UNIVERSIDADE DA CORUÑA

**ESCUELA UNIVERSITARIA POLITÉCNICA**

**Grado en Ingeniería Eléctrica**

**Grado en Ingeniería Electrónica Industrial y Automática**

# TRABAJO FIN DE GRADO

TFG. Nº: **770G01A021**

TÍTULO: **MEJORA DEL ALGORITMO DE CONTROL DE UNA PLANTA DE LABORATORIO E IDENTIFICACIÓN DE LA MISMA**

AUTOR: GABRIEL CALVO RODRÍGUEZ

TUTOR: JOSE LUIS CALVO ROLLE

SUPLENTE: JOSÉ LUIS CASTELEIRO ROCA

FECHA: **SEPTIEMBRE DE 2013**

Fdo.: EL AUTOR Fdo.: EL TUTOR

TÍTULO: **MEJORA DEL ALGORITMO DE CONTROL DE UNA PLANTA DE LABORATORIO E IDENTIFICACIÓN DE LA MISMA**

---

# ÍNDICE GENERAL

---

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

**AVDA. 19 DE FEBRERO, S/N**

**15405 - FERROL**

FECHA: **SEPTIEMBRE DE 2013**

AUTOR: EL ALUMNO

Fdo.: GABRIEL CALVO RODRIGUEZ

1. ÍNDICE GENERAL……2
2. MEMORIA……..……..9
2.1 Objeto. ……..…….…… 9
2.2 Alcance...……..……... 9
2.3 Antecedentes..……..…… 10
2.4 Normas y referencias……..…… 10
2.4.1 Bibliografía…….. 10
2.4.2 Programas de cálculo…… 11
2.5 Definiciones y abreviaturas..…… 11
2.6 Requisitos del diseño ……..……11
2.7 Análisis de soluciones...……...12
2.7.1 Regulador anti Windup.……... 12
2.7.2 Cambio diseño planta.……..…… 14
2.7.3 Identificación planta.……..…… 16
2.8 Resultados finales..……17
2.8.1 Descripción general.…….. 17
2.8.2 Regulador PID…… 22
2.8.2.1 Introducción…..……... 22
2.8.2.2 Implementación en Matlab…..……..….. 25
2.8.2.2.1 Introducción…….……... 25
2.8.2.2.2 Modificaciones.....……... 26
2.8.3 Bloque anti Windup..…… 28
2.8.3.1 Back calculation……..……...31
2.8.3.1.1 Análisis……..…….. 31
2.8.3.1.2 Implementación..…….…….. 33
2.8.3.1.3 Resultados…….…… 35
2.8.3.1.4 Conclusiones…….…… 36
2.8.3.2 Tracking Mode.…….……36
2.8.3.2.1 Análisis…… 36
2.8.3.2.2 Implementación…….. 37
2.8.3.2.3 Resultados……..… 41
2.8.3.2.4 Conclusiones…..…… 45
2.8.3.3 Integración condicional……..……46

**2.8.3.3.1 Análisis…… ...46**
**2.8.3.3.2 Implementación…… 46**
**2.8.3.3.3 Resultados…… 47**
**2.8.3.3.4 Conclusiones…… 48**
**2.8.4 Identificación planta……...49**
**2.8.4.1 Descripción general……49**
**2.8.4.2 Método mínimos cuadrados...……50**
**2.8.4.3 Métotodo mínimos cuadrados recursivos……..55**
**2.8.4.3.1 Explicación matemática…… 55**
**2.8.4.3.2 Implementación en Matlab…… 60**
**2.8.4.3.3 Resultados…… 65**
**2.8.4.3.4 Conclusiones…… 65**
**2.9 Orden de prioridad entre los documentos básicos……66**
**3. ANEXOS……..67**
**3.1 Contenido …… 69**
**3.1.1 Documentación de partida……..69**
**3.1.2 Cálculos……70**
**3.1.3 Otros anexos……70**
**3.1.3.1 Código empleado……70**
**3.1.3.1.1 Configuracionviejo…… 71**
**3.1.3.1.2 PIDSetupviejo..…… 71**
**3.1.3.1.3 Llamar_PIDSetupviejo…… 71**
**3.1.3.1.4 Regulador_backcalcul…… 72**
**3.1.3.1.5 PIDactualizar…… 74**
**3.1.3.1.6 Configuracionrapida...…… 74**
**3.1.3.1.7 PIDSetup……..…… 74**
**3.1.3.1.8 Llamar_PIDSet…..…….. 75**
**3.1.3.1.9 Regulador_3senhales...…… 75**
**3.1.3.1.10 Regulador_integracioncondicionada…..... 78**
**3.1.3.1.11 Identificación…… 80**
**4. PLIEGO DE CONDICIONES…....……83**
**4.1 Especificaciones de los materiales……..……..85**
**4.1.1 Introducción……85**

**4.1.2 Selección de componentes…… 85**
**4.1.2.1 Sistema de adquisición de datos…..……85**
**4.1.2.2 Actuador…….…….…… 86**
**4.1.2.3 Planta……..…….. 86**
**4.1.3 Especificaciones de montaje…..……..……... 86**
**4.1.4 Condiciones software……..……..……... 87**
**4.1.5 Condiciones hardware ..…… 87**
**4.1.6 Pruebas…….……..……... 88**
**5. ESTADO DE MEDICIONES…….…..……..89**
**5.1 Listado de materiales…….…..……91**
**5.2 Mano de obra…….……..……91**
**5.2.1 Desarrollo del proyecto……...…….. 91**
**5.2.2 Montaje del proyecto…..…… 92**
**6. PRESUPUESTO.....…….……93**
**6.1 Cuadro de precios…..…….. 95**
**6.2 Gastos de desarrollo……..…….. 95**
**6.2.1 Coste de personal……..……..…… 95**
**6.3 Coste total……..……..…….. 96**

TÍTULO: **MEJORA DEL ALGORITMO DE CONTROL DE UNA PLANTA DE LABORATORIO E IDENTIFICACIÓN DE LA MISMA**

---

# MEMORIA

---

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

**AVDA. 19 DE FEBRERO, S/N**

**15405 - FERROL**

FECHA: **SEPTIEMBRE DE 2013**

AUTOR: EL ALUMNO

Fdo.: GABRIEL CALVO RODRIGUEZ

# ÍNDICE

**2. MEMORIA……...9**

**2.1 Objeto. …… 9**

**2.2 Alcance ...…….. 9**

**2.3 Antecedentes..…….…… 10**

**2.4 Normas y referencias……..…… 10**

**2.4.1 Bibliografía…….……10**

**2.4.2 Programas de cálculo…… 11**

**2.5 Definiciones y abreviaturas ……..…… 11**

**2.6 Requisitos del diseño..……..…… 11**

**2.7 Análisis de soluciones…….…... 12**

**2.7.1 Regulador anti Windup……..……12**

**2.7.2 Cambio diseño planta……..……..…14**

**2.7.3 Identificación planta…….……..……16**

**2.8 Resultados finales.……17**

**2.8.1 Descripción general…… 17**

**2.8.2 Regulador PID……. 22**

**2.8.2.1 Introducción…..……..……... 22**

**2.8.2.2 Implementación en Matlab…..…….……25**

**2.8.2.2.1 Introducción……..……25**

**2.8.2.2.2 Modificaciones.. ……26**

**2.8.3 Bloque anti Windup……..……28**

**2.8.3.1 Back calculation……..……....31**

**2.8.3.1.1 Análisis……..…….. 31**

**2.8.3.1.2 Implementación……..……33**

**2.8.3.1.3 Resultados…….…… 35**

**2.8.3.1.4 Conclusiones…….……36**

**2.8.3.2 Tracking Mode. ……..……36**

**2.8.3.2.1 Análisis……..36**

**2.8.3.2.2 Implementación……...37**

**2.8.3.2.3 Resultados.……..….41**

**2.8.3.2.4 Conclusiones ……..……...45**

**2.8.3.3 Integración condicional……46**

**2.8.3.3.1 Análisis……46**

**2.8.3.3.2 Implementación……46**

**2.8.3.3.3 Resultados…… 47**

**2.8.3.3.4 Conclusiones……48**

**2.8.4 Identificación planta……49**

**2.8.4.1 Descripción general…….. 49**

**2.8.4.2 Método mínimos cuadrados…….. 50**

**2.8.4.3 Métotodo mínimos cuadrados recursivos…… 55**

**2.8.4.3.1 Explicación matemática.……55**

**2.8.4.3.2 Implementación en Matlab……60**

**2.8.4.3.3 Resultados……63**

**2.8.4.3.4 Conclusiones……64**

**2.9 Orden de prioridad entre los documentos básicos……64**

## 2.1 Objeto

En el presente proyecto se pretende realizar una mejora en un regulador mediante la implementación de un bloque anti Windup. El bloque ha de ser programado mediante línea de comandos en MATLAB y se comprobará su funcionamiento en una planta de laboratorio. Además, se desarrollará una función que permita obtener la función de transferencia del sistema a través del uso del algoritmo RLS (mínimos cuadrados recursivo).

## 2.2 Alcance

El proyecto consta del estudio de las distintas posibilidades de realización de un bloque anti Windup y sus implementaciones. Las distintas pruebas y comparaciones se realizarán en una planta de laboratorio donde la temperatura será controlada por el usuario, pudiendo ser modificada en tiempo real. La respuesta del sistema será monitorizada en tiempo real y los parámetros que definen el funcionamiento del sistema serán modificables durante el tiempo de ejecución.

También se realizará una identificación de la planta mediante el método de mínimos cuadrados recursivos, su implementación en el regulador y comparación de resultados obtenidos.

Todos los resultados obtenidos serán analizados y comparados con distintas mediciones para obtener unas adecuadas conclusiones.

De una manera más concisa, se enumeran a continuación las distintas partes que justifican el alcance del proyecto.

- Diseño de un bloque de programa anti Windup.
- Inclusión de esta nueva función en el regulador existente.
- Comparación de la respuesta del regulador original, y con la inclusión de esta nueva función.
- Implementación del algoritmo RLS para la identificación de la planta.
- Sintonización del regulador para la función de transferencia obtenida.

- Comparación de los resultados obtenidos con el regulador sintonizado de manera empírica y la teórica.

## 2.3 Antecedentes

Este proyecto es una modificación y mejora del proyecto de fin de carrera nº 770611A415 realizado por Gabriel Calvo Rodríguez. El código del citado proyecto ha sido empleado como base sobre la cuál realizar modificaciones. En este proyecto se han empleado y modificado las funciones *Regulador_1*, *Configuracionrapida* y *PIDSetup*.

Cabe nombrar también el proyecto nº770611A310 redactado por Francisco Zayas Gato. En este proyecto se han empleado el actuador, sensor y acondicionador de señal diseñados por él.

## 2.4 Normas y referencia

### 2.4.1 Bibliografía

Libros:

– D. Montgomery, G. Runger: Probabilidad y estadística aplicadas a la ingeniería.

– C. Angulo, C.Raya: Tecnología de sistemas de control.

Bibliografía digital:

http://materias.fi.uba.ar/6631/material/Clase_05_Minimos_Cuadrados.pdf

http://alumnos.elo.utfsm.cl/~aflores/identsistemas.pdf

http://www.depeca.uah.es/depeca/repositorio/asignaturas/32328/Tema5.pdf

http://www.control-class.com/Tema_2/Slides/Tema_2_IdentificacionMinimosCuadrados.pdf

### 2.4.2 Programas de cálculo

Los programas informáticos empleados para la realización de este proyecto son:

- MatlabR2010b: Programa de cálculo matemático empleado en la resolución de todos los cálculos e incógnitas.
- EdrawMindMap 6.5: Programa de dibujo asistido por ordenador empleado en la creación de gráficos, esquemas y flujogramas.

## 2.5 Definiciones y abreviaturas

PID: Proporcional Integral Derivativo.

TAD: Tarjeta de adquisición de datos.

RLS: Mínimos cuadrados recursivos.

## 2.6 Requisitos del diseño

La planta empleada para la realización de este proyecto es una caja de metacrilato con 2 bombillas incandescentes de 100W cada una, que se usan para emular el comportamiento de un horno con el objetivo de mantener una temperatura constante en el interior. La temperatura se medirá mediante un LM35 con un rango de medida de -55ºC a 150ºC. EL sensor estará situado a una distancia de 10 cm para evitar que salga de su rango de medida.

Para la implementación del regulador se necesitará de un sistema de adquisición de datos, para ello se ha elegido una TAD NI-DAQ USB 6229 de la casa National Instruments. La citada tarjeta se encargará de enviar una señal de control de 0 a 10 V al actuador para que este entregue de forma lineal del 0 al 100% de la potencia de la red a la planta. Este sistema recibirá también la señal de salida del acondicionador del sensor de temperatura, La señal de salida del acondicionador indica la temperatura en el interior del horno con un factor de escala de 0,1V/ºC (0-10V, 0-100ºC). El procesamiento y tratado de estas señales para un correcto funcionamiento del regulador se realizará mediante un PC.

## 2.7 Análisis de las soluciones

El proyecto puede dividirse en 2 partes principales, la primera es la implementación de un bloque anti Windup, con una comparación respecto al funcionamiento inicial, y la segunda es la implementación de algoritmo de identificación mediante RLS y la realización de pruebas y comparaciones.

### 2.7.1 Regulador anti Windup

El error debido al efecto Windup es un error provocado por una confrontación entre un sistema real (con limitaciones y saturación) y un sistema ideal. En toda la implementación práctica, la señal de salida del controlador debe estar limitada por dos razones:

- La primera es que su magnitud no debe de exceder el rango permitido por el sistema de adquisición de datos, y la segunda razón es que su valor no debe de exceder el rango permitido por el actuador.
- La segunda es la lentitud de respuesta del sistema, limitado por la potencia máxima del sistema. Esto provocará que el controlador calcule una señal de control dirigida al actuador mayor de la admisible, por lo que saturará. Como el controlador no tiene en cuenta esta situación, el error y, por tanto, la señal se irán acrecentando más y más. Cuando el estado de la planta se acerque a la consigna, el sistema se comportará como si tuviera “inercia” (o recordara) y mantendrá el alto valor de la señal de control incluso cuando el error es 0. Este efecto es debido a la parte integral del controlador, que es una suma acumulativa el error. La consecuencia de esto efecto es que el lazo de realimentación se rompe y el sistema opera como un sistema en lazo abierto porque el actuador permanecerá en su límite independientemente de la salida del proceso.

Cabe decir que con el tiempo este error iría disminuyendo, pero provocaría oscilaciones indeseadas, que en ciertos casos hasta podrían dañar al sistema.

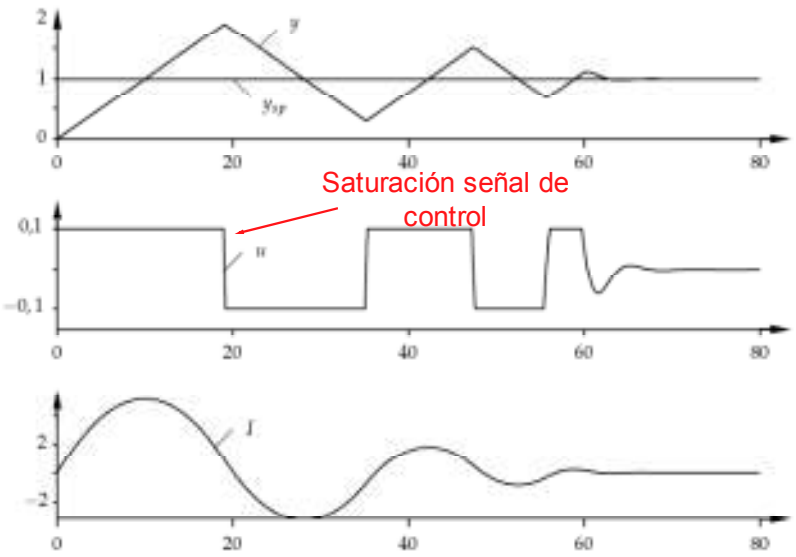



Figura 2.7.1.1-Efecto windup

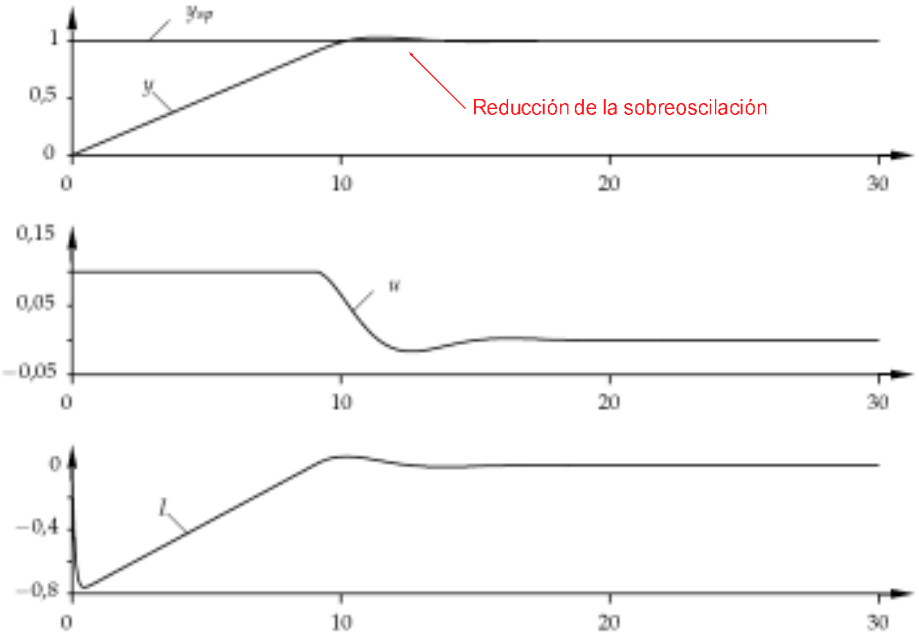



Figura 2.7.1.2 – Respuesta del sistema con regulador anti-windup

En el programa original ya existía una sentencia que se encargaba de actualizar el valor de salida del controlador, pero resultaba insuficiente. En caso de que la señal de control a enviar al actuador superara los límites establecidos (0 y 10 V), se le actualizaría el valor para ajustarlo al intervalo preestablecido y se envía el valor corregido a las variables de memoria. Esta pequeña modificación ayuda al programa a que no exista un desfase entre la variable de memoria y la señal real enviada al actuador.

```
if SALIDA < 0
        SALIDA = 0;
end
if SALIDA>10
        SALIDA = 10;
end
```

Tras un estudio de las distintas posibilidades para un controlador anti Windup más complejo, se escogieron 3 distintas formas de implementación: “Back-Calculation”, integración condicional y “Tracking Mode”. Su implementación, comprobación y análisis se encuentra en el apartado 2.8.3.

### 2.7.2 Cambio diseño planta

Durante la realización de este proyecto se realizó la implementación del bloque anti Windup con “Tracking Mode”. Las características de este bloque requerían el empleo de 2 bits para la selección de modo de funcionamiento. Sin embargo, el diseño previo del regulador empleaba como señales de control 2 señales analógicas.

El método original resultaba engorroso, poco práctico y muy limitado, puesto que en laboratorio solamente hay una fuente de tensión con 2 salidas. El añadir más fuentes de tensión no era una idea práctica, pues supone demasiado cableado y no soluciona los problemas de funcionalidad.

La solución propuesta fue aprovechar una de las facetas de la TAD, los canales de señales digitales. Aprovechando la señal de 5V de la tarjeta (patilla 96) con una Project Board se implementó un sistema de conexiones rápido e intuitivo.

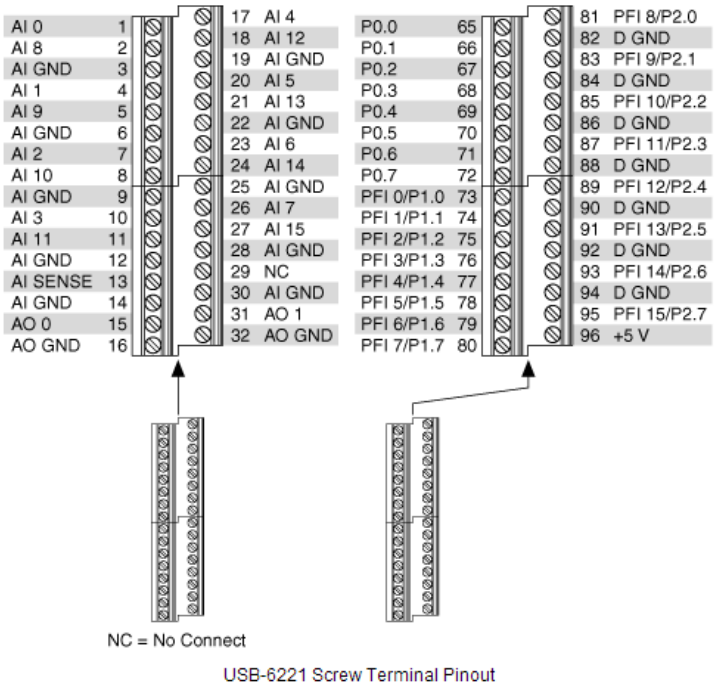



Figura 2.7.2.1 – Conexiones TAD

Como se puede ver en la imagen 2.7.2.2, al estar los pines interconectados entre sí, los 5V llegan a los distintos cables empleados.

Figura 2.7.2.2 – Conexiones empleadas

Simplemente con conectar/desconectar los cables en la Project Board se le puede enviar una señal de orden al PID.

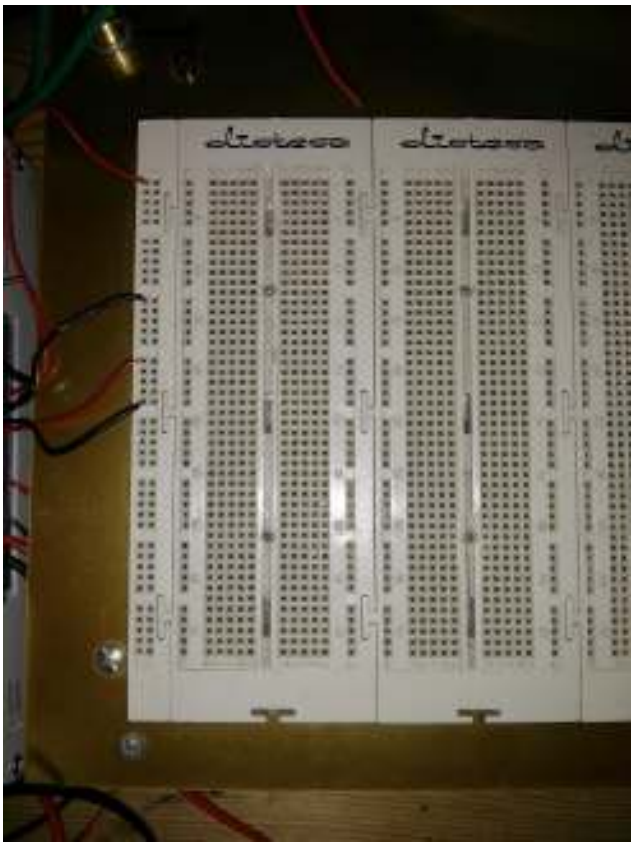


Figura 2.7.2.3 - Conexiones Project Board

Otra de las ventajas de este método es que los canales analógicos y digitales empleados funcionan de forma independiente, pudiendo seleccionar y obtener datos de cada uno de forma adecuada y en el momento óptimo.

### 2.7.3 Identificación planta

Para la identificación del sistema, fue necesario realizar un estudio sobre los posibles métodos a emplear, como puede ser los basados en el error del sistema o el sistema de mínimos cuadrados.

Sin embargo, el método de los mínimos cuadrados recursivos tiene ciertas ventajas. Es un método paramétrico, por lo que se calculan los valores de un modelo ya

ideado. Además, su baja necesidad de realizar cálculos permite realizar el proceso de identificación mientras se comprueban los resultados en tiempo real.
Esto lo hace el método idóneo para la identificación de un sistema y el cálculo de su función de transferencia.

## 2.8 Resultados finales

### 2.8.1 Descripción general

Los objetivos de este proyecto son la mejora diseño de un controlador para una planta tipo horno industrial como el de la figura 2.8.1.1 mediante un bloque anti Windup. También se procederá a la identificación de la planta mediante el método de los mínimos cuadrados recursivos.

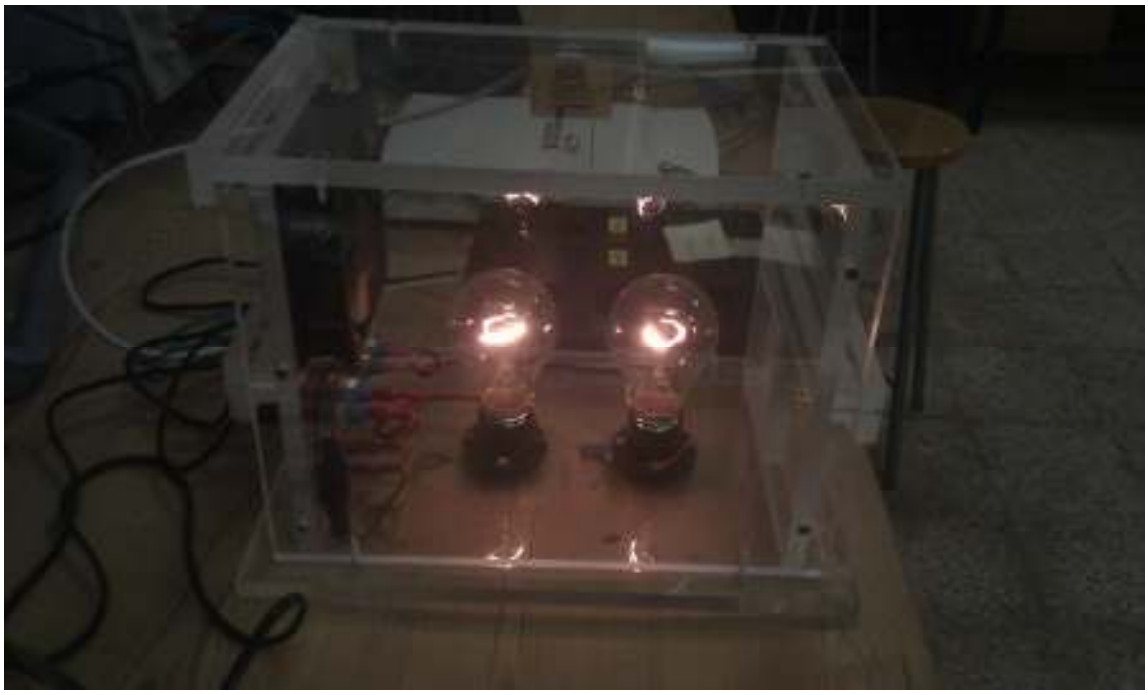


Figura 2.8.1.1 - Planta

El programa empleado será Matlab y se programará mediante línea de comandos. El usuario podrá salir del proceso de regulación o modificar los parámetros del regulador durante su ejecución, de una forma rápida y efectiva. Por lo tanto el

usuario podrá modificar el regulador de modo que este cumpla todos los requisitos y necesidades que este crea necesario. El programa monitoreará la señal de temperatura de la planta en tiempo real mediante una gráfica.

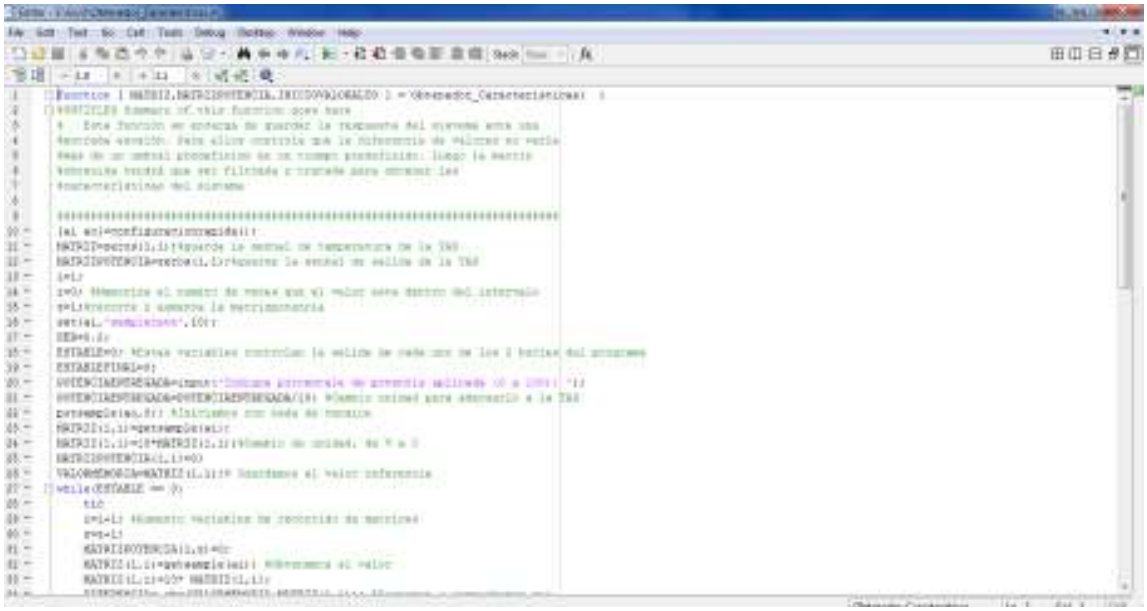


Figura 2.8.1.2 – Ejemplo programación mediante línea de comandos

A pesar de estar ya completamente implementado, es conveniente explicar el funcionamiento de un sistema de control con realimentación, donde se distinguen los siguientes bloques funcionales:

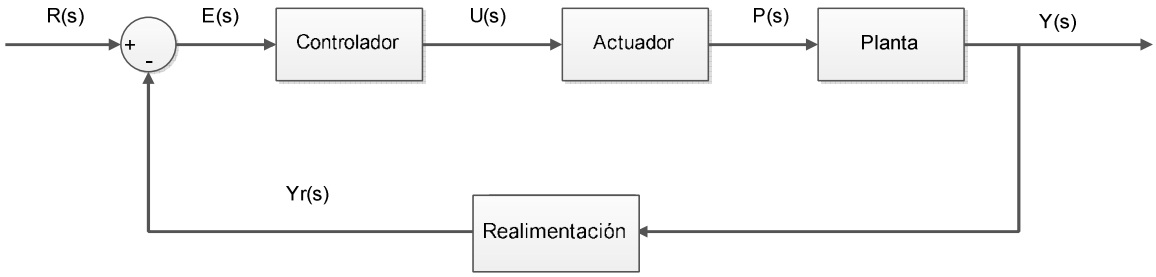



Figura 2.8.1.3 – Sistema con realimentación

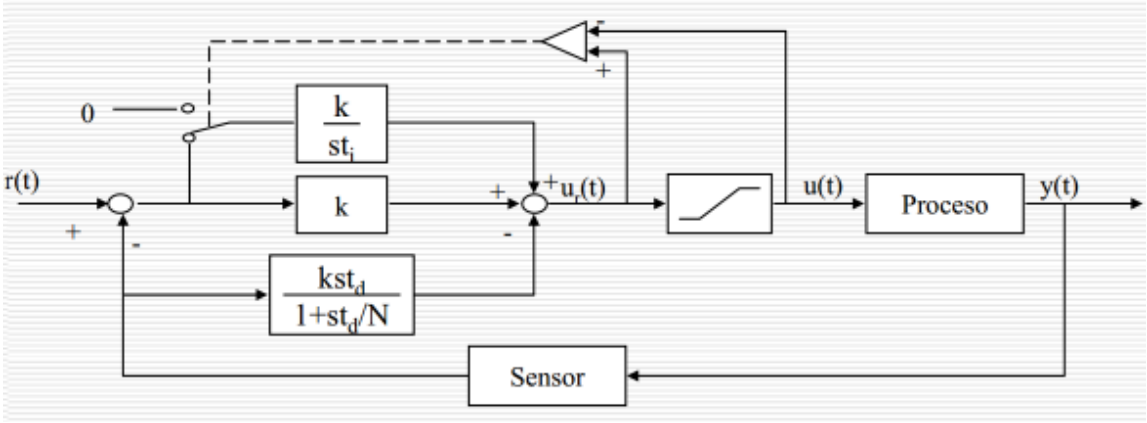
- Controlador
- Actuador
- Planta
- Sensor

Durante el proceso en el que la señal de entrada *R(s)* provoca una respuesta en forma de señal de salida *Y(s)* los distintos bloques funcionarán de tal modo que la planta trabaje de forma adecuada. La señal de entrada se considera la temperatura deseada por el usuario (denominada consigna o punto de trabajo) y la señal de salida es la temperatura que hay en el interior de dicha planta.

En este proyecto solamente se diseñará una mejora del controlador de la planta, aunque el conocimiento total del sistema es necesario para comprender la necesidad de ésta.

El controlador recibe la diferencia entre los valores de punto de trabajo y estado actual de la planta. El punto de trabajo *R(s)* vendrá definido por el usuario y la señal *Yr(s)* será transmitida a través del sensor. El controlador actuará de forma acorde con sus características definidas por el usuario y en base a los estados anteriores de las señales *R(s)* e *Yr(s)*. Tras haber realizado una serie de cálculos, enviará la señal *U(s)* al actuador. La señal *U(s)* se encarga de controlar la potencia aplicada por el actuador a la planta.

Para evitar la aparición del error producido por el efecto Windup, es necesario implementar un bloque funcional que lo minimice, para ello ha de situarse después del cálculo de la señal en el regulador, como vemos en la imagen 2.8.1.4.




2.8.1.4 – Esquema bloque anti Windup

El actuador entrega el 100% de la potencia que adquiere al conectarse a la red eléctrica cuando la señal de control U(s) es de 10V. En caso de ser menor, entregará una potencia menor. Este proceso es controlado mediante un Triac. En las figuras 2.8.1.5, 2.8.1.6 y 2.8.1.7 se puede apreciar cómo la potencia aplicada varía en función de la señal de control. En la primera imagen se aplican 2 voltios como señal de control, en la siguiente 5 y en la última 8.

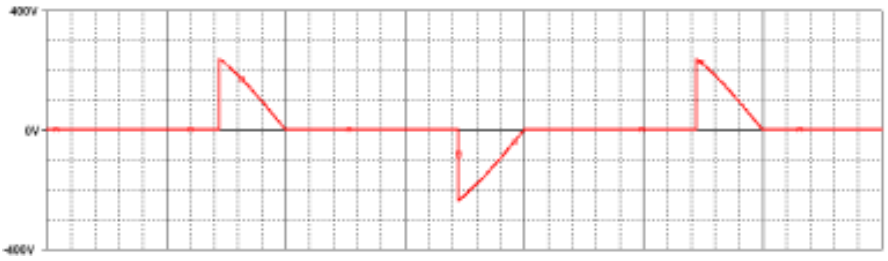


Figura 2.8.1.5– Respuesta actuador ante señal de control de 2 V

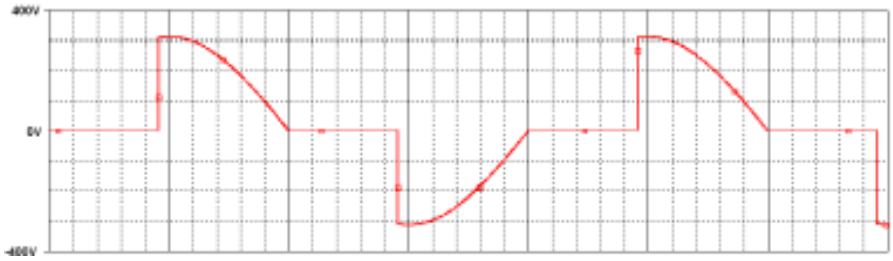


Figura 2.8.1.6– Respuesta actuador ante señal de control de 5 V

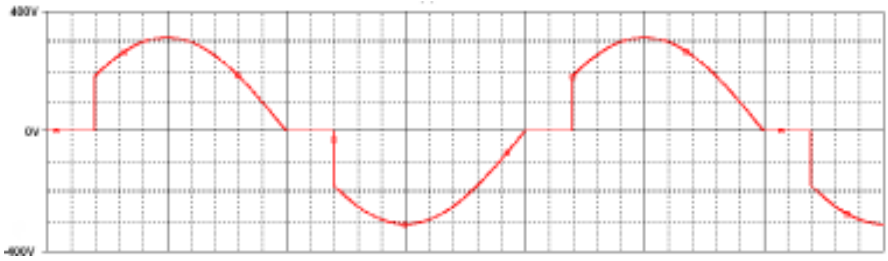


Figura 2.8.1.7 – Respuesta actuador ante señal de control de 8 V

Finalmente la potencia *P(s)* se transmite a la planta, donde alimenta las bombillas. Al iluminarse las bombillas provocan calor que hará aumentar la temperatura dentro de la planta. Dentro de la planta existe un sensor de temperatura LM35 que se encarga de transformar la temperatura de la planta *Y(s)* en una señal de voltaje. La señal del sensor será de 0 a 1V, la señal será amplificada para que su rango sea 0-10V y así concuerde con el rango de tensión de entrada de la TAD y así aprovechar toda su escala. A esta señal se le denominará *Yr(s)* y es muy importante que sea de la misma naturaleza que la consigna. Esto es muy importante dado que, al ser la señal de realimentación, necesita ser de la misma naturaleza que la consigna para obtener una señal de error correcta.

Existen una serie de limitaciones debido a la naturaleza del sistema. Los sistemas de control de temperatura presentan una respuesta bastante lenta que puede provocar ciertos problemas. Otro de los puntos a destacar es su falta de sistema de

refrigeración, lo que impide que la temperatura de la planta sea menor a la temperatura ambiente. Solamente cuenta con un sistema de ventilación, usado como interferencia o para para acelerar el enfriamiento natural de la planta.

Otro punto importante a tener cuenta es la cercanía de cables y metacrilato a las bombillas. Si se aplica una alta potencia durante un período largo de tiempo se puede llegar a fundir el plástico que recubre los cables o el metacrilato puede llegar a deteriorarse. Teniendo esto en cuenta, se recomienda que la temperatura empleada como punto de trabajo permanezca entre el intervalo de 30-80 ºC.

### 2.8.2 Regulador PID

#### 2.8.2.1 Introducción

A pesar de estar ya completamente implementado, es necesario hacer una pequeña reseña al regulador PID ya implementado con anterioridad, ya que el completo entendimiento de su funcionamiento es vital para la comprensión de cómo afecta el bloque anti Windup.

Un PID es un mecanismo de control por realimentación que calcula la desviación o error entre un valor medido y el valor que se quiere obtener para aplicar una acción correctora que ajuste el proceso. El algoritmo de cálculo del control PID se da en tres parámetros distintos: el proporcional, el integral y el derivativo. El valor proporcional determina la reacción del error actual. El integral genera una corrección proporcional a la integral del error, esto nos asegura que aplicando un esfuerzo de control suficiente, el error de seguimiento se reduce a cero. El derivativo determina la reacción del tiempo en el que el error se produce. La suma de estas 3 acciones es usada para ajustar al proceso vía un elemento de control, en este caso la temperatura de la planta.

$$u(t) = Ki(t)\int_0^t e(\tau)d\tau + Kp(t)e(t) + Kd\frac{d\,e(t)}{dt}$$

(2.8.2.1.1)

El modelo matemático de este regulador se basa en 3 parámetros Kp, Ki, Kd:

En esta fórmula se puede identificar:

- u(t)= Señal de control.
- e(t)= Señal de error.
- Ki= Ganancia integral.
- Kp= Ganancia proporcional.
- Kd= Ganancia diferencial.

Una forma más fácil de ver es mediante el siguiente flujograma:

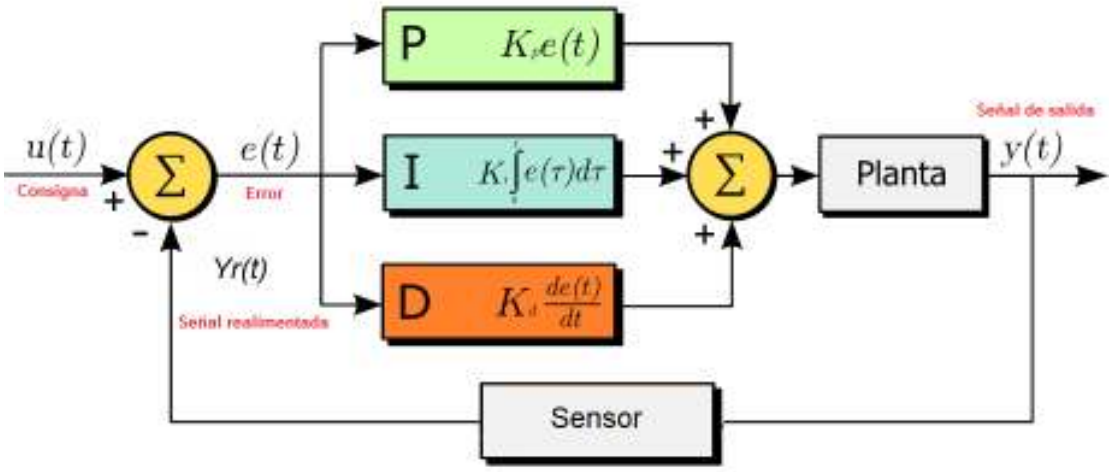



Figura 2.8.2.1.1- Modelo matemático sistema con regulador PID

Como podemos ver, la señal de error *e(t)* se dirige a los 3 bloques funcionales que dependen de los parámetros, donde es transformada de forma distinta en cada bloque. Luego, estas 3 nuevas señales se suman y se aplican a la planta donde se genera la señal de salida. Para el cálculo de la señal de error, es necesario un sensor que capte la señal de salida *Y(t)*, la transforme en *Yr(t)* para que sea acorde con la consigna *u(t)* y de la diferencia de ambas se obtenga la señal de error.

El modelo matemático mostrado en la imagen 2.8.2.1.1 es el más utilizado en temas educacionales debido a su sencillez. Sin embargo, en la práctica no tiene aplicación útil debido a que en él se muestra la posibilidad de no implementar la acción proporcional del regulador. La acción proporcional es la acción inmediata de control y por tanto es importante que aparezca siempre, por tanto no tiene sentido la implementación de acciones integrales y/o derivativas sin acción proporcional.

Para la realización de este proyecto, se ha empleado el formato estándar (ISA) como modelo para el regulador. La fórmula del regulador en formato estándar es la siguiente:

$$u(t) = K\left[E(t) + \frac{1}{Ti}\int_0^t e(\tau)d\tau + Td\frac{de(t)}{dt}\right]$$

(2.8.2.1.2)

Donde:
K : Ganancia proporcional.
Ti : Tiempo integral.
Td : Tiempo derivativo.

Estos nuevos parámetros dependen de los valores de los parámetros anteriores.
Se puede establecer una equivalencia:

| | |
|---|---|
| K | Kp |
| Ti | Kp/Ki |
| Td | Kd/Kp |

Tabla 2.8.2.1.1 – Relación parámetros

Es muy importante destacar que durante la realización de las pruebas con distintos modelos se han mantenido los valores de los parámetros para comprobar su comportamiento. Esto significa que tendrán un comportamiento similar, pero no igual, ya que se darán casos en los que alguna de sus componentes no forme parte del regulador (como es el caso de un regulador PD o un regulador PI). Esto se ha realizado de esta forma que los comportamientos de los reguladores no sufran grandes variaciones y sus comportamientos puedan ser comparados. Sin embargo, para una optimización del sistema es recomendable que se realicen una serie de pruebas y mediciones para mejorar el comportamiento del sistema en cada uno de los distintos casos.

#### 2.8.2.2 Implementación en Matlab

##### 2.8.2.2.1 Introducción

El regulador empleado como base es la función *Regulador_1* del PFC 770611A415 y es un regulador de posición. En caso de necesitar una explicación más en detalle de su funcionamiento, por favor diríjase al citado proyecto en el apartado 2.8.7.2.2.2 de su memoria.

En el flujograma a continuación se puede observar su funcionamiento.

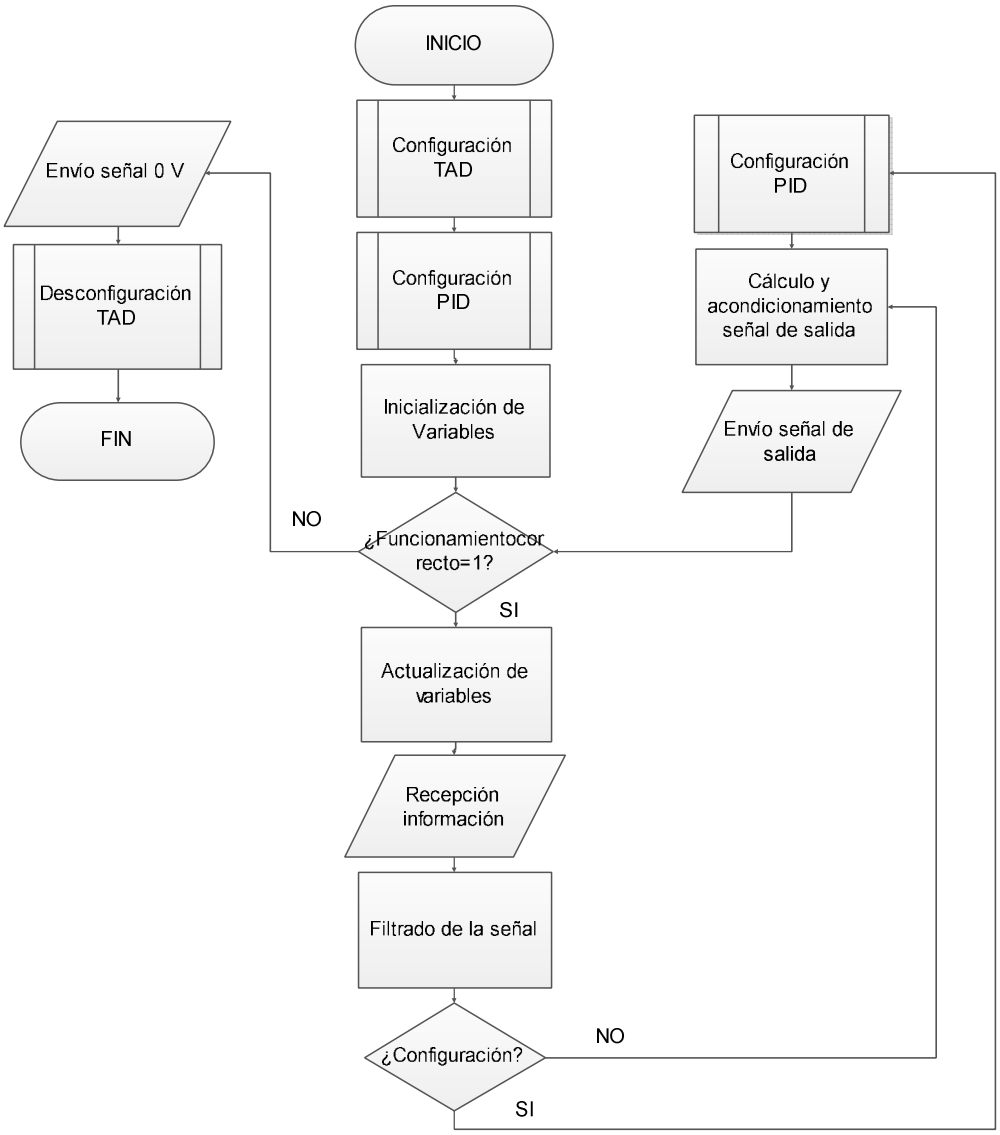



Figura 2.8.2.2.1.1 – Flujograma función Regulador1

### 2.8.2.2.2 Modificaciones

Como ya se ha citado anteriormente, durante la implementación del bloque “Tracking Mode” como bloque anti Windup surgieron una serie de cuestiones que necesitaron de una modificación de la planta para su correcto funcionamiento. En este apartado se procederá a explicar qué variaciones se han realizado, su implementación y funcionamiento.

En el regulador original se empleaban 3 señales de entrada analógica y una de salida, la adquisición de las 3 señales de entrada se realizan a la vez con una única sentencia.

```
ai=analoginput('nidaq','Dev1');
addchannel(ai,0);
addchannel(ai,5);
addchannel(ai,6);
ao=analogoutput('nidaq','Dev1');
addchannel(ao,0);
```

Sin embargo, al necesitarse ahora más canales para una correcta selección del modo de trabajo, se decidió por añadir un canal de entrada digital, del cual se emplearían 4 bits. Por lo tanto ahora se emplea un canal analógico de entrada, otro de salida y 4 bits digitales.

```
ai=analoginput('nidaq','Dev1');
addchannel(ai,0);
addchannel(ao,0);
dio=digitalio('nidaq','Dev1');
addline(dio,0:3,0,'in');
```

Ahora la función configuracionrapida tiene nuevos argumentos de salida.

```
[AI AO DIO]=configuracionrapida();
```

La llamada a los distintos medios de entrada se realizará de forma separada e independiente.

```
memoriaentradasucia(1,1)=getsample(AI);
valordigitalentrada=getvalue(DIO);
```

### 2.8.3 Bloque anti Windup

Como se ha indicado anteriormente, se han implementado 3 distintos bloques anti Windup para comprobar su funcionamiento. En los siguientes apartados, se procederá a realizar una explicación detallada de cada uno de ellos, su implementación y a la obtención de una serie de conclusiones en función de los resultados obtenidos.

El comportamiento de los distintos reguladores con bloque anti Windup serán comparados con los el comportamiento de un regulador PID con sus parámetros obtenidos a partir de la fórmula de Chien et al para sistemas con cambio de carga con 0% de sobreoscilación. Se ha escogido esta implementación puesto que ha sido la que mejores resultados ha obtenido durante las pruebas realizadas en el proyecto nº770611A415. La consigna se establecerá a 30ºC y se medirá su tiempo de respuesta, establecimiento y pico.

Esta implementación se obtiene a partir de la sintonización en cadena abierta. La sintonía consiste en la medición de ciertas características de la respuesta del sistema en cadena abierta (sin realimentación). La estimulación realizada será una entrada tipo escalón.

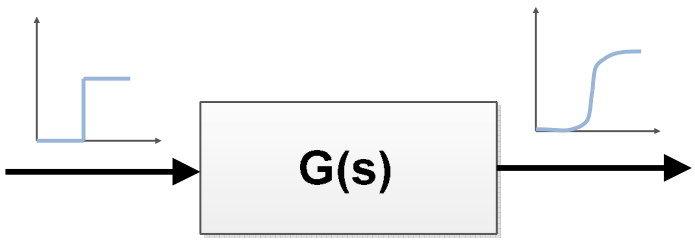



Figura 2.8.3.1 – Respuesta sistema 1er orden

La señal que conforma su salida es la respuesta de un sistema de 1º orden con retardo. La forma de la señal es similar a una S, la señal se caracteriza por 3

constantes: Tiempo de retardo (T) y la constante de tiempo (L) y una ganancia estática (K). Su fórmula es:

$$G_p(s) = K\frac{e^{-Ls}}{Ts+1} \qquad (2.8.3.1)$$

La obtención de estos parámetros se realiza de forma gráfica:

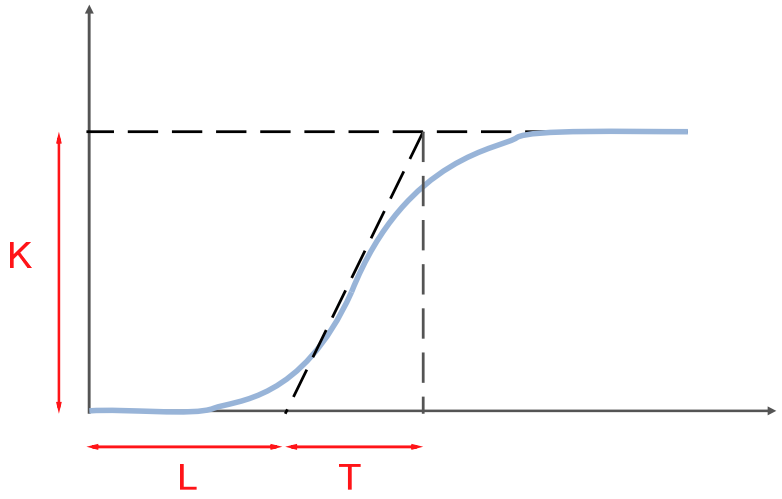



Figura 2.8.3.2 – Cálculo parámetros lazo abierto

Para las distintas pruebas realizadas, se ha determinado que se emplearán los siguientes parámetros:

| | |
|---|---|
| K | 1 |
| L | 5s |
| T | 450s |

Tabla 2.8.3.1 – Parámetros empleados

Siendo los parámetros del regulador, configurado según Chien et al:

| Kp | Ki | Td |
|---|---|---|
| $\frac{0.95T}{KL}$ | $2.4L$ | $0.42L$ |
| 85.5 | 12 | 2.1 |

Tabla 2.8.3.2 - Parámetros Chien et al

En la imagen a continuación se muestra la respuesta del sistema con sus características:

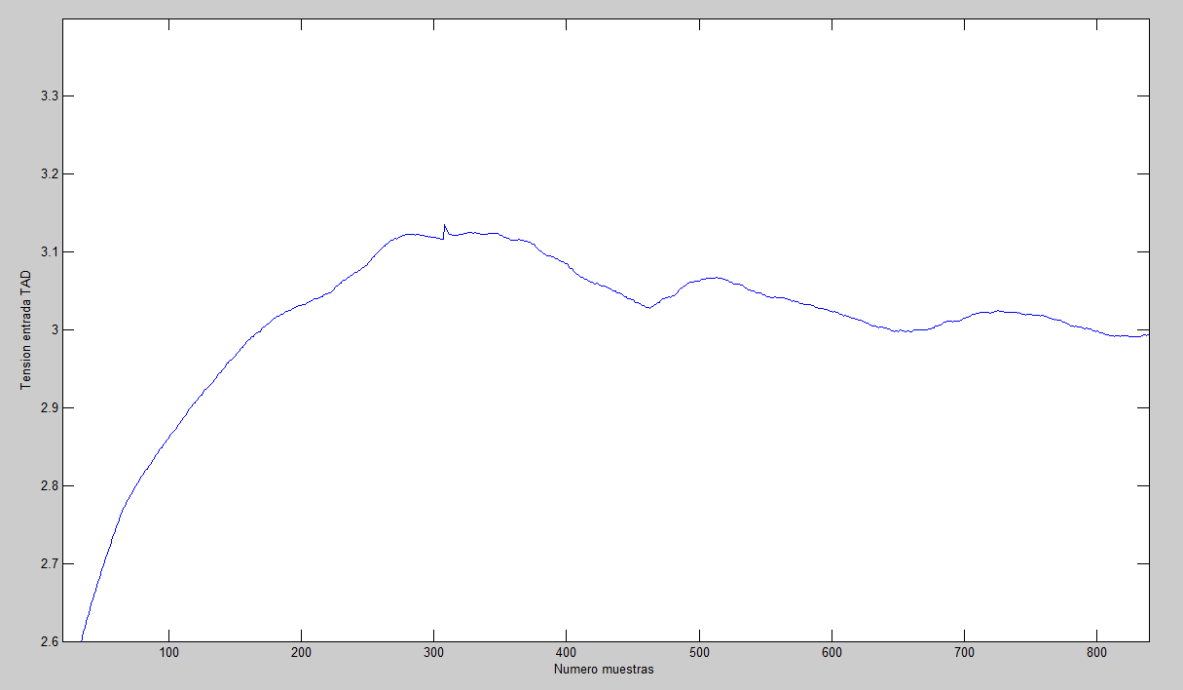



Figura 2.8.3.3 – Respuesta regulador con parámetros Chien et Al

Tiempo de respuesta: 12.2s.

Tiempo de establecimiento: 22s.

Tiempo de pico: 61s.

### 2.8.3.1 Back Calculation

#### 2.8.3.1.1 Análisis

El método de *Back Calculation* funciona de la siguiente forma: cuando la salida del actuador se satura, la acción integral es recalculada de forma que el nuevo valor aporte una salida en el límite. Es conveniente no rehacer el integrador instantáneamente, sino con un cierto retardo de constante de tiempo.

El sistema presenta un lazo de control adicional generado al medir la salida real del actuador *u(t)* y la salida del controlador *v(t)*, generando una señal de diferencia *e(t)*.

$$e_s(t) = u(t) - v(t)$$

(2.8.3.1.1.1)

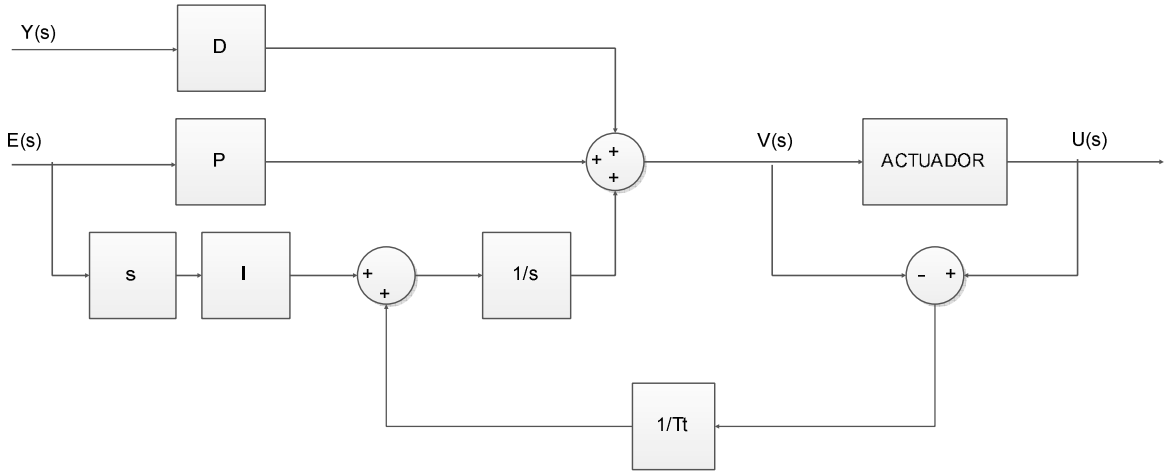



Figura 2.8.3.1.1.1 - Flujograma regulador con bloque Back Calculation

Puesto que la voluntad es que la entrada de la acción integral no aumente cuando el actuador está saturado, la señal $e_s(t)$ ha de cumplir una condición. Se puede establecer la entrada modificada en el integrador de la siguiente forma:

$$e_s(t) * \frac{1}{T_t} + \frac{K}{T_i} * e(t)$$

(2.8.3.1.1.2)

Por lo tanto:

$$e_s(t) = -\frac{KT_t}{T_i} * e(t)$$

(2.8.3.1.1.3)

Aplicando esta definición, la salida del controlador queda definida de la siguiente forma:

$$v(t) = L + \frac{KT_t}{T_i} * e(t)$$

(2.8.3.1.1.4)

Dónde:

L: Valor de saturación del actuador.

$T_t$: Constante de tiempo de seguimiento.

$T_i$: Constante de tiempo de la parte integral.

El valor de la constante de tiempo de seguimiento, $T_t$, es necesario que tenga un valor entre las constantes de tiempo de la parte integral y derivativa para que no produzca un error debido a un ruido repentino ni que produzca un reset de la acción integral.

$$T_d < T_t < T_i$$

(2.8.3.1.1.5)

En general, se define:

$$T_t = \sqrt{T_i T_d}$$

(2.8.3.1.1.6)

En este caso, al estar en un ámbito discreto, el valor de $T_t$ es el valor de período de muestreo.

Como se puede observar, este método trabaja en el ámbito del tiempo continuo (s), y no en un ámbito de tiempo discreto. Esto puede provocar que este método no sea aplicable y el resultado sea un controlador que no cumpla unos requisitos mínimos o que sea inestable.

#### 2.8.3.1.2 Implementación

La función empleada para implementar este regulador es la función *Regulador_backcalculation*. Este modo se creó antes de las variaciones exigidas por

el “Tracking Mode”, como se puede apreciar en el apartado 2.7.2, por lo que la configuración y cierta nomenclatura es distinta.
El regulador funciona del mismo modo que el regulador original, con sus señales de control sin variar.

```
TENSIONESENTRADA=getsample(AI);
CONTROL=TENSIONESENTRADA(1,2);
LLAMADACONFIGURAR=TENSIONESENTRADA(1,3);
```

Sin embargo, una nueva función de transferencia ha de ser calculada en cada nuevo estado de la planta. Para ello se realiza una llamada a la función *PIDactualizar*, esta función necesita de la entrada de 2 datos SALIDAPLANTA y SALIDAACTUADOR. La variable SALIDAPLANTA es el valor de señal teórica dirigida a la planta por el actuador (calculada por un regulador PID), mientras que SALIDAACTUADOR es la señal que realmente envía el actuador, saturado en caso de necesidad.

```
SALIDAPLANTA=NUM(1)*MEMORIAERROR(1,1);
SALIDAPLANTA=SALIDAPLANTA+(NUM(2)*MEMORIAERROR(1,2));
SALIDAPLANTA=SALIDAPLANTA+(NUM(3)*MEMORIAERROR(1,3));
SALIDAPLANTA=SALIDAPLANTA-(DEN(2)*MEMORIASALIDA(1,2));
SALIDAPLANTA=SALIDAPLANTA-(DEN(3)*MEMORIASALIDA(1,3));
SALIDAPLANTA=SALIDAPLANTA/DEN(1);
SALIDAACTUADOR=SALIDAPLANTA;

if SALIDAACTUADOR < 0
        SALIDAACTUADOR = 0;
end
if SALIDAACTUADOR>10
        SALIDAACTUADOR = 10;
end
```

Estas señales, que se calculan al final de cada ciclo, sirven como valores de referencia para el cálculo de $e_s(t)$ del siguiente estado y serán enviados como valores de entrada para la función *PIDactualizar*.

```
function [NUM,SIZENUM,DEN,SIZEDEN] =
PIDactualizar(SALIDAPLANTA,SALIDAACTUADOR)
ERRORACTUADOR=SALIDAACTUADOR-SALIDAPLANTA;
Per = 0.2;
K =108;
Ti =10+(ERRORACTUADOR/(Per));
Td = 2.26;
G = K*tf([Ti*Td Ti 1], [Ti 0]);
TF = c2d(G,Per,'tustin');
NUM=TF.num{1};
SIZENUMTOTAL=size(NUM);
SIZENUM=SIZENUMTOTAL(2);
DEN=TF.den{1};
SIZEDENTOTAL=size(DEN);
SIZEDEN=SIZEDENTOTAL(2);
```

Vemos como el error de la planta es la diferencia entre las 2 variables de entrada de la función y como afecta al cálculo de una nueva parte integral, tal como se ha mostrado en el diagrama anteriormente. A continuación se procede al cálculo de una nueva función de transferencia y su separación en bloques para permitir el cálculo.
El cálculo de la nueva señal de salida será análogo a la versión original del regulador.

```
SALIDAPLANTA=NUM(1)*MEMORIAERROR(1,1);
SALIDAPLANTA=SALIDAPLANTA+(NUM(2)*MEMORIAERROR(1,2));
SALIDAPLANTA=SALIDAPLANTA+(NUM(3)*MEMORIAERROR(1,3));
SALIDAPLANTA=SALIDAPLANTA-(DEN(2)*MEMORIASALIDA(1,2));
SALIDAPLANTA=SALIDAPLANTA-(DEN(3)*MEMORIASALIDA(1,3));
SALIDAPLANTA=SALIDAPLANTA/DEN(1);
SALIDAACTUADOR=SALIDAPLANTA;
```

#### 2.8.3.1.3 Resultados

Los resultados obtenidos se muestran en la figura 2.8.3.1.3.1 y muestran como la temperatura aumenta sin apreciarse ningún tipo de regulación. Al comprobarse que el sistema es inestable se ha procedido a la paralización del sistema. En 2 minutos ya se habían alcanzado aproximadamente los 45ºC, por lo que evitar que la temperatura siga subiendo resulta indispensable para evitar daños en la planta.

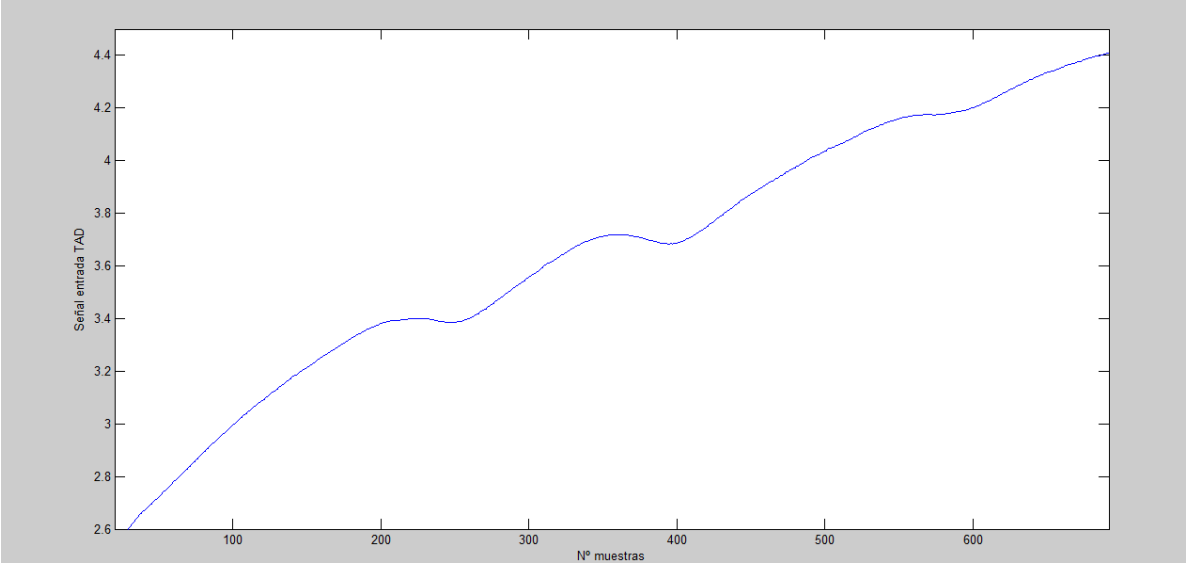



Figura 2.8.3.1.3.1 – Respuesta sistema con bloque Back caltulation

#### 2.8.3.1.4 Conclusiones

Que el nuevo sistema resulte totalmente inestable, cuando debería ser una mejora de su comportamiento es debido a la naturaleza de las señales. Todas las señales aquí mostradas trabajan en un ámbito continuo (t), sin embargo, la planta real trabaja con señales discretas.

Podría resultar posible que con la variación de los parámetros resultara un sistema estable, aunque este resultado no obedecería al planteamiento del Back Calculation.

Esto es una clara muestra de la diferencia de comportamiento debido a la naturaleza de sus señales y por qué, lo que a priori resulta una solución adecuada, resulta no serlo.

### 2.8.3.2 Tracking Mode

#### 2.8.3.2.1 Análisis

Un controlador en modo seguimiento o “Tracking Mode” tiene dos métodos de funcionamiento. El primero de ellos es el funcionamiento normal de un regulador, sin embargo, es posible también dirigir de forma externa la acción integral.

Para ello se implementará una señal externa que modificará el comportamiento del controlador. Esta señal W(s) no se obtiene a partir del comportamiento del actuador, si no que se dispone de una señal de seguimiento o *tracking* externo.

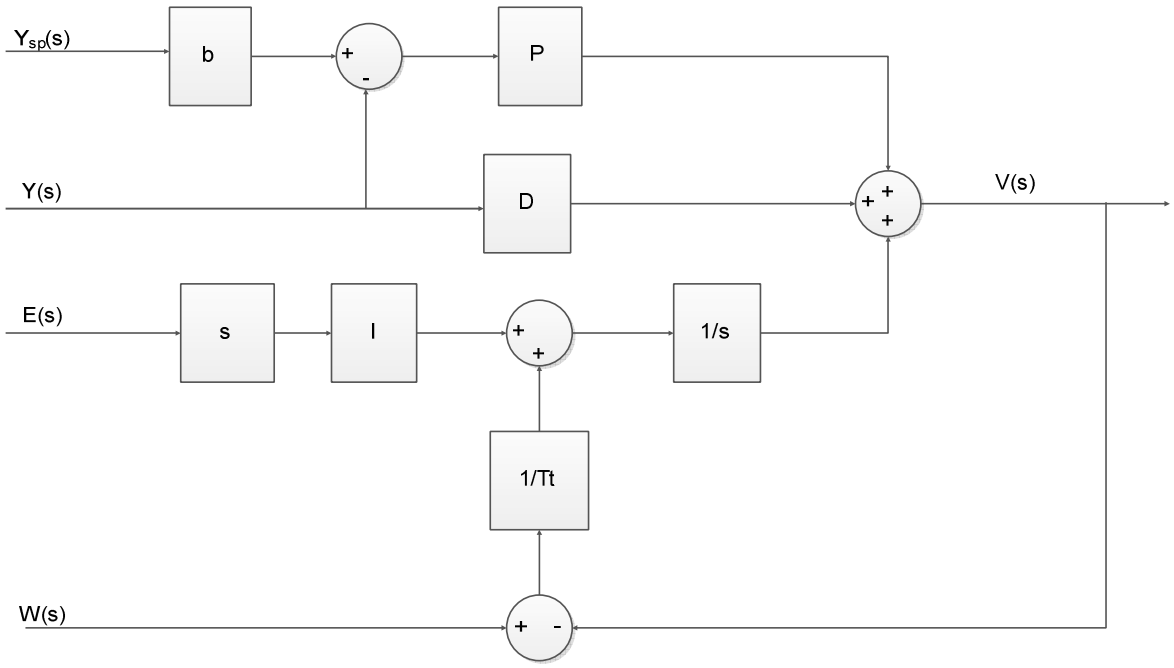



Figura 2.8.3.2.1.1 – Flujograma regulador con bloque Tracking Mode

Este método presenta una gran serie de ventajas cuando se construyen sistemas complejos con numerosos selectores y controles en cascada.

### 2.8.3.2.2 Implementación

La función de Matlab empleada para este regulador con bloque anti Windup incluído es la función *Regulador_3senhales* (escrito de esta forma debido a que la versión de Matlab empleada no permite el uso de acentos y "Ñ").

Durante la implementación de este método se han detectado una serie de problemas. No se dispone de una señal externa que controle el funcionamiento del controlador, por lo que el usuario ha de ser el encargado de modificar el modo de funcionamiento del controlador. Este inconveniente supuso una remodelación de los

sistemas de control de la planta, que se vio modificada como se puede apreciar en el apartado 2.7.2, añadiendo una señal digital para facilitar el control de la planta.

```
dio=digitalio('nidaq','Dev1');
addline(dio,0:3,0,'in');
```

Esta nueva situación, a pesar de ser un gran contratiempo, permite al usuario la implementación de otros métodos de control y uso. Al trabajar con un horno didáctico, este método le permite al usuario conocer el comportamiento de los distintos tipos de reguladores. Por lo que, a pesar de no poder ser totalmente implementado, el resultado obtenido tiene un gran valor didáctico.

En este caso es necesario realizar el cálculo de las distintas funciones de transferencia para los 3 reguladores empleados mediante la función *PIDSetup*. Sería necesario también un cambio de las componentes de cada regulador para mejorar su funcionamiento, siendo la forma más recomendada la empírica.

```
[consigna,Per,tfPID,tfPI,tfPD]=PIDsetup();
```

Dentro de la función se aprecia cómo se tratan las distintas funciones de transferencias de forma independiente para no afectar a las demás. Primero se establecen las funciones de transferencia de las distintas partes de un regulador PID, como se puede ver en el siguiente cuadro de texto . A continuación se establecen las distintas funciones de transferencia de cada tipo de regulador y a continuación se procede a su discretización. En esta función también se determinarán los valores del período y de la consigna.

```
pp =tf(108,1);
pi =tf(1,[12 0]);
pd =tf([2.1 0],1);
g1=pp*(1+pi+pd); %PID
g2=pp*(1+pi);   %PI
g3=pp*(1+pd);   %PD
tfPID = c2d(g1,Per,'tustin');
tfPI = c2d(g2,Per,'tustin');
tfPD = c2d(g3,Per,'tustin');
```

Como se ha indicado anteriormente, me he valido de la señal de 5V de la TAD para alimentar un canal de entrada/salida digital de la tarjeta. Únicamente 4 pines han sido necesitados, 2 de ellos como selectores de modo de funcionamiento del controlador:

| Valor de selector | Tipo de regulador |
|---|---|
| 11 | PID |
| 10 | PD |
| 01 | PI |

Tabla 2.8.3.2.2.1 – Configuración regulador

```
valordigitalentrada=getvalue(DIO);
if (valordigitalentrada(1,1)==1 && valordigitalentrada(1,2)==1)
        num=tfPID.num{1};
        den=tfPID.den{1};
        modofuncionamiento(1,j)=3;
                        …
elseif (valordigitalentrada(1,1)==1 && valordigitalentrada(1,2)==0)
        num=tfPI.num{1};
        den=tfPI.den{1};
        modofuncionamiento(1,j)=1;
                        …
elseif (valordigitalentrada(1,1)==0 && valordigitalentrada(1,2)==1)
        num=tfPD.num{1};
        den=tfPD.den{1};
        modofuncionamiento(1,j)=2;
                …
end
```

Como se puede apreciar, no se considera como opción el empleo de un regulador únicamente proporcional.

La matriz *modofuncionamiento* le proporcionará al usuario la información sobre que regulador está siendo empleado en cada momento para comprobar cómo afecta al estado de la planta.
También ha sido empleada una señal que controla el funcionamiento y salida del controlador. Esta señal ha de permanecer siempre activa, ya que en caso de detectarse un valor de "0", el programa se cerrará.

La última señal empleada será la encargada de entrada en modo configuración, por lo que ha de permanecer de forma habitual desactivada. En el cuadro de texto a continuación se muestran las líneas de comando donde las señales de control tienen relevancia.

```
while funcionamientocorrecto==1

        valordigitalentrada=getvalue(DIO);
        funcionamientocorrecto=valordigitalentrada(1,4);
        llamadaconfigurar=valordigitalentrada(1,3);
        if llamadaconfigurar==1
                [consigna,tfPID,tfPI,tfPD]=llamar_PIDSetup(Per);
        end
end
putsample(AO,0);
stop(AI);
stop(AO);
delete(AI);
delete(AO);
clear AI
clear AO
disp('Salida');
```

Aquí se puede apreciar la asignación de bits de la señal digital a variables empleadas en este regulador. La señal *funcionacorrecto* controla la salida del bucle principal y ha de estar siempre a 1 para que el controlador funcione. En el caso de la variable

*llamadaconfigurar*, ha de estar siempre a 0, simplemente con que en un ciclo se active se realiza la llamada a la función de configuración *llamar_PIDSetup*.

#### 2.8.3.2.3 Resultados

A pesar de no poder replicar el modelo, si se pueden realizar pruebas del comportamiento del sistema con distintas configuraciones. Para ello también se enviará información del modo de funcionamiento del regulador.
Se han realizado numerosas pruebas para poder mostrar los distintos modos de funcionamiento de un regulador. En las siguientes imágenes se les mostrarán por cada medición 3 gráficas, la primera es el estado en Voltios de la planta (0-10V, 0º-100ºC), el modo de funcionamiento del regulador empleado (3:PID, 2: PD, 1:PI) y la tensión que se le aplica al actuador.

En la primera medición se implementó un PID para conocer su comportamiento y cuando estuviera estabilizado, se comprobaría el comportamiento de los otros modos, primero el PI y luego el PD.

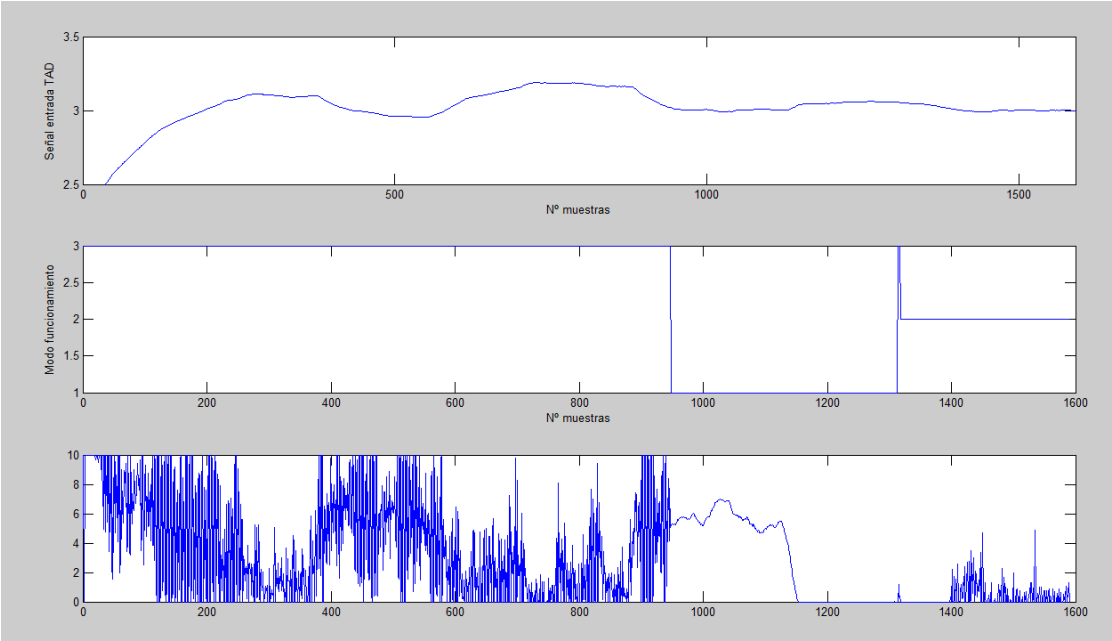



Figura 2.8.3.2.3.1 – Prueba regulador con inicio PID

Tiempo de respuesta: 13.6s.
Tiempo de establecimiento: 23.2s.
Tiempo de pico: 76.4s.

Siguiendo el comportamiento del regulador PID, se presenta una sobreoscilación. Cuando se configura el regulador como un sistema de 1º orden (PI, PD), esta sobreoscilación desaparece. Otro detalle es la variación de la señal aplicada al actuador cuando se emplea un PID y un PD. Esto es debido a la acción derivativa y al ser tan evidente, puede ser una causa de sobreoscilamiento.

También se observa un mejor comportamiento de los sistemas de 1º orden ante perturbaciones. A continuación se comprobará el comportamiento de un regulador PI:

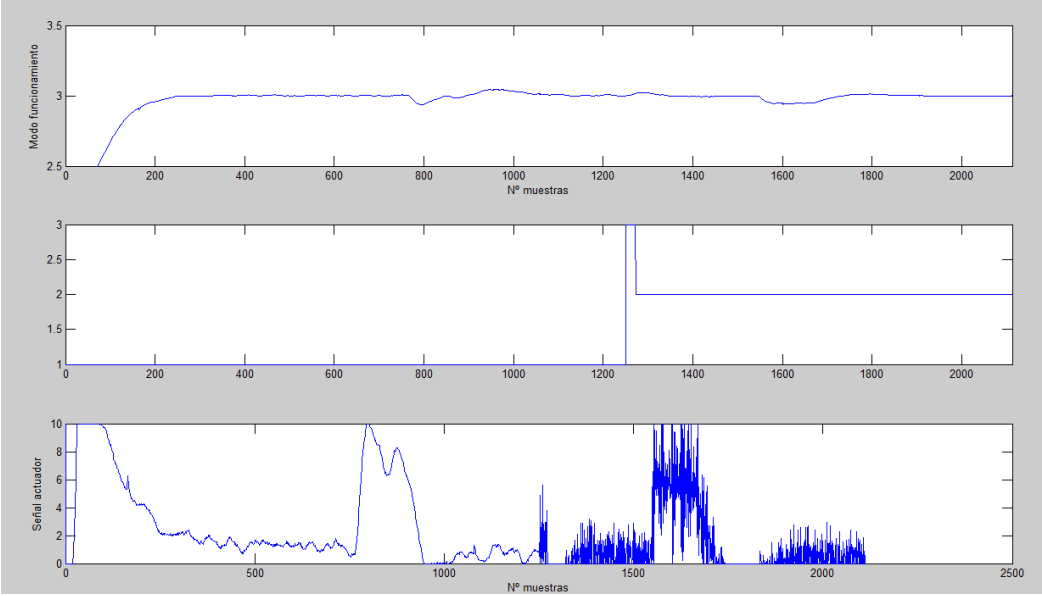



Figura 2.8.3.2.3.2 – Prueba regulador con inicio PI

Tiempo de respuesta: 19.2s.

Tiempo de establecimiento: 28s.

Tiempo de pico: 46s.

El tiempo de respuesta y establecimiento de un regulador PIes mucho mayor que en un PID, aunque al no haber sobreoscilación, su tiempo de pico es menor. En esta prueba se ha vuelto a comprobar la respuesta ante perturbaciones, siendo bastante buena tanto en un PI como en un PD.

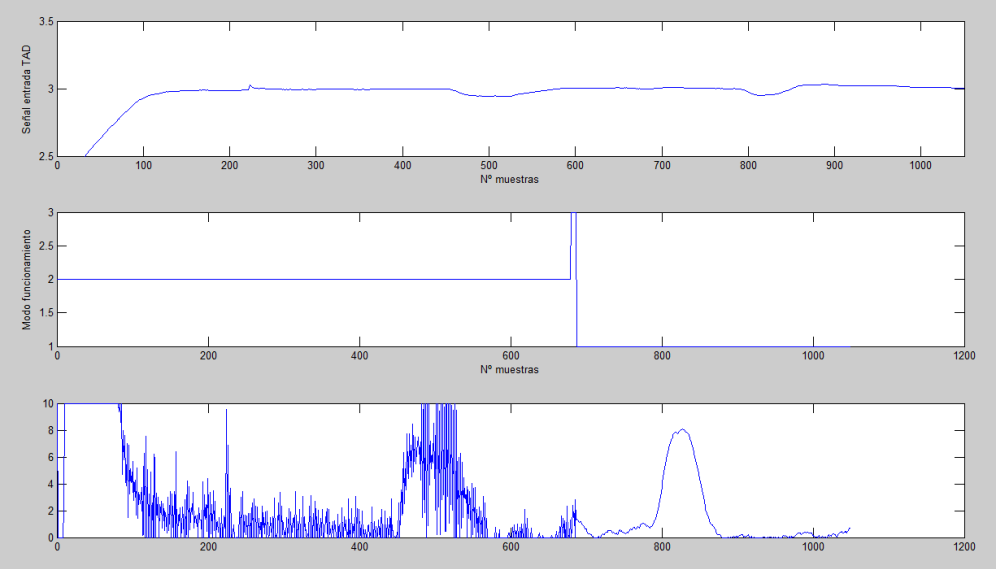



Figura 2.8.3.2.3.3 – Prueba regulador con inicio PD

Tiempo de respuesta: 14.6s.
Tiempo de establecimiento: 17.8s.
Tiempo de pico: 45.2s.

El tiempo de respuesta es similar al obtenido en un PID, aunque su tiempo de establecimiento sí que es bastante menor, al igual que su tiempo de pico (debido a la falta de sobreoscilación).

Dados los resultados obtenidos, he procedido a modificar la parte derivativa del regulador, siendo ahora Td=0.5, para comprobar cómo afecta al sistema y si se puede considerar una mejora del mismo.

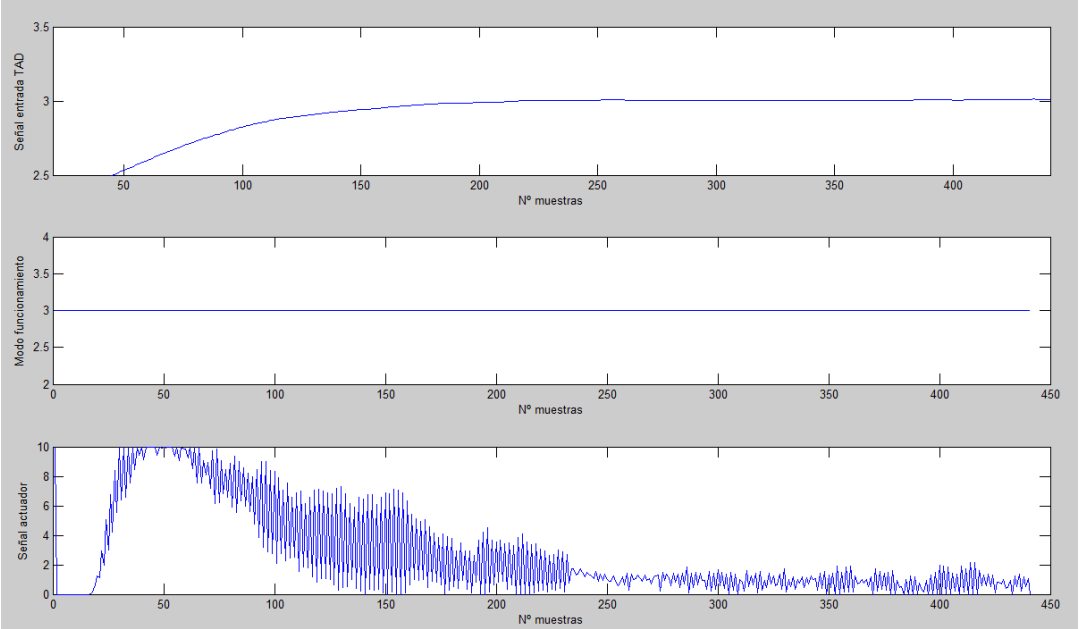



Figura 2.8.3.2.3.1 – Prueba nuevo regulador PID

Tiempo de respuesta: 15.6s.
Tiempo de establecimiento: 21s.
Tiempo de pico: 46.8s.

Se elimina la sobreoscilación, el tiempo de establecimiento y el tiempo de pico, aunque sus valores siguen siendo peores que los obtenidos en un PD. Sin embargo, hay que tener en cuenta que en relación al PD, se ha estabilizado la señal enviada al actuador, que resulta fundamental para alargar la vida de las bombillas, ya que a más oscilación, antes se estropearán.

### 2.8.3.2.4 Conclusiones

A pesar de no haber sido posible la implementación de un regulador con bloque anti Windup “Tracking Mode”, el regulador con selector de modo de funcionamiento ha resultado tener un gran valor didáctico y práctico. Puesto que el conocimiento en del funcionamiento de los distintos parámetros mostrados permite la muestra del comportamiento de los distintos reguladores que factores se ven implicados en ello.

### 2.8.3.3 Integración condicional

#### 2.8.3.3.1 Análisis

Esta alternativa a los otros modelos anteriormente vistos implica la no integración cuando el estado del control está lejos del estado estable; la acción integral sólo se utiliza cuando se cumplen ciertas condiciones; en caso contrario el término integral se mantiene constante.

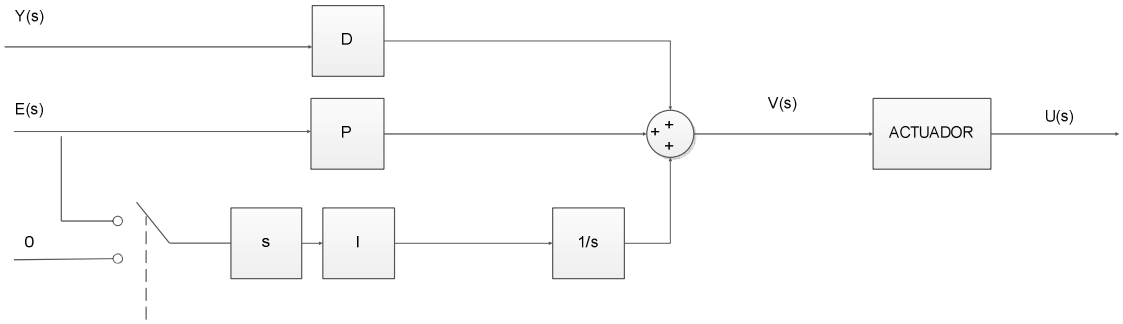



Figura 2.8.3.3.1.1 – Regulador con bloque anti Windup de integración condicionada

Se pueden establecer diversas condiciones que provocan la desconexión del integrador, he aquí algunos ejemplos.

- Desconexión cuando el error de control es elevado.
- Desconexión durante el período de saturación.
- Desconexión cuando el controlador se satura y la actualización del regulador se produce de forma que causa una señal de control aún más saturada.

#### 2.8.3.3.2 Implementación

Comprobados los altos niveles de saturación que se apreciaban en la planta (incluso valores del orden de miles), el método de desconexión para un error de control elevado ha sido considerado el más apto porque la eliminación de la parte integral no afecta en gran medida al comportamiento de la planta.

La función aquí empleada es *Regulador_integracioncondicionada*. La función tiene muchas similitudes con *Regulador_3senhales*, compartiendo incluso funciones

aunque no necesite ciertas variables que calculan. Como se puede apreciar en el siguiente caso, aunque se calcule *tfPI*, no va a ser empelada, puesto que los únicos reguladores aquí empleados serán un PID y un PD.

```
[AI AO DIO]=configuracionrapida();
[consigna,Per,tfPID,tfPI,tfPD]=PIDsetup();
```

La única característica destacable de este modelo es el cálculo de la salida, que depende del valor de la señal de error, guardada debidamente en la matriz *memoriaerror*. En caso de ser mayor a 3 grados se mantiene como un PD, y en caso de ser inferior funciona como un PID. Como la señal está tratada para adaptarse a las características de la TAD, el valor máximo de error será de 0.3ºV.

```
if abs(memoriaerror(1,1))<0.3
        num=tfPID.num{1};
        den=tfPID.den{1};
        salida=num(1)*memoriaerror(1,1);
        salida=salida+(num(2)*memoriaerror(1,2));
        salida=salida+(num(3)*memoriaerror(1,3));
        salida=salida-(den(2)*memoriasalida(1,2));
        salida=salida-(den(3)*memoriasalida(1,3));
        salida=salida/den(1);
else
        num=tfPD.num{1};
        den=tfPD.den{1};
        salida=num(1)*memoriaerror(1,1);
        salida=salida+(num(2)*memoriaerror(1,2));
        salida=salida-(den(2)*memoriasalida(1,2));
        salida=salida/den(1);
        memoriasalida(1,1)=salida;
end
```

#### 2.8.3.3.3 Resultados

El funcionamiento de este regulador con bloque anti Windup ha resultado totalmente satisfactorio. Como se puede comprobar en la imagen 2.8.3.3.2.1.

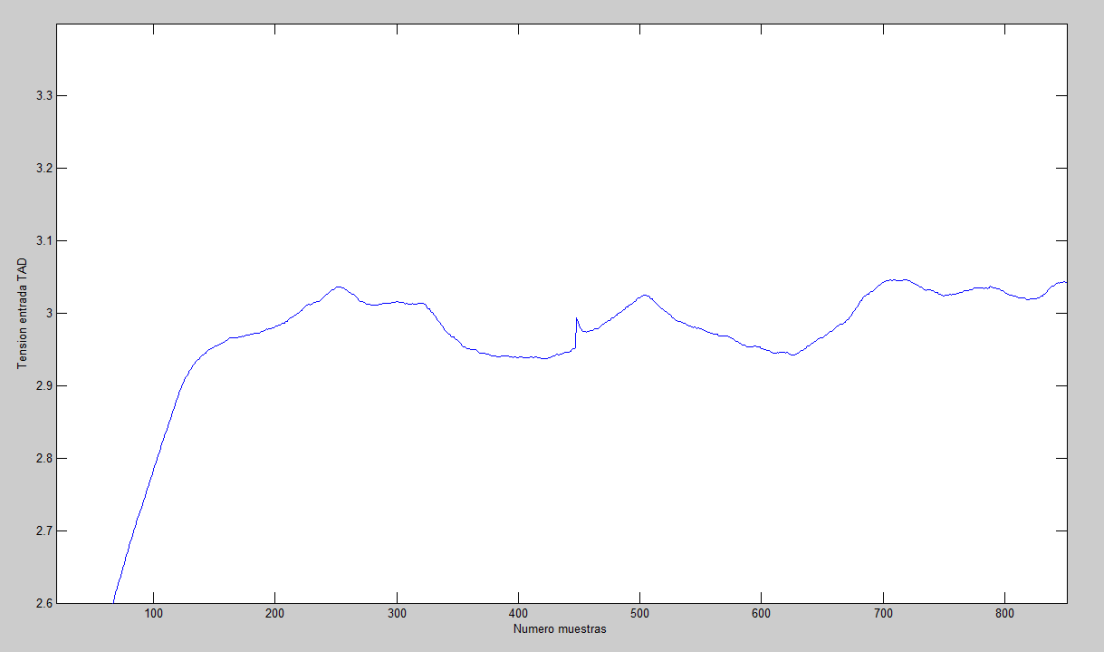



Figura 2.8.3.3.2.1 – Regulador PID integración condicionada

Tiempo de respuesta: 16.8s.

Tiempo de establecimiento: 22.8s.

Tiempo de pico: 50.6s.

Los tiempos son similares a los de un regulador PD, debido a que la mayor parte del tiempo está funcionando de tal modo. Las diferencias de tiempo se deben, aparte de lo ya mencionado, a variaciones de temperatura en el laboratorio, corrientes de aire que afectan al sistema debido a fallos de aislamiento… Son muchos los factores que pueden afectar al sistema, ya que es muy sensible a cualquier perturbación externa.

#### 2.8.3.3.4 Conclusiones

La integración condicionada se ha mostrado como un método anti Windup efectivo y de fácil implementación. Rápido, sin mucha oscilación del estado de la planta y sin error considerable. El único defecto que se podría apreciar es que provoca oscilación en la señal al actuador, lo que disminuye la esperanza de vida de las bombillas, aunque no es tan evidente como en el caso de un regulador PID.

### 2.8.4 Identificación planta

#### 2.8.4.1 Descripción general

En este subapartado se pretenderá explicar de forma adecuada la obtención de los parámetros de la planta del sistema mediante el método de los mínimos cuadrados recursivos o RLS. La utilidad de este método radica en que al conocer la función de transferencia de la planta del sistema, se puede diseñar el adecuado regulador para ella.
Este método es una identificación paramétrica, es decir se parte de que la estructura de las ecuaciones es conocida y son sus coeficientes lo que se desea determinar.
Se estudiará primero el algoritmo de mínimos cuadrados para obtener unos coeficientes constantes con las N primeras medidas. En la práctica, como el tiempo de cómputo es elevado, la identificación suele realizarse off-line.

A continuación el algoritmo de mínimos cuadrados recursivos irá modificando estos coeficientes secuencialmente, con cada nueva medida posterior a N y se obtendrán los valores en régimen permanente de estos coeficientes. Este proceso puede realizarse on-line, por lo que los coeficientes serán dinámicos, adaptándose en cada instante para que el error entre el modelo y la planta sea mínimo.

En la imagen 2.8.4.1.1 se muestra el proceso de identificación de la planta del sistema. Se pueden apreciar tanto la planta G(z) y el modelo $G_m(z)$, su diferencia I(z) será la señal a minimizar. Cuando el error sea 0, se considerará la planta como identificada y su función de transferencia será la misma que la del modelo.

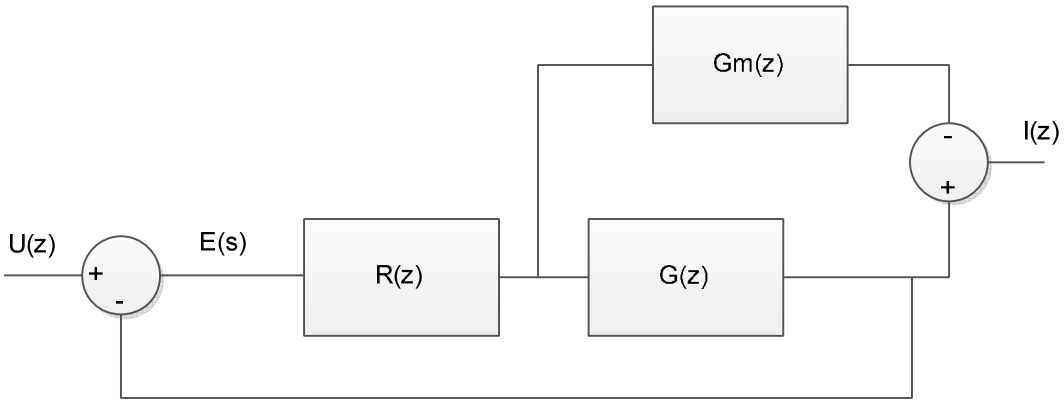



Figura 2.8.4.1.1 - Esquema identificación planta mediante RLS

### 2.8.4.2 Método mínimos cuadrados

En un modelo matemático con una planta de r salidas y n entrada se han realizado n pruebas. De las variables de salida y se sabe que sigue un comportamiento lineal respecto a otras variables xi algunas de las cuales pueden ser valores de y en instantes precedentes o entradas. En cualquier caso al vector x se le considerará como las entradas al sistema puesto que todos sus valores serán conocidos. Pudiendo establecer las siguientes igualdades:

$$\widehat{y_1} = \widehat{a_{11}}x_1 + \widehat{a_{12}}x_2 + \cdots + \widehat{a_{1n}}x_n$$
$$\widehat{y_2} = \widehat{a_{21}}x_1 + \widehat{a_{22}}x_2 + \cdots + \widehat{a_{2n}}x_n$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$\widehat{y_r} = \widehat{a_{r1}}x_1 + \widehat{a_{r2}}x_2 + \cdots + \widehat{a_{rn}}x_n$$

(2.8.4.2.1)

En el caso de que se introduzcan k conjuntos de n entradas, siendo k>>n para conseguir una estimación aceptable de los coeficientes a identificar, las ecuaciones del modelo correspondientes a las medidas realizadas serían:

$$\widehat{y_1}(0) = \widehat{a_{11}}x_1(0) + \widehat{a_{12}}x_2(0) + \cdots + \widehat{a_{1n}}x_n(0)$$
$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$
$$\widehat{y_r}(0) = \widehat{a_{r1}}x_1(0) + \widehat{a_{r2}}x_2(0) + \cdots + \widehat{a_{rn}}x_n(0)$$
$$\widehat{y_1}(1) = \widehat{a_{11}}x_1(1) + \widehat{a_{12}}x_2(1) + \cdots + \widehat{a_{1n}}x_n(1)$$
$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$
$$\widehat{y_r}(1) = \widehat{a_{r1}}x_1(1) + \widehat{a_{r2}}x_2(1) + \cdots + \widehat{a_{rn}}x_n(1)$$
$$\widehat{y_1}(k) = \widehat{a_{11}}x_1(k) + \widehat{a_{12}}x_2(k) + \cdots + \widehat{a_{1n}}x_n(k)$$
$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$
$$\widehat{y_r}(k) = \widehat{a_{r1}}x_1(k) + \widehat{a_{r2}}x_2(k) + \cdots + \widehat{a_{rn}}x_n(k)$$

(2.8.4.2.2)

Podría escribirse de en forma matricial de la siguiente manera:

$$\hat{\mathbf{y}}(\mathrm{k}) = \mathbf{M}\hat{\mathbf{a}}$$

(2.8.4.2.3)

El objetivo de mínimos cuadrados es minimizar la suma del error cuadrático cometido en las k medidas.

$$\underset{\text{krx1}}{\begin{bmatrix} \hat{y}_1(0) \\ \cdots\cdots \\ \hat{y}_r(0) \\ \hat{y}_1(1) \\ \cdots\cdots \\ \hat{y}_r(1) \\ \cdots\cdots \\ \hat{y}_1(k) \\ \cdots\cdots \\ \hat{y}_r(k) \end{bmatrix}} = \underset{\text{krxnr}}{\begin{bmatrix} \mathbf{x}^T(0) & \mathbf{0} & \cdots & \mathbf{0} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{x}^T(0) \\ \mathbf{x}^T(1) & \mathbf{0} & \cdots & \mathbf{0} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{0} & \mathbf{0} & & \mathbf{x}^T(1) \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{x}^T(k) & \mathbf{0} & \cdots & \mathbf{0} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{x}^T(k) \end{bmatrix}} \underset{\text{nrx1}}{\begin{bmatrix} \hat{a}_{11} \\ \vdots \\ \hat{a}_{1n} \\ \hat{a}_{21} \\ \vdots \\ \hat{a}_{2n} \\ \vdots \\ \hat{a}_{r1} \\ \\ \hat{a}_{rn} \end{bmatrix}}$$

(2.8.4.2.4)

Para ello se define el error ***e***(k) como la diferencia entre el valor medido en la salida de la planta real ***y***(k), es decir:

$$\vec{\mathbf{e}}(\mathrm{k}) = \begin{bmatrix} \mathbf{e}(0) \\ \mathbf{e}(1) \\ \cdots \\ \mathbf{e}(\mathrm{k}) \end{bmatrix} = \begin{bmatrix} \mathrm{e}_1(0) \\ \cdots\cdots\cdots \\ \mathrm{e}_r(0) \\ \mathrm{e}_1(1) \\ \cdots\cdots\cdots \\ \mathrm{e}_r(1) \\ \cdots\cdots\cdots \\ \mathrm{e}_1(\mathrm{k}) \\ \cdots\cdots\cdots \\ \mathrm{e}_r(\mathrm{k}) \end{bmatrix} = \begin{bmatrix} \mathrm{y}_1(0) - \hat{\mathrm{y}}_1(0) \\ \cdots\cdots\cdots \\ \mathrm{y}_r(0) - \hat{\mathrm{y}}_r(0) \\ \mathrm{y}_1(1) - \hat{\mathrm{y}}_1(1) \\ \cdots\cdots\cdots \\ \mathrm{y}_r(1) - \hat{\mathrm{y}}_r(1) \\ \cdots\cdots\cdots \\ \mathrm{y}_1(\mathrm{k}) - \hat{\mathrm{y}}_1(\mathrm{k}) \\ \cdots\cdots\cdots \\ \mathrm{y}_r(\mathrm{k}) - \hat{\mathrm{y}}_r(\mathrm{k}) \end{bmatrix}$$

(2.8.4.2.5)

Se trata de minimizar el índice de comportamiento:

$$\mathrm{J} = \frac{1}{2}\vec{\mathbf{e}}^{\mathrm{T}}(\mathrm{k})\vec{\mathbf{e}}(\mathrm{k}) = \frac{1}{2}\begin{bmatrix}\mathbf{e}(0) & \cdots & \mathbf{e}(\mathrm{k})\end{bmatrix}\begin{bmatrix}\mathbf{e}(0) \\ \cdots \\ \mathbf{e}(\mathrm{k})\end{bmatrix}$$

(2.8.4.2.6)

Para obtener así el vector a$ óptimo cuyos componentes son los coeficientes buscados.

$$\vec{\mathbf{e}}(\mathrm{k}) = \vec{\mathbf{y}}(\mathrm{k}) - \hat{\vec{\mathbf{y}}}(\mathrm{k}) - \mathbf{M}\hat{\mathbf{a}}$$

(2.8.4.2.7)

El mínimo cumplirá:

$$\frac{dJ}{d\hat{\mathbf{a}}} = \frac{\partial J}{\partial \vec{\mathbf{e}}(k)} \frac{\partial \vec{\mathbf{e}}(k)}{\partial \hat{\mathbf{a}}} = -\vec{\mathbf{e}}(k)\mathbf{M}^T = \left[\mathbf{M}\hat{\mathbf{a}} - \bar{\mathbf{y}}(\mathbf{k})\right]\mathbf{M}^T = 0$$

(2.8.4.2.8)

Por lo que:

$$\hat{\vec{\mathbf{y}}}(k) = \mathbf{M}\hat{\mathbf{a}}$$

(2.8.4.2.9)

Y como la matriz **M** no será cuadrada, en general, de la expresión anterior no podría despejarse **a**. Teniendo en cuenta que $\mathbf{M^TM}$ si es cuadrada y admitirá inversa, puede ponerse:

$$\mathbf{M^T M}\hat{\mathbf{a}} = \mathbf{M^T}\vec{\mathbf{y}}(\mathbf{k})$$
$$\hat{\mathbf{a}} = (\mathbf{M^T M})^{-1}\mathbf{M^T}\vec{\mathbf{y}}(\mathbf{k})$$

(2.8.4.2.10)

También se podría haber deducido expresando el índice de comportamiento de la forma:

$$J = \frac{1}{2}\vec{\mathbf{e}}^T(k)\vec{\mathbf{e}}(k) = \frac{1}{2}\left[\bar{\mathbf{y}}(k) - \mathbf{M}\hat{\mathbf{a}}\right]^T\left[\bar{\mathbf{y}}(k) - \mathbf{M}\hat{\mathbf{a}}\right] = \frac{1}{2}\left[-\hat{\mathbf{a}}^T\mathbf{M}^T + \bar{\mathbf{y}}^T(k)\right]\left[\bar{\mathbf{y}}(k) - \mathbf{M}\hat{\mathbf{a}}\right] =$$
$$= \frac{1}{2}\left[-\hat{\mathbf{a}}^T\mathbf{M}^T\bar{\mathbf{y}}(k) + \bar{\mathbf{y}}^T(k)\bar{\mathbf{y}}(k) + \hat{\mathbf{a}}^T\mathbf{M}^T\mathbf{M}\hat{\mathbf{a}} - \bar{\mathbf{y}}^T(k)\mathbf{M}\hat{\mathbf{a}}\right]$$

(2.8.4.2.11)

Y teniendo en cuenta que la derivada de una matriz **A** respecto a un vector x cumple:

$$\frac{d(\mathbf{Ax})}{d\mathbf{x}} = \mathbf{A^T}$$

$$\frac{\mathrm{d}(\mathbf{x^T A x})}{\mathrm{d}\mathbf{x}} = \mathbf{A^T x} + \mathbf{A x}$$

$$\frac{\mathrm{d}(\mathbf{x^T A})}{\mathrm{d}\mathbf{x}} = \mathbf{A}$$

(2.8.4.2.12)

Derivando J con respecto a **a** e igualando a cero para minimizar el índice de comportamiento cuadrático, y despejando:

$$\hat{\mathbf{a}} = [\mathbf{M^T M}]^{-1}\mathbf{M^T}\vec{\mathbf{y}}(\mathbf{k})$$

(2.8.4.2.13)

Esta última ecuación proporciona los coeficientes óptimos buscados. Para que exista solución es necesario que la matriz $\mathbf{M^T M}$ no sea singular, para que su inversa exista.

Este procedimiento es poco práctico en tiempo real, al exigir un elevado tiempo de computación, además en el cálculo de matrices inversas pueden cometerse errores importantes dependiendo de la precisión del computador.

Este procedimiento es poco práctico en tiempo real y suele realizarse off-line.

El organigrama sería el siguiente:

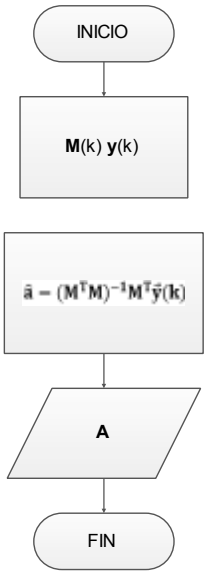



Figura 2.8.4.2.1 – Organigrama método mínimos cuadrados

### 2.8.4.3 Método mínimos cuadrados recursivos

#### 2.8.4.3.1 Explicación matemática

El procedimiento de este método se basa en la actualización de los pesos de los componentes de un regulador para que el error de cálculo entre el modelo real y el teórico sea mínimo. Para ello emplea un proceso recursivo que añade información en cada uno de los nuevos estados.

El proceso necesita unos datos de inicialización, bien pueden estimarse con k medidas o pueden ser inicializados desde un punto inicial no arbitrario (como se indicará más adelante).

El modelo de funcionamiento es el siguiente:

Se trata de que el proceso recursivo proporcione los parámetros del nuevo estados **a**(k+1), partiendo de los valores obtenidos en k y teniendo en cuenta la última medida.

En la observación k+1 la entrada será x(k+1) y la salida de la planta y(k+1) que serán vectores de n y r componentes respectivamente. Si denominamos al valor de los coeficientes en ese instante a$ (k+1), las ecuaciones matriciales serían:

$$\begin{bmatrix} \vec{\mathbf{y}}(\mathrm{k}) \\ \hline \mathbf{y}(\mathrm{k}+1) \end{bmatrix} = \begin{bmatrix} \mathbf{M} \\ \hline \mathbf{x}^{\mathrm{T}}(\mathrm{k}+1) & \cdots & \mathbf{0} \\ \cdots & \cdots & \cdots \\ \mathbf{0} & \cdots & \mathbf{x}^{\mathrm{T}}(\mathrm{k}+1) \end{bmatrix} \hat{\mathbf{a}}(\mathrm{k}+1) + \begin{bmatrix} \vec{\mathbf{e}}(\mathrm{k}) \\ \hline \mathbf{e}(\mathrm{k}+1) \end{bmatrix}$$

(2.8.4.3.1.1)

Como se puede observar, a la notación empleada para el cálculo de mínimos cuadrados, se les han añadido los datos del nuevo estado. Pudiendo establecer estas igualdades:

$$\vec{\mathbf{y}}(\mathrm{k}+1) = \begin{bmatrix} \vec{\mathbf{y}}(\mathrm{k}) \\ \hline \mathbf{y}(\mathrm{k}+1) \end{bmatrix}$$

$$\mathbf{M}(\mathrm{k}+1) = \begin{bmatrix} \mathbf{M} \\ \hline \mathbf{x}^{\mathrm{T}}(\mathrm{k}+1) & \cdots & \mathbf{0} \\ \cdots & \cdots & \cdots \\ \mathbf{0} & \cdots & \mathbf{x}^{\mathrm{T}}(\mathrm{k}+1) \end{bmatrix}$$

$$\vec{\mathbf{e}}(\mathrm{k}+1) = \begin{bmatrix} \vec{\mathbf{e}}(\mathrm{k}) \\ \hline \mathbf{e}(\mathrm{N}+1) \end{bmatrix}$$

(2.8.4.3.1.2)

Pudiendo decir:

$$\vec{\mathbf{y}}(\mathrm{k}+1) = \mathbf{M}(\mathrm{k}+1)\hat{\mathbf{a}}(\mathrm{k}+1) + \vec{\mathbf{e}}(\mathrm{k}+1)$$

(2.8.4.3.1.3)

Y aplicando la estimación óptima obtenida por mínimos cuadrados:

$$\hat{\mathbf{a}}(\mathrm{k}+1) = [\mathbf{M}^{\mathbf{T}}(\mathrm{k}+1)\mathbf{M}(\mathrm{k}+1)]^{-\mathbf{1}}\mathbf{M}^{\mathbf{T}}(\mathrm{k}+1)\vec{\mathbf{y}}(\mathbf{k}+)$$

(2.8.4.3.1.4)

Se puede simplificar esta expresión mediante la relación de Woodbury, que establece:

$$(\mathbf{A}+\mathbf{BD})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{I}+\mathbf{DA}^{-\mathbf{1}}\mathbf{B})^{-1}\mathbf{DA}^{-\mathbf{1}}$$

(2.8.4.3.1.5)

Estableciendo las siguientes relaciones:

$$\mathbf{A} = \mathbf{M}^{\mathrm{T}}\mathbf{M} \qquad \mathbf{B} = \begin{bmatrix} \mathbf{x}(\mathrm{k}+1) & \cdots & \mathbf{0} \\ \cdots & \cdots & \cdots \\ \mathbf{0} & \cdots & \mathbf{x}(\mathrm{k}+1) \end{bmatrix} \qquad \mathbf{D} = \begin{bmatrix} \mathbf{x}^{\mathrm{T}}(\mathrm{k}+1) & \cdots & \mathbf{0} \\ \cdots & \cdots & \cdots \\ \mathbf{0} & \cdots & \mathbf{x}^{\mathrm{T}}(\mathrm{k}+1) \end{bmatrix} = \mathbf{B}^{\mathbf{T}}$$

Llamaremos **P**(k+1) a la siguiente relación:

$$\mathbf{P}(\mathrm{k}+1) = [\mathbf{M}^{\mathrm{T}}(\mathrm{k}+1)\mathbf{M}(\mathrm{k}+1)]^{-1} = (\mathbf{A}+\mathbf{BD})^{-1}$$

(2.8.4.3.1.6)

Aplicando la relación de matrices anteriormente mencionada, resultaría:

$$\begin{aligned} \boldsymbol{P}(\mathrm{k}+1) &= [\mathbf{M}^{\mathrm{T}}\mathbf{M}]^{-1} - [\mathbf{M}^{\mathrm{T}}\mathbf{M}]^{-1}\mathbf{B}(\mathbf{I}+\mathbf{B}^{\mathrm{T}}[\mathbf{M}^{\mathrm{T}}\mathbf{M}]^{-1}\mathbf{B})^{-1}\mathbf{B}^{\mathrm{T}}[\mathbf{M}^{\mathrm{T}}\mathbf{M}]^{-1} \\ &= \mathbf{P}(\mathrm{k}) - \mathbf{P}(\mathrm{k})\mathbf{B}(\mathbf{I}+\mathbf{B}^{\mathrm{T}}\mathbf{P}(\mathrm{k})\mathbf{B})^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{P}(\mathrm{k}) = [\mathbf{I}-\mathbf{W}(\mathrm{k}+1)\mathbf{B}^{\mathrm{T}}]\mathbf{P}(\mathrm{k}) \end{aligned}$$

(2.8.4.3.1.7)

Donde:

$$\mathbf{P}(\mathrm{k}) = [\mathbf{M}^{\mathrm{T}}(\mathrm{k})\mathbf{M}(\mathrm{k})]^{-1}$$

$$\mathbf{W}(\mathrm{k}+1) = \mathbf{P}(\mathrm{k})\mathbf{B}(\mathbf{I} + \mathbf{B}^{\mathrm{T}}\mathbf{P}(\mathrm{k})\mathbf{B})^{-1}$$

Siendo **P**(k) la matriz de covarianza de las estimaciones.

Con estas relaciones se puede establecer la siguiente igualdad:

$$\hat{\mathbf{a}}(\mathrm{k}+1) = [\mathrm{I} - \mathbf{W}(\mathrm{k}+1)\mathbf{B}^{\mathrm{T}}]\mathbf{P}(\mathrm{k})[\mathbf{M}^{\mathrm{T}}(\mathrm{k})\vec{\mathbf{y}}(\mathrm{k}) + \mathbf{B}\mathbf{y}(\mathrm{k}+1)]$$

(2.8.4.3.1.8)

Y teniendo en cuenta el resultado de mínimos cuadrados:

$$\hat{\mathbf{a}}(\mathrm{k}) = [\mathbf{M}^{\mathrm{T}}(\mathrm{k})\mathbf{M}(\mathrm{k})]^{-1}\mathbf{M}^{\mathrm{T}}(\mathrm{k})\vec{\mathbf{y}}(\mathrm{k}) = \mathbf{P}(\mathrm{k})\mathbf{M}^{\mathrm{T}}(\mathbf{k})\vec{\mathbf{y}}(\mathrm{k})$$

(2.8.4.3.1.9)

Operando en 2.8.4.3.1.9 se obtiene:

$$\hat{\mathbf{a}}(\mathrm{k}+1) = \hat{\mathbf{a}}(\mathrm{k}) + \mathbf{P}(\mathrm{k}+1)\mathbf{B}\mathbf{y}(\mathrm{k}+1) - \mathbf{W}(\mathrm{k}+1)\mathbf{B}^{\mathrm{T}}\hat{\mathbf{a}}(\mathrm{k})$$

(2.8.4.3.1.10)

Y como:

$$\begin{aligned}\mathbf{P}(\mathrm{k}+1)\mathbf{B} &= [\mathbf{I} - \mathbf{W}(\mathrm{k}+1)\mathbf{B}^{\mathrm{T}}]\mathbf{P}(\mathrm{k})\mathbf{B} = [\mathbf{I} - (\mathrm{k})\mathbf{B}(\mathbf{I} + \mathbf{B}^{\mathrm{T}}\mathbf{P}(\mathrm{k})\mathbf{B})^{-1}\mathbf{B}^{\mathrm{T}}]\mathbf{P}(\mathrm{k})\mathbf{B} \\ &= \mathbf{P}(\mathrm{k})\mathbf{B}[\mathbf{I} - (\mathbf{I} + \mathbf{B}^{\mathrm{T}}\mathbf{P}(\mathrm{k})\mathbf{B}^{-1})\mathbf{B}^{\mathrm{T}}\mathbf{P}(\mathrm{k})\mathbf{B}] \\ &= \mathbf{P}(\mathrm{k})\mathbf{B}(\mathrm{I} + \mathbf{B}^{\mathrm{T}}\mathbf{P}(\mathrm{k})\mathbf{B})^{-1}[\mathbf{I} + \mathbf{B}^{\mathrm{T}}\mathbf{P}(\mathrm{k})\mathbf{B} - \mathbf{B}^{\mathrm{T}}\mathbf{P}(\mathrm{k})\mathbf{B}] = \mathbf{P}(\mathrm{k})\mathbf{B}(\mathbf{I} + \mathbf{P}(\mathrm{k})\mathbf{B})^{-1} \\ &= \mathbf{W}(\mathrm{k}+1)\end{aligned}$$

(2.8.4.3.1.11)

Resulta:

$$\hat{\mathbf{a}}(\mathrm{k}+1) = \hat{\mathbf{a}}(\mathrm{k}) + \mathbf{W}(\mathrm{k}+1)[\mathbf{y}(\mathrm{k}+1) - \mathbf{B}^{\mathrm{T}}\hat{\mathbf{a}}(\mathrm{k})] = \hat{\mathbf{a}}(\mathrm{k}) + \mathbf{W}(\mathrm{k}+1)\mathbf{e}(\mathrm{k}+1)$$

(2.8.4.3.1.12)

Es decir, con el error de la última medida se mejora la estimación anterior. En caso de no realizar un proceso de mínimos cuadrados, los valores iniciales de $W_o$ y $P_o$ han de ser establecidos por el usuario:

$$\mathbf{P_o} = \frac{1}{\partial}\mathbf{I} \text{ Siendo } \partial \ll 1$$

$$\mathbf{W_0} = \mathbf{0}$$

(2.8.4.3.1.13)

Se puede observar el procedimiento del sistema de la siguiente manera:

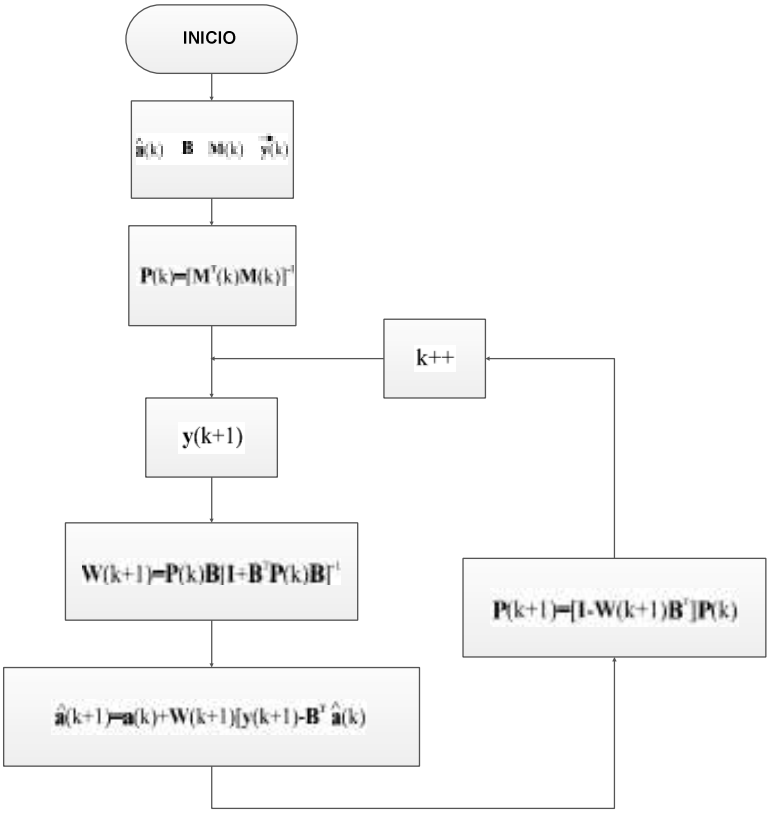



Figura 2.8.4.3.1.1 – Organigrama método mínimos cuadrados recursivos

#### 2.8.4.3.2 Implementación en Matlab

La identificación del sistema se va a implementar con la función *Identificacion* y seguirá el siguiente modelo:

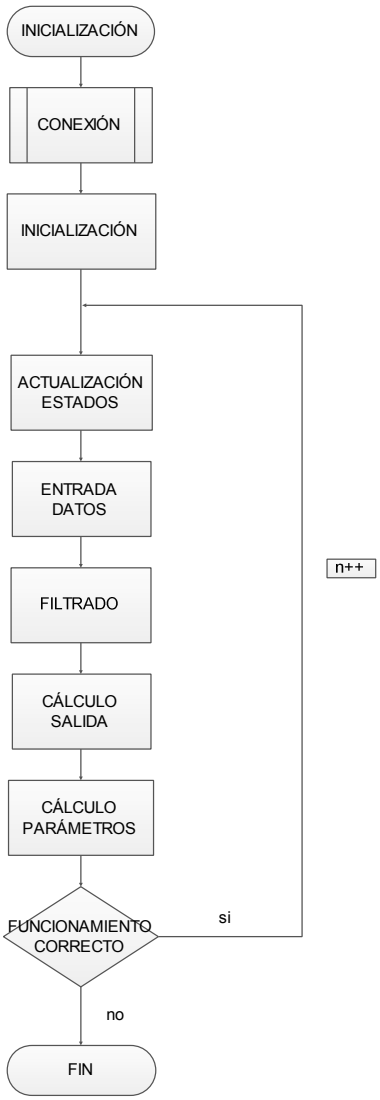



figura 2.8.4.3.2.1 – Flujograma identificación

En cada ciclo se calculará $P_{n+1}$ y $w_{n+1}$ siendo el primero la matriz auxiliar y $w_{n+1}$ la matriz de componentes del regulador. Según el apartado 2.8.4.3.1 los podemos establecer como:

$$P_{n+1} = P_n - \frac{P_n x_{n+1} x_{n+1}^T P_n}{1 + x_{n+1}^T P_n x_{n+1}}$$

$$w_{n+1} = w_n + P_{n+1} x_{n+1} (d_{n+1} - x_{n+1}^T w_n)$$

(2.8.4.3.2.1)

Que serán las fórmulas empleadas en esta función.

Antes de realizar la implementación es necesario establecer una serie de premisas esenciales para el desarrollo del proceso identificador. Estos parámetros dependen de la función de transferencia que se haya ideado para la planta. En este caso se ha establecido la siguiente:

$$G(z) = \frac{B_0}{A_2 z^2 + A_1 z^1 + A_0}$$

(2.8.4.3.2.2)

Por lo tanto, la identificación trabajará en base a esta función de transferencia que requiere que haya una matriz de entrada de datos y una matriz de parámetros del regulador de dimensiones 4x1.

Para el cálculo es necesario desarrollar la función de transferencia según la ecuación de diferencias para así obtener el valor de la señal de salida calculada por el modelo empleado. La ecuación 2.8.4.3.2.3 es la función de transferencia con la señal de entrada y de salida indicadas de forma independiente, siendo *Y(z)* el estado de la planta calculado por el modelo y *U(z)* la tensión aplicada al actuador calculada por el regulador.

$$\frac{Y(z)}{U(z)} = \frac{B_0}{A_2 z^2 + A_1 z^1 + A_0}$$

(2.8.4.3.2.3)

En la ecuación 2.8.4.3.2.4 se muestra la misma ecuación desarrollada para el cálculo de la señal de salida:

$$Y(z) = \frac{B_0 U(z-2) - A_1 Y(z-1) - A_0 Y(z-2)}{A_2}$$

(2.8.4.3.2.4)

Siendo:

Y(z): Estado actual de la planta calculado por el modelo.

U(z-2): Tensión aplicada al actuador en el estado 2 anterior.

Y(z-1): Estado de la planta calculado por el modelo en el estado 1 anterior.

Y(z-2): Estado de la planta calculado por el modelo en el estado 2 anterior.

Por lo tanto, los elementos de cada una de las matrices serán:

| Elemento | Matriz entrada datos | Matriz parámetros |
|---|---|---|
| 1x1 | U(z-2) | $B_0$ |
| 2x1 | Y(z) | $A_2$ |
| 3x1 | Y(z-1) | $A_1$ |
| 4x1 | Y(z-2) | $A_2$ |

Tabla 2.8.4.3.2.1 – Valores matrices RLS

Siendo en la función las variables:

x= Matriz de entrada de datos.

w= Matriz de parámetros del regulador.

Gran parte del código empleado para la realización de la identificación del sistema es código empleado en otras funciones, ya que necesita que la señal sea regulada con un PID.

```
[AI AO DIO]=configuracionrapida();
[consigna,Per,tfPID,tfPI,tfPD]=PIDsetup();
```

Centrándonos en el código específico de la identificación y sus peculiaridades hay que decir que la identificación de la planta permanece durante los primeros 30 segundos (150 ciclos) inactiva. La razón de ello es que así se evitan problemas derivados del empleo de un filtro en la señal de estado de la planta que provoca

valores extraños en las primeras mediciones, aparte de que así se permite que no se realice la identificación mientras el sistema tiene un retraso (la respuesta del sistema se ve retrasada durante los primeros segundos dada su naturaleza y al empleo del citado filtro). Para ello se emplea la variable j, que se incrementa en cada ciclo. En la recopilación de datos, si todavía no se procede a la identificación, se memorizarán los valores reales.

```
if j > 150
          disp(' Funcionando')
          x(2,1)=(w(1,1)*x(1,1));
          x(2,1)=(x(2,1))-(w(3,1)*x(4,1))-(w(4,1)*x(4,1));
          x(2,1)=x(2,1)/1;
else
          x(2,1)=memoriaentradalimpia(1,1);
end
if j > 150
         k=p*x*inv(1+x'*p*x);
         w=w+k*(memoriaentradalimpia(1,1)-x'*w);
         p=(I-k*x')*p;
         w(2,1)=1;
end
```

#### 2.8.4.3.3 Resultados

El programa ha identificado correctamente la planta, siendo la matriz *w* obtenida:

```
ans =
   0.0001135
   1.0000
   -1.586
   0.5856
```

Para saber que la señal ha sido correctamente identificada hay que ver su señal de error. En la figura 2.8.4.3.3.1 se muestra la señal de error de error en % y cómo se va

reduciendo con el tiempo. Es importante que la señal de error sea menor a un 10% para considerar la planta adecuadamente identificada.

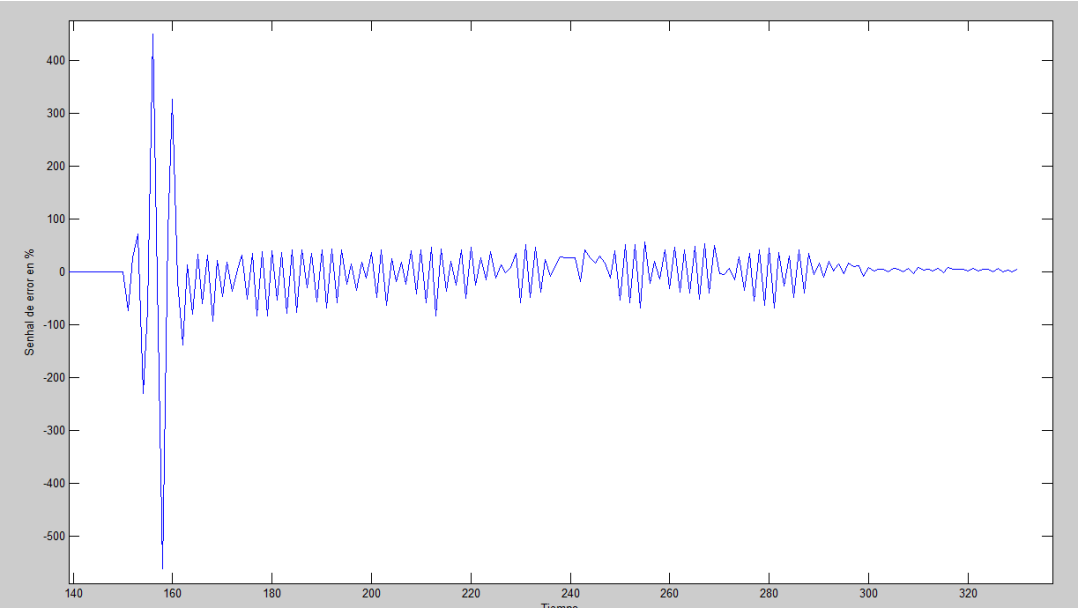



Figura 2.8.4.3.3.1 – Señal de error en %

La señal de error permanece a 0 en las primeras mediciones ya que, como se ha indicado anteriormente, la identificación espera un tiempo antes de inicializarse. Los primeros valores de error son muy altos, aunque al avanzar los ciclos, la señal de error se estabiliza. En la figura 2.8.4.3.3.2 se muestra un detalle de la señal de error en las últimas mediciones.

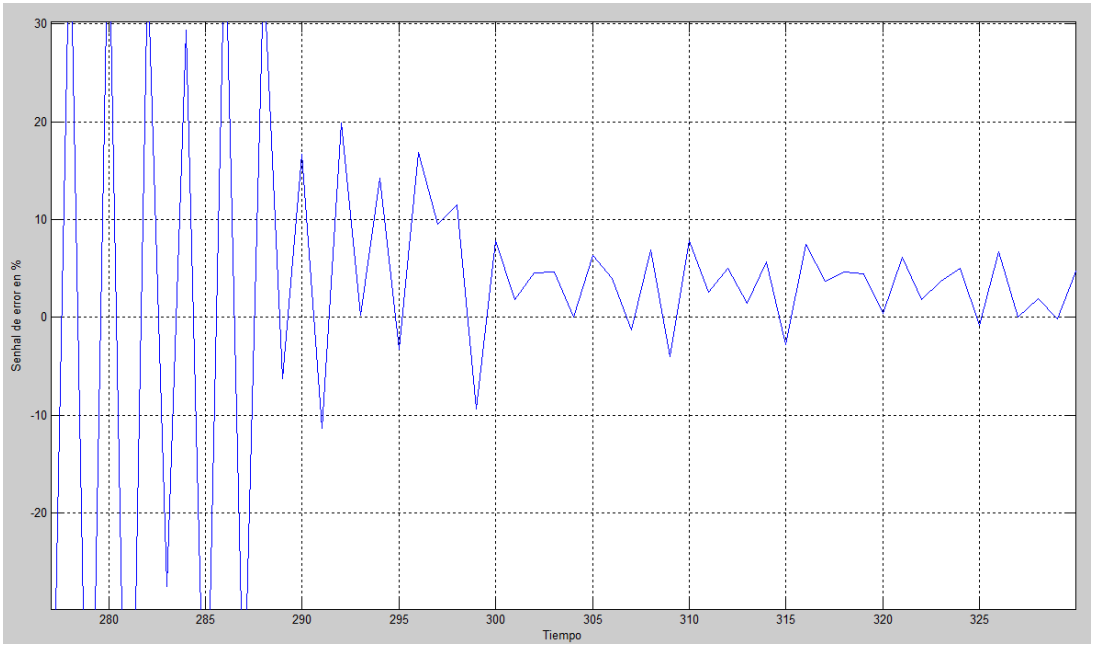



Figura 2.8.4.3.3.2 - Detalle últimas mediciones señal error en %

Con esto se puede establecer la función de transferencia de la planta como:

$$G(z) = \frac{0.0001135}{z^2 - 1.586z^1 + 0.5856} \qquad (2.8.4.3.4.1)$$

### 2.8.4.3.4 Conclusiones

El método RLS es una útil herramienta para la mejora de la regulación de un sistema. Pues conocer la función de transferencia de una planta permite conocer su comportamiento ante todo tipo de estímulos y así poder implementar un regulador que resulte óptimo.

## 2.9 Orden de prioridad de los documentos básicos

A continuación se establece el orden de prioridad de los documentos básicos del Proyecto:

1. ÍNDICE.
2. MEMORIA.
3. ANEXOS.
4. PLIEGO DE CONDICIONES.
5. ESTADO DE MEDICIONES.
6. PRESUPUESTO.

TÍTULO: **MEJORA DEL ALGORITMO DE CONTROL DE UNA PLANTA DE LABORATORIO E IDENTIFICACIÓN DE LA MISMA**

---

# ANEXOS

---

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

**AVDA. 19 DE FEBRERO, S/N**

**15405 - FERROL**

FECHA: **SEPTIEMBRE DE 2013**

AUTOR: EL ALUMNO

Fdo.: GABRIEL CALVO RODRIGUEZ

# ÍNDICE

**3. ANEXOS……67**

**3.1 Contenido. ……...69**

**3.1.1 Documentación de partida.....…….. 69**

**3.1.2 Cálculos……..…….……..… 70**

**3.1.3 Otros anexos…..…….……..…… 70**

**3.1.3.1 Código empleado……..…….…….……..70**

**3.1.3.1.1 Configuracionrapidaviejo……..71**

**3.1.3.1.2 PIDSetupviejo…….……..71**

**3.1.3.1.3 Llamar_PIDSetupviejo……..…… 71**

**3.1.3.1.4 Regulador_backcalculation..……72**

**3.1.3.1.5 PIDactualizar……74**

**3.1.3.1.6 Configuracionrapida..……..74**

**3.1.3.1.7 PIDSetup……74**

**3.1.3.1.8 Llamar_PIDSetup……..75**

**3.1.3.1.9 Regulador_3senhales……75**

**3.1.3.1.10 Regulador_integracioncondicionada...……78**

**3.1.3.1.11 Identificación…..……..80**

# 3. ANEXOS

## 3.1 Contenido

A lo largo de este apartado, se adjuntará toda la información adicional que no ha sido tratada con anterioridad y que es necesaria para el completo desarrollo del presente proyecto.

### 3.1.1 Documentación de partida

UNIVERSIDADE DA CORUÑA

ASIGNACIÓN DE TRABALLO FIN DE GRADO

Curso 2012-2013

En virtude da solicitude de TFG efectuada por:
*(En virtud de la solicitud de TFG efectuada por:)*

**Apelidos e nome:** Calvo Rodríguez, Gabriel
***(Apellidos y nombre)***

**D.N.I.** 32705227D **Data da solicitude:** Febreiro - 2013
***(D.N.I.)*** ***(Fecha de solicitud)***

alumnola desta escola na Titulación de Grao en Enxeñería en Electrónica Industrial e Automática, comunícolle que a Comisión de Proxectos decidiu asignarlle o seguinte Traballo fin de Grao:
*(alumno/a de esta escuela en la titulación de Grado en Ingeniería Eléctrica, le comunico que la Comisión de Proyectos ha decidido asignarle el siguiente Trabajo Fin de Grado:)*

**Título T.F.G:** MEJORA DEL ALGORITMO DE CONTROL DE UNA PLANTA DE LABORATORIO E IDENTIFICACIÓN DE LA MISMA

**Nº T.F.G:** 77G01A021

**Titor: (Tutor:)** Calvo Rollé, José Luis

**suplente** José Luis Casteleiro Roca

A descrición e obxectivos do proxecto son os que figuran no reverso desta documento:
*(La descripción y objetivos del Trabajo son los que figuran en el reverso de este documento):*

Ferrol, a 12 de marzo de 2013

Figura 3.1.1.1 Solicitud de proyecto de fin de grado (Parte posterior)

Figura 3.1.1.2 Solicitud de proyecto de fin de grado (Parte anterior)

### 3.1.2 Cálculos

En este apartado se detallarán todos los cálculos realizados para el diseño del presente proyecto. Los cálculos seguirán una estructuración por apartados de una forma paralela a la memoria.

### 3.1.3 Otros anexos

#### 3.1.3.1 Código empleado

A continuación se muestra el código empleado en este proyecto. Pido disculpas de antemano por los errores ortográficos que puedan aparecer, pero la versión de Matlab empleada en el laboratorio empleaba un teclado británico y no tenía ni acentos ni la letra *Ñ.*

### 3.1.3.1.1 Configuracionrapidaviejo

```
function [ ai ao ] = configuracionrapidaviejo(  )
%==========================================================================%
%Esta funcion se encarga de realizar la conexion entre la TAD y el PC
% y la configuracion de los distintos canales de entrada y salida.
%Se establecen 3 canales analogicos de entrada y uno de salida. Esta
%funcion se empleara unicamente con el regulador "Back Calculation".
%==========================================================================%
ai=analoginput('nidaq','Dev1');
addchannel(ai,0);
addchannel(ai,5);
addchannel(ai,6);
ao=analogoutput('nidaq','Dev1');
set(ai,'inputtype','singleended');% Configura el tipo de entrada
set(ai,'samplespertrigger',900);
set(ai,'samplerate',100);
addchannel(ao,0);
end
```

### 3.1.3.1.2 PIDSetupviejo

```
function [TF,CONSIGNA,K,Per] = PIDsetup ()
%==========================================================================%
%En esta funcion se detallan los valores del regulador PID de posicion
%y permite el inicio de la regulacion. Empleada unicamente para el
%regulador con bloque anti Windup Back Calculation
%==========================================================================%
Per = 0.2;
K =108;
Ti =10;
Td = 2.26;

%Se le preguntara al usuario por el valor de la consigna en grados y se
%adecuara a las unidades de la TAD.
CONSIGNAGRADOS=input('Indique consigna en grados: ');
CONSIGNA=CONSIGNAGRADOS/10;

%Se calcula la funcion de transferencia discretizada
G = K*tf([Ti*Td Ti 1], [Ti 0]);
TF = c2d(G,Per,'tustin');


end
```

### 3.1.3.1.3 llamar_PIDSetuoViejo

```
function [ TF,CONSIGNA,K,Ti,Td,Per ] = llamar_PIDSetupViejo(  )
%==========================================================================%
%Cuando el usuario quiere cambiar los valores de las constantes del PID de
%posicion hace una llamada a esta funcion, donde se le preguntara por los
%distintos valores a introducir.
%Esta funcion se empleara unicamente con el regulador "Back Calculation".
%==========================================================================%
Per = 0.2;
Per=input('Indique período: ');
K=input('Indique constante proporcional: ');
Ti=input('Indique constante integral: ');
Td=input('Indique derivativa:' );
CONSIGNAGRADOS=input('Indique consigna en grados: ');
CONSIGNA=CONSIGNAGRADOS/10;
%Tambien calcula la nueva funcion de transferencia discretizada.
```

```
G = K*tf([Ti*Td Ti 1], [Ti 0]);
TF = c2d(G,Per,'tustin');

end
```

### 3.1.3.1.4 Regulador_backcalculation

```
function [MATRIZSALIDA] = Regulador_backcalculation( )
%=========================================================================%
% Esta funcion emplea el esquema "Back calculation" para el calculo del
% bloque anti Windup. En cada nuevo ciclo del bucle establece una nueva FT
%del regulador dependiente de la senhal de error entre la senhal enviada al
%actuador y la que realmente este puede enviar (Saturacion).
%Huelga indicar que la nomenclautra es distinta a otros reguladores y
%algunas funciones de configuracion son distintas puesto que este regulador
%emplea 3 senhales analogicas de entrada y ninguna digital.
%=========================================================================%

    FUNCIONAMIENTOCORRECTO=1;
    [AI AO]=configuracionrapidaviejo(); %Configurcion canales entrada y salida
    [TF,CONSIGNA,K,Per]=PIDsetupviejo(); %Configuracion del PID

    %Inicializacion de variables
    NUM=TF.num{1};
    SIZENUMTOTAL=size(NUM);
    SIZENUM=SIZENUMTOTAL(2);
    DEN=TF.den{1};
    SIZEDENTOTAL=size(DEN);
    SIZEDEN=SIZEDENTOTAL(2);
    MEMORIAERROR=zeros(1,SIZEDEN);
    MEMORIASALIDA=zeros(1,SIZEDEN);
    SALIDAPLANTA=0;
    SALIDAACTUADOR=0;
    TENSIONESENTRADA=zeros(1,3);
    MEMORIAENTRADASUCIA=zeros(1,2);
    MEMORIAENTRADALIMPIA=zeros(1,2);
    MATRIZSALIDA=zeros(1,1);
    J=0;
    plot(MATRIZSALIDA);
    xlabel('Numero de muestras');
    ylabel('Tension de entrada en la TAD');

   while FUNCIONAMIENTOCORRECTO==1
            tic
            LLAMADAREGULADOR=0;
            J=J+1;

            %Actualizacion de los valores de error y salida
            for i=SIZENUM:-1:2
                MEMORIAERROR(1,i)=MEMORIAERROR(1,i-1);
            end
            MEMORIASALIDA(1,3)=MEMORIASALIDA(1,2);
            MEMORIASALIDA(1,2)=SALIDAPLANTA;

            %Entrada y filtracion de datos
            TENSIONESENTRADA=getsample(AI);
            MEMORIAENTRADASUCIA(1,2)=MEMORIAENTRADASUCIA(1,1);
            MEMORIAENTRADASUCIA(1,1)=TENSIONESENTRADA(1,1);
            MEMORIAENTRADALIMPIA(1,2)=MEMORIAENTRADALIMPIA(1,1);
            MEMORIAENTRADALIMPIA(1,1)=0.1813*MEMORIAENTRADASUCIA(1,2);
```

```
MEMORIAENTRADALIMPIA(1,1)=MEMORIAENTRADALIMPIA(1,1)+(0.8187*MEMORIAENTRADALIMPIA(
1,2));
            MATRIZSALIDA(1,J)=MEMORIAENTRADALIMPIA(1,1);
            MEMORIAERROR(1,1) =CONSIGNA-MEMORIAENTRADALIMPIA(1,1);
            CONTROL=TENSIONESENTRADA(1,2);
            LLAMADACONFIGURAR=TENSIONESENTRADA(1,3);

            %Variables de control
            if CONTROL>5
                FUNCIONAMIENTOCORRECTO=0;
                disp('Salida solicitada por el usuario');
            end

            if LLAMADACONFIGURAR>5
                LLAMADAREGULADOR=1;
                [TF,CONSIGNA,K,Ti,Td,Per]=llamar_PIDSetupViejo();
                NUM=TF.num{1};
                DEN=TF.den{1};

            end

            %Esta es la funcion que se encarga de calcular la nueva TF
            [NUM,SIZENUM,DEN,SIZEDEN] =
PIDactualizar(SALIDAPLANTA,SALIDAACTUADOR,K,Ti,Td);


            %Calculo senhal que se aplica al actuador segun la TF del regulador
            SALIDAPLANTA=NUM(1)*MEMORIAERROR(1,1);
            SALIDAPLANTA=SALIDAPLANTA+(NUM(2)*MEMORIAERROR(1,2));
            SALIDAPLANTA=SALIDAPLANTA+(NUM(3)*MEMORIAERROR(1,3));
            SALIDAPLANTA=SALIDAPLANTA-(DEN(2)*MEMORIASALIDA(1,2));
            SALIDAPLANTA=SALIDAPLANTA-(DEN(3)*MEMORIASALIDA(1,3));
            SALIDAPLANTA=SALIDAPLANTA/DEN(1);
            SALIDAACTUADOR=SALIDAPLANTA;

            %Acondicionado de señal y tiempo. La senhal al controlador va
            %de 0 a 10V.
            if SALIDAACTUADOR < 0
                SALIDAACTUADOR = 0;
            end
            if SALIDAACTUADOR>10
                SALIDAACTUADOR = 10;
            end
            putsample(AO,SALIDAACTUADOR)
            toc
            if (toc<Per)
                pause(Per-toc)
            elseif (LLAMADAREGULADOR==0)
                disp(' Se ha excedido el tiempo del período')
                FUNCIONAMIENTOCORRECTO=0;

            end

        %Muestreo del estado de la planta por pantalla.
        plot(MATRIZSALIDA);
        xlabel('Numero de muestras');
        ylabel('Tension de entrada en la TAD');

   end
    %Se envia al controlador una senhal de 0V para apagar las bombillas y
    %se para el transito de informacion. Tambien se eliminan los objetos de
    %la tarjeta de la memoria.
    putsample(AO,0);
    stop(AI);
```

```
    stop(AO);
    delete(AI);
    delete(AO);
    clear AI
    clear AO
    disp('Salida del programa');
end
```

### 3.1.3.1.5 PIDactualizar

```
function [NUM,SIZENUM,DEN,SIZEDEN] =
PIDactualizar(SALIDAPLANTA,SALIDAACTUADOR,K,Ti,Td)
%=========================================================================%
% El objetivo de esta funcion es el calculo de una nueva FT dependiente de
% la diferencia entre la salida calculada en al planta y la que realmente
% envia el actuador. Para ello se recalcula una nueva parte integral y se
% actualizan los valores de la FT.
%=========================================================================%
ERRORACTUADOR=SALIDAACTUADOR-SALIDAPLANTA;
Per = 0.2;
Ti =Ti+(ERRORACTUADOR/(Per));
G = K*tf([Ti*Td Ti 1], [Ti 0]);
TF = c2d(G,Per,'tustin');
NUM=TF.num{1};
SIZENUMTOTAL=size(NUM);
SIZENUM=SIZENUMTOTAL(2);
DEN=TF.den{1};
SIZEDENTOTAL=size(DEN);
SIZEDEN=SIZEDENTOTAL(2);

end
```

### 3.1.3.1.6 configuracionrapida

```
function [ ai ao dio] = configuracionrapida(  )
%=========================================================================%
%Esta funcion se encarga de realizar la conexion entre la TAD y el PC
% y la configuracion de los distintos canales de entrada y salida.
%Establece una senhal analogica de entrada "ai" y otra de salida "ao".
%Tambien se define un canal digital "dio" de 4 bit en el puerto 0.
%=========================================================================%
ai=analoginput('nidaq','Dev1'); %Canal entrada analogico
addchannel(ai,0);
ao=analogoutput('nidaq','Dev1');%Canal de salida analogico
set(ai,'inputtype','singleended'); %Configuracion tipo de entrada.
set(ai,'samplespertrigger',900);
set(ai,'samplerate',100);
addchannel(ao,0);
dio=digitalio('nidaq','Dev1');%Canal digital de 4 bits.
addline(dio,0:3,0,'in');
end
```

### 3.1.3.1.7 PIDsetup

```
function [consigna,Per,tfPID,tfPI,tfPD] = PIDsetup ()
%=========================================================================%
%En esta funcion se detallan los valores del regulador PID de posicion
%y permite el inicio de la regulacion. Empleada para reguladores con
%bloques anti Windup de integracion condicionada y para el regulador con la
```

```
%senhal de seleccion de modo de funcionamiento. Devuelve de forma
%independiente las funciones de transferencia de un regulador PID, PI y PD.
%=========================================================================%
Per = 0.2;
pp =tf(108,1);
pi =tf(1,[12 0]);
pd =tf([2.1 0],1);

consignagrados=input('Indique consigna en grados: ');
consigna=consignagrados/10;

g1=pp*(1+pi+pd); %PID
g2=pp*(1+pi);    %PI
g3=pp*(1+pd);    %PD

tfPID = c2d(g1,Per,'tustin');
tfPI = c2d(g2,Per,'tustin');
tfPD = c2d(g3,Per,'tustin');

end
```

### 3.1.3.1.8 llamar_PIDSetup

```
function [consigna,tfPID,tfPI,tfPD] = llamar_PIDSetup(Per)
%=========================================================================%
%Cuando el usuario quiere cambiar los valores de las constantes de un
%regulador se hace una llamada a esta funcion, donde se le preguntara por los
%distintos valores a introducir.
%El programa calculara las funciones de transferencias de un PID, PI y PD
%por separado y las devolvera como valores de salida.
%=========================================================================%
K=input('Indique constante proporcional: ');
Ti=input('Indique constante integral: ');
Td=input('Indique derivativa:' );
consignagrados=input('Indique consigna en grados: ');
consigna=consignagrados/10;

pp =tf(K,1);
pi =tf(1,[Ti 0]);
pd =tf([Td 0],1);

g1=pp*(1+pi+pd); %PID
g2=pp*(1+pi);    %PI
g3=pp*(1+pd);    %PD

tfPID = c2d(g1,Per,'tustin');
tfPI = c2d(g2,Per,'tustin');
tfPD = c2d(g3,Per,'tustin');

end
```

### 3.1.3.1.9 Regulador_3senhales

```
function [matrizsalida,modofuncionamiento,matrizpotencia] = Regulador_3senhales(
)
%=========================================================================%
%Implementacion de un regulador con selector de modo de funcionamiento.
%Este metodo tiene un gran poder didactico, puesto que permite al usuario
%conocer los distintos modos de funcionamiento de un regulador y su
```

```
%comportamiento.
%=========================================================================%
    %Configuracion Tarjeta y PID
    funcionamientocorrecto=1;
    [AI AO DIO]=configuracionrapida();
    [consigna,Per,tfPID,tfPI,tfPD]=PIDsetup();

    %Inicializacion de variables. Se guardara en memoria el estado de la
    %planta, la potencia aplicada y el modo de funcionamiento.
    num=tfPID.num{1};
    sizenumtotal=size(num);
    sizenum=sizenumtotal(2);
    den=tfPID.den{1};
    sizedentotal=size(den);
    sizeden=sizedentotal(2);
    memoriaerror=zeros(1,sizeden);
    memoriasalida=zeros(1,sizeden);
    salida=0;
    valordigitalentrada=zeros(1,4);
    memoriaentradasucia=zeros(1,2);
    memoriaentradalimpia=zeros(1,2);
    matrizsalida=zeros(1,1);
    j=0;
    plot(matrizsalida);
    xlabel('Numero de muestras');
    ylabel('Tension de entrada en la TAD');
    modofuncionamiento=zeros(1,1);
    matrizpotencia=zeros(1,1);

   while funcionamientocorrecto==1
             tic
             llamadaconfigurar=0;
             j=j+1;
             %Actualizacion los valores de error y salida.
             for i=sizenum:-1:2
                 memoriaerror(1,i)=memoriaerror(1,i-1);
             end
             memoriasalida(1,3)=memoriasalida(1,2);
             memoriasalida(1,2)=salida;

            % Filtrado senhal entrada
             memoriaentradasucia(1,2)=memoriaentradasucia(1,1);
             memoriaentradasucia(1,1)=getsample(AI);
             memoriaentradalimpia(1,2)=memoriaentradalimpia(1,1);
             memoriaentradalimpia(1,1)=0.1813*memoriaentradasucia(1,2);

memoriaentradalimpia(1,1)=memoriaentradalimpia(1,1)+(0.8187*memoriaentradalimpia(
1,2));
             matrizsalida(1,j)=memoriaentradalimpia(1,1);
             memoriaerror(1,1) =consigna-memoriaentradalimpia(1,1);

             %Entrada senhales digitales y asignacion de variables
             valordigitalentrada=getvalue(DIO);
             funcionamientocorrecto=valordigitalentrada(1,4);
             llamadaconfigurar=valordigitalentrada(1,3);

             %Configuracion nueva TF.
             if llamadaconfigurar==1
                 [consigna,tfPID,tfPI,tfPD]=llamar_PIDSetup(Per);
             end

             %Seleccion de modo de funcionamiento en relacion al estado de
             %los 2 primeros bits de la senhal digital.
             % Calculo senhal salida PID
             if (valordigitalentrada(1,1)==1 && valordigitalentrada(1,2)==1)
                 num=tfPID.num{1};
```

```
                den=tfPID.den{1};
                salida=num(1)*memoriaerror(1,1);
                salida=salida+(num(2)*memoriaerror(1,2));
                salida=salida+(num(3)*memoriaerror(1,3));
                salida=salida-(den(2)*memoriasalida(1,2));
                salida=salida-(den(3)*memoriasalida(1,3));
                salida=salida/den(1);
                modofuncionamiento(1,j)=3;
                %Calculo senhal salida PI
            elseif (valordigitalentrada(1,1)==1 && valordigitalentrada(1,2)==0)
                num=tfPI.num{1};
                den=tfPI.den{1};
                salida=num(1)*memoriaerror(1,1);
                salida=salida+(num(2)*memoriaerror(1,2));
                salida=salida-(den(2)*memoriasalida(1,2));
                salida=salida/den(1);
                modofuncionamiento(1,j)=1;
                %Calculo senhal salida PD
            elseif (valordigitalentrada(1,1)==0 && valordigitalentrada(1,2)==1)
                num=tfPD.num{1};
                den=tfPD.den{1};
                salida=num(1)*memoriaerror(1,1);
                salida=salida+(num(2)*memoriaerror(1,2));
                salida=salida-(den(2)*memoriasalida(1,2));
                salida=salida/den(1);
                modofuncionamiento(1,j)=2;
            end

            %Acondicionado de señal y tiempo.
            if salida < 0
                salida = 0;
            end
            if salida>10
                salida = 10;
            end
            matrizpotencia(1,j)=salida;

            putsample(AO,salida)
            toc;
            if (toc<Per)
                pause(Per-toc)
            elseif (llamadaconfigurar==0)
                disp(' Se ha excedido el tiempo del período')
                funcionamientocorrecto=0;

            end

        plot(matrizsalida);
        xlabel('Numero de muestras');
        ylabel('Tension de entrada en la TAD');
    end

    %Se envia al controlador una senhal de 0V para apagar las bombillas y
    %se para el transito de informacion. Tambien se eliminan los objetos de
    %la tarjeta de la memoria.
    putsample(AO,0);
    stop(AI);
    stop(AO);
    delete(AI);
    delete(AO);
    clear AI
    clear AO
    disp('Salida');
end
```

### 3.1.3.1.10 Regulador_integracioncondicionada

```
function [matrizsalida] = Regulador_integracioncondicionada( )
%=========================================================================%
%Implementacion de un regulador con un bloque anti Windup de integracion
%condicionada. Mientras el error de la planta sea superior a un determinado
%umbral, el regulador actuara como un PD. Cuando sea lo suficientemente
%pequenho, se comportara como un PID.
%=========================================================================%

    %Configuracion Tarjeta y PID
    funcionamientocorrecto=1;
    [AI AO DIO]=configuracionrapida();
    [consigna,Per,tfPID,tfPI,tfPD]=PIDsetup();

    %Inicializacion de variables
    num=tfPID.num{1};
    sizenumtotal=size(num);
    sizenum=sizenumtotal(2);
    den=tfPID.den{1};
    sizedentotal=size(den);
    sizeden=sizedentotal(2);
    memoriaerror=zeros(1,sizeden);
    memoriasalida=zeros(1,sizeden);
    salida=0;
    valordigitalentrada=zeros(1,4);
    memoriaentradasucia=zeros(1,2);
    memoriaentradalimpia=zeros(1,2);
    matrizsalida=zeros(1,1);
    j=0;
    plot(matrizsalida);
    xlabel('Numero de muestras');
    ylabel('Tension de entrada en la TAD');

   while funcionamientocorrecto==1
            tic
            llamadaconfigurar=0;
            j=j+1;
            %Actualizacion los valores de error y salida.
            for i=sizenum:-1:2
                memoriaerror(1,i)=memoriaerror(1,i-1);
            end
            memoriasalida(1,3)=memoriasalida(1,2);
            memoriasalida(1,2)=salida;

           % Filtrado senhal entrada
            memoriaentradasucia(1,2)=memoriaentradasucia(1,1);
            memoriaentradasucia(1,1)=getsample(AI);
            memoriaentradalimpia(1,2)=memoriaentradalimpia(1,1);
            memoriaentradalimpia(1,1)=0.1813*memoriaentradasucia(1,2);

memoriaentradalimpia(1,1)=memoriaentradalimpia(1,1)+(0.8187*memoriaentradalimpia(
1,2));
            matrizsalida(1,j)=memoriaentradalimpia(1,1);
            memoriaerror(1,1) =consigna-memoriaentradalimpia(1,1);

            %Entrada senhales digitales y asignacion de variables
            valordigitalentrada=getvalue(DIO);
            funcionamientocorrecto=valordigitalentrada(1,4);
            llamadaconfigurar=valordigitalentrada(1,3);

            %Configuracion nueva TF.
            if llamadaconfigurar==1
                [consigna,tfPID,tfPI,tfPD]=llamar_PIDSetup(Per);
            end
```

```
            %Calculo nueva senhal de salida. El error controlara que FT se
            %empleara.
            if abs(memoriaerror(1,1))<0.03
                num=tfPID.num{1};
                den=tfPID.den{1};
                salida=num(1)*memoriaerror(1,1);
                salida=salida+(num(2)*memoriaerror(1,2));
                salida=salida+(num(3)*memoriaerror(1,3));
                salida=salida-(den(2)*memoriasalida(1,2));
                salida=salida-(den(3)*memoriasalida(1,3));
                salida=salida/den(1);
            else
                num=tfPD.num{1};
                den=tfPD.den{1};
                salida=num(1)*memoriaerror(1,1);
                salida=salida+(num(2)*memoriaerror(1,2));
                salida=salida-(den(2)*memoriasalida(1,2));
                salida=salida/den(1);
                memoriasalida(1,1)=salida;

            end

            %Acondicionado de señal y tiempo.
            if salida < 0
                salida = 0;
            end
            if salida>10
                salida = 10;
            end
            putsample(AO,salida)
            toc;
            if (toc<Per)
                pause(Per-toc)
            elseif (llamadaconfigurar==0)
                disp(' Se ha excedido el tiempo del período')
                funcionamientocorrecto=0;

            end

        %Muestreo del estado de la planta por pantalla.
        plot(matrizsalida);
        xlabel('Numero de muestras');
        ylabel('Tension de entrada en la TAD');
    end

    %Se envia al controlador una senhal de 0V para apagar las bombillas y
    %se para el transito de informacion. Tambien se eliminan los objetos de
    %la tarjeta de la memoria.
    putsample(AO,0);
    stop(AI);
    stop(AO);
    delete(AI);
    delete(AO);
    clear AI
    clear AO
    disp('Salida del programa');
end
```

### 3.1.3.1.12 Identificacion

```
function [w] = Identificacion( )
%=========================================================================%
%El objetivo de esta funcion es identificar la funcion de transferencia de
%la planta para asi poder disenhar un adecuado regulador PID, siendo
%obtenida a partir de la variable de salida w.
%=========================================================================%

%Variables empleadas para el regulador PID y el calculo de la senhal a aplicar al
actuador.
    funcionamientocorrecto=1;
    [AI AO DIO]=configuracionrapida();
    [consigna,Per,tfPID,tfPI,tfPD]=PIDsetup();
    Per=0.2;
    num=tfPID.num{1};
    sizenumtotal=size(num);
    sizenum=sizenumtotal(2);
    den=tfPID.den{1};
    sizedentotal=size(den);
    sizeden=sizedentotal(2);
    memoriaerror=zeros(1,sizeden);
    memoriasalida=zeros(1,sizeden);
    memoriaentradasucia=zeros(1,2);
    memoriaentradalimpia=zeros(1,2);
    valordigital=zeros(1,4);
    salida=0;
    j=0;

    %Variables empleadas para la identificacion de la planta.
    I=eye(4);
    x=zeros(4,1); %Datos de entrada
    %Vtad -2, estado planta, estado planta-1, estado planta -2.
    w=zeros(4,1);%Parametros PID: B0 A2 A1 A0.
    w(2,1)=1; %Se considera A2 siempre como la unidad.
    %Matrices auxiliares.
    p=1000*rand(4,4);
    k=zeros(4,1);

    while funcionamientocorrecto==1
        tic


    %Actualizacion de variables dependientes del estado.
        j=j+1;
        for i=sizenum:-1:2
            memoriaerror(1,i)=memoriaerror(1,i-1);
        end
        memoriasalida(1,3)=memoriasalida(1,2);
        memoriasalida(1,2)=salida;

        x(1,1)=memoriasalida(1,3);
        x(4,1)=x(3,1);
        x(3,1)=x(2,1);
        salida=0;

   %Filtrado senhal entrada TAD.
         memoriaentradasucia(1,2)=memoriaentradasucia(1,1);
         memoriaentradasucia(1,1)=getsample(AI);
         memoriaentradalimpia(1,2)=memoriaentradalimpia(1,1);
         memoriaentradalimpia(1,1)=0.1813*memoriaentradasucia(1,2);

memoriaentradalimpia(1,1)=memoriaentradalimpia(1,1)+(0.8187*memoriaentradalimpia(
1,2));
```

```
        memoriaerror(1,1) =consigna-memoriaentradalimpia(1,1);
        valordigital=getvalue(DIO);
        funcionamientocorrecto=valordigital(1,4);

%Calculo senhal enviada al actuador.
      salida=num(1)*memoriaerror(1,1);
      salida=salida+(num(2)*memoriaerror(1,2));
      salida=salida+(num(3)*memoriaerror(1,3));
      salida=salida-(den(2)*memoriasalida(1,2));
      salida=salida-(den(3)*memoriasalida(1,3));
      salida=salida/den(1);

%Calculo de los nuevos valores de estado de planta calculados por el
%modelo calculado. La espera ayuda a evitar errores debido al empleo de
%un filtro en la senhal de entrada real y a que la planta tenga tiempo
%de estabilizarse.
      if j > 150
          disp(' Funcionando')
          x(2,1)=(w(1,1)*x(1,1));
          x(2,1)=(x(2,1))-(w(3,1)*x(4,1))-(w(4,1)*x(4,1));
          x(2,1)=x(2,1)/1;
      else
          x(2,1)=memoriaentradalimpia(1,1);
      end

  %Calculo variables RLS.

      if j > 150


           w=w+(p*x*inv(1+x'*p*x))*(memoriaentradalimpia(1,1)-x'*w);
           p=(I-(p*x*inv(1+x'*p*x))*x')*p;
           w(2,1)=1;
      end

  %Envio senhal real al actuador.
      if salida < 0
          salida = 0;
      end
      if salida>10
          salida = 10;
      end

      %El error determina la diferencia de valor entre el estado real de
      %la planta y el valor calculador por el modelo. Es muy importante
      %ya que muestra el nivel de precisión del modelo estimado.
      Error=memoriaentradalimpia(1,1)-x(3,1)
      estadoplanta=memoriaentradalimpia(1,1)

      putsample(AO,salida)
      toc;

      if (toc<Per)
          pause(Per-toc)
      else
          disp(' Se ha excedido el tiempo del período')
          funcionamientocorrecto=0;
      end


  end

  %Se envia al controlador una senhal de 0V para apagar las bombillas y
  %se para el transito de informacion. Tambien se eliminan los objetos de
```

```
    %la tarjeta de la memoria.
    putsample(AO,0);
    stop(AI);
    stop(AO);
    delete(AI);
    delete(AO);
    clear AI
    clear AO
    disp('Salida');

end
```

TÍTULO: **MEJORA DEL ALGORITMO DE CONTROL DE UNA PLANTA DE LABORATORIO E IDENTIFICACIÓN DE LA MISMA**

---

# PLIEGO DE CONDICIONES

---

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

**AVDA. 19 DE FEBRERO, S/N**

**15405 - FERROL**

FECHA: **SEPTIEMBRE DE 2013**

AUTOR: EL ALUMNO

Fdo.: GABRIEL CALVO RODRIGUEZ

# ÍNDICE

**4. Pliego de condiciones…..……83**

**4.1 Especificaciones de los materiales……..……85**

**4.1.1 Introducción…….……85**

**4.1.2 Selección de componentes…….……..85**

**4.1.2.1 Sistema de adquisición de datos…….……85**

**4.1.2.2 Actuador……..……86**

**4.1.2.3 Planta.……86**

**4.1.3 Especificaciones de montaje……..…… 86**

**4.1.4 Condiciones software……..…87**

**4.1.5 Condiciones hardware.……..……87**

**4.1.6 Pruebas.……88**

# 4. PLIEGO DE CONDICIONES

## 4.1 Especificaciones de los materiales

### 4.1.1 Introducción

Los componentes empleados en la realización de este proyecto se han seleccionado por cumplir aquellas características necesarias para la realización de este proyecto.

### 4.1.2 Selección de los componentes

El proveedor de cada uno de los distintos componentes ha de proveer la garantía de un correcto funcionamiento de cada uno de los distintos componentes. El proveedor ha de comprometerse a que los componentes que supla han de cumplir con las especificaciones por las cuales han sido diseñados, evitando cualquier tipo de copia que pueda provocar un mal funcionamiento del dispositivo.

A continuación se definirán los estándares mínimos que han de cumplir los distintos elementos para poder ser incluidos en el montaje. Se han de detallar las distintas comprobaciones realizadas en cada componente para garantizar que su comportamiento está dentro de las características definidas por el proveedor, y por ende es apto para su uso.

#### 4.1.2.1 Sistema de adquisición de datos

Es necesaria la realización de una verificación del correcto funcionamiento del sistema de adquisición de datos. Una vez correctamente conectado e instalado en el PC se realizarán una serie de pruebas mediante el software *Measurement and automation,* incluído en el paquete NI DAQ USB-6229, y mediante lista de instrucciones de Matlab. Se realizarán pruebas diferenciadas para los canales de entrada y los canales de salida empleados.

Las pruebas referidas a los distintos canales de entrada de datos tratan de verificar la correcta adquisición de datos con la frecuencia de muestreo empleada

en el regulador y la correspondencia entre en valor digital del canal de entrada y el valor analógico de la señal aplicada.

Para los canales de salida analógica de datos se han realizado pruebas de generación de señales. El valor de las señales generadas permanecerá siempre dentro del rango máximo soportado por el sistema. Se comprobará la naturaleza de dichas señales y se verificará la correspondencia entre la generación desde software y los resultados obtenidos en el bloque de conexiones externo.

#### 4.1.2.2 Actuador

El actuador se encarga de aportar potencia a la planta de forma lineal controlado mediante una señal de 0-10 V. Para comprobar el buen funcionamiento del actuador y la adecuación a sus características. Para ello se realizarán una serie de pruebas que consisten en enviar una señal de control dentro de los límites de funcionamiento y comprobar la señal de salida. El % de potencia aplicado a de variar de forma lineal con la señal de control. También se ha de controlar la no existencia de interferencias cuando el valor de la señal de control es 0.

#### 4.1.2.3 Planta

La planta de verificar el estado de todas las conexiones de la planta y del funcionamiento de la misma. Su funcionamiento se comprobará conectando los conectores de las bombillas a la red eléctrica y los conectores del ventilador a una fuente de tensión para comprobar que su funcionamiento es el adecuado.

El sensor de temperatura junto al circuito acondicionador tienen una sensibilidad de 100mV/ºC. Ha de comprobarse la tensión de salida a una temperatura conocida y verificar que se adecua a su sensibilidad.

### 4.1.3 Especificaciones de montaje

A continuación se especifican los requisitos a cumplir para un correcto montaje y conexionado del sistema:

- La TAD se conectará al PC en una de sus conexiones hembra USB libres mediante un cable USB doble macho estándar A/B. Un canal de entrada analógica de datos y un canal de salida analógica de datos estarán conectados al actuador y 2 canales de entrada analógica de datos estarán conectados a una fuente de tensión. El conexionado de la TAD al actuador y a la fuente de tensión se realizará mediante cable de 2 vías, en el extremo perteneciente al actuador o a la fuente tendrá conectores tipo banana y en el extremo de la TAD tendrá cable pelado.

- El conexionado de la salida del actuador a los conectores de las bombillas pertenecientes a la planta se realizará mediante cable de 2 vías con conectores tipo banana. El conexionado del actuador con el sensor de temperatura y su acondicionador se realizará a través de cable modular RJ45 con conectores del mismo tipo.

### 4.1.4 Condiciones software

Se ha empleado Matlab y su addon GUI para el diseño e implementación del regulador y de la interfaz gráfica. El equipo empleado para llevarlo a cabo ha de cumplir los siguientes requisitos:

- Sistemas operativos compatibles:
  - Windows XP Service Pack 3.
  - Windows XP x64 Edition Service Pack 2.
  - Windows Server 2003 R2 Service Pack 2.
  - Windows Vista Service Pack 2.
  - Windows Server 2008 Service Pack 2 o R2.
  - Windows 7.

- Matlab R2007a o versiones más recientes.

### 4.1.5 Condiciones hardware

El equipo ha de cumplir con una serie de requisitos mínimos para un funcionamiento óptimo del sistema de control:

- Procesador Intel o AMD x86 que soporte el set de instrucciones SSE2
- 1GB de memoria libre en espacio duro (Se recomienda 3-4)
- 1 GB de memoria RAM (Se recomienda 2048)

### 4.1.6 Pruebas

Este proyecto asegura el correcto funcionamiento del sistema en condiciones de trabajo normales y bajo un empleo adecuado. Sin embargo, será necesaria la prueba del mismo en instalaciones provisionales para usos de larga duración. Durante la fase de instalación será necesaria la realización de pruebas que aseguren el correcto funcionamiento del mismo. Las pruebas son:

- Comprobar el estado de cables y de las conexiones antes de realizar el arranque.
- Antes de emplear el regulador, ha de verificar las señales de entrada y salida de la TAD para comprobar su correcto funcionamiento. Esto servirá también para verificar el funcionamiento del sensor de temperatura y el acondicionamiento de la señal de salida de la planta.
- Comprobar, antes de emplear el regulador, el correcto aporte de potencia a la carga mediante el software *Measurement & Automation* o mediante cualquier modo manual. Esto servirá para confirmar el correcto funcionamiento de la placa de control lineal.
- Es recomendable el cierre de cualquier aplicación innecesaria en el ordenador para evitar el consumo de memoria que pueda conllevar problemas con el tiempo de respuesta debido a la saturación de la CPU.
- Una vez finalizado cada ensayo/prueba, se habrá de enviarle al orden a la TAD de tener una señal de salida de 0V para evitar que se siga alimentando a la planta pese haber terminado de trabajar con ella (Aparte de evitar el uso innecesario de los distintos componentes y del consumo de energía). En caso de producirse un bloqueo del último de valor de salida de la TAD, habrá de reiniciarse la tarjeta mediante el software *Measurement & Automation.*

TÍTULO: **MEJORA DEL ALGORITMO DE CONTROL DE UNA PLANTA DE LABORATORIO E IDENTIFICACIÓN DE LA MISMA**

---

# ESTADO DE MEDICIONES

---

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

**AVDA. 19 DE FEBRERO, S/N**

**15405 - FERROL**

FECHA: **SEPTIEMBRE DE 2013**

AUTOR: EL ALUMNO

Fdo.: GABRIEL CALVO RODRIGUEZ

# ÍNDICE

**5. ESTADO DE MEDICIONES……..……..89**

**5.1 Listado de materiales……...……..91**

**5.2 Mano de obra…….…… 91**

**5.2.1 Desarrollo del proyecto……91**

**5.2.2 Montaje del proyecto…..……...92**

# 5. ESTADO DE MEDICIONES

## 5.1 Listado de materiales

| Imagen | Descripción | Referencias | Cantidad(uds.) |
|---|---|---|---|
|  | Tarjeta de adquisición de datos NI DAQ USB-6229 | - | 1 |
|  | Cable USB doble macho A/B | - | 1 |
|  | Project Board | - | 1 |

Tabla 5.1.1 – Listado de materiales sistema de adquisición de datos

## 5.2 Mano de obra

Se detalla a continuación el tiempo estimado para el desarrollo y el montaje del presente proyecto.

### 5.2.1 Desarrollo del proyecto

| Tarea | Personal | Tiempo |
|---|---|---|
| Diseño Técnico | Ingeniero Técnico Industrial | 50 |
| Corrección y ajuste | Ingeniero Técnico Industrial | 20 |
| Memoria | Ingeniero Técnico Industrial | 30 |
| Total | Ingeniero Técnico Industrial | 100 |

Tabla 5.2.1.1 – Desglose del trabajo de desarrollo del proyecto

### 5.2.2 Montaje del proyecto

| Tarea | Personal | Tiempo |
|---|---|---|
| Montaje del sistema | 1 Oficial de 2ª | 1 Hora |
| Prueba y control de montaje | Ingeniero Técnico Industrial | 1 Hora |
| Total | - | 2 Horas |

5.2.2.1 – Desglose del trabajo de montaje del proyecto

TÍTULO: **MEJORA DEL ALGORITMO DE CONTROL DE UNA PLANTA DE LABORATORIO E IDENTIFICACIÓN DE LA MISMA**

---

# PRESUPUESTO

---

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

**AVDA. 19 DE FEBRERO, S/N**

**15405 - FERROL**

FECHA: **SEPTIEMBRE DE 2013**

AUTOR: EL ALUMNO

Fdo.: GABRIEL CALVO RODRIGUEZ

# ÍNDICE

**6. PRESUPUESTO...……93**

**6.1 Cuadro de precios…..……95**

**6.2 Gastos de desarrollo……95**

**6.2.1 Coste de personal……..……95**

**6.3 Coste total……..……96**

# 6 PRESUPUESTO

## 6.1 Cuadro de precios

| Descripción | Cantidad (uds.) | Precio unitario(€) | Precio total(€) |
|---|---|---|---|
| NI USB-6229 M Series, 26 cm, Screw Term, EURO (240V) | 1 | 1,549.00 | 1,549.00 |
| Bread Board Jumpwires 83x55mm | 1 | 7.54 | 7.54 |
| Importe total | | | 1,556.54 |

Tabla 6.1.1 - Cuadro de precios sistema de adquisición de datos.

## 6.2 Gastos de desarrollo

### 6.2.1 Coste de personal

| Tipo de empleado | Pago por hora |
|---|---|
| Ingenierio técnico Industrial | 40€/hora |
| Operario de 2ª | 15€/hora |

Tabla 6.2.1.1 – Coste de mano de obra

## 6.3 Coste total

| Operación | | Importe (€) |
|---|---|---|
| Materiales | Sistema de adquisición de datos | 1,549.00 |
| | Project Board | 7.54 |
| Personal | Ingeniero Técnico Industrial | 100 x 40€/h = 4000 |
| | Operario de 2ª | 2 x 15€/h = 30 |
| **TOTAL** | | **5,586.54 €** |
| | **+21% I.V.A** | **6,759.71 €** |

Tabla 6.3.1 - Coste total

El coste total asciende a la cantidad **de seis mil setecientos cincuenta y nueve euros con setenta y un céntimos (6,759.71€).**

El Ingeniero Técnico Industrial

Fdo: Gabriel Calvo Rodríguez

Ferrol, Septiembre 2011