Capítulo 5
Programación del algoritmo en LabVIEW

En este capítulo se describen las funciones que se emplearon para implementar el control PID wavenet en LabVIEW. El algoritmo wavenet fue implementado en LabVIEW para efectuar simulaciones numéricas a nivel de PC en Windows. Posteriormente, se desarrolla la programación de las funciones de adquisición y salida de datos del módulo CompactRIO y el código se implementó en el FPGA para efectuar pruebas con el simulador de procesos PCS327. Los algoritmos programados son los obtenidos en el capítulo anterior para el controlador PID y la red wavenet.

5.1 Programación del algoritmo y código en LabVIEW

La implementación del controlador PID Wavenet se realiza empleando tres programas intercomunicados. El programa básico controla la adquisición y emisión de datos en el FPGA. Un segundo programa, ejecutado en el RT Host, es el propio PID wavenet y se encarga de generar la señal de control que requiere la planta. El tercer programa se ejecuta en Windows, y se encarga de las funciones de despliegue y almacenamiento de los datos generados durante la ejecución. Estos tres programas funcionan simultáneamente, y se comunican mediante el uso de nodos de entrada/salida y variables compartidas.

5.1.1. Programación en el FPGA cRIO-9102

La implementación del programa requiere comunicación con los módulos de entrada y salida del módulo cRIO-9102 para la lectura y envío de datos. El programa que se describe a continuación obtiene los datos de calibración de las tarjetas, los cuales se usan para

hacer las conversiones de valores binarios a valores nominales de voltaje requeridos para el control, como el primer paso en la programación del código.

El programa emplea una estructura tipo secuencia en que primero se declara que no se tienen los datos de calibración listos, después se obtienen los datos de calibración con un FPGA I/O Property node y finalmente se avisa al programa de control que ya se tienen los datos listos.

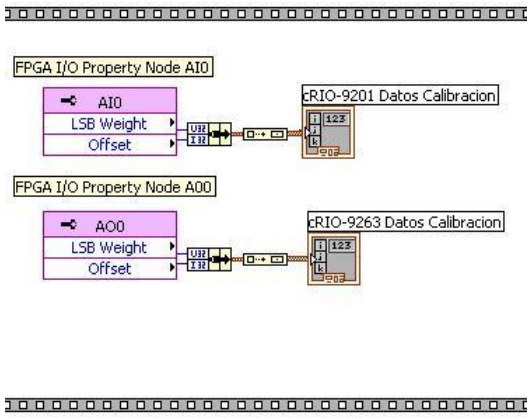


Figura 5.1. Nodos de propiedad FPGA I/O.

El segundo paso en la secuencia es el ciclo de lectura y escritura de datos. Dentro de un ciclo while se introduce una nueva secuencia de tres pasos: un retraso de 10 [ns], las funciones de lectura y escritura y una interrupción que reinicia el ciclo. El retraso se agrega como un elemento de seguridad. En este programa se emplea una entrada analógica de datos, la AI0 del módulo 9201, que envía la lectura del estado de la planta al programa de control. La señal de control generada por el programa de control se transmite a la planta a través de la salida analógica AO0 del módulo 9263. El último elemento del ciclo es una interrupción, empleada para avisar al programa de control que los datos nuevos están disponibles, que se envían los que se tenían, y se declara la condición “ready” para el envío y recepción de una nueva pareja de datos.

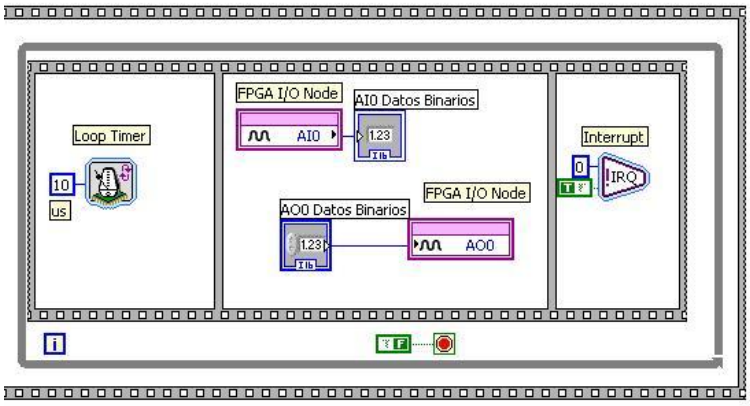


Figura 5.2. Ciclo de lectura y escritura de datos.

Este programa se compila y posteriormente se carga en el FPGA.

5.1.2. Programación en el HOST cRIO 9002

La rutina principal de control se ejecuta en el HOST cRIO9002. De inicio se requiere una etapa de comunicación con las variables que se leen y escriben en el FPGA, así como variables compartidas entre el HOST y la interfaz con el usuario en Windows. Este VI contiene también los algoritmos de control expuestos en el Capítulo 4, cuya programación es explicada a continuación.

En el diagrama de bloques del programa se crea una referencia al VI en el FPGA, definiendo el archivo de bits usado para determinar los controles disponibles en el FPGA y el dispositivo que se va a usar como execution target. La referencia a un VI debe cerrarse forzosamente para liberar recursos de hardware y evitar errores en la implementación de un programa posterior. La función Close FPGA VI Reference detiene el VI que corre en el dispositivo RIO y limpia la memoria usada por la interfaz del FPGA.

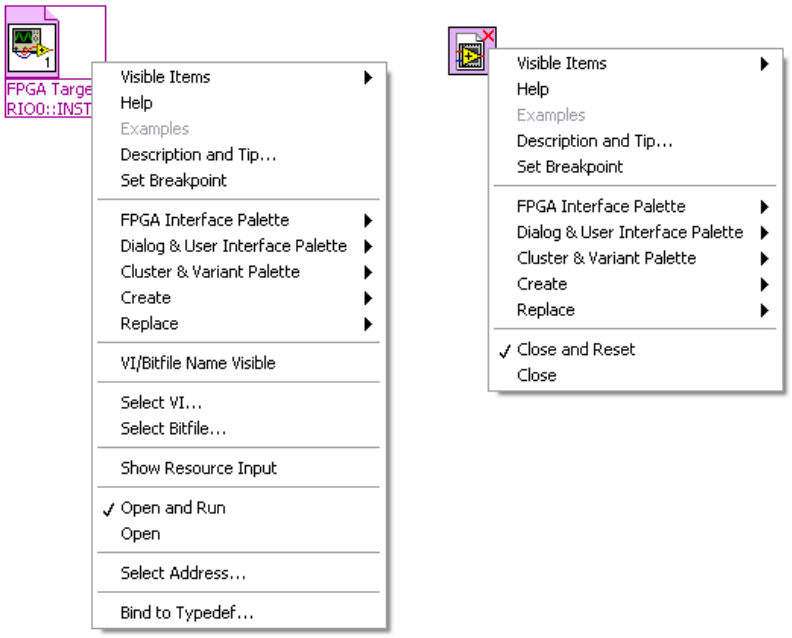


Figura 5.3. Open FPGA VI Reference y Close FPGA VI Reference.

La manipulación de datos para generar la señal de control y comunicarla al FPGA requiere una función de lectura/escritura: Read/Write Control (FPGA Interface).

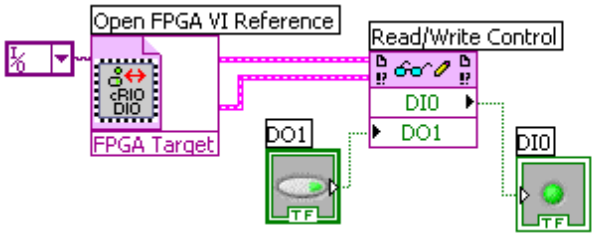
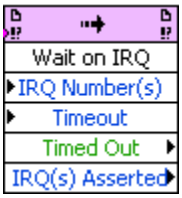
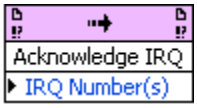


Figura 5.4. Read/Write Control.

El ciclo de control emplea una interrupción para sincronizar el VI del FPGA con el VI del HOST. La función Invoke Node (FPGA Interface), configurada en Wait on IRQ, espera el número de interrupción establecido en el VI del FPGA (o genera una señal de error si se excede el tiempo establecido de espera). Esta función avisa cuando recibe la interrupción y después envía una señal de aviso. Posteriormente se manda un reconocimiento por medio de otro Invoke Node configurado en Acknowledge IRQ para avisar a la fuente de interrupciones que se ha terminado el procesamiento de los datos actuales y que el VI de control en el HOST está listo para recibir un nuevo bloque de información.





Wait on IRQ Acknowledge IRQ

Figura 5.5. Funciones de interrupción.

El módulo cRIO 9201 entrega valores binarios tras efectuar una lectura de voltaje. Antes de procesar los datos es necesario convertirlos en valores nominales de voltaje. La conversión se efectúa por medio de un VI que recibe como parámetros de entrada el modelo de la tarjeta 9201, los datos de calibración de la tarjeta, y el valor binario proveniente de la misma; la salida es el valor nominal de voltaje que entra al control PID como variable de proceso. La Figura 5.6 muestra la ubicación de este VI en el diagrama de bloques.

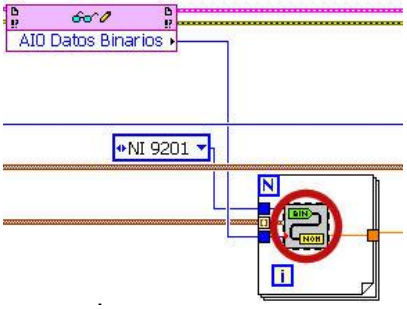


Figura 5.6. VI Binary to Nominal.vi en el ciclo de control.

De manera análoga, el módulo de salida cRIO 9263 requiere sean enviados valores binarios, por lo que es necesario convertir los valores nominales de voltaje a valores binarios. Para esto se incluye el subVI Nominal to Binary.vi que recibe como parámetros de entrada el modelo del módulo al cual se van a mandar estos datos, los datos de calibración del módulo, y el valor nominal de voltaje que se quiere convertir. La Figura 5.7 contiene la forma de incluir este VI en el ciclo de control.

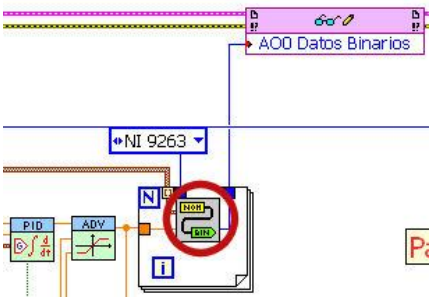


Figura 5.7. VI Binary to Nominal.vi en el ciclo de control.

El valor de voltaje obtenido de la planta se utiliza en la ley de control descrita en (4.5), que se implementa como se muestra en la Figura 5.8. La señal leída del voltaje de la planta se compara con la función de referencia (definida dentro de este mismo programa) para generar la señal de error con que actúa el PID. Se emplean nodos de corrimiento en la estructura iterativa del programa para contener los valores pasados del error de seguimiento y de la entrada a la planta.

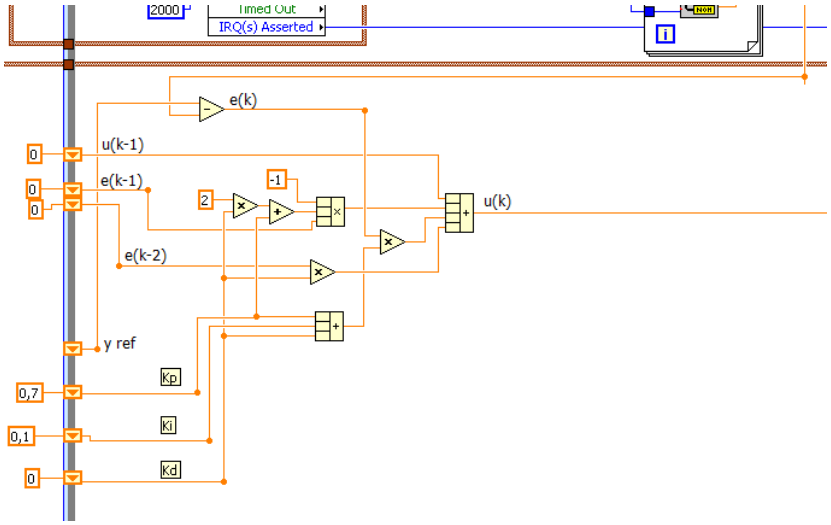


Figura 5.8. Implementación de la ley de control en el VI.

Se crea una variable compartida del tipo arreglo de 1-D en la estructura del proyecto, llamada Variables Control, cuya función es almacenar los valores de parámetros y variables de control para la comunicación con la interfaz de usuario. Este arreglo contiene la iteración (tick) del ciclo de control, el valor de referencia, la variable del proceso, la señal de control, una señal que indica si el ciclo de control está trabajando dentro del ciclo de 5 [ms] asignado a la tarea, el valor de la estimación de la red wavenet y las ganancias del controlador PID.

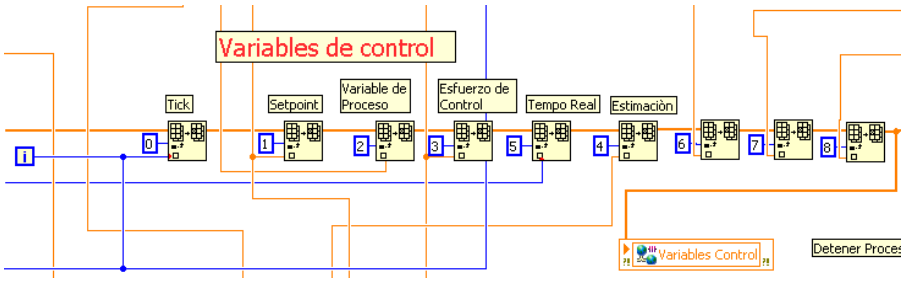


Figura 5.9. Variables de control en el programa en el FPGA.

La red wavenet empleada para actualizar los parámetros del controlador recibe la lectura del voltaje del proceso como una de las variables de entrada que requiere para la sintonización. La red wavenet requiere los vectores A,B y W característicos de la red wavenet, el vector Ψ y su derivada, dos parámetros C y D del filtro IIR y la salida anterior de la wavenet z[k-1]. Todo esto se incluye dentro de un ciclo FOR con un control numérico para elegir la cantidad de épocas de aprendizaje que realiza la red.

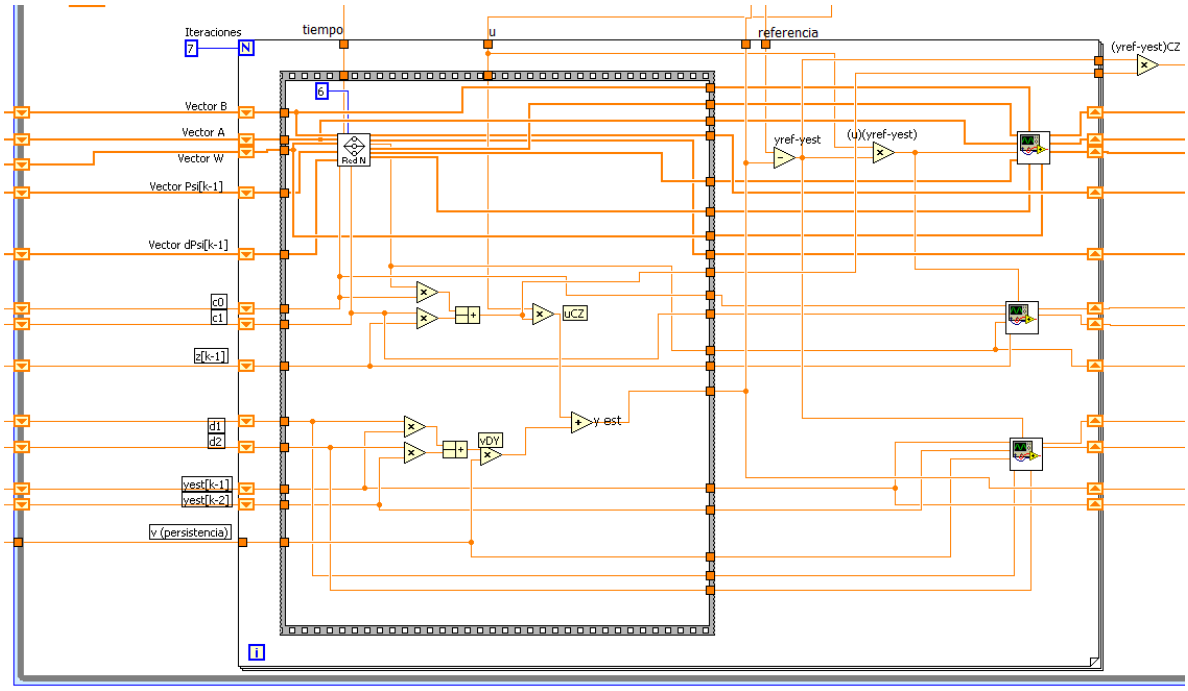


Figura 5.10. Implementación del algoritmo wavenet.

Dentro de este ciclo se incluye un subVI que incluye la red, y varias estructuras que forman parte del algoritmo de aprendizaje. La Figura 5.11 muestra el código de la red wavenet. Se utiliza un ciclo FOR, en el cual las iteraciones están determinadas por la cantidad de neuronas, para calcular la aproximación preliminar z[k] y cada valor de los vectores auxiliares del algoritmo de actualización de la red neuronal. Dentro de este ciclo

FOR existe un subprograma que calcula el valor de cada señal wavelet y su derivada parcial respecto a b.

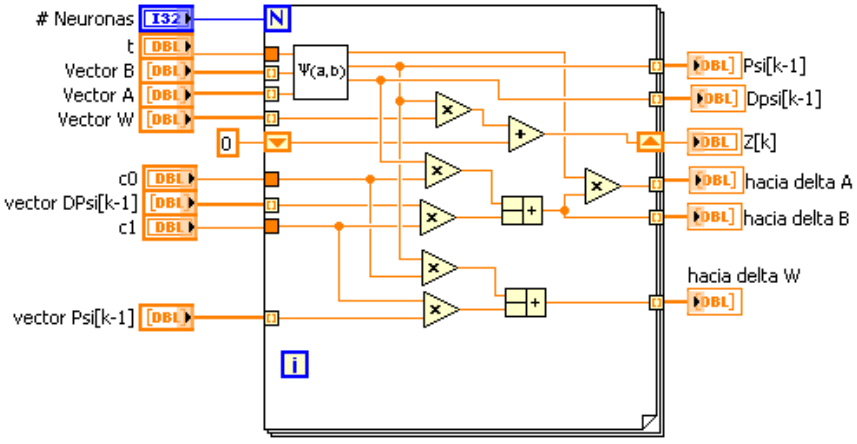


Figura 5.11. SubVI Red wavenet.

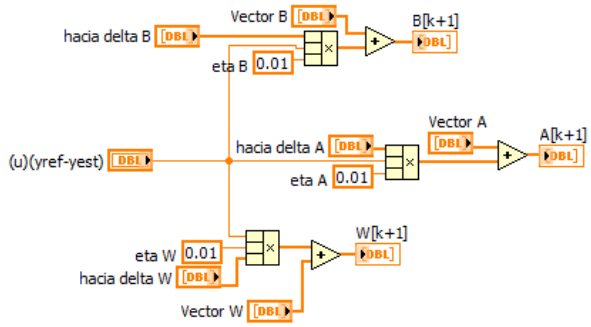


Figura 5.12. Subrutina de aprendizaje de los vectores A, B y W de la red neuronal.

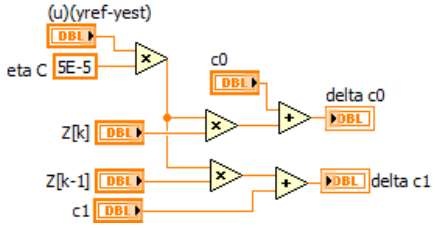


Figura 5.13. Subrutina de aprendizaje de los parámetros C del filtro IIR.

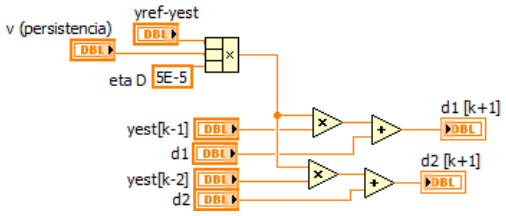


Figura 5.14. Subrutina de aprendizaje de los parámetros D del filtro IIR.

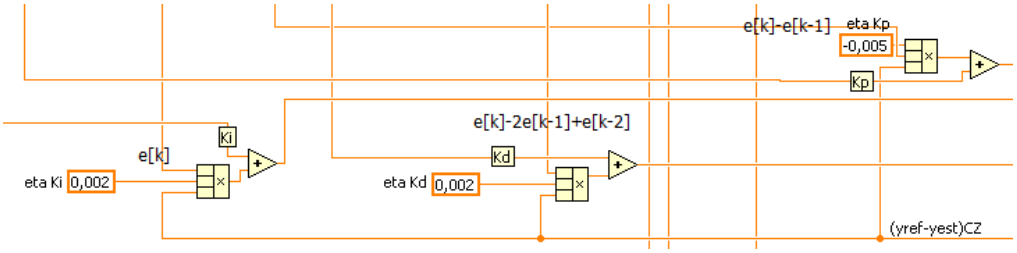


Figura 5.15. Subrutina de aprendizaje de las ganancias del control PID.

El diagrama de bloques del código completo se anexa a continuación. Se destacan las tres estructuras principales: el control PID descrito en primer lugar, la etapa de comunicación de datos por medio de variables compartidas y la red neuronal wavenet. La combinación de estos tres elementos es la implementación del algoritmo descrito en el Capítulo 4.

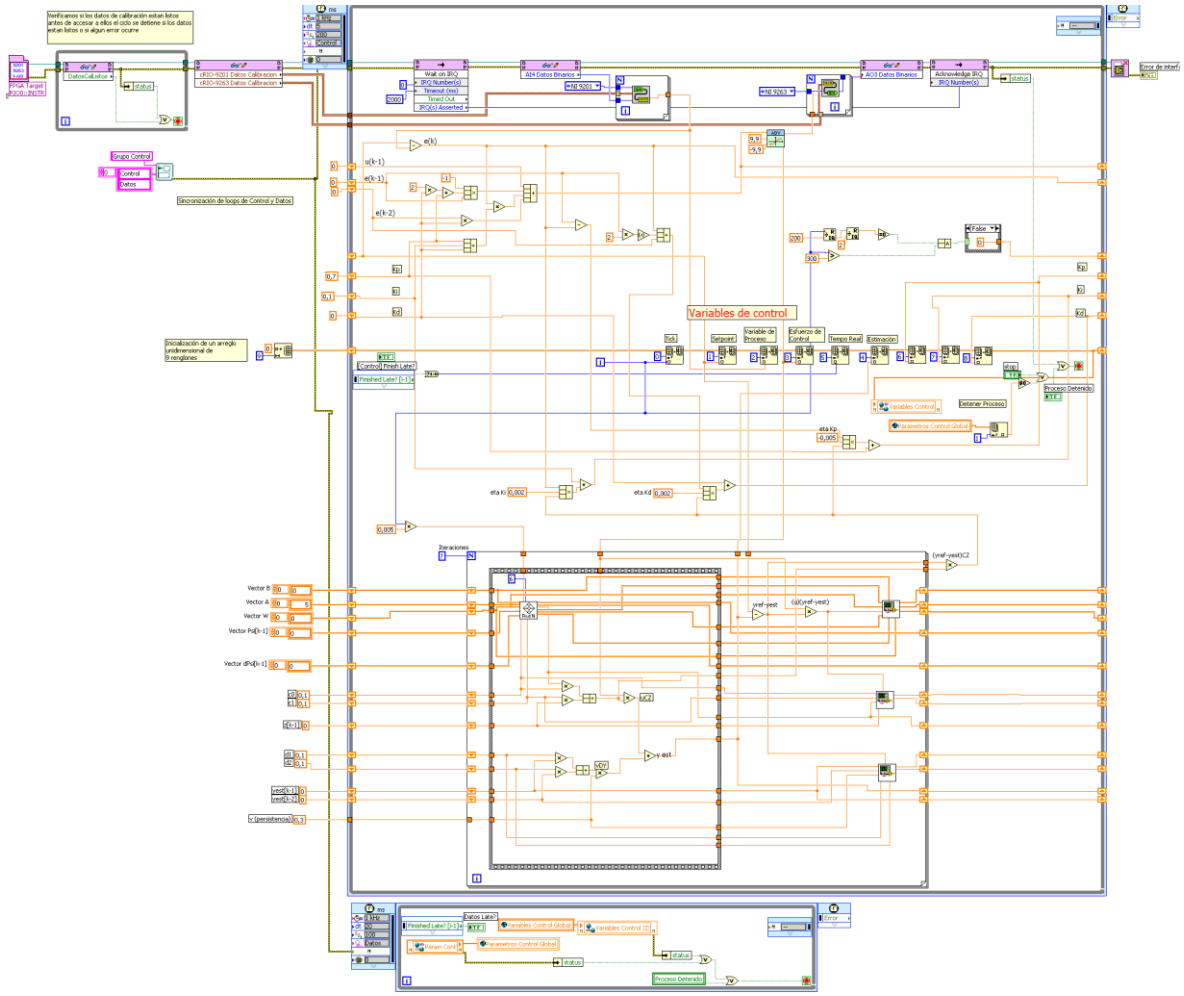


Figura 5.16. Diagrama de bloques del VI de control en el HOST cRIO.

5.1.3 Programación de la interfaz de usuario

La interfaz con el usuario se ejecuta en Windows. Cumple tres propósitos principales: recibir del usuario la referencia, desplegar los datos obtenidos y escribirlos en un archivo de texto. La Figura 5.17 muestra el panel de usuario, provisto de un control numérico para cambiar la señal de referencia:

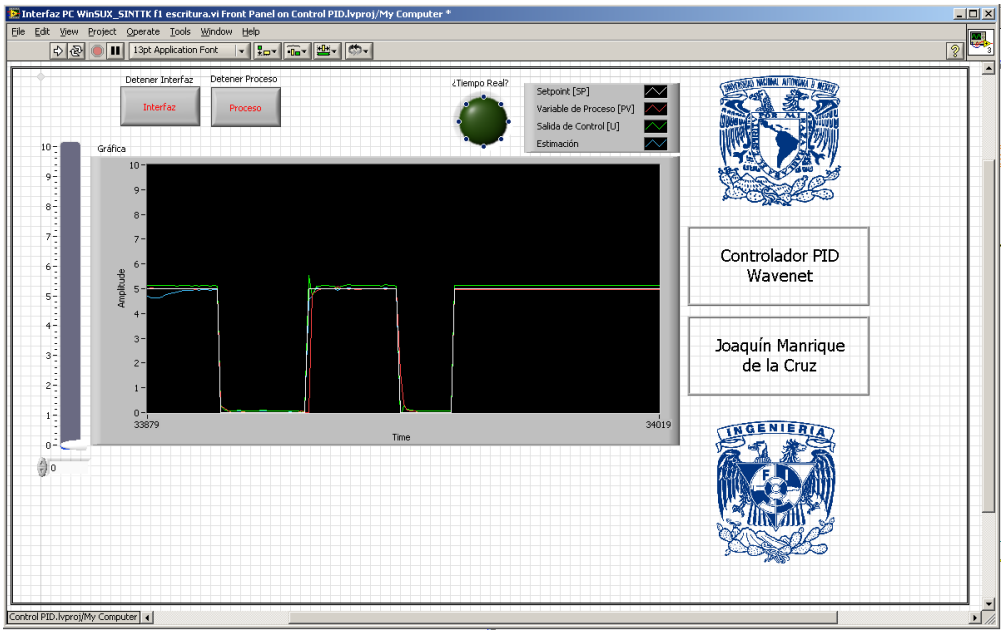


Figura 5.17. Panel de la interfaz de usuario.

Esta VI recibe tres entradas diferentes del usuario: la señal de referencia, la señal de paro del controlador en el HOST cRIO y la señal de paro de la interfaz. La señal de referencia y la señal de paro del controlador se transmiten por medio de variables compartidas. Del VI en el HOST cRIO se leen la señal de control, la variable del proceso y la señal generada por la wavenet, todas para su escritura en un archivo de texto, del cual posteriormente se reconstruye la gráfica de comportamiento del sistema. Aunque este programa tiene la capacidad de que el usuario controle la variable de referencia, en las pruebas se deshabilita esta opción y la señal de referencia se genera directamente en el FPGA.

Se utiliza la variable global Param Cont para transmitir dos señales al programa en el FPGA: señal de referencia y el comando de detener el proceso.

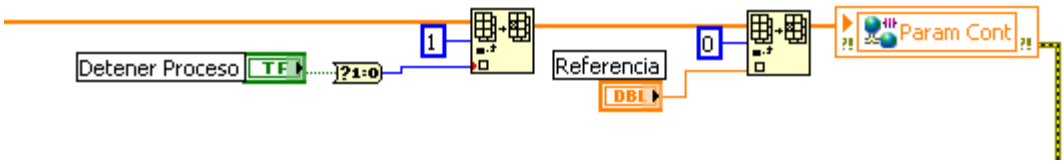


Figura 5.18. Parámetros de la interfaz de usuario.

El programa lee la variable compartida Variables Control, que contiene los valores de la señal de referencia, la lectura del proceso, la señal de control, la estimación de la wavenet y la señal de confirmación de tiempo real. Estas variables, generadas por el FPGA se despliegan en una gráfica en el panel frontal de la aplicación en windows.

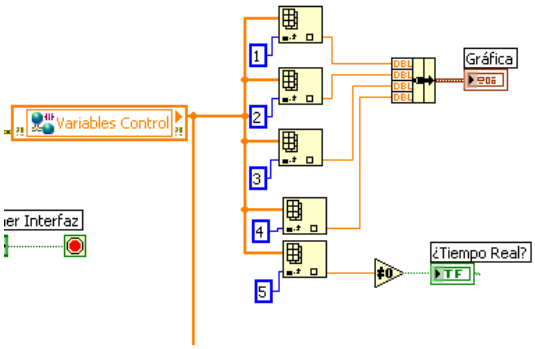


Figura 5.19. Lectura y despliegue de datos del FPGA.

Las lecturas de esta variable se almacenan en un archivo de texto. Las gráficas del comportamiento del controlador, que se presentan el capítulo siguiente, se reconstruyeron en Matlab a partir de estos archivos .TXT.

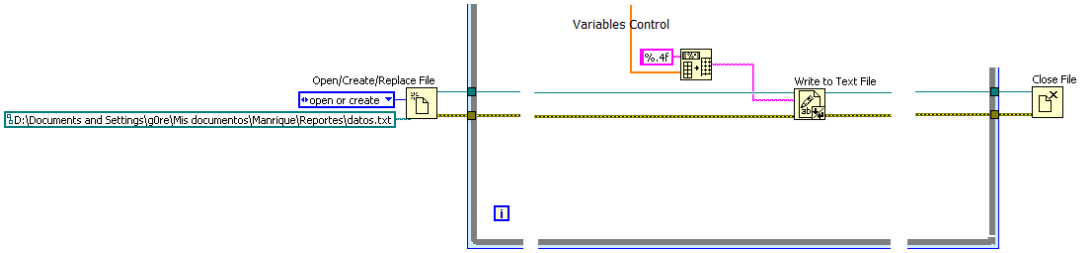


Figura 5.20. Comandos de escritura de datos a archivo.

La integridad del código se muestra en la Figura 5.21 a continuación:

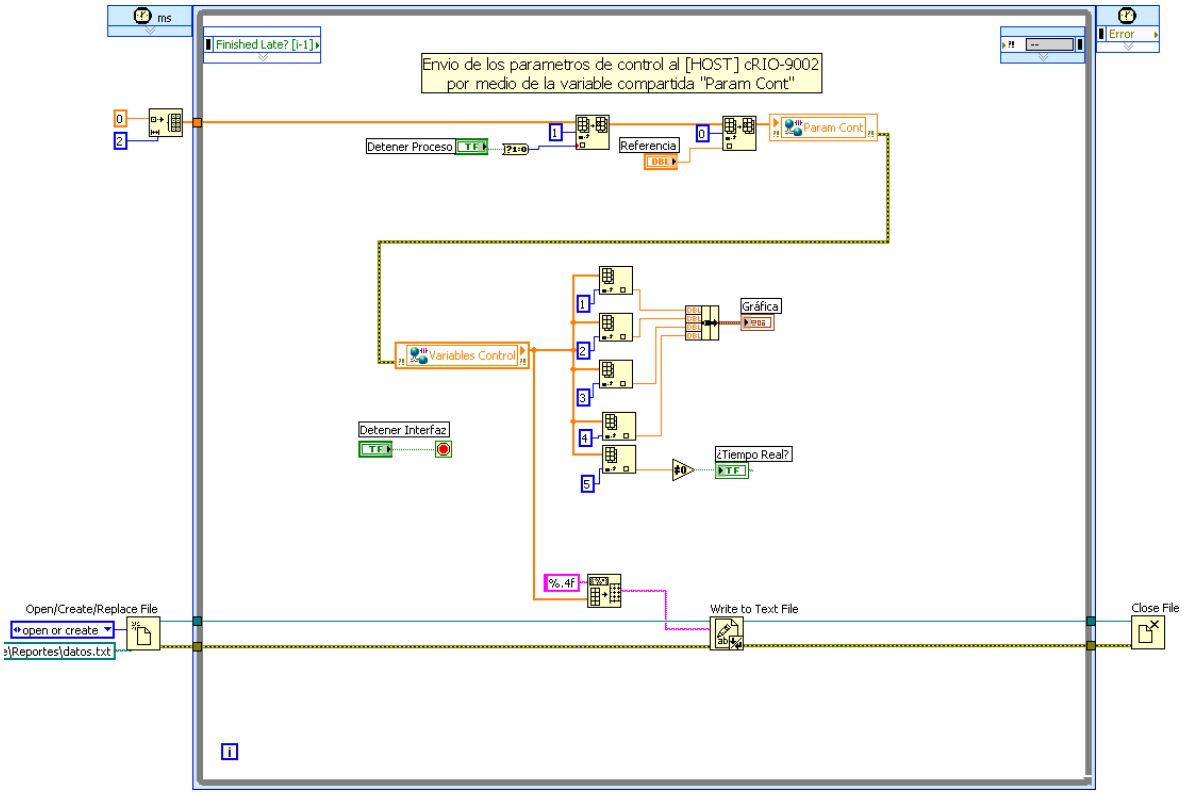


Figura 5.21. Diagrama de bloques de la interfaz de usuario.