UNIVERSIDADE DA CORUÑA

ESCUELA UNIVERSITARIA POLITÉCNICA

Grado en Ingeniería Electrónica Industrial y Automática

TRABAJO DE FIN DE GRADO

TFG Nº: **770G01A132**

TÍTULO: **DESARROLLO DE UN SISTEMA BASADO EN TÉCNICAS DE APRENDIZAJE COMPUTACIONAL PARA LA DETECCIÓN DE ANOMALÍAS EN UNA PLANTA DE NIVEL**

AUTOR: **CHAS GESTAL, BRAIS**

TUTOR: **FONTENLA ROMERO, OSCAR**
CALVO ROLLE, JOSÉ LUÍS

FECHA: **DICIEMBRE DE 2017**

Fdo.: EL AUTOR

Fdo.: EL TUTOR

TÍTULO: **DESARROLLO DE UN SISTEMA BASADO EN TÉCNICAS DE APRENDIZAJE COMPUTACIONAL PARA LA DETECCIÓN DE ANOMALÍAS EN UNA PLANTA DE NIVEL**

ÍNDICE GENERAL

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: DICIEMBRE DE 2017

AUTOR: EL ALUMNO

Fdo.: CHAS GESTAL, BRAIS

Índice de figuras

Índice de tablas

Listado de códigos de programación

TÍTULO: **DESARROLLO DE UN SISTEMA BASADO EN TÉCNICAS DE APRENDIZAJE COMPUTACIONAL PARA LA DETECCIÓN DE ANOMALÍAS EN UNA PLANTA DE NIVEL**

MEMORIA

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: DICIEMBRE DE 2017

AUTOR: EL ALUMNO

Fdo.: CHAS GESTAL, BRAIS

Índice del documento MEMORIA

1 Objeto

El presente proyecto tiene por objeto el desarrollo en Matlab de un sistema para la comunicación y el control automático de una planta de nivel de líquidos y la posterior detección de posibles anomalías, empleando para esto último un algoritmo del ámbito del aprendizaje computacional.

2 Alcance

1. Desarrollo de un regulador de tipo PID en Matlab para el control de la planta.

2. Desarrollo de un sistema de comunicación entre la planta y el usuario mediante la utilización de una tarjeta de adquisición de datos.

3. Análisis de los posibles algoritmos de detección de anomalías a utilizar y su aplicación en el ámbito de la planta.

4. Realización de pruebas en diversas condiciones de trabajo y análisis de los resultados obtenidos en la planta de laboratorio.

3 Antecedentes

En este capítulo se describe la planta que se utiliza para el presente proyecto y los conocimientos previos a la realización del presente trabajo. Se habla también del control automático y los reguladores PID ya que se realiza el control de esta planta de laboratorio mediante un regulador PID y también se trata el tema de la Inteligencia Artificial, que mediante una red neuronal artificial se ha utilizado para confeccionar el sistema de detección de anomalías.

3.1. Planta de laboratorio

La planta que vemos en la figura 3.1.0.1, situada en el *Laboratorio de Optimización y Control* de la *Escuela Universitaria Politécnica* de la *Universidade da Coruña*, ha sido utilizada antes del presente trabajo para múltiples prácticas de Ingeniería de Control y diferentes proyectos por lo que su funcionamiento está contrastado. Su control se realiza desde *Matlab* y a través, o bien de una tarjeta de adquisición de datos o de un *Arduino*. En este caso se utilizará una tarjeta de la marca *National Instruments* y modelo *USB-6008*. Como su nombre indica, esta se comunica con el ordenador mediante USB.

La planta cuenta con dos depósitos de agua que llamaremos D1 y D2. D2 está inicialmente vacío y D1 lleno de agua. Cuando desde el ordenador se envía una señal mediante la tarjeta a la planta, esta le llega entre 0 y 10V a un variador (V) *Altivar 31* de la marca *Schneider*. Al recibir esta señal, el variador, de 0.5 caballos de potencia, comienza a alimentar una bomba de agua entre 0 y 50 Hz. La bomba en ese momento recoge agua del depósito D1 y la bombea hasta D2. Este segundo depósito está comunicado a su vez con el primero mediante dos tuberías: la primera tiene una llave de paso manual normalmente cerrada y la segunda, de menos diámetro, comunica los dos depósitos directamente por lo que siempre que haya agua en D2, fluirá hacia D1.

En D2 hay 3 sensores. Dos de ellos, de tipo bolla, son de seguridad y se emplean para que el agua no sobrepase los límites del depósito. No se puede leer su estado desde fuera. El tercero, un sensor de ultrasonidos (*S18UUA* de la marca *Banner*), se utiliza para medir la cantidad de agua que hay en el depósito. Este envía una señal de 0 a 10V a la tarjeta y de ahí al ordenador, donde se interpretará la lectura y se adaptará a las necesidades del programa.

En D1 también hay un sensor de tipo bolla que para la planta en caso de que se acabe el agua.

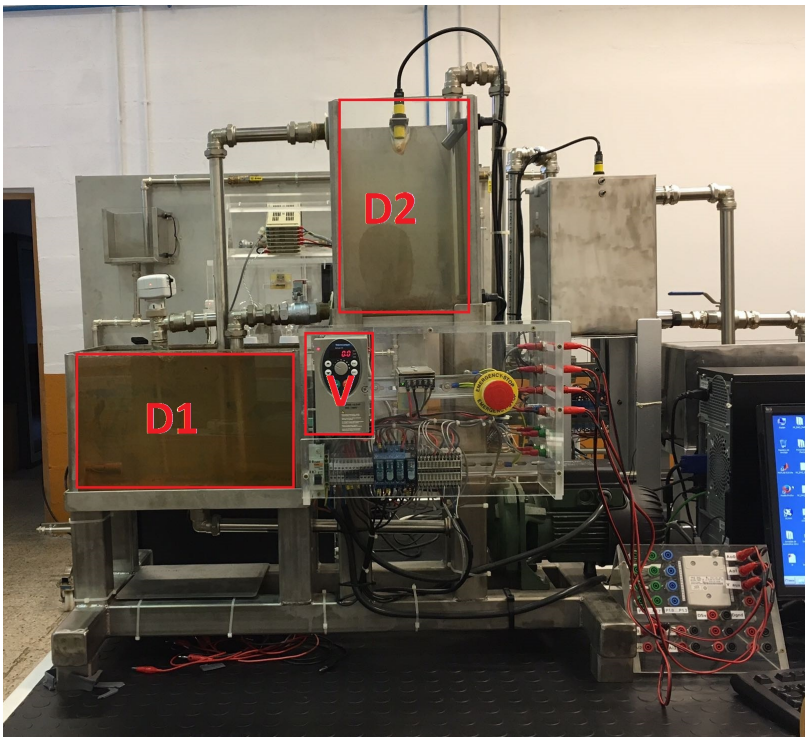

Figura 3.1.0.1 – Planta de agua en el laboratorio

3.2. Control

El Control es indispensable en cualquier planta industrial y se ocupa de mantener a un sistema en las condiciones deseadas en base al estado de sus variables. Esta rama de la ingeniería nace a mediados del siglo XX con aportaciones de autores como Rudolf Kalman, Norbert Wiener y David G. Luenberger[i] y avanza durante décadas hasta la actualidad, donde se emplean ordenadores, autómatas o microcontroladores para realizar el control. De esta forma se han mejorado las condiciones de muchos operarios en sus puestos de trabajo así como se han agilizado sus procesos.

3.2.1. Control discreto

Hoy en día la mayoría del control se realiza de forma digital debido al bajo coste de los sistemas electrónicos y a las ventajas de tratar señales digitales. Además, los programas son muy flexibles y tienen una gran capacidad para la toma de decisiones. En el presente trabajo, debido a la parte de detección de anomalías, sería impensable un control analógico, por lo que se ha diseñado un regulador PID digital.

3.2.1.1. Regulador PID

El primer gran avance en el campo del control discreto se produce en el siglo XX de la mano de Nicholas Minorsky, que publica *Estabilidad direccional de cuerpos dirigidos automáticamente* donde analiza las características de los reguladores de tipo PID. En dicha publicación describe cómo aplicar las tres partes del regulador para el gobierno de la dirección del buque

estadounidense *USS New Mexico*. Por ello se le considera inventor de los reguladores tipo PID.

Estos reguladores son los más utilizados en el presente en la industria. Consta de tres partes: la proporcional, la derivativa y la integral. Cada una de estas partes tiene una función dentro del controlador.

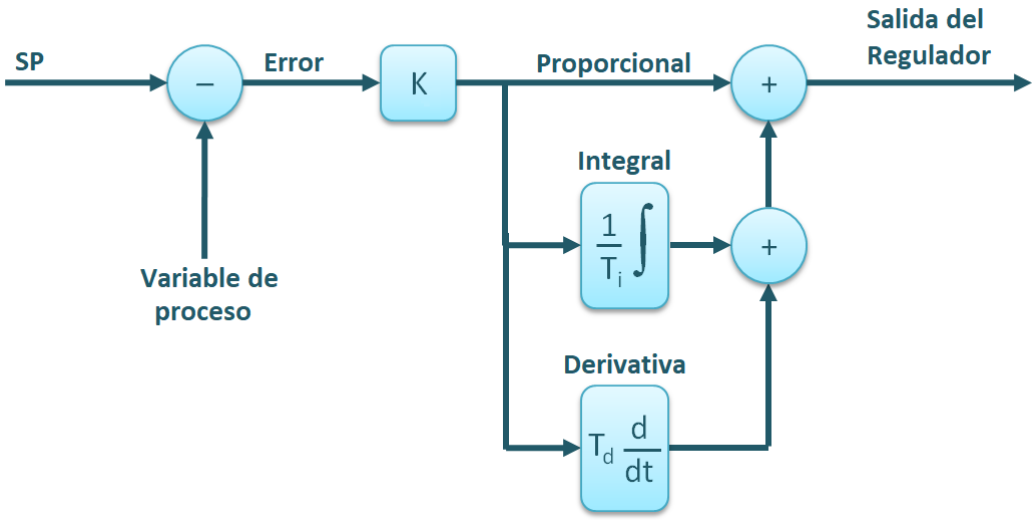

Figura 3.2.1.1 – Diagrama de bloques de un regulador PID

- Parte proporcional[2]

  Es el producto del error en el instante anterior por una constante llamada constante proporcional («K»).

- Parte integral

  Es el producto del error en el instante anterior por una constante llamada constante proporcional («K») y además por la inversa de la constante integral («T_i») multiplicada por la integral del error.

- Parte derivativa

  Es el producto del error en el instante anterior por una constante llamada constante proporcional («K») y además por la constante derivativa («T_d») multiplicada por la derivada del error.

3.3. Inteligencia artificial

La inteligencia artificial tiene como objetivo el desarrollo de métodos que doten de un comportamiento inteligente a las máquinas. En realidad, el concepto de inteligencia artificial no es tan reciente como parece. Desde los tiempos de Alan Turing (al que se considera el padre de la misma) y la construcción de su dispositivo Bombe, que permitió descifrar los códigos de la máquina Enigma alemana, han pasado más de setenta años. Hoy en día los sistemas de

inteligencia artificial son parte de la vida cotidiana (reconocimiento de voz, control de sistemas, drones,...). La parte de la inteligencia artificial de la que se ocupa el presente trabajo son las redes neuronales artificiales.

3.3.1. Redes neuronales artificiales

Una red neuronal es un modelo computacional formado por una gran cantidad de unidades neuronales simples. El comportamiento es parecido al observado en los axones de las neuronas en los cerebros biológicos. Cada unidad neuronal está conectada con muchas otras y los enlaces entre ellas pueden incrementar o inhibir el estado de activación de las neuronas adyacentes. Cada neurona opera individualmente empleando funciones de suma. Existe un umbral en cada conexión y en la propia unidad, de forma que la señal debe superar un límite antes de propagarse a otra neurona. Estos sistemas aprenden y se forman a sí mismos, en lugar de ser programados de forma explícita, y destacan en áreas donde la detección de patrones o características es difícil de expresar con la programación tradicional.

Las neuronas se agrupan por capas como se puede ver en la figura 3.3.1.1.

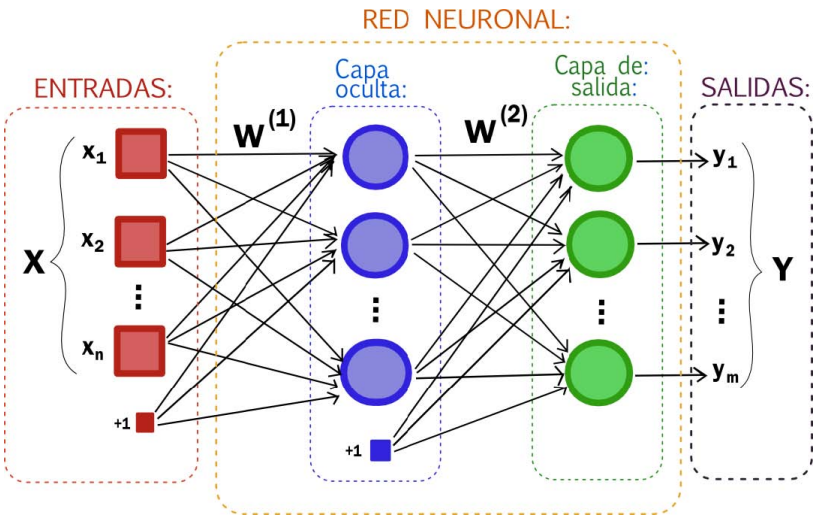

Figura 3.3.1.1 – Esquema de una red neuronal

Las redes neuronales se entrenan con los datos de entrenamiento[5]. Estos datos han sido etiquetados antes por un experto humano como funcionamientos normales. El objetivo del entrenamiento de la red es optimizar los pesos $W^{(1)}$ y $W^{(2)}$ que se ven en la figura 3.3.1.1 para que después esta detecte un dato de funcionamiento anómalo cuando se le introduzcan datos de test. Para el aprendizaje, estos sistemas emplean habitualmente algoritmos iterativos con el objetivo de reducir cierta función objetivo basado en un error en la salida de la red. Este aprendizaje puede ser supervisado o no supervisado. En este caso, un experto humano etiqueta los datos como anómalos o normales por lo que el aprendizaje es supervisado.

Una red neuronal puede estar alimentada de forma cíclica o recurrente. La que se elige en este caso, el autoencoder, tiene alimentación hacia delante.

4 Normas y referencias

4.1. Bibliografía

[1] BILLINGSLEY, JOHN; *Essentials of Control Techniques and Theory*, 1ª ed. Boca Raton, CRC Press, (2009).

[2] AL-HADITHI, BASIL M.; *Sistemas Discretos de Control*, 1ª ed. Madrid, Vision Libros, (2011).

[3] OGATA, KATSUHIKO; *Sistemas de control en tiempo discreto*, 2ª Ed. Minnesota, Pearson educación, (1996).

[4] SAKURADA, M. Y YAIRI, T.; *Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction, Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, (2014).

[5] PIMENTEL, M. AF Y CLIFTON, D. A Y CLIFTON, L. Y TARASSENKO, L.; *A review of novelty detection, Signal Processing*, (2014), vol. 99, págs. 215–249.

4.2. Software utilizado

4.2.1. Software de cálculo

Matlab R2014a 32-bit.

4.2.2. Software de edición

Para redactar el presente trabajo se ha utilizado el lenguaje de edición Latex. Para ello ha sido necesario contar con los siguientes programas:

- TexnicCenter: Editor de latex.
- Miktex: Compilador de Latex.
- Adobe Reader: Visualizador de PDF.

4.3. Otras referencias

[i] *Orígenes del PID*, Organización, [Consulta 16 octubre 2017]. Disponible en: https://poloestable.wordpress.com/2009/11/02/origenes-del-pid/

5 Definiciones y abreviaturas

AUC: Área debajo de la curva.

Consigna: valor final al que se desea llegar en un sistema de control.

DAQ: Data AcQuisition.

PDF: Portable Document Format.

PID: Proporcional-Integral-Derivativo.

ROC: Receiver Operating Characteristic.

SP: Set Point. Sinónimo de consigna.

USB: Universal Serial Bus.

6 Requisitos de diseño

Se realizará un regulador PID con tiempo de muestreo constante de 1 segundo. Cada valor de consigna utilizado dispondrá de unos parámetros diferentes, calculados para cada caso.

Además, se integrará un algoritmo de detección de anomalías con una red neuronal que detectará el comportamiento anómalo del sistema en base a los datos de salida del mismo y a un umbral, establecido por el desarrollador, que dicta el límite entre un dato anómalo y un dato de funcionamiento normal.

7 Análisis de las soluciones

En este capítulo se van a analizar las soluciones empleadas para cada parte del sistema de detección de anomalías. Desde el método de comunicación entre la planta y el software hasta el tipo de red neuronal utilizada pasando por aspectos de diseño del regulador PID.

7.1. Comunicación Matlab-Planta

Para la comunicación entre la planta y la DAQ se emplean cuatro scripts ya desarrollados anteriormente:

- DAQ_Start: inicia la comunicación con la tarjeta.
- DAQ_Stop: interrumpe la comunicación con la tarjeta.
- DAQ_Read: devuelve un valor entre 0 y 100 correspondiente al tanto por ciento de llenado del tanque D2.
- DAQ_Write: envía un valor entre 0 y 100 correspondiente al tanto por ciento de potencia que se quiere emplear en la bomba.

7.2. Constante derivativa

Una vez calculada la constante derivativa T_d se ha decidido multiplicarla por 2 para aumentar su efecto. Esta decision se ha tomado en base a la experiencia en trabajos anteriores en donde se mejoran las especificaciones temporales del sistema notablemente. En la figura 7.2.0.1 se puede ver la respuesta del sistema con un PID con las constantes calculadas según las fórmulas de *Ziegler Nichols* modificadas para poca sobreoscilación.

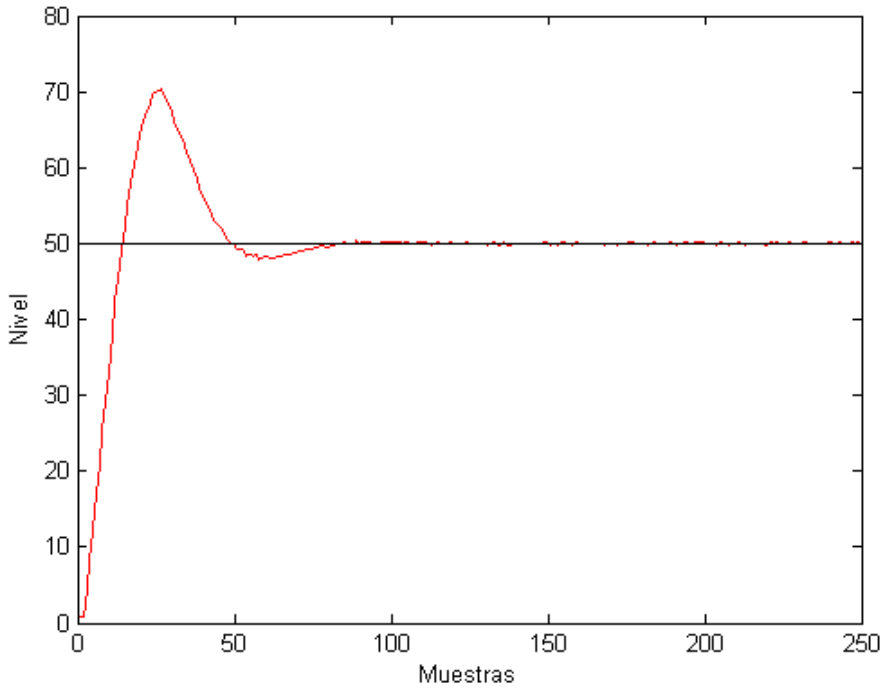

Figura 7.2.0.1 – PID con T_d calculada

Si ahora se dobla esa T_d con el resto de parámetros iguales en el mismo PID la respuesta es la de la figura 7.2.0.2.

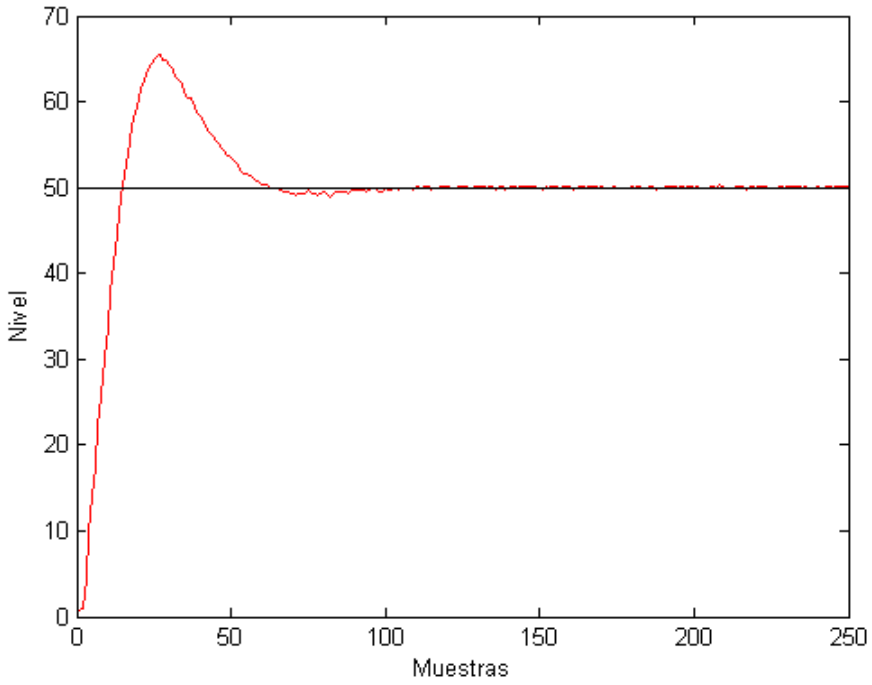

Figura 7.2.0.2 – PID con T_d doble

Se puede ver como mejoran la sobreoscilación y el tiempo de establecimiento, por eso se

toma la decisión de modificar este parámetro.

7.3. Selección del tipo de regulador

A la hora de escoger el tipo de regulador que se desea, se elige el de tipo PID en base a la experiencia que hay sobre esta planta obtenida en trabajos precedentes: es un sistema con poca inercia, un P o un PD generaría un error de posición elevado y el PI no mantendría bien todas las especificaciones en el tiempo.

7.4. Red de neuronas empleada

Entre todas las opciones de redes de neuronas artificiales se ha escogido un autoencoder[5] debido al problema que se quiere resolver: la detección de anomalías. Un autoencoder es una Red Neuronal simple, con solo tres capas: la de entrada, la de salida y la oculta. Está entrenada para reconstruir las entradas a las salidas. Evidentemente esta reconstrucción no es exacta ya que la capa oculta filtra la información que no le parece relevante, priorizando los datos que le parecen más útiles. Para esta reconstrucción se siguen dos procesos: codificación y decodificación. A medida que se desarrolla el proceso de entrenamiento se va minimizando el error de reconstrucción entre la entrada y salida, obteniendo así la nueva función aprendida.

Como en este tipo de redes se busca copiar la entrada en la salida, es fundamental que ambas tengan las mismas dimensiones.

Este tipo de red es muy utilizado para generación de características como la clasificación y detección de lesiones en imágenes médicas. En el caso del presente trabajo también se busca generar una característica, la anomalía, otro motivo por el que se utiliza este tipo de red.

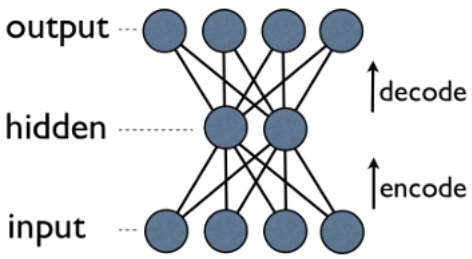

Figura 7.4.0.1 – Autoencoder

Como se puede comprobar con el presente trabajo, una red neuronal de este tipo aporta información muy precisa y comete poco error. Esto le da ventaja frente a un sistema clásico en el que las reglas para que un valor fuese declarado como anómalo serían iteraciones impuestas por el experto humano. Un ejemplo de esto sería imponer una condición en la que si determinada variable de entrada supera un valor fijado por el usuario se declararía anómala y en el caso contrario se declararía normal. Este tipo de sistemas son de fácil implementación

pero mucho más difíciles de ajustar ya que no aprenden solos. El error sería también mucho más grande.

8 Resultados finales

El presente trabajo tiene dos objetivos principales. El primero de ellos es controlar la planta de agua del laboratorio con un regulador y el segundo detectar posibles anomalías en su funcionamiento. Para una mejor comprensión del sistema se dispone del diagrama de bloques de la figura 8.0.0.1.

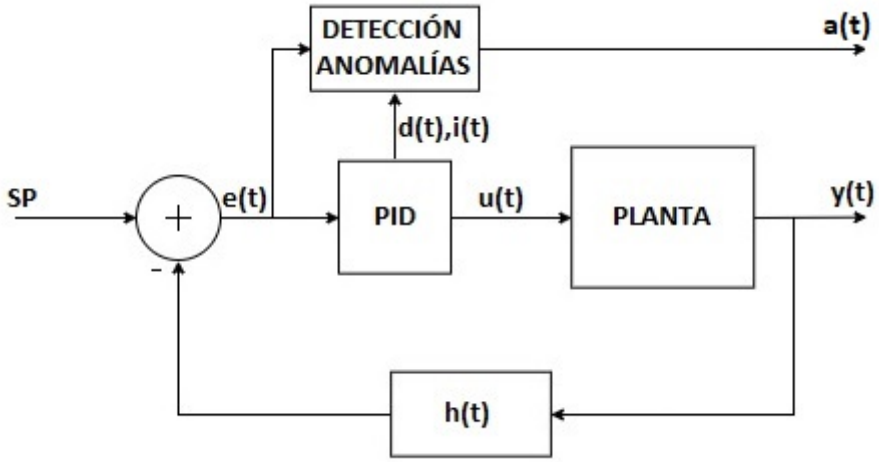

Figura 8.0.0.1 – Diagrama de bloques general

La entrada del sistema será el nivel en porcentaje al que se quiere elevar el agua en el depósito (SP). Este se compara con el valor actual y se genera una señal de error ($e(t)$). A continuación esa señal entra en el regulador, que con la señal de control ($u(t)$) lleva el sistema hasta la zona deseada. De la planta se extrae la lectura del nivel y se acondiciona ($H(t)$) para volver a comparar con el SP. Aparte de eso, se tiene el bloque de detección de anomalías, que tiene de entradas el error ($e(t)$) y los parámetros del regulador. A su salida informa del comportamiento normal o anómalo del sistema.

8.1. Regulador

8.1.1. Ecuaciones del regulador

En primer lugar, se confecciona el regulador PID en Matlab y se plantea su ecuación continua:

$$C(s) = K \cdot \left(1 + \frac{1}{T_i s} + \frac{T_d s}{1 + \frac{T_d}{N} s}\right) \quad (8.1.1.1)$$

Como se trabaja en discreto, hay que discretizar la ecuación. Para ello se emplea la discretización de Tustin (ver sección de cálculos), obteniendo como resultado las siguientes ecuaciones para el PID:

- El error: $e[k] = r[k] - y[k]$.
- La parte integral: $i[k] = C_1(e[k] + e[k-1]) + i[k-1]$.
- La parte derivativa: $d[k] = C_2(y[k] - y[k-1]) + C_3 d[k-1]$

La ecuación de la señal de control resulta de la forma $u[k] = K(e[k] + i[k] + d[k])$.

8.1.2. Obtención de parámetros del regulador

Una vez obtenidas estas ecuaciones se ha implementado el regulador en Matlab, en base a diferentes pruebas, se ha comprobado que para este sistema se mejora la respuesta eliminando la constante proporcional de la parte derivativa y de la integral. El código se puede ver en los anexos de código. Se ha elegido en principio una consigna (SP) del 50 % para ajustar el regulador y seleccionar las ecuaciones que finalmente se usarán.

Además de eso se necesita calcular las constantes mencionadas anteriormente para las ecuaciones. Para medir las características de la respuesta del sistema (el periodo de oscilación sostenida T_c y la ganancia proporcional crítica K_c) existen varios métodos:

- Método de oscilación sostenida de *Ziegler y Nichols*.
- Diagramas de Bode.
- Método *Relay-Feedback*.

El primer método es fundamental en el ajuste de reguladores PID pero supone aumentar la ganancia proporcional llevando al sistema a una zona límite de estabilidad (oscilación) y corriendo el riesgo de llevarlo a la inestabilidad. En cuanto a los diagramas de Bode, para este sistema no es sencillo practicar un análisis frecuencial, por lo que el método *Relay-Feedback* es en este caso el más adecuado.

Este método, desarrollado por Aström y Hägglud, consiste en hacer oscilar el sistema en torno a al punto de operación mediante el uso de un relé con histéresis como el de la siguiente figura:

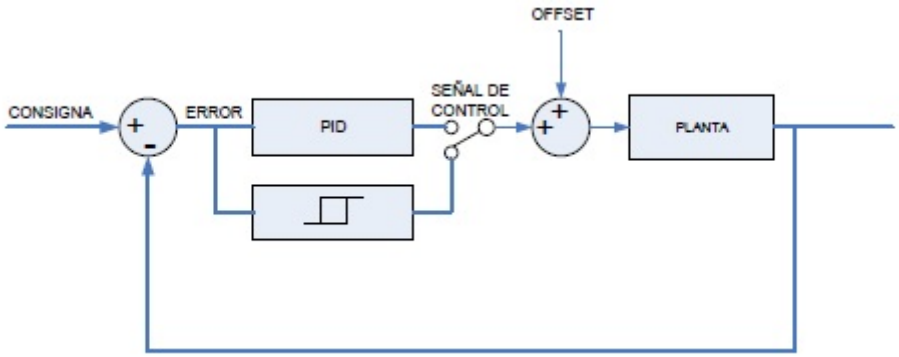

Figura 8.1.2.1 – Esquema del Relay-Feedback

El relé que se emplea para el experimento tiene una amplitud $d = 50$ y un ancho de ventana de histéresis $h = 5$. Estos valores se utilizan en base a la experiencia previa sobre esta planta. El periodo de oscilación de este relé es semejante al periodo de oscilación sostenida T_c que se quiere obtener.

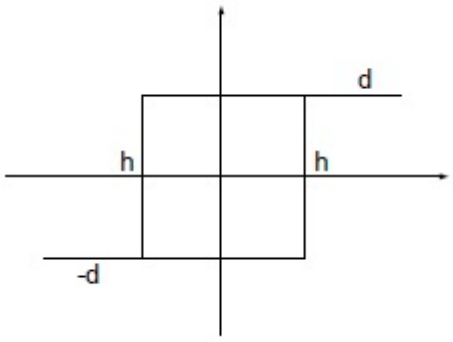

Figura 8.1.2.2 – Histéresis del Relay-Feedback

El código del script de Matlab se puede ver en el anexo de códigos de programación. En él, se establece un tiempo de funcionamiento de noventa segundos para conseguir varias oscilaciones completas. A continuación, se genera una matriz de dos filas (el tiempo y el nivel de llenado) y noventa columnas para cada uno de los valores. El periodo de muestreo es de 1 segundo en todos los casos. Para tener la seguridad de que el tiempo de muestreo es exacto se incorporan los comandos *tic* y *toc*, que permiten calcular el tiempo de la ejecución del programa para poder restarlo así al tiempo que pasa el programa pausado y conseguir un tiempo de muestreo exacto. Una vez hecho esto, el programa entra en un bucle *for* en el que se genera la oscilación. Una vez transcurridos los noventa segundos se sale del bucle, se para la planta y se guarda la matriz generada en un documento de Excel para extraer los datos de nivel y tiempo. Esto también podría hacerse guardando la propia matriz como se hará más adelante. Por último, se detiene la comunicación con la tarjeta.

Una vez guardados los datos se procede a su análisis para extraer las dos constantes que se buscan. En primer lugar se busca una oscilación estable por lo que se desecha la primera, en la que el sistema traía más inercia de lo normal. En cuanto se tiene dicha oscilación, se mira

el periodo en la fila de índices. En este caso $T_c = 15s$. A continuación, dentro de ese periodo se mira el valor máximo y mínimo que se ha alcanzado. Se resta el valor mínimo del máximo y se obtiene así el valor de la amplitud $a = 12{,}25$. A partir de esos datos se puede calcular la ganancia proporcional crítica K_c de la siguiente manera:

$$K_c = \frac{4 \cdot d}{\pi\sqrt{a^2 - h^2}} = \frac{4 \cdot 50}{\pi\sqrt{12{,}25^2 - 5^2}} = 5{,}69 \tag{8.1.2.1}$$

Una vez que se tiene T_c y K_c se procede a calcular el resto de constantes del PID. Como se ha dicho en el análisis de las soluciones, las fórmulas elegidas son las de *Ziegler Nichols* modificadas con poca sobreoscilación (ver apartado de cálculos):

$$K = 1{,}88 \tag{8.1.2.2}$$

$$T_i = 7{,}50 \tag{8.1.2.3}$$

$$T_d = 5{,}00 \tag{8.1.2.4}$$

Además de eso, por la experiencia adquirida en la planta en anteriores proyectos, se sabe que se mejora la respuesta del sistema multiplicando por dos la T_d. Por tanto, la constante T_d utilizada es:

$$T_d = 10{,}00 \tag{8.1.2.5}$$

Una vez obtenidas las constantes, se completa el código con el regulador PID (Apartado de códigos de programación) y se prueba su funcionamiento, obteniendo la respuesta subamortiguada que se muestra a continuación:

Una vez hecho esto para el punto de trabajo del 50 %, se repetirá lo mismo para obtener los parámetros del PID en torno a otros dos puntos de funcionamiento: 70 % y 20 %. Se utilizan tres puntos de trabajo diferentes para comprobar como el sistema de detección de anomalías se adapta sin dejar de funcionar. Se hace oscilar el sistema en torno a los nuevos puntos, se extraen de nuevo T_c y K_c y se aplican las mismas fórmulas (Ver en sección de cálculos). Los resultados son los siguientes:

- Consigna de nivel de un 70 %

  $K_c = 5{,}59$

  $T_c = 16{,}00s$

  $K = 1{,}84$

  $T_i = 8{,}00$

  $T_d = 5{,}33 \cdot 2 = 10{,}66$

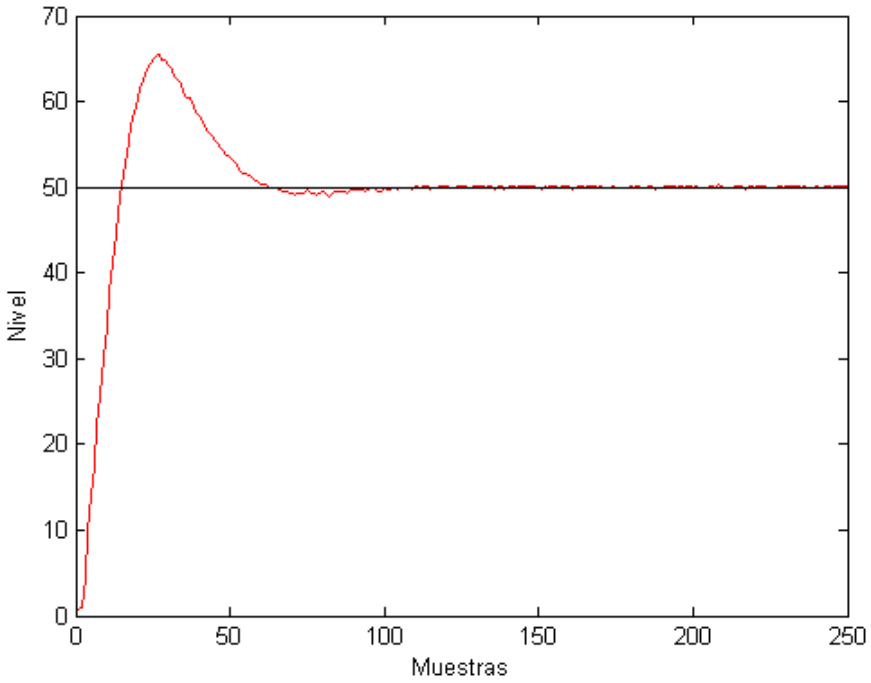

Figura 8.1.2.3 – Respuesta del regulador PID

- Consiga de nivel de un 20 %

  $K_c = 2{,}06$

  $T_c = 19{,}00s$

  $K = 0{,}68$

  $T_i = 9{,}50$

  $T_d = 6{,}33 \cdot 2 = 12{,}66$

8.2. Sistema de detección de anomalías

A continuación se va a describir el funcionamiento del sistema de detección de anomalías. Primero se extraerán los datos del funcionamiento de la planta, se guardarán y posteriormente se estudiará la utilidad de cada uno en la detección de anomalías.

8.2.1. Toma de datos

Se modifica el script del PID para hacer una toma de datos (ver el código de programación) y se recogen datos durante 3350 segundos funcionando con tres consignas diferentes sobre los reguladores desarrollados en el apartado anterior: primero 50, después 70 y por último 20. Se utilizan 3 consignas para luego poder estudiar si hace falta entrenar a un sistema de detección de anomalías para cada punto de funcionamiento o se puede utilizar uno solo. Una vez realizado el experimento sobre la planta se recoge una matriz de 3350 columnas y 7 filas en las que se guarda:

- Fila 1: índices de las muestras.
- Fila 2: nivel.
- Fila 3: error.
- Fila 4: parte integral del PID.
- Fila 5: parte diferencial del PID.
- Fila 6: consigna.
- Fila 7: señal de control.

La respuesta del sistema, representada junto a la consigna es la que se muestra en la figura 8.2.1.1:

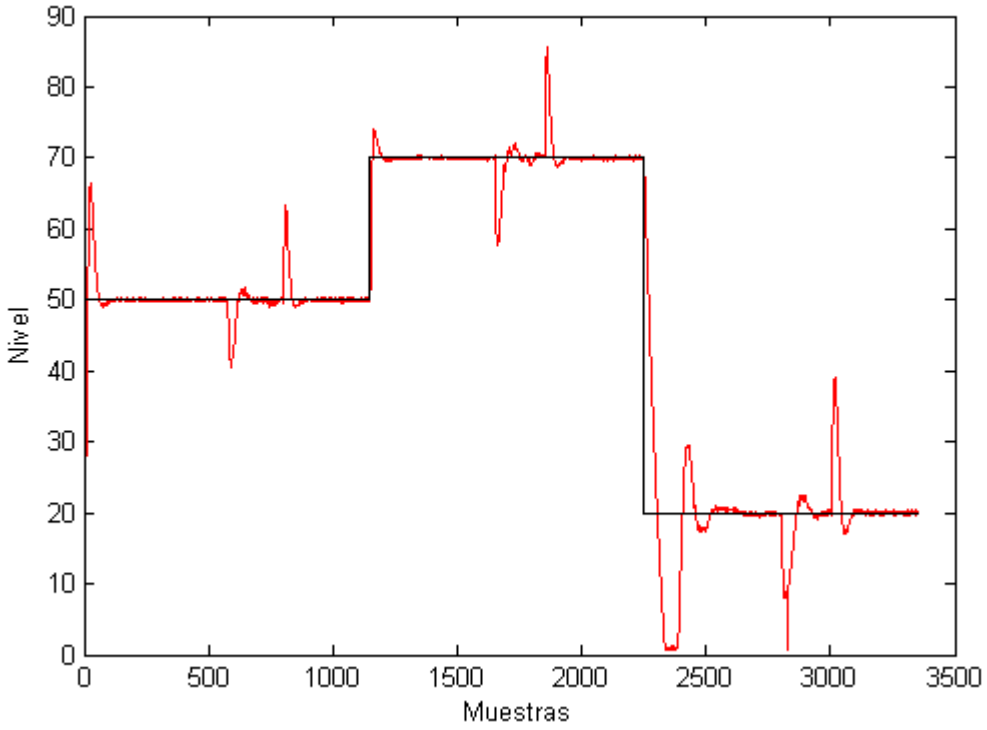

Figura 8.2.1.1 – Respuesta del sistema para las tres consignas empleadas en el experimento. En negro la consigna y en rojo la salida de la planta

8.2.2. Red neuronal

Se tiene el conjunto de datos mencionado anteriormente y además, otra matriz de 1 fila y tantas columnas como muestras en la que se etiquetan los datos como anómalos o negativos (0) y como funcionamiento normal o positivos (1) por un experto humano. Esto se hace para supervisar el entrenamiento del sistema de detección de anomalías.

Se definen cuatro posibles resultados en un experimento de predicción binario como el que se va a realizar en el presente trabajo. Si el resultado de una exploración es positivo y se había

etiquetado a ese valor como positivo, entonces se conoce como un Verdadero Positivo (VP); en cambio si el valor real es negativo se conoce como un Falso Positivo (FP). De la misma forma, se tiene un Verdadero Negativo (VN) cuando tanto la exploración como el valor etiquetado son negativos, y un Falso Negativo (FN) cuando el resultado de la predicción es negativo pero el valor etiquetado por el experto humano es positivo.

El sistema decide si un dato es positivo o negativo en función de su error. Si el error supera un umbral establecido es negativo y si no lo supera es positivo. Ese error se calcula matemáticamente en base a las 6 salidas del autoencoder como se puede ver en el código de programación.

En los experimentos realizados para el presente trabajo se calcula el porcentaje de acierto de cada combinación de variables de entrada para determinar qué combinación es la más útil para la detección de anomalías. Según se varía el umbral de discriminación[4] (valor a partir del cual decidimos que un caso es un positivo) el porcentaje de acierto varía. Este umbral se ajusta ayudándose de los percentiles. Estos percentiles dividen los datos de entrenamiento de la red en cien partes de igual tamaño entre el dato de entrenamiento con menor error y el que mayor error tiene. El programa calcula el porcentaje de acierto cogiendo cada percentil y después trabaja automáticamente con el mejor para los datos de test.

Cabe señalar aquí que para entrenar al sistema de detección de anomalías se emplean parte de los datos etiquetados como positivos (la parte destinada a entrenamiento por la instrucción *cvpartition*) por el experto humano y para el test se emplean los restantes datos positivos y todos los etiquetados negativos. Todos los datos que se utilizan para test son ciegos para el sistema, lo cual quiere decir que no se ha entrenado con esos mismos datos.

Se parte de un programa desarrollado anteriormente en otro proyecto. Dicho programa se utilizaba para clasificar tipos de plantas. Consta de un script principal donde se procesan los datos de entrada y salida de la red utilizando un autoencoder y de varios scripts con las funciones necesarias para el funcionamiento del algoritmo.

En este caso se modificará solo el script principal (Ver la explicación en la sección de códigos de programación, TFG) para adaptar a la planta el algoritmo. El nuevo script muestra 4 gráficas de las que se puede extraer diferente información:

- Gráfica 1:

  En el eje vertical presenta error y en el horizontal las distintas muestras. Muestra al principio, en negro, los datos del estado normal de funcionamiento que utiliza el sistema para entrenar. A continuación, se muestran los datos de test, en rojo el error de los datos etiquetados como anómalos y por último, en verde, el restante de los datos etiquetados como normales pero que el sistema no toma para el entrenamiento. Un ejemplo de esta gráfica es la figura 8.2.2.1.

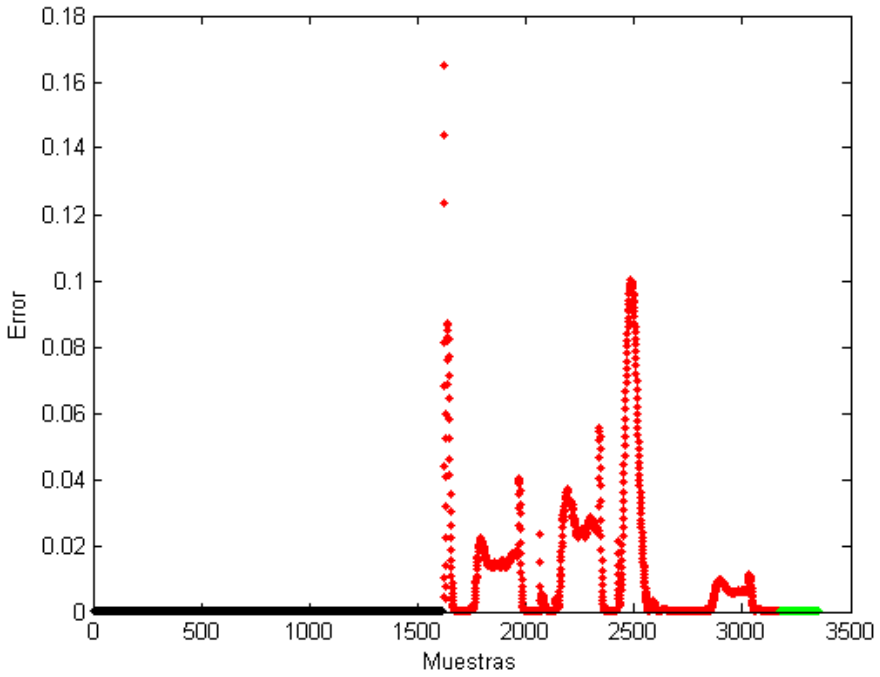

Figura 8.2.2.1 – Errores de entrenamiento (negro) y errores de test (rojo y verde)

- Gráfica 2:

  En el eje vertical presenta nivel y en el horizontal las distintas muestras. La forma de la gráfica es la del nivel del depósito de agua y sobre la misma se presentan en rojo los datos etiquetados de forma manual por un experto humano como anormales y en negro los datos etiquetados de la misma forma como funcionamiento normal. Se puede ver un ejemplo en la figura 8.2.2.2.

- Gráfica 3:

  En el eje vertical presenta error y en el horizontal las distintas muestras. Aquí se presentan los datos de test (los que no se utilizan para el entrenamiento del sistema). Hasta la línea vertical, el error de los datos de test etiquetados como anómalos y tras la línea vertical los datos de test etiquetados como normales.

  En cuanto a los colores, los datos a la izquierda de la línea son etiquetados como anormales y por lo tanto rojos. Si en ese parte aparece algún dato verde significa que el sistema lo etiqueta como normal siendo realmente anómalo. Si a la derecha de la línea aparece algún dato rojo significa que el sistema lo etiqueta como anómalo siendo realmente correcto. Ejemplo en la figura 8.2.2.3

- Gráfica 4:

  Esta es la gráfica que más información puede aportar. En el eje vertical presenta nivel y en el horizontal las distintas muestras. Sobre la curva de nivel se presentan en rojo los datos anómalos y en negro los datos correctos como en la gráfica 2. Además de eso

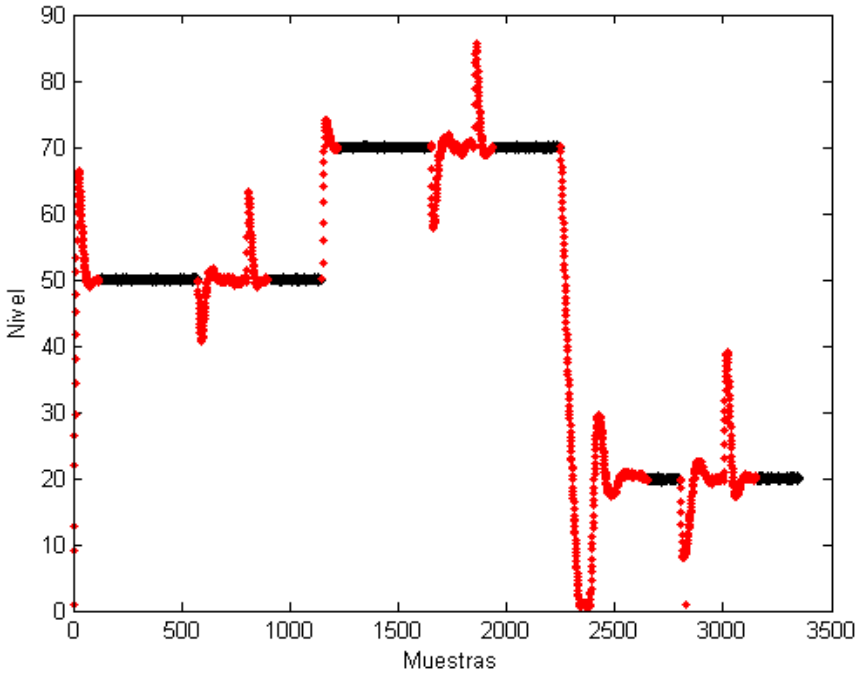

Figura 8.2.2.2 – Nivel real de la planta (en rojo los datos etiquetados como anómalos)

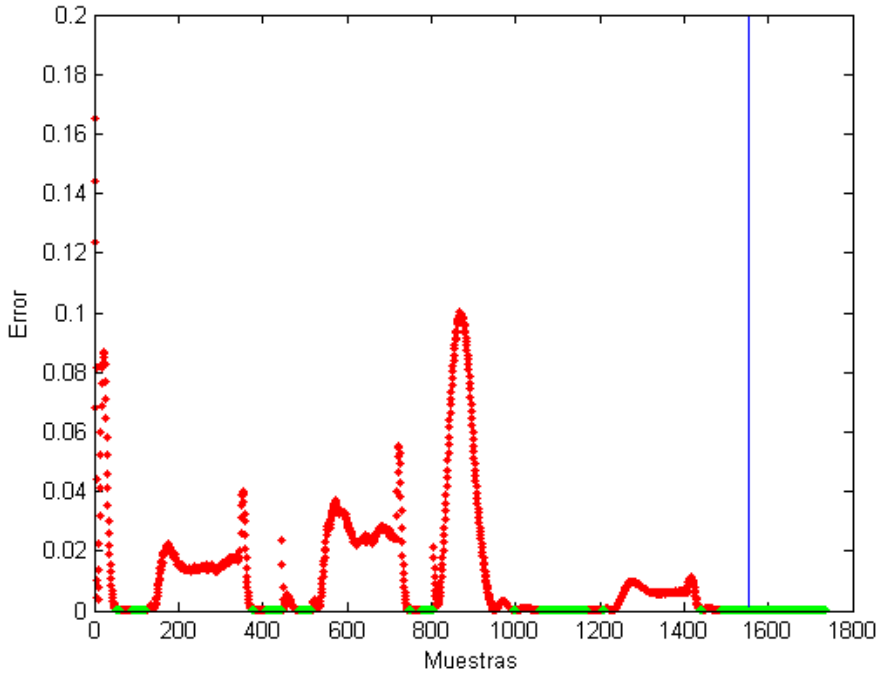

Figura 8.2.2.3 – Clasificación del sistema: en rojo los datos clasificados como anómalos y en verde como normales

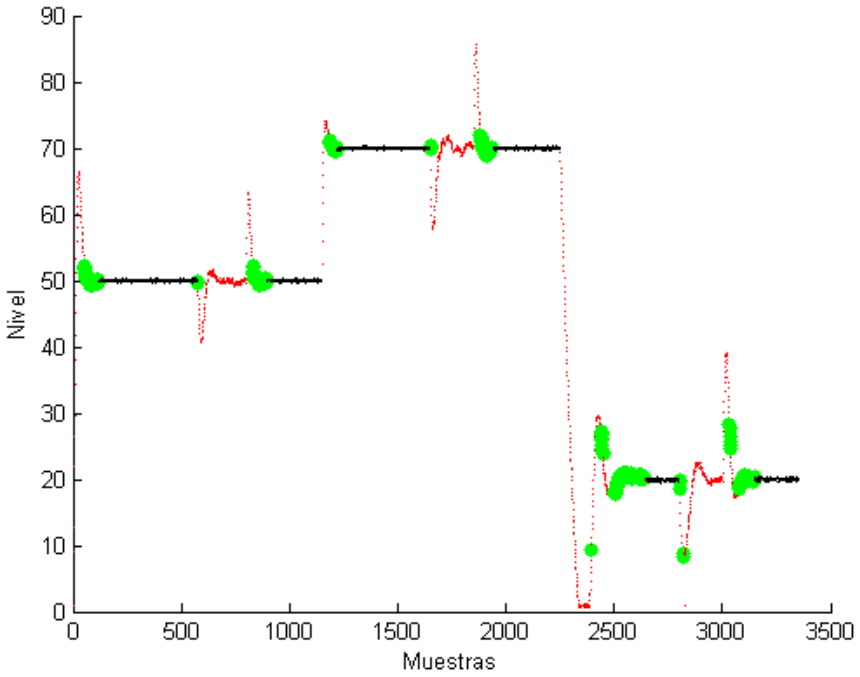

Figura 8.2.2.4 – En verde los datos anómalos que el sistema etiqueta como normales

se le añade la información de la gráfica 3, poniendo en verde los valores etiquetados como anómalos que el sistema tilda de normales, lo que serían errores del sistema de detección de anomalías. Ejemplo en la figura 8.2.2.4.

Cabe destacar que muchos de los errores cometidos por el sistema (falsos positivos) son en zonas que podrían considerarse como normales por lo que, aunque se catalogan como errores del sistemas, podrían no serlo dependiendo de la referencia humana empleada.

Una vez desarrollado el código, se va a tratar de determinar la mejor combinación de variables de entrada al algoritmo para la detección de anomalías en el funcionamiento del sistema. Al programa se le introducen las dos matrices de las que se ha hablado: la de las etiquetas y la de datos. Dentro de la de los datos no se cogerán todas las variables si no que se estudiarán las más apropiadas. Se pueden descartar los índices ya que son totalmente independientes del funcionamiento de la planta por lo que realmente hay 6 variables para estudiar.

8.2.3. Aspectos generales del sistema

Uno de los objetivos de este estudio es determinar si es necesario entrenar a un sistema de detección de anomalías diferente para cada consigna, por eso se utilizan tres valores tan diferentes como 50, 70 y 20. Se trata de demostrar aquí que el funcionamiento del sistema es correcto pese a ser entrenado para diferentes consignas al mismo tiempo. Para ello se le introducen los datos de entrenamiento, se ejecuta y se observa el resultado en la figura 8.2.3.1:

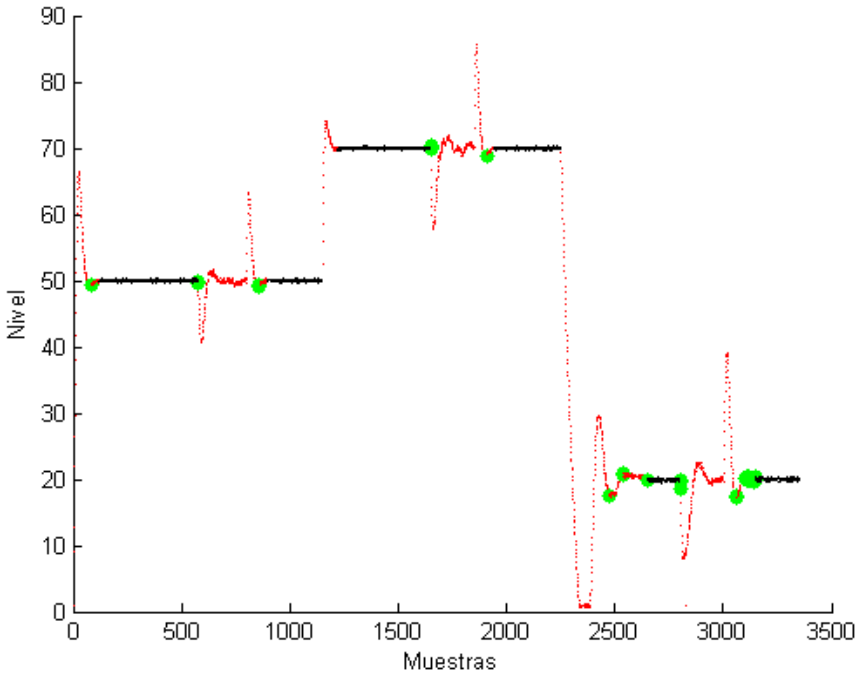

Figura 8.2.3.1 – Primer experimento: Aspectos generales del sistema.

Como se ve en la imagen, el sistema acierta o se equivoca por igual independientemente del punto de trabajo. Por eso se puede concluir que no es necesario entrenar un sistema para cada punto de funcionamiento.

Cabe destacar para este caso y para los restantes, que los datos en torno al punto de funcionamiento del 20 % son menos correctos que el resto. Esto se debe a que con tan poca agua el efecto del oleaje es importante, el agua se mueve bastante en régimen permanente y eso hace también difícil encontrar unos parámetros para el regulador PID tan ajustados como el resto. Además de eso, en esa parte se produce mucha sobreoscilación pero es debida a que el cambio de 70 a 20 es muy grande y se hace lentamente ya que es en vaciado por lo que se acumula mucha marte integral que luego se tiene que descontar con la sobreoscilación.

8.2.4. Análisis de las variables de entrada

En este apartado se estudia cuál es la mejor combinación de variables de entrada para el sistema de detección de anomalías. Para ello se hace un experimento incluyendo las 6 variables mencionadas anteriormente (nivel, error, parte integral del PID, parte diferencial del PID, señal de control y consigna) y otros 6 experimentos excluyendo en cada uno una de estas variables.

8.2.4.1. Experimento con todas las variables

En el primero de los experimentos se incluyen las seis variables para luego compararlo con los restantes en los que se suprimen las diferentes variables una a una. En este caso se muestra en la figura 8.2.4.1 el resultado del experimento en el que se tienen en verde los falsos positivos predecidos por el sistema sobre la curva de nivel. Como se mencionaba anteriormente, estos falsos positivos podrían no serlo dependiendo del criterio humano que se emplee para etiquetar los datos. Esta figura servirá para comparar con el resto de figuras de los demás experimentos.

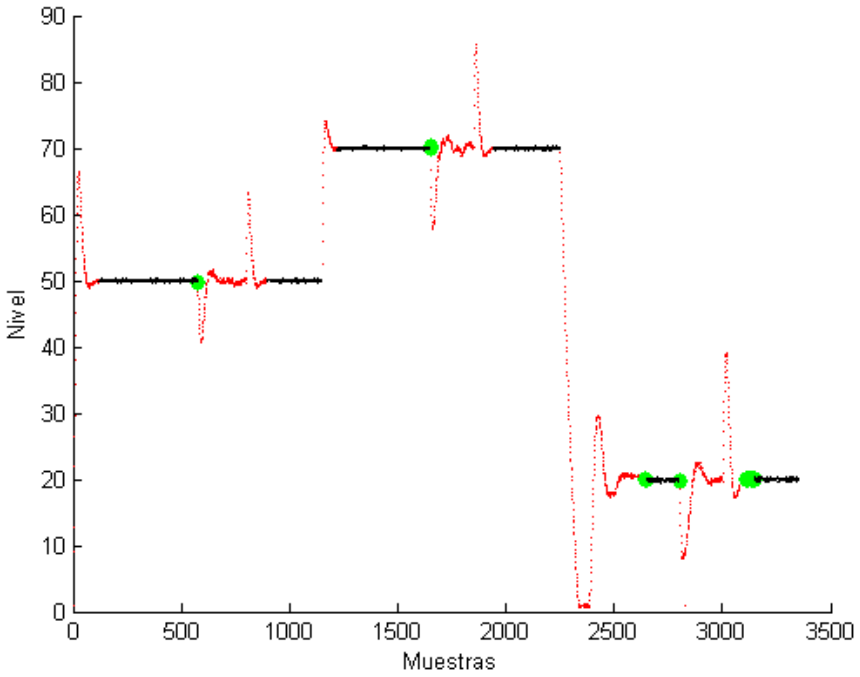

Figura 8.2.4.1 – Segundo experimento: todas las variables

La tabla 8.2.4.1 muestra el porcentaje de acierto por percentiles como se explicaba anteriormente. Aquí se ve que el el percentil 93 se obtiene un área del 97.78 %, lo que, como se verá más adelante, es un resultado muy bueno.

8.2.4.2. Experimento sin la consigna

Se estudiará aquí si conviene introducir la consigna como una de las entradas a la red neuronal. Para esto se elimina la señal de consigna de las entradas del sistema de detección de anomalías para ver el efecto que esto produce. Se ejecuta el programa resultando la gráfica 8.2.4.2.

Se puede ver en la figura 8.2.4.2 como empeora claramente la precisión del sistema sin la presencia de la consigna. Esto se debe a que con la consigna el sistema diferencia claramente entre un punto de trabajo u otro. Se demuestra la diferencia de funcionamiento de forma detallada con la tabla 8.2.4.2, en la que la máxima tasa de acierto es del 92.10 % en el percentil

Percentil	Acierto	Percentil	Acierto	Percentil	Acierto
50	76.78 %	67	85.38 %	84	94.93 %
51	77.06 %	68	85.94 %	85	95.18 %
52	78.18 %	69	86.77 %	86	95.11 %
53	78.74 %	70	88.14 %	87	95.61 %
54	79.02 %	71	88.42 %	88	95.48 %
55	79.58 %	72	88.70 %	89	95.35 %
56	80.41 %	73	88.95 %	90	96.55 %
57	81.25 %	74	89.50 %	91	96.92 %
58	81.53 %	75	89.75 %	92	97.28 %
59	81.81 %	76	90.59 %	93	97.78 %
60	82.09 %	77	90.87 %	94	97.68 %
61	82.09 %	78	91.71 %	95	97.42 %
62	82.93 %	79	92.51 %	96	97.26 %
63	82.93 %	80	93.07 %	97	97.13 %
64	83.49 %	81	93.04 %	98	96.94 %
65	84.01 %	82	93.60 %	99	97.11 %
66	84.54 %	83	93.84 %	100	92.31 %

Tabla 8.2.4.1 – Tabla acierto por percentiles en el experimento con todas las variables

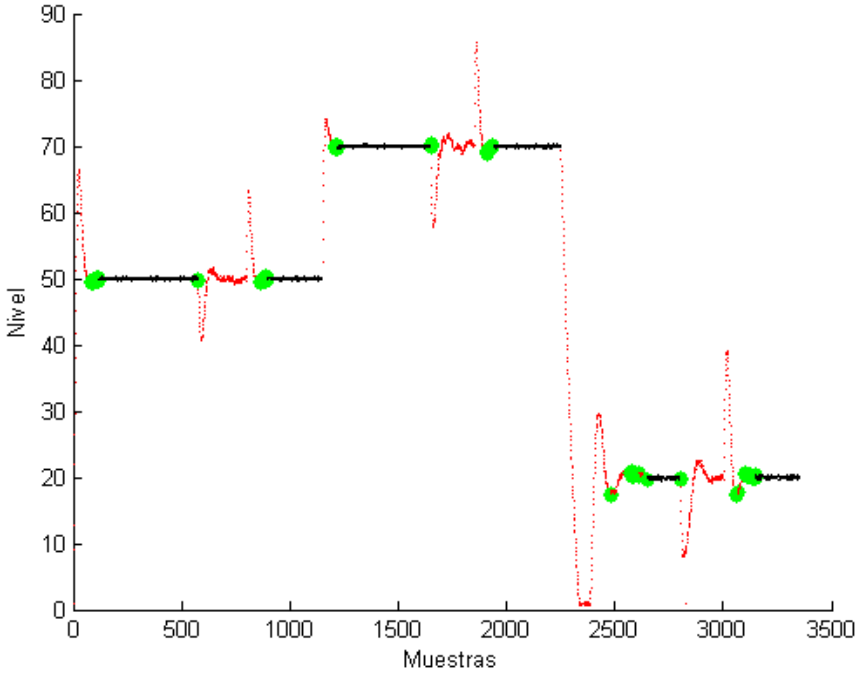

Figura 8.2.4.2 – Tercer experimento: gráfica sin la consigna

Percentil	Acierto	Percentil	Acierto	Percentil	Acierto
50	74.46 %	67	80.26 %	84	88.17 %
51	75.21 %	68	80.54 %	85	89.16 %
52	75.42 %	69	81.80 %	86	89.46 %
53	75.67 %	70	82.20 %	87	89.86 %
54	76.16 %	71	82.42 %	88	89.66 %
55	76.31 %	72	84.00 %	89	89.91 %
56	77.12 %	73	84.46 %	90	90.15 %
57	77.33 %	74	85.41 %	91	90.55 %
58	77.30 %	75	85.25 %	92	91.01 %
59	77.45 %	76	85.62 %	93	91.22 %
60	77.98 %	77	86.86 %	94	91.13 %
61	78.13 %	78	86.88 %	95	91.67 %
62	78.10 %	79	86.72 %	96	92.07 %
63	78.03 %	80	87.33 %	97	91.88 %
64	78.22 %	81	88.53 %	98	92.05 %
65	79.86 %	82	88.37 %	99	92.10 %
66	80.11 %	83	88.43 %	100	90.83 %

Tabla 8.2.4.2 – Tabla acierto por percentiles en el experimento sin la consigna

99, lo cual es mucho peor dato que el obtenido anteriormente en la tabla 8.2.4.1. Se recuerda que gran parte de los datos que detecta el sistema como correctos y que el experto humano había etiquetado como anómalos se producen en el momento en el que el sistema está casi estabilizado después de un cambio de punto de funcionamiento o tras una anomalía. Esta diferencia de interpretación entre el sistema de detección de anomalías y el experto se debe a una diferencia de criterios pero de todas maneras, esta diferencia está más marcada en este experimento que en el anterior.

Con la tabla 8.2.4.2 se puede concluir definitivamente que es positivo para la detección de anomalías en este sistema incluir la consigna como variable de entrada. Se ha implementado el programa de tal forma que se configure automáticamente para usar el percentil que de mejor resultado.

8.2.4.3. Experimento sin la señal de control

Para estudiar la importancia de utilizar la señal de control como entrada se siguen los mismos pasos. Se ejecuta el programa con todas las variables menos la señal de control y se compara con el experimento completo. El resultado es el de la figura 8.2.4.3.

Cabe destacar que podría ser intuitivo no incluir esta señal ya que sus tres componentes (error, parte diferencial y parte derivativa) por separado ya están representadas pero a la vista de las gráficas parece que el error del sistema es menor si le incluye. Se demuestra con la tabla 8.2.4.3.

En este caso también queda demostrada la ventaja de incluir la señal de control a pesar de estar incluidas ya sus tres componentes. Sin introducirla el acierto máximo se produce en el percentil 95 y es del 92.46 % mientras que anteriormente se había obtenido un acierto del 97.78 % en la tabla 8.2.4.1.

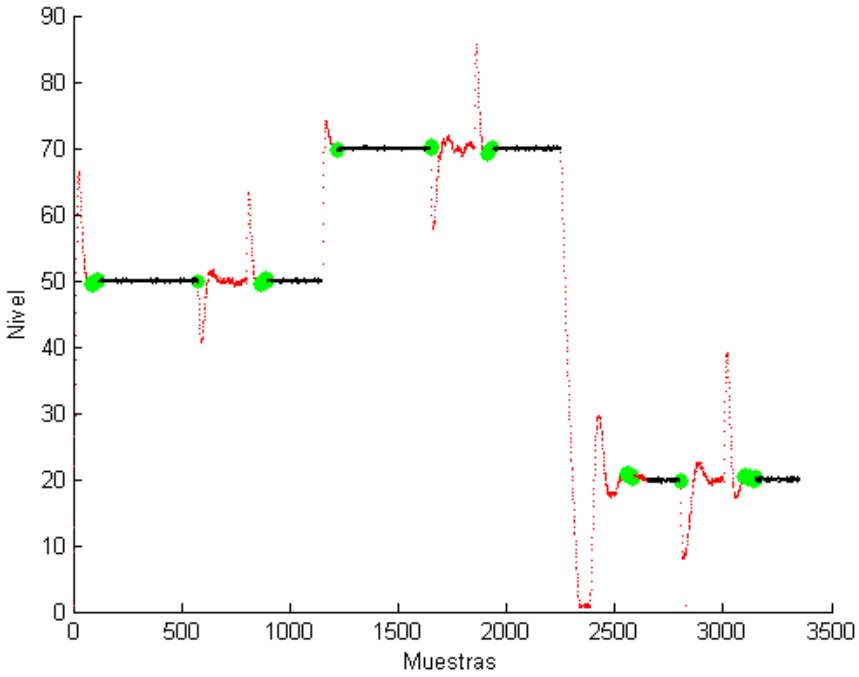

Figura 8.2.4.3 – Cuarto experimento: gráfica sin la señal de control

Percentil	Acierto	Percentil	Acierto	Percentil	Acierto
50	74.60 %	67	83.74 %	84	90.14 %
51	75.16 %	68	83.95 %	85	90.54 %
52	75.41 %	69	85.32 %	86	91.43 %
53	75.62 %	70	85.81 %	87	91.40 %
54	76.46 %	71	86.62 %	88	91.26 %
55	76.46 %	72	86.58 %	89	91.25 %
56	77.55 %	73	87.32 %	90	91.49 %
57	78.07 %	74	87.20 %	91	91.38 %
58	78.60 %	75	87.72 %	92	91.53 %
59	79.37 %	76	87.69 %	93	91.90 %
60	79.62 %	77	87.94 %	94	92.38 %
61	80.36 %	78	88.18 %	95	92.46 %
62	80.86 %	79	88.02 %	96	92.32 %
63	81.66 %	80	88.54 %	97	92.23 %
64	82.22 %	81	89.00 %	98	92.33 %
65	82.72 %	82	89.46 %	99	91.76 %
66	82.96 %	83	89.46 %	100	89.64 %

Tabla 8.2.4.3 – Tabla acierto por percentiles en el experimento de la señal de control

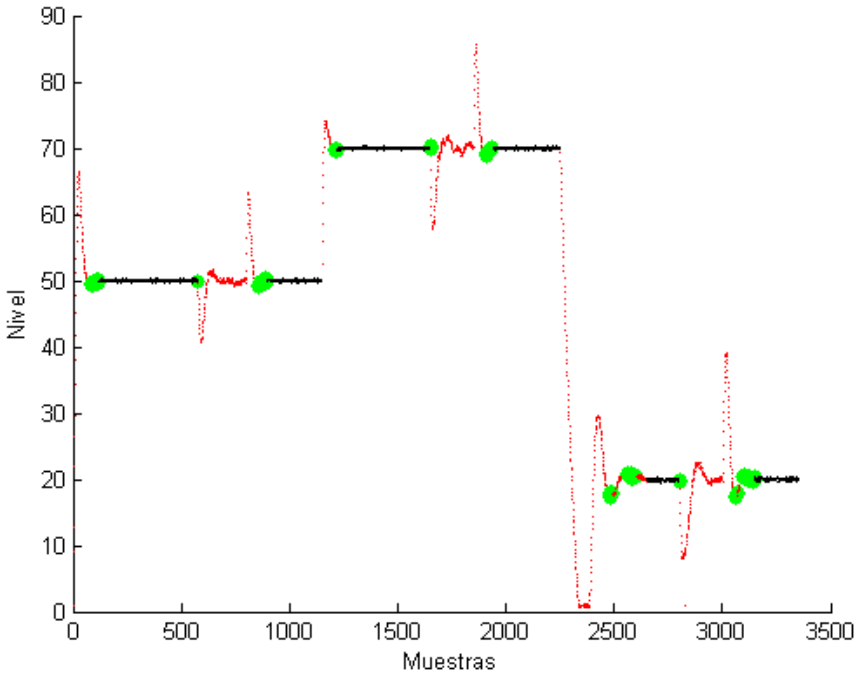

Figura 8.2.4.4 – Quinto experimento: gráfica sin el nivel

8.2.4.4. Experimento sin el nivel

Se introducen las variables utilizadas anteriormente eliminando el nivel. Se ejecuta el programa. La tabla resultante con la tasa de acierto por percentil es la tabla 8.2.4.4.

En la tabla 8.2.4.4 se demuestra que el nivel es una de las variables más importantes puesto que solo se obtiene el 91.18 % de acierto en el mejor percentil, el 96. La diferencia es muy grande. Por otro lado se puede decir que el sistema sin la variable nivel es menos sensible a los cambios de punto de funcionamiento. Esto es una variable a favor de excluirlo de la entrada pero no suficiente. En conclusión, el nivel debe estar presente en este sistema de detección de anomalías.

8.2.4.5. Experimento sin el error

Teniendo el nivel y la consigna al mismo tiempo se puede pensar que tener el error no aporta nada ya que este es la diferencia entre uno y otro pero en este caso se trata del error multiplicado por la constante proporcional (K) del PID. De esta forma se le da más peso al error, importante para detectar errores de posición en las anomalías. La gráfica sin el error es la de la figura 8.2.4.5.

La tabla resultante con la tasa de acierto por percentil es la tabla 8.2.4.5.

Sin el error el máximo acierto que se consigue es en el percentil 94 con tan solo un 91.38 % de aciertos mientras que con él se consigue un 97.78 % en la tabla 8.2.4.1. Queda demostrado así que pese a tener el error representado con el nivel y la consigna, el incluirlo aporta más exactitud aparte de hacerlo más sensible a los errores de posición.

Percentil	Acierto	Percentil	Acierto	Percentil	Acierto
50	74.39 %	67	83.18 %	84	88.09 %
51	74.57 %	68	83.30 %	85	88.18 %
52	75.07 %	69	83.45 %	86	87.89 %
53	75.59 %	70	83.57 %	87	88.35 %
54	75.78 %	71	83.97 %	88	88.57 %
55	76.17 %	72	84.15 %	89	89.09 %
56	76.42 %	73	83.96 %	90	89.71 %
57	77.79 %	74	84.11 %	91	89.94 %
58	78.31 %	75	84.63 %	92	89.88 %
59	78.18 %	76	85.19 %	93	90.37 %
60	79.30 %	77	85.28 %	94	90.92 %
61	79.24 %	78	85.64 %	95	90.73 %
62	79.67 %	79	86.04 %	96	91.18 %
63	80.56 %	80	86.57 %	97	91.04 %
64	81.24 %	81	86.72 %	98	90.78 %
65	81.48 %	82	87.89 %	99	90.10 %
66	82.19 %	83	87.88 %	100	89.10 %

Tabla 8.2.4.4 – Tabla acierto por percentiles en el experimento sin el nivel

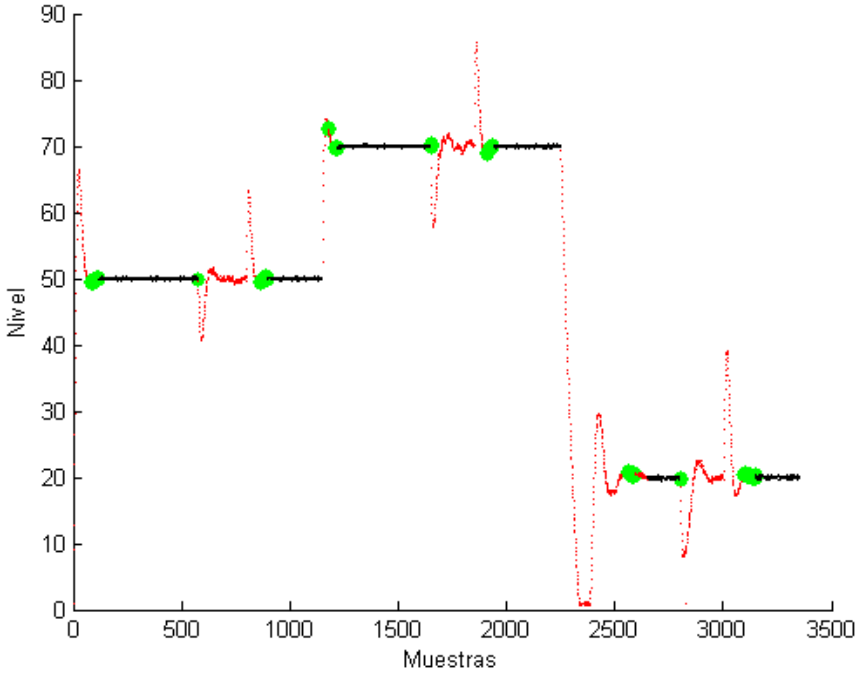

Figura 8.2.4.5 – Sexto experimento: gráfica sin el error

Percentil	Acierto	Percentil	Acierto	Percentil	Acierto
50	76.59 %	67	83.60 %	84	88.03 %
51	77.83 %	68	84.09 %	85	88.71 %
52	77.83 %	69	84.59 %	86	89.66 %
53	77.80 %	70	84.58 %	87	89.91 %
54	78.29 %	71	85.07 %	88	89.62 %
55	78.82 %	72	85.22 %	89	89.61 %
56	79.90 %	73	85.72 %	90	89.73 %
57	79.77 %	74	86.18 %	91	89.81 %
58	79.74 %	75	86.02 %	92	90.03 %
59	80.83 %	76	85.92 %	93	90.59 %
60	81.88 %	77	85.86 %	94	91.38 %
61	81.78 %	78	86.81 %	95	91.37 %
62	81.78 %	79	87.28 %	96	91.29 %
63	81.90 %	80	87.43 %	97	91.21 %
64	82.15 %	81	87.33 %	98	91.31 %
65	82.30 %	82	87.33 %	99	90.87 %
66	83.10 %	83	87.57 %	100	88.48 %

Tabla 8.2.4.5 – Tabla acierto por percentiles en el experimento sin el error

8.2.4.6. Experimento sin la parte diferencial

Se tratará a la parte diferencial y a la integral por separado de la señal de control pese a formar parte ambas de ella. De esta forma se puede estudiar su valía por separado. En la figura 8.2.4.6 se ve lo que pasa si se elimina la parte diferencial.

En la tabla 8.2.4.6 se ve como el sistema acierta mucho menos si no tiene a la entrada la parte diferencial del regulador. El percentil con el que más se acierta es el 97, con un porcentaje de 91.56 %. En cambio, si se introduce dicho dato a la entrada se obtiene un acierto máximo del 97.78 % en el percentil 93 como se puede ver en la tabla 8.2.4.1. Esto supone una mejoría muy grande y deja patente la importancia de que la parte diferencial esté presente.

8.2.4.7. Experimento sin la parte integral

Por último se estudia la parte integral. En la figura 8.2.4.7 se ve lo que pasa si se elimina la parte integral.

En la tabla 8.2.4.7 se tiene un acierto del 91.99 % en el percentil 97 sin la parte integral mientras que, anteriormente, el la tabla 8.2.4.1 se tiene un acierto del 97.78 %. La diferencia es notable por lo que se decide incluir también esta parte en los datos de entrada.

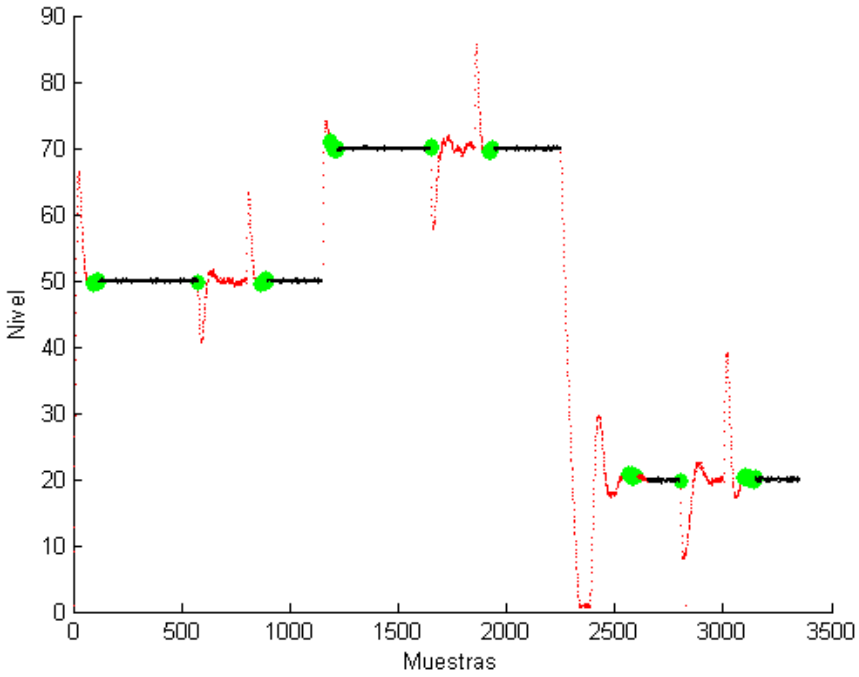

Figura 8.2.4.6 – Séptimo experimento: gráfica sin la parte diferencial del PID

Percentil	Acierto	Percentil	Acierto	Percentil	Acierto
50	74.12 %	67	82.91 %	84	89.07 %
51	74.61 %	68	83.93 %	85	88.97 %
52	75.42 %	69	84.18 %	86	88.75 %
53	75.70 %	70	84.74 %	87	88.96 %
54	75.67 %	71	85.26 %	88	88.70 %
55	76.44 %	72	85.26 %	89	88.97 %
56	76.41 %	73	86.04 %	90	89.52 %
57	77.18 %	74	86.50 %	91	89.32 %
58	77.99 %	75	87.03 %	92	89.52 %
59	77.89 %	76	86.96 %	93	89.39 %
60	78.94 %	77	86.93 %	94	89.97 %
61	80.06 %	78	87.21 %	95	89.74 %
62	80.87 %	79	87.88 %	96	90.40 %
63	80.84 %	80	88.10 %	97	91.56 %
64	81.33 %	81	88.31 %	98	90.98 %
65	81.86 %	82	89.09 %	99	89.91 %
66	82.14 %	83	89.14 %	100	88.36 %

Tabla 8.2.4.6 – Tabla acierto por percentiles en el experimento de la parte diferencial del PID

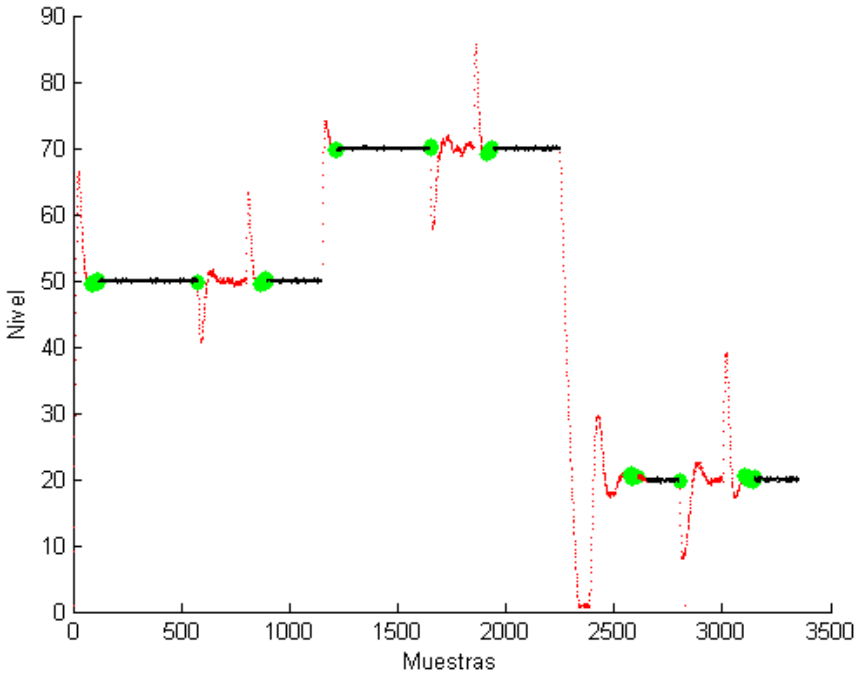

Figura 8.2.4.7 – Octavo experimento: gráfica sin la parte integral del PID

Percentil	Acierto	Percentil	Acierto	Percentil	Acierto
50	74.67 %	67	83.43 %	84	87.26 %
51	74.67 %	68	83.68 %	85	87.29 %
52	75.51 %	69	84.46 %	86	87.40 %
53	76.34 %	70	84.70 %	87	87.24 %
54	76.62 %	71	85.23 %	88	88.38 %
55	76.90 %	72	85.63 %	89	88.25 %
56	77.18 %	73	86.09 %	90	89.18 %
57	78.27 %	74	86.55 %	91	89.64 %
58	79.35 %	75	86.36 %	92	89.58 %
59	79.32 %	76	86.23 %	93	89.63 %
60	80.13 %	77	86.76 %	94	91.23 %
61	80.37 %	78	86.71 %	95	91.35 %
62	80.90 %	79	86.83 %	96	91.79 %
63	81.67 %	80	86.61 %	97	91.99 %
64	82.48 %	81	87.00 %	98	91.20 %
65	82.76 %	82	86.68 %	99	90.84 %
66	82.97 %	83	87.11 %	100	89.67 %

Tabla 8.2.4.7 – Tabla acierto por percentiles en el experimento de la parte integral del PID

8.3. Conclusiones

De esta serie de experimentos se pueden extraer tres conclusiones fundamentales:

- Se demuestra que no es necesario entrenar a un sistema para cada punto de funcionamiento ya que con un entrenamiento solo el sistema responde bien. Esto se puede ver en la figura 8.2.3.1.
- El autoencoder es una buena elección ya que se demuestra sobradamente que, si se le introducen las entradas adecuadas, detecta perfectamente los funcionamientos anómalos de la planta de laboratorio.
- Se ha demostrado también que la mejor combinación de variables de entrada son las seis principales: nivel, error, consigna, señal de control del PID, parte proporcional del PID y parte integral del PID.

TÍTULO: **DESARROLLO DE UN SISTEMA BASADO EN TÉCNICAS DE APRENDIZAJE COMPUTACIONAL PARA LA DETECCIÓN DE ANOMALÍAS EN UNA PLANTA DE NIVEL**

ANEXOS

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: DICIEMBRE DE 2017

AUTOR: EL ALUMNO

Fdo.: CHAS GESTAL, BRAIS

Índice del documento ANEXOS

9 Documentación de partida

9.1. Propuesta inicial de asignación del TFG

ESCUELA UNIVERSITARIA POLITÉCNICA

ASIGNACIÓN DE TRABAJO FIN DE GRADO

En virtud de la solicitud efectuada por:
En virtud da solicitude efectuada por:

APELLIDOS, NOMBRE: *Chas Gestal,Brais*
APELIDOS E NOME:

DNI: [redacted] **Fecha de Solicitud:** OCT2017
DNI: *Fecha de Solicitude:*

Alumno de esta escuela en la titulación de Grado en Ingeniería en Electrónica Industrial y Automática, se le comunica que la Comisión de Proyectos ha decidido asignarle el siguiente Trabajo Fin de Grado:

O alumno de esta escola na titulación de Grado en Enxeñería en Electrónica Industrial e Automática, comunícaselle que a Comisión de Proxectos ha decidido asignarlle o seguinte Traballo Fin de Grado:

Título T.F.G:DESARROLLO DE UN SISTEMA BASADO EN TÉCNICAS DE APRENDIZAJE COMPUTACIONAL PARA LA DETECCIÓN DE ANOMALÍAS EN UNA PLANTA DE NIVEL

Número TFG: *770G01A132*

TUTOR:(Titor) *Fontenla Romero,Oscar*

COTUTOR/CODIRECTOR: *José Luis Calvo Rolle*

La descripción y objetivos del Trabajo son los que figuran en el reverso de este documento*:*

A descrición e obxectivos do proxecto son os que figuran no reverso deste documento.

Ferrol a Viernes, 20 de Octubre del 2017

Retirei o meu Traballo Fin de Grado o dia _____ de _____ do ano _____

Fdo: Chas Gestal,Brais

DESCRIPCIÓN Y OBJETIVO:El presente proyecto tiene por objeto el desarrollo en Matlab de un sistema para la comunicación y el control automático de una planta de nivel de líquidos y la posterior detección de posibles anomalías, empleando para esto último un algoritmo del ámbito del aprendizaje computacional.

Alcance:
- Desarrollo de un regulador de tipo PID en Matlab para el control de la planta.
- Desarrollo de un sistema de comunicación entre la planta y el usuario mediante la utilización de una tarjeta de adquisición de datos.
- Análisis de los posibles algoritmos de detección de anomalías a utilizar y su aplicación en el ámbito de la planta.
- Realización de pruebas en diversas condiciones de trabajo y análisis de los resultados obtenidos en la planta de laboratorio.

10 Cálculos

10.1. Discretización del PID

La discretización del PID se realiza mediante transformada bilineal, más conocida como Método de Tustin.

$$C(s) = K \cdot \left(1 + \frac{1}{T_i s} + \frac{T_d s}{1 + \frac{T_d}{N} s}\right) \tag{10.1.0.1}$$

Siendo N un filtro paso bajo para eliminar el ruído.
Discretizando:

$$1 \to 1 \tag{10.1.0.2}$$

$$\frac{1}{T_i s} \to \frac{T_s}{2T_i} \frac{z+1}{z-1} \tag{10.1.0.3}$$

$$\frac{T_d s}{1 + \frac{T_d}{N} s} \to \frac{T_d \frac{2}{T_s}(z-1)}{z + 1 + \frac{T_d}{N}\frac{2}{T_s}(z-1)} = \frac{2T_d}{\frac{2T_d}{N} + T_s} \frac{z-1}{z - \frac{\frac{2T_d}{N} - T_s}{\frac{2T_d}{N} + T_s}} \tag{10.1.0.4}$$

Agrupando expresiones en coeficientes:

$$C_1 = \frac{T_s}{2T_i} \tag{10.1.0.5}$$

$$C_2 = \frac{2T_d}{\frac{2T_d}{N} + T_s} \tag{10.1.0.6}$$

$$C_3 = \frac{\frac{2T_d}{N} - T_s}{\frac{2T_d}{N} + T_s} \tag{10.1.0.7}$$

Donde T_s es el tiempo de muestreo (1s), T_d la constante derivativa, T_i la constante integral y N un filtro paso bajo. De aquí se obtiene:

- El error: $e[k] = r[k] - y[k]$.
- La parte integral: $i[k] = C_1(e[k] + e[k-1]) + i[k-1]$.
- La parte derivativa: $d[k] = C_2(y[k] - y[k-1]) + C_3 d[k-1]$

Quedando la señal de control final con la forma: $u[k] = K(e[k] + i[k] + d[k])$

10.2. Constantes del PID

Estos parámetros para ajustar el regulador se calculan para cada punto de funcionamiento. En primer lugar, como se explica en el análisis de las soluciones, se prueban los diferentes métodos de ajuste en torno a un SP del 50 %. Se obtiene del script con el rele antes mencionado un periodo (T_c) de 15 segundos y una amplitud (a) de 12.25. Se sabe que $d = 50$ y que $H = 5$ y se necesita calcular K_c:

$$K_c = \frac{4d}{\pi \cdot \sqrt{a^2 - H^2}} = \frac{200}{\pi \cdot \sqrt{12{,}25^2 - 5^2}} = 5{,}69 \tag{10.2.0.1}$$

Una vez calculada K_c, el primer método que se prueba para calcular las constantes es el de *Ziegler Nichols* en cadena cerrada:

$$K = 0{,}60 \cdot K_c = 0{,}60 \cdot 5{,}69 = 3{,}41 \tag{10.2.0.2}$$

$$T_i = 0{,}50 \cdot T_c = 0{,}50 \cdot 15{,}00 = 7{,}50 \tag{10.2.0.3}$$

$$T_d = 0{,}125 \cdot T_c = 0{,}125 \cdot 15{,}00 = 1{,}88 \tag{10.2.0.4}$$

A partir de estas fórmulas, algunos autores han propuesto modificaciones para mejorar la respuesta. Aquí se tendrán en cuenta dos modificaciones: una que mejora la sobreoscilación y otra que la anula.

Fórmulas de *Ziegler Nichols* modicicadas: poca sobreoscilación.

$$K = 0{,}33 \cdot K_c = 0{,}33 \cdot 5{,}69 = 1{,}88 \tag{10.2.0.5}$$

$$T_i = 0{,}50 \cdot T_c = 0{,}50 \cdot 15{,}00 = 7{,}50 \tag{10.2.0.6}$$

$$T_d = 0{,}33 \cdot T_c = 0{,}33 \cdot 15{,}00 = 5{,}00 \tag{10.2.0.7}$$

Fórmulas de *Ziegler Nichols* modicicadas: sin sobreoscilación.

$$K = 0{,}20 \cdot K_c = 0{,}20 \cdot 5{,}69 = 1{,}14 \tag{10.2.0.8}$$

$$T_i = T_c = 15{,}00 \tag{10.2.0.9}$$

$$T_d = 0{,}33 \cdot T_c = 0{,}33 \cdot 15 = 5{,}00 \tag{10.2.0.10}$$

Aparte de esto, las fórmulas se calculan también por el método de *Tyreus-Luyben*:

$$K = 0{,}45 \cdot K_c = 0{,}45 \cdot 5{,}69 = 2{,}56 \tag{10.2.0.11}$$

$$T_i = 2{,}20 \cdot T_c = 2{,}20 \cdot 15{,}00 = 33{,}00 \tag{10.2.0.12}$$

$$T_d = T_c/6{,}3 = 15{,}00/6{,}3 = 2{,}38 \tag{10.2.0.13}$$

10.3. Constantes para SP de 20 y 70

Para un punto de funcionamiento de 20:

$$Tc = 19 \tag{10.3.0.1}$$

$$K_c = \frac{4d}{\pi \cdot \sqrt{a^2 - H^2}} = \frac{80}{\pi \cdot \sqrt{13{,}3197^2 - 5^2}} = 2{,}06 \tag{10.3.0.2}$$

$$K = 0{,}33 \cdot K_c = 0{,}33 \cdot 2{,}06 = 0{,}68 \tag{10.3.0.3}$$

$$T_i = 0{,}50 \cdot T_c = 0{,}50 \cdot 19{,}00 = 9{,}50 \tag{10.3.0.4}$$

$$T_d = 0{,}33 \cdot T_c = 0{,}33 \cdot 19{,}00 = 6{,}33 \tag{10.3.0.5}$$

Para un punto de funcionamiento de 70:

$$Tc = 16 \tag{10.3.0.6}$$

$$K_c = \frac{4d}{\pi \cdot \sqrt{a^2 - H^2}} = \frac{280}{\pi \cdot \sqrt{16{,}69^2 - 5^2}} = 5{,}59 \tag{10.3.0.7}$$

$$K = 0{,}33 \cdot K_c = 0{,}33 \cdot 5{,}59 = 1{,}85 \tag{10.3.0.8}$$

$$T_i = 0{,}50 \cdot T_c = 0{,}50 \cdot 16{,}00 = 8{,}00 \tag{10.3.0.9}$$

$$T_d = 0{,}33 \cdot T_c = 0{,}33 \cdot 16{,}00 = 5{,}33 \tag{10.3.0.10}$$

Se recuerda que más tarde se multiplicará T_d por dos para mejorar la respuesta como se ha explicado en el análisis de las soluciones.

11 Códigos de programación

11.1. Parámetros del regulador PID

Este código es el utilizado para el Relay-Feedback del que se extraen los parámetros para calibrar el regulador PID.

Código 11.1: Código Relay-Feedback

```
% Inicialización de la tarjeta.
DAQ_Start;
tic

n = 90;                    % 1 minuto y medio
valores=zeros(2,n);
m = 1;
for i=1:n

    valor = DAQ_Read
    valores(1,i) = valor;
    valores(2,i) = i;

% Visualizar datos
    plot(valores(1,:))

    if valor < 45
        DAQ_Write(100,0);
    end
    if valor > 55
        DAQ_Write(0,0);
    end

%Paro a través de teclado
key=get(gcf,'currentcharacter');
if key=='q'
     break
end

% Pause for 1 seconds
```

```
    pause(m-toc)
    tic
end;

DAQ_Write(0,0)

% Datos a Excel
    filename = 'C:\Users\alumnado\Documents\MATLAB\Brais_Chas\Datos70.xlsx';
    xlswrite(filename,valores);

% Parar DAQ
DAQ_Stop
```

11.2. Regulador PID

En esta sección se especifica el código del script de Matlab con el que se controla la planta y se extraen luego sus datos en una matriz para el control de las anomalías.

Código 11.2: Código PID

```
% Inicialización de la tarjeta.
DAQ_Start;
tic

% Constantes.
SP = 50;                    % Consigna.
K = 1.88;
Ti = 7.5;
Td = 5;
tiempo = 250;
a=1;

%Vectores de datos:
V_Error = zeros(tiempo,1);
V_Nivel = zeros(tiempo,1);
V_SP = zeros(tiempo,1);
U_Integral = zeros(tiempo,1);
U_Diferencial = zeros(tiempo,1);

N=0.8;
Ts=1;                %Periodo de muestreo.

%Usando la discretización de Tustin:
C1 = (Ts/(2*Ti));
C2 = (2*Td)/(((2*Td)/N)+Ts);
C3 = (((2*Td)/N)-Ts)/(((2*Td)/N)+Ts);
```

```
for i=1:tiempo

    V_SP(i,1) = SP;
    V_Nivel(i,1) = DAQ_Read;
    V_Error(i,1)=K*(SP-V_Nivel(i,1));
    U_Integral(i,1)=C1*(V_Error(i,1)+V_Error(a,1))+U_Integral(a,1);
    U_Diferencial(i,1)=C2*(V_Error(i,1)-V_Error(a,1))+C3*(U_Diferencial(a,1));

    DAQ_Write(((V_Error(i,1))+(U_Integral(a,1))+(U_Diferencial(i,1))),0);

    plot(1:tiempo,V_Nivel(:,1),'r-',1:tiempo,V_SP(:,1),'k-');

    a=i;
    pause(1-toc);         %Temporización de 1 periodo menos el tiempo gastado
                          %en operacciones.
    tic;

    % Para parar el programa con la letra Q. También se para el script usando
    % Ctrl + C.
    key=get(gcf,'currentcharacter');
    if key=='q'
         break             % Rompe el bucle.
    end

    valores(1,i) = i
    valores(2,i) = V_Nivel(i,1);
    valores(3,i) = V_Error(i,1);
    valores(4,i) = U_Integral(i,1);
    valores(5,i) = U_Diferencial(i,1);

end

% Datos a Excel
    filename = 'C:\Users\alumnado\Documents\MATLAB\Brais_Chas\Datos1.xlsx';
    xlswrite(filename,valores);

% Al acabar el programa.
DAQ_Write(0,0)
DAQ_Stop
```

Este es el código utilizado para la toma de los datos que luego se utilizarán para los experimentos.

Código 11.3: Toma de datos

```
% Inicialización de la tarjeta.
DAQ_Start;
tic
```

```
% Constantes.
SP = 50;                    % Consigna.
K = 1.88;
Ti = 7.5;
Td = 10;
tiempo = 3350;
a=1;

%Vectores de datos:
V_Error = zeros(tiempo,1);
V_Nivel = zeros(tiempo,1);
V_SP = zeros(tiempo,1);
U_Integral = zeros(tiempo,1);
U_Diferencial = zeros(tiempo,1);

N=0.8;
Ts=1;                %Periodo de muestreo.

%Usando la discretización de Tustin:
C1 = (Ts/(2*Ti));
C2 = (2*Td)/(((2*Td)/N)+Ts);
C3 = (((2*Td)/N)-Ts)/(((2*Td)/N)+Ts);

for i=1:tiempo

    if i==1150
        SP = 70;                    % Consigna nueva.
        K = 1.847;
        Ti = 8;
        Td = 10.666;
    end

    if i==2250
        SP = 20;                    % Consigna nueva.
        K = 0.6807;
        Ti = 9.5;
        Td = 12.666;
    end

    V_SP(i,1) = SP;
    V_Nivel(i,1) = DAQ_Read;
    V_Error(i,1)=K*(SP-V_Nivel(i,1));
    U_Integral(i,1)=C1*(V_Error(i,1)+V_Error(a,1))+U_Integral(a,1);
    U_Diferencial(i,1)=C2*(V_Error(i,1)-V_Error(a,1))+C3*(U_Diferencial(a,1));

    DAQ_Write(((V_Error(i,1))+(U_Integral(a,1))+(U_Diferencial(i,1))),0);

    plot(1:tiempo,V_Nivel(:,1),'r-',1:tiempo,V_SP(:,1),'k-');
```

```
    a=i;
    pause(1-toc);          % Temporización de 1 periodo menos el tiempo gastado
                           % en operacciones.
    tic;

    % Para parar el programa con la letra Q. También se para el script usando
    % Ctrl + C.
    key=get(gcf,'currentcharacter');
    if key=='q'
        break              % Rompe el bucle.
    end

    valores(1,i) = i
    valores(2,i) = V_Nivel(i,1);
    valores(3,i) = V_Error(i,1);
    valores(4,i) = U_Integral(i,1);
    valores(5,i) = U_Diferencial(i,1);
    valores(6,i) = SP;
    valores(7,i) = ((V_Error(i,1))+(U_Integral(a,1))+(U_Diferencial(i,1)));

end

% Datos a Excel
    filename = 'C:\Users\alumnado\Documents\MATLAB\Brais_Chas\Datos2TD.xlsx';
    xlswrite(filename,valores);

% Al acabar el programa.
DAQ_Write(0,0)
DAQ_Stop
```

11.3. Detección de anomalías

Este es el código del programa principal en el que se encuentra el autoencoder.

Código 11.4: Sistema de detección de anomalías

```
close all;
clear all;

NKFOLD = 10;
NORMALIZACION = 1;     % 0 sin normalizar - 1 entre [0 1] - 2 zscore

NEURALFUN = 2;         % Función neuronal en la salida: 1-lineal, 2-logsig,
                       % 3-tanh
cNor = 1;              % Clase considerada como normal para entrenar (0 o 1)
                       % o (1,2,3...)
claseN = cNor;
```

```
cOut = abs(cNor-1);  % Clase considerada como outlier para entrenar

umbral = 28;

% Selecciona la función neuronal
if (NEURALFUN == 7)
    fh     = 'logsig';           % Función de activación en la capa oculta
    fn     = 'lineal_m';         % Función de activación en la capa de salida
    finv   = 'ilineal_m';
    fderiv = 'dlineal_m';
elseif (NEURALFUN == 2)
    fh     = 'logsig';           % Función de activación en la capa oculta
    fn     = 'ilogsig';          % Función de activación en la capa de salida
    finv   = 'logsig_m';
    fderiv = 'dilogsig';
elseif (NEURALFUN == 3)
    fh     = 'tansig';           % Función de activación en la capa oculta
    fn     = 'atanh';            % Función de activación en la capa de salida
end


% Carga los datos
disp('Valores');
load datosplanta50_70_20_2D.mat;

variables = [1, 2, 3, 4, 5, 6]           % Las filas que quiero utilizar de
                                         % la matriz de datos.
XX = valores_20_50_70_2D(variables,:);
dd = anomalias_2D(1,:);
cNor=1;
cOut=0;

HS=length(variables)-1;    % Número de neuronas en la capa oculta (una menos
                           % que variables)

%%%%%%%Normalización previa de los datos
if (NORMALIZACION == 1)
    for i = 1:size(XX,1)
        XX(i,:) = (XX(i,:)-min(XX(i,:)))/(max(XX(i,:))- min(XX(i,:))) *
        (0.99995-0.00005) + 0.00005;
    end
elseif(NORMALIZACION == 2)
    XX = zscore(XX,1,2);
end

XNor = XX(:,dd==cNor);
classNor = dd(dd==cNor);
XOut = XX(:,dd==cOut);
classOut = dd(dd==cOut);
```

```
[~,nNor] = size(XNor);

for hs=HS

    hiddenSize=hs;
    cd SVD;

    CVO = cvpartition(nNor,'KFold', NKFOLD);

    for i = 1:1
        % Se seleccionan los datos de train y test para esta iteración
        Xtrain     = XNor(:,CVO.training(i));
        classTrain = classNor(CVO.training(i));
        Xtest      = [XOut XNor(:,CVO.test(i))];
        classTest  = [classOut classNor(CVO.test(i))];
        [~,n] = size(Xtrain);
        [~,ntest] = size(Xtest);

        % Entrenamiento del Autoencoder
        [salidasTr, salidasTest, W1, W2, tr_time, tst_time] = SVD_autoencoder
        (Xtrain,Xtest,hiddenSize,fh,fn,finv,fderiv);

        tr_times(i) = tr_time;
        tst_times(i) = tst_time;

        % Cálculo del error de reconstrucción a la salida
        errores = (salidasTr(1:end,:)-Xtrain(1:end,:)).^2';
        med_errores = mean(errores,2);

        erroresTest = (salidasTest(1:end,:)-Xtest(1:end,:)).^2';
        med_erroresTest = mean(erroresTest,2);

        erroresTest1 = (salidasTest(1:end,1:length(classOut))-
        Xtest(1:end,1:length(classOut))).^2';
        med_erroresTest1 = mean(erroresTest1,2);

        erroresTest2 = (salidasTest(1:end,length(classOut)+1:end)-
        Xtest(1:end,length(classOut)+1:end)).^2';
        med_erroresTest2 = mean(erroresTest2,2);


        Clase = cNor*ones(1,ntest);
        maxErrTr = max(med_errores);
        meanErrTr = mean(med_errores);
        stdErrTr = std(med_errores);
        % Clasificación: criterio 1 (máximo)

        for per=50:100
            cr = per-49;
            Clase = cNor*ones(1,ntest);
```

```
    percentil(cr) = prctile(med_errores,per);
    Clase(med_erroresTest > percentil(cr)) = cOut;
    Aciertos(i,cr) = 100*sum(Clase == classTest)/ntest;
    [tp(i,cr), fp(i,cr)] = AUC(classTest, Clase, cOut);
    acc(i,cr) = trapz([0 fp(i,cr) 1],[0 tp(i,cr) 1]);
end

%% GRAFICA ERROR DE RECONSTRUCCIÓN

n2 = ntest-length(classOut);

if(i==1)
    figure;
    plot(1:n,med_errores,'k.');
    hold on;
    plot(n+1:n+length(classOut),med_erroresTest1,'r.');
    plot(n+length(classOut)+1:n+length(classOut)+n2,
    med_erroresTest2,'g.');
    ylabel('Error')
    xlabel('Muestras')
    % title('Errores de entrenamiento (negro) y errores de test
    % (rojo y verde)');

    % plot(n+1:n+length(classOut),valores_20_50_70_2D(1,dd==0)/100,
    % 'm');

    figure;
    plot(find(anomalias_2D==1),
    valores_20_50_70_2D(1,anomalias_2D==1),'k.');
    hold on;
    plot(find(anomalias_2D==0),
    valores_20_50_70_2D(1,anomalias_2D==0),'r.');
    ylabel('Nivel')
    xlabel('Muestras')
    % title('Nivel real de la planta (en rojo los datos etiquetados
    % como anómalos)');

    [~,umbral] = max(acc(i,:));

    anomaliasTest1 = (med_erroresTest1 > percentil(umbral));
    anomaliasTest2 = (med_erroresTest2 > percentil(umbral));

    figure;
    indices = find(anomaliasTest1==1);
    plot(indices,med_erroresTest1(anomaliasTest1),'r.');
    hold on;
    indices = find(anomaliasTest1==0);
    plot(indices,med_erroresTest1(~anomaliasTest1),'g.');
    % plot(1:length(classOut),valores_20_50_70_2D(1,dd==0)/200,'m');
    plot([length(classOut) length(classOut)],[0 0.2]);
```

```
            ylabel('Error')
            xlabel('Muestras')
            % title('Clasificación del sistema: en rojo los datos clasificados
            % como anómalos y en verde como normales');

            indices = find(anomaliasTest2==1);
            plot(length(classOut)+1+indices,med_erroresTest2(anomaliasTest2),
            'r.');
            indices = find(anomaliasTest2==0);
            plot(length(classOut)+1+indices,med_erroresTest2(~anomaliasTest2),
            'g.');

            figure;
            ylabel('Nivel')
            xlabel('Muestras')
            %title('En verde los datos anomalos que el sistema etiqueta como
            %normales');
            hold on;
            indices  = find(anomalias_2D==0);
            indices1 = find(anomaliasTest1);
            for i = 1:length(indices)
                if (anomaliasTest1(i)==1)
                    plot(indices(i), valores_20_50_70_2D(1,indices(i)),'r.',
                    'MarkerSize',1);
                else
                    plot(indices(i), valores_20_50_70_2D(1,indices(i)),'g.',
                    'MarkerSize',20);
                end
                plot(find(anomalias_2D==1),valores_20_50_70_2D
                (1,anomalias_2D==1),'k.','MarkerSize',1);
            end
        end
    end

    %-------------------------------

    fprintf('\n----------------\nTest Accuracy\n----------------');
    for j = 1:cr  % Medidas
        fprintf('\n %i & %.2f\\%% \\\\ ', j+49,  100*mean(acc(:,j)));
    end
    fprintf('\n');

    cd ..;

end
```

TÍTULO: **DESARROLLO DE UN SISTEMA BASADO EN TÉCNICAS DE APRENDIZAJE COMPUTACIONAL PARA LA DETECCIÓN DE ANOMALÍAS EN UNA PLANTA DE NIVEL**

PLANOS

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: DICIEMBRE DE 2017

AUTOR: EL ALUMNO

Fdo.: CHAS GESTAL, BRAIS

En el presente trabajo no aplica una sección de planos.

TÍTULO: **DESARROLLO DE UN SISTEMA BASADO EN TÉCNICAS DE APRENDIZAJE COMPUTACIONAL PARA LA DETECCIÓN DE ANOMALÍAS EN UNA PLANTA DE NIVEL**

PLIEGO DE CONDICIONES

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: DICIEMBRE DE 2017

AUTOR: EL ALUMNO

Fdo.: CHAS GESTAL, BRAIS

Índice del documento PLIEGO DE CONDICIONES

En el presente trabajo no aplica un pliego de condiciones.

TÍTULO: **DESARROLLO DE UN SISTEMA BASADO EN TÉCNICAS DE APRENDIZAJE COMPUTACIONAL PARA LA DETECCIÓN DE ANOMALÍAS EN UNA PLANTA DE NIVEL**

ESTADO DE MEDICIONES

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: DICIEMBRE DE 2017

AUTOR: EL ALUMNO

Fdo.: CHAS GESTAL, BRAIS

Índice del documento ESTADO DE MEDICIONES

En el presente trabajo no aplica una sección de estado de mediciones.

TÍTULO: **DESARROLLO DE UN SISTEMA BASADO EN TÉCNICAS DE APRENDIZAJE COMPUTACIONAL PARA LA DETECCIÓN DE ANOMALÍAS EN UNA PLANTA DE NIVEL**

PRESUPUESTO

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: DICIEMBRE DE 2017

AUTOR: EL ALUMNO

Fdo.: CHAS GESTAL, BRAIS

Índice del documento PRESUPUESTO

12 Presupuesto

12.1. Materiales, mano de obra y elementos auxiliares

Concepto	Cantidad	Precio unitario	Total
Mano de obra de un graduado en Ingeniería Electrónica y Automática	420 horas	50€	21.000€
Licencia de Matlab	1 licencia	2.000€	2.000€
TOTAL:			23.000€

12.2. Amortizaciones

Elemento	Precio	Amortización Total	Amortización en 3 meses	Coste
Ordenador portátil	745€	4 años	6,25 %	46,57€
TOTAL:				46,57€

12.3. Totales

Elemento	Coste
Mano de obra y auxiliares	23.000€
Amortizaciones	46,57€
TOTAL:	23.046,57€

TÍTULO: **DESARROLLO DE UN SISTEMA BASADO EN TÉCNICAS DE APRENDIZAJE COMPUTACIONAL PARA LA DETECCIÓN DE ANOMALÍAS EN UNA PLANTA DE NIVEL**

ESTUDIOS CON ENTIDAD PROPIA

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: DICIEMBRE DE 2017

AUTOR: EL ALUMNO

Fdo.: CHAS GESTAL, BRAIS

Índice del documento ESTUDIOS CON ENTIDAD PROPIA

En el presente trabajo no se realizan estudios con entidad propia.