presión a la entrada de la cámara de combustión, los caudales másicos de EGR y oxígeno y la presión en el interior de la cámara de combustión. Las señales simuladas poseen características diferentes, y por tanto la simulación de las mismas se tiene que hacer de acuerdo a las mismas para obtener la información necesaria para representarlas de manera óptima. En la realidad, en los ensayos experimentales, los sensores trasmiten información de las magnitudes físicas a partir de la variación del voltaje que proporcionan. En el proyecto si bien los modelos matemáticos sustituyen al motor real, siguiendo ese paralelismo, la generación de señales analógicas de las variables simuladas sustituiría las señales que proporcionarían sensores situados en el motor.

2.2.1 Temperatura a la entrada de la cámara de combustión.

En los motores de combustión interna la temperatura a la entrada de la cámara de combustión no debe ser elevada, pues se produciría una disminución del rendimiento volumétrico del motor. La sobrealimentación debe ir acompañada del enfriamiento del aire antes de entrar al cilindro mediante el uso de un intercooler. Este parámetro también influye en las tensiones térmicas del motor ya que la temperatura máxima durante la combustión depende de la temperatura de admisión. En los motores se utiliza un sensor para controlar está temperatura situado generalmente en el tubo de salida del intercooler. La ECU lo utiliza para comprobar la racionalidad de las medidas comparándolo con el CTS (cooland temperature sensor) ya que ambos deberían proporcionar la misma tensión de salida en un motor frío. Cuando el sensor esta frío la señal de tensión que proporciona es alta y esta disminuirá a medida que se calienta debido a la disminución de la resistencia del mismo.

2.2.2 Temperatura a la salida de la cámara de combustión, compresor y turbina.

La temperatura a la salida de la cámara de combustión debe ser medida lo más cerca del colector de escape antes del turbocompresor. Es importante controlarla pues si es excesiva puede provocar daños permanentes en el motor. Los motores diésel llevan sensores con el objetivo de controlar dicha temperatura. También disponen de sensores para controlar las temperaturas de salida del compresor y de la turbina.

2.2.3 Presión a la entrada de la cámara de combustión.

La presión a la entrada de la cámara es proporcional a la cantidad de aire introducido en el motor .Las presiones máximas que se pueden alcanzar en un motor diésel están limitadas y dependen entre otros factores de la presión a la entrada de la cámara de combustión. La presión también es un indicador de la carga del motor, en los motores diésel se utilizan sensores para medir la presión absoluta en el tubo de admisión, entre el intercooler y el motor, respecto a un vacío de referencia y no respecto a la presión atmosférica. Proporciona una indicación directa de la carga del motor y por tanto del combustible que es necesario inyectar para que el motor funcione eficientemente.

2.2.4 Caudal másico de aire

Otra señal implementada es el caudal de aire a la salida del intercooler. Generalmente se utilizan sensores con el objetivo de medir la cantidad de aire aspirado por el motor en cada instante y por tanto la ECU en base a la indicación de este sensor modifica el tiempo de inyección. La ventaja del sensor es que proporciona directamente la cantidad de aire, no obteniéndose está a través de cálculos a partir de la información proporcionada por otros sensores. Este tipo de sensores es utilizado tanto en motores de encendido provocado como en motores diésel. En estos últimos la función del mismo es calcular la cantidad de recirculación de gases de escape para conseguir bajas emisiones de NOx y la cantidad de combustible inyectado.

2.2.5 Caudal másico de EGR

La recirculación de gases de escape es una técnica empleada con el objetivo de reducir los contaminantes emitidos por el motor. Los motores llevan sensores, incorporados en la válvula de EGR, con el objetivo de controlar el caudal recirculado.

2.2.6 Presión en la cámara de combustión.

La última señal simulada, es la presión en la cámara de combustión, el número de muestras requeridas, a lo largo del tiempo, para conocer con detalle la misma es mucho mayor que en las señales anteriores. Generalmente se utilizan sensores, de material piezoeléctrico, con el objetivo de controlar la presión en cámara, garantizando no sobrepasar las tensiones mecánicas admisibles en el motor.

Todas las señales generadas serán convenientemente escaladas, en función del rango de salida de la tarjeta de adquisición de datos, para poder ser enviadas como salidas analógicas de voltaje sin que se produzca saturación en el envío, y garantizando que las señales se puedan reconocer adecuadamente. La ECU se calibrará de acuerdo a las funciones de conversión utilizadas.

Capítulo 3. Software y Hardware

3.1 Software utilizado

En el proyecto realizado se ha empleado Simulink para la simulación de un modelo matemático de un motor diésel sobrealimentado .Posteriormente se ha compilado en un archivo .dll utilizando Real Time Workshop y Visual Studio Profesional. El modelo compilado se ha integrado conjuntamente con otro modelo creado en LabVIEW, que simula la presión en la cámara de combustión. La integración de ambos modelos se ha realizado mediante el uso de Ni VeriStand, consiguiendo la ejecución conjunta de ambos a pesar de proceder de entornos de programación diferentes.

3.1.1 Visual Studio Profesional

Visual Studio Profesional es el programa utilizado para lograr la compilación del modelo Simulink en un archivo .dll. Se trata de un entorno de desarrollo integrado para Windows que soporta diversos lenguajes de programación como C++, C, Java, Python, Ruby, entre otros, al igual que entornos de desarrollo web.

3.1.2 Simulink

En el proyecto, Simulink se empleará para realizar los cambios de código necesarios para que un modelo matemático de un motor pueda ser compilado a código C ,utilizando la *toolbox* Simulink Coder (Real Time Workshop). Simulink es una herramienta que permite construir y simular modelos matemáticos interconectados entre si generando un modelo más complejo, o sistema. Dispone de una interfaz de usuario gráfica para elaborar los modelos como diagramas de bloques. Los modelos simulados pueden ser continuos, representados por ecuaciones con variables continuas en el tiempo, o discretos. En Simulink gracias al concepto subsistema se puede establecer una jerarquía en los modelos, un subsistema contiene en su interior una serie de bloques que conforman un modelo. Mediante el programa Real Time Workshop se pueden traducir modelos Simulink a archivos ejecutables en un hardware concreto. Entre las aplicaciones del RTW se encuentra, el procesado de señales en

tiempo real, las simulaciones a alta velocidad, las simulaciones HIL, la generación de código C.

3.1.3 LabVIEW

LabVIEW es un lenguaje de programación gráfico usado por numerosos ingenieros para sistemas hardware y software de pruebas, control y diseño. La ventaja de la programación gráfica es que es mucho más intuitiva que la programación textual, la cual requiere gran cantidad de tiempo en dominar la sintaxis propia del lenguaje. Además es más fácil de entender para los ingenieros, los cuales generalmente están familiarizados con la modelización gráfica de procesos. Otra de las ventajas es la gran cantidad de funciones prediseñadas que dispone, lo que facilita al usuario la programación, entre ellas librerías de adquisición de datos lo que permite comunicarse con numerosas tarjetas y aparatos. En labVIEW el código no se ejecuta siguiendo el orden de ubicación, a diferencia del lenguaje textual, en el que las líneas de código se ejecutan según el orden en el que están escritas. La ejecución de los programas se rige por lo que se conoce como flujo de datos, las diferentes funciones del programa están unidas por cables por los que circula la información. Cualquier nodo del programa no se ejecuta hasta que no está disponible la información en todas sus entradas.

Los programas realizados en LabVIEW se llaman instrumentos virtuales VIs. Los instrumentos virtuales constan de

- Panel frontal: Esta formados por controles e indicadores, que representas las entradas y salidas del VI. Los controles son variables de entrada que pueden ser modificadas por el usuario. Los indicadores son variables de salida e indican los resultados del programa.

- Diagramas de bloques: Están formados entre otros por bucles, constantes, operaciones matemáticas, sub VIs, funciones, conectados entre sí mediante cables a través de los cuales se transfieren los datos.

- Icono y panel conector: Situados en la esquina superior derecha, el icono es la representación gráfica del VI y el panel conector es el conjunto de entradas y salidas.

LabVIEW permite situar VIs dentro de otros VIs, en programas extensos los Sub VIs permiten agrupar el código y facilitar la comprensión del programa.

3.1.4 Ni VeriStand

El software Ni VeriStand permite configurar aplicaciones de pruebas en tiempo real importando modelos compilados procedentes de múltiples entornos de programación, será el programa empleado para lograr la ejecución conjunta de ambos modelos. Permite ejecutar archivos compilados de modelos Simulink .mdl (archivo .dll) asi como.vi procedentes de labVIEW (archivo compilado .lvmodel).

NI VeriStand Engine es el mecanismo controla la temporización del sistema completo, formado por una serie de *Timed Loops,* cuyo tiempo de ejecución esta controlados por eventos de una resolución de microsegundos. El tiempo de ejecución de los modelos se puede determinar al cargar los mismos en Ni VeriStand, el cual permite ejecutar varios modelos a diferentes frecuencias cada uno. Además de configurar el tiempo de ejecución de los modelos Ni Veristand también permite modificar el estado inicial del modelo, los parámetros del mismo, así como las entradas por defecto.

Dispone de un asistente para reconocer el hardware disponible *Hardware Discovery Wizard*, reconoce dispositivos de adquisición de datos DAQ , módulos FPGA como DAQ de la Serie R, así como hardware para redes automotrices como CAN, LIN y FlexRay. Ni VeriStand permite trabajar con una gran variedad de productos de adquisición de datos, entradas analógicas, salidas analógicas y E/S digitales.

Permite añadir canales calculados, utilizando la herramienta *Calculated Channel,* que pueden estar en función de las entradas y salidas de los modelos previamente compilados y añadidos. La herramienta *System Configuration Mappings* ,permite mapear las entradas y salidas de los modelos, los canales calculados así como las diferentes salidas y entradas analógicas entre sí.

3.2 Hardware utilizado

En el proyecto realizado se utiliza un PXI y un PC para controlar y ejecutar conjuntamente los dos modelos implementados.

3.2.1 PXI

El sistema PXI desarrollado por National Instruments en 1998 para pruebas, medidas y control, cuenta con numerosas áreas de aplicación :Hardware-in-the-loop test,

pruebas de audio y video, aplicaciones militares. El sistema consta de tres componentes: un chasis, modulos de entrada y salida y un controlador embebido o externo, permitiendo desarrollar sistemas en una gran diversidad de entornos software: Ni labVIEW, Ni TestStand, Ni VeriStand.

3.2.1.1 Chasis Ni PXI- 1078

El chasis sirve de alojamiento tanto para los módulos como para el procesador, dispone de varias ranuras en las cuales se insertarán. El control del chasis se puede realizar con PC externo o embebido en el sistema. El chasis utilizado se trata del NI PXIe-1078, este chasis dispone de 9 ranuras, aceptando módulos PXI Express en todas las ranuras y PXI híbridos en hasta cinco de ellas, dispone de características integradas de temporización y sincronización, relojes de referencia de 10 y 100 MHz, un ancho de banda del sistema 1.75 GB/s y por ranura de hasta 250MB/s.

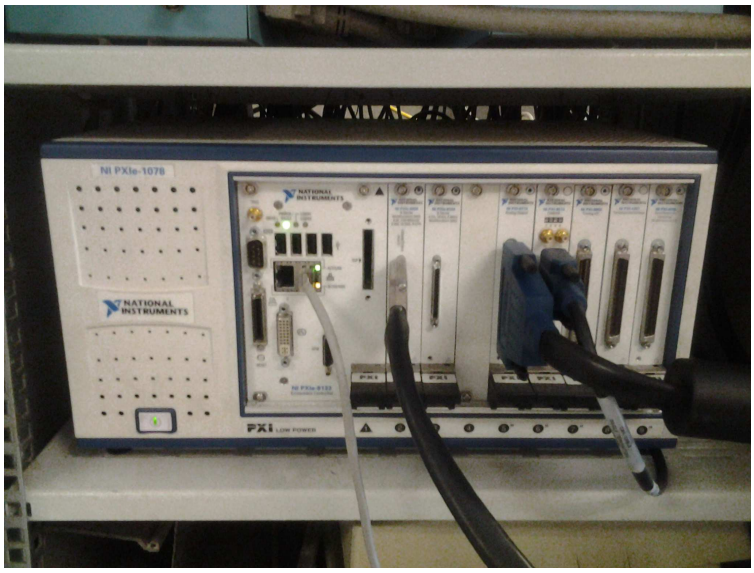


Figura 5. Chasis Ni PXI-1078

3.2.1.2 Controlador embebido PXI 8133

Los controladores embebidos están diseñados para dedicarse exclusivamente a unas pocas funciones, habitualmente en un sistema en tiempo real. Integran un procesador, memoria y distintos periféricos. El controlador utilizado en el proyecto se trata del NI PXIe-8133 cuyas características son las siguientes:

- Procesador Intel Core i7-820QM de cuatro núcleos.
- Memoria In-ROM.
- Dos tarjetas Ethernet de 10/100/1000BASE-TX.
- Diagnósticos de disco duro y de memoria.
- Plataforma para aplicaciones de labVIEW Real-Time.
- 4 canales de Hi-Speed USB.
- Ancho de banda con el sistema de hasta 8GB/s.

Figura6. Controlador embebido PXI.8133

3.2.1.3 Tarjeta de salida analógica Ni PXI 6713

Cada chasis dispone de un número de ranuras donde se pueden conectar los distintos módulos: de entrada y salida analógicos y digitales, tarjetas con prototipos, Carriers, CAN,Procesamiento Digital de Señales, Tarjetas de adquisición de imágenes, entre otros. El módulo utilizado en este trabajo para poder obtener señales análogicas se trata la tarjeta NI PXI-6713, un módulo de salida analógico que posee las siguientes características:

- Número de canales: 8 canales de salida analógicos
- Resolución: 12 bits
- Velocidad de muestreo: 1 MS/s
- Máximo Voltaje de entrada analógica: 10 V

Además el NI PXI- 6713 dispone de:

- Contadores de cuenta hacia arriba o hacia abajo, que pueden ser empleados en la temporización de la adquisición de datos.
- 8 canales de entrada y salida digitales.

Figura7. Tarjeta de salida analógica Ni PXI 6713

3.2.2. Ordenador ASUS

El ordenador personal elegido para el desarrollo del proyecto ha sido el Ordenador ASUS G2S 7R023G cuyas características se detallan a continuación:

- Intel Core 2 Duo T7500 / 2.2 GHz
- Velocidad bus de datos de 800 MHz
- Disco duro de 160 GB
- Pantalla de 17 pulgadas
- Resolución máxima:1440 x 900
- Soporte LAN inalámbrica

Figura 8 Ordenador Asus

Capítulo 4. Descripción sistema de simulación

El sistema de simulación está compuesto por dos modelos, un modelo Simulink de valores medios y un modelo labVIEW encargado del cálculo de la presión en la cámara de combustión. El principal reto surge al trabajar con dos modelos en entornos de programación diferentes. El objetivo es conseguir la integración y simulación conjunta de ambos modelos, la cual se logra mediante el uso del software NI VeriStand. El modelo Simulink se compila y se añade a NI VeriStand donde se ejecutará. Posteriormente se creará un proyecto en tiempo real en labVIEW que integrará tanto el modelo de presión como un programa encargado del acceso a VeriStand y control de la ejecución del modelo compilado, permitiendo modificar las entradas y obteniendo el valor de las salidas. La dinámica del modelo Simulink es mucho más lenta que la del modelo labVIEW y se trata de un modelo de valores medios por esa razón la ejecución del mismo se puede llevar a cabo en el ordenador dejando el PXI exclusivamente dedicado la ejecución del modelo de presión.
La comunicación entre ambos programas integrados en el proyecto en tiempo real creado, ejecutados en hardware diferente, se realizara mediante variables compartidas. Posteriormente las variables simuladas se escalaran y se enviaran como salidas analógicas de voltaje utilizando la librería DAQmx proporcionada por labVIEW.

Las ventajas del uso de NI VeriStand es el ahorro de tiempo que supone trabajar con modelos compilados, no siendo necesario volver a programar el código completo en labVIEW para ejecutar el mismo en tiempo real junto con el modelo de presión. De esta manera, simplemente es necesaria una visión general del funcionamiento del programa, de las entradas, de las salidas y de los diferentes subsistemas que forman el modelo, sin ser necesario comprender al detalle las ecuaciones que rigen su funcionamiento.

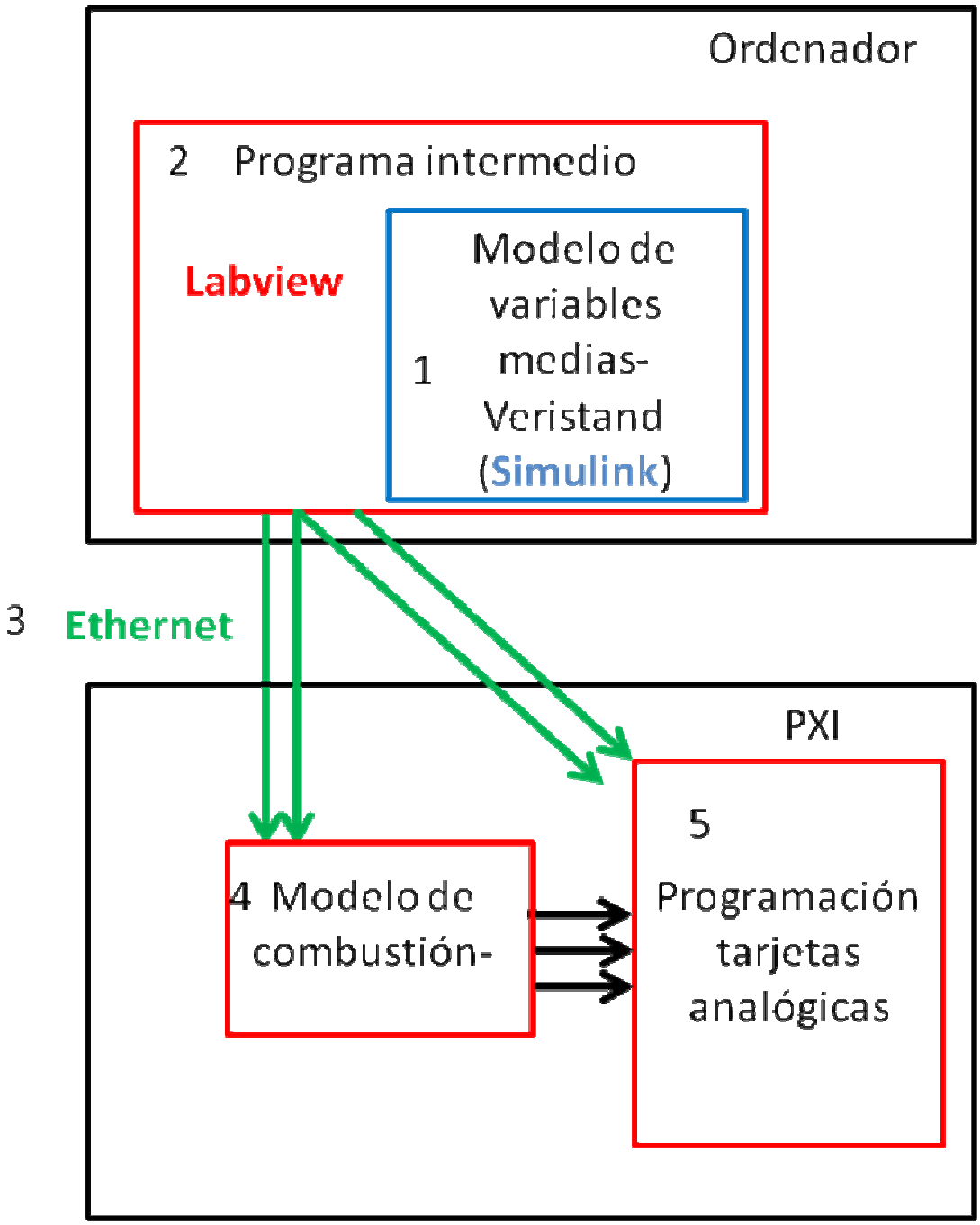


Figura 9. Esquema sistema de simulación

Esquema del sistema de simulación

1. Compilación modelo Simulink y ejecución con Ni VeriStand.

Se dispone de un modelo matemático de un motor diésel sobrealimentado con una turbina de geometría variable y EGR de alta presión, se trata de un modelo de valores medios que calcula presiones, temperaturas, caudales másicos, fracciones másicas de oxígeno, entre otros, en los diferentes puntos del motor. El objetivo es compilar ese modelo (crear un archivo .dll) y ejecutar el mismo en Ni VeriStand. La frecuencia de ejecución la determinaremos al añadir el modelo compilado en Ni VeriStand.

2.Control de la ejecución de Ni VeriStand desde LabVIEW.

Aunque el modelo previamente compilado se ejecutará en Ni VeriStand, el control de la ejecución se realizará desde labVIEW, programa que dispone de funciones prediseñadas para controlar la ejecución de modelos en VeriStand, con estas funciones se crea un programa capaz de modificar las entradas del modelo y leer las salidas del mismo.

3. Envío de variables por Ethernet.

El programa LabVIEW creado se ejecutará en el PC. Las entradas y las salidas del modelo de valores medios se enviarán por Ethernet mediante variables compartidas a un programa labVIEW que se ejecutará en el PXI. Las entradas del modelo ejecutado en el PXI son entradas y salidas del modelo ejecutado en Ni VeriStand.

4. Estructura del archivo que calcula la presión en la cámara de combustión y la ejecuta en tiempo real.

En labVIEW se implementa un modelo que calcula la presión en cámara de combustión en un motor diésel basándose en la ley de combustión de Wiebe ejecutando la señal creada en tiempo real. El programa se divide en 3 partes. En la inicialización, se realizan todos los cálculos que se mantendrán constantes durante la ejecución del programa, cálculo del volumen de la cámara de combustión, lectura de las tablas de datos experimentales, creación de los canales virtuales de salida analógicos y configuración de la temporización de dichos canales.

El cuerpo del programa está formado un bucle, encargado del cálculo de la presión. Las entradas del modelo son variables compartidas recibidas por Ethernet procedentes del archivo.vi que esta ejecutándose en el PC. El código contenido en el bucle genera un vector que contiene los valores de la presión en la cámara de combustión en diferentes ángulos del cigüeñal durante el ciclo de compresión-combustión-expansión. La velocidad de ejecución del bucle dependerá de las revoluciones del motor.

5. Generación de señales analógicas.

En último lugar se generarán señales analógicas tanto de la presión en la cámara de combustión como de las salidas del modelo Simulink compilado, recibidas por Ethernet, utilizando la interfaz NI DAQmx.

4.1 Modelo de variables medias

4.1.1 Modelo Simulink

El modelo Simulink compilado se trata de un modelo de valores medios que simula el comportamiento de un motor diésel con un sistema de recirculación de gases de escape de alta presión y sobrealimentado mediante una turbina de geometría variable, el modelo implementado simula las condiciones estáticas de operación así como las dinámicas. Al tratarse de un modelo de valores medios ignora detalles cíclicos del motor, como el movimiento del pistón, la compresión expansión y combustión de los gases, proporcionando potencia y par constante.

El modelo está formado por una serie de subsistemas que son modelos de las diferentes partes del motor: Intercooler, turbo, filtro de aire, colector de admisión, colector de escape, cámara de combustión, sistema de recirculación de gases de escape, filtro de partículas diésel.
Algunos de estos subsistemas contienen otros en su interior por ejemplo en el caso del turbo se encuentran los siguientes subsistemas: Turbina, Compresor y Inercia del turbo.
La función de los diferentes subsistemas es:

- Intercooler: Al comprimirse los gases por el sobrealimentador estos se calientan. Dicho calentamiento no es deseado porque disminuye la densidad de los mismo y por consiguiente la masa de oxígeno por unidad de volumen que entrará al cilindro. El intercooler es un intercambiador cuya misión es disminuir la temperatura de los gases aumentando el rendimiento volumétrico y por tanto la potencia del motor.

- Filtro de aire: La función del filtro de aire es eliminar las impurezas. Su uso reduce los posibles daños en el motor y garantiza una combustión óptima.

- Sistema de recirculación de gases de escape de alta presión: El EGR consiste en introducir parte de los gases de escape en el motor, con el objetivo de reducir las emisiones contaminantes de óxidos de nitrógeno, debido a la disminución la temperatura de la combustión. El modelo compilado presenta EGR de alta

presión, los gases de escape se recirculan desde la salida de la cámara de combustión antes de su paso por la turbina hasta la salida del intercooler.

- Filtro de partículas diésel: La función de este filtro, ubicado en el tubo de escape, es retener las partículas sólidas generadas en la combustión. Cuando la cantidad de partículas es elevada se produce automáticamente la regeneración del filtro, está consiste en inyectar en algunos ciclos del motor más carburante, aumentando la temperatura de los gases de escape y consiguiendo así incinerar las partículas.

- Colector de admisión: Es el conducto que conduce el aire a los cilindros del motor.

- Turbocompresor de geometría variable: La principal desventaja de los turbos convencionales es su comportamiento a bajas revoluciones ya que la turbina apenas es impulsada por los gases de escape, la alternativa al problema sería el uso de turbocompresores de geometría variable, este tipo de turbocompresores eliminan el retraso de respuesta del turbo presentando buena eficiencia tanto a altas como a bajas revoluciones. Utilizan unos alabes móviles cuya posición dependerá de las revoluciones del motor. La posición de los alabes será parcialmente cerrada a bajas revoluciones y más abierta a medida que aumentan estas.

- Cámara de combustión. Es el lugar donde se realiza el proceso de combustión.

Todos estos subsistemas forman un sistema principal cuyas entradas y salidas serán las reconocidas en la compilación, se podrán modificar las entradas y obtener el valor de las salidas, las de los subsistemas no serán reconocidas, por ello es importante que el sistema principal tenga un aspecto similar al de la siguiente imagen. En el programa original las diferentes entradas se realizaban desde el Workspace de Matlab, en el programa a compilar las entradas deben ser Inports para que NI VeriStand las identifique como entradas y no como parámetros del modelo.

El esquema del modelo compilado y el sistema principal son los siguientes

- Esquema modelo compilado

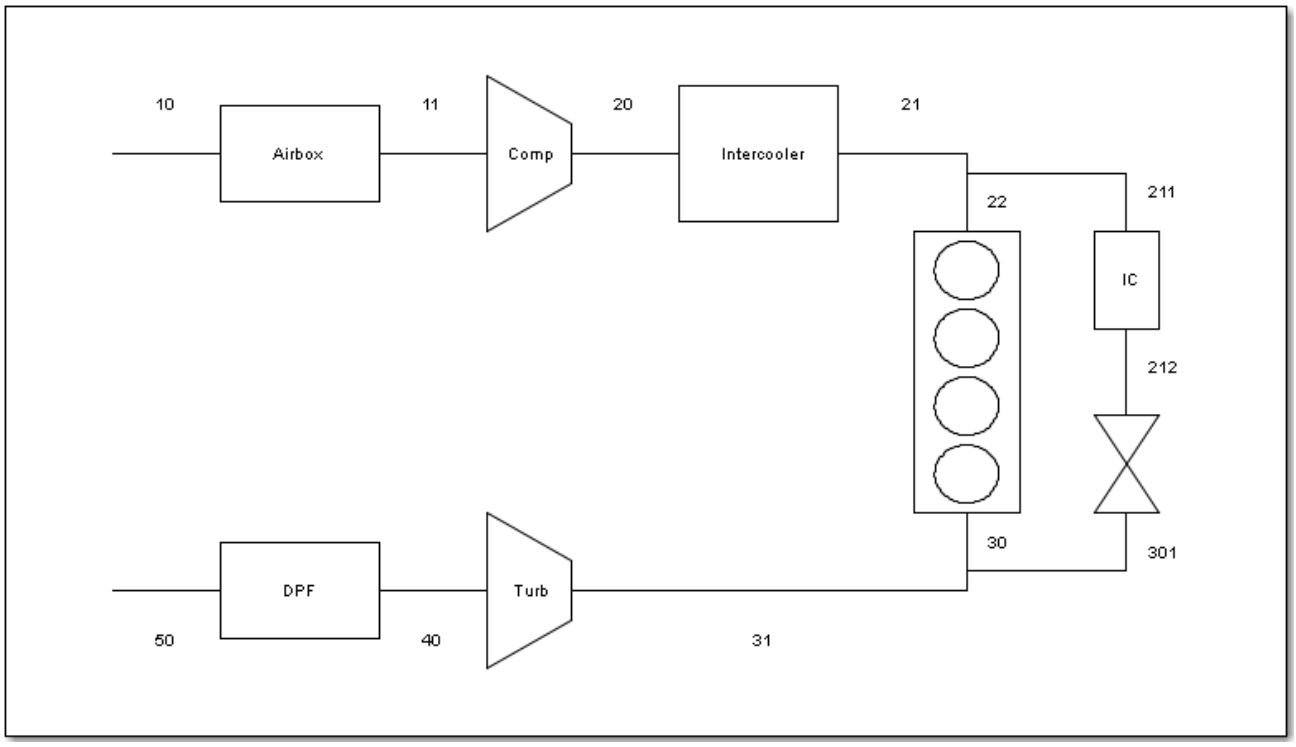


Figura 10 Modelo Simulink Compilado

- Sistema principal

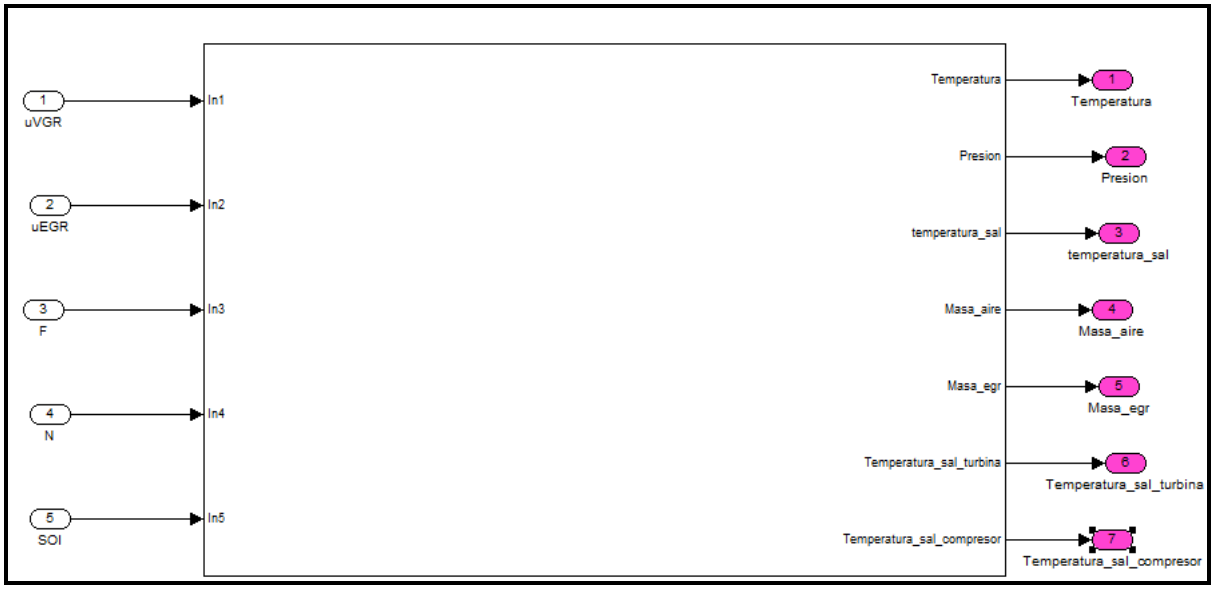


Figura 11 Sistema principal modelo compilado

El modelo calcula presiones, temperaturas, fracciones másicas de oxígeno y EGR y caudales másicos en los diferentes puntos del esquema anterior.
El modelo también permite calcular la eficiencia y la potencia mecánica del compresor y de la turbina, las emisiones de NOx y de hollín, el par efectivo y potencia del motor. Para poder conocer sus valores deben ser salidas del sistema principal en Simulink.

4.1.2. Proceso de compilación

VeriStand es un entorno de integración de modelos que permite fácilmente ejecutar y conectar simultáneamente modelos de múltiples entornos de programación previamente compilados y conectar a dispositivos E/S de hardware de Ni.
El primer objetivo será la compilación del modelo creado en Simulink software en un archivo .dll compatible con Ni VeriStand.
Se utiliza The Mathworks Inc Real- Time Workshop software (Simulink Coder) y Microsoft Visual Studio para convertir el modelo .mdl en un archivo .dll. . The Real-Time Workshop genera código C de los diagramas realizados en Simulink. Posteriormente el compilador convierte el código C en un archivo .dll que se ejecutará desde Ni VeriStand.
El modelo Simulink será para VeriStand como una caja negra input-output, la ventaja de este procedimiento de compilación del modelo .mdl es permitirnos ejecutarlo en Ni VeriStand desde LabVIEW y añadir más código al programa sin necesidad de volver a programar el archivo .mdl en labVIEW, lo que supondría conocer las ecuciones que rigen el modelo, sin embargo compilando el mismo, simplemente es necesario tener una visión general de su funcionamiento.

En el anexo 1.2 se puede consultar la guía del proceso de compilación de modelos Simulink, los cambios necesarios a realizar, y ejecución de los mismo en VeriStand.

En el anexo 1.4 explica como realizar la compilación de modelos LabVIEW, y los requerimiento necesarios para poder efectuarla con éxito.

El siguiente esquema describe el proceso de creación del archivo .dll y el software utilizado en dicho proceso.

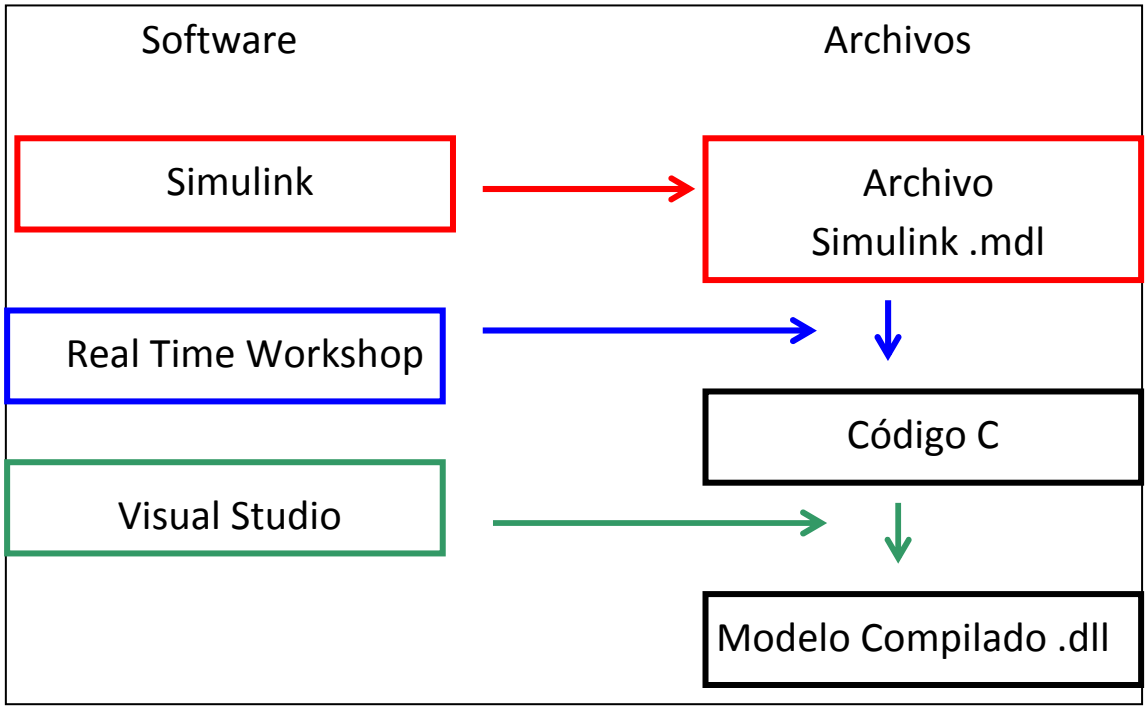


Figura12. Software utilizado y archivos generados en el proceso de compilación

4.1.2.1 Versiones de software utilizadas en el proceso de compilación.

Versiones de software utilizadas para realizar el proceso de compilación son las siguientes:

- The MathWorks Inc. MATLAB software and The MathWorks, Inc Simulink application software 2011a
- The MathWorks Inc. Real-Time Workshop software (Simulink Coder) 2011a
- Microsoft Visual Studio 2008 Professional
- Ni VeriStand 2012

En el Anexo 1.1 se podrán consultar todas las versiones compatibles.

4.1.2.2 Compilación del modelo .dll a través de VeriStand.

VeriStand reconoce entradas, salidas, parámetros y señales del modelo compilado. El modelo Simulink está formado por una serie de subsistemas, se añade un bloque *Inport* para la señal que entra al subsistema y un bloque *Outport* para la señal que sale del mismo. Estas entradas y salidas generadas automáticamente al crear un subsistema en Simulink no estarán disponibles en VeriStand como *inports* and *outports*. VeriStand solo reconocerá las del sistema principal no las de los subsistemas que contiene este.

En el archivo compilado las entradas y salidas son las siguientes:

Entradas	
mf	Cantidad de combustible injectado en mg/cc
N	Régimen de giro en rpm
SOI	*Start of injection en º*
uEGR	Tasa de recirculación de gases de escape. Posición de la valvula. 1 completamente abierta 0 completamente cerrada
uVGT	Turbina de geometría variable 1 completamente abierta 0 cuando está cerrada

Tabla 1. Entradas Modelo Simulink

Salidas	
Presión	Presión a la entrada de la cámara de combustión

Temperatura	Temperatura la entrada de la cámara de combustión
Temperatura_sal	Temperatura a la salida de la cámara de combustión
Temperatura_sal_turbina	Temperatura a la salida de la turbina
Temperatura_sal_compresor	Temperatura a la salida del compresor
Masa_aire	Caudal másico de aire a la salida del intercooler
Masa_egr	Caudal másico de egr

Tabla 2. Salidas Modelo Simulink

El valor por defecto de todas las entradas en Ni VeriStand es 0, se puede cambiar dicho valor antes de ejecutar el modelo. Es conveniente modificar los valores para evitar valores no válidos. Por ejemplo cualquier división entre 0 daría como resultado un valor no válido.

Respecto a los parámetros, los modelos pueden contener dos tipos de parámetros: parámetros globales y parámetros locales. Los globales por defecto se aplican al modelo actual y a cualquier parámetro global con el mismo nombre en otro modelo. En cambio los parámetros locales se aplican a un subsistema o bloque específico del modelo al que pertenecen. En este caso todos los parámetros del modelo compilado son locales pues pertenecen a subsistemas del modelo Simulink.

Por defecto, los valores iniciales de los parámetros son los del modelo Simulink. Sin embargo, Ni VeriStand puede aplicar valores leídos de un archivo de texto a los parámetros, esto es útil si se desean cambiar los mismo sin recompilar el modelo. Alternativamente se pueden cambiar en el modelo Simulink y volver a compilarlo.

Ni VeriStand reconoce como parámetros las ganancias, las constantes, las saturaciones tanto su valor inferior como superior ,las memorias y los integradores, en el caso del modelo previamente compilado.

4.1.2.3 Frecuencia de ejecución del modelo

Ni Verstand Engine, mostrado en la siguiente imagen, es el responsable de la ejecución, entre otros, del hardware I\O, de los modelos, de las alarmas, para lo cual dispone de numerosos *Timed Loops*.

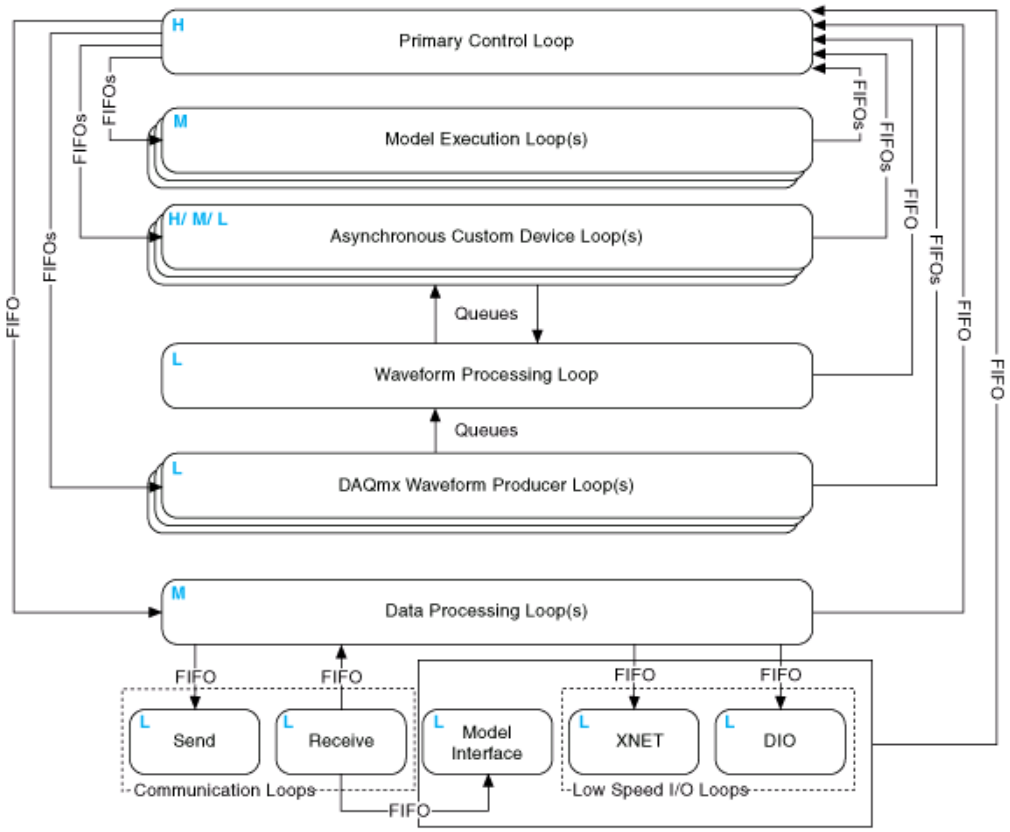


Figura 13. Ni VeriStand Engine

De los anteriores *Timed Loops* mostrados en la figura, el llamado *Model Execution Loop* determinará la frecuencia de ejecución del modelo. El número de Model Execution Loops dependerá del número de modelos añadidos, en Ni Veristand se pueden ejecutar varios modelos en paralelo cada uno a diferente frecuencia. La siguiente ecuación describe como VeriStand Engine ejecuta el modelo, la frecuencia de ejecución del mismo dependerá del Primary Control Loop y del model decimation que se establezcan en NI Veristand.

Model Execution Loop= Primary Control Loop / model decimation

4.2. Simulación de la combustión

En primer lugar se muestra el sistema caja negra del modelo, el objetivo es a partir de unos parámetros de entrada, entradas y salidas del modelo Simulink compilado, obtener la presión en la cámara de combustión, un vector con muestras de la misma a lo largo del ciclo de alta presión. La etapa de renovación de la carga se simulará como una función constante seguida de una función de tipo exponencial. Finalmente se calculará el número de muestras que debe tener la señal de presión para enviar la misma de manera satisfactoria como una salida analógica de voltaje en tiempo real.

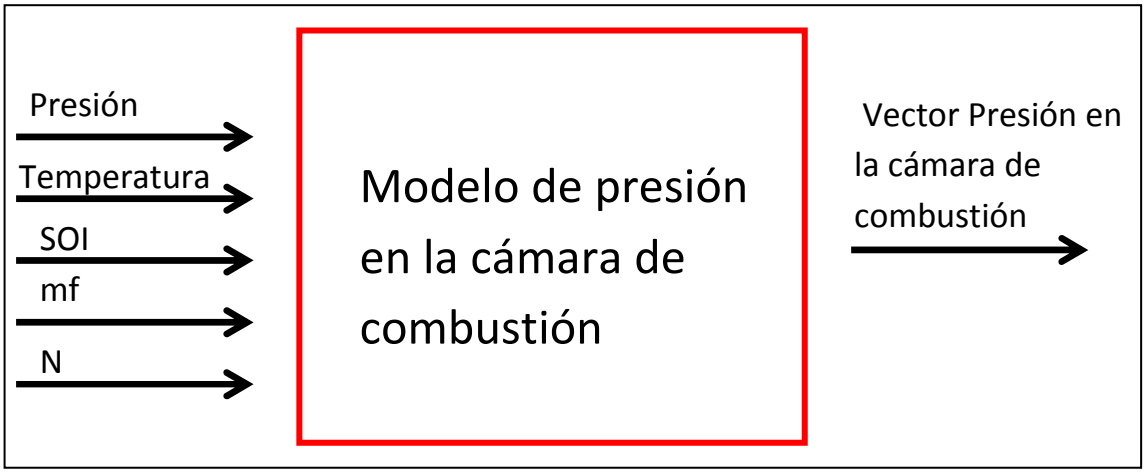


Figura 14. Sistema caja negra del modelo de presión en la cámara de combustión

4.2.1 Estructura interna del modelo

El modelo constará de una serie de Sub VIs conectados entre sí, con el objetivo de jerarquizar el modelo. Las diferentes etapas y cálculos se agruparan de manera que la programación sea más clara y fácil de entender y en el caso de que se produzcan errores sean más sencillos de corregir. En los siguientes apartados se describen las ecuaciones matemáticas de las diferentes partes del modelo.

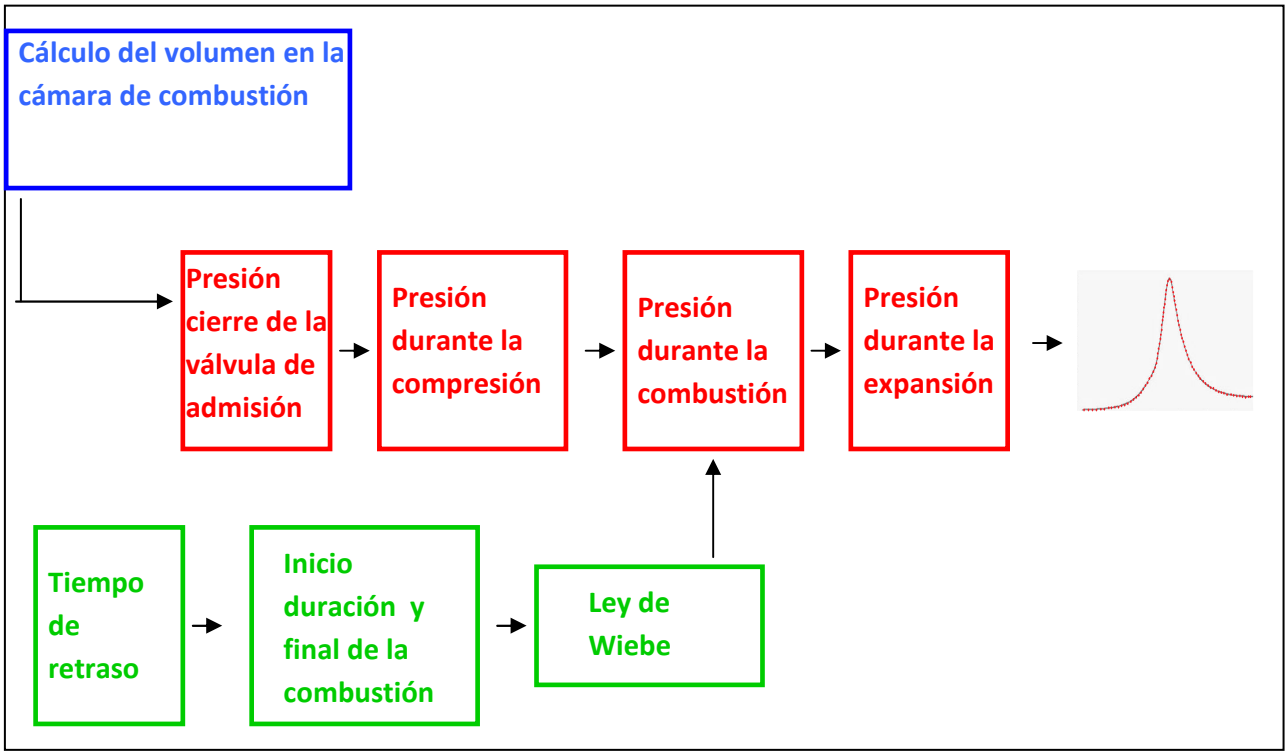


Figura 15 Estructura interna modelo de presión

4.2.1.2 Cálculo del volumen de la cámara de combustión

El objetivo es calcular el volumen de la cámara de combustión en función de los siguientes parámetros geométricos:

- Longitud de la biela Lb
- Longitud de la manivela Lm
- Diámetro de la cámara de combustión D1
- Ángulo que forma la vertical que pasa por el centro del cigüeñal y por el bulón con la manivela θ
- Relación de compresión Kc

En primer lugar se calcula, la distancia entre el bulón y el centro del cigüeñal, en función del ángulo.

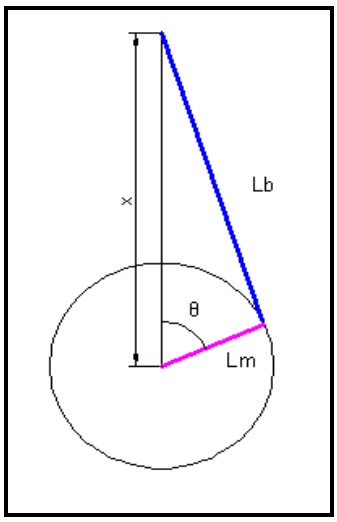


Figura16. Sistema biela-manivela

$$Lb^2 = Lm^2 + x^2 - 2 * Lm * x * \cos(\theta)$$

$$Lb^2 - Lm^2 = x^2 - 2 * Lm * x * \cos(\theta) + Lm^2 * [(\cos(\theta))^2 + (\sin(\theta))^2 - 1]$$

$$Lb^2 - Lm^2 * (\sin(\theta))^2 = x^2 - 2 * Lm * x * \cos(\theta) + Lm^2 * (\cos(\theta))^2$$

$$Lb^2 - Lm^2 * (\sin(\theta))^2 = (x - Lm * \cos\theta)^2$$

$$x = Lm * \cos(\theta) + \sqrt{Lb^2 - Lm^2 * \sin^2(\theta)}$$

El volumen mínimo de la cámara de combustión se calculará a partir del volumen desplazado y la relación de compresión

$$Vd = \frac{\pi * d_1}{4} * Lm * 2$$

$$Vcc = \frac{Vd}{1 + K_c}$$

Posteriormente, a partir de los cálculos realizados, se obtiene la fórmula que permite conocer el volumen en función de ángulo.

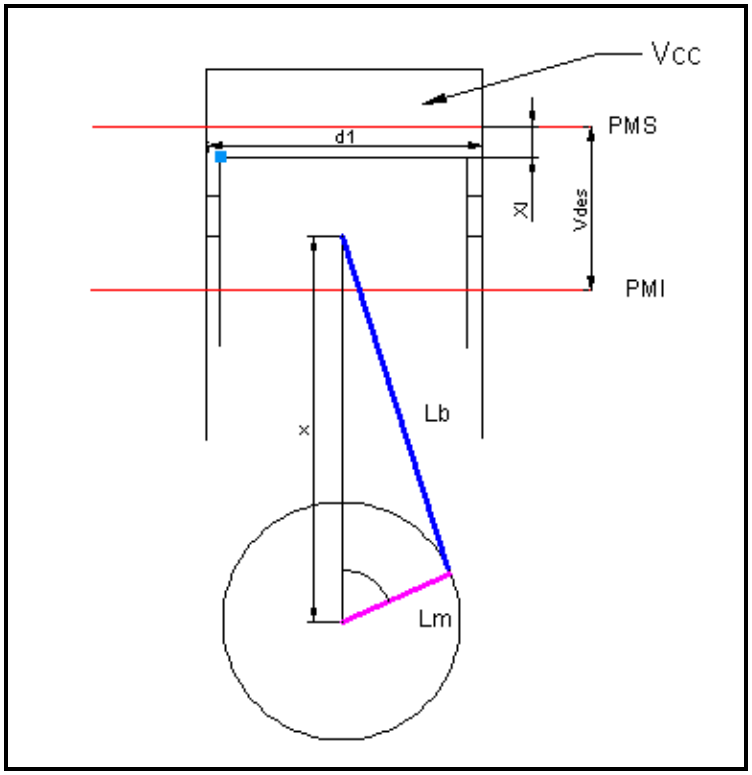


Figura 17. Mecanismo pistón

$$x_l = Lb + Lm - Lm * \cos(\theta) + \sqrt{Lb^2 - Lm^2 * \sin^2(\theta)}$$

$$Volumen = \frac{\pi * d_1^2}{4} * x_l + V_{cc}$$

La siguiente ecuación se implementará en labVIEW, se realizará en la inicialización del programa pues los valores calculados se mantienen intactos a lo largo de la ejecución del bucle. La resolución del vector ángulos elegido determinará la del vector volumen y por consiguiente la del vector de presión generado.

4.2.1.2 Presión en el instante del cierre de la válvula de admisión.

El objetivo es crear un vector que contenga muestras del ciclo de alta presión. El vector se creará a partir de un valor inicial, obtenido del modelo Simulink compilado, la presión en el instante de cierre de las válvulas de admisión, instante que coincidirá con el punto muerto inferior.

El vector presión se creará en primer lugar con un número de muestras suficientemente elevado como para representar con detalle la señal, este número de muestras coincidirá con las del vector volumen anteriormente creado.

4.2.1.3 Cálculo de la presión durante la compresión

Una vez inicializada la presión se procede al cálculo de la misma en los diferentes ángulos del cigüeñal durante el proceso de compresión.

La compresión se trata de un proceso politrópico, conocido el volumen de la cámara de combustión para cada ángulo y el valor de presión en un punto se puede calcular el valor de la misma en el resto aplicando la ecuación que rige los procesos politrópicos.

$$P(\theta)=\left(\frac{V_{PMC}}{V(\theta)}\right)^{K} * P_{PMC}$$

- K coeficiente politrópico del proceso
- Volumen en el punto de referencia (apertura válvula de admisión)
- Presión en el punto de referencia (apertura válvula de admisión)

4.1.2.4 Cálculo de la presión durante la combustión

En el siguiente apartado se explicarán tanto los conceptos teóricos como las ecuaciones utilizadas para comprender el cálculo de la presión durante la combustión.

4.1.2.4.1 Tiempo de retraso

Es el tiempo comprendido entre el inicio de la inyección y el inicio de la combustión. Generalmente tiene una duración comprendida entre 0.5 y 1 ms y es debido a dos retrasos que se producen en la cámara de combustión

-Retraso físico: Relacionado con la atomización y vaporización del combustible, y con la mezcla del combustible evaporado con el aire. Depende principalmente de las

condiciones de inyección, para garantizar una buena atomización el combustible se inyecta a elevadas presiones por orificios de diámetro muy pequeño. Asimismo las temperaturas elevadas alcanzadas por la compresión facilitan la evaporación y la mezcla

-Retraso químico: Conjunto de prerreacciones de rotura de las cadenas de hidrocarburos previas a la combustión. El retraso químico depende principalmente de las condiciones de temperatura, cuando mayor sea la temperatura
mas rápidamente se producirán las reacciones. También depende en menor medida de la presión así como del número de cetano del combustible.

El tiempo de retraso global, no es la suma algebraica, ya que ambos procesos se solapan comenzando las prerreaciones químicas antes de que se finalice la evaporación y mezcla del combustible. Generalmente el tiempo de retraso químico es más largo que el físico.

Se han implementado en labVIEW las siguientes ecuaciones para calcular el tiempo de retraso

$$P_m = Presión(\theta_{SOI})$$

$$T_m = \left(\frac{P_m * Volumen(\theta_{SOI}) * Temperatura(\theta_{IVC})}{Presión(\theta_{IVC}) * Volumen(\theta_{IVC})} \right)$$

$$Tiempo\ de\ retraso = 0{,}40 * \left(\frac{P_m^{-1{,}16}}{100000} \right) * e^{\frac{4820}{Tm}}$$

4.2.1.4.2 Inicio, duración y final de la combustión

Tras la inyección del combustible y después de un tiempo de retraso tiene lugar el inicio de la combustión, la combustión en los motores al contrario de lo que sucede en los ciclo ideales no es instantánea. El tiempo de combustión se ha obtenido a partir de tablas procedentes de ensayos experimentales, dependiendo de las revoluciones y de la masa de combustible inyectada.

	6,10E+00	1,18E+01	1,58E+01	1,80E+01	2,24E+01	3,08E+01	3,46E+01
1,25E+03	3,18E+01	3,00E+01	3,46E+01	3,72E+01	4,23E+01	4,56E+01	4,72E+01
1,50E+03	3,28E+01	3,69E+01	3,98E+01	4,13E+01	4,43E+01	5,01E+01	4,77E+01
2,00E+03	3,33E+01	3,94E+01	4,35E+01	4,58E+01	4,67E+01	4,84E+01	4,77E+01
3,00E+03	3,59E+01	3,48E+01	3,40E+01	3,35E+01	3,56E+01	3,94E+01	4,17E+01
4,00E+03	3,00E+01	3,38E+01	3,65E+01	3,80E+01	4,10E+01	4,67E+01	4,99E+01

	4,21E+01	4,67E+01	5,43E+01	6,00E+01	6,44E+01
1,25E+03	4,98E+01	5,14E+01	5,41E+01	5,61E+01	5,76E+01
1,50E+03	4,28E+01	4,62E+01	5,17E+01	5,59E+01	5,91E+01
2,00E+03	4,64E+01	4,56E+01	5,06E+01	5,43E+01	5,72E+01
3,00E+03	4,62E+01	4,89E+01	5,60E+01	6,12E+01	6,53E+01
4,00E+03	5,62E+01	6,32E+01	7,50E+01	8,38E+01	9,06E+01

Tabla 3. Tiempo de duración de la combustión

La tendencia del TOC es un aumento con el régimen de giro y con el combustible inyectado.
Las tablas anteriores se cargaran en la inicialización del programa y no dentro del bucle, de esta manera se consigue optimizar el código.

4.2.1.4.3 Ley de wiebe

En un motor la masa de combustible se quema a lo largo del tiempo de combustión. La fracción de calor liberado en función del ángulo del cigüeñal va desde 0 (no hay combustión) hasta 1 (la combustión se ha completado) siguiendo una función de tipo S. Generalmente las leyes que representan la fracción de combustible quemado son de tipo exponencial, la utilizada en este caso para modelar la combustión es la llamada ley de Wiebe.

En primer lugar se calculan los parámetros experimentales de la ley de Wiebe

$$D_b = TOC * 0{,}8$$

$$D_d = TOC * 0{,}1$$

$$m = \left(\frac{\ln\left(\frac{\ln(1-0{,}1)}{\ln(1-0{,}85)} \right)}{\ln(D_d) - \ln(D_d + D_b)} \right) - 1$$

$$a = -\ln(1-0{,}1) * \left(\frac{TOC}{D_d} \right)^{m+1}$$

Una vez calculados los parámetros experimentales se calcula la fracción de calor liberado.

$$x = 1 - e^{-a * \left(\frac{\theta - \theta_{SOC}}{TOC} \right)^{m+1}}$$

Y Finalmente el calor liberado

$$\frac{dQ}{d\theta} = m * H \frac{dx}{d\theta}$$

La presión se calcularía aplicando la siguiente fórmula

$$P_2 = \frac{\frac{dQ}{d\theta} + \frac{V_2 * P_1}{\gamma - 1}}{\frac{\gamma}{\gamma - 1}\frac{dV}{d\theta} + \frac{V_2}{\gamma - 1}}$$

4.2.1.5 Cálculo de la presión durante la expansión

La etapa de expansión sigue al igual que la compresión un proceso politrópico. Se obtiene a partir de la siguiente fórmula.

$$P(\theta) = \left(\frac{V_3}{V(\theta)} \right)^K * P_3$$

- Ke coeficiente politrópico del proceso
- V_3 Volumen en el punto de referencia (final de la combustión)

- P_3 presión en el punto de referencia

4.2.2 Señal generada

Al final se obtiene la señal de presión discretizada. Formada por muestras de la misma a lo largo de las tres etapas compresión, combustión y expansión. En el apartado dedicado a la generación de salidas analógicas se explican las modificaciones que hay que realizar a la señal de presión discretizada para conseguir enviar la misma en tiempo real.

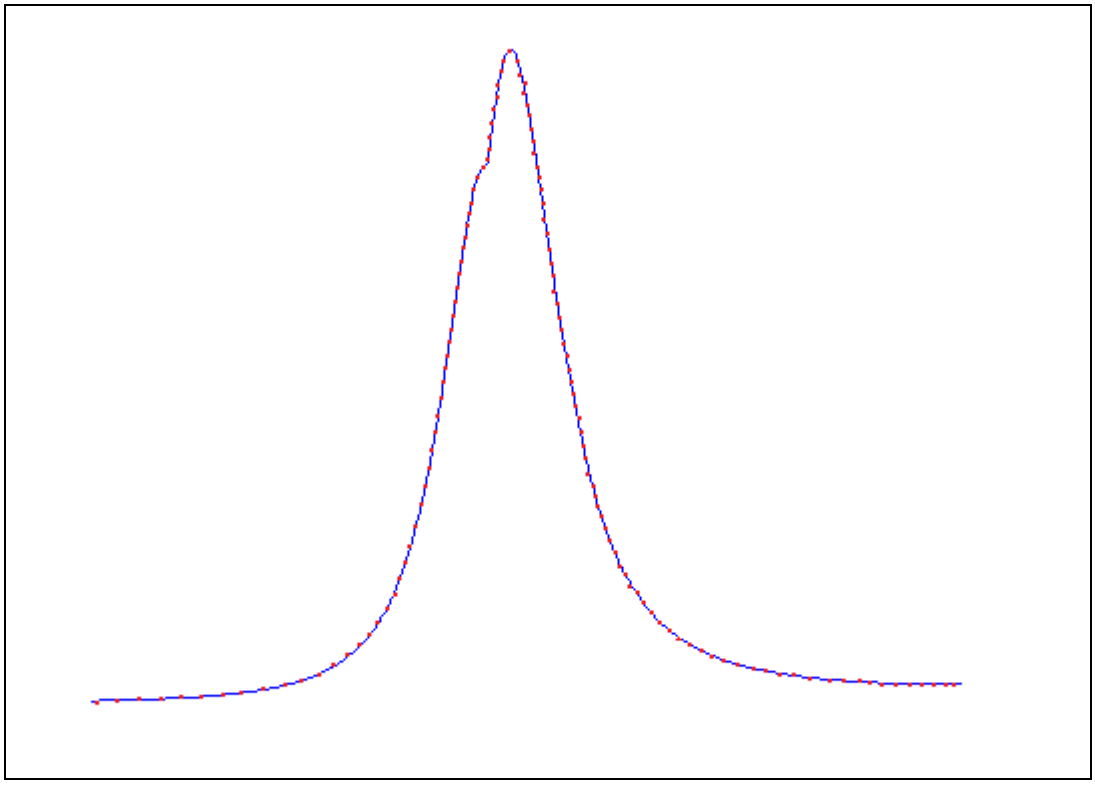
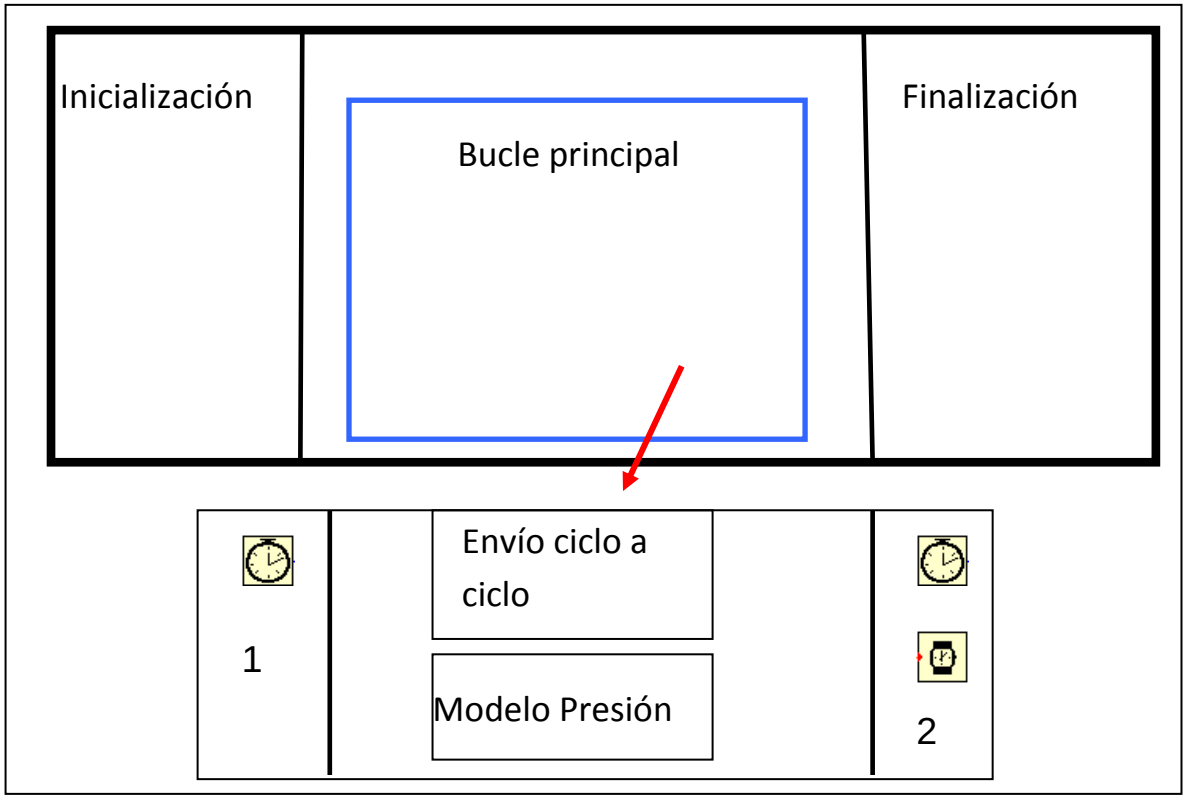


Figura 18. Señal de presión discretizada

4.2.3 Tiempo ejecución del bucle

El bucle encargado del cálculo de la presión en el cilindro se trata de un bucle cuyo tiempo de ejecución dependerá de las revoluciones del motor, consiguiendo así enviar

la señal de presión en tiempo real. El bucle se divide en 3 partes que se ejecutarán secuencialmente. En primer lugar un indicador tiempo nos indicará el valor del instante en que ha comenzado la iteración, posteriormente se ejecutará el código anteriormente descrito. Una vez obtenido el vector de presión discretizado, convenientemente escalado, se escribirán las muestras en la correspondiente tarjeta de salida analógica. Todo este proceso conlleva un tiempo de ejecución, el tiempo en el que finaliza la ejecución del código se conoce gracias a otro indicador. Por tanto para conseguir ejecutar el ciclo en tiempo real debemos restar el tiempo que ha tardado el código anterior en ejecutarse al tiempo que dura un ciclo. Consiguiendo ejecutar los cálculos ciclo a ciclo.



Temporización 1: Tiempo de inicio de ejecución de la iteración

Temporización 2: Tiempo de finalización del código ejecutado

Espera: Tiempo de duración de un ciclo - Duración del código ejecutado

4.3. Comunicación de datos

El modelo Simulink compilado y el modelo labVIEW deben conectarse para funcionar conjuntamente. La integración de ambos se realizará dentro de un proyecto en tiempo real, en primer lugar se implementará un modelo capaz de acceder al Workspace de VeriStand y por tanto de controlar la ejecución del modelo compilado, posteriormente se conectaran las salidas y entradas de dicho modelo, mediante el uso de variables compartidas, con el modelo de presión en cámara ejecutado en LabVIEW.

4.3.1 Ejecución de VeriStand

Uno de los componentes de un proyecto en Ni VeriStand es el Workspace que permite observar y editar las variables de los modelos previamente compilados y añadidos al *System Explorer*. Su función y aspecto es similar al panel frontal de labVIEW formado por controles y indicadores que representan las entradas y salidas del modelo. Una vez creado y añadido el modelo compilado a Ni VeriStand, el objetivo es controlar la ejecución del mismo desde LabVIEW, utilizando la API que este proporciona.

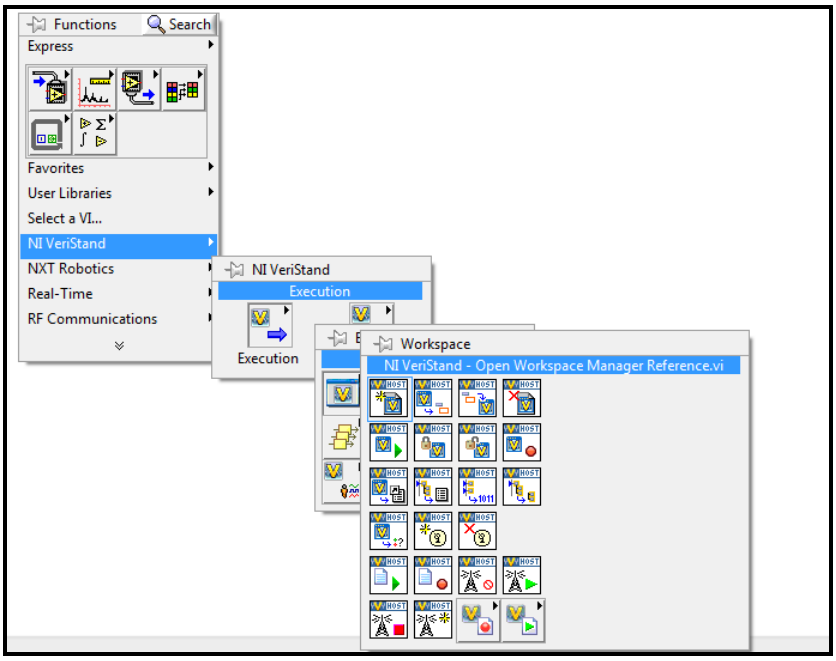


Figura 19. NI VeriStand API for Workspace

Las entradas del modelo compilado se modifican desde LabVIEW, el cual dispone de funciones ya diseñadas para conectar y desconectar el sistema, abrir y cerrar el Workspace de Ni VeriStand y acceder a los canales del mismo. Una vez conectado el sistema el modelo estará continuamente ejecutándose, a la frecuencia que se ha determinado al cargar el mismo en Ni VeriStand, hasta que se desconecte el sistema y se cierre el Workspace. Desde LabVIEW se pueden modificar los valores de entrada del modelo así como obtener las salidas del mismo. Cualquier aspecto referente a la frecuencia de ejecución del modelo o a la tarjeta donde se quiere ejecutar el mismo se debe modificar previamente en Ni VeriStand, LabVIEW simplemente se utiliza para dar valor a las diferentes entradas del modelo.
El programa creado se divide en tres partes. En la inicialización se conectará el sistema y se abrirá el Workspace. El bucle principal se trata de un Timed Loop que envía las entradas a Ni VeriStand y lee el valor de las salidas con una frecuencia 100ms. Finalmente si se produce un error o si se finaliza la ejecución del programa, se cerrará el Workspace y se desconectará el sistema.

Es importante que el modelo no este en estado de ejecución en VeriStand cuando se ejecute el .vi creado. Ya que este .vi se encarga de conectar el sistema y abrir el Workspace, si el sistema estuviera previamente conectado daría un error el programa. Para acceder a los diferentes canales de entrada y salida del Workspace es necesario indicar el path completo en el apartado *Channel* correspondiente.

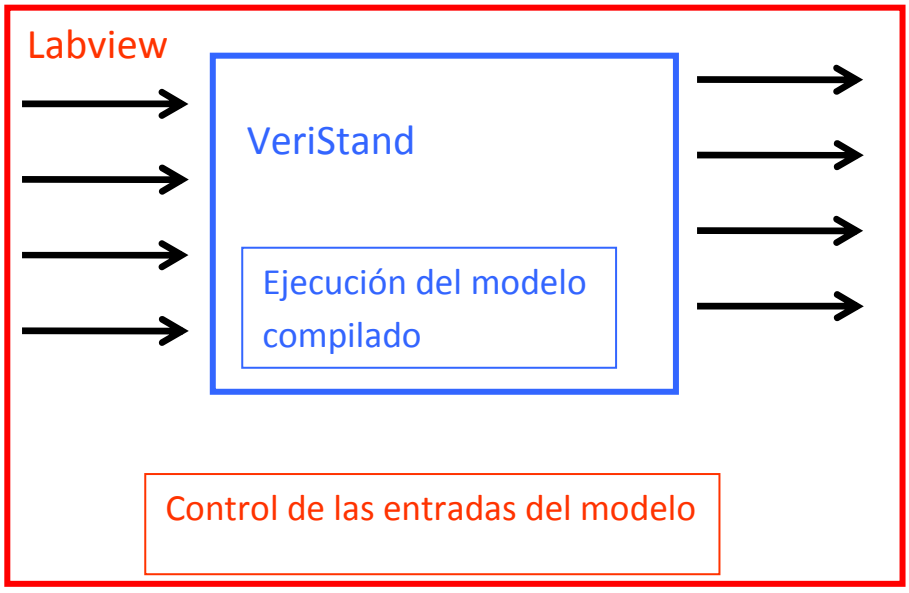


Figura 20. Ejecución de Ni VeriStand.

4.3.2 Envío de variables por Ethernet.

En el proyecto realizado se tienen dos archivo .vi realizados en LabVIEW. Uno se ejecutará en el PC y otro en el PXI. El primer archivo, anteriormente descrito, se encarga de acceder a VeriStand donde se está ejecutando el modelo compilado. El segundo archivo se encarga de simular la presión en la cámara de combustión de un motor y ejecutar esa presión ciclo a ciclo en tiempo real. Las entradas y salidas de los modelos anteriores son las siguientes.

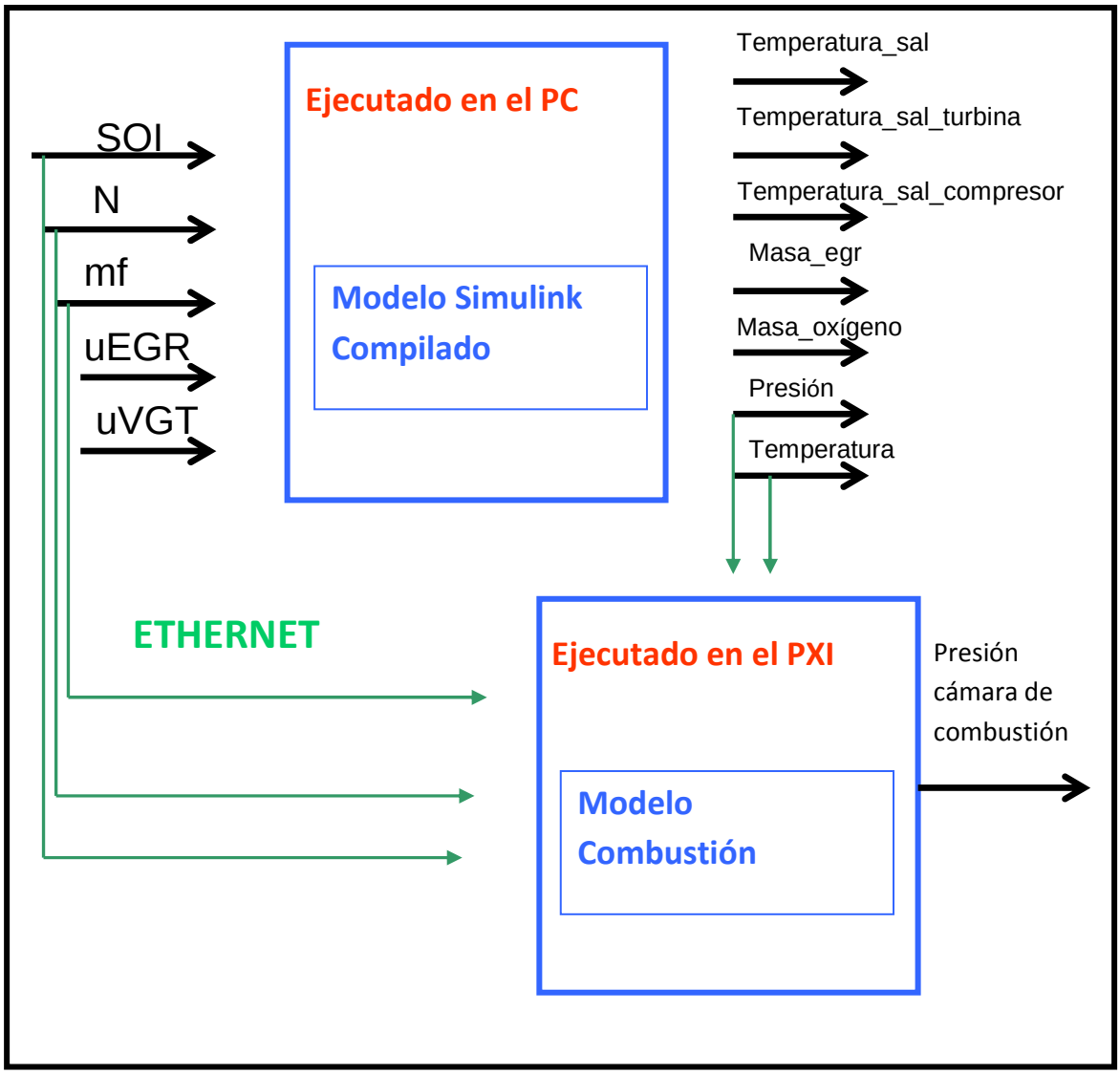


Figura 21. Envío de variables por Ethernet

Se puede comprobar que las entradas del modelo ejecutado en el PXI son entradas y salidas del modelo Simulink compilado. El envío se realizara por Ethernet mediante variables compartidas en labVIEW, este tipo de variables se utilizan dentro de un proyecto y permiten intercambiar información entre Vis ejecutándose en hardware distinto, en este caso un PC y un PXI.

4.3.3 Salidas analógicas

LabVIEW permite, utilizando tarjetas de adquisición de datos, generar y adquirir señales eléctricas tanto digitales como analógicas. Proporciona varios drivers para la generación y adquisición de las mismas. En el proyecto realizado se utiliza la librería mxDAQ que presenta numerosas ventajas respecto a la Ni DAQ tradicional. Entre ellas mayor robustez y estabilidad, es una API más simple que requiere menos funciones para la adquisición y posee mayor integración con el MAX.

De la librería mxDAQ cabe destacar los siguientes Vis

- **DAQmx Create Virtual Channel**

La función crea uno o múltiples canales virtuales y los añade a una tarea, si no se especifica la tarea también la crea automáticamente.

- **DAQmx Read**

Lee muestras del canal virtual especificado, la lectura puede ser de una muestra o de varias y tanto analógica como digital.

- **DAQmx Write**

Escribe muestras en el canal virtual que se especifique, al igual que en el caso anterior se pueden escribir una o varias muestras tanto analógicas como digitales.

- **DAQmx Timing**

Permite configurar parámetros relativos al tiempo. Estos parámetros serán el número de muestras recibidas o enviadas, la frecuencia de envío de las mismas así como el tipo de ejecución, continua o discreta. Dispone de un reloj de muestreo, que controla la velocidad de adquisición y generación de los datos.

- **DAQmx Start Task**

Transforma una tarea al estado de ejecución. Si no empleas este VI, cuando se ejecute DAQmx Read or DAQmx Write en múltiples ocasiones, por ejemplo dentro de un bucle, la tarea se ejecuta y para repetidamente lo cual disminuye el rendimiento de nuestra aplicación.

- **DAQmx Stop Task**

Detiene una tarea, la cual vuelve al estado anterior de la ejecución de DAQmx Start Task.

El flujo de trabajo utilizando NI DAQmx sería el siguiente

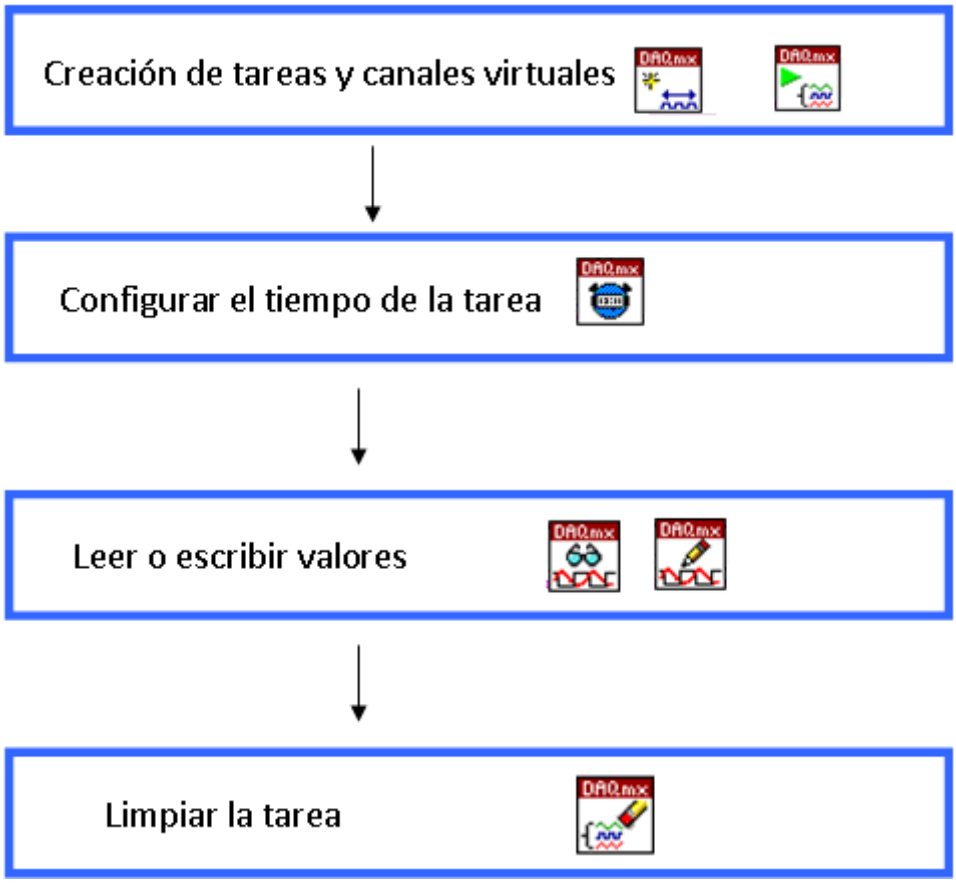


Figura22. Flujo de trabajo utilizando la librería NI DAQ mx

Envio señal analógica de presión

En la generación de la señal analógica de presión hay que tener en cuenta los siguientes aspectos. La frecuencia de muestreo de la tarjeta de adquisición de datos , un parámetro configurado en la inicialización del programa, será siempre la misma durante la ejecución. El número de muestras enviadas dependerá de las revoluciones del motor. Al ser la frecuencia de muestreo constante (muestras/s), el número de muestras enviadas determinará el tiempo que se están enviando muestras del ciclo de alta presión. El tiempo de envío de muestras debe coincidir con el tiempo de duración de una revolución del motor que es el tiempo en el que se realiza el ciclo de presión. A medida que el número de revoluciones del motor sea mayor el número de muestras enviadas tiene que ser menor al durar menos milisegundos una revolución el tiempo de envío de las muestras es más pequeño.

El vector de presión creado, siempre tiene el mismo número de elementos dependiendo de la resolución elegida en la inicialización del programa. Se debe interpolar el mismo para que tenga el número de muestras necesarias para que el tiempo de envío dure exactamente una revolución del motor. El envío de las muestras se realizará de modo continuo. Al enviarse en modo continuo una vez enviadas todas las muestras generadas del ciclo de alta presión se continuará enviando la última muestra hasta que se envíe un nuevo conjunto de muestras a la tarjeta. Este hecho produciría un salto no deseado en la señal. Para ello simulamos una función exponencial que tendrá como valor inicial la presión en el escape y como valor final la presión en la admisión. Esta señal no ocupará todo el ciclo de renovación de la carga sino parte de este.

5.Resultados y conclusiones

En el siguiente apartado se comprobará en primer lugar que el modelo Simulink compilado se está ejecutando en Ni VeriStand de manera correcta. Se mostrarán las gráficas de los resultados obtenidos ejecutando el modelo en Simulink y en VeriStand controlando la ejecución desde labVIEW donde se guardarán los resultados para realizar la comparación.

En segundo lugar es necesario comprobar que el ciclo de presión se está ejecutando en tiempo real. Se efectuarán cambios en las diferentes entradas del modelo y se mostrarán los resultados obtenidos.

5.1 Variables Medias

Es necesario comprobar que la compilación del modelo Simulink, previamente verificado con datos reales, se ha realizado de manera satisfactoria. Esto implica que los cambios efectuados en el modelo para realizar la compilación y permitir la ejecución en NI VeriStand no han alterado el programa. Además implica que los bloques de los diferentes subsistemas han sido reconocidos así como los archivos .mat de variables cargados. Al realizar la comprobación, obteniendo los datos desde el archivo labVIEW creado para controlar la ejecución, se verifica que el número de muestras obtenido mediante el programa labVIEW es significativo para representar las señales satisfactoriamente.

En el archivo labVIEW se implementa una temporización con el objetivo de calcular el tiempo de ejecución del programa. Dicha temporización se compara con la de una fuente externa, un periodo de tiempo suficientemente grande como garantizar el funcionamiento en tiempo real.

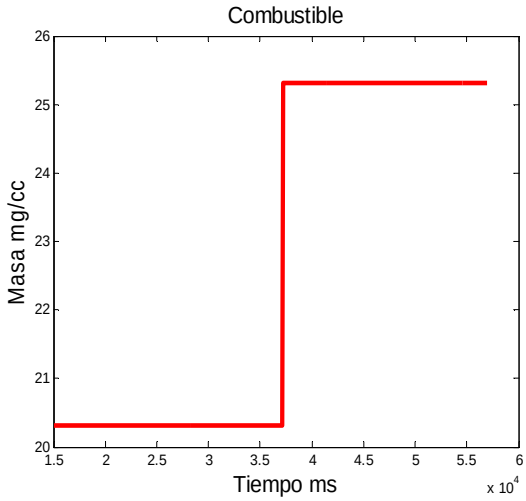
En las siguientes gráficas se muestran los resultados obtenidos modificando cada una de las entradas del modelo, comprobando que la compilación se ha efectuado con éxito.

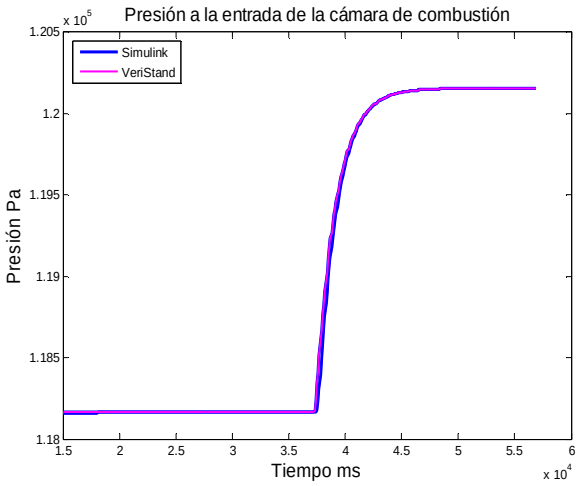
5.1.1 Modificación del combustible inyectado.

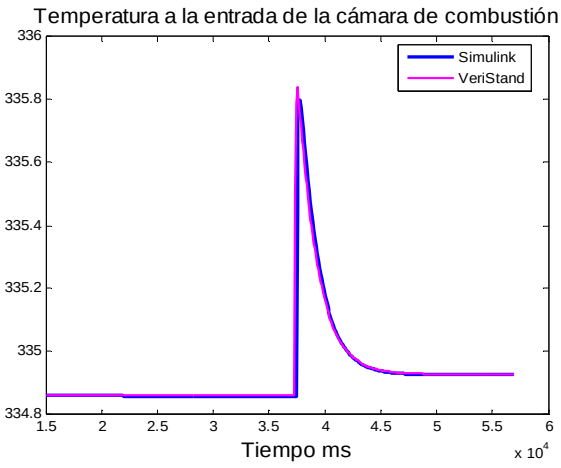
Sistema en régimen estático en T=15 segundos

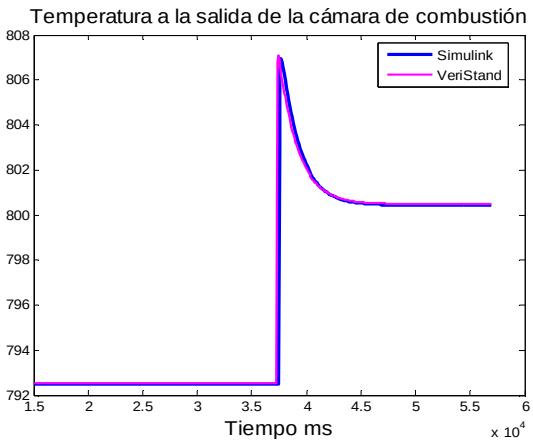
N=2250	Mf= 20.31 mg/cc	SOI=-1.992º	Egr=0.3455	Vgt=0.5657

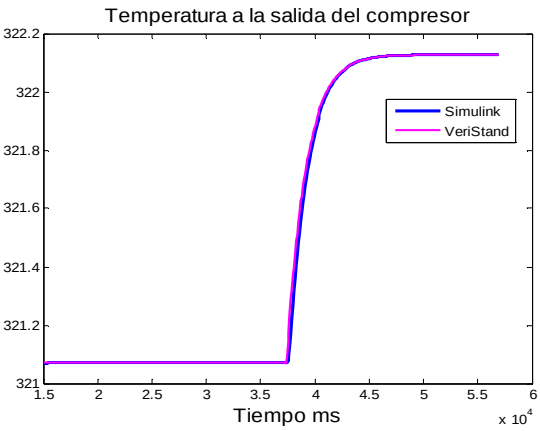
Cambio del combustible inyectado en 37.4 segundos a 25.31 mg/cc

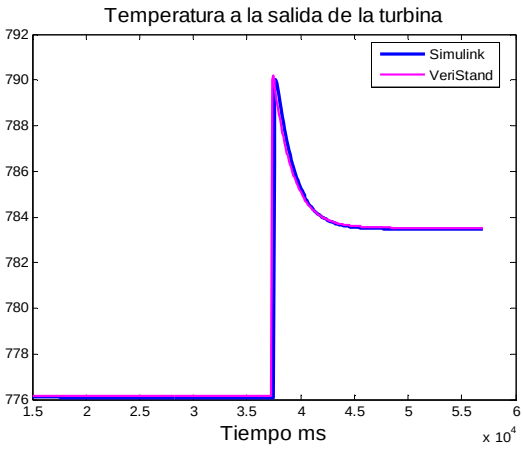










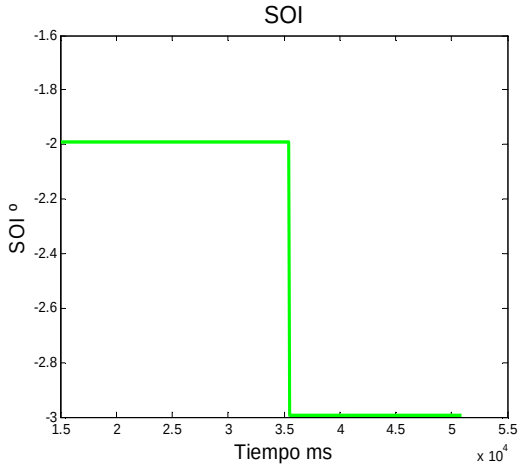


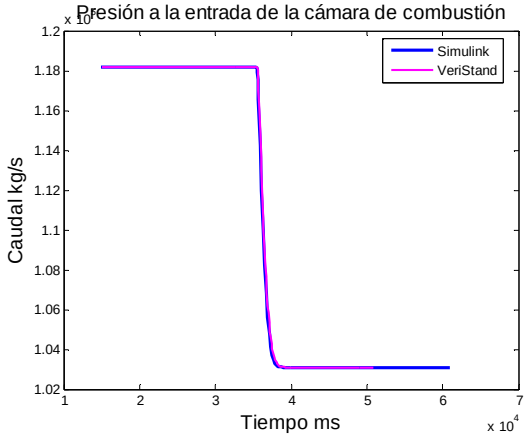
5.1.2 Modificación del inicio de la inyección.

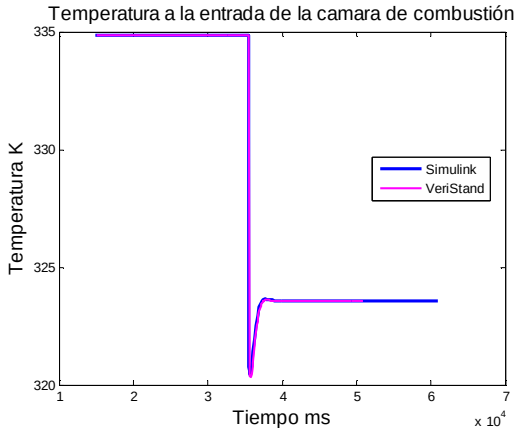
Sistema en regimen estático en T=15 segundos

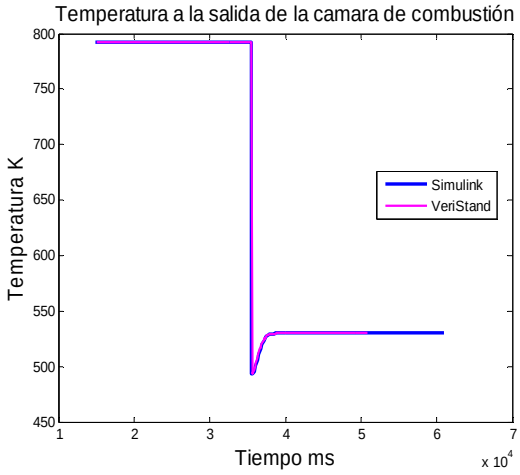
N=2250	Mf= 20.31 mg/cc	SOI=-1.992º	Egr=0.3455	Vgt=0.5657

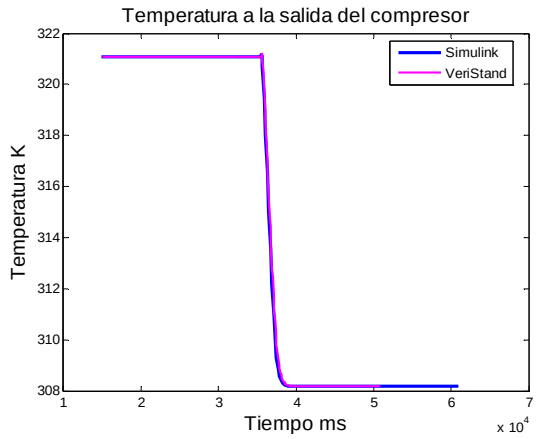
Cambio del inicio de la inyección en 35.5 segundos a -2.992º

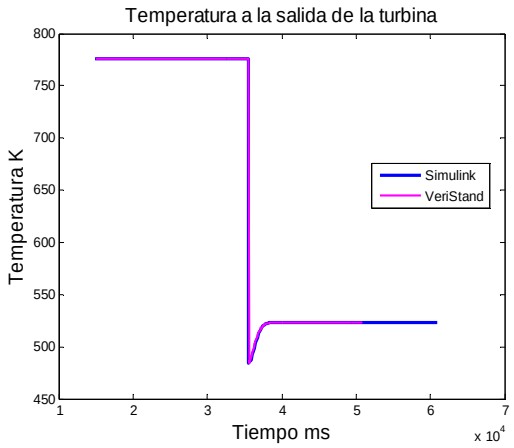










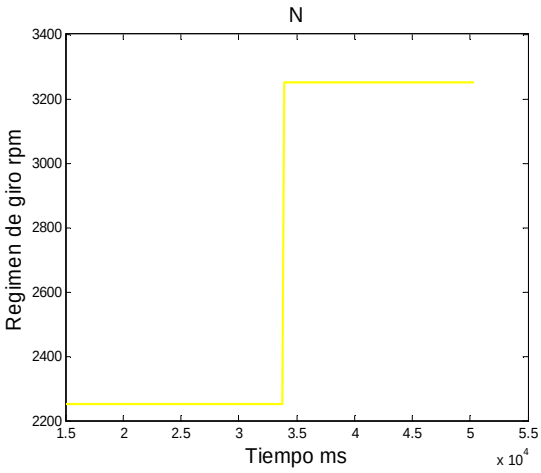


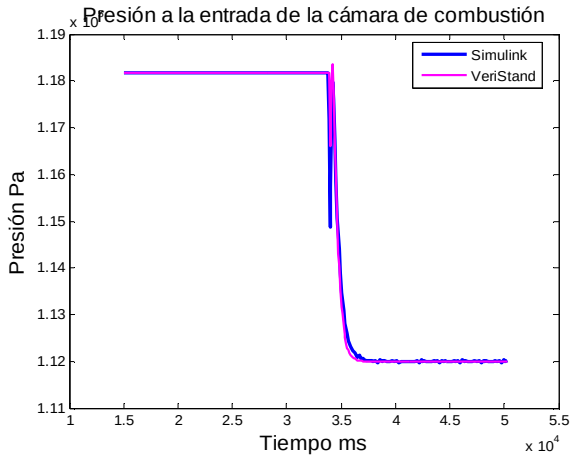
5.1.3 Modificación del régimen de giro.

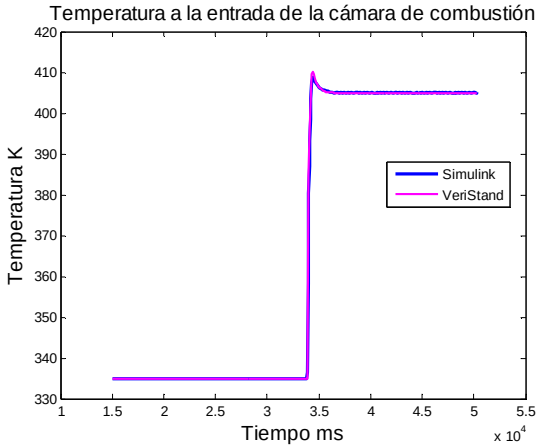
Sistema en régimen estático en T=15 segundos

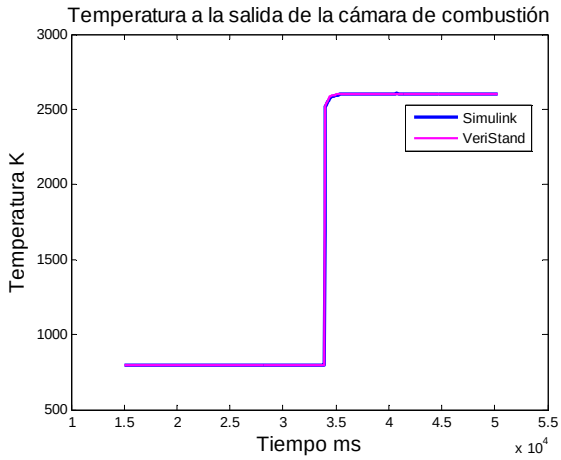
N=2250	Mf= 20.31 mg/cc	SOI=-1.992º	Egr=0.3455	Vgt=0.5657

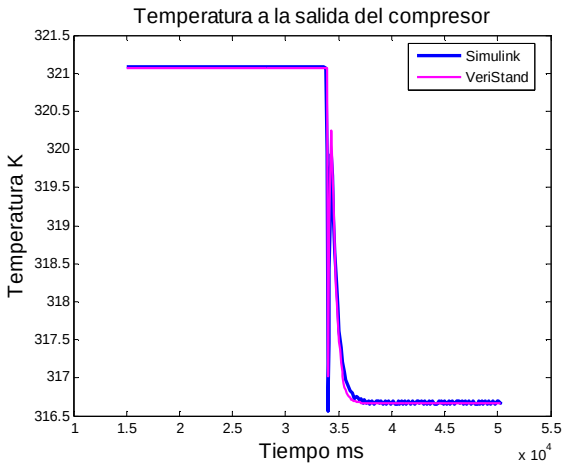
Cambio del regimen de giro en 33.9 segundos a 3250 rpm

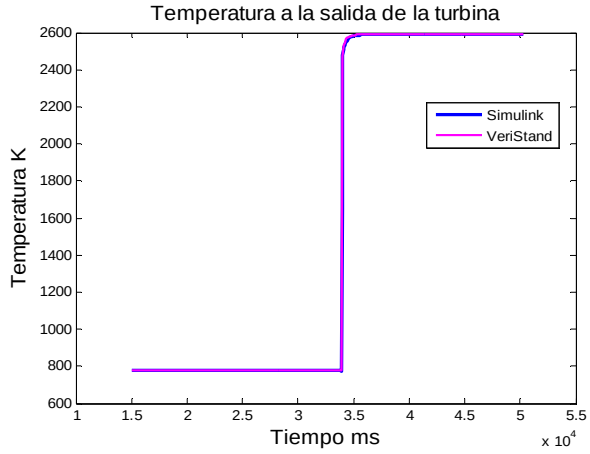










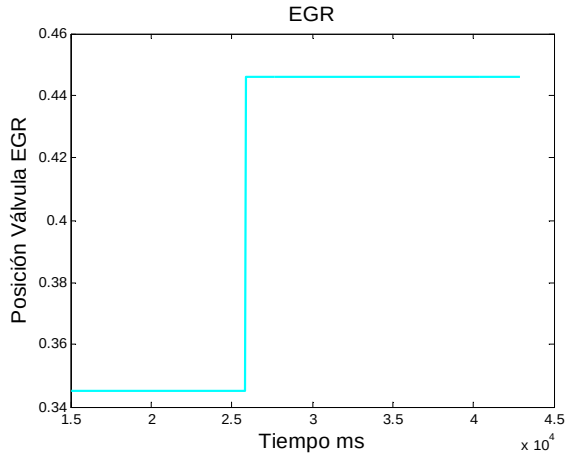


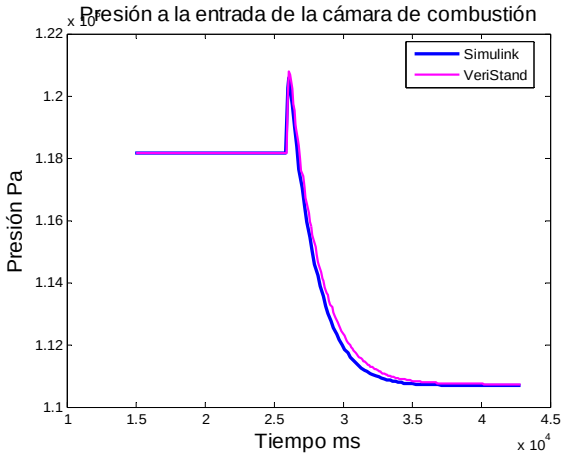
5.1.4 Modificación de la posición de la válvula de egr.

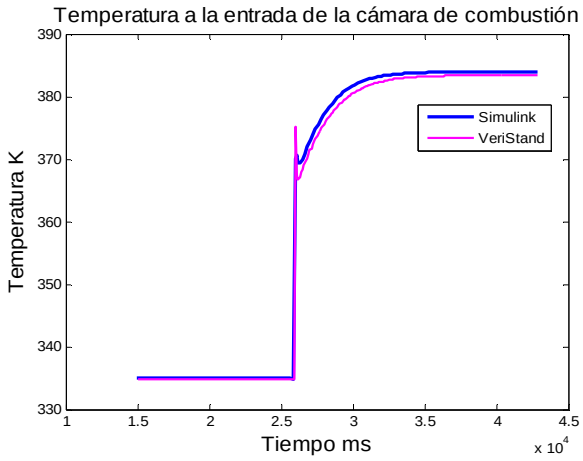
Sistema en régimen estático en T=15 segundos

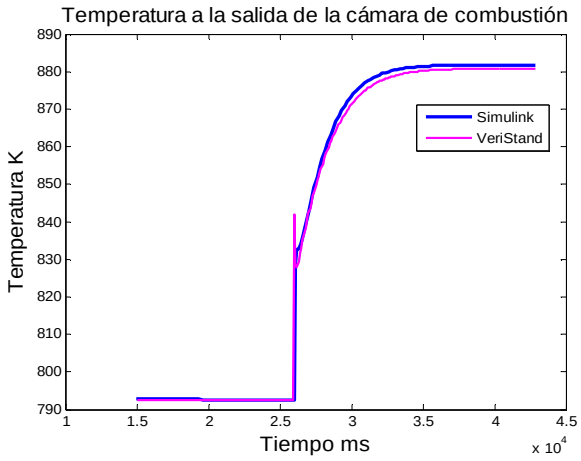
N=2250	Mf= 20.31 mg/cc	SOI=-1.992º	Egr=0.3455	Vgt=0.5657

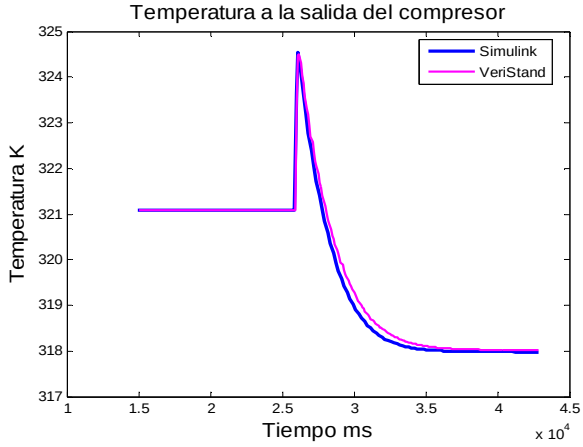
Cambio de la posición de la válvula de recirculación de gases de escape en 25.9 segundos 0.446

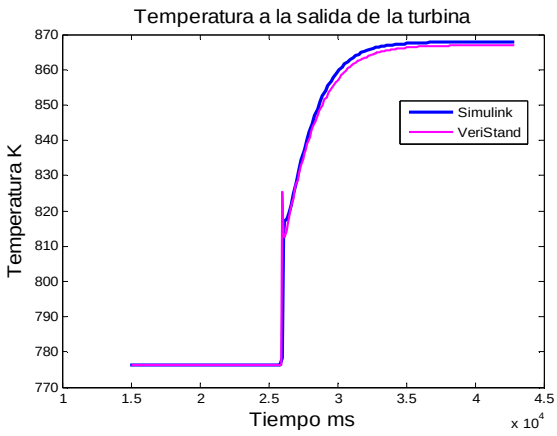










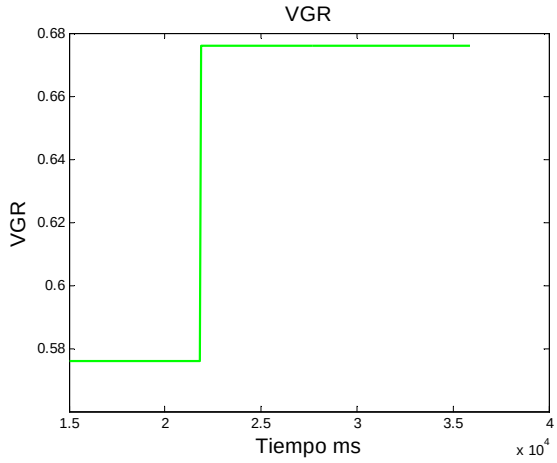


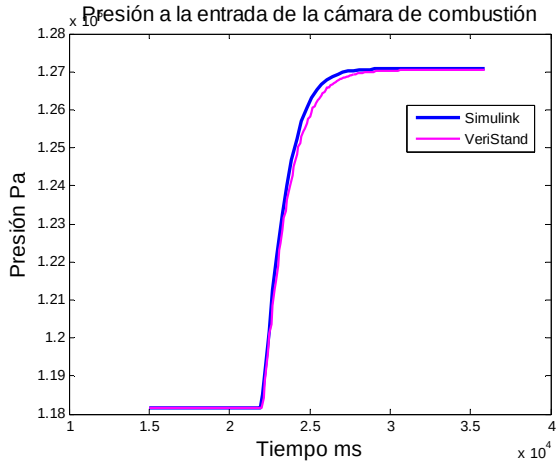
5.1.5 Modificación de la turbina de geometría variable.

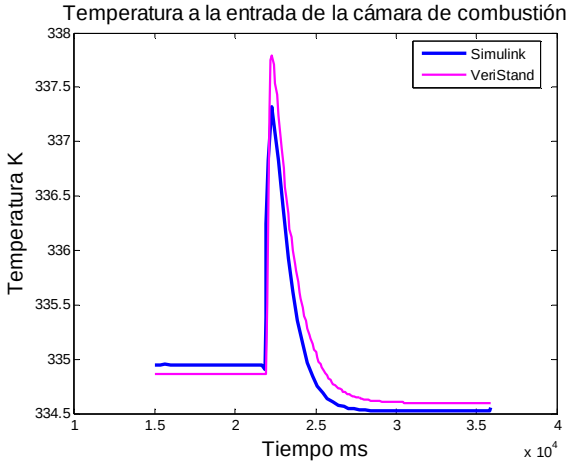
Sistema en régimen estático en T=15 segundos

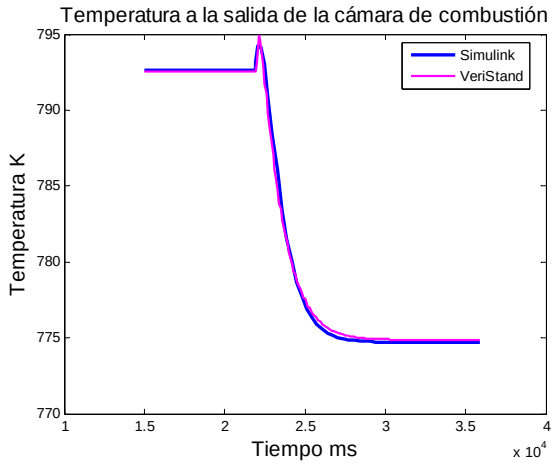
N=2250	Mf= 20.31 mg/cc	SOI=-1.992º	Egr=0.3455	Vgt=0.5657

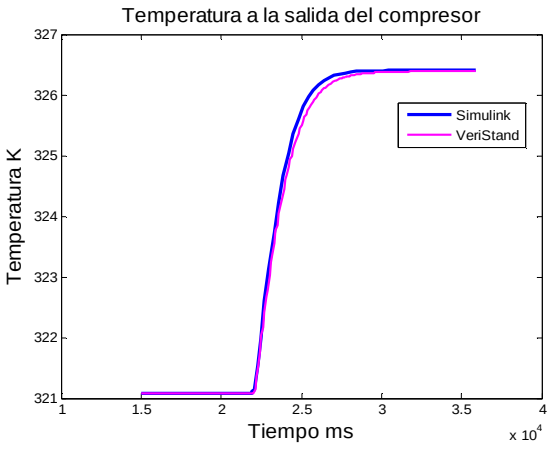
Cambio de la abertura de la turbina de en 21.9 segundos a 0.676

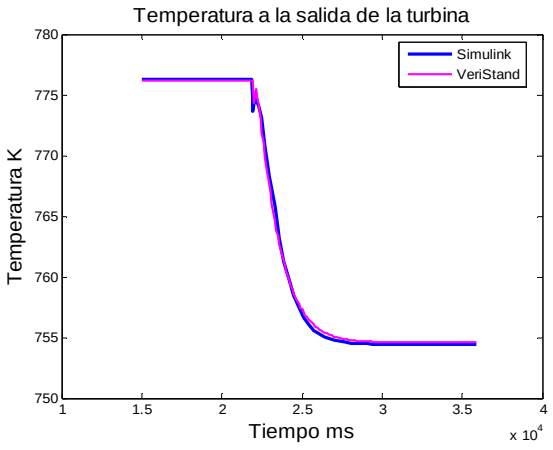












5.2 Ciclo de presión.

El próximo objetivo es comprobar que la señal de presión generada se está enviando en tiempo real. Para ello se realizarán diferentes modificaciones en las señales de entrada del modelo y se comprobará que el tiempo de ejecución de cada iteración en el modelo coincide con el tiempo de ejecución de un ciclo.

Se comprobará además que el modelo reacciona a los cambios efectuados en las diferentes variables de entrada, ya que se ejecuta ciclo a ciclo, y un cambio en cualquiera de las variables debe quedar reflejado inmediatamente en el ciclo posterior.

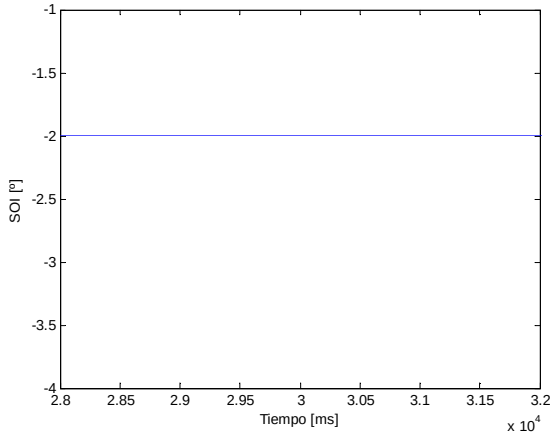
Para comprobar externamente el funcionamiento en tiempo real, se implementará un contador del número de ciclos ejecutados, comprobando que el tiempo de ejecución externo coincide con el tiempo de ejecución de dichos ciclos en labVIEW.

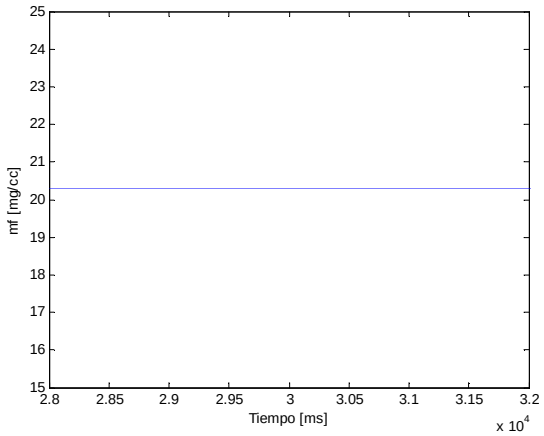
Las gráficas mostradas serán las correspondientes a las entradas del modelo de presión y a la señal de presión generada. Se mostrará además una gráfica donde se indica el inicio de cada iteración comprobando que realmente cambia al modificarse las revoluciones del motor.

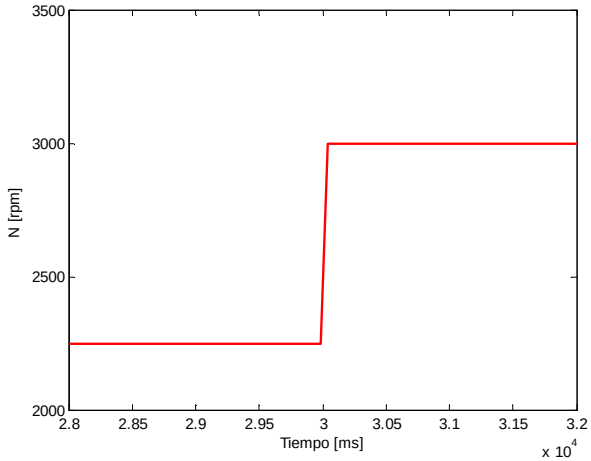
El resto de gráficas correspondientes a cambios efectuados en el inicio de la inyección y la masa de combustible pueden consultarse en el anexo 1.6.

5.2.1 Modificación del régimen de giro.

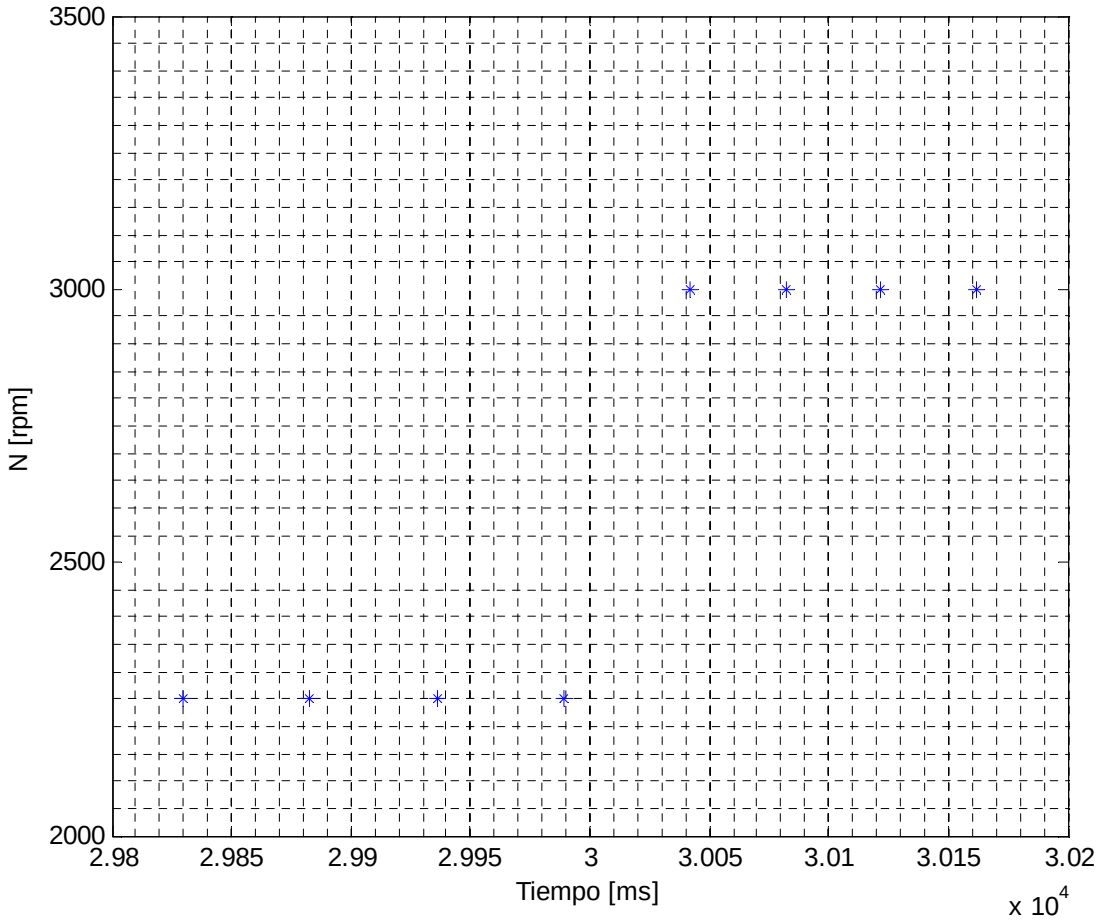
En primer lugar se muestran las entradas del modelo de presión:





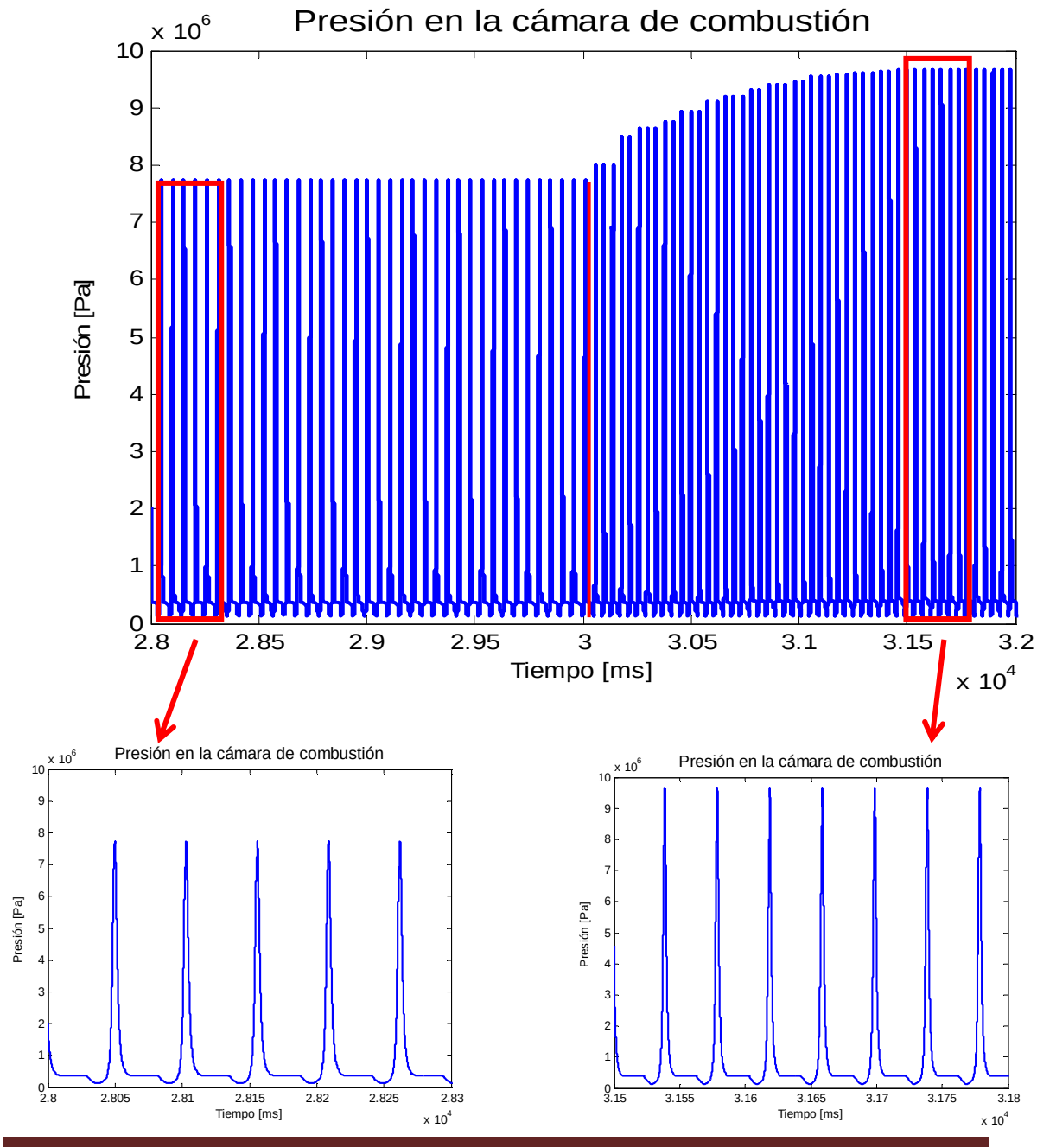


La modificación realizada en este caso es el régimen de giro del motor, el resto de entradas se mantiene constantes. En las siguientes gráficas se comprueba que realmente el sistema responde correctamente ciclo a ciclo. Inicialmente el régimen de giro del motor es 2250 rpm lo que equivale a una duración del ciclo de 53.33 milisegundos. Debido a que el programa se está ejecutando ciclo a ciclo las iteraciones se efectúan cada 53.33 milisegundos. Se puede comprobar en la siguiente gráfica, que muestra el instante de inicio de cada iteración, como realmente la modificación del régimen de giro modifica inmediatamente el régimen de ejecución del bucle. Una vez modificado el régimen de giro a 3000 rpm el tiempo de ejecución del bucle encargado del cálculo de la presión se ha modificado a 40 milisegundos.

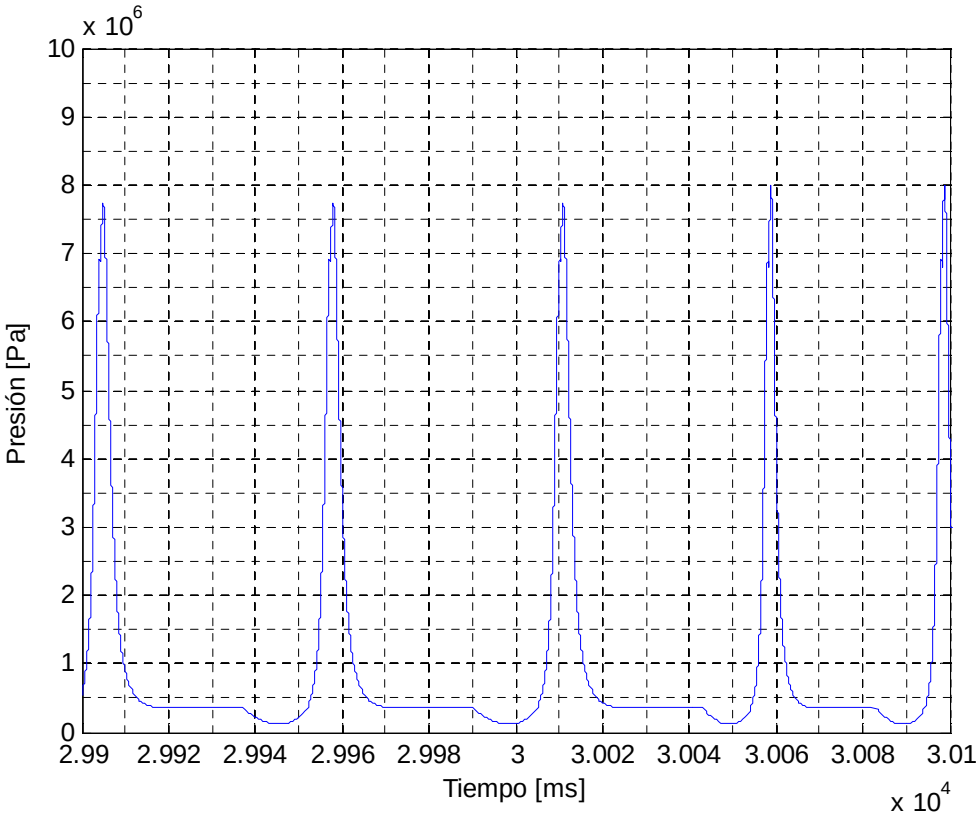


Señal de presión cámara de combustión

Las siguientes imágenes muestran la señal de presión en los dos regimenes estacionarios y en el régimen dinámico mostrando que realmente se ejecuta ciclo a ciclo. Los dos estado estacionarios están representados en el mismo intervalo de tiempo. Se comprueba que el número de ciclos obtenidos es mayor al aumentar las revoluciones del motor



Se puede comprobar como el programa se ejecuta ciclo a ciclo, el cambio efectuado afecta al siguiente ciclo tanto en su tiempo de ejecución como en el valor del mismo.



Capítulo 6. Presupuesto del proyecto

6.1.Precios descompuestos.

1. Unidad Personal

Unidad	Sección 01 Personal	Cantidad	Precio Unitario €	Precio Total €
horas	**Horas de ejecución del proyecto** El proyecto se ha realizado en el centro de motores térmicos de la Universidad Politècnica de València	350	32€	11200€
	TOTAL SECCION			11200€

2. Unidad Software

Unidad	Sección 02 Software	Cantidad	Precio Unitario €	Precio Total €
Ud.	**LabVIEW 2012 Profesional** LabVIEW es un lenguaje de programación gráfico usado por numerosos ingenieros para sistemas hardware y software de pruebas, control y diseño. LabVIEW Profesional permite la integración con cualquier dispositivo hardware y el acceso a más de 850 funciones facilitando la programación al usuario.	1	4.820 €	4.820 €

Unidad	Sección 02 Software	Cantidad	Precio Unitario €	Precio Total €
	Sistema de Desarrollo Completo de NI VeriStand formado por:	1		
	1.NI VeriStand Full Development License, Include 1 Year SSP			
Ud.		1	5500 €	5500€
	2.NI VeriStand Real-Time Engine			
Ud.		1	495 €	495€
	Ni VeriStand permite configurar pruebas en tiempo real usando modelos procedentes de múltiples entornos de programación. Permite la comunicación con tarjetas de adquisición de datos analógicas y digitales.			
Ud.	**Simulink** Simulink es una herramienta que permite construir y simular modelos matemáticos. Dispone de una interfaz de usuario gráfica para elaborar los modelos como diagramas de bloques. Se integra dentro de MATLAB permitiendo exportar los resultados de la simulación a este	1	3.000 €	3.000 €
Ud.	**Simulink Coder (Real Time Workshop)** Simulink Coder genera, a partir de los diagramas de Bloques, código que puede ser empleado en aplicaciones en tiempo real y hardware-in-the-loop test.	1	3.250 €	3.250 €
	TOTAL SECCION			17.065€

3. Unidad Hardware

Unidad	Sección 03 Hardware	Cantidad	Precio Unitario €	Precio Total €
Ud.	**NI PXIe-1078, 9-Slot 3U PXI Express Chasis** El chasis NI PXIe-1078 de nueve ranuras acepta módulos PXI Express en cada una de ellas y soporta módulos compatibles con PXI híbrido en hasta cinco ranuras. El chasis incluye relojes de referencia de 10 y 100 MHz para la sincronización y temporización así como un bus de disparo.	1	1.910 €	1.910 €
Ud.	**NI PXIe-8133** El NI PXIe-8133 es un controlador embebido para usarse en sistemas PXI Express. Dispone de un procesador Intel Core i7-820QM de cuatro núcleos y Memoria In-ROM. NI PXIe-8133 es idóneo para aplicaciones de adquisición de datos e instrumentación modular.	1	5.690 €	5.690 €

Unidad	Sección 03 Hardware	Cantidad	Precio Unitario €	Precio Total €
Ud.	**NI PXI-6713** Tarjeta de salida analógica de 8 canales con una frecuencia de muestreo de 1MHz y resolución de 12 bits. Dispone de contadores para configurar la temporización de la adquisición de datos. El máximo voltaje de entrada analógico es 10 V.	1	1.720 €	1.720 €
Ud.	**ASUS g2s-7r023g** Ordenador portátil con procesador Intel Core 2 Duo T7500 / 2.2 GHz y 160 GB de memoria. Pantalla de 17 pulgadas con resolución máxima de 1440 x 900	1	1.751 €	1.751 €
	TOTAL SECCION			11071€

6.2 Presupuesto total de ejecución material.

SECCIÓN 1 PERSONAL	11200€
SECCIÓN 2 SOFTWARE	17065€
SECCIÓN 3 HARDWARE	11071€

Total Presupuesto Ejecución Material	39336€

6.3 Presupuesto de inversión.

Total Presupuesto Ejecución Material	39336€
13% Gastos Generales	5113,68€
6% Beneficio Industrial	2360,16€
Suma de Gastos Generales y Beneficio Industrial	7473,84€
Total Presupuesto de Ejecución por Contrata	46809,84€
21% IVA	9830,0664€
Presupuesto de Inversión	56639,90€

Ascendiendo el presupuesto de inversión a la cantidad de CINCUITA Y SEIS MIL SEISCIENTOS TREINTA Y NUEVE EUROS Y NOVENTA CENTIMOS

Bibliografía

Libros y documentos Web consultados

[1] Jose Rafael Lajara Vizcaíno: 'labVIEW Entorno gráfico de programacíon'

[2] Procesos y tecnología de máquinas y motores térmicos

[3] http://www.ni.com/white-paper/2835/es/

[4] Guia de Usuario Version 2.1 Real Time Workshop

[5] http://www.ni.com/white-paper/13033/en/

[6] http://www.adi.com/technology/tech-apps/what-is-hardware-in-the-loop-simulation/

[7] Rick Bitter: LabVIEW Advance programming techniques

[8] http://www.ni.com/white-paper/9366/es/

[9] NI PXIe- 8133 User Manual

[10] Telemark University College: Hardware-in-the-loop simulation

[11] SAE Technical Paper: 2003-01-0759

[12] SAE Technical Paper: 2002-01-0371

Anexos.

1.1 Versiones compatibles de software y proceso de compilación

Versiones de software compatible obtenido pagina Web National Instruments

NI VeriStand/ Model Interface Toolkit	LabVIEW	The MathWorks, Inc. Software[1]	Microsoft Visual C++ Software Versions (Model Compilation for Windows and Phar Lap ETS Targets)	GCC Compiler (Model Compilation for VxWorks Targets)	Windows
2013 SP1	2013 and 2013 SP1	14.x, R2006a, R2006b, R2007a, R2007b, R2008a, R2008b, R2009a, R2009b, R2010a, R2010b, R2011a[4], R2011b[4], R2012a[4], R2012b[4], R2013a	6.0, .NET 2003, 2005 (Professional or Express) , 2008 (Professional or Express), 2010 (Professional or Express), Windows SDK 7.0 for Windows 7 and .NET Framework 3.5 Service Pack, Windows SDK 7.0 for Windows 7 and .NET Framework 4.0	3.4.4	Windows 8 (32-bit and 64-bit) Windows 7 (32-bit and 64-bit) Windows Vista (32-bit and 64-bit) Windows XP SP3 or later (32-bit only) Windows Server 2003R2 (32-bit only) Windows Server2008 R2 (64-bit only)
2013	2013	14.x, R2006a, R2006b, R2007a, R2007b, R2008a, R2008b, R2009a, R2009b, R2010a, R2010b, R2011a[4], R2011b[4], R2012a[4], R2012b[4],	6.0, .NET 2003, 2005 (Professional or Express) , 2008 (Professional or Express), Windows SDK for Windows 7 and .NET Framework 3.5 Service Pack 1	3.4.4	Windows 8 (32-bit and 64-bit) Windows 7 (32-bit and 64-bit) Windows Vista (32-bit and 64-bit) Windows XP SP3 or later (32-bit only) Windows Server 2003 R2 (32-bit only) Windows Server2008 R2

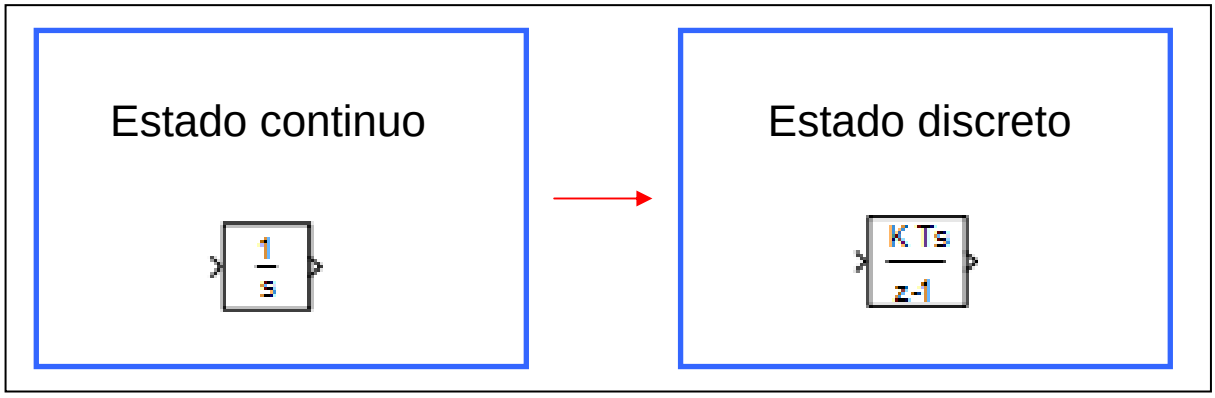
		R2013a[4]			(64-bit only)
2012	2012	14.x, R2006a, R2006b, R2007a, R2007b, R2008a, R2008b, R2009a, R2009b, R2010a, R2010b, R2011a[4], R2011b[4], R2012a[4], R2012b[4]	6.0, .NET 2003, 2005 (Professional or Express) or 2008 (Professional or Express)	3.4.4	Windows 8 (32-bit and 64-bit) Windows 7 (32-bit and 64-bit) Windows Vista (32-bit and 64-bit) Windows XP SP3 or later (32-bit only) Windows Server 2003R2 (32-bit only) Windows Server 2008 R2 (64-bit only)
2011 and 2011 SP1	2011 and 2011 SP1	14.x, R2006a, R2006b, R2007a, R2007b, R2008a, R2008b, R2009a, R2009b, R2010a, R2010b, R2011a[4]	6.0, .NET 2003, 2005 (Professional or Express) or 2008 (Professional or Express)	3.4.4	Windows 7 (32-bit and 64-bit) Windows Vista (32-bit and 64-bit) Windows XP SP2 or later (32-bit only) Windows Server 2003 R2 (32-bit only) Windows Server 2008 R2 (64-bit only)
2010	2010 and 2010 SP1	14.x, R2006a, R2006b, R2007a, R2007b, R2008a, R2008b, R2009a, R2009b, R2010a	6.0, .NET 2003, 2005 (Professional or Express) or 2008 (Professional or Express)	3.4.4	Windows7 (32-bit and 64-bit) Windows Vista (32-bit and 64-bit) Windows XP SP2 or later (32-bit only) Windows Server 2003R2 (32-bit only) Windows Server2008 R2 (64-bit only)
2009	2009 and 2009 SP1	14.x, R2006a, R2006b, R2007a,	6.0, .NET 2003, 2005 (Professional or Express) or 2008 (Professional or Express)	3.4.4	Windows 7 (32-bit and 64-bit) Windows Vista (32-bit and 64-

		R2007b, R2008a, R2008b, R2009a			bit) Windows XP SP2 or later (32-bit only) Windows Server 2003R2 (32-bit only) Windows Server 2008 R2 (64-bit only)

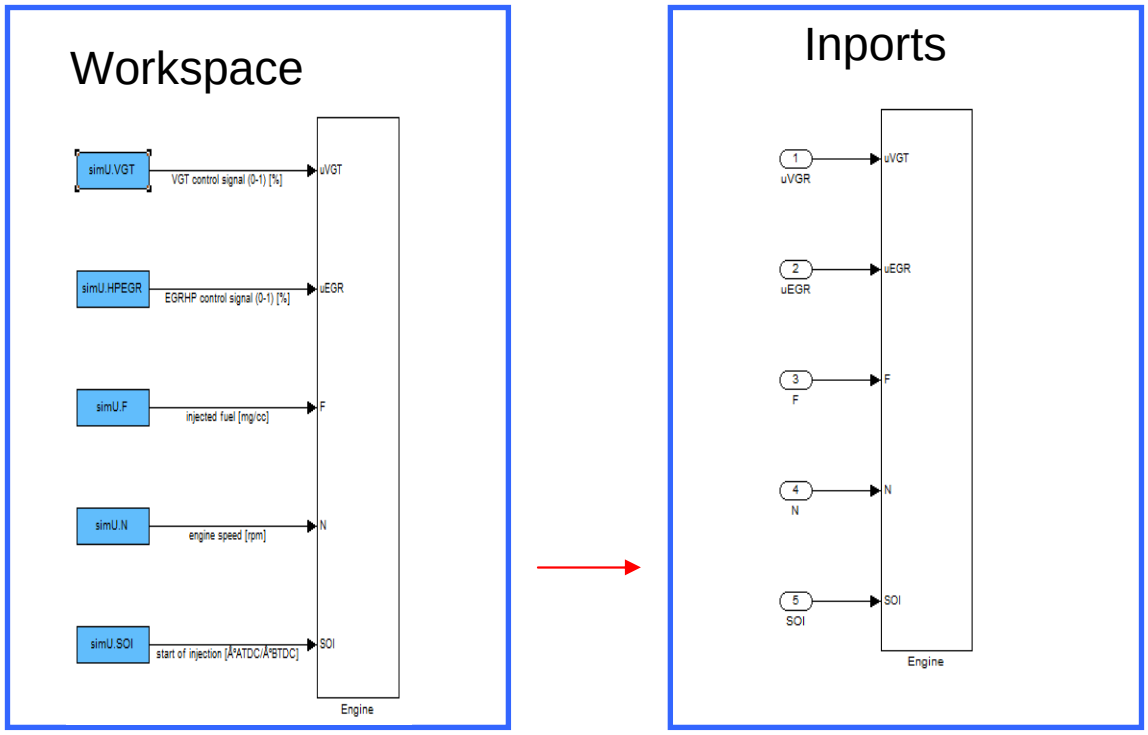
Guia del proceso de compilación de un modelo Simulink y ejecución en NI VeriStand

Cambios a efectuar antes de la compilación del modelo

Para poder crear el modelo compilado y ejecutarlo en VeriStand con éxito hay que modificar parte del código. En primer lugar los bloques que definen estados continuos deben ser modificados por bloques de estados discretos: en el modelo a compilar los integradores deben ser modificados



Por otra parte las entradas del modelo no deben realizarse desde el Workspace de Matlab, deben ser bloques *Inports.*

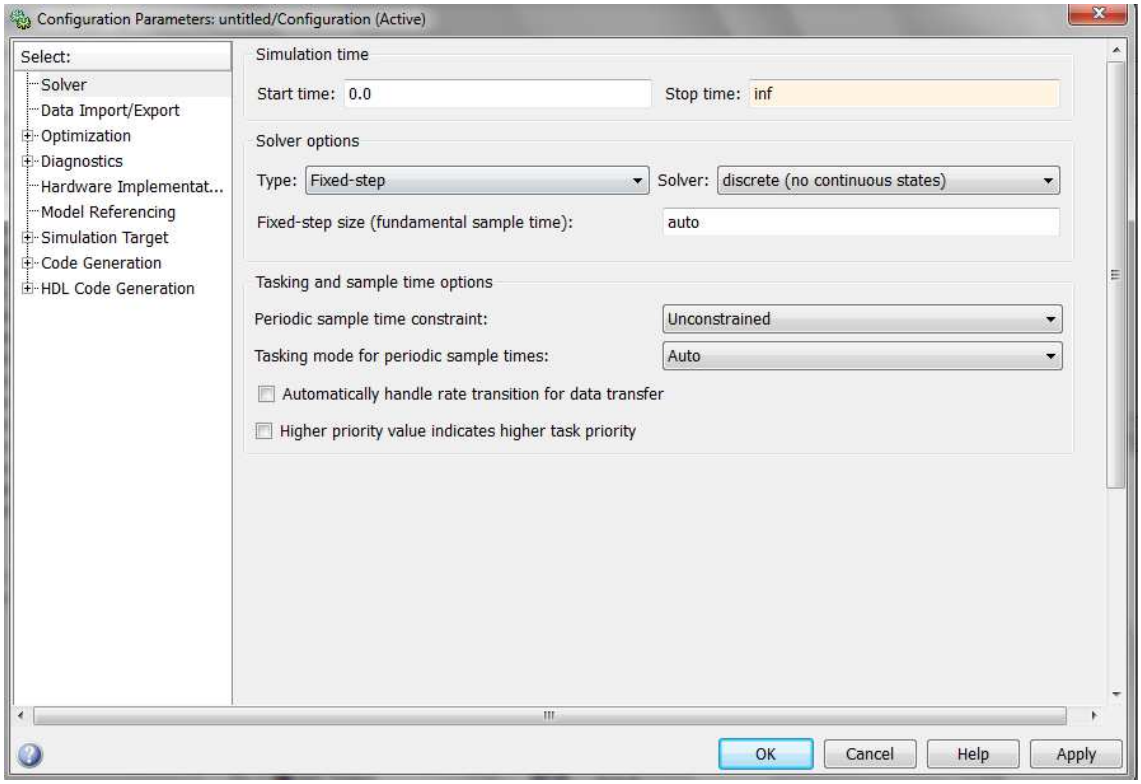


Proceso de compilación

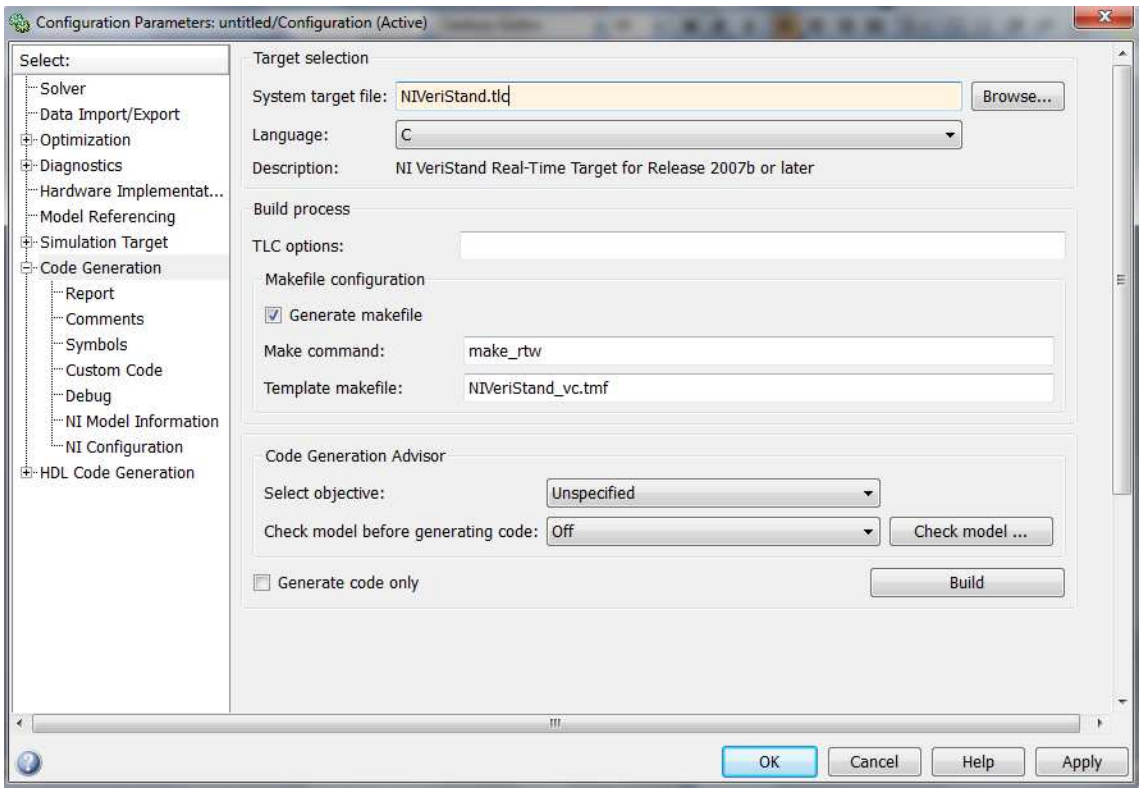
Una vez efectuados los cambios requeridos se procede al proceso de compilación. Ejecutando mex –setup en MATLAB software y seleccionando el compilador previamente instalado, en este caso Visual Studio 2008, se procede a la construcción del modelo compilado. Se inicia Simulink y se carga el modelo a compilar. Procediendo posteriormente a la configuración de los parámetros de la simulación.

Dentro de la pestaña Solver se selecciona las siguientes opciones

- Stop time: inf
- Type: Fixed-step
- Solver: discrete (no continous states)

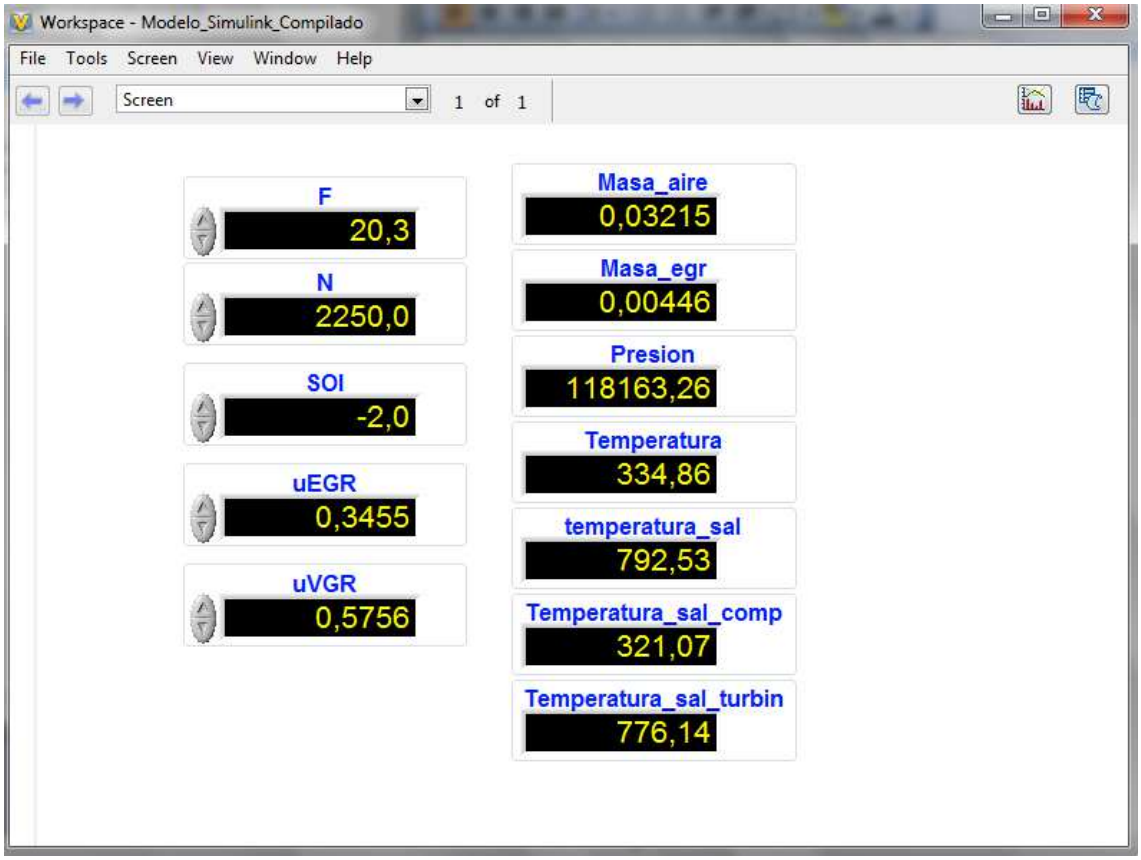


Posteriormente en Code Generator se selecciona la tarjeta NIVeriStand.tlc-NI Real-Time Target de entre las diferentes opciones y finalmente se pulsa Build para empezar a construir el modelo compilado.



Después del proceso de construcción el mensaje, *### Successful complexion of Real-Time Workshop build procedure for model* , aparecera en pantalla indicando que se ha completado el proceso de compilación con éxito.

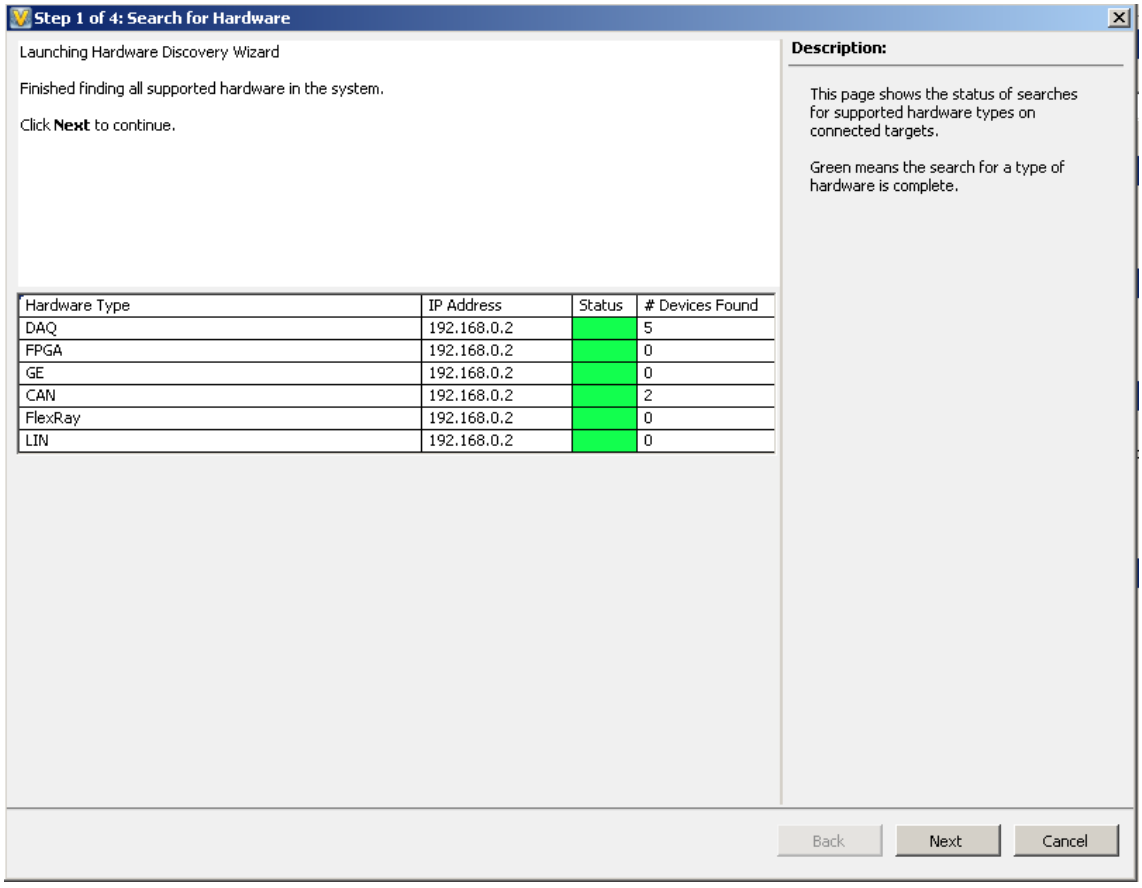
El archivo .dll creado debe añadirse a Ni VeriStand, donde se ejecutará. En el System Explorer de Ni VeriStand en el apartado Model, la opción Add Simulation Model permite cargar los diferentes modelos compilados. Una vez añadido el modelo y configurados los aspectos referentes ejecución del mismo. Se cierra el System Explorer y se accede al WorkSpace de Ni VeriStand. En el Workspace se pueden editar las entradas del modelo observar el valor de las salidas.


Workspace - Modelo_Simulink_Compilado
File Tools Screen View Window Help
Screen
1 of 1
F
20,3
N
2250,0
SOI
-2,0
uEGR
0,3455
uVGR
0,5756
Masa_aire
0,03215
Masa_egr
0,00446
Presion
118163,26
Temperatura
334,86
temperatura_sal
792,53
Temperatura_sal_comp
321,07
Temperatura_sal_turbin
776,14

1.3 Ejecución del modelo compilado en el PXI utilizando Ni VeriStand.

VeriStand permite ejecutar modelos en el PXI y conectar las salidas de los modelos con canales analógicos. Se selecciona la tarjeta en la que se quiere ejecutar el modelo en este caso será el PXI- 8311, indicando el sistema operativo de la misma y la dirección IP indicada en el MAX.

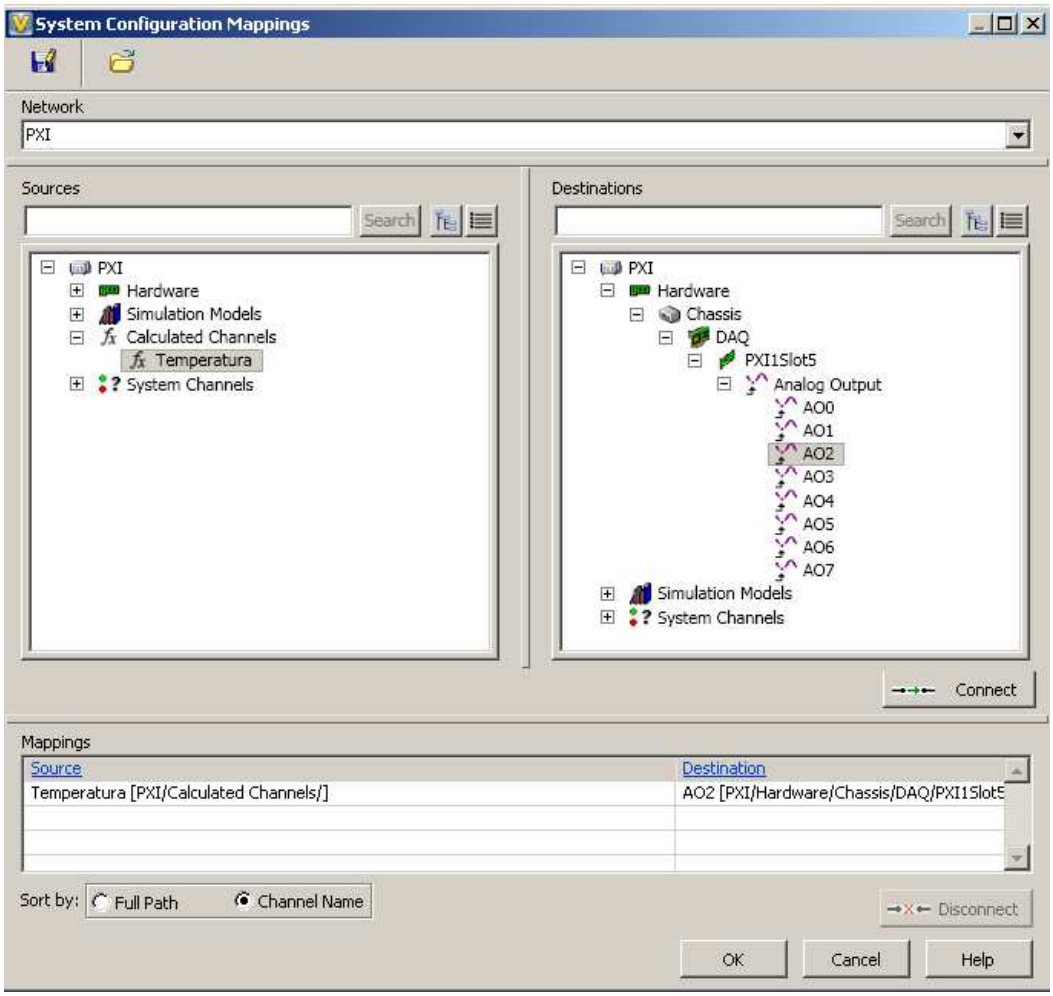
Ni Veristand proporciona una herramienta llamada Hardware Discovery Wizard, para detectar los dispositivos hardware en la tarjeta con la que se está trabajando y automáticamente añadirlos al System Definition, la alternativa seria introducirlos manualmente uno por uno.



En el dispositivo de adquisición de datos previamente detectado y añadido con el Hardware Discovery Wizard, añadiremos los diferentes canales. En realidad en este caso solo sería necesario añadir un canal pues solo vamos a conectar una de las salidas del modelo con un canal analógico.

Mapping System channels

Ni VeriStand proporciona una herramienta que permite conectar fácilmente canales. Esta herramienta permite conectar rápidamente modelos previamente compilados con canales físicos de entrada salida. De esta manera las salidas de modelos compilados Simulink (archivo .dll) o Labview (archivo .lvmodel) pueden ser salidas analógicas conectando ambas. En este caso una salida del modelo Simulink (temperatura a la entrada de la cámara de combustión) serán una salida analógica. Nuestra salida analógica debe estar en el rango [-10 ,10] V de manera que para que no se produzca saturación multiplicaremos la temperatura por 0.01 mediante la herramienta Calculated Channels y será este canal el que conectaremos con la salida analógica.



1.4Proceso de compilación de modelos Labview

VeriStand interactúa con modelos procedentes de múltiples entornos de programación entre ellos labVIEW. Permite convertir labVIEW VIs en archivos compilados .lvmodel. Es necesario que las versiones utilizadas Labview y VeriStand sean compatibles, de no ser así labVIEW no permitirá compilar el modelo.

Existe un paralelismo entre la compilación de modelos procedentes de Simulink y labVIEW. Al igual que en Simulink, Ni VeriStand reconoce inports outports y parameters al añadir los modelos labVIEW compilados. El archivo VI a compilar tiene en la parte derecha superior lo que se conoce como panel conector, se deben asignar a el las entradas y las salidas del modelo para que VeriStand las reconozca, si no se asignaran el modelo compilado sería como una caja negra a la cual no se tiene acceso, ni para modificar las entradas ni para leer las salidas de la misma. Los controles e indicadores se asignan al panel conector, y se configuran los controles los cuales pueden ser parámetros en NI Veristand o entradas. Al igual que en Simulink se importará el valor por defecto de los diferentes parámetros del modelo pero no el de las entradas y salidas cuyo valor será 0. El tipo de datos soportados para los controles y indicadores del modelo son: valores booleanos, numéricos, vectores de una dimensión tanto booleanos como numéricos y clusters, cualquier otro tipo de dato no válido dará un error en la compilación, por ejemplo si se trata de compilar un modelo cuya entrada es un vector de dos dimensiones.

Otra analogía con la compilación de modelos en Simulink es que en el caso de labVIEW las entradas y salidas de los subVIs no están disponibles como entradas y salidas que el usuario pueda modificar una vez compilado el modelo al igual que ocurría con los subsistemas en Simulink.

Un aspecto a tener en cuenta en el proceso de compilación es el uso de vectores como entradas ya que estos tienen que estar inicializados, de no ser así se tendrá un error al añadir el modelo en Ni VeriStand, ya que los parámetros de entrada serán inválidos. Por ello en el menú edición está disponible la opción Make Current Values default, activándolo evitaremos el error aunque el valor importado será cero y no el valor por defecto del archivo vi.

Otro aspecto a tener el cuenta es el uso de bucles en los modelos a compilar, VeriStand engine proporciona bucles Timed Loops para ejecutar los modelos, por lo tanto el modelo labVIEW no debe contener ningún bucle.

Teniendo presente los aspectos anteriormente mencionados se puede realizar ya la compilación del modelo desde labVIEW.

1.5. Tipo de señales y características de las tarjetas de salida analógicas.

1.5.1 Tipo de señales

Las señales eléctricas pueden ser

- **Analógicas**: señales continuas que pueden tomar cualquier valor en cualquier instante de tiempo

- **Digitales**: señales discretas que recogen solo determinados valores para todo el tiempo.

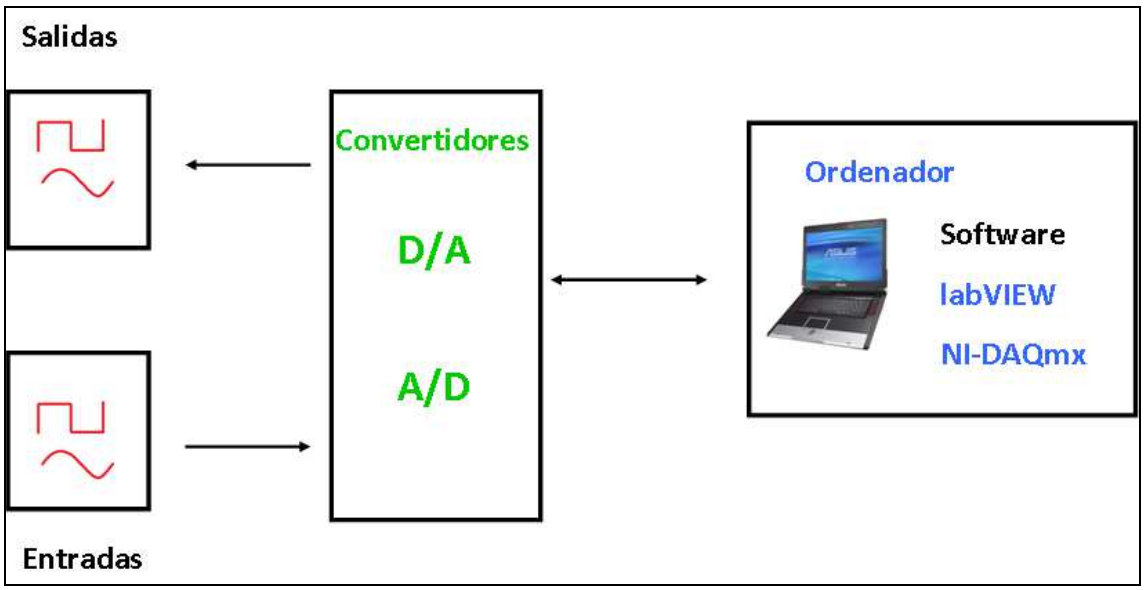
1.5.1 Conversión A/D y D/A

Conversión análogica-digital ADC

Las magnitudes físicas son analógicas pero el procesado de señales en los ordenadores se realiza de forma digital, las señales analógicas de entrada deberán ser muestreadas y representadas digitalmente, muestrear significa discretizar nuestra señal, si la adquisición es continua el tiempo entre cada muestra será el mismo y dependerá de la frecuencia de muestreo. Tras el muestreo, el rango de voltaje de entrada de la tarjeta de adquisición se dividirá en diversos niveles, cada uno de los cuales tendrá asociado un código binario. Más niveles se traduce en un mayor número de bits utilizados pero en más resolución en la conversión analógico-digital. Al disponer de un número finito de niveles el valor real de la muestra difiere del valor asignado tras la digitalización. La diferencia entre ambos valores se conoce como error de discretización.

Conversión digital-analógica DAC

Los canales de salida analógicos disponen de convertidores digitales analógicos, ya que los ordenadores trabajan con código binario. Las diferentes entradas digitales formadas por un número de bits son transformadas a salidas analógicas usando este tipo de convertidores.



1.5.2 Tarjetas Adquisición.

Las tarjetas de adquisición de datos se encargan de la comunicación con el ordenador y por lo tanto disponen de convertidores ADC y DAC. Las características más importantes de las tarjetas de adquisición son las siguientes.

- **Número de canales analógicos**: Las tarjetas de adquisición de datos permiten adquirir y generar medidas analógicas, el número de canales indica la cantidad de medidas que se pueden realizar con una misma tarjeta

- **Velocidad de muestreo**: indica el número de muestras digitales adquiridas , mayor velocidad de muestreo implica mayor número de muestras procesadas.

- **Rango de entrada**: Rango de voltaje entre el cual debe estar la señal de entrada y la de salida. Si enviamos valores no contenidos en ese rango se producirá una saturación.

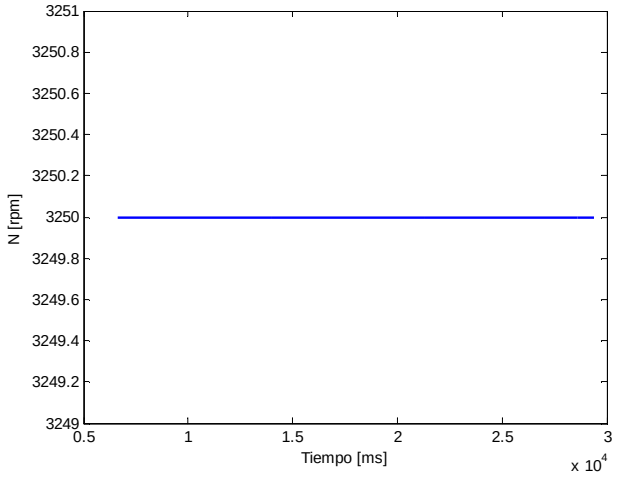
- **Resolución**: La resolución es el número de bits que utiliza el convertidor para representar la muestra. Las variaciones en la señal que la tarjeta puede detectar son menores cuanto mayor es el numero de bits de esta.

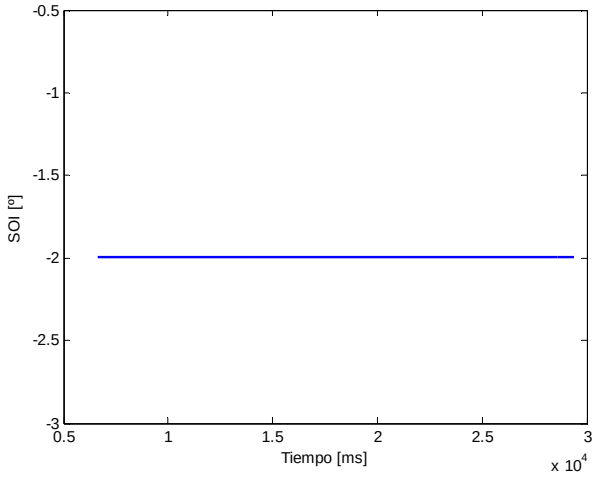
1.6 Modelo de Presión

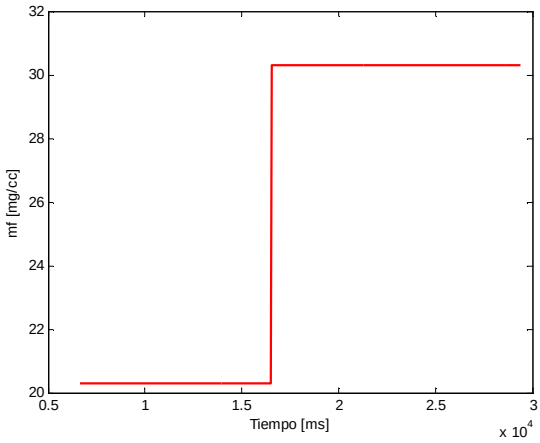
En el siguiente apartado se muestra la gráfica de la señal de presión al modificar la masa de combustible. De la misma manera que el apartado resultados y conclusiones se ha mostrado la señal de presión al modificar el régimen de giro del motor. Se comprueba que la señal responde correctamente a las modificaciones y que efectivamente se ejecuta ciclo a ciclo.

1.6.1 Modificación de la cantidad de combustible

En primer lugar se muestrán las entradas del modelo

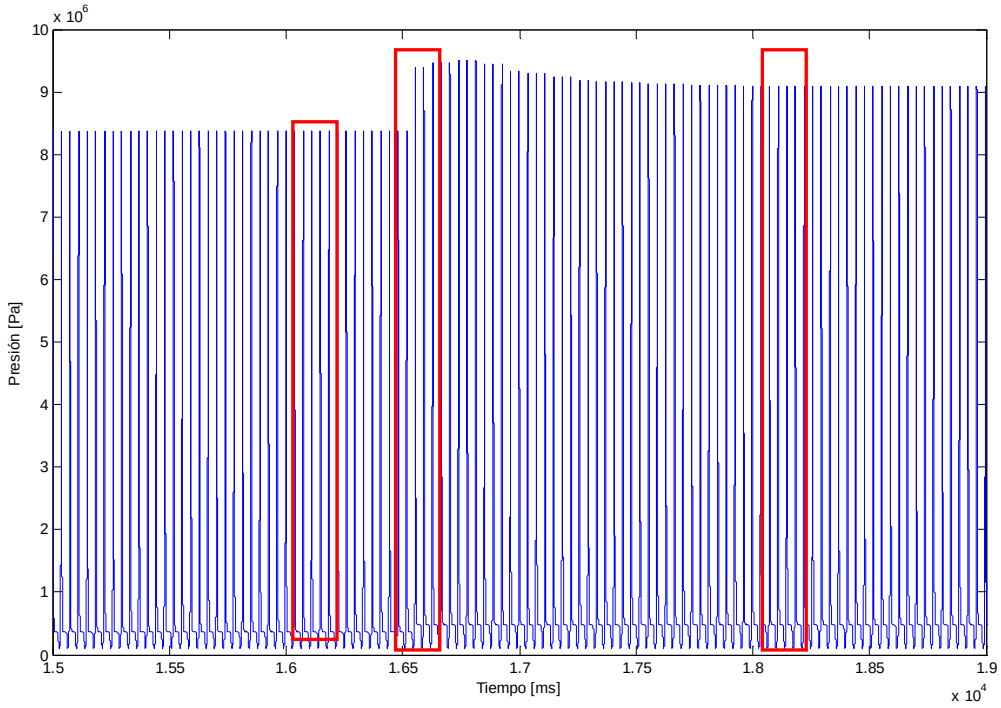




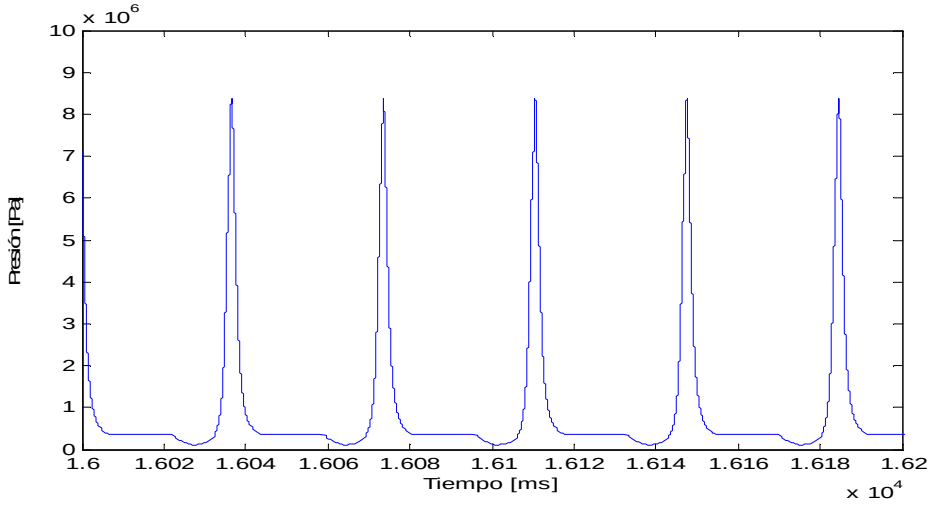


En las siguientes gráficas se muestra la señal de presión en los dos estados estáticos y en el régimen dinámico.

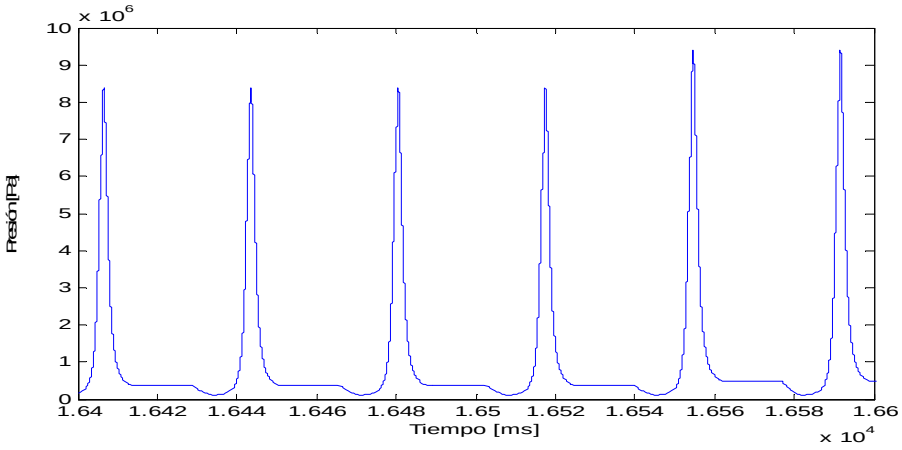
Se comprueba como en todos los casos la duración del ciclo es la misma, y que al cambio del combustible inyectado, efectuado en el instante T=16502 milisegundos, la señal responde inmediatamente al siguiente ciclo.



Estado inicial



Regimen dinámico



Estado estático