DESARROLLO DE SISTEMA FPGA APLICADO A CONTROL E IDENTIFICACIÓN DE UN MÓDULO EXPERIMENTAL PARA REGULACIÓN DE PH

Juan Soto-Bohórquez

Piura, marzo de 2013

Facultad de Ingeniería

Departamento de Ingeniería Mecánico-Eléctrica

Soto, J. (2013). *Desarrollo de sistema FPGA aplicado a control e identificación de un módulo experimental para regulación de pH* (Tesis de pregrado en Ingeniería Mecánico-Eléctrica). Universidad de Piura, Facultad de Ingeniería. Programa Académico de Ingeniería Mecánico-Eléctrica. Piura, Perú.

U N I V E R S I D A D D E P I U R A

FACULTAD DE INGENIERÍA

“Desarrollo de sistema FPGA aplicado a control e identificación de un módulo experimental para regulación de pH”

Tesis para optar el Título de

Ingeniero Mecánico – Eléctrica

Juan Carlos Soto Bohórquez

ASESOR: Dr. Ing. William Ipanaqué Alama

Piura, Marzo 2013

Prólogo

La medición del nivel de pH es una herramienta muy importante en el sector industrial, su control automático es relevante en procesos de tratamiento de aguas, procesamiento de imagen, robótica, control numérico, inspección geológica, sistemas de adquisición de datos, entre otros.

Esta tesis muestra una aplicación para un sistema embebido con arquitectura FPGA (*Field Program Gate Array*) aplicándolo al control en un sistema no lineal. Para ello se ha implementado un regulador PI con ganancia programable (*Gain Scheduling*) ya que es una de las técnicas no complejas para el control de procesos no lineales. Así también hacemos uso de un software para la modelación de procesos en sistemas de tiempo real (*DSPACE*).

Las aplicaciones de sistemas embebidos se han incrementado notablemente, unas de las técnicas denominada FPGA permite implementar estructuras en paralelo y segmentado, dispone también de adecuados recursos [11], [12]. Los FPGA han demostrados buenas prestaciones en el control avanzado [2], [9]; en [3] se ilustra una aplicación en cascada con un PID para regular la salida de un sistema de salinización de agua. Una aplicación en FPGA ilustra el control de un péndulo invertido en [20]. Una interesante revisión sobre herramientas de diseño y el espectro de aplicaciones se describe en [7], [14]. Estas herramientas requieren de una metodología apropiada para su implementación, para que estimulen las prestaciones de aplicaciones en tiempo real. Sistemas embebidos basados en FPGA tienen características atractivas como: disminución de costos debido al alto desarrollo de tecnología VLSI, sistemas embebidos con restricciones de tiempo real, los tiempos de ejecución e implementación se reducen notablemente; siendo una buena opción en el diseño de prototipos [4], [5]. Así también posee gran flexibilidad en la adaptación de diversas aplicaciones, sobre todo en aquellas donde se requiere alta frecuencia de trabajo, un alto grado de paralelismo y reconfigurabilidad [17] - [19].

En este trabajo se presenta una metodología de diseño e implementación, de un sistema embebido, con una aplicación en tiempo real [1], [6]. La aplicación está constituida por un proceso con muy alta no linealidad; a pesar de eso el control con ganancia programable empleado en el FPGA ha dado buenos resultados.

Esta tesis propone un controlador adaptativo programado por medio de un algoritmo de control PI con *Gain Scheduling*, en un rango donde solo es afectada la no linealidad estática. De esta forma se procede a linealizar por tramos, encontrando los parámetros del controlador PI por medio de sintonización por el método de asignación de polos.

Los procesos industriales en su mayoría presentan un comportamiento no lineal, y al aproximarlo a uno lineal causa deficiencias en el control automático. Por lo que se busca implementar nuevas estrategias de control avanzado, así como la técnica presente en esta tesis, control adaptativo programado por medio de un algoritmo PI con *Gain Scheduling*, permitiendo afrontar el problema.

Se ha implementado un control automático sobre un módulo existente en el laboratorio de electrónica y automática de la Universidad de Piura. Además se ha desarrollado hardware de última generación como es la tarjeta FPGA diseñada.

Los resultados experimentales han sido muy satisfactorios, el pH se llega a controlar de buena forma.

La velocidad de procesamiento de la tarjeta FPGA, 50 Mhz, hacen a este dispositivo sea capaz de controlar procesos de dinámica lenta como procesos de dinámica rápida.

Durante el desarrollo de la tesis en el laboratorio de sistemas automáticos de control, nos permitió la publicación de los siguientes artículos relacionados:

- Control con ganancia programable en proceso no lineal con sistema embebido FPGA, Congreso latinoamericano de control automático CLCA-2012.
- Diseño, construcción y puesta en funcionamiento de tarjeta PCB basada en FPGA aplicada a modulo experimental de laboratorio, INTERCON 2010.
- Estrategia de control con ganancia variable en sistemas embebidos basado en FPGA aplicado a un motor DC, INTERCON 2009.

Así también debido a la actividad de investigación y desarrollo (I+D) en el laboratorio de Sistemas automáticos de control, y gracias a las competencias adquiridas, permitieron participar en este otro artículo:

- Desarrollo de software para control de módulo *WorkStation de Festo* con PLC *Siemens Simatic S7-314C* para aplicaciones industriales, INTERCON 2010, Juan Carlos Soto Bohórquez, Juan Fidel Córdova Rosales, William Ipanaqué Alama, Puno, Perú.

Así también ha permitido la presentación de una solicitud de patente por modelo de utilidad ante INDECOPI:

- Tarjeta controlador para procesos automáticos con tecnología FPGA.

Agradezco a todas las aquellas personas que de una u otra forma me ayudaron en el desarrollo de esta tesis. De una manera especial agradezco al Dr. Ing, William Ipanaqué Alama, mi asesor, por su constante dedicación. A mis compañeros del Departamento de Electrónica y Automática de la Universidad de Piura, Julia Acuña, Ivan Belupú, José Carlos Oliden, José Manrique, entre otros.

Gracias a mi madre por ser ejemplo, modelo, sacrificio y enseñarme a soñar en grande, para mí la mejor herencia. Así también a mi esposa Carina por darme los días más felices de mi vida y apoyo constante.

Resumen

Esta tesis muestra una aplicación para un sistema embebido con arquitectura FPGA (*Field Program Gate Array*) aplicándolo al control en un sistema no lineal. Para ello se ha desarrollado un sistema embebido e implementado un regulador PI con ganancia programable ya que es uno de las técnicas no complejas para este tipo de procesos no lineales. La medición del nivel de pH es una herramienta muy importante en el sector industrial.

El desarrollo de esta tesis ha llevado a la presentación de artículos en diferentes congresos como son:

- Control con ganancia programable en proceso no lineal con sistema embebido FPGA, Congreso latinoamericano de control automático CLCA-2012.
- Diseño, Construcción y puesta en funcionamiento de tarjeta PCB basada en FPGA aplicada a modulo experimental de laboratorio, INTERCON 2010.
- Estrategia de Control con ganancia variable en sistemas embebidos basado en FPGA aplicado a un motor DC, INTERCON 2009.

Así también el autor cuenta con una presentación de solicitud de patente ante INDECOPI por modelo de utilidad:

- Tarjeta Controlador para procesos automáticos con tecnología FPGA.

Índice

Introducción

El desarrollo de esta tesis comprende el estado del arte en temas basados con arquitectura FPGA, diseño de sistemas electrónicos PCBs, implementación de un controlador PI con ganancia programable *(Gain Scheduling*) y modelación de procesos en sistemas de tiempo real (*DSPACE*).

Los FPGA han demostrados buenas prestaciones en el control avanzado [20]. Los sistemas embebidos basados en tecnología FPGA tienen características atractivas, tales como: disminución de costos debido al alto crecimiento tecnológico, sistemas embebidos con restricciones de tiempo real [9]-[10], los tiempos de ejecución e implementación se reducen notablemente; siendo una buena opción en el diseño de prototipos [1]. Así también posee gran flexibilidad en la adaptación de diversas aplicaciones, sobre todo en aquellas donde se requiere alta frecuencia de trabajo, alto grado de paralelismo y reconfigurabilidad [17]-[19].

En el diseño y fabricación de sistemas electrónicos *Printes Circuit Board (PCB)* [16], encontramos una variedad de software; dentro de los cuales podemos citar los siguientes programas: *Altium Designer, Orcad, Eagle, Proteus,* etc. Permitiendo el desarrollo de los planos esquemáticos del proyecto y la generación de archivos de fabricación *(GERBER)*.

Las técnicas de control adaptativo, entre ellas, el controlador *Gain Scheduling*, ofrece ventajas para el control de procesos con propiedades dinámicas variables.

La modelación del proceso en tiempo real, regulación de pH, se llevó a cabo por medio del software *DSPACE*, el cual simplifica el desarrollo de sistemas complejos. Se basa en una herramienta de diagrama de bloques basada en el desarrollo *Simulink – Matlab.*

Se presenta una alternativa de control basada en sistemas embebidos, en particular basado en una arquitectura FPGA para la implementación de controladores, un regulador PI con ganancia programable [15]. Así también el sistema PCB diseñado se le ha incluido propiedades de comunicación hacia un sistema SCADA.

La tesis se desarrolla en cinco capítulos. El primer capítulo introduce los conceptos básicos que involucran al *pH*. El segundo capítulo describe la arquitectura FPGA y lenguajes de descripción de hardware *(HDLs)*. El tercer capítulo repasa la teoría de identificación y controladores PID. En el cuarto capítulo describe detalladamente la técnica de desarrollo de software y hardware. El quinto capítulo muestra los resultados experimentales de la implementación del control PI con ganancia programable *(Gain Scheduling)*. Así mismo se

incluyen 4 anexos: En el Anexo A, presentamos los archivos de fabricación, llamados archivos GERBER. En el Anexo B, incluimos un manual de configuración del componente Ethernet *WIZNET 110S*. En el Anexo C, incluimos un manual de la tarjeta desarrollada FPGA. En el Anexo D incluimos pruebas de identificación y control a nivel de simulación en el rango donde el pH es altamente no lineal.

Esta tesis propone un controlador adaptativo programado por medio de un algoritmo de control PI con *Gain Scheduling*, en rango donde solo es afectada la no linealidad estática.

Los parámetros del controlador PI por medio de sintonización por el método de asignación de polos para cada tramo.

Se propone una alternativa de control utilizando un sistema que cuenta con un alto grado de paralelismo, Los FPGAs son una arquitectura adecuada para la implementación de controladores como el PID, Predictivo, *Fuzzi Logic*, entre otros. En este caso se ha implementado un algoritmo de control PI con *Gain Scheduling* debido a la gran no linealidad estática del proceso.

Los procesos industriales en su mayoría presentan un comportamiento no lineal, y al aproximarlo a uno lineal causa deficiencias en el control automático. La técnica presente en esta tesis, control adaptativo programado por medio de un algoritmo *PI con Gain Scheduling,* ha permitido afrontar el problema.

Se ha implementado un control automático sobre un módulo existente en el laboratorio de electrónica y automática de la universidad de Piura. Desarrollando hardware de última generación como es la tarjeta FPGA diseñada.

Los resultados experimentales han sido muy satisfactorios, el pH se llega a controlar de buena forma.

La velocidad de procesamiento de la tarjeta FPGA, 50 Mhz, hacen a este dispositivo sea capaz de controlar procesos de dinámica lenta como procesos de dinámica rápida.

Durante el desarrollo de la tesis en el laboratorio de sistemas automáticos de control, nos permitió la publicación de los siguientes artículos relacionados:

- Control con ganancia programable en proceso no lineal con sistema embebido FPGA, Congreso Latinoamericano de Control Automático CLCA-2012.
- Diseño, Construcción y puesta en funcionamiento de tarjeta PCB basada en FPGA aplicada a modulo experimental de laboratorio, INTERCON 2010.
- Estrategia de Control con ganancia variable en Sistemas Embebidos basado en FPGA aplicado a un motor DC, INTERCON 2009.

Así también debido a la actividad de investigación y desarrollo (I+D) en el laboratorio de Sistemas Automáticos de Control, y gracias a las competencias adquiridas, permitieron participar en este otro artículo:

- Desarrollo de Software para control de módulo *WorkStation de Festo* con PLC *Siemens Simatic S7 314C* para aplicaciones industriales, INTERCON 2010, Juan

Carlos Soto Bohórquez, Juan Fidel Córdova Rosales, William Ipanaqué Alama, Puno, Perú.

Así también ha permitido la presentación de una solicitud de patente por modelo de utilidad ante INDECOPI:

- Tarjeta Controlador para procesos automáticos con tecnología FPGA.

Capítulo 1

Definiciones básicas

1.1 Sistemas embebidos

La definición de "sistema embebido" es muy diversa y muy difícil de precisar. Debido a la constante evolución en los avances tecnológicos y la disminución en el costo para implementar diversos componentes hardware y software [7].

Se encontrará probablemente algunas definiciones, es importante conocer el significado por qué puede no ser exacta hoy en día. Las siguientes son algunas de las descripciones más comunes de un sistema embebido.

- *Un sistema embebido está más limitado funcionalmente en hardware y/o software que una computadora personal (PC).* Posee limitaciones de hardware, como en rendimiento del procesador, consumo de potencia, memoria, funcionalidad de hardware, entre otros. En software las limitaciones son relativas, cuentan con aplicaciones a escala reducida, no poseen un sistema operativo (OS) ó es limitado. Actualmente ésta definición es parcialmente cierta, tanto hardware y software encontrados en los ordenadores de ayer y hoy han sido construidos en los más complejos diseños de sistemas embebidos.

- *Un sistema embebido es diseñado para llevar a cabo funciones dedicadas.* La mayoría de dispositivos embebidos son diseñados para funciones específicas. Sin embargo hoy en día se encuentran dispositivos con sistemas embebidos diseñados para realizar una variedad de funciones principales.

- *Un sistema embebido en un sistema computacional con alta calidad y requisitos de fiabilidad como otros tipos de sistemas computacionales.* Algunas familias de dispositivos embebidos poseen una alta calidad y requisitos de fiabilidad.

- *Algunos dispositivos son llamados sistemas embebido, tales como PDAs ó web pads, pero no lo son realmente.* No todas las definiciones de sistemas tradicionales son actualmente sistemas embebidos.

Otras definiciones [10].

• “Un sistema embebido es cualquier dispositivo que incluye un computador programable, pero en sí mismo no es un computador de propósito general”.

• “Un sistema embebido es un sistema electrónico que contiene un microprocesador o microcontrolador; sin embargo, no pensamos en ellos como un computador”.

• “Las personas usan el término sistema embebido para referirse a cualquier sistema de cómputo escondido en algún producto o dispositivo”.

• “Un sistema embebido es un sistema cuya función principal no es computacional, pero es controlado por un computador integrado. Este computador puede ser un microcontrolador o un microprocesador. La palabra embebido implica que se encuentra dentro del sistema general, oculto a la vista, y forma parte de un todo de mayores dimensiones”.

Un sistema embebido posee hardware de computador junto con software embebido como uno de sus componentes más importantes. Es un sistema electrónico dedicado para aplicaciones o productos. Puede ser un sistema independiente o parte de un sistema mayor.

Un sistema embebido tiene tres componentes principales:

1. Hardware.

2. Un software primario o aplicación principal. Este software o aplicación lleva a cabo una tarea en particular, o en algunas ocasiones una serie de tareas.

3. Un sistema operativo que permite supervisar la(s) aplicación(es), además de proveer los mecanismos para la ejecución de procesos. En muchos sistemas embebidos es requerido que el sistema operativo posea características de tiempo real.

Los rápidos progresos de las tecnologías *Very Large Scale Integration (VLSI) y Electronic Design automation (EDA),* han hecho posible el desarrollo de complejos y compactos controladores de alto rendimiento para sistemas electrónicos industriales.
El uso de modernas herramientas *EDA* nos permite crear, simular y verificar un diseño, sin comprometer hardware; verificando el correcto funcionamiento del producto final.

En el diseño de circuitos electrónicos industriales, se tiene que considerar varios criterios.
Así como el costo, el consumo de energía (esencial en caso de los sistemas embebidos), el rendimiento de las aplicaciones y, sobre todo, compatibilidad para integración con hardware elegido.

1.1.1 Arquitectura de un sistema embebido

La arquitectura de un sistema embebido es una abstracción de un hardware embebido, siendo una generalización del sistema pues no muestra detalles e información, tal como el código fuente de un software ó el diseño de un hardware electrónico.

En el nivel de arquitectura, los componentes hardware y software en un sistema embebido están representados como algunos elementos de interacción.

Los elementos son representaciones de hardware y/o software, cuyo detalle de implementación ha sido solo abstraída, dejando solamente información del comportamiento e inter-relación.

Los elementos de la arquitectura pueden estar integrados dentro del mismo dispositivo embebido ó externamente interactuando con los elementos internos.

En conclusión, la arquitectura embebida incluye elementos del mismo dispositivo embebido y elementos que interactúan con este.

1.1.2 Modelo de un sistema embebido

Existe una variedad de herramientas los cuales introducen conceptos técnicos y fundamentales de sistemas embebidos [17]. El modelo referido se muestra en la Fig. 1.1.

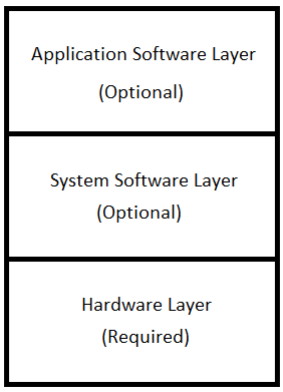


Fig. 1.1 Modelo de sistema embebido

El modelo presente cuenta con una arquitectura de alto nivel, introduce la mayoría de elementos localizados dentro de un sistema embebido. Todos los modelos tienen al menos una capa, *hardware layer*, o todas las capas (*hardware, system software and application software layers*).

La capa *Hardware* tiene la mayoría de componentes físicos. Las otras capas contiene el software el cuál es procesado por el sistema embebido.

El modelo de un sistema Embebido se puede asociar a un modelo universal, como es el, *Open Systems Interconnection (OSI) reference model.*

Los requerimientos de un sistema pueden ser agrupados en un modelo OSI, el cual fue creado en 1980 por la *International Organization for Standardization (ISO)*

1.1.3 Tecnología FPGA

En los últimos años la utilización de los dispositivos *FPGA's (Field Programmable Gate Array)* para la realización de procesado digital ha tomando gran relevancia, reemplazando cada vez más a los *ASIC's* y a los *DSP's* que requieren gran cómputo de datos, así como, el operador *FFT*, filtros *FIR, IIR*.

Los *FPGA's* son dispositivos lógicos de propósito general, programable por los usuarios, contienen celdas lógicas idénticas que se puede ver como componentes estándar, también llamados bloques lógicos, se interconectan por medio de una matriz de cables y *switches* programables.

Dependiendo de la arquitectura, se encuentran de diferentes tamaños, estructura, número de bloques y puertos de entradas y salidas.

Debido a su capacidad de reconfiguración y velocidad, son dispositivos que ofrecen una gran variedad de aplicaciones.

Las conexiones de los *switches* y las funciones de los bloques lógicos, son definidas por el software de programación. Permite también validar un diseño para una mayor eficiencia del dispositivo.

1.2 Definición de ácidos y bases

Según el químico sueco ***Svante Arrhenius*** los ácidos son sustancias capaces de ionizarse en agua para formar iones hidrógeno *(H^+)* y las bases o sustancias alcalinas son sustancias capaces de ionizarse en agua para ceder iones hidroxilo *(OH^-)*.

En 1923, el químico danés ***Johannes Brönsted*** junto al químico británico ***Thomas Lowry***, plantearon otra teoría más aceptable que indica que los ácidos son sustancias capaces de ceder protones (iones hidrógeno) y las bases son sustancias capaces de aceptar dichos protones, por lo tanto no es necesaria la presencia de un medio acuoso. Esta teoría sostiene que una reacción ácido-base es la transferencia de un protón de un ácido hacia una base. Una reacción acuosa que se produce entre un ácido y una base forma generalmente sal y agua, así:

$$Ácido + Base \rightarrow Sal + Agua$$

La definición de *Brønsted-Lowry* es la definición más ampliamente usada; salvo que se especifique de otra manera, se asume que las reacciones ácido-base involucran la transferencia de un catión hidrón *(H+)* de un ácido a una base.

En 1923, Lewis también propuso una nueva teoría respecto al concepto de ácidos y bases, aunque esta teoría no tendría repercusión hasta años más tarde. Según la teoría de Lewis una base es aquella sustancia que puede donar un par de electrones. El ion OH−, al igual que otros iones o moléculas como el *NH3, H2O,* etc., tienen un par de electrones no enlazantes, por lo que son bases. Todas las bases según la teoría de *Arrhenius* o la de *Brønsted y Lowry* son a su vez bases de *Lewis*.

1.2.1 Clasificación de ácidos y bases

Existen muchas clasificaciones de ácidos y bases según el aspecto que se desea analizar, la más utilizada es según el grado de ionización en solución acuosa, los ácidos y bases se clasifican como fuertes y débiles.

Un ácido fuerte es un ácido que se disocia por completo en solución acuosa para ganar electrones (donar protones), generalmente en una solución acuosa en condiciones normales de presión y temperatura, la concentración de iones hidronio es igual a la concentración de ácido fuerte introducido en la solución. Aunque por lo general se asume que los ácidos fuertes son los más corrosivos, esto no es siempre cierto. En su mayoría son inorgánicos, tales como: *ácido clorhídrico, ácido sulfúrico, ácido nítrico, ácido perclórico.*

La definición típica de ácido débil es un ácido que no se disocia completamente. La diferencia que separa las constantes de disociación ácida en los ácidos fuertes de la de todos los otros ácidos es tan pequeña que se trata de una demarcación razonable. Entre éstos se encuentran el *ácido fluorhídrico, el ácido acético, el ácido carbónico.*

La base fuerte es aquella que se disocia cuantitativamente en disolución acuosa, en condiciones de presión y temperatura constantes. Además fundamentalmente son capaces de aceptar protones *H+*. Entre ellas se encuentran: *hidróxido de sodio, hidróxido de potasio e hidróxido de calcio.*

Una base débil también aporta *iones OH−* al medio, pero está en equilibrio el número de moléculas disociadas con las que no lo están, y entre ellas encontramos *hidróxido de amonio, hidracina.*

Tabla1.1 Clasificación de ácidos y bases

	Ácidos	***K_a (en mol l^{-1})***	***Bases***	***K_b (en mol l^{-1})***
Fuertes	HCl		$NaOH$	
	HBr		hidróxidos del grupo 1	
	HI		$Ca(OH)_2$	
	H_2SO_4	(fuerte en la 1ª disociación)	hidróxidos del grupo 2, salvo Be	
	HNO_3			
	$HClO_4$			
Débiles	HIO_3	$1{,}7\ 10^{-1}$	$N(C_2H_5)_3$	$1{,}0\ 10^{-3}$
	H_2SO_3	$1{,}6\ 10^{-2}$	$N(CH_3)_3$	$6{,}5\ 10^{-5}$
	$HClO_2$	$1{,}0\ 10^{-2}$	NH_3	$1{,}8\ 10^{-5}$
	H_3PO_4	$7{,}1\ 10^{-3}$ (1ª disociación)	Piridina, C_5H_5N	$1{,}8\ 10^{-9}$
	HNO_2	$4{,}3\ 10^{-4}$ (1ª disociación)	Urea, $CO(NH_2)_2$	$1{,}3\ 10^{-14}$
	HF	$3{,}5\ 10^{-4}$		
	$HCOOH$	$1{,}8\ 10^{-4}$		
	C_6H_5COOH	$6{,}5\ 10^{-5}$		
	CH_3COOH	$1{,}8\ 10^{-5}$		
	H_2CO_3	$4{,}3\ 10^{-7}$ (1ª disociación)		
	$HClO$	$3{,}0\ 10^{-8}$		
	$HBrO$	$2{,}0\ 10^{-9}$		
	$B(OH)_3$	$7{,}2\ 10^{-10}$		
	HCN	$4{,}9\ 10^{-10}$		
	HIO	$2{,}3\ 10^{-11}$		

1.2.2 Propiedades de los ácidos y bases

Las propiedades de la mayoría de los ácidos protónicos son:

- Presentan sabor ácido como es el caso del ácido cítrico en la naranja y el limón.
- Son corrosivos y producen quemaduras a la piel.
- Produce efervescencia al contacto con el mármol.

- Hacen cambiar el color de muchos indicadores (tintes de color fuerte). Cambian el color del papel tornasol azul a rosa, el anaranjado de metilo de anaranjado a rojo y deja incolora a la *fenolftaleína*.
- Reaccionan con óxidos metálicos e hidróxidos metálicos para formar sales y agua.
- Reaccionan con sales de ácidos más débiles para formar ácidos más débiles.
- Las soluciones acuosas de ácidos protónicos conducen corriente eléctrica porque están ionizadas.

Las propiedades de la mayoría de las soluciones acuosas básicas son:

- Presentan sabor amargo.
- Produce sensación resbalosa o jabonosa.
- Hacen cambiar el olor de muchos indicadores. Azulean el papel de tornasol.
- Reaccionan neutralizando los ácidos.
- Generan precipitados (sustancias en fase sólida en el seno de un líquido) al ser puestas en contacto con ciertas sales metálicas (por ejemplo, de calcio y de magnesio).
- Reaccionan con ácidos protónicos para formar sales y agua.
- Sus soluciones acuosas conducen la corriente eléctrica porque están ionizadas o disociadas.
- Pueden originar irritaciones a la piel y son muy inflamables.

1.3 Definición del potencial de hidrógeno (*pH*)

El potencial de hidrogeno (pH), es una medida de la acidez o la alcalinidad de una solución acuosa. Indica la concentración de iones hidronio $[H_3O^+]$ presentes en algunas sustancias.

Por lo general, la medida se realiza en estado líquido, pero también se puede utilizar para gases. Para definirlo matemáticamente se expresa como el negativo del logaritmo de la concentración molar de iones hidrógenos H^+. Además, se debe tener en cuenta que se trabaja con pH en lugar de PH_3O debido a que el ión H_3O es representado por H^+.

$$pH = -log_{10} H^+ = -log_{10} H_3O^+$$

Es mejor trabajar con esta definición pues resulta una escala más fácil de visualizar al no trabajar con números muy pequeños (mayor longitud).

El pH es una medida muy útil pues nos ayuda a saber el grado de acidez o basicidad de una sustancia, el rango en que se encuentra está entre 0 y 14. La zona ácida está definida por los valores de *pH* menores a 7 *([H+] > [OH-])* y la zona básica, por los valores de *pH* mayores a 7 *([H+] < [OH-])*. Si el valor de pH es 7, la solución será neutra, lo que significa que la concentración de ácido es igual a la de base *([H+] = [OH-])*. A distintas temperaturas, el valor de *pH* neutro puede variar debido a la constante de equilibrio del agua *(Kw)*.

Tabla 1.2 Escala de pH

Zona	pH	[H+] (mol/l)	[OH-] (mol/l)
	0	1	0.00000000000001
	1	0.1	0.0000000000001
	2	0.01	0.000000000001
Acida	3	0.001	0.00000000001
	4	0.0001	0.0000000001
	5	0.00001	0.000000001
	6	0.000001	0.00000001
Neutra	7	0.0000001	0.0000001
	8	0.00000001	0.000001
	9	0.000000001	0.00001
	10	0.0000000001	0.0001
Alcalina	11	0.00000000001	0.001
	12	0.000000000001	0.01
	13	0.0000000000001	0.1
	14	0.00000000000001	1

1.3.1 Métodos de Medición de pH

La medición del pH de una solución es uno de los procedimientos más usados en la industria, sin embargo existen distintos métodos de medición según la exactitud y precisión deseada en la aplicación.

Sin embargo, el gusto ó sabor, fue el primer "método" utilizado para la determinación de la acidez; para luego pasar a la utilización de elementos naturales indicadores, que variaban su color en función de la acidez o alcalinidad de un medio. Llevaron a utilización de papeles indicadores. Actualmente, la acidez o alcalinidad se cuantifican muy bien con la ayuda de electrodos y *pH-metros.* Entre los distintos métodos tenemos:

Papel Indicador de pH

El papel está impregnado con un indicador universal o mezcla de indicadores cualitativos, que al introducirse en la solución a analizar adopta un color diferente, según el grado de acidez, luego se compara con un diagrama de colores y se tiene un valor aproximado de *pH*. El papel de litmus o papel tornasol es el más conocido. Es el método más inexacto además de barato con respecto a los demás métodos.

Fig. 1.2 Cinta de papel indicador de pH

Indicadores químicos

Se trata de ácidos o bases débiles que presentan diferente color según el *pH*. El procedimiento es agregarlas a las soluciones para luego compararlas con soluciones estándar de pH conocido, que también han sido afectadas por el mismo indicador y así conocer el valor aproximado de *pH*.

Los ejemplos más conocidos son el naranja de metilo que varía de color rojo a naranja en el intervalo de *pH* 3,1 - 4,4 y la fenolftaleína que varía de incoloro a rosado / violeta y va desde *pH* 8 a *pH* 10.

Fig. 1.3 Indicador naranja de metilo

Fig. 1.4 Fenolftaleína en entorno básico (izq.) y extremadamente ácido (der.)

PH-metro

El pH-metro es un sensor utilizado en el método electroquímico para medir el *pH* de una disolución. Este es un método más exacto que consiste en un sensor que internamente realiza una medida de la diferencia de potencial entre dos electrodos, uno de referencia interna (generalmente un pH igual a 7 que es considerado neutro) y otro externo (a medir).

Como los valores de diferencia de potencial con muy pequeños, viene incluido un amplificador de estos valores de voltaje.

Las tecnologías disponibles para la medición de pH pueden clasificarse en dos grandes grupos: Electroquímicos y Ópticos. Dentro de los electroquímicos el sensor de pH de vidrio, los electrodos de membrana de líquido, y los electrodos de estado sólido (el más utilizado es el antimonio).

E principio de funcionamiento de un pH-metro es:

El circuito básico de un *pH*-metro se presenta en la Fig. 1.5. El funcionamiento del pH-metro se basa en una celda que posee dos electrodos, uno de referencia y el otro que es sensible al ion hidrógeno se inserta en la solución a la cual se le quiere medir el pH.

Además está conectado a un amplificador seguidor de voltaje (10^{12} Ω) que tiene grandes corrientes de salida y también llamado buffer, pues se trata de un amplificador de potencia y no de voltaje (*Vi* no cambia). El segundo amplificador es de voltaje, el voltaje V_0 de salida es mucho mayor que *Vi*.

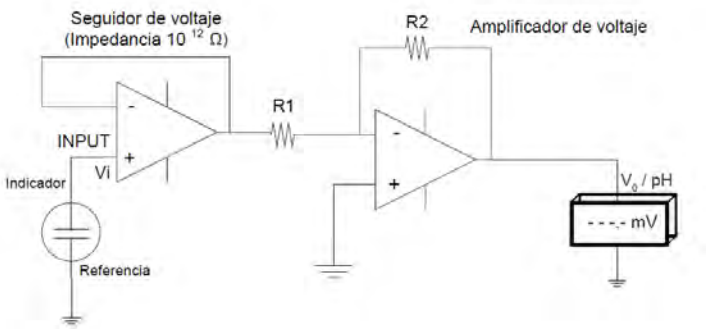


Fig. 1.5 Circuito simplificado de un pH-metro.

El pH-metro está diseñado para registrar un valor de potencial de cero voltios a pH 7. Los electrodos de vidrio y de referencia son también diseñados para obtener el mismo potencial cuando los electrodos se sumergen en una solución buffer de pH 7. El modelo matemático que explica este principio se expresa en la siguiente ecuación.

$$E_V = -\frac{RT}{F}(1000)Ln(10)(pH - 7) \qquad (4.1)$$

Dónde:
E_V es el potencial medido en mili voltios.
R= 8.31441 $J\ mol^{-1}\ k^{-1}$
F= 96 484.56 $C\ mol^{-1}$
T= (273.15 + t °C) K

En resumen, se mide el potencial generado por un electrodo de vidrio que es sensible a la actividad del ión H^+, este potencial es comparado contra un electrodo de referencia, que genera un potencial constante e independiente del *pH*. El electrodo de referencia que se utiliza es el de *calomel* saturado con cloruro de potasio, el cual sirve como puente salino que permite el paso de los mili-voltios generados hacia al circuito de medición.

En el siguiente esquema se muestran los electrodos utilizados:

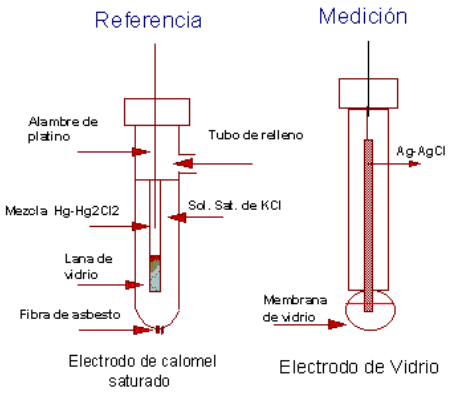


Fig. 1.6 Esquema de los electrodos para medir el *pH*

1.4 Aplicaciones de medición y control de *pH*

La medición y control del *pH* es de mucha importancia, como por ejemplo: en la biología, la química, la medicina, entre otras. Las aplicaciones se encuentran en muchos campos tales como: tratamiento de la pureza del agua en la entrada a una caldera, regulación de velocidad en reacciones químicas, tratamiento y neutralización de aguas residuales para su posterior utilización, regulación de acidez y control de activación de bacterias en la cerveza, regulación del *pH* en la fermentación del etanol, sulfitación en el jugo de caña, entre otros.

Algunas aplicaciones industriales son:

En la industria alimenticia es muy importante evitar la contaminación para garantizar un buen producto libre de microorganismos dañinos, y donde los niveles de *pH* demuestran las condiciones higiénicas en el proceso de transformación del producto.

La concentración iónica del hidrógeno afecta a esos microorganismos y también a la acción de los bactericidas, por lo tanto el índice de *pH* influye directamente en el control aplicado para evitar la activación de microorganismos y bacterias.

Por ejemplo, el tratamiento de alimentos en una atmósfera modificada con *pH* inferior de 4,6 puede inhibir la proliferación de microorganismos como el *"Clostridiumbotulinum"* peligroso para la salud humana.

a. Industria azucarera

El objetivo del control de *pH* en la industria azucarera es evitar la generación de microorganismos dañinos para el producto. Un caso especial es la *dextranasa* que es un compuesto no deseado sintetizado por microorganismos contaminantes que al aumentar su viscosidad daña al producto.

b. Industria cervecera

En la producción de cerveza es muy importante mantener un nivel de *pH* adecuado, pues de ser mayor y no seguirse la medida estándar es posible la aparición de agentes patógenos y al ser menor se produce acidez. Cualquier variación del nivel de *pH* origina un cambio en el sabor característico de cada cerveza.

c. Industria de bebidas gasificadas

Las bebidas gasificadas presentan un valor de *pH* bajo, alrededor de 3 y 4 por ende es un medio desfavorable para el desarrollo de microorganismos. Es importante que el agua empleada para su fabricación no deba salirse de los límites de *pH* entre 6.5 y 9.5 para conservar adecuadamente sus propiedades.

d. Industria lechera

La conservación higiénica de la leche se puede verificar con la medición del *pH*, desde la recolección hasta la entrega. Durante la conservación, el *pH* es determinante para predecir si hay contaminación ocasionada por el amoniaco, usado para conservar el frío en la refrigeración.

- El valor adecuado de pH para la leche debe ser 6.8, si fuese menor es posible una infección del ganado.
- El valor de *pH* para la leche en queso debe encontrarse entre 6.1 y 6.5. Y para la elaboración de los quesos y para su maduración el *pH* debe estar entre 4.1 y 5.3 para disminuir la velocidad de crecimiento de los agentes patógenos.
- El valor de *pH* para el yogurt debe encontrarse entre 4 y 4.4 para una mejor conservación.

e. Control de *pH* en las plantas residuales

Las aguas residuales son los residuos líquidos provenientes de los distintos procesos de la industria. Por lo general, portadoras de productos químicos muy dañinos para la salud y el medio ambiente; por lo que es necesario su tratamiento antes de su evacuación al medio ambiente. Una planta de tratamiento de aguas residuales debe lograr eliminar toda contaminación química y bacteriológica del agua, que pueda ser nociva para los seres humanos, la flora y la fauna; de manera que el agua sea dispuesta en el ambiente en forma segura.

El objetivo de la etapa física es el asentamiento de sólidos pesados, reducción de aceites, grasas y arenas. En la etapa biológica se busca degradar las bacterias y otros microorganismos de las aguas residuales. Es en la última etapa que se utiliza el control de pH como parte de unos estándares aprobados legalmente para la evacuación de esta agua, para ello es necesaria un previo proceso de desinfección.

f. Industria minera

Debido a la continua adición de metales pesados, los procesos de la industria minera son muy contaminantes, pues hacen que el agua no pueda reutilizarse para el riego o evacuación sin previo tratamiento. Para regular la salida a veces se utiliza la neutralización con ácidos y bases y de esta forma poder desecharlos a los ríos sin causar una gran contaminación.

g. Tratamiento de agua para entrada a calderas

El vapor generado en las calderas es utilizado en muchos procesos en los que se necesita calor, por ello es muy útil en todas las plantas industriales.

Algunos requisitos para el buen funcionamiento de las calderas son que el agua de ingreso esté libre de impurezas, y que el valor de *pH* sea neutro, pues al presentarse una predominancia ácida o básica se produciría corrosión o incrustaciones.

Capítulo 2

Descripción de la arquitectura FPGA y lenguaje VHDL

2.1 Descripción de la arquitectura FPGA

2.1.1 Arquitectura FPGA

La arquitectura del FPGA *(Field Programmable Gate Arrays)* está conformada por una matriz de bloques lógicos configurables *(Configurable Logic Blocks CLBs)* interconectados entre sí. Los cuales implementan la lógica generadora de funciones. La Fig. 2.1 muestra la arquitectura de un FPGA de la marca *XILINX*.

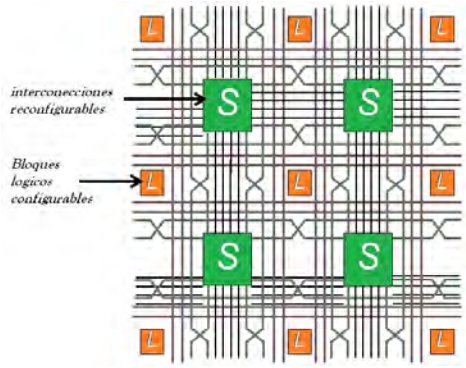


Fig. 2.1 Arquitectura general de un FPGA

Un sistema FPGA contiene cinco elementos principales los cuales son:

Bloques de entrada y salida.

Bloques de lógica configurable.

Bloques de compensación y distribución de reloj.

Bloque de memoria RAM.

Bloques de enlaces o matriz de interconexión.

A continuación se explicará acerca de cada uno de ellos.

a) Bloque de entrada/salida (IOB)

Los **IOB** proveen la interface entre los pines del integrado y la lógica interna. Las interfaces de entrada-salida son otro de los componentes particulares que tienen los FPGA's. Estos están divididos en bancos configurados, para interconectar la lógica de diferentes estándares eléctricos de manera independiente. Estos bancos se configuran aplicando diferentes tensiones de entrada a los pines denominados *Vcc y Vref.*

Las entradas de reloj están asociadas a diferentes bancos de entrada/salida, para permitir que haya diferentes dominios de reloj con interfaces eléctricas diferentes.
La Fig. 2.2 muestrea la configuración de los bloques de entrada y salida.

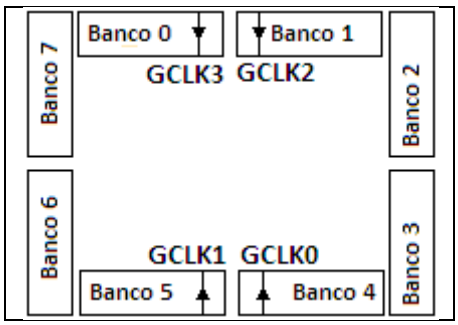


Fig. 2.2 Configuración de los bloques I/O agrupados en bancos

Generalmente los bloques de entrada y salida tienen elementos de almacenamiento integrados, que permiten controlar los tiempos de propagación entre pines del integrado y la lógica interna.

b) Bloque lógico configurable (CLB)

Son bloques básicos que se utilizan en la implementación de un circuito digital. Todas las FPGA tienen algún tipo de CLB. Este es el corazón de la FPGA, y permite implementar las diferentes funciones lógicas. La Fig. 2.3 muestra un **CLB** básico.

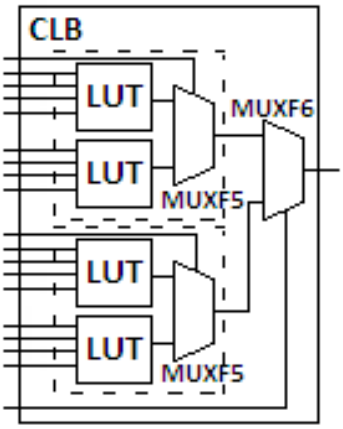


Fig. 2.3 Bloque lógico configurable

Cada ***CLB*** está compuesto por dos bloques iguales denominados ***slice***. En la Fig. 2.3 cada uno contiene dos generadores de funciones ***LUT,*** los cuales son elementos basados en memoria ***RAM*** y un multiplexor ***(MUXF5).***

El multiplexor **MUXF5** combina los resultados de los generadores de funciones dentro de cada ***slice*** del ***CLB.***

Los dos **slice** están unidos por un multiplexor ***MUXF6***, el cual selecciona la salida del ***CLB***. Esto permite implementar cualquier función de 6 entradas, un multiplexor de 8:1 o determinadas funciones lógicas de hasta 19 entradas. Además de poder implementarse lógica combinacional, cada uno contiene recursos para implementar circuitos secuenciales y operaciones aritméticas eficientes.

c) Bloque de distribución y compensación de reloj *(DLL)*

Estos bloques controlan los dominios de reloj dentro del integrado y compensan los retardos que pueda haber entre el reloj externo e interno.

Consiste en bloques de control integrados a la red de distribución de reloj. Esta asegura retardos parejos a todos los bloques lógicos de la FPGA. Cada fabricante utiliza una arquitectura diferente para el control y distribución de reloj.

La Fig. 2.4 muestra la configuración de una unidad de compensación y distribución de reloj de una tarjeta SPARTAN del fabricante *XILINX*.

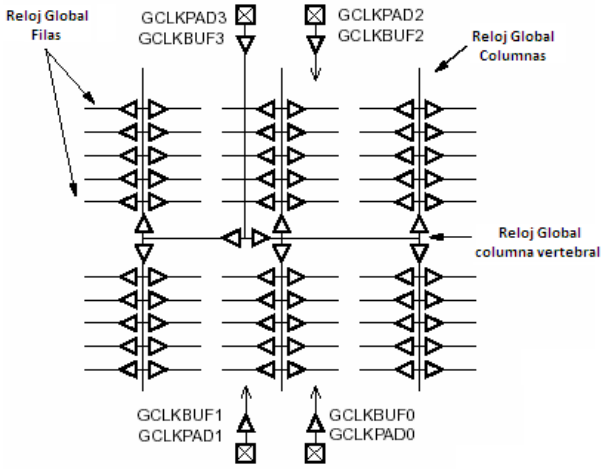


Fig. 2.4 Bloque de compensación y distribución de reloj

La familia *Spartan* tiene bloques específicos para el control de reloj, denominados ***DLL*** *(Delay Locked Loop),* estos bloques sincronizan el reloj interno con el reloj externo del sistema, y controlan el desplazamiento de fase entre los relojes, así también sincronizan los diferentes dominios de reloj asegurando un retardo de distribución del reloj para la lógica interna de la FPGA.

La Fig. 2.5 muestra una unidad de sincronización de reloj.

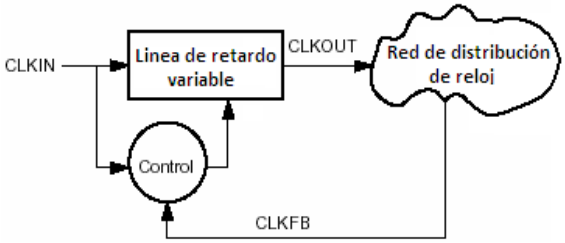


Fig. 2.5 configuración de reloj

Estos bloques permiten controlar la fase y frecuencia del reloj, el sistema de distribución de reloj y el estándar eléctrico de interface entre otras cosas.

d) Bloque de memoria (BLOCK RAM)

Estos bloques son memorias dedicadas, integradas dentro de la lógica programable. La memoria es un componente muy utilizado en diseños digitales.

Varias familias de FPGA contienen bloques de memoria embebida integradas. En general estos bloques básicos de memoria pueden utilizarse en diferentes configuraciones para generar memorias *RAM y ROM* de diferentes tamaños.

La Fig. 2.6 muestra un esquema de una celda de memoria básica de la familia *Spartan.* Este bloque se denomina ***BLOCK RAM*** y es una memoria de puerto doble, es decir, que puede leer y escribir al mismo tiempo.

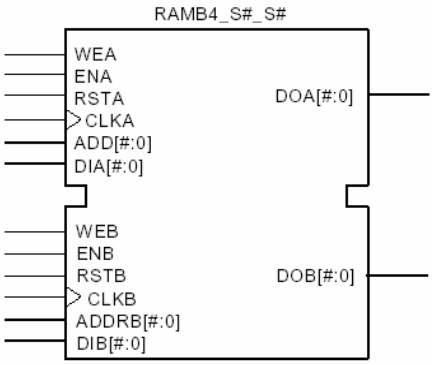


Fig. 2.6 bloque de memoria básica

Para entender la versatilidad de estos bloques de memoria, en la Fig. 2.7 se muestra las opciones de configuración.

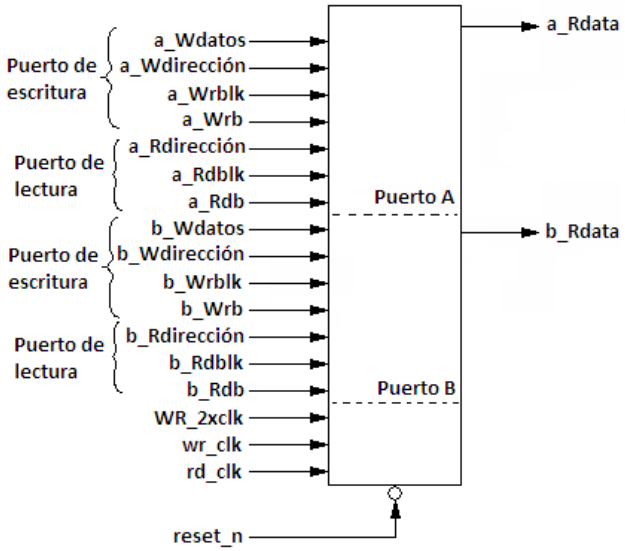


Fig. 2.7 Opciones de configuración de una memoria

e) Enlaces o matriz de interconexión

Es una estructura versátil y multinivel de interconexión entre los componentes de la FPGA.

Para poder implementar los elementos lógicos presentados en las secciones anteriores, no solo se debe configurar adecuadamente, también deben conectarse entre sí. La estructura de interconexión interna de un FPGA consiste en un conjunto de cables que se unen mediante elementos programables.

f) Herramientas de localización e interconexión (*place and route*)

Son las encargadas de decidir en qué elementos lógicos se implementará la lógica diseñada por el usuario, y como deben programarse estas interconexiones, para que el diseño funcione según las especificaciones de tiempo y retardos que se han definido.

Las FPGA contienen recursos de interconexión dedicados a señales de tres estados, de entrada, salida y global para la distribución de reloj y señales específicas.

Los recursos de interconexión local, mostrados en la Fig. 2.8 permiten hacer las conexiones entre los elementos internos de un bloque lógico o *CLB*, como las tablas de búsqueda *(LUT),* los *Flip-Flop.*

Además, provee conexiones a los *GRM* y a los *CLB*. Las conexiones a los *CLB* adyacentes permiten optimizar los diseños al evitar los retardos y la utilización de recursos de la matriz general de inter-conexión.

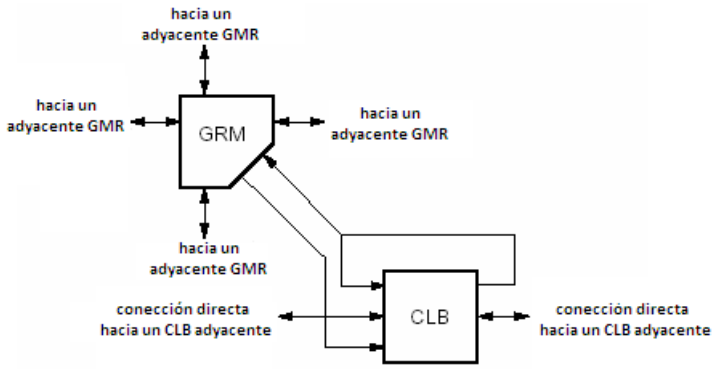


Fig. 2.8 Recursos de interconexión de elementos internos

La mayor parte de las señales se conectarán a través de la matriz general de inter-conexión *(GRM)*.

2.1.2Aplicaciones

Cualquier circuito de aplicación específica puede ser implementado en un FPGA, siempre y cuando esta disponga de los recursos necesarios.

Existen multitud de aplicaciones tanto aeronáutica y defensa, que emplean FPGA's, debido a las buenas características que éstas ofrecen. Así tenemos: misiles guiados, radares y sonares, satélites, dispositivos de exploración, entre otros.

a) Sistemas de imágenes para medicina

Se emplean los FPGA's con más frecuencia para el tratamiento de imágenes biomédicas obtenidas mediante procesos de Tomografía por emisión de positrones PET, escáner CT, rayos X, imágenes tridimensionales, etc.

Requiriendo mayor capacidad de procesamiento, mayor resolución y que puedan desarrollarse en tiempo real. Las FPGA's se adaptan bien a estas necesidades.

b) Sistemas de visión para computadoras

En el mundo actual se cuenta con más cantidad de dispositivos que disponen de un sistema de visión artificial. Ejemplo de esto son las cámaras de video-vigilancia, robots, etc.

Muchos de estos dispositivos precisan de un sistema para conocer posición, reconocer objetos en un entorno, rostros de personas, y poder actuar e interactuar con ellos de forma adecuada. Esta característica requiere el manejo de elevados volúmenes de imágenes, y en la mayoría de ocasiones, en tiempo real.

c) Reconocimiento de voz

El reconocimiento de voz es uno de los objetivos en la relación hombre - máquina. Este es una técnica empleada en seguridad, sistemas de recuperación de información, etc., y se espera que en el futuro su campo de aplicación aumente.

El desarrollo de sistemas que reconocen el habla humana son utilizados para evitar el teclado y el computador es quien se encarga de poner los caracteres en pantalla.

En este contexto, la FPGA resulta muy eficiente a la hora de realizar la comparación de la voz de una persona con unos patrones previamente almacenados.

d) Radioastronomía

La radioastronomía es la ciencia que se encarga de estudiar los objetos lejanos y los fenómenos que ocurren en el universo, mediante la recolección y el análisis de ondas de radio emitidas por esos objetos.

De forma similar a las aplicaciones anteriores, precisa del procesamiento de una gran cantidad de información, donde el FPGA puede aportar todo su potencial.

Cabe notar que el uso en otras áreas es cada vez mayor, sobre todo en aquellas aplicaciones que requieren alta frecuencia de trabajo y un alto grado de paralelismo.

2.1.3 Principales fabricantes

Los principales fabricantes de FPGA's son ***Xilinx*** *y* ***Altera***.

Sin embargo tenemos también:

Lattice Semiconductor, lanzo al mercado un dispositivo FPGA de tecnología de 90nm, con tecnología no volátil, es decir con una ROM incluida, lo que ha sido copiado por distintas compañías inmediatamente.

Actel tiene FPGA's basadas en tecnología Flash reprogramable.

Quick logic tiene productos basados en antifusibles es decir no reprogramables.

Atmel es un fabricante de productos reconfigurables con microcontroladores AVR en un mismo encapsulado.

Achronix Semiconductor desarrolla FPGA's de alta velocidad de hasta 2GHz.

MathStar ofrece lo que ellos llaman FPOA (arreglo de objetos de matriz programable).

2.2 Lenguaje de descripción de hardware VHDL

2.2.1VHDL

VHDL es el acrónimo que representa la combinación de *VHSIC y HDL*, donde *VHSIC* es el acrónimo de *Very High Speed Integrated Circuit y HDL* es a su vez el acrónimo de *Hardware Description Language.*

Es un lenguaje de descripción de hardware para circuitos integrados de alta velocidad, con una sintaxis amplia y flexible. El propósito es especificar y documentar circuitos digitales utilizando un lenguaje formal. Tiene como objetivos el modelado y la síntesis automática.

La síntesis automática transforma de una descripción *VHDL* a un circuito digital llamado ***Netlist***. Es parte de una especificación con un determinado nivel de abstracción, llegando a una implementación más detallada.

Este lenguaje se compone de un conjunto de bloques o módulos, donde cada uno de ellos contiene declaraciones o instrucciones que describen y estructuran el comportamiento del sistema.

Las principales características del lenguaje *VHDL* son:

Especifica los circuitos electrónicos en un formato adecuado para ser interpretado tanto por máquinas como por personas.

Está normalizado, garantizando su compatibilidad con cualquier otra herramienta que respete las indicaciones especificadas en la norma oficial.

Es un lenguaje ejecutable, permite que la descripción textual del hardware se materialice en una representación del mismo.

Lenguaje adaptable a distintas metodologías de diseño y es independiente de la tecnología, lo que permite, en el primer caso, cubrir el tipo de necesidades de los distintos géneros de instituciones, compañías y organizaciones relacionadas con el mundo de la electrónica digital, y en el segundo, facilita la actualización y adaptación de los diseños a los avances de la tecnología en cada momento.

Garantías durante el mantenimiento, en el desarrollo de proyectos, no habrá que sustituir componentes o realizar modificaciones en circuitos desarrollados.

2.2.2Biblioteca *(library)*

La librería contiene las descripciones de todos los elementos que componen el circuito.

Lo que inicialmente es uno o varios ficheros de diseño con la descripción hardware, pasa a ser una única librería de diseño después de la compilación; a esta se le llama *WORK*.

Sirve para compartir diseños previamente compilados sin revelar el contenido.
Los elementos que componen una librería, ver Fig. 2.9, es lo que se llaman unidades, así tenemos, la entidad, la arquitectura, los paquetes, el cuerpos de los paquetes, y las configuraciones.

Al realizar una descripción en *VHDL*, estas unidades se suelen introducir en un mismo fichero, o en varios.

A las unidades de tipo declarativo, que son, la entidad, paquete y configuración, se las conoce como unidades primarias.

A las arquitecturas y cuerpo de los paquetes, se las llama unidades secundarias.

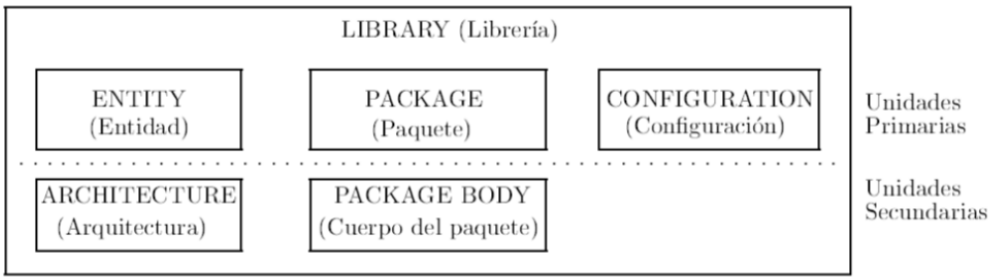


Fig. 2.9 Unidades de la librería del VHDL

Existe un mecanismo para incorporar elementos de otras librerías a nuestro propio diseño.

Este mecanismo es mediante la inclusión al inicio del fichero de diseño de una clausula ***LIBRARY,*** ver Fig. 2.10. Luego de esta sentencia se escribe la lista de bibliotecas que se desea que sean visibles.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
```

Fig. 2.10 asignación de las librerías

La biblioteca ***IEEE*** contiene algunos tipos y funciones complementarias a las que se cargan por defecto. Dentro de esta biblioteca se encuentra la *STD_LOGIC_1164* y *NUMERIC_STD*, estas contienen definición de tipos y funciones para trabajar.

*2.2.3*Entidad *(Entity)*

Una entidad representa la implementación física, define el nombre y la interfaz de entradas y salidas del componente. Por lo tanto tendrá puertos para servir de interface entre las señales externas e interna.

Una entidad también se puede decir que es el circuito esquemático descrito con sentencias. La interfaz es definida por un conjunto de puertos, mientras que la implementación queda oculta al resto del circuito, como un modelo de caja negra.

Se habla de entidades vacías en simulación.

No pueden existir dos entidades con el mismo nombre dentro del diseño.

En la Fig. 2.11 se muestra un ejemplo de asignación de una entidad.

```
entity TOP is
  port(    clk, reset   : in  STD_LOGIC;
           btn_tx       : in  STD_LOGIC;
           btn_rx       : in  STD_LOGIC;
           rx           : in  STD_LOGIC;
           tx           : out STD_LOGIC;
           rx_empty     : out STD_LOGIC;
           tx_full      : out STD_LOGIC;
           o_data       : in  STD_LOGIC_VECTOR(7 downto 0);
           i_data       : out STD_LOGIC_VECTOR(7 downto 0);
           rx_done_tick : out STD_LOGIC;
           tx_done_tick : out STD_LOGIC
           );

end TOP;
```

Fig. 2.11 asignación de la entidad

La primera línea indica el nombre de la entidad, en este caso TOP. Las siguientes líneas especifican las entradas y salidas del Puerto. El formato básico para declarar una entrada o salida es: **nombre : modo tipo**

nombre : Definido por el usuario, representa a la entidad.

modo : Indica si la señal es de entrada, salida o ambos.

tipo : indica si es un bit, un vector, carácter u entero.

La última línea indica el fin de la entidad. La entidad describe al circuito o proyecto visto externamente.

2.2.4Paquete *(Package)*

Un paquete es un conjunto de declaraciones, constantes, tipos de datos, subprogramas, para luego ser utilizados por más de un diseño o entidad. Reúne un grupo de declaraciones relacionadas y es compilada por separado. Estos constan de dos partes, las declaraciones y el cuerpo del paquete *(PACKAGE y PACKAGE BODY).*

La declaración *(PACKAGE)* debe incluir información de una o más entidades del diseño. Como declaración de tipo de datos, señales, subprogramas y componentes. El cuerpo del paquete *(PACKAGE BODY)* incluye la implementación de los subprogramas declarados en el *PACKAGE*. A continuación, Fig. 2.12, se muestra la declaración de paquetes y de cuerpos de los paquetes:

```
-- Declaracion de paquete
PACKAGE nombre IS
declaraciones
END nombre;


-- Declaracion del cuerpo
PACKAGE BODY nombre IS
declaraciones, instrucciones, etc.
END nombre;
```

Fig. 2.12 Declaración del Package

El nombre del *PACKAGE* y de *PACKAGE BODY* debe coincidir.

A continuación, en la Fig. 2.13, se muestra un ejemplo de este tipo de declaraciones, donde al principio se declaran unos tipos y cabeceras de función, y posteriormente se definen las funciones en un *PACKAGE BODY*.

```
PACKAGE pack_ej IS
   SUBTYPE nombre IS bit_vector(23 DOWNTO 1);
   SUBTYPE edad IS bit_vector(7 DOWNTO 0);
   CONSTANT inicio: nombre;
   FUNCTION edad_int(valor: edad) RETURN integer;
   FUNCTION int_edad(valor: integer) RETURN edad;
END pack_ej;


PACKAGE BODY pack_ej IS
   CONSTANT inicio: nombre:=X"FFF000";
   FUNCTION edad_int(valor: edad) RETURN integer IS
      --cuerpo de función
   END edad_int;
   FUNCTION int_edad(valor: integer) RETURN edad IS
      --cuerpo de función
   END int_edad;
END pack_ej;
```

Fig. 2.13 Declaración del *Package* y *Package Body*

2.2.5Arquitectura *(Arquitecture)*

Describe el funcionamiento de la entidad a la que está asociada. Pueden existir varias arquitecturas pero tan sólo una entidad, para elegir con cuál de esas arquitecturas se va a trabajar lo determinará la configuración.

En una arquitectura se realizan las operaciones con las señales, es decir se implementa el Hardware.

Se compone de dos partes:

La declaración, donde se incluyen las señales, variables y componentes que se va a implementar.

El cuerpo, donde se incluye la implementación de la entidad por medio de instrucciones ó sentencias. La Fig. 2.14 se describe la arquitectura y sus partes.

```
architecture Behavioral of spi_clk is
signal count1     : std_logic_vector(5 downto 0):=("000000");  --declaración de señales
                                                               --componentes y nuevos t

begin
process(CLK,RST)
 begin
   if RST='1' then
      count1<="000000" ;
   elsif CLK='1' and CLK'event then
      if count1<="001111" then
      START_ADC<='1';
      START_DAC<='1';
      elsif count1>"010000" and count1<="110000" then          -- cuerpo del programa
      START_ADC<='0';                                          --incluyen instrucciones
      START_DAC<='0';
         if count1<="110000" then
            count1<="000000";
         end if;
      end if;
      count1<=count1 + 1 ;
   end if;
end process;


end Behavioral;
```

Fig. 2.14 Declaración de la Arquitectura

En general, dentro de la arquitectura en VHDL la declaracion del modelo se puede realizar utilizando los siguientes estilos de programación:

a. Flujo de datos *(Data Flow)*

Esta posibilidad de descripción, refiriéndose al nivel de abstracción, se conoce como descripción a nivel de transferencia entre registros *RTL (Register Transfer Level).*

VHDL es un lenguaje concurrente, como consecuencia no seguirá el orden en que están escritas las instrucciones a la hora de ejecutar el código. De hecho, si hay dos instrucciones, no tiene porqué ejecutarse una antes que otra, pueden ejecutarse a la vez.

En lenguajes como *C* la ejecución de las sentencias se realizan en serie. Estas son ejecutadas una tras otra. Para especificar la funcionalidad de un sistema que permita paralelismo, es necesario utilizar un lenguaje que permita una especificación concurrente o paralela, de sus instrucciones, siendo esta ejecución serie un caso particular de la ejecución concurrente.

A continuación se muestra en la Fig. 2.15 un ejemplo de esta arquitectura.

```
entity mux is

   port(a,b             : in  STD_LOGIC_VECTOR (10 DOWNTO 0);
        asss,kess,ksas  : out STD_LOGIC);
end mux;

architecture Behavioral of mux is
begin

asss <= '1' when a>b else '0';
kess <= '1' when a=b else '0';
ksas <= '1' whwn a<b else '0';

end Behavioral;
```

Fig. 2.15 Ejemplo usando la arquitectura Data Flow

El ejemplo de la Fig. 2.15 es una descripción *RTL* de un comparador de 2 buses (a y b), ambos de 11 bits. Este tiene tres *salidas (asss, kess, ksas),* las cuales solo una de ellas se pondrá en 1 lógico, tenindo en cuenta las siguientes funciones lógicas a>b, a=b ó a<b.

Estas instrucciones son ejecutadas de forma concurrente. Si alguna de ellas cambia entonces se ejecutan las instrucciones que se ven afectadas. Por lo tanto las tes instrucciones serán ejecutadas de forma paralela.

Se usarán las siguientes sentencias: ***when***, ***else***, ***with***, ***select***, propias del estilo Flujo de datos.

b. Comportamental *(Behavioral)*

Este tipo de arquitectura permite una programación en serie o algorítmica. Describe el algoritmo que refleja la funcionalidad o el comportamiento de dicho componente de manera secuencial.

En *VHDL* las sentencias secuenciales se encuentran dentro de los llamados procesos, con la palabra clave *PROCESS*. Por lo tanto cada vez que se requiera de una descripción en serie, se deberá utilizar un bloque de este tipo.

Los procesos son ejecutados en paralelo entre sí, mientras que internamente la ejecución de las instrucciones son en serie.

Dentro de este bloque puede haber muchas instrucciones, asignaciones, condiciones, bucles.

La activación o ejecución del bloque *PROCESS* se da de dos maneras. Cuando uno de los argumentos que interviene en la lista sensible cambia. La otra opción consiste en utilizar una sentencia *WAIT* en algún lugar dentro de este bloque.

A continuación en la Fig. 2.16 se muestra un ejemplo de una descripción comportamental de un comparador.

```
architecture Behavioral of comparator is
begin
   process(a,b)
   begin
      ross <= '0';
      anna <= '0';
      karl <= '0';
      if a>b then
         ross <= '1';
      elsif a<b then
         karl <= '1';
      elsif a<b then
         anna <= '1';
      end if ;
   end process;
end Behavioral;
```

Fig. 2.16 Ejemplo usando la arquitectura Behavioral

c. Estructural *(Structural)*

Esta descripción se basa en modelos lógicos ya establecidos como restadores, sumadores, compuertas, ente otros. Define componentes e interconexiones y resulta muy útil para aprovechar.

Permite la incorporación al diseño de elementos de biblioteca, realización de diseños jerárquicos a partir de otros componentes. Esta descripción incluye la definición de la interfase entre unos elementos y otros.

Definición de componentes

En el lenguaje *VHDL* es necesario declarar el componente asi como la interconexiones entre ellos. Es necesario conectar estos con la entidad correspondiente. La Fig. 2.17 hace referencia a la definición de un componente.

```
component llaveand
   port(a ,b : in bit;
            c : out bit);
end component llaveand;
```

Fig. 2.17 declaración de componentes

Referencia de componentes

Copia un componente de la arquitectura y permite referenciarlo ó instanciarlo tantas veces sea necesario. Es el nombre con que se le da a la copia particular; es posible reproducir un componente, una entidad o una configuración.

De forma parecida que en la entidad, los componentes son declarados con la palabra ***COMPONET.*** Estacláusula obliga al componente al que se referencia esté asociado a una entidad. Las cláusula ***GENERIC MAP*** y ***PORT MAP*** especifican la definición de los parámetros así como el paso de los puertos de conexión. La Fig. 2.18 hace referencia del componente mux_1.

```
mux_1 : mux
       GENERIC MAP (
         C_AWIDTH => C_AWIDTH, C_DWIDTH => C_DWIDTH)
       PORT MAP (
         control => ctrl, entrada1 => e1, entrada2 => e2, salida => sal );
```

Fig. 2.18 Referencia de componentes

Repetición de estructuras

El hardware muchas veces está formado por estructuras que se repiten. La declaración de ***GENERATE*** crea múltiples copias de componentes, procesos o bloques, tantas veces como se le indique.

Esta sentencia es concurrente, por lo que no se debe usar dentro de un *process*. En el siguiente ejemplo siguiente muestra la estructura de la declaración. Se tiene el componente ***COMP***, el cual tiene una entrada *X* y una salida Y de tipo *bit*. La función GENERATE crea un conjunto de estos que se conectan a los elementos correspondientes de *A y B*. La función ***'range*** indica a ***I*** tome valores desde 0 hasta 7, el cual es el rango de *A,* ver Fig. 2.19.

```
component COMP
   port (X : in bit;
         Y : out bit);
end component;

signal A, B: BIT_VECTOR(0 to 7);

GEN: for I in A'range generate
U: COMP port map (X => A(I), Y => B(I));
end generate GEN;
```

Fig. 2.19 Declaración de la sentencia Generate

La Fig. 2.20 muestra en detalle el hardware generado.

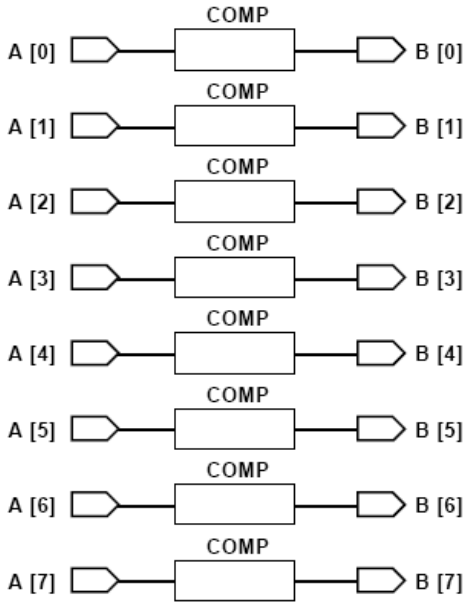


Fig. 2.20 Hardware generado con la sentencia Generate

2.2.6Tipos de datos

La asignación de valores a un objeto debe tomar en cuenta el tipo de dato sobre el cual está definido.

En el paquete ***STD*** de la librería ***STANDARD*** predefine una serie de tipos de datos.
Los tipos de datos se diferencian entre sí por el conjunto de valores que se definen en cada tipo y por las operaciones que se pueden realizar con ellos. En la **Tabla 2.1** se describe los tipos de datos.

Tabla 2.1 Descripción de tipos de datos

Tipo	Escalar o Matricial	Ejemplo	Comentarios
Bit	escalar	'0', '1'	Corresponden a los niveles lógicos 0 y 1
Std_logic	escalar	'U', 'X', '0', '1', 'Z', 'W', 'L', 'H', '-'	No inicializado, Valor desconocido, Nivel digital '0', Nivel digital '1', Alta impedancia, Señal digital desconocida Señal digital '0' Señal digital '1' Señal sin importancia
boolean	escalar	'true',	Verdadero

		'false'	falso
character	escalar	1, -245, 16#2ª3E#, 8#3477#	Enteros en base 10 Enteros en base 16 Enteros en base 8
Real	escalar	-1.0, 2.0E+38	Números con punto flotante
Time	Escalar	1 ns, 10 us, 1000 ms, 2 sec	Nanosegundos, microsegundos Milisegundos, segundos
String	Matricial	“start bit”	Secuencia de caracteres
Bit_vector	matricial	“00110000” x“A32A”	Vector de 8 bits Vector de 16 bits en hexadecimal
Std_logic_vector	matricial	“00Z1”	Vector de señales

En la **Tabla 2.2** se describe los tipos de objetos.

Tabla 2.2 Descripción de tipos de objetos

Objeto	Ejemplo	Comentarios
Variable	**variable x : integer;**	**Define una variable computacional**
Sigal	**signal x: integer;**	**Define una señal eléctrica**
Type e array	**Type rom3x3 is array (0 to 2, 0 to2) of std_logic;**	**Define un tipo de señal std_logic de tipo matricial**
Range	**D : in integer range 0 to 255;**	**Restringe una señal entre dos valores**
Constant	**constant PI : real := 3.14159;**	**Define una constante**

En la Fig. 2.21 se muestra un ejemplo de asignación de objetos y su tipo.

```
signal   contador : integer range 1 to 100;
variable retardo  : time range 10 ns to 500 ns;
variable reset    : bit:= '0';
signal   barramento_dados: std_logic_vector (7 downto 0);
type     rom2x3 is array (0 to 2, 0 to 1) of std_logic;
constant mem_rom: rom2x3:= ( ('0','1','0'),
                             ('1','1','0'));
variable start: bit;
```

Fig. 2.21 Asignación de objetos y tipo

2.2.7Operadores

El lenguaje *VHDL* permite la definición de operadores por el usuario y la sobrecarga de nombres, es decir, permite que dos operadores compartan el mismo nombre si se diferencian en el tipo de datos con que operan.

La **Tabla 2.3** muestra una relación de operadores y tipos de datos definidos en la librería ***IEEE STD_LOGIC_1164 PACKAGE***

Tabla 2.3 Relación de operadores y tipos de datos

operator	description	Tipo de datos y operadores	Tipo de datos y operadores
a * * b a * b a / b a + b a - b	exponentiatión multiplicatión division addition substraction	integer	integer
a & b	concatenation	1-D array, element	1-D array
a = b	equal to	Any scalar or 1-D array	boolean
a /= b	not equal to		
a < b	less than		
a <= b	less than or equal to		
a > b	greater than		
a >= b	greater than or equal to		
not a	negation	Boolean Std_logic Std_logic_vector	Same as operand
a and b	and		
a or b	or		
a xor b	xor		

La **Tabla 2.4** muestra una relación de operadores y tipos de datos definidos en la librería ***IEEE NUMERIC_STD PACKAGE***

Tabla 2.4 Operadores y tipos de datos definidos en librería IEEE Numeric_std_package

Overloaded operator	Description	Data type of operands	Data types of result
a * b a + b a - b	arithmetic operation	unsigned, natural, Signed, integer	unsigned signed
a = b a /= b a < b a <= b a > b a >= b	relational operation	unsigned, natural, signed, integer	boolean

La **Tabla 2.5** muestra la relación de operandos para convertir datos de tipo ***STD_LOGIC_VECTOR and NUMERIC***.

Tabla 2.5 Conversión de datos entre std_logic_vector y tipo numérico

Data type of a	To data type	Conversión function/type casting
unsigned, signed	Std_logic_vector	std_logic_vector(a)
signed, std_logic_vector	unsigned	unsigned(a)
unsigned, std_logic_vector	signed	Signed(a)
unsigned, signed	integer	to_integer(a)
natural	unsigned	to_unsigned(a, size)
integer	signed	to_signed(a, size)

En el estándar *VHDL*, las operaciones aritméticas se definen por el tipo de datos entero y para el tipo de datos natural, que es un subtipo de entero que contiene cero y positivo enteros.

A continuación se describe algunos de ellos.

a) Operador de concatenación

El operando de concatenación ***(&)*** tiene como matriz resultante la suma de las dimensiones de las matrices sobre las que opera: *A <= B&C;*

b) Operadores aritméticos

***** (Exponencial)*** Sirve para elevar un número a una potencia determinada: 4**2 es equivalente a 4^2. El número de la base puede ser entero o real, pero el exponente debe de ser de valor entero.

ABS( ) (Valor absoluto) Devuelve el valor absoluto de su argumento que puede ser de cualquier tipo numérico.

**** (Multiplicación)*** Sirve para multiplicar dos números de tipo numérico.

/ (División) Sirve para dividir dos números de tipo numérico.

+ ***(Suma y signo positivo)*** Este operador sirve para indicar suma si va entre dos operandos, también indica el signo si va al inicio de una expresión. Opera sobre valores numéricos de cualquier tipo.

- ***(Resta y signo negativo)*** Este operador sirve para indicar sustracción si va entre dos operandos, también indica el signo si va al inicio de una expresión. Opera sobre valores numéricos de cualquier tipo.

c) Operadores de desplazamiento

SLL, SRL Desplazamiento lógico de un vector, rellena con ceros los huecos libres. SLL desplaza a la izquierda, mientras SRL a la derecha.

SLA, SRA Desplazamiento aritmético a izquierda y derecha.

ROL, ROR Rotación a izquierda y a derecha. Es como el de desplazamiento pero los huecos son ocupados por los bits que van quedando fuera.

d) Operadores relacionales

=, /= *(igualdad)* Devuelve *TRUE* si los operandos son iguales y *FALSE* en caso contrario respectivamente.
<,<=,>,>= *(menor mayor)* Los datos que pueden manejar son siempre de tipo escalar o matrices de una sola dimensión de tipos discretos. Devuelven *TRUE o FALSE*. Ver Fig. 2.22.

```
A<= 2 ;
B<= A + C ;
A<= D + 1 ;
E<= A*2 ;
```

Fig. 2.22 Operadores relacionales

e) Operadores lógicos

NOT, AND, NAND, OR, NOR y XOR. Actúan sobre los tipos bit, *bit_vector* y *boolean*. La operación se realizará bit a bit en el caso de utilizar este tipo de operadores en un vector. Ver Fig. 2.23.

```
z := x * ( -y)
z := x / (not y)
```

Fig. 2.23 Operadores lógicos

2.2.8Declaración de constantes, variable y señal

Las variables y las constantes tienen el mismo significado que en cualquier otro lenguaje. Mientras que las señales representan conexiones reales en el circuito.

Las variables son elementos abstractos y son asignadas dentro de un proceso *(PROCESS)* o subprograma, donde las sentencias son ejecutadas en serie. Toma el valor que se le asigna y puede ser alterado en cualquier instante, ver Fig. 2.24.

```
VARIABLE contador: natural := 0;
VARIABLE selector: bit_vector(31 DOWNTO 0);
```

Fig. 2.24 Declaración de variables

Las señales pueden ser declaradas únicamente en las arquitecturas, paquetes *(PACKAGE),* o en los bloques concurrentes *(BLOCK)* y pueden ser usadas en cualquier parte del programa y son visibles por todos los procesos y bloques dentro de una arquitectura, representan interconexiones entre bloques dentro de la arquitectura.

Las entradas y salidas declaradas en la entidad son consideradas señales, ver Fig. 2.25, cabe mencionar que las salidas no pueden ser leidas, no pueden formar parte de una lista sensible; a la vez una entrada no se le puede asignar un valor en la descripción.

```
SIGNAL selec: std_logic := '0';
SIGNAL datos: std_logic_vector(7 DOWNTO 0);
```

Fig. 2.25 Declaración de señales

Las constantes son declaradas en los mismos lugar que las variables y señales, ver Fig. 2.26. Es un elemento que se inicializa en un determinado valor y no puede ser alterado, conserva su valor.

```
CONSTANT e: real := 2.71828;
CONSTANT retraso: time := 10 ns;
```

Fig. 2.26 Declaración de constantes

Capítulo 3

Estrategia de identificación y control

3.1 Identificación de Sistemas

La identificación es una herramienta que ayuda a encontrar el modelo matemático de un proceso mediante una serie de experimentos de entrada y salida, sin necesidad de conocer las leyes internas que gobiernan el comportamiento del sistema, ver Fig. 3.1. Por esta razón, los modelos obtenidos mediante identificación se denominan modelos de "caja negra".

Básicamente, un modelo es una herramienta que permite predecir el comportamiento de un sistema sin necesidad de experimentar sobre él.

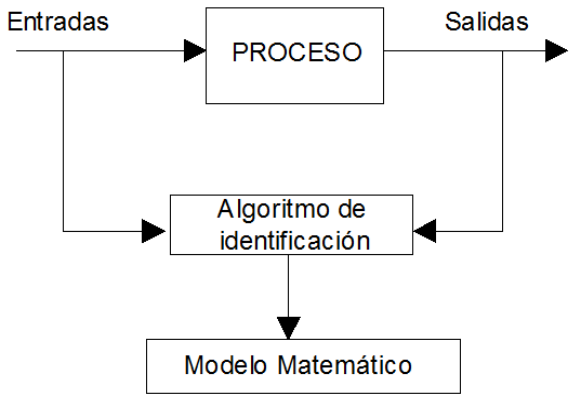


Fig. 3.1 Modelo de identificación de sistemas

El conocimiento del modelo es necesario para el diseño e implementación de un sistema de control. Para lograr tal objetivo se realizan los siguientes pasos:

1. Toma de datos experimentales.

2. Procesado y análisis.

3. Selección del modelo y estimación de parámetros.

4. Validación y aceptación.

3.1.1 Modelación

En muchas aplicaciones son útiles modelos descriptivos de sistemas dinámicos. Básicamente hay dos campos de construcción de modelos:

- **Modelación matemática**: Este es un enfoque analítico. Las ecuaciones del modelo son obtenidas a partir del conocimiento de las leyes físicas y químicas que gobiernan el comportamiento del proceso.

- **Identificación de sistemas**: Este es un enfoque experimental. El modelo es obtenido a partir de un grupo de datos experimentales recolectados de una prueba en la planta, es decir, es un método que no se basa en leyes físicas, sino en experiencias.

Los modelos obtenidos mediante identificación tienen las siguientes propiedades en contraste con los modelos basados solamente en modelación matemática:

- Tiene limitada validez (son válidos para sistemas lineales y en un punto de trabajo).

- Dan una pequeña intuición física, en muchos casos los parámetros del modelo no

- tienen significado físico directo.

- Son relativamente fáciles de construir y usar.

3.1.2 Métodos de identificación

Considerando las características de los modelos que se pretenden obtener, se establecen los siguientes métodos de identificación:

- Métodos de identificación de modelos no paramétricos.

- Métodos de identificación de modelos paramétricos.

A. Métodos de identificación de modelos no paramétricos

Estos métodos tienen el objetivo de obtener curvas o gráficos que caracterizan el comportamiento dinámico del proceso. Son además basados en distintos criterios de análisis entre los cuales están los siguientes:

Análisis transitorio: La entrada es un escalón o un impulso y los datos de salida constituyen el modelo.

Análisis en frecuencia: La entrada es sinusoidal. Para un sistema lineal en estado estacionario la salida será también sinusoidal. Los cambios de amplitud y de fase darán la respuesta en frecuencia.

Análisis de correlación: La entrada es un ruido blanco. Una función normalizada de covarianza cruzada entre la salida y entrada proporciona una estimación de las funciones de peso.

Análisis espectral: La respuesta en frecuencia puede ser estimada para una entrada arbitraria mediante visión del espectro cruzado entre la salida y la entrada.

Los métodos de identificación no paramétricos estudiados, se enfocan a la obtención de funciones transferencia de primer y segundo orden.

B. Métodos de identificación de modelos paramétricos

Los modelos paramétricos, a diferencia de los anteriores, quedan descritos mediante una estructura y un número finito de parámetros que relacionan las señales de interés del sistema (entradas, salida y perturbaciones).

En muchas ocasiones es necesario realizar la identificación de un sistema del cual no se tiene ningún tipo de conocimiento previo. En estos casos, se suele recurrir a modelos estándar, cuya validez para un amplio rango de sistemas dinámicos ha sido comprobada experimentalmente.

Generalmente estos modelos permiten describir el comportamiento de cualquier sistema lineal. La dificultad radica en la elección del tipo de modelo (orden del mismo, número de parámetros, etc.) que se ajuste satisfactoriamente a los datos de entrada y salida obtenidos experimentalmente.

a. Modelos paramétricos

Generalmente los modelos paramétricos se describen en el dominio discreto, puesto que los datos requeridos para la identificación se obtienen por muestreo. En el caso de que se requiera un modelo continuo, siempre es posible realizar una transformación del dominio discreto al continuo.

La forma general de la estructura de un modelo paramétrico en discreto es la siguiente:

$$y(t) = G(q^{-1}, \theta)u(t) + H(q^{-1}, \theta)e(t) \tag{3.1}$$

donde:

$y(t)$ es la salida del sistema.

$u(t)$ es la entrada del sistema.

$e(t)$ es un ruido blanco no medible.

G y H son funciones de transferencia en el operador de retardo q, el cual está definido en forma genérica como.

$$\begin{aligned} q^{-1}f(t) &= f(t-1) \\ f(t+1) &= qf(t) \end{aligned} \tag{3.2}$$

θ es el vector paramétrico que contiene los coeficientes de G y H

El modelo de la ecuación (3.1) es representado en la Fig. 3.2.

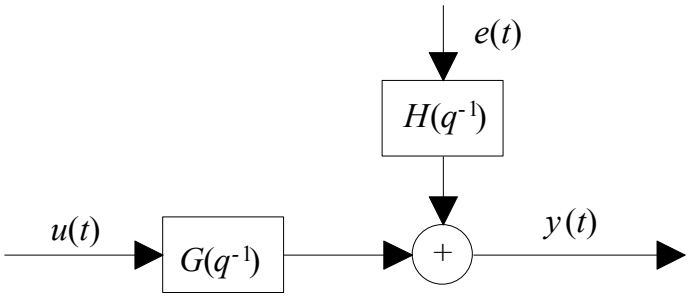


Fig. 3.2 Diagrama de bloques de un modelo general.

$G(q^{-1},\theta)$ y $H(q^{-1},\theta)$ son funciones del vector paramétrico θ. En muchos estas son de orden finito.

A continuación describiremos las estructuras de modelos típicos especificando la parametrización.

Para un caso general, se puede escribir la ecuación (3.3) como:

$$y(t) = \frac{B(q^{-1})}{A(q^{-1})F(q^{-1})}u(t) + \frac{C(q^{-1})}{A(q^{-1})D(q^{-1})}e(t) \tag{3.3}$$

Donde:

$$G(q^{-1}) = \frac{B(q^{-1})}{A(q^{-1})F(q^{-1})} \quad \text{y} \quad H(q^{-1}) = \frac{C(q^{-1})}{A(q^{-1})D(q^{-1})} \tag{3.4}$$

con

$$\begin{aligned} A(q^{-1}) &= 1 + a_1q^{-1} + a_2q^{-2} + \ldots + a_{na}q^{-na} \\ B(q^{-1}) &= b_1q^{-1} + b_2q^{-2} \ldots + b_{nb}q^{-nb} \\ C(q^{-1}) &= 1 + c_1q^{-1} + c_2q^{-2} + \ldots + c_{nc}q^{-nc} \\ D(q^{-1}) &= 1 + d_1q^{-1} + d_2q^{-2} + \ldots + c_{nd}q^{-nd} \\ F(q^{-1}) &= 1 + f_1q^{-1} + f_2q^{-2} + \ldots + c_{nf}q^{-nf} \end{aligned} \tag{3.5}$$

El vector paramétrico θ contiene los coeficientes a_i, b_i, c_i, d_i y f_i de las funciones de transferencia anteriores.

$$\theta = [a_1 \ldots . a_{na} \quad b_1 \ldots . b_{nb} \quad c_1 \ldots . c_{nc} \quad d_1 \ldots . d_{nd} \quad \ldots \quad f_1 \ldots . f_{nf}\,] \tag{3.6}$$

Para elegir la estructura de este tipo de modelos hay que determinar el orden de cada uno de los polinomios anteriores, es decir na, nb, nc, nd, nf y el retardo entre la entrada y la salida nk.

Una vez elegidos estos valores, sólo queda determinar el vector de coeficientes (a_i, b_i, c_i, d_i y f_i) que hacen que el modelo se ajuste a los datos de entrada-salida del sistema real.

A continuación se presenta algunas estructuras particulares, ver Tabla 3.1.

Tabla 3.1 Estructuras de modelos paramétricos

Tipo de modelo	***Condición***	***Estructura resultante***
ARX	$\mathrm{F}(\mathrm{q}^{-1}) = \mathrm{D}(\mathrm{q}^{-1}) = \mathrm{C}(\mathrm{q}^{-1}) = 1$	$A(\mathrm{q}^{-1})y(t) = \mathrm{B}(\mathrm{q}^{-1})u(t) + e(t)$
Output Error (OE)	$\mathrm{C}(\mathrm{q}^{-1}) = \mathrm{D}(\mathrm{q}^{-1}) = \mathrm{A}(\mathrm{q}^{-1}) = 1$	$y(t) = \frac{\mathrm{B}(\mathrm{q}^{-1})}{\mathrm{F}(\mathrm{q}^{-1})}u(t) + e(t)$
Armax	$\mathrm{F}(\mathrm{q}^{-1}) = \mathrm{D}(\mathrm{q}^{-1}) = 1$	$A(\mathrm{q}^{-1})y(t) = \mathrm{B}(\mathrm{q}^{-1})u(t) + \mathrm{C}(\mathrm{q}^{-1})e(t)$
Box Jenkins (BJ)	$\mathrm{A}(\mathrm{q}^{-1}) = 1$	$y(t) = \frac{\mathrm{B}(\mathrm{q}^{-1})}{\mathrm{F}(\mathrm{q}^{-1})}u(t) + \frac{\mathrm{C}(\mathrm{q}^{-1})}{\mathrm{D}(\mathrm{q}^{-1})}e(t)$
PEM	$\mathrm{A}(\mathrm{q}^{-1}) \neq 1$	$y(t) = \frac{\mathrm{B}(\mathrm{q}^{-1})}{\mathrm{F}(\mathrm{q}^{-1})\mathrm{A}(\mathrm{q}^{-1})}u(t) + \frac{\mathrm{C}(\mathrm{q}^{-1})}{\mathrm{D}(\mathrm{q}^{-1})\mathrm{A}(\mathrm{q}^{-1})}e(t)$

Los diagramas de bloques equivalentes para cada uno de los modelos anteriores son:

Estructura ARX:

$$y = (\mathrm{B}u + e)/\mathrm{A} \tag{3.6}$$

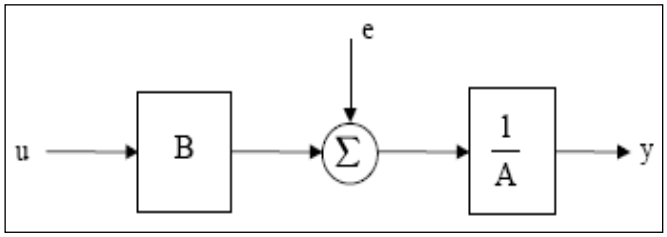


Fig. 3.3 Estructura ARX

$$A(q^{-1}) = 1 + a_1 q^{-1} + a_2 q^{-2} + \ldots + a_{na} q^{-na}$$
$$B(q^{-1}) = b_1 q^{-1} + b_2 q^{-2} \ldots + b_{nb} q^{-nb} \tag{3.7}$$

Estructura OE:

$$y = \mathrm{B}u/\mathrm{A} + e \tag{3.8}$$

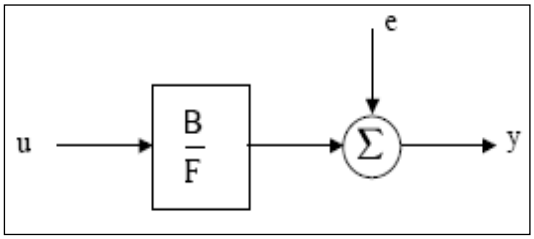


Fig. 3.4 Estructura OE

Donde:

$$A(q^{-1}) = 1 + a_1 q^{-1} + a_2 q^{-2} + \ldots + a_{na} q^{-na}$$
$$B(q^{-1}) = b_1 q^{-1} + b_2 q^{-2} \ldots + b_{nb} q^{-nb} \tag{3.8}$$

Estructura ARMAX:

$$y = (\mathrm{B}u + \mathrm{C}e)/\mathrm{A} \tag{3.9}$$

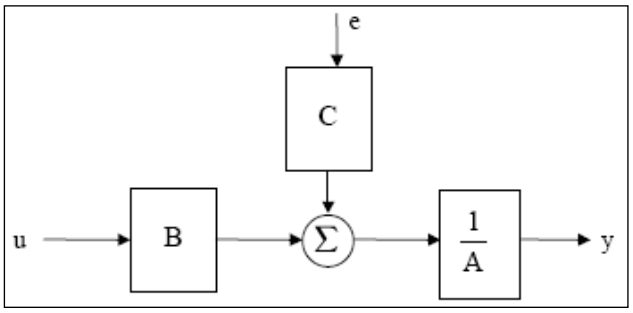


Fig. 3.5 Estructura ARMAX

Donde:

$$A(q^{-1}) = 1 + a_1 q^{-1} + a_2 q^{-2} + \ldots + a_{na} q^{-na}$$
$$B(q^{-1}) = b_1 q^{-1} + b_2 q^{-2} \ldots + b_{nb} q^{-nb} \tag{3.10}$$
$$C(q^{-1}) = 1 + c_1 q^{-1} + c_2 q^{-2} + \ldots + c_{nc} q^{-nc}$$

Estructura BJ:

$$y = \mathrm{B}u/\mathrm{A} + \mathrm{C}e/\mathrm{D} \tag{3.11}$$

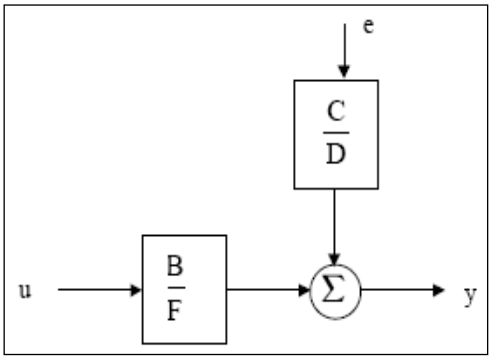


Fig. 3.6 Estructura BJ

Donde:

$$\begin{aligned} A(q^{-1}) &= 1 + a_1 q^{-1} + a_2 q^{-2} + \ldots + a_{na} q^{-na} \\ B(q^{-1}) &= b_1 q^{-1} + b_2 q^{-2} \ldots + b_{nb} q^{-nb} \\ C(q^{-1}) &= 1 + c_1 q^{-1} + c_2 q^{-2} + \ldots + c_{nc} q^{-nc} \\ D(q^{-1}) &= 1 + d_1 q^{-1} + d_2 q^{-2} + \ldots + c_{nd} q^{-nd} \end{aligned} \tag{3.12}$$

Estructura PEM

$$y = \mathrm{B}u/\mathrm{FA} + \mathrm{C}e/\mathrm{DA} \tag{3.13}$$

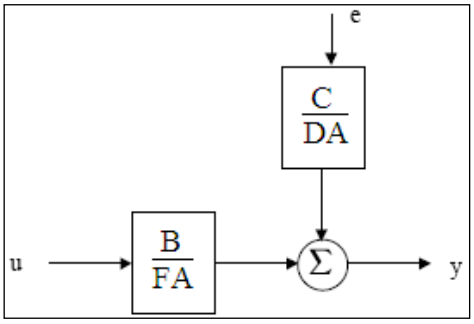


Fig. 3.7 Estructura PEM

Donde:

$$\begin{aligned} A(q^{-1}) &= 1 + a_1 q^{-1} + a_2 q^{-2} + \ldots + a_{na} q^{-na} \\ B(q^{-1}) &= b_1 q^{-1} + b_2 q^{-2} \ldots + b_{nb} q^{-nb} \\ C(q^{-1}) &= 1 + c_1 q^{-1} + c_2 q^{-2} + \ldots + c_{nc} q^{-nc} \end{aligned} \tag{3.14}$$

$$D(q^{-1}) = 1 + d_1 q^{-1} + d_2 q^{-2} + \ldots + c_{nd} q^{-nd}$$
$$F(q^{-1}) = 1 + f_1 q^{-1} + f_2 q^{-2} + \ldots + c_{nf} q^{-nf}$$

3.1.3 Validación del modelo

La validación de los modelos calculados es uno de los pasos esenciales en un proceso de identificación. Luego que la fase de estimación de parámetros ha sido realizada, tenemos que validar el modelo obtenido.

Validación significa evaluar un grado de confidencia de nuestro modelo. Esto puede ser muy impreciso y, en realidad, no existe una medida de validación definitiva.

En lugar de una prueba simple, validar el modelo involucra analizar la respuesta del modelo bajo diferentes puntos de vista. Es este grupo de pruebas y la coherencia de los resultados obtenidos, es cual nos permite aceptar el modelo. Hay varias formas de probar un modelo, de los cuales tenemos:

- Respuestas cualitativas del modelo.
- Validación cruzada.
- Índices de performance.

A continuación veremos en forma breve cada uno de estos métodos.

Respuestas cualitativas
Una forma natural de probar la respuesta de modelo, es ver su respuesta al paso o impulso y hacer juicios basados en las características conocidas del proceso. El modelo debe proporcionar respuestas lógicas similares a las esperadas del proceso, ver Fig. 3.8.

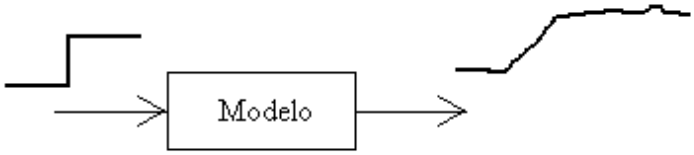


Fig. 3.8 Modelo y respuesta a entrada escalón

Podemos verificar en relación a la respuesta esperada del proceso, el orden de magnitud de varios parámetros del modelo, tales como, la ganancia, tiempo de retraso, tiempo de establecimiento, etc.

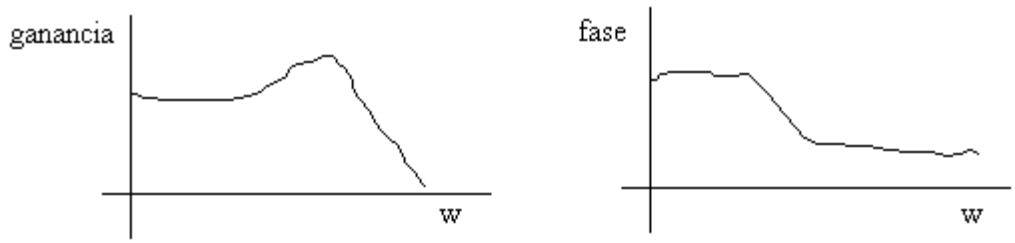


Fig. 3.9 Evaluación cualitativa del modelo

La repuesta del modelo también puede ser analizada en el dominio de la frecuencia. Los diagramas de *Bode* y *Nyquist* permiten ver otras características del modelo, Fig. 3.9, tales como rechazo al ruido, velocidad de respuesta, resonancia, etc. que pueden ser observados a través del ancho de banda, pendiente en las altas frecuencia, pico de resonancia, etc.

Este grupo de pruebas da una primera idea de la calidad del modelo, que debe ser complementada con otro criterio.

Validación cruzada

Un paso necesario en la validación del modelo, es comparar gráficamente en el dominio del tiempo, la respuesta del modelo $\boldsymbol{y_m}$ y $\boldsymbol{y}$, al introducirle a ambos la entrada de datos experimentales $\boldsymbol{u}$, ver Fig. 3.10.

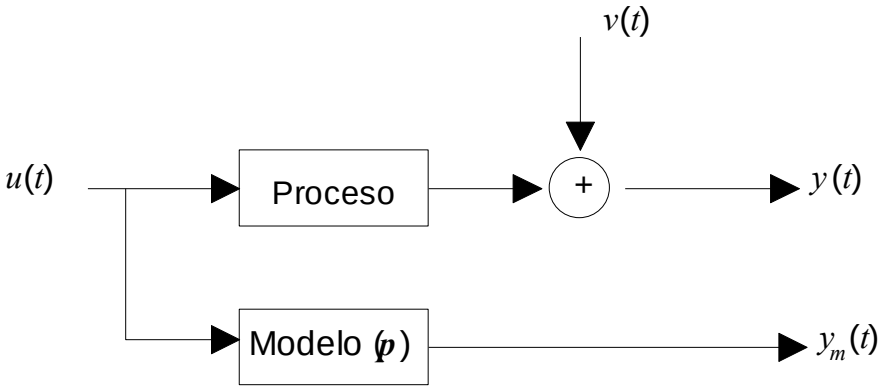


Fig. 3.10 Validación cruzada del modelo

La comparación entre la respuesta de $\boldsymbol{y_m}$ y $\boldsymbol{y}$, conviene hacerla en relación a otros conjuntos de datos experimentales distintos a aquellos que sirvieron para la identificación, ver Fig. 3.11. De este modo, se puede estudiar la independencia de los resultados del conjunto de los experimentos particulares.

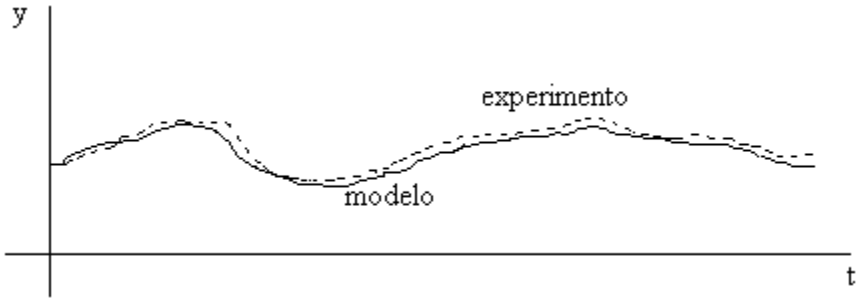


Fig. 3.11 Comparación del modelo obtenido y el sistema real

Índices de performance

Una medida cuantitativa de la calidad del modelo puede ser obtenida usando los índices RMS (Media cuadrática relativa) y MSE (Error cuadrático medio), ver ecuación 3.15 y 3.16:

$$RMS_i = \sqrt{\frac{\sum_{k=1}^{N}\left(y_{ri}(k) - y_{mi}(k)\right)^2}{\sum_{k=1}^{N} y_{ri}(k)^2}};\quad i = 1,2,\ldots \tag{3.15}$$

$$MSE_i = \frac{1}{N}\sum_{k=1}^{N}\left(y_{ri}(k) - y_{mi}(k)\right)^2;\quad i = 1,2,\ldots \tag{3.16}$$

Los índices anteriores pueden ser usados para comparar los modelos obtenidos mediante varios algoritmos de identificación.

El índice *RMS* indica una medida de los errores respecto a los valores reales de salida en un experimento, es decir, si durante un experimento se obtiene un *RMS* menor que 1, se puede decir que el modelo obtenido es bueno.

El índice *MSE* indica una varianza de los errores absolutos a lo largo de un experimento.

Este índice puede ser empleado para calcular un error promedio extrayéndole la raíz cuadrada. Dependiendo del rango de trabajo en que se encuentre la variable de salida, se puede extraer conclusiones de cuan bueno es el modelo obtenido.

3.2 Controladores PID

Las estrategias de control PID cuentan con un esquema de ajuste puramente empírico. Está basado por la combinación de tres acciones básicas de control: proporcional, integral y derivativa.

La Fig. 3.12 muestra un sistema a lazo cerrado, en el cual C(s) representa el controlador.

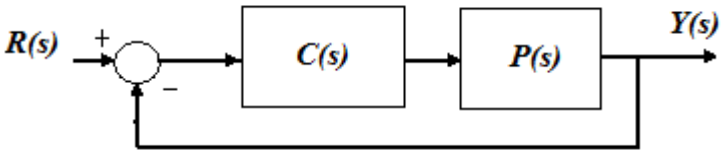


Fig. 3.12 Sistema a lazo cerrado

Luego la función de transferencia del controlador PID resulta ser la ecuación (3.17), esta configuración es conocida como PID ideal.

$$C(s) = \frac{u(s)}{e(s)} = K_c\left(1 + \frac{1}{T_i s} + T_d s\right) \tag{3.17}$$

Donde u(s) y e(s) son la señal de control y el error respectivamente.

La descripción del PID ideal puede escribirse también como la ecuación (3.18), conocida también como PID paralelo.

$$C(s) = \frac{u(s)}{e(s)} = \left(K_p + \frac{K_i}{s} + K_d s\right) \tag{3.18}$$

Donde las relaciones entre las ecuaciones (3.17) y (3.18) son las siguientes.

$$K_p = K_c \tag{3.19}$$

$$K_i = \frac{K_c}{T_i} \tag{3.20}$$

$$K_d = K_c T_d \tag{3.21}$$

Tipos de controladores

Los controladores típicos en sistemas de control en tiempo continuo son:

- Control proporcional *(P)*
- Control proporcional *derivativo (PD)*
- Control proporcional integral *(PI)*
- Control proporcional integral derivativo *(PID)*

A continuación se estudia el funcionamiento de estos controladores, enfatizando sus efectos sobre el estado estacionario del sistema, que determinan su precisión, y sobre el estado transitorio, determinado bajo las especificaciones dinámicas requeridas.

3.2.1 Control proporcional

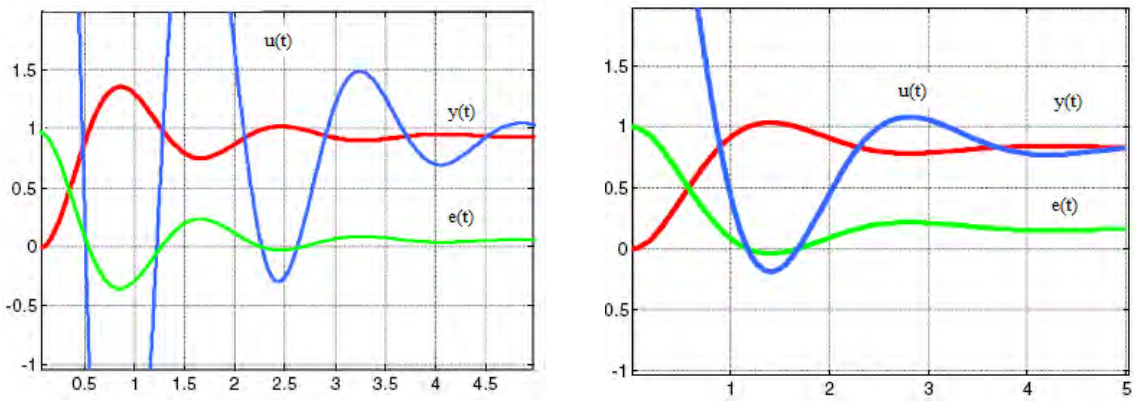
El controlador proporcional genera a la salida una señal de control que es proporcional a la señal de error. Según la ecuación (3.22).

$$u(s) = K_c e(s) \tag{3.22}$$

Con lo cual, la función de transferencia del controlador proporcional es:

$$C(s) = K_c \tag{3.23}$$

En la Fig. 3.13 se puede observar las respuestas típicas (señal de error, señal de control y señal de salida) de un control proporcional con dos valores diferentes de ganancia proporcional.



(a) Control proporcional con k=15 (b) Control proporcional con k=5

Fig. 3.13 Control proporcional

Cuanto mayor es la ganancia del control proporcional, mayor es la señal de control generada para un mismo valor de señal de error, lo cual en ocasiones resulta en saturación de actuadores. Además se puede decir que para una señal de control determinada cuanto mayor es la ganancia del control proporcional, menor es la señal de error actuante. En conclusión, el aumento de la ganancia del control proporcional permite reducir el error en estado estacionario.

3.2.2 Control proporcional integral

La acción puramente integral genera una señal de control proporcional a la integral de la señal del error, ecuación 3.24:

$$u(s) = \frac{Ki}{s} e(s) \tag{3.24}$$

La característica más importante de este tipo de control, es que la acción correctora se efectúa mediante la integral del error, ello permite decir que el control integral, proporciona una señal de control que es función de la propia 'historia' de la señal de error; permitiendo obtener una señal de control diferente de cero aunque la señal de error sea cero.

El control integral permite obtener un error estacionario nulo en un sistema de control, mediante la introducción de un elemento integrador en la función de transferencia a lazo abierto.

La Fig. 3.14 muestra una gráfica típica de la señal de control y del error integral, donde se observa que la *señal u(t)* corresponde al área de la señal *e(t)*.

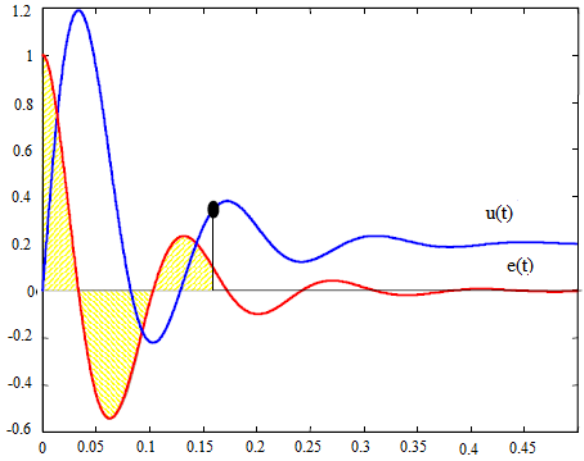


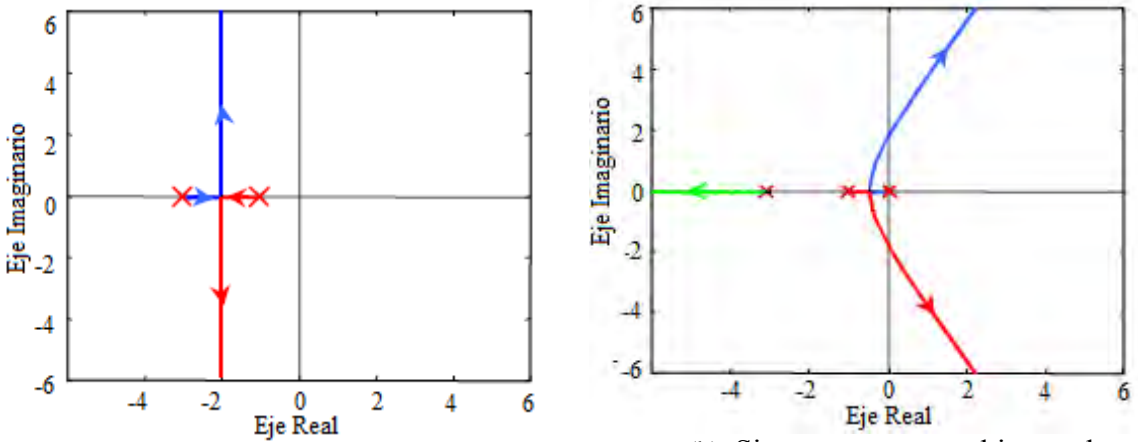
Fig. 3.14 Señal de error e(t) y señal de control u(t)

Si se calcula el error en régimen estacionario ante una entrada al escalón, se evaluaría la ecuación (3.25).

$$e_{ss} = \lim_{s\to 0} \frac{1}{1+C(s)P(s)} \tag{3.25}$$

Si el controlador *C(s)* tiene un elemento integrador (polo en *s=0*) entonces $e_{ss} \to 0$.

Sin embargo, la acción de control integral empeora de un modo substancial la estabilidad relativa del sistema, aumentando el sobre impulso de la respuesta transitoria, pudiéndose obtener inclusive un sistema inestable, debido a que al incorporar un polo en lazo abierto en el origen se desplaza el lugar geométrico de raíces del sistema hacia el semiplano derecho (Fig. 3.15). Por esta razón, en la práctica, la acción integral suele acompañarse por otras acciones de control.



(a) Sistema sin control integral (b) Sistema con control integral

Fig. 3.15 Comportamiento del lugar geométrico de las raíces

Debido al problema que presenta la acción de control integral, se utiliza una combinación proporcional integral *(PI),* la cual genera una señal resultante de la combinación de la acción proporcional y la acción integral conjuntamente. La función transferencia de este controlador se muestra en la ecuación (3.26).

$$u(s) = Kc\left[1 + \frac{1}{Tis}\right]e(s) \tag{3.26}$$

La estructura en diagrama de bloques del controlador PI se muestra en la Fig. 3.16.

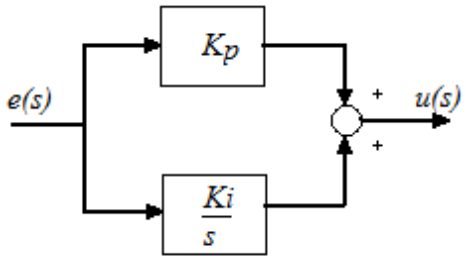
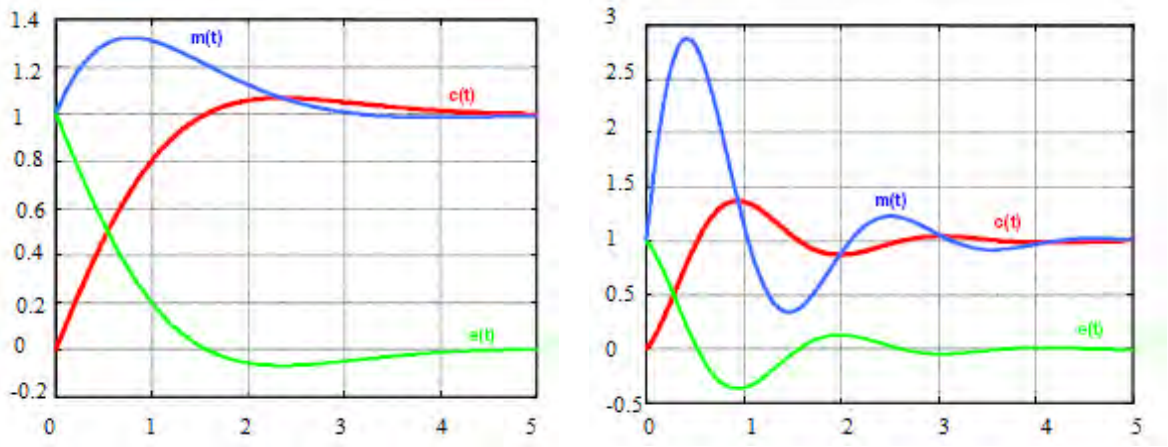


Fig. 3.16 Diagrama de bloques de un control PI

El control integral proporcional combina las ventajas de la acción proporcional y de la acción integral; la acción integral elimina el error estacionario, mientras que la acción proporcional reduce el riesgo de inestabilidad que conlleva la introducción de la propia acción integral.

En la Fig. 3.17 se observa las respuestas temporales de un sistema con control proporcional integral.



(a) Control PI con K=1, Ki=2 (b) Control PI con K=1, Ki=10

Fig. 3.17 respuestas temporales de un sistema con control PI

3.2.3 Control proporcional derivativo

La acción de control derivativa genera una señal de control proporcional a la derivada de la señal de error, tal como lo muestra la ecuación (3.27).

$$u(s) = K_d s.e(s) \tag{3.27}$$

De este modo, el control derivativo mediante la derivada de la señal del error 'conoce' sus características dinámicas (crecimiento o decrecimiento), produciendo una corrección antes de que la señal de error sea excesiva. A este efecto se le denomina acción anticipativa.

La acción de control derivativa añade sensibilidad al sistema, y tiene efecto de aumento en la estabilidad relativa.

Sin embrago, el control derivativo no puede utilizarse en solitario porque es incapaz de responder a una señal de error constante.

$$e(t) = cte. \Longrightarrow u(t) = 0 \tag{3.28}$$

En conclusión, con un control derivativo un sistema no alcanzaría nunca el estado estacionario. El control derivativo siempre debe utilizarse en combinación con otros controles por su influencia estabilizadora mediante la acción anticipativa.

La acción proporcional derivativa (PD) genera una señal que es el resultado de la combinación de la acción proporcional y la acción derivativa conjuntamente. La función de transferencia que describe a este controlador se muestra a continuación.

$$u(s) = K_c[1 + T_d s]e(s) \tag{3.29}$$

La estructura en diagrama de bloques se muestra en la Fig. 3.18.

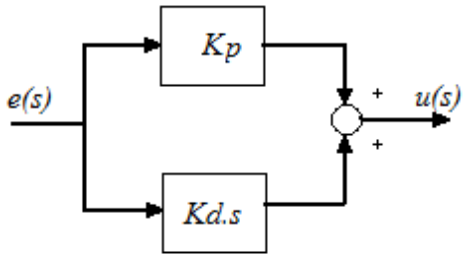


Fig. 3.18 Diagrama de bloques de un control PD

El control proporcional derivativo, proporciona al sistema una mayor estabilidad relativa, que se traduce en una respuesta transitoria con menor sobreimpulso. Sin embargo, cuando la influencia del control es muy grande, el sistema de control tiende a ofrecer una respuesta excesivamente lenta.

Existen dos posibles métodos de diseño, según se priorice el cumplimiento de las condiciones de régimen estacionario, ó según se priorice el transitorio en las respuestas temporales.

El primer método obtiene una determinada respuesta temporal transitoria, quedando el régimen estacionario de la respuesta temporal en función del diseño realizado.

El segundo método fija una determinada respuesta en régimen permanente, quedando las características del estado transitorio en función del diseño realizado.

3.2.4 Control proporcional integral derivativo

La acción de control proporcional integral derivativa *(PID),* genera una señal resultado de la combinación de la acción proporcional, la acción integral y la derivativa.
La función de transferencia que define a este controlador es la (3.30).

$$u(s) = K_c\left[1 + T_d s + \frac{1}{T_i s}\right] \tag{3.30}$$

La estructura en diagrama de bloques se muestra en la Fig. 3.19.

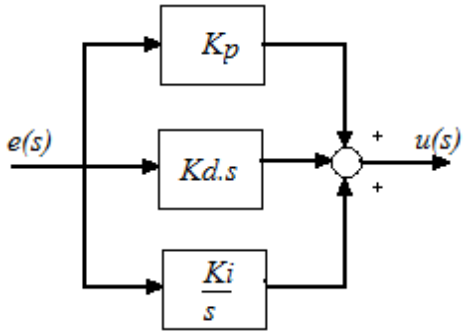


Fig. 3.19 Diagrama de bloques de un control PID

La acción de control *PID* permite eliminar el error en estado estacionario, logrando una buena estabilidad relativa del sistema de control. La mejora en estabilidad relativa, implica una respuesta transitoria con tiempos de adquisición, y valores sobre-oscilaciones pequeñas.

Capítulo 4

Desarrollo software y hardware

4.1 Desarrollo hardware

4.1.1 Diseño de esquemáticos

Para el diseño del circuito esquemático se desarrolló bajo el software *Altium Designer*. cuenta con librerías para componentes superficiales y brinda la posibilidad de diseño de tarjetas *PCB* multicapa. Este software reduce el tiempo de desarrollo ya que cuenta con muchas herramientas para este tipo de proyectos.

El proyecto consiste en el diseño de un determinado circuito, para lo cual se debe contar con las librerías. Es importante determinar las dimensiones de la *PCB* desde el inicio de proyecto.

En esta etapa del diseño se considera los siguientes procedimientos, los cuales se explican a continuación:

Primero se crea un nuevo proyecto y es guardado con un nombre, ver Fig. 4.1. Posteriormente se agregan al proyecto las hojas de los esquemáticos que son requeridos.

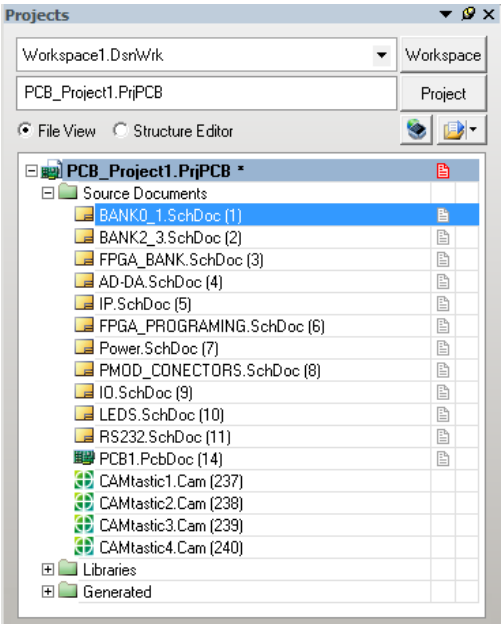


Fig. 4.1 Proyecto de Esquemático de la Board FPGA

El segundo paso corresponde al diseño de los esquemáticos. A continuación se muestran algunos de ellos.

La Fig. 4.2 muestra el circuito de programación por comunicación *JTAG,* el cual es el acrónimo de *Joint Test Action Group.*

Ya que posee una sola línea de datos, este protocolo es necesariamente serial, como el *Serial Peripheral Interface.*

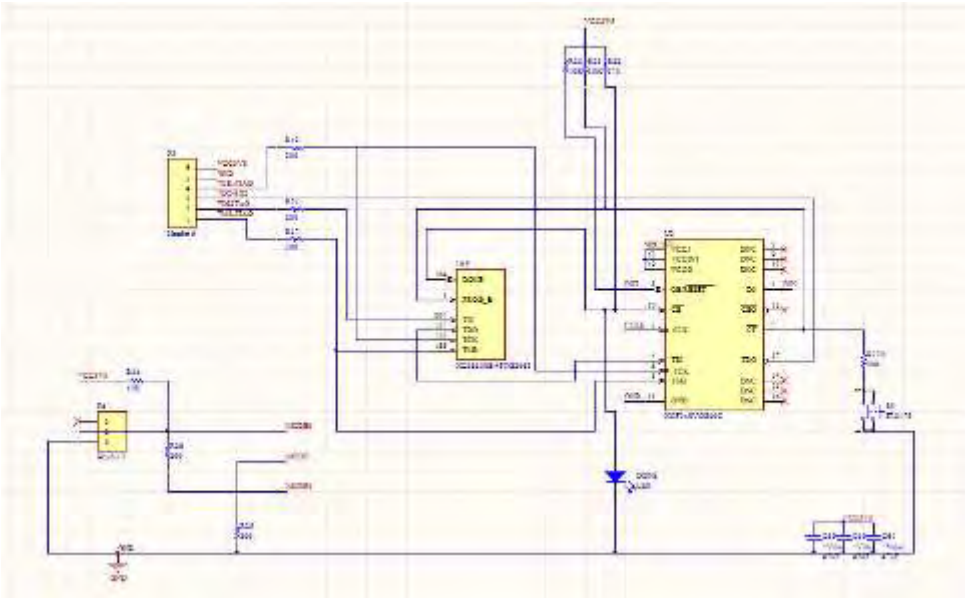


Fig. 4.2 Esquemático de Programación JTAG

La Fig. 4.3 y Fig. 4.4 vemos el esquemático del circuito de conversión Analógico-Digital (ADC) y el circuito de conversión Digital – Analógico (DAC).

El circuito ADC es el encargado de recoger el dato en niveles de voltaje de 0-3.3V provenientes del sensor y digitalizarlos. Este circuito de conversión ADC cuenta también con un amplificador programable y una resolución de 14 bits.

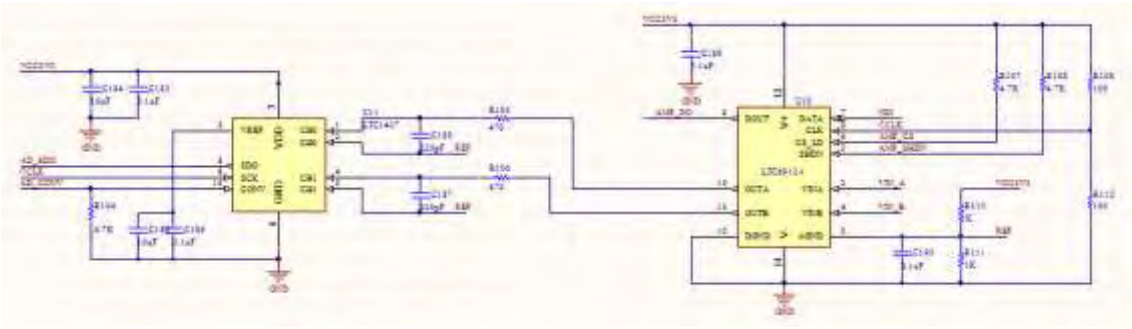


Fig. 4.3 Esquemático de Conversión ADC

El circuito DAC es el encargado de convertir un dato en digital a una señal analógica.

EL circuito DAC entrega niveles de tensión de 0-3.3V. Este circuito cuenta con una resolución de 12 bits.

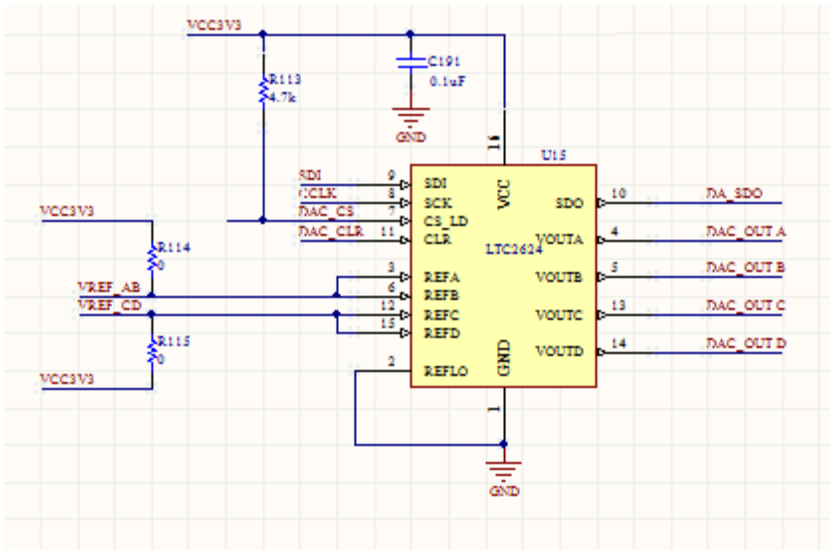


Fig. 4.4 Esquemático de Conversión DAC

Los integrados LTC6912 y LTC2624, (ADC y DAC respectivamente) envían los datos en forma serial, estos trabajan en protocolo SPI.

En la Fig. 4.5 vemos el circuito esquemático que encarga de entregar los niveles de voltaje necesario para el funcionamiento tanto de FPGA con de los demás circuitos que se encuentran en la PCB diseñada.

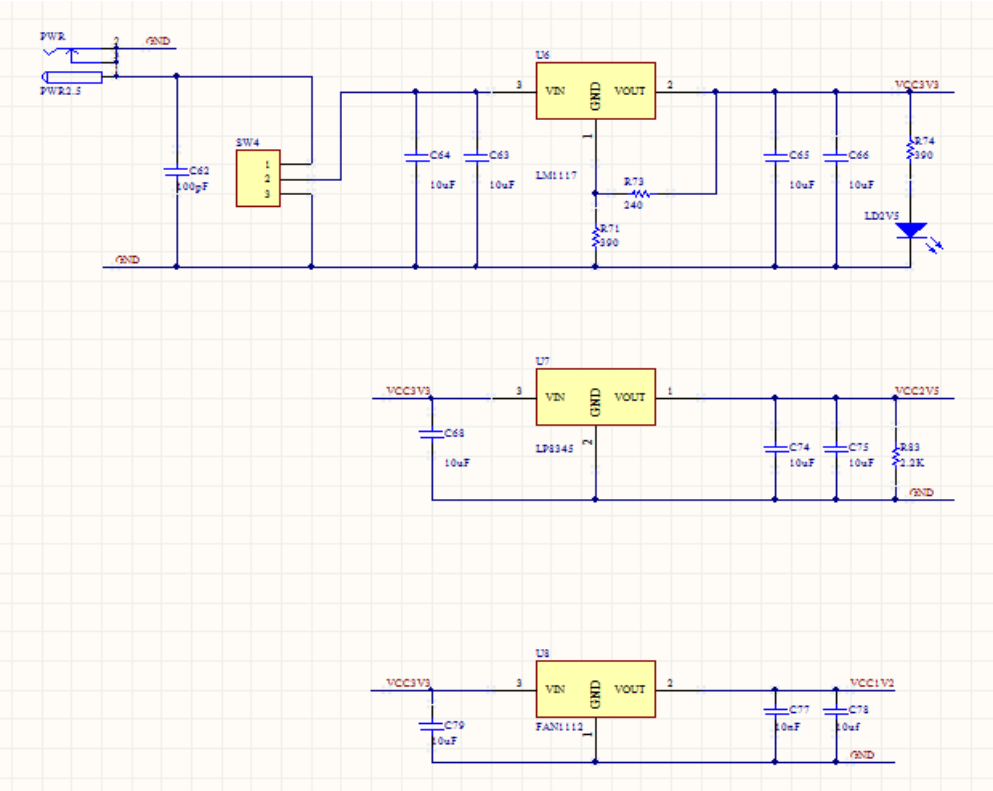


Fig. 4.5 Esquemático de Alimentación

En la Fig. 4.6 tenemos el circuito de comunicación *RS-232*, es también un protocolo muy común utilizado por varios dispositivos, para la interacción con otros dispositivos de su entorno, en este caso, con el sistema *SCADA*.

Esta envía y recibe bytes de información, en un bit a la vez. Se transmite los datos en formato *ASCII*. Para realizar la comunicación se utilizan 3 líneas de transmisión, en las cuales se tiene: línea de referencia *(GND),* línea de transmisión de *datos (RS-TX)* y línea de recepción de datos (RS-RX).

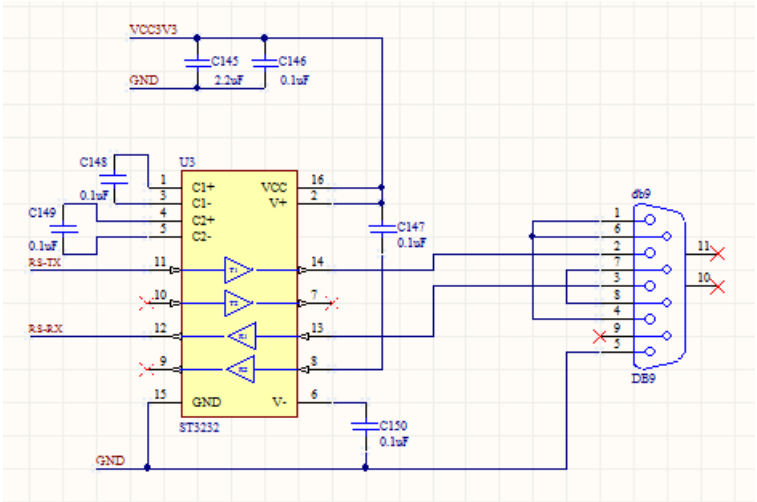


Fig. 4.6 Esquemático de Comunicación RS-232

La PCB diseñada cuenta también con un conjunto de leds, interruptores y pulsadores. La Fig. 4.7 se muestra el esquema del circuito.

Estos están conectados a los pines del FPGA, los cuales deben de ser configurados como entrada o salida de tipo bit.

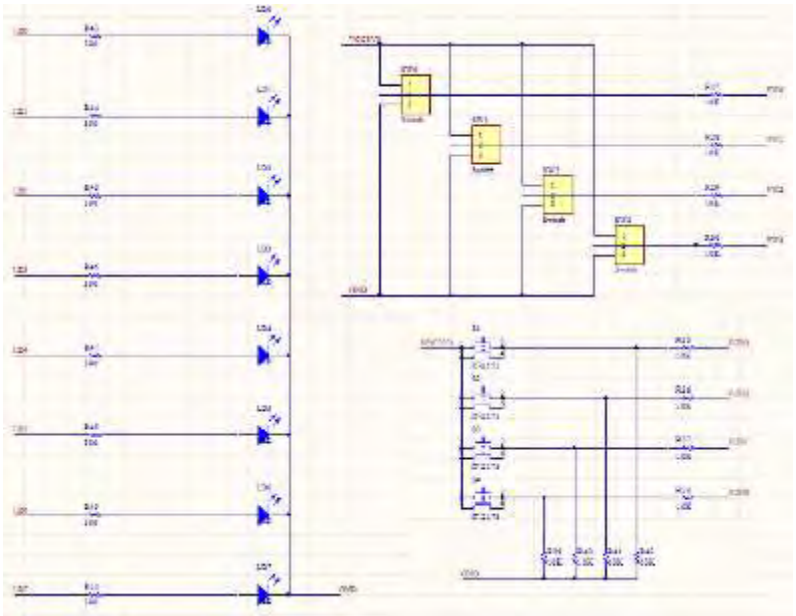


Fig. 4.7 Esquemático de diferentes señales de entrada y salida

Compilación de esquemáticos

Consiste en la depuración de errores presentes en las hojas de los circuitos esquemáticos, así como también la numeración e identificación de los componentes. En la Fig. 4.8 podemos observar el menú de compilación.

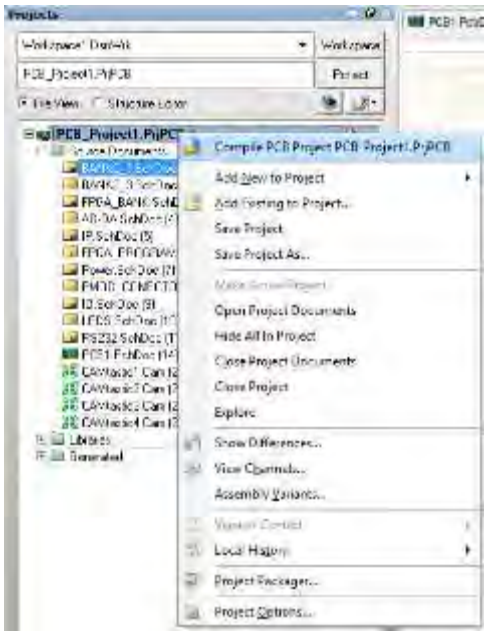


Fig. 4.8 Compilación de los Esquemáticos

4.1.2 Diseño del PCB (print circuit board)

Para el desarrollo de la tarjeta *PCB* se manejó el software *Altium Designer,* en versión *DEMO*, herramienta *CAD* que nos ayudará a trabajar la capa *Hardware Layer* del modelo del sistema embebido.

Es un programa para el diseño electrónico en todas sus fases, desarrollo de circuito esquemático, simulación, implementación de *FPGA*, y hasta desarrollo de código para microprocesadores.

El diseño del *PCB*, placa de circuito impreso, consiste en disponer de los elementos y el ruteo de las pistas en software. Para ello se debe importar los archivos esquemáticos creados y compilados anteriormente.

Se desarrolla primero los circuitos esquemáticos, luego se realiza el ruteo de las pistas y posteriormente la simulación en 3D.

El ruteo muestra la disposición real de los dispositivos electrónicos, es la generación de pistas de cobre por software, por donde viajan las señales.

Posteriormente para la construcción del prototipo, se necesita un archivo máquina, el cuál tiene la información de la *PCB*, y es cargada a una máquina *CNC (Computer Numerical Control)* para la fabricación de ésta.

A continuación se muestran algunas reglas de diseño que se han tenido en cuenta para la fabricación de la tarjeta PCB.

En la Fig. 4.9 se muestra la asignación de número de capaz y su configuración. Se estableció el diseño de la tarjeta con 4 capaz.

- La capa superior, llamada *Component Side,* hay componentes *SMD* soldados.
- La segunda capa es la *Ground Plane (GND),* es la referencia del sistema.
- La tercera capa es la *Power Plane* (3.3V), capa de alimentación a 3.3V DC.
- La capa *Solder Side,* es la capa inferior, hay componentes *SMD* soldados.

Las capas van unidas por vías pasantes como se puede apreciar en la Fig. 4.9.

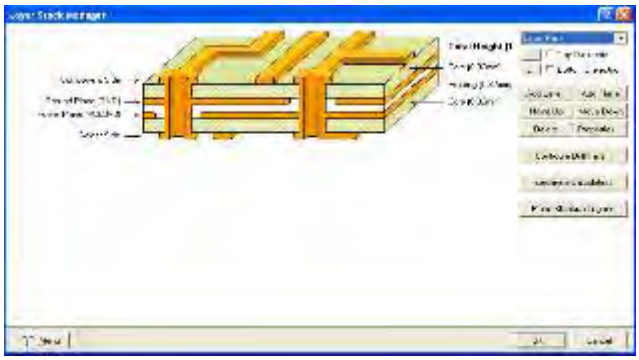


Fig. 4.9 Asignando el número de capas

Una de las reglas de diseño configurada es la distancia mínima entre una vía y la pista, esta es de 0.127mm.

En la Fig. 4.10 se tiene la asignación de este parámetro.

Fig. 4.10 Asignación de distancia entre vía y pista

En la Fig. 4.11 que se muestra la asignación del ancho de pista, la cual tiene un valor mínimo de 9 mil (milésimas de pulgada) y un valor máximo de 24 mil.

La medida por defecto que toma el software al momento de generar la pista es de 9mil, se puede variar hasta 24 mil.

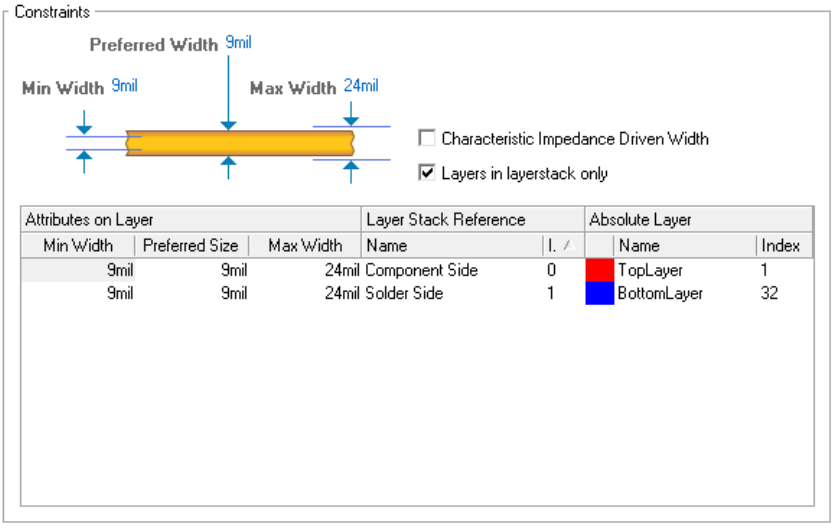


Fig. 4.11. Asignando dimensión de pista.

En la Fig. 4.12, se asigna el valor de ángulo con el cual una pista hace un cambio de dirección, se ha considerado de 45º que es lo recomendable. Así también la dimensión de la longitud, de tramo recto para cambio de dirección, se ha tomado un mínimo de 10 mil y un máximo de 150 mil.

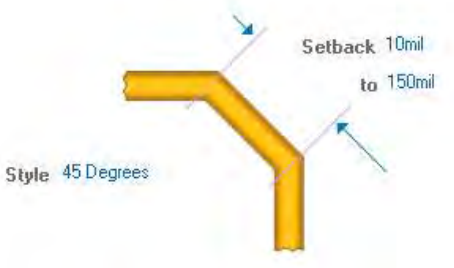


Fig. 4.12. Asignando ángulo para cambio de dirección de la pista

La Fig. 4.13 se aprecia la asignación de las dimensiones para la vía pasante *(hole),* sirve para que una pista pueda cambiar a otra capa.

Se asignan valores máximos, mínimos y preferencial.

Fig. 4.13. Asignación de distancias para las vías pasantes

En la Fig. 4.14 nos muestra el diseño final de la tarjeta PCB; los componentes han sido colocados en sus lugares respectivos, permitiendo el camino más corto hacia el FPGA.

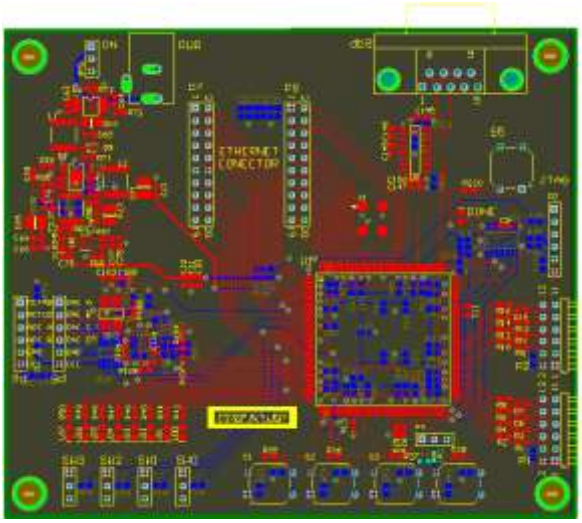


Fig. 4.14. Diseño de PCB

Vista en 3D de la tarjeta

A continuación se muestran diferentes figuras en 3D obtenidas por el software, el cual nos muestra en modo de simulación en software, de esta forma apreciamos la tarjeta PCB con los componentes respectivos antes de la fabricación.

La Fig. 4.15 se tiene una vista frontal de la PCB.

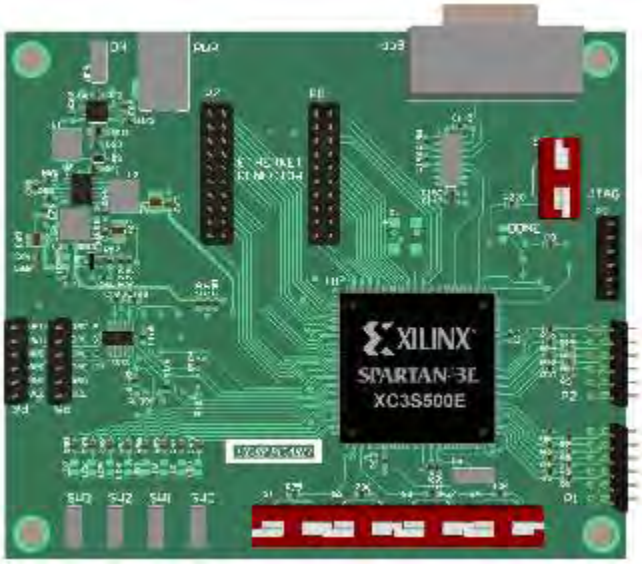


Fig. 4.15. Vista frontal en 3D de la PCB

La Fig. 4.16 tenemos otra vista de la PCB.

Fig. 4.16 Vista en 3D de la PCB

La Fig. 4.17 nos muestra otra vista de la tarjeta PCB.

Fig. 4.17 Vista de la capa superior en 3D de la PCB

La Fig. 4.18 nos muestra la capa *Solder Side* desde otro ángulo de perspectiva.

Fig. 4.18 Vista de la capa inferior en 3D de la PCB

4.1.3 Archivos de fabricación. *(GERBER)*

Los archivos *GERBER* es un formato de impresión utilizados en la producción de circuitos impresos creado por *Gerber Systems Corporation.* Permite al fabricante obtener toda la información para la fabricación de la tarjeta. El conjunto de ficheros están formado por:

- GBL - *Gerber Bottom Layer* - Capa inferior.
- GTL - *Gerber Top Layer* - Capa superior.
- GBS - *Gerber Bottom Solder Resist* - Máscara de soldadura inferior.
- GTS - *Gerber Top Solder Resist* - Máscara de soldadura superior.
- GBO - *Gerber Bottom Overlay* - Revestimiento superior.
- GTO - *Gerber Top Overlay* - Revestimiento inferior.
- GBP - *Gerber Bottom Paste* - Pasta de soldadura inferior.
- GTP - *Gerber Top Paste* - Pasta de soldadura superior.
- GKO - *Gerber Keep* - Out Layer.
- GM1 - *Gerber Mechanical 1* - Mecánica 1.
- GM2 - Gerber Mechanical 2 - Mecánica 2.
- GPT - *Gerber Top Pad Master* - Máscara de pads superior.
- GPB - *Gerber Bottom Pad Master* - Máscara de pads inferior.

La Fig. 4.19 muestra los archivos máquina de la PCB.

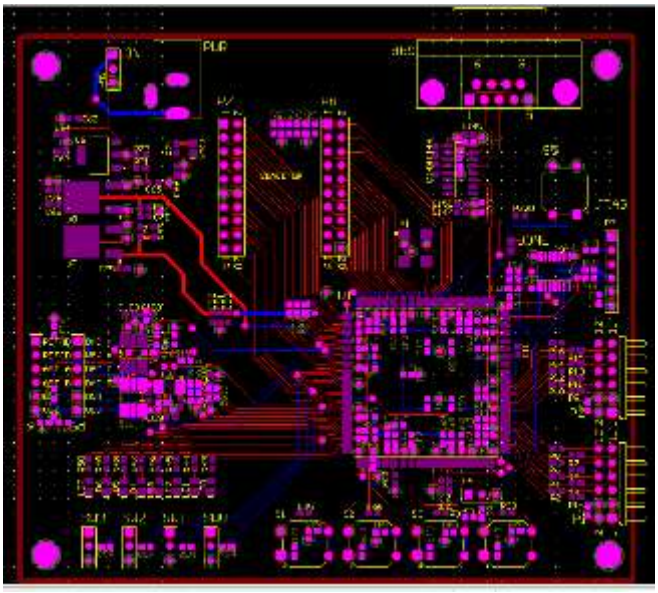


Fig. 4.19 Archivo Gerber de fabricación

4.1.4 Componentes de la PCB

La tarjeta desarrollada, consta de los siguientes periféricos, los cuales van a llevar las distintas señales como son de comunicación, de sensores, de actuación.

La PCB cuenta con los siguientes puertos:

- 2 canales Analógicos digitales ADC.
- 4 canales de salida Analógicos DAC.
- 1 puerto UART.
- 4 Interruptores.
- 5 pulsadores.
- 8 Leds.
- 1 puerto JTAG para programación.
- 4 conectores PMOD.
- 2 Conectores de Propósito General.

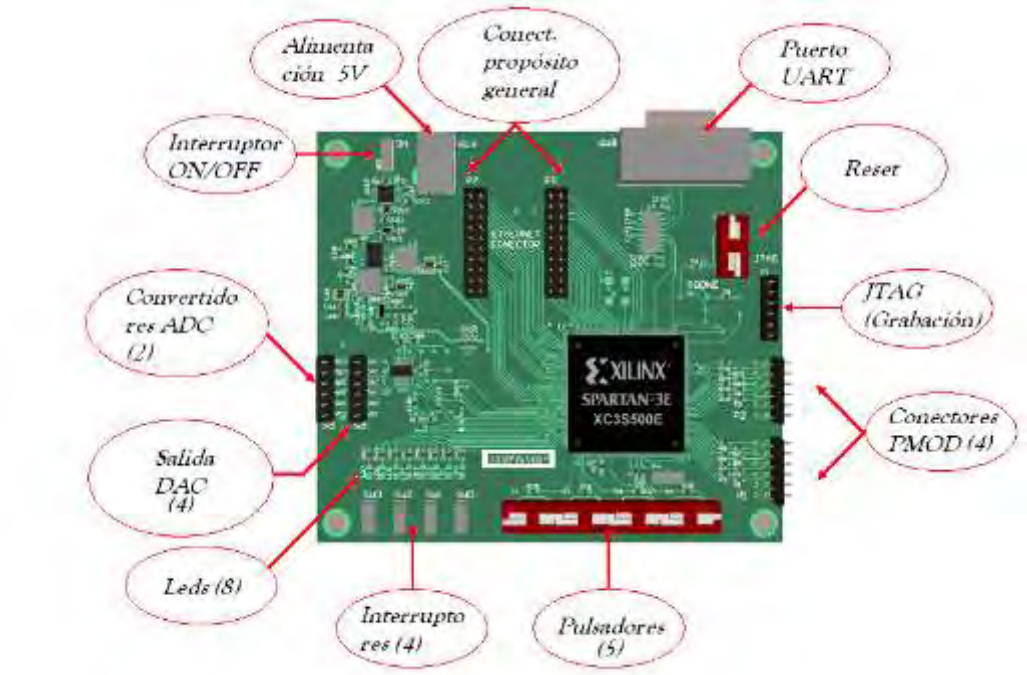


Fig. 4.20 Interfaces de entrada y salida de la PCB

A continuación se mencionan cada uno de estos.

Convertidores ADC: La tarjeta consta de 2 convertidores ADC de 14 bits, los cuales se encargan de transformar la señal analógica a digital. Posee también asimismo un amplificador de ganancia programable, el cual servirá para elevar los niveles de tensión que estén en rangos de mV.

Convertidores DAC: Así mismo costa también de 4 convertidores DAC, los cuales se encargar de convertir un dato digital a analógico. Esta señal será enviada a algún dispositivo de potencia o de actuación.

Puerto JTAG: Es el puerto por el cual se realiza la descripción de hardware al FPGA. El archivo .bit previamente realizado en VHDL y sintetizado, se graba al FPGA por este puerto. Así también la tarjeta posee una memoria RAM, para asegurar que el programa no se pierda al cortar la alimentación, al momento de grabación se debe indicar.

Reset: Este consta de un pulsador el cual da un reset al FPGA. Si el programa descargado en la FPGA no ha sido grabado en memoria, la descripción del hardware creado desaparece por completo del integrado.

Puerto UART: Este se utiliza para una comunicación por protocolo RS-232, Este enviará información hacia el sistema SCADA.

Conectores PMOD: Se utilizan para conectar algún dispositivo externo, ya sea una tarjeta PMOD ADC ó PMOD de puente H; estos periféricos son hardwares

externos que se acoplan a la tarjeta para algún propósito específico no contemplado en el diseño. En esta Board contamos con 4.

LED: Se han colocado 8 leds, los cuales son señales de salida para el FPGA, esto con el fin de tener algún dispositivo que me permita indicar alguna falla u otro fin que se estime conveniente.

Pulsadores: Se han colocado 5 pulsadores a la PCB, los cuales serán señales de entrada digitales al FPGA, para propósito general.

Interruptores: Posee 4 interruptores, son entradas digitales al FPGA para propósito general.

Conector de Propósito General: Este conector consta de 2 headers de 20 entradas cada uno, está diseñado de forma que se podrá conectar un dispositivo de propósito específico. Lo utilizaremos posteriormente para el tema de la comunicación Ethernet.

Interruptor ON/OFF: Consta de un interruptor para alimentar al sistema.

4.1.5 Construcción de prototipo

La tarjeta *PCB* diseñada en la Universidad de Piura ha sido fabricada en una empresa de Taiwán, *SPEED CIRCUIT*.

En la Fig. 4.21, se muestra la tarjeta *PCB*, fabricada, sin ningún componente montado, igual como se muestra en la etapa de diseño de software, a continuación se presentaran etapas de soldado de los componentes.

Fig. 4.21 Tarjeta PCB sin componentes montados

Se comienza a montar los componentes, en la Fig. 4.22, se aprecia el montado del circuito de alimentación, el cual estará encargado de alimentar la *PCB*.

Fig. 4.22 Tarjeta PCB con el circuito de alimentación

Como vemos a continuación en la Fig. 4.23, se monta el integrado *FPGA*, este es de la familia *Spartan 3E XC3S500E-4PQG208C.*

La etapa de soldado del *integrado FPGA* se hace con mucho cuidado, de modo de no dañar los pines y procurando que no se suelden entre ellos, siendo perjudicial para el *FPGA*, ya que podría ocasionar un cortocircuito y dañar el integrado.

Así también se puede observar el circuito de reloj montado, este es un oscilador de 50 MHz, que proporcionará la señal de reloj del sistema.

Fig. 4.23 Tarjeta PCB con integrado FPGA montado

En la Fig. 4.24, tenemos la placa PCB terminada, incluyendo los pines de entrada para programación por *JTAG,* así como también el circuito de comunicación *RS-232*.

También se puede observar los pines de propósito general *PMOD* y también los conectores en donde se podrá conectar el sistema de comunicación Ethernet.

Fig. 4.24 Tarjeta PCB con componentes montados

En la siguiente imagen, Fig. 4.25, vemos el circuito de comunicación Ethernet montado en la tarjeta FPGA.

Fig. 4.25 Tarjeta PCB sin componentes montados

En la Fig. 4.26 se tiene otra vista de la tarjeta PCB incluyendo el componente de comunicación Ethernet.

Fig. 4.26 Vista de la tarjeta FPGA con componente Ethernet montado

4.2 Algoritmos de control de tarjeta FPGA

El desarrollo de la descripción de hardware se realizó usando el software *ISE Simulator de Xilinx*, en código VHDL. Se realizó también la simulación de los diferentes periféricos que se utilizarán para el control del módulo implementado, como son el bloque ADC para leer la señal del sensor (PV), el bloque ADC para enviar la señal de actuación (MV), el bloque de control PI, un bloque de comunicación serial RS-232.

El código ha sido escrito en su mayoría en el lenguaje de descripción de hardware VHDL, también se utilizó el System Generator.

4.2.1 ADC

La programación del convertidor Analógico-Digital se escribió en VHDL. A este bloque se le llamó ADC, el cual lo utilizaremos para digitalizar el dato que viene del sensor, previo acondicionamiento de la señal.

La Fig. 4.27 muestra el código del convertidor ADC.

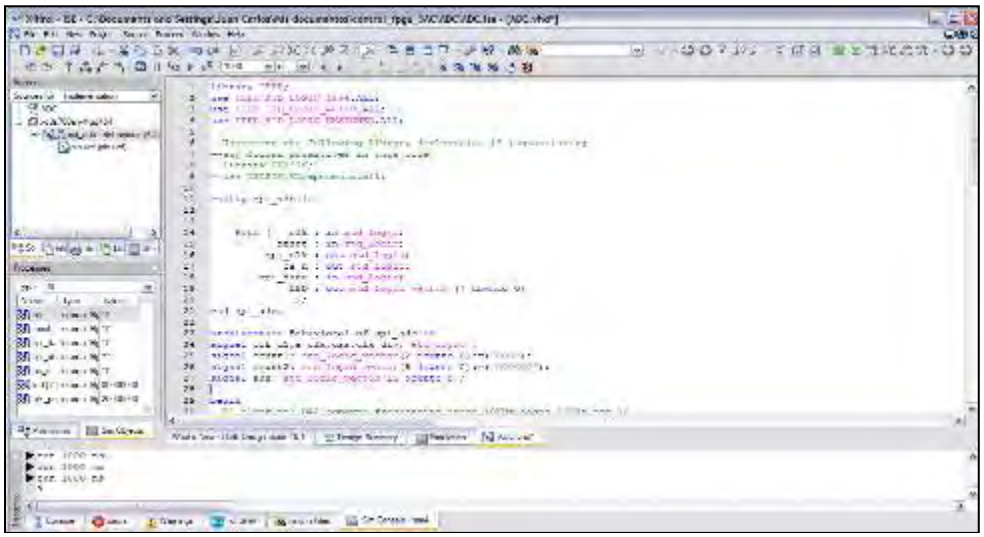


Fig. 4.27 Código del bloque ADC

Se observa el bloque generado después de ser sintetizado, ver la Fig. 4.28. Aquí podemos observar las entradas y las salidas del FPGA.

clk, es una señal de reloj de 50 MHz con cual trabaja el FPGA.

reset, es una señal de entrada que se activará desde el exterior por medio de un pulsador.
spi_data, es la señal ya digitalizada la cual ingresa al FPGA de forma serial por protocolo SPI.

LED, es una señal de salida que usaremos para poder leer el dato, los 8 bits más significativos.

spi_clk, es una señal de salida del FPGA la cual es una señal de reloj para el integrado ADC.

cs_n, es una señal de salida del FPGA el cuál se encargará de activar el integrado, este señal nos fija el tiempo de muestreo.

La tarjeta FPGA cuenta con un convertidor ADC la cual tiene una frecuencia de trabajo de hasta 20MHz.

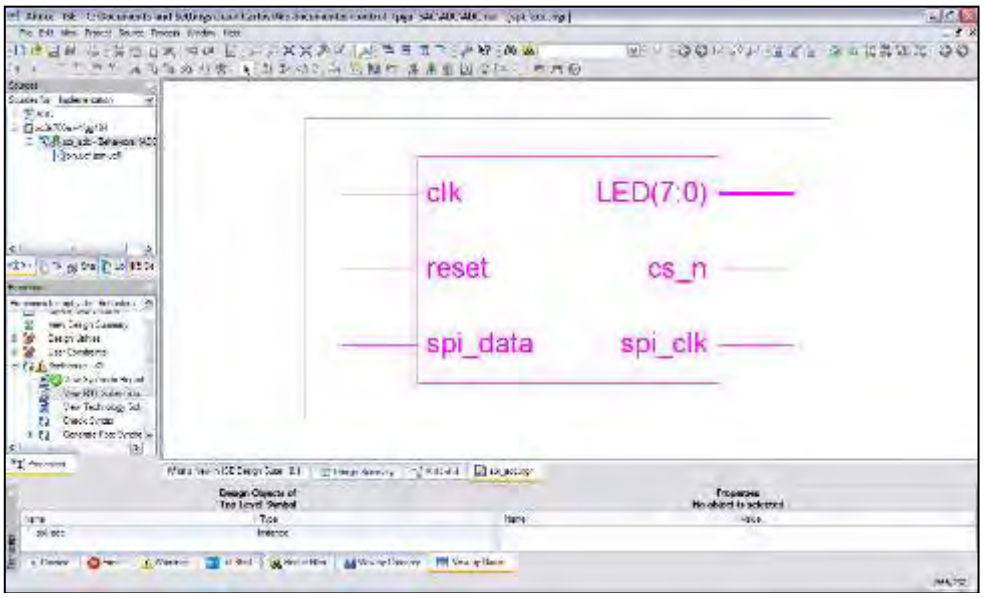


Fig. 4.28 Bloque ADC

En la Fig. 4.29 se muestra el circuito sintetizado que se implementará en la FPGA.

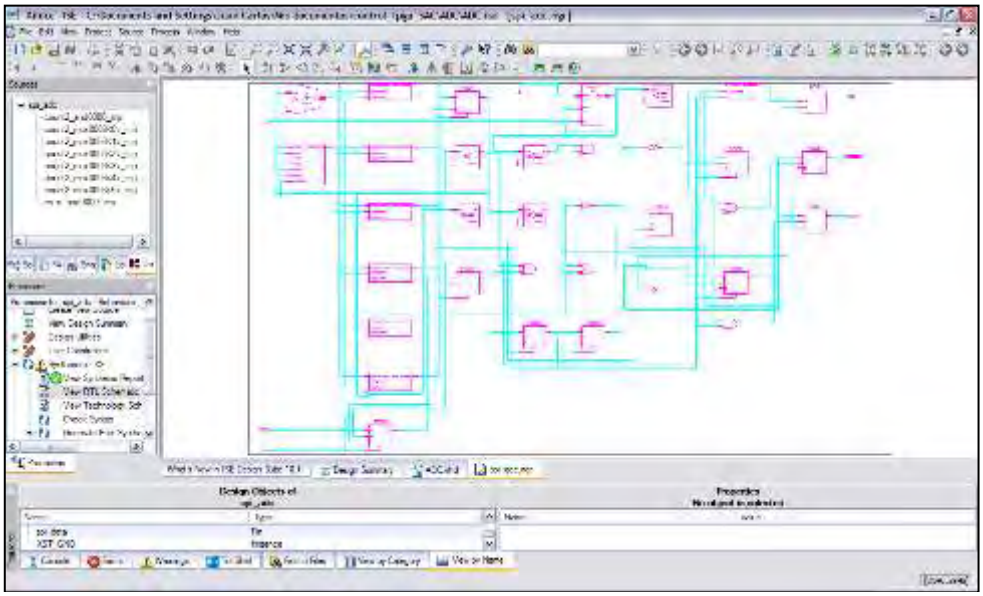


Fig. 4.29 Circuito sintetizado del convertidor ADC

La Fig. 4.30 muestra la simulación del convertidor. Se observa el comportamiento de las diferentes señales que compone el bloque ADC. Como *el spi_clk, el cs_n, clk, reset*.

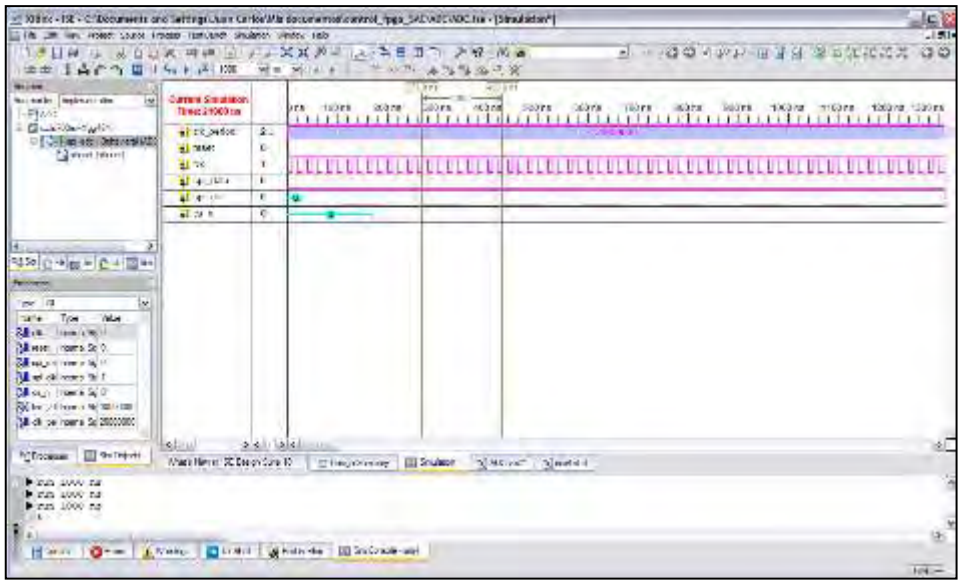


Fig. 4.30 Simulación del convertidor ADC

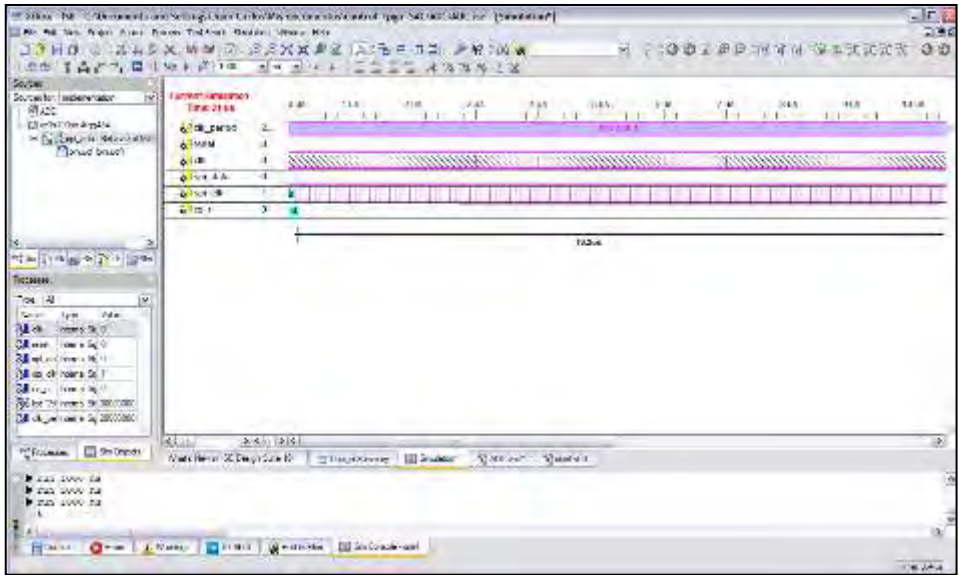


Fig. 4.31 Simulación del convertidor ADC

4.2.2 DAC

La programación del convertidor Digital-Analógico se realizó en lenguaje VHDL. Se hizo un bloque al que se llamó DAC, el cual lo utilizaremos para enviar la señal de actuación.

La señal está en el rango de 0-3.3V. En la Fig. 4.32 se muestra el código en VHDL para el convertidor DAC.

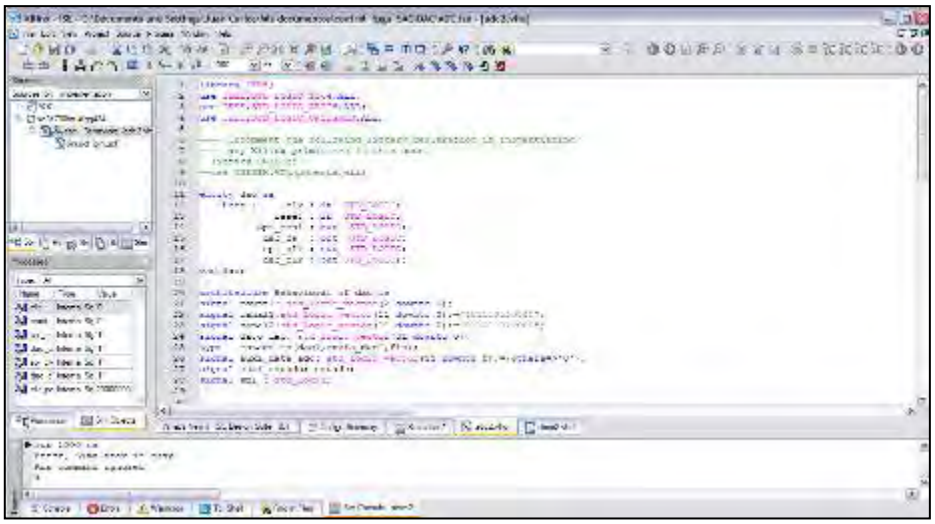


Fig. 4.32 Código del convertidor DAC

Se observa el bloque generado después de ser sintetizado, ver Fig. 4.33. Aquí podemos observar las entradas y las salidas del FPGA.

clk, es la señal de reloj de 50MHz con el que trabaja el *FPGA Spartan 3AN.*

reset, es una señal de entrada que se activará desde el exterior por medio de un pulsador.
spi_mosi, es el dato digital que será convertido en valor analógico, este dato es enviado por el FPGA de forma serial en protocolo SPI.

spi_clk, señal de reloj, está en función de la frecuencia de trabajo del integrado, el cual trabaja hasta una frecuencia de 10kHz.

dac_cs, es una señal de salida del FPGA, se encarga de activar el convertidor DAC.

dac_clear, es una señal de salida del FPGA el cual se encarga de borrar el dato digital del convertidor DAC.

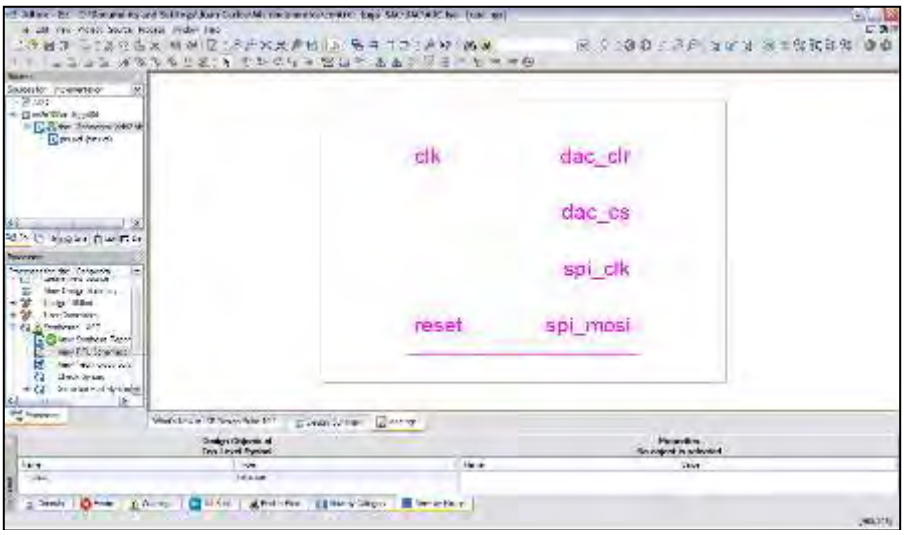


Fig. 4.33 Entidad del convertidor DAC

El circuito que se implementará en la FPGA después de haberlo sintetizado se aprecia en la Fig. 4.34.

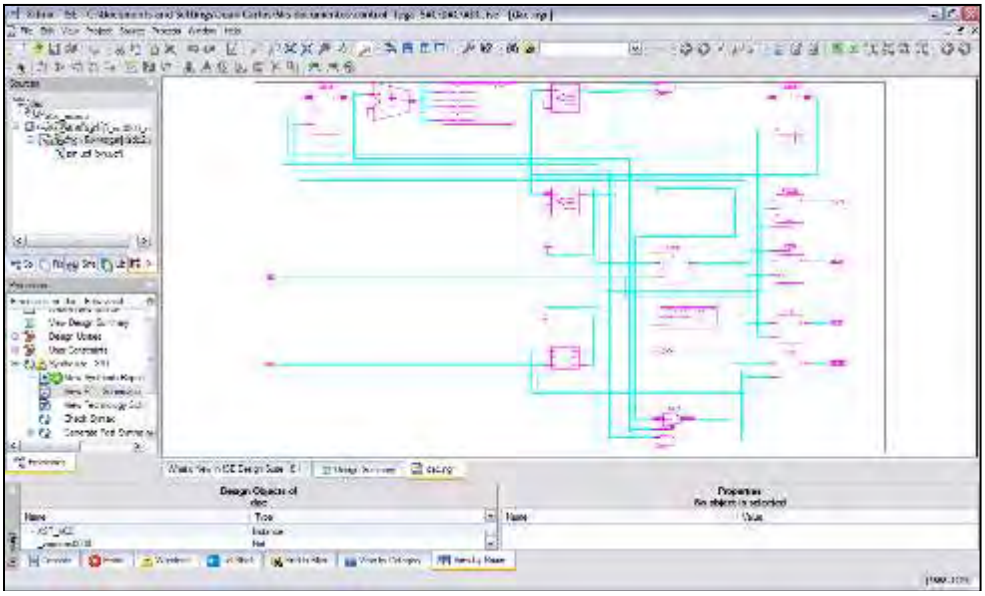


Fig. 4.34 Netlist del convertidor DAC

Las figuras siguientes muestran la simulación del convertidor DAC hecha en entorno *Xilinx* por medio del software *ISE-Simulator,* ver Fig. 4.35 y Fig. 4.36.

Se observa el comportamiento de las diferentes señales que gobiernan el bloque ADC. Como el *spi_clk, dac_cs, clk, reset, spi_mosi, dac_clear.*

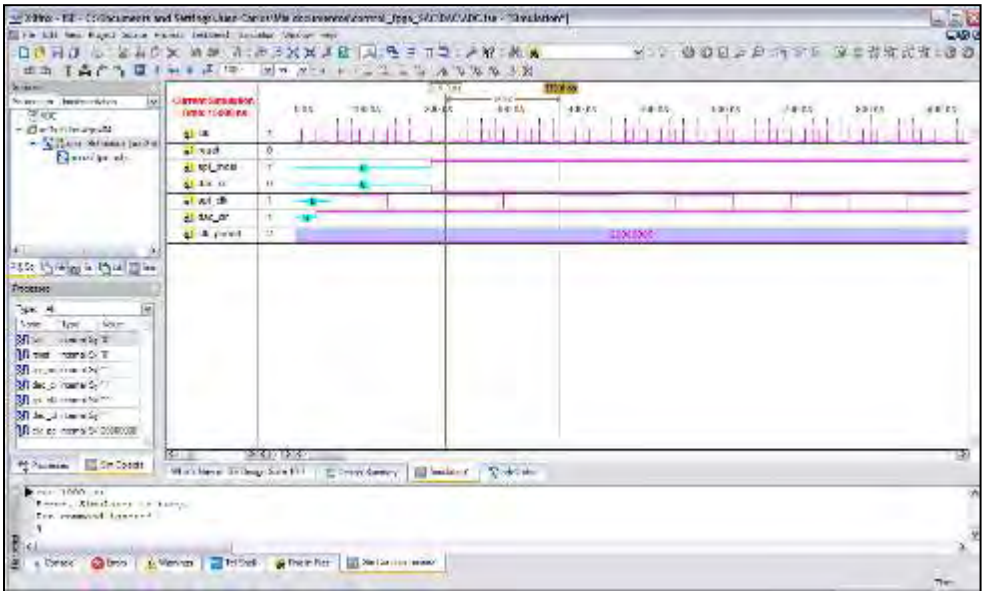


Fig. 4.35 Simulación del convertidor DAC

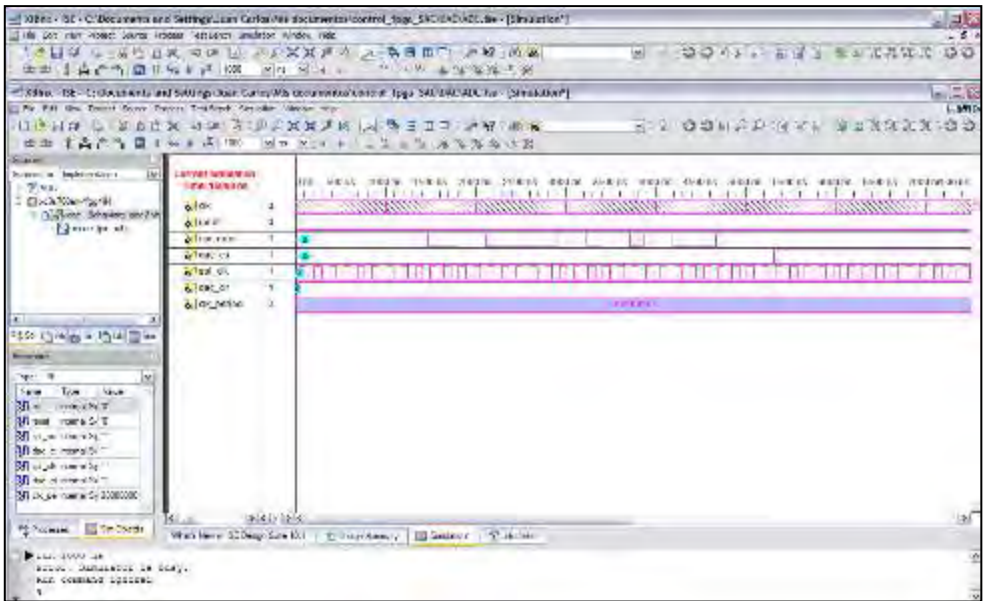


Fig. 4.36 Simulación del convertidor DAC

Como se puede apreciar las señales del *spi_mosi* corresponden al dato que será digitalizado, es enviado por un pin del FPGA de forma serial al integrado DAC, la señal *dac_cs* se encuentra en cero lógico, mientras que la señal de *dac_clr* se encuentra en estado de uno lógico.

4.2.3 Comunicación UART

La programación del bloque de comunicación UART se ha realizado en lenguaje de descripción de hardware *VHDL*. A este bloque se llamó UART, el cual lo utilizaremos para comunicar la PC con la tarjeta FPGA.

El código del bloque *UART* se aprecia en la Fig. 4.37.

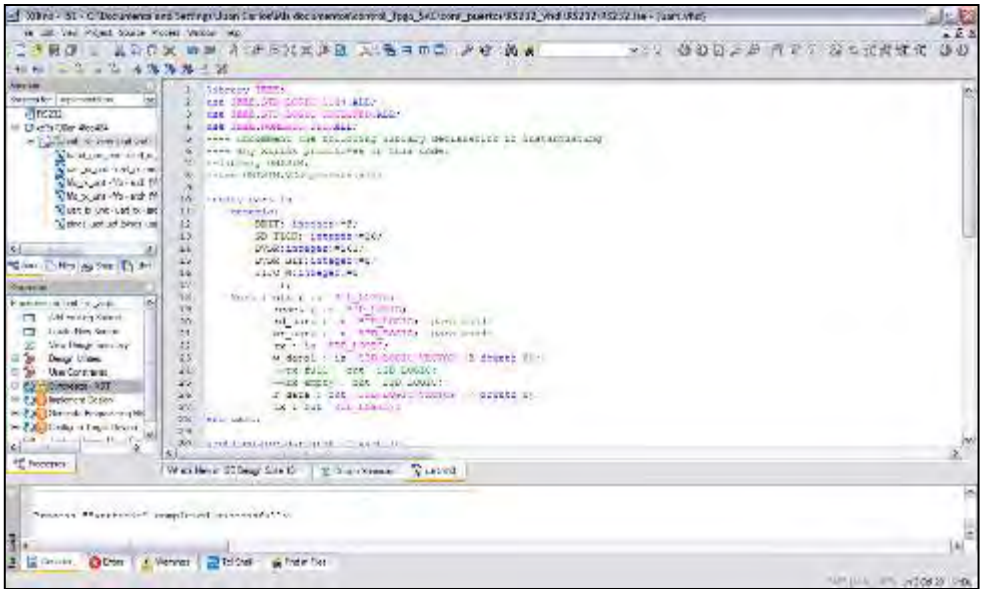


Fig. 4.37 Código del bloque UART

En la Fig. 4.38 se muestra la entidad del bloque que se implementará en la FPGA.

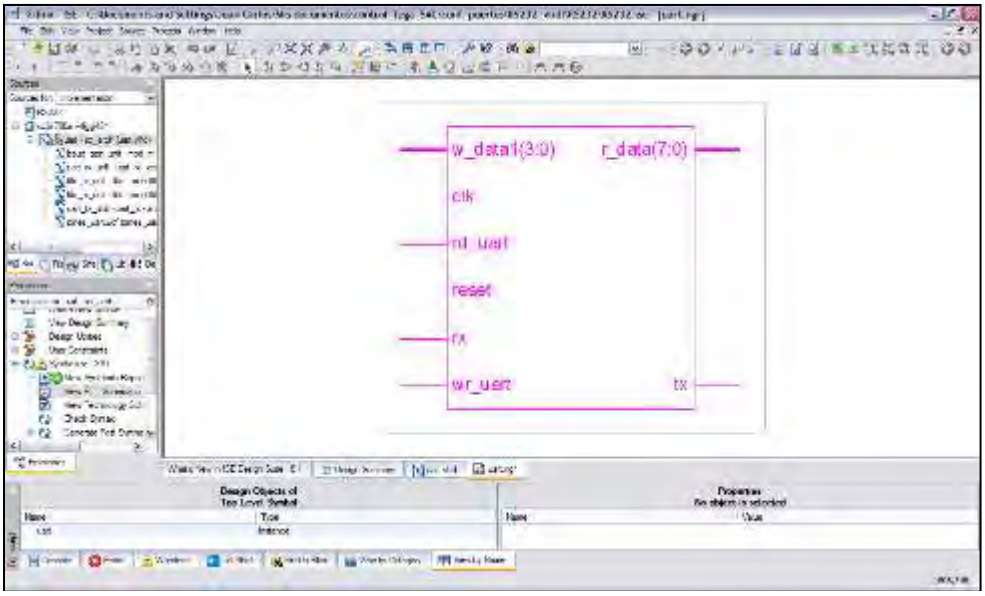


Fig. 4.38 Entidad del bloque UART

En la Fig. 4.39 se muestra el circuito sintetizado que se implementará en la FPGA.

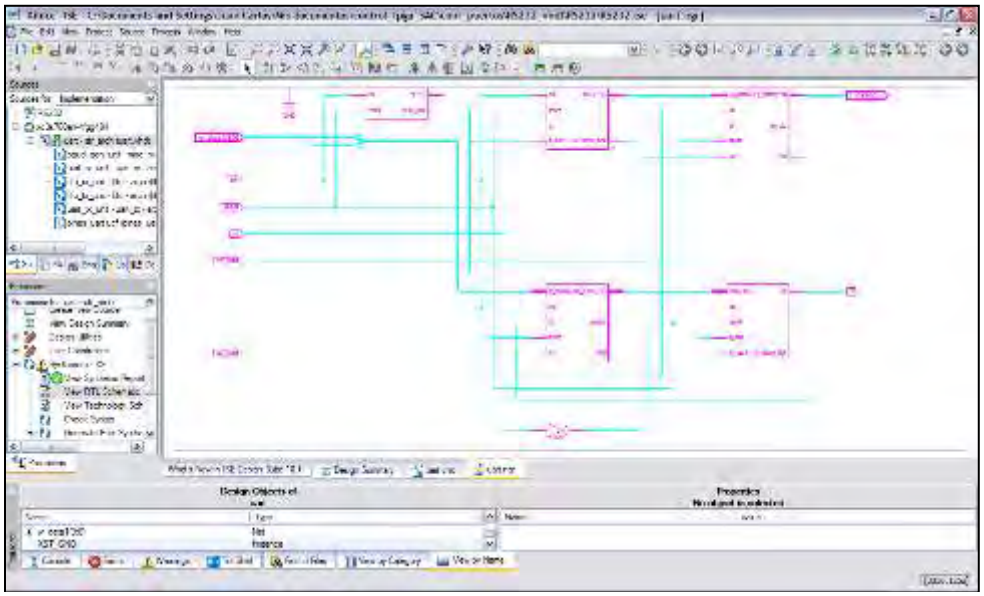


Fig. 4.39 Netlist del bloque UART

La figura siguiente muestra la simulación hecha en entorno *Xilinx,* por medio del software *ISE-Simulator,* ver Fig. 4.40.

Aquí se ve el comportamiento de las diferentes señales que compone el bloque.

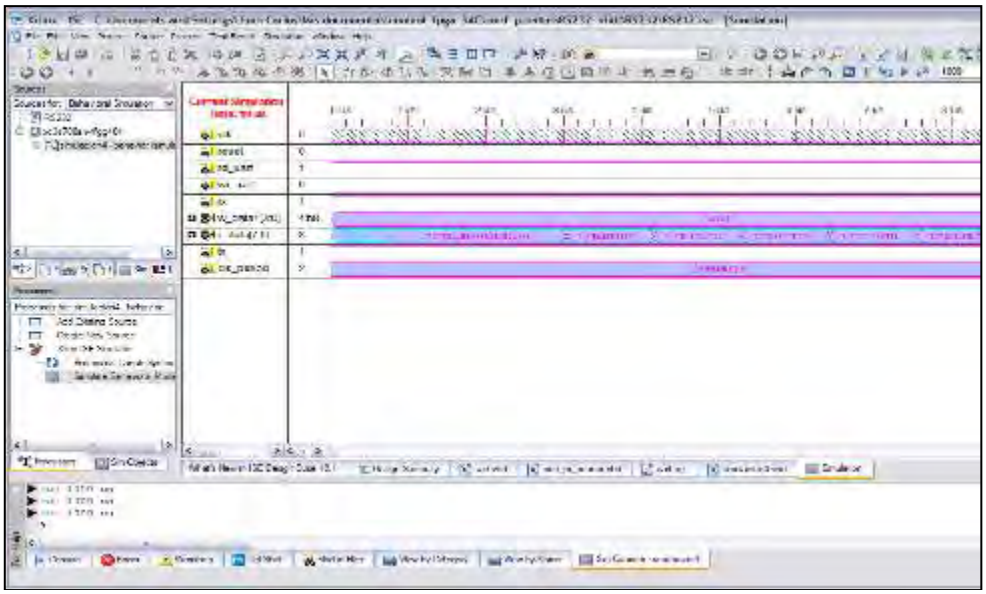


Fig. 4.40 Simulación del bloque UART

4.2.4 Controlador PI

El controlador PI se diseñó usando el software *System Generator de Xilinx*, ver Fig. 4.41, trabaja en *Software MatLab* en entorno *Simulink.*

Permite realizar simulaciones y analizar el comportamiento del sistema, así como también generar el código VHDL.

La programación del *PI* en *System Generator* se realiza en la ventana de Simulink, aquí se implementa la ecuación recursiva del controlador.

Posteriormente generamos el código VHDL, el código generado se guarda en un archivo con extensión .vhdl para luego ser unido a un código principal.

Diseñar el controlador en *System Generator* es una ventaja, ya que se obtiene el código VHDL de forma sencilla, aunque la optimización de código no es buena.

En la Fig. 4.41 se observa la programación de la ecuación recursiva del controlador PI en *System Generator*.

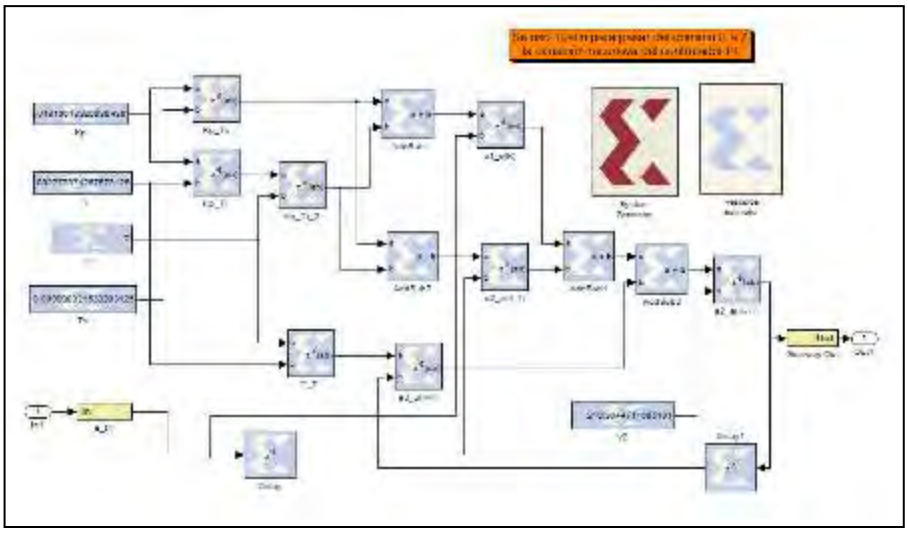


Fig. 4.41 Controlador PI usando *System Generator*

Se consideró una función de transferencia del sistema que se obtuvo por identificación.

La simulación se realizó tomando en cuenta la obtención del código *VHDL* del controlador, el cual será luego implementado en un FPGA. A esta se le dio un disturbio, así como un ruido blanco, Fig. 4.42.

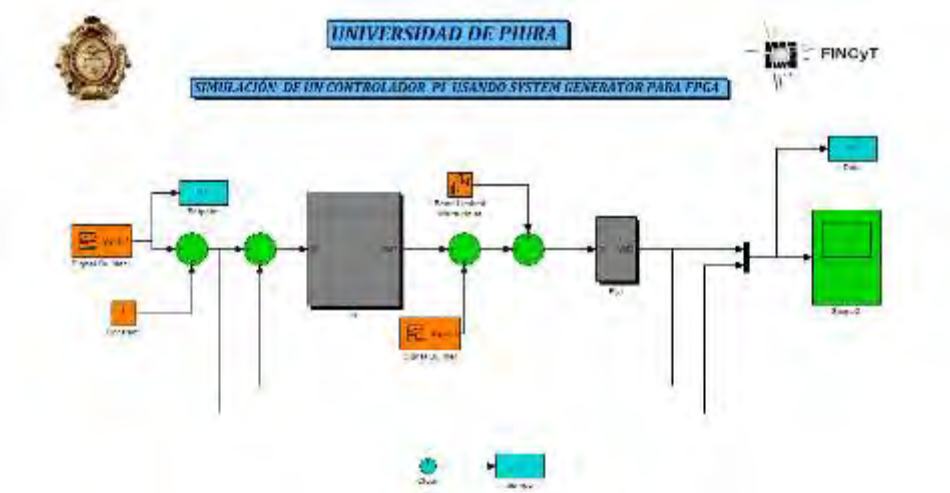


Fig. 4.42 Controlador PI en *System Generator*

En la gráfica siguiente, Fig. 4.43, se observa el comportamiento del controlador PI. En el momento que se presenta el disturbio, el sistema vuelve a su valor estacionario, llegando a controlar el proceso, este en el tercer cambio de referencia.

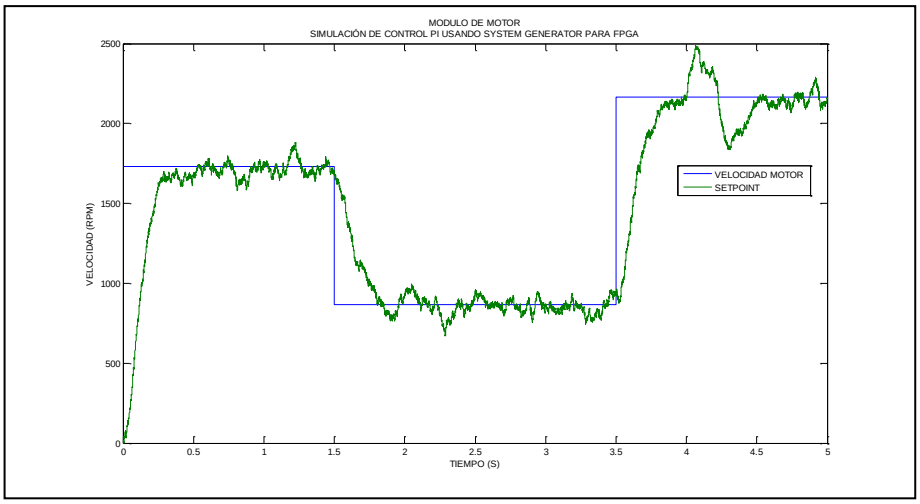


Fig. 4.43 Comportamiento del sistema a lazo cerrado

EL bloque que se muestra a continuación, Fig. 4.44, se encarga de compilar el código generando archivo *VHDL.*

Se ingresan datos de configuración, así como la familia del FPGA, la frecuencia de la señal de reloj, el lenguaje de descripción de hardware a utilizar; aquí nos muestra dos opciones, *VHDL y VERILOG,* usamos el primero ya que estamos familiarizados con este.

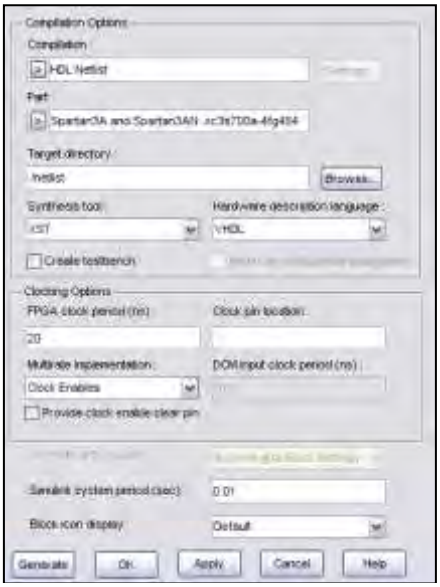


Fig. 4.44 Bloque que permite compilar a lenguaje *VHDL*

En la Fig. 4.45 vemos el código en *VHDL* del controlador PI luego de ser compilado en *System Generator.* Este bloque PI es unido al resto de bloques, como el *ADC, DAC, UART* por medio de una arquitectura Estructural.

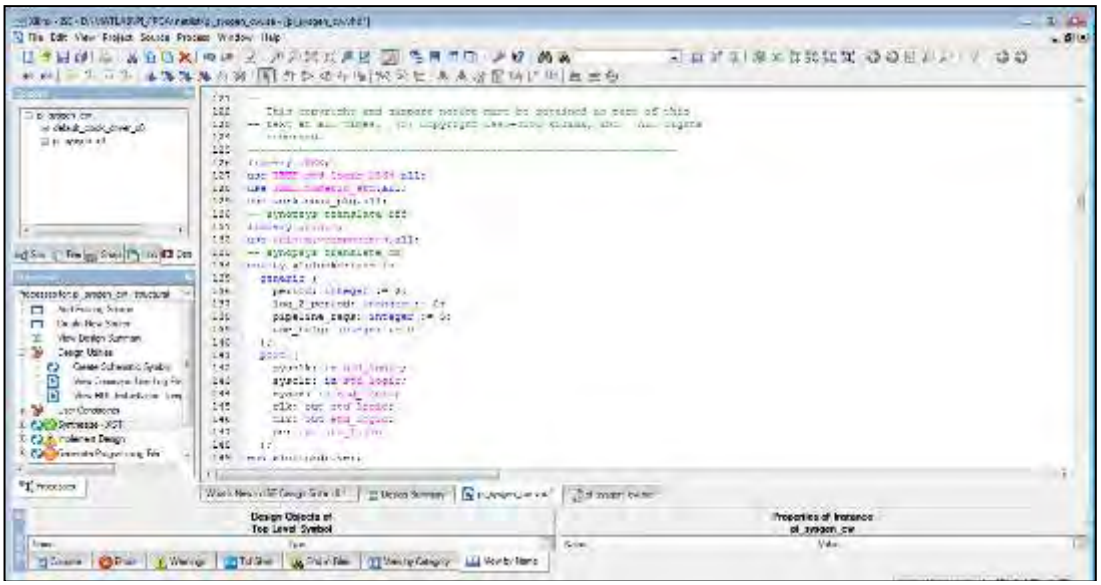


Fig. 4.45 Código *VHDL* del controlador PI

A continuación en la Fig. 4.46 y Fig. 4.47 se muestra la entidad del bloque, así como también la descripción del hardware; es decir el circuito que se implementará en el FPGA.

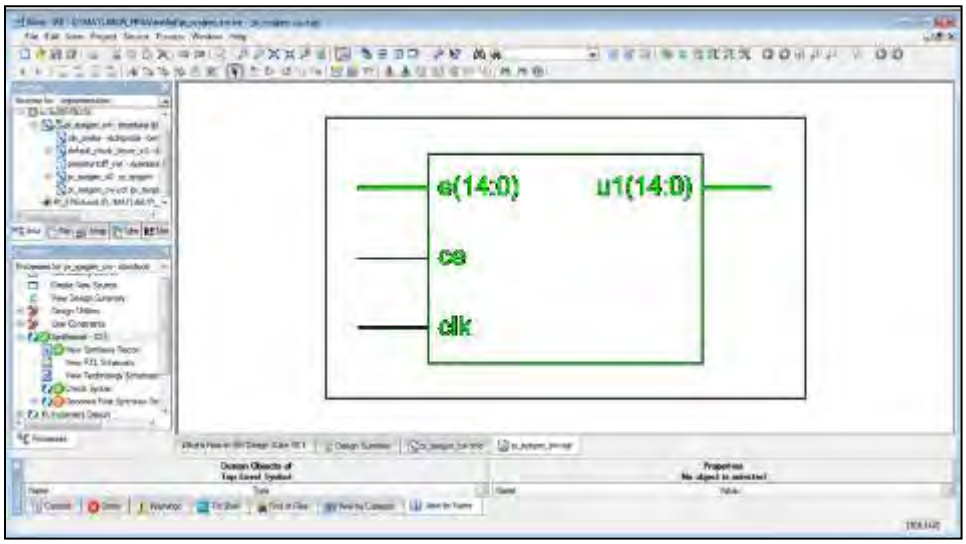


Fig. 4.46 Entidad del código de PI

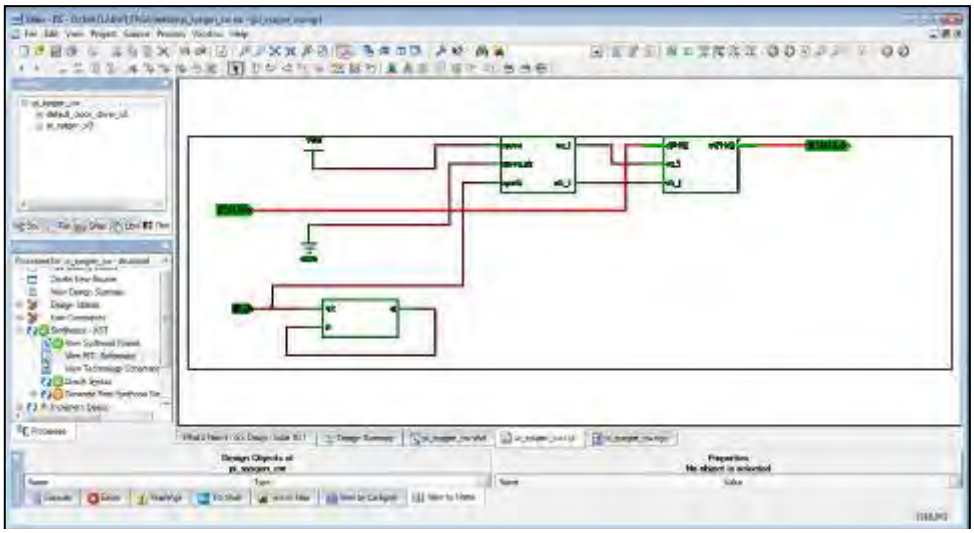


Fig. 4.47 ***Netlist*** **del controlador PI**

4.2.5 Entidad principal

Se desarrolló un bloque llamado *TOP GENERAL*, el cual une los bloques ya programados anteriormente como son, *ADC, DAC, UART, PI.* Estos se unen por medio de la arquitectura estructural.

En la Fig. 4.48, se muestra el código del bloque *TOP GENERAL* en *VHDL*.

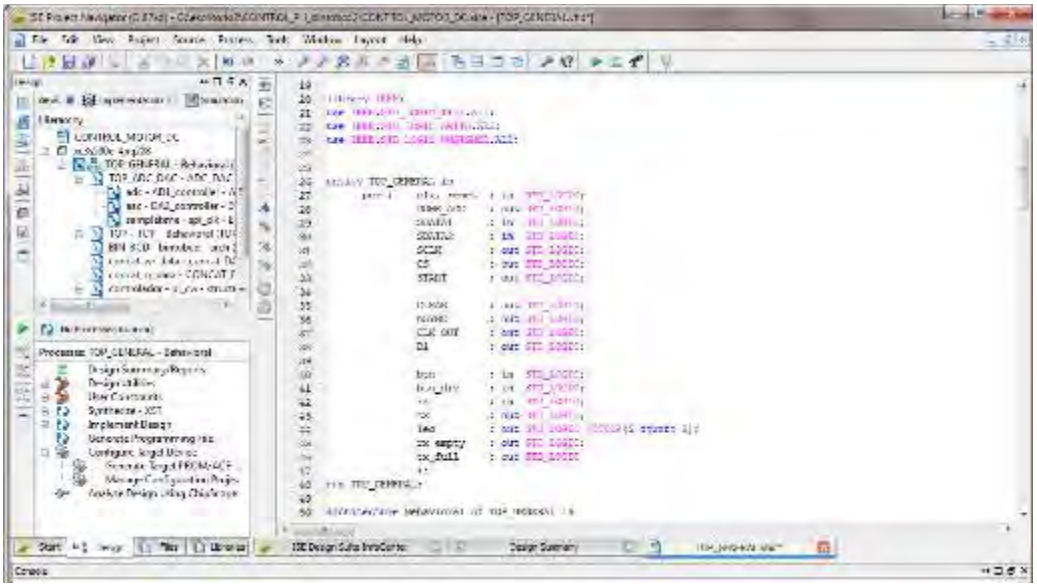


Fig. 4.48 Código del bloque de control TOP GENERAL

En la Fig. 4.49 se muestra la entidad del bloque *TOP GENERAL*, el cual cuenta con señales de entradas y salidas.

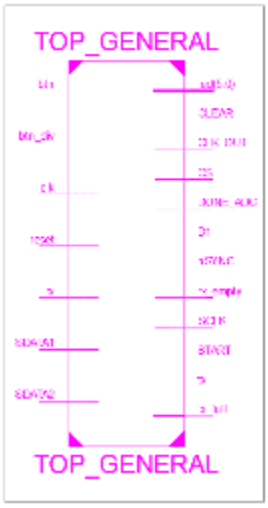


Fig. 4.49 Entidad del bloque de control PID

En la Fig. 4.50 vemos la descripción de hardware que se implementará en la FPGA; este agrupa el bloque *ADC*, el bloque *PI,* el bloque *DAC*, y comunicación *UART*.

Todos ellos se unen por medio de cables de interconexión programables, vistos en el capitulo anterior.

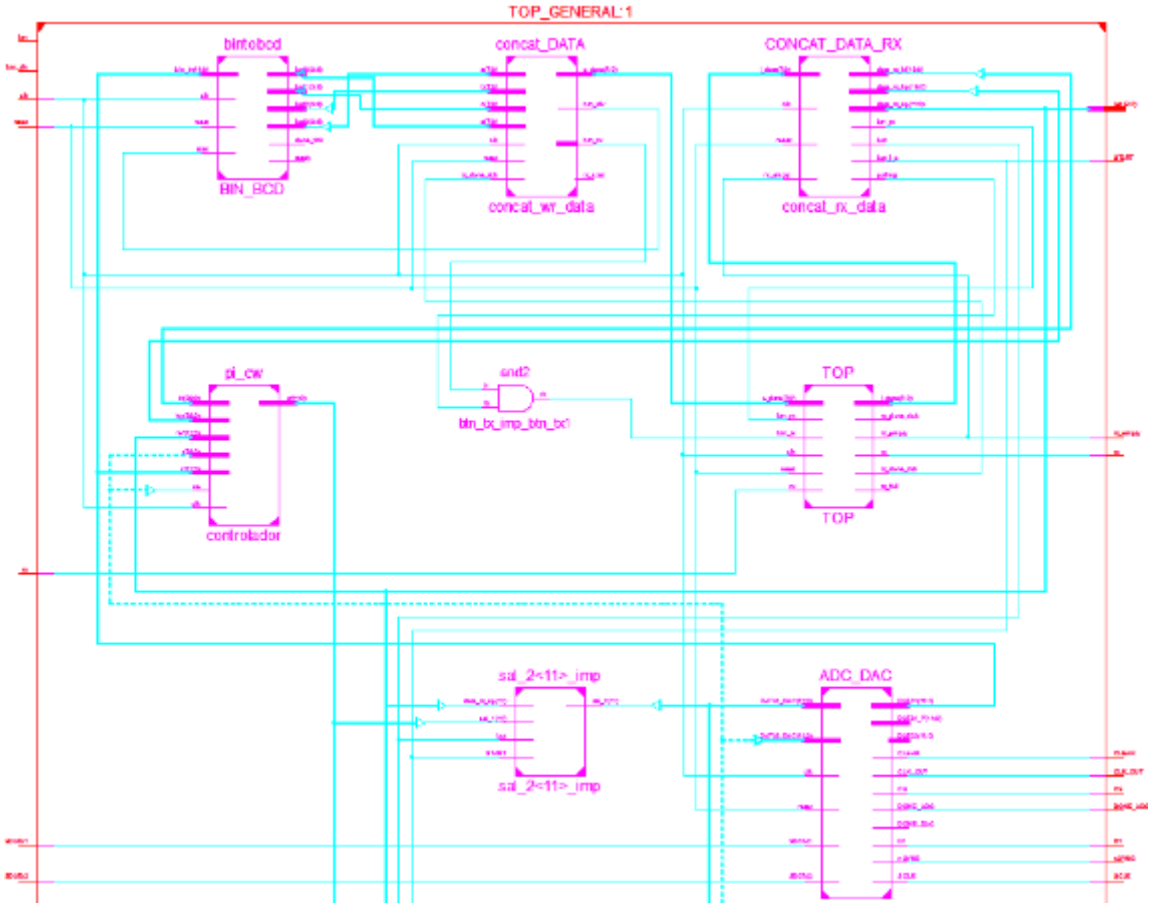


Fig. 4.50 *Netlist* del bloque *TOP GENERAL*

4.3 Breve descripción de la herramienta DSPACE

La tecnología *dSPACE* está formada por *hardware* y *software.* El software es un conjunto de programas informáticos que requiere el hardware para su correcto funcionamiento, además está compuesta por una serie de herramientas para llevar a cabo un proceso de control, cabe mencionar que estas herramientas trabajan en conjunto con *MatLab - Simulink.*

Cuenta con una interfaz de tiempo real (RTI), software para pruebas y experimentación; entre ellos *ControlDesk,* que es el que utilizaremos para simular nuestro proceso con señales reales de entrada y salida, software para medición y calibración; así como modelos para simulación en automoción.

En la parte de hardware, trabaja con una tarjeta integrada la cual se instala en la tarjeta madre de una computadora personal. Usaremos la tarjeta DS1104, la cual además hace uso de un panel (Panel de Conectores y LEDs) para hacer más fáciles las conexiones entre el sistema a controlar y la tarjeta.

4.3.1 Software controlDesk

Provee funciones de control, monitoreo y automatización experimental, por lo tanto hace más eficiente el desarrollo de control. Desde esta interfaz podemos controlar un proceso en tiempo real. Hace uso del diagrama de bloques en entorno *Simulink,* mediante las librerías de *dSPACE*.

Se observa en tiempo real como varían las respuestas de un proceso al variar algún parámetro de entrada.

4.3.2 DS1104 R&D controller Board

Es un hardware que se instala dentro de la tarjeta madre de la PC (requiere un sistema de 32-bit), ha sido diseñada para el desarrollo controladores digitales multi-variables de alta velocidad y para simulaciones en tiempo real en varios campos; complementa la PC para desarrollar sistemas capaces de realizar un control más rápido.

Sus interfaces de entrada y salida la hacen ideal para desarrollos de control muy variados. Su control y procesamiento en tiempo real es posible porque internamente está dividida en bloques funcionales que hacen más rápido el procesamiento. Además posee conversores analógicos-digitales y digitales-analógicos para facilitar la comunicación con el exterior, lo que posibilita la toma de datos en tiempo real.

4.3.3 CP1104 conector and led panel (CLP)

Provee acceso a las salidas y entradas de la tarjeta DS1104. Los dispositivos pueden ser fácilmente conectados o desconectados mediante conectores analógicos (de cable coaxial), o digitales (conectores sub-D). Además, el panel posee una serie de LED's en los cuales se aprecia el estado de las señales digitales.

4.4 Dispositivo ethernet WIZNET

El desarrollo de la comunicación Ethernet, se ha llevado a cabo a través de la configuración del módulo *WizNet110S*, Fig. 4.51, el cual está basado en el chip W5100, integra tanto la pila *TCP/IP* como la *MAC* y capa física de Ethernet. Este módulo funciona como un Gateway que convierte desde el protocolo RS-232 al protocolo *TCP/IP* y viceversa. Permite entre otras cosas dar conectividad Ethernet a los equipos que no lo poseen.

Fig. 4.51 Circuito RS232 – Ethernet

Este dispositivo se comunica al sistema *SCADA*, por medio de una conexión ethernet. Los datos que transmitirá serán la variable del proceso *(PV),* la señal de actuación *(MV),* el *setpoint (SP)* y los parámetros del controlador.

A continuación, en la Fig. 4.52, nos muestra la tarjeta *FPGA* con el componente *WIZNET*, trabaja en óptimo desempeño tanto la transmisión y recepción.

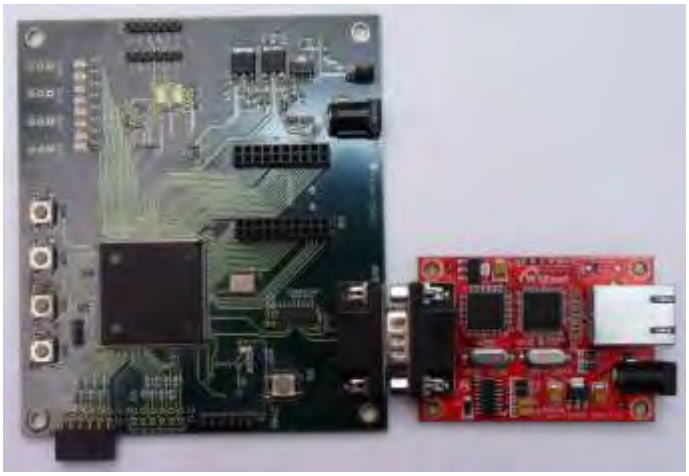


Fig. 4.52 Sistema PCB-FPGA conectado al dispositivo *WizNet*

4.5 Módulo de laboratorio

El módulo del laboratorio está compuesto por un conjunto de elementos de distintas funciones, entre estos componentes tenemos: 2 bombas, 1 Ph-metro, 1 agitador magnético, 1 electrodo, reactivos, entre otros.

El proceso consiste en un sistema de inyección continua de solución de un ácido fuerte *(HCl)* que empleará como variable de control el flujo de una base fuerte (*NaOH*). Al mezclarse estas dos sustancias reaccionan químicamente en el reactor y finalmente se obtiene como señal de salida el *pH* de la solución final.

Cada solución de ácido y base se agrega al reactor por medio de una bomba independiente, que en el caso de la bomba de *NaOH* se regula con niveles de corriente entre 4-20 mA.

4.5.1 Bomba del ácido

Bomba dosificadora proporcional ON-OFF, marca *Walchem,* modelo EZB30D2VC, cual se muestra en la Fig. 4.53. La serie EZ cuenta con capacidades de hasta 24 LPH y presiones hasta 150 PSI. Su diseño compacto y práctico de control digital la hacen ideal para tratamiento de agua (coagulación/floculación y desinfección), calderos y torres de enfriamiento.

Esta bomba mantendrá un flujo constante de solución ácida *(acido clorhídrico HCl*) durante todo momento, brinda un flujo máximo de 0.2 L/min, puede ser regulado manualmente.

Fig. 4.53 Bomba del ácido

4.5.2 Bomba de la base

Bomba proporcional dosificadora, marca *Walchem*, modelo *EHE35E2-VC*, Fig. 4.54, con regulación manual según perilla y con excitación externa, Fig. 4.55. Trabaja con base, NaOH.

Las bombas EHE ofrecen caudales hasta 76 LPH y presiones de descarga hasta 150 PSI (10 Bar). Sus circuitos basados en microprocesador permiten al operador ajustar la velocidad de la bomba y establecer una señal de entrada de 4-20 mA a través del panel digital. La serie EHE viene con modo de control externo análogo y digital.

Su flujo teórico máximo es 0.52 L/min, pero al igual que la bomba de ácido el flujo real máximo puede ser mayor desde un 70% hasta un 100% más, por eso se regulará con la entrada de corriente.

Fig. 4.54 Bomba de base

4.5.3 PH-metro

El electrodo es de la marca *HANNA Instrument*, modelo HI 1006-3205, cuerpo de vidrio, mide de 0 a 14 pH. Temperatura de operación ente 0 a 100 °C, se utilizará a temperatura ambiente, aproximadamente 25°C. De mantenimiento húmedo.

Es necesario para calibración y limpieza se utilicen los siguientes buffers.

Fig. 4.56 Buffers de Calibración y Mantenimiento

Las soluciones reguladoras o *"buffer"* son capaces de mantener la acidez o basicidad de un sistema dentro de un intervalo reducido de pH. Estas soluciones contienen como especies predominantes, un par ácido/base conjugado en concentraciones apreciables.

4.5.4 Transductor de pH

Dispositivo transductor de marca *Hanna Instrument*, convierte la señal de baja potencia que recibe de la sonda de *pH* y la convierte en una señal analógica entre 4-20 mA, la cual es enviada al dispositivo controlador, siendo esta la señal del proceso *(PV).* Ver Fig. 4.57.

La temperatura también influye en la lectura del *pH* pero como se trabajará con temperatura constante, se utiliza una resistencia de 100 ohm.

Fig. 4.57 Transductor de pH

Características:

Ideal para el control constante de pH en procesos industriales.

Salida en corriente 4-20mA.

Indicadores LED que identifican el modo de funcionamiento.

Ideal para el proceso de control de pH.

LCD integrado.

Rango de medición entre 0 a 14 de pH. Resolución de 0.01.

Posee una prueba de auto-diagnóstico para probar el estado del electrodo y el instrumento.

4.5.5 Agitador magnético

Elemento muy importante para este módulo, gracias a una cápsula imantada que gira dentro del reactor se puede lograr la uniformidad de la solución ó producto de la mezcla de acido y base logrando un valor de pH homogéneo. Ver Fig. 4.58.

Características:

Regulación de velocidad de hasta 1200 rpm.

Blindado contra emisión de radiofrecuencias.

Estructura metálica recubierta con pintura epoxi

Potencia: 40 W

Voltaje: 115 V o 230 V / 50-60 Hz

Peso: 1,8 Kg

Dimensiones (LxHxP): 171x75x190 mm

Fig. 4.58 Agitador magnético

4.5.6 Reactor

Es donde se lleva a cabo la reacción química, entre el acido y la base, tratando de conseguir una mezcla homogénea. Cuenta con líneas de entrada y salida para sustancias químicas que en este caso son el acido clorhídrico *HCl* y el hidróxido de sodio *NaOH*.

Otro elemento importante es la tubería de descarga, que nos ayuda a mantener una misma altura de la solución final en el reactor, ya que este debe de trabajar a un volumen constante.

Fig. 4.59 Reactor

4.5.7 Reactivos

Como soluciones para la reacción se utilizarán ácido clorhídrico e hidróxido de sodio, con las que se logra una reacción de ácido y base fuertes con generación de sal y agua como producto final.

Ácido clorhídrico *(HCl):*

- Tanto el cloruro de hidrógeno como el ácido clorhídrico son corrosivos. El *HCl* es muy corrosivo por sus vapores en concentraciones altas (mayores al 10%).

- En altas concentraciones puede provocar irritaciones y quemaduras, además de causar problemas respiratorios por exposición prolongada de estos vapores. Es recomendable que su almacenamiento se haga con botellas de vidrio.

- Es un ácido muy utilizado en la industria por ser relativamente barato y efectivo.

- Se emplea comúnmente como reactivo químico y se trata de un ácido fuerte que se disocia completamente en disolución acuosa.

Hidróxido de Sodio *(NaOH):*

- Es una base fuerte, segura y barata en estado sólido, pero muy soluble. Es por ello que es fácil de manejar.

- Es necesario disolverla en agua destilada obteniendo una solución básica de acuerdo al nivel de molaridad que se desee.

- La mejor forma de almacenar esta base es utilizando envases plásticos en estado sólido, y además se debe tener en cuenta que a mayor exposición del *NaOH*, mayor será su absorción de *CO2* de la atmosfera y de este forma disminuye su nivel de *pH.*

- El hidróxido de *sodio (NaOH)* o hidróxido sódico, también conocido como sosa cáustica o soda cáustica, es usado en la industria (principalmente como una base química) en la fabricación de papel, tejidos, y detergentes, jabones, crayón, papel, explosivos, pinturas y productos de petróleo. También se usa en el procesamiento de textiles de algodón, lavandería y blanqueado. Se encuentra comúnmente en limpiadores de desagües y hornos.

4.5.8 Reservorios

Se utilizarán un balde de 20 L. para cada solución de ácido y base con agua destilada.

Fig. 4.60 Reservorios de *NaoH* y *HCl*

Capítulo 5

Validación experimental: control de pH

5.1 Identificación de módulo de pH

La identificación del sistema se utilizó el modelo validado [6] que se cuenta, usando el sistema *DSPACE* en *MatLab.* Este permitió contar con un modelo SISO; la entrada de caudal de la solución base, es expresada en niveles de voltaje entre 0-3.3 V, la salida de pH expresada en los mismos niveles de voltaje.

Estas señales son conectadas a la tarjeta *FPGA,* en los puertos DAC y ADC respectivamente.

La tarjeta FPGA cuenta con comunicación Ethernet, por donde envía las señales de la variable del proceso *(PV).* El controlador se encuentra inmerso en el chip FPGA y es el encargado de mantener esta variable del proceso en un valor de referencia, por medio de un algoritmo de control avanzado; en este caso un *PI* con *Gain Scheduling*.

Se cuenta con un sistema *SCADA* para supervisión, control y almacenamiento de datos del proceso, así también cuenta con configuración de alarmas.

5.1.1 Identificación no paramétrica

El experimento de identificación se desarrolló por medio de entradas de tipo escalón, Los datos de entrada y salida fueron guardados en archivos de tipo plano (.txt) los cuales fueron importados posteriormente al software Matlab para su análisis.

Las figuras siguientes muestran la respuesta del proceso ante una señal de tipo escalón. La variable manipulable (entrada) y la variable del proceso (salida) respectivamente.

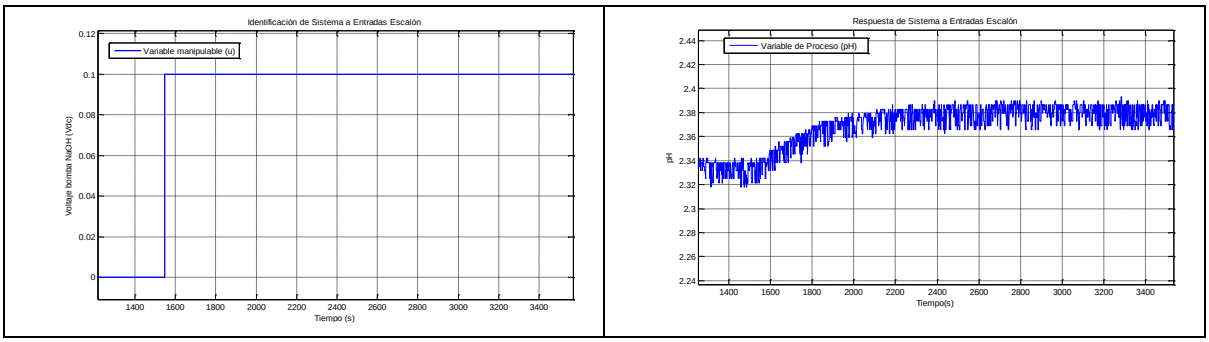


Fig. 5.1 Respuesta del sistema ante entrada escalón de 0 - 0.1 Vdc

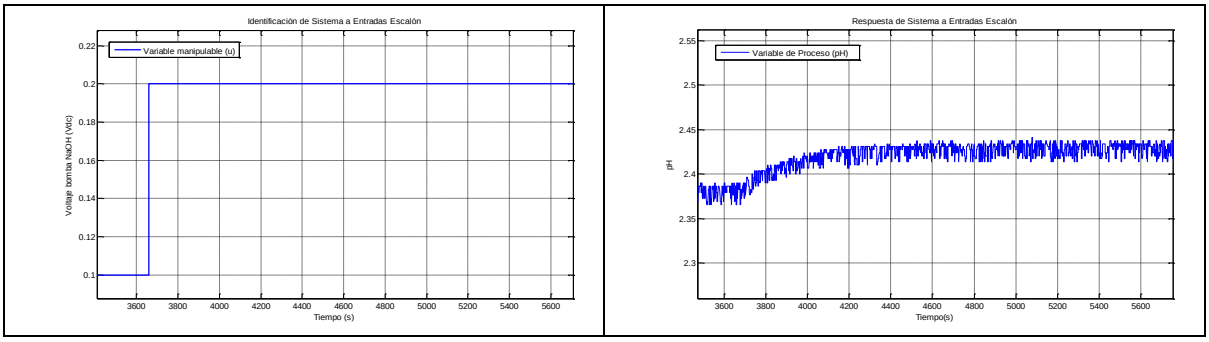


Fig. 5.2 Respuesta del sistema ante entrada escalón de 0.1 - 0.2 Vdc

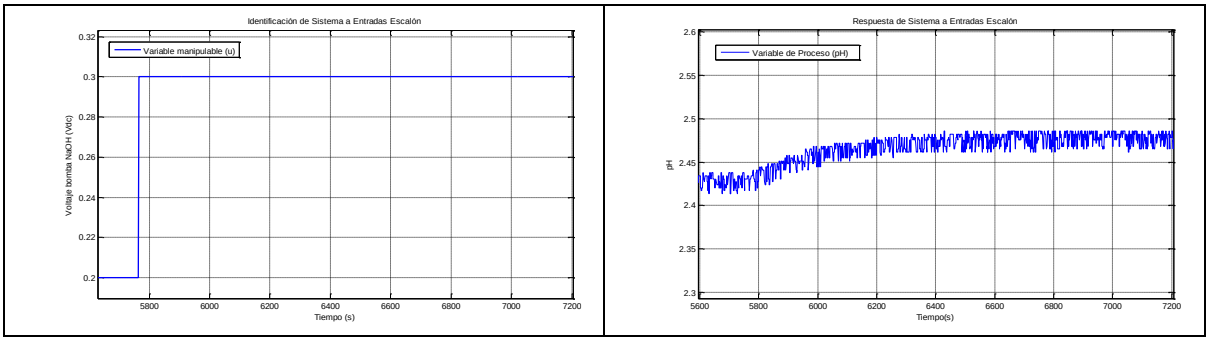


Fig. 5.3 Respuesta del sistema ante entrada escalón de 0.2 - 0.3 Vdc

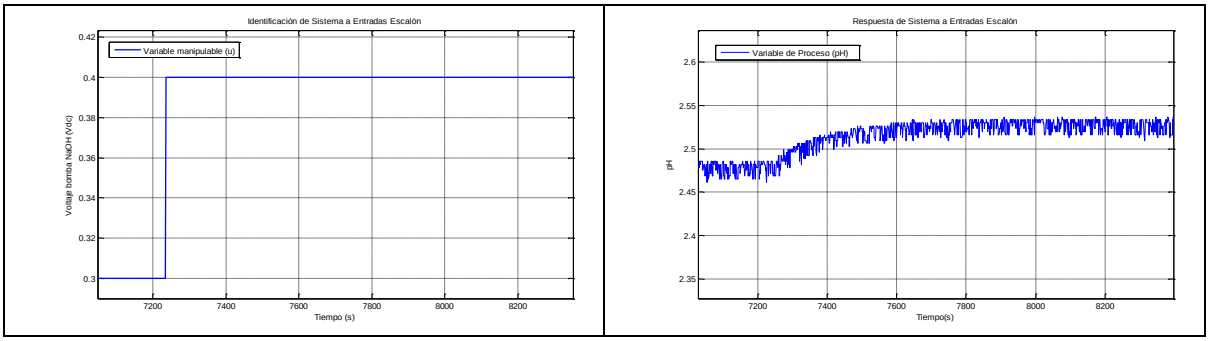


Fig. 5.4 Respuesta del sistema ante entrada escalón de 0.3 - 0.4 Vdc

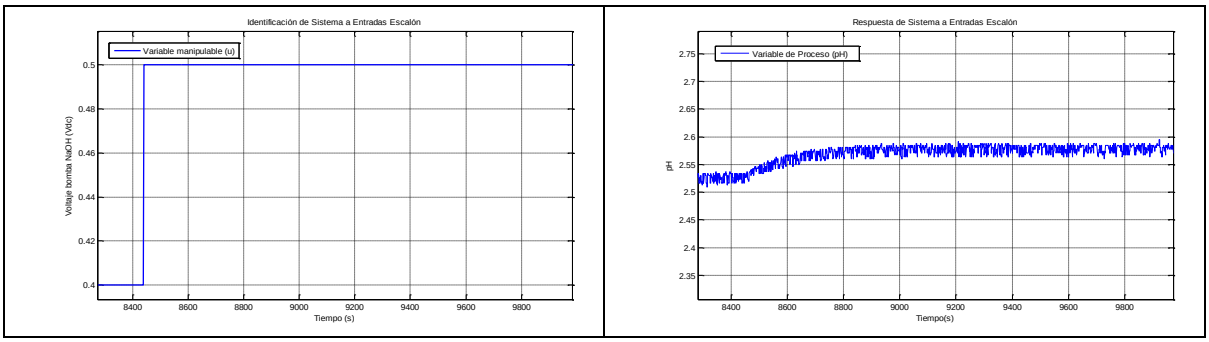


Fig. 5.5 Respuesta del sistema ante entrada escalón de 0.4 - 0.5 Vdc

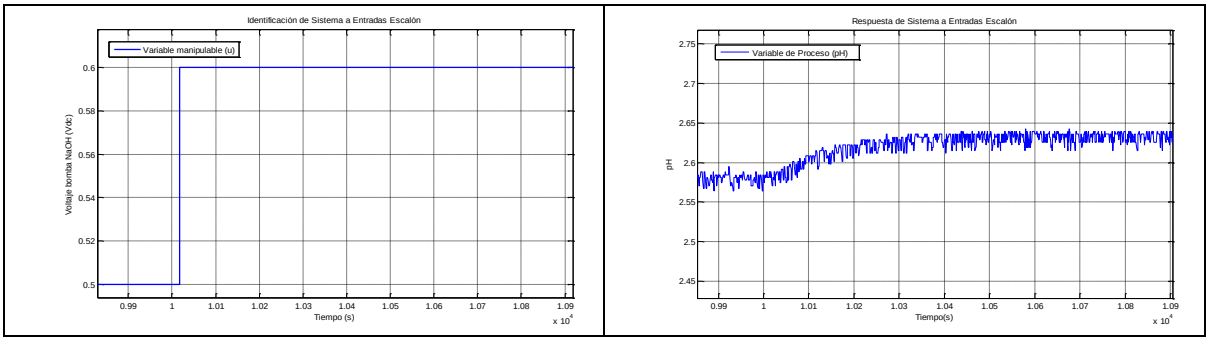


Fig. 5.6 Respuesta del sistema ante entrada escalón de 0.5 - 0.6 Vdc

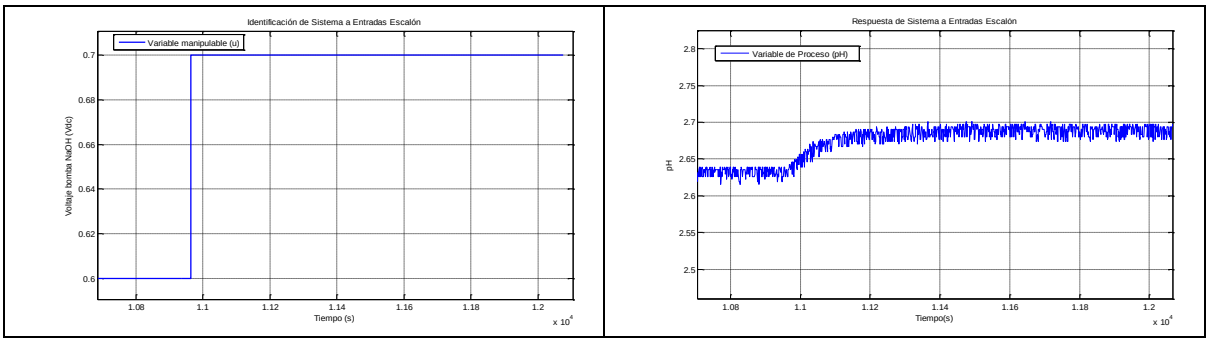


Fig. 5.7 Respuesta del sistema ante entrada escalón de 0.6 - 0.7 Vdc

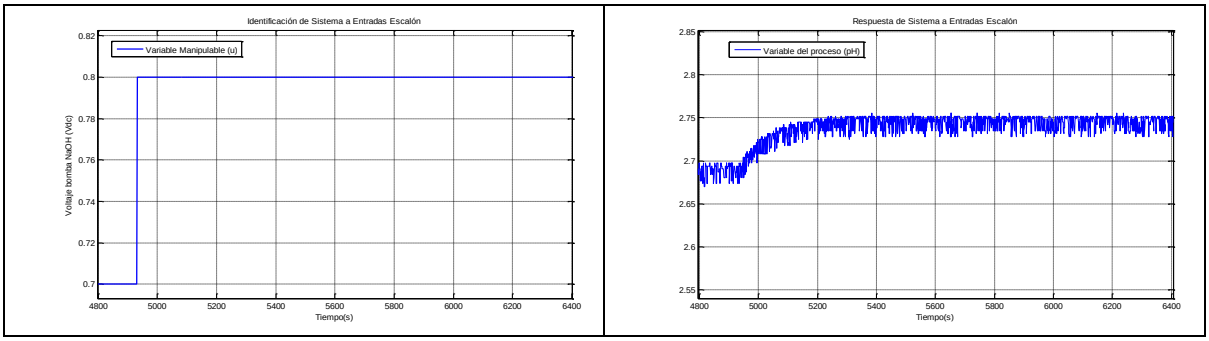


Fig. 5.8 Respuesta del sistema ante entrada escalón de 0.7 - 0.8 Vdc

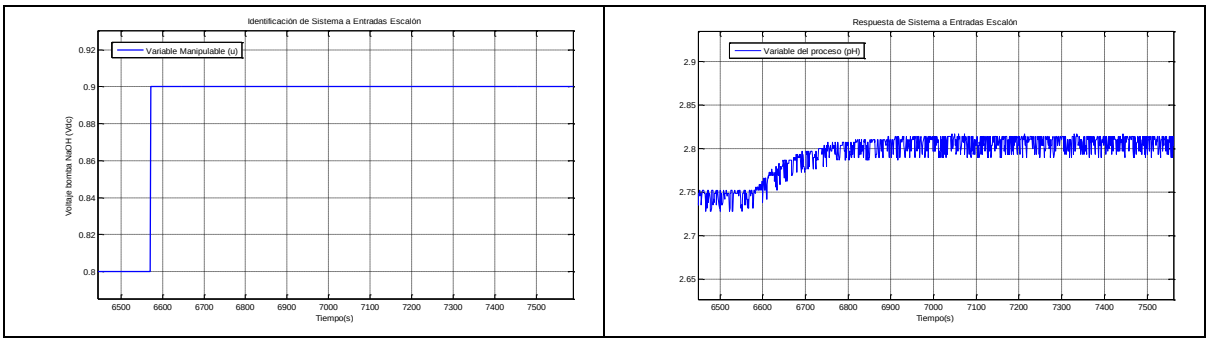


Fig. 5.9 Respuesta del sistema ante entrada escalón de 0.8 - 0.9 Vdc

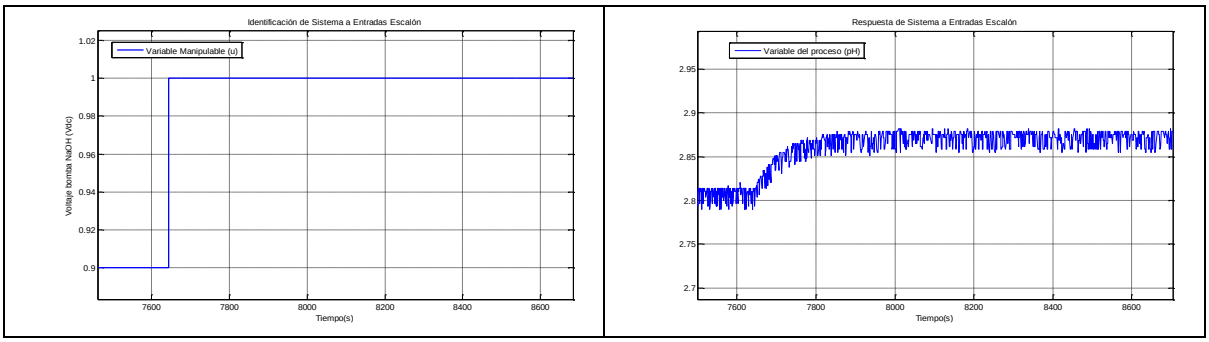


Fig. 5.10 Respuesta del sistema ante entrada escalón de 0.9 - 1 Vdc

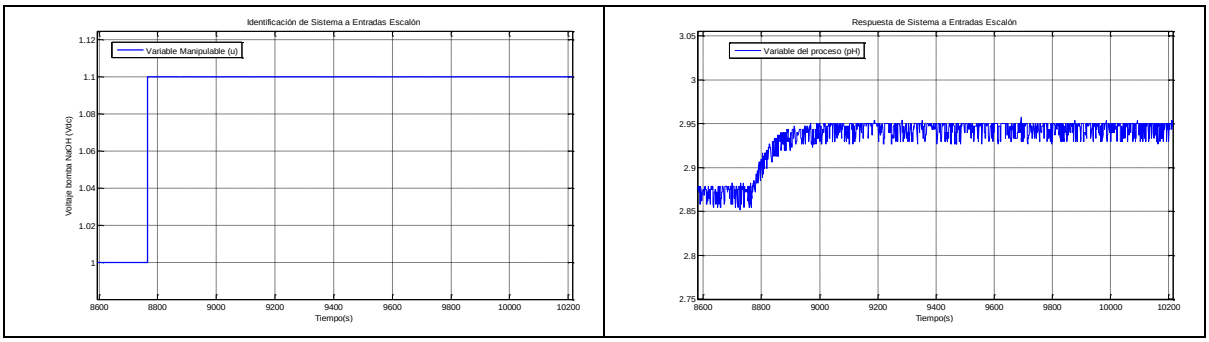


Fig. 5.11 Respuesta del sistema ante entrada escalón de 1 - 1.1 Vdc

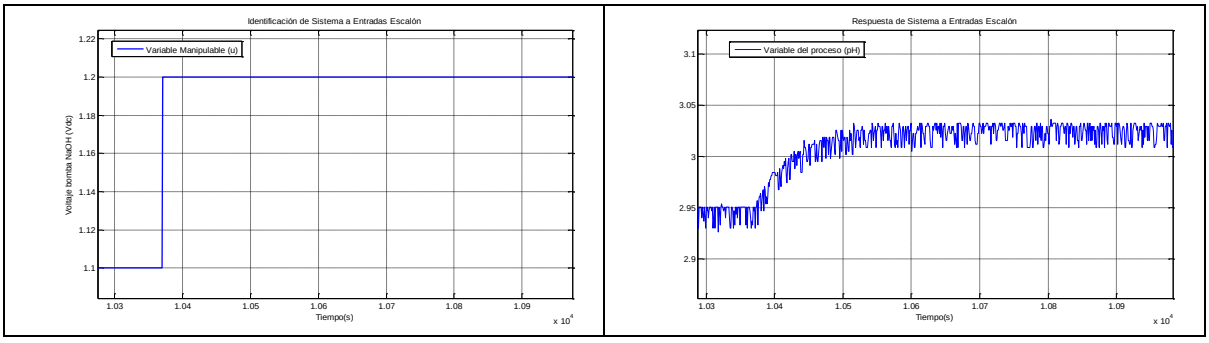


Fig. 5.12 Respuesta del sistema ante entrada escalón de 1.1 - 1.2 Vdc

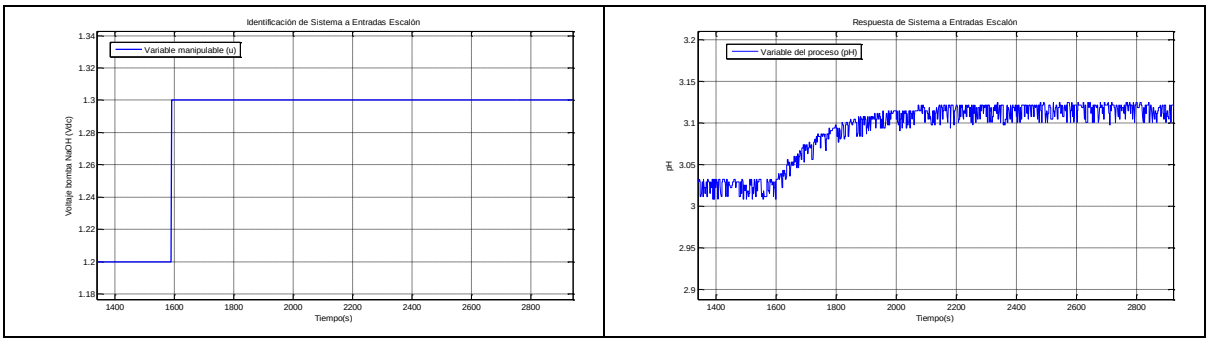


Fig. 5.13 Respuesta del sistema ante entrada escalón de 1.2 - 1.3 Vdc

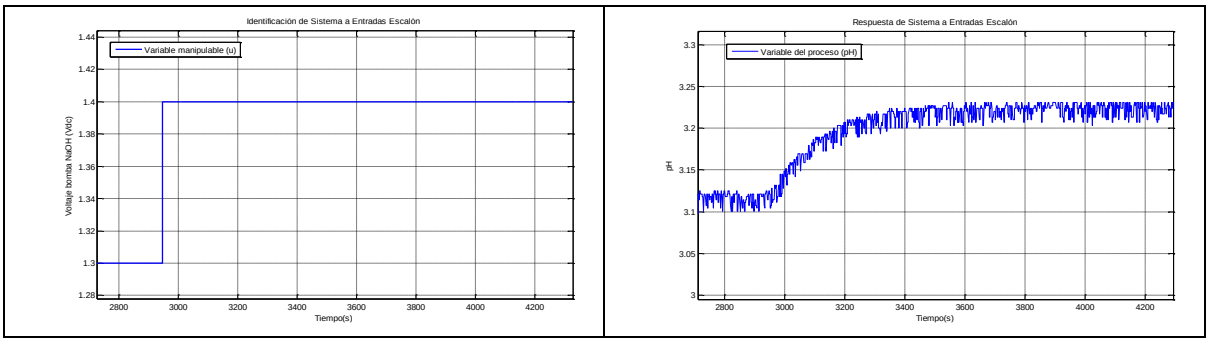


Fig. 5.14 Respuesta del sistema ante entrada escalón de 1.3 - 1.4 Vdc

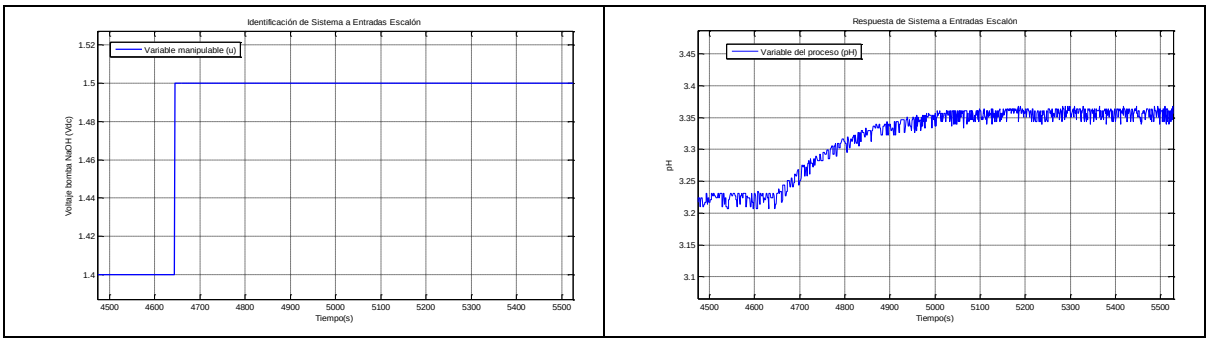


Fig. 5.15 Respuesta del sistema ante entrada escalón de 1.4 - 1.5 Vdc

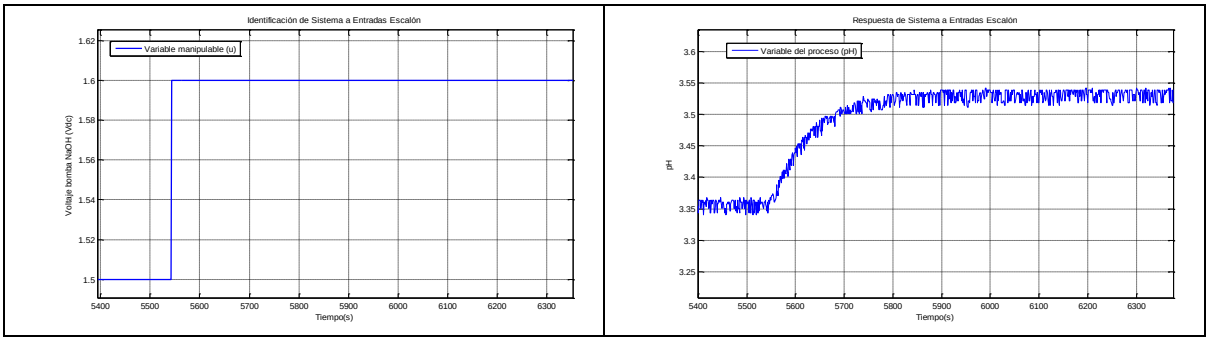


Fig. 5.16 Respuesta del sistema ante entrada escalón de 1.5 - 1.6 Vdc

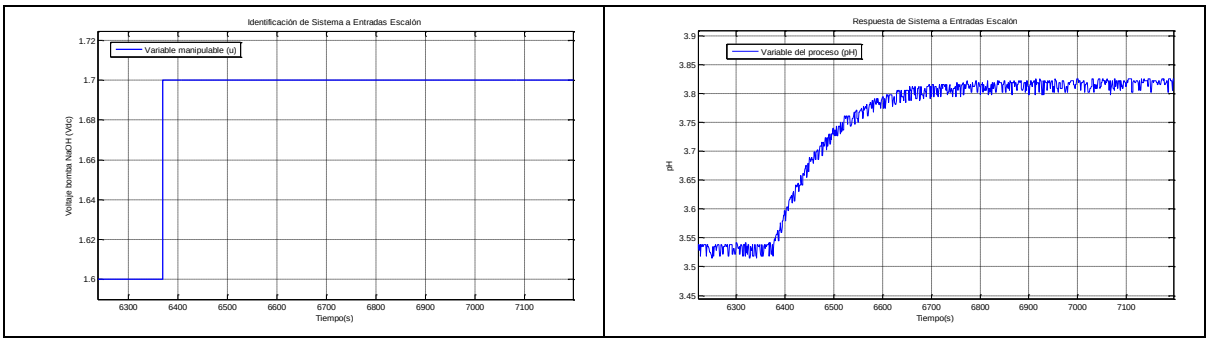


Fig. 5.17 Respuesta del sistema ante entrada escalón de 1.6 - 1.7 Vdc

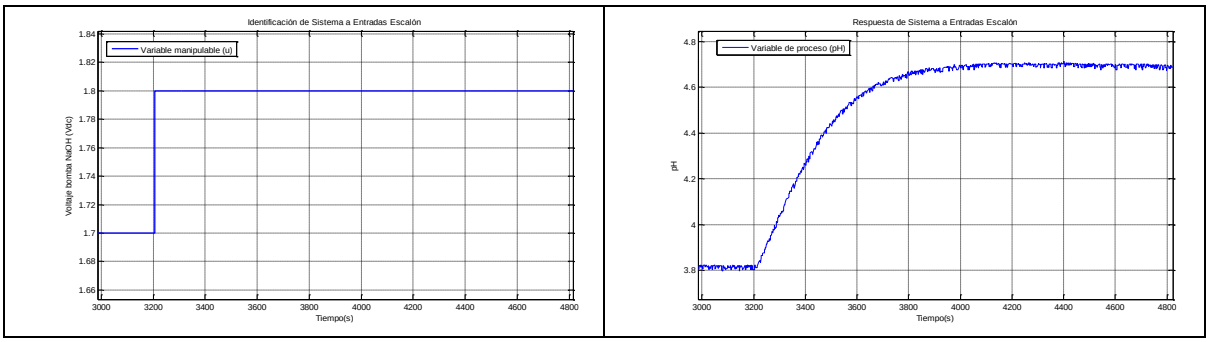


Fig. 5.18 Respuesta del sistema ante entrada escalón de 1.7 - 1.8 Vdc

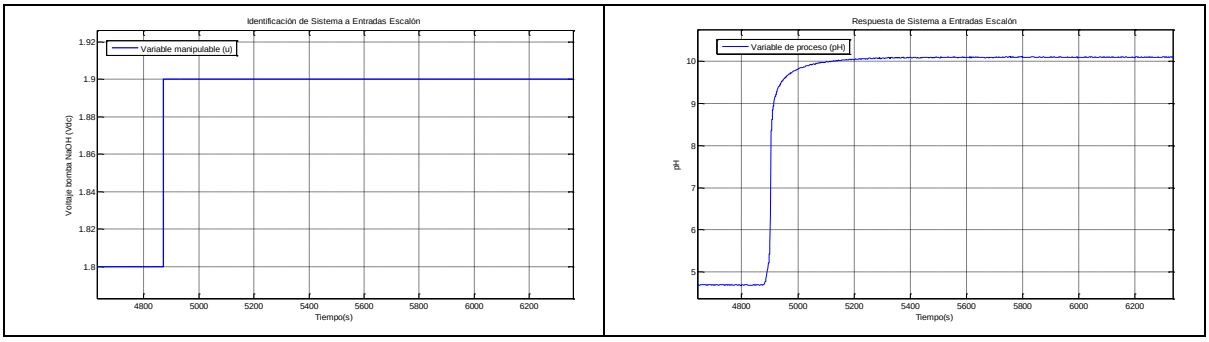


Fig. 5.19 Respuesta del sistema ante entrada escalón de 1.8 - 1.9 Vdc

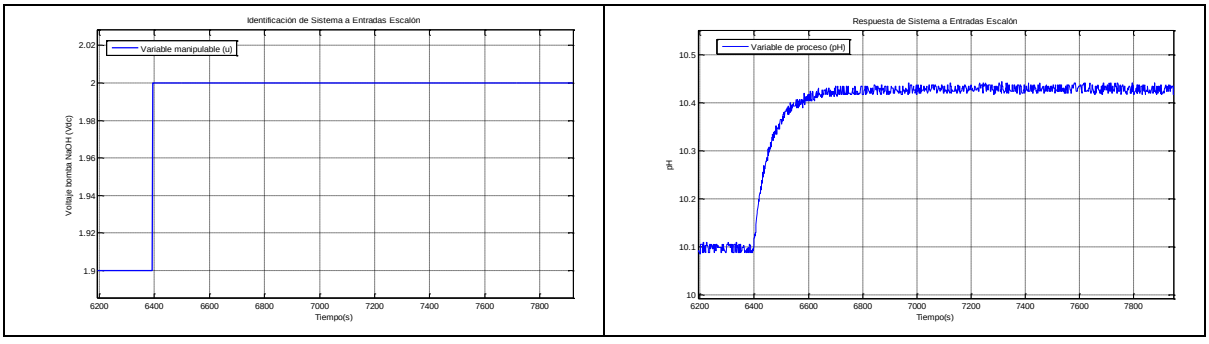


Fig. 5.20 Respuesta del sistema ante entrada escalón de 1.9 - 2 Vdc

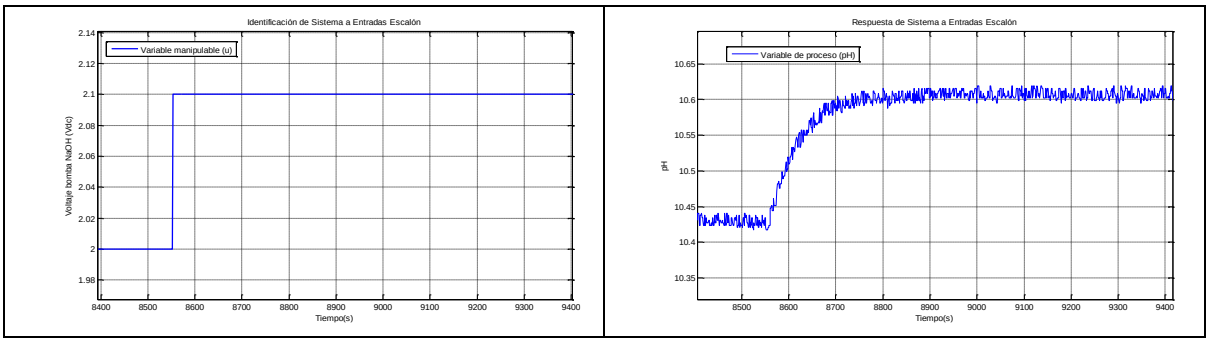


Fig. 5.21 Respuesta del sistema ante entrada escalón de 2 - 2.1 Vdc

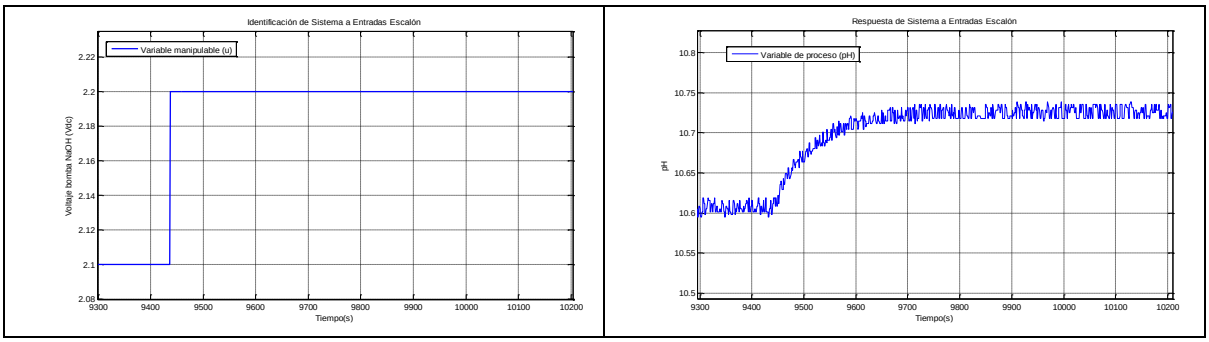


Fig. 5.22 Respuesta del sistema ante entrada escalón de 2.1 – 2.2 Vdc

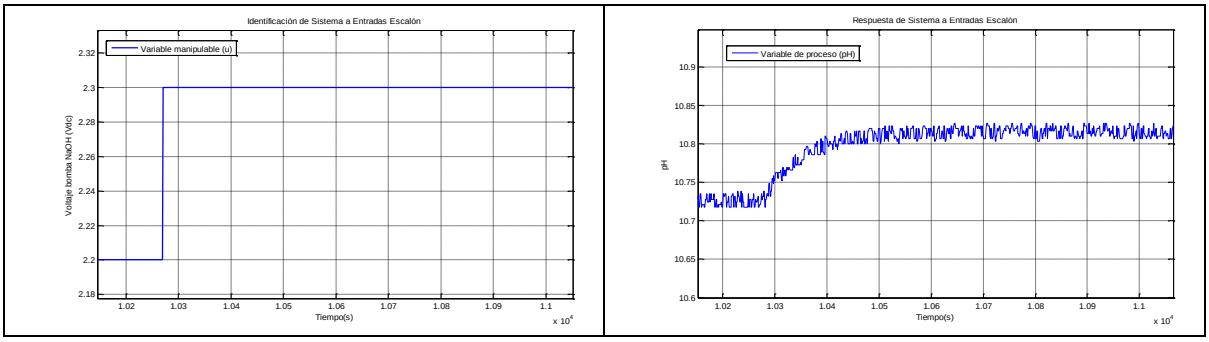


Fig. 5.23 Respuesta del sistema ante entrada escalón de 2.2 – 2.3 Vdc

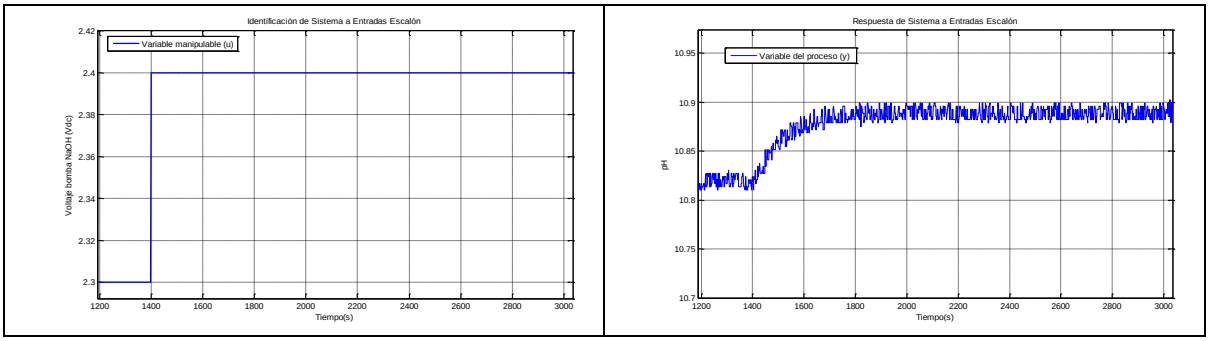


Fig. 5.24 Respuesta del sistema ante entrada escalón de 2.3 – 2.4 Vdc

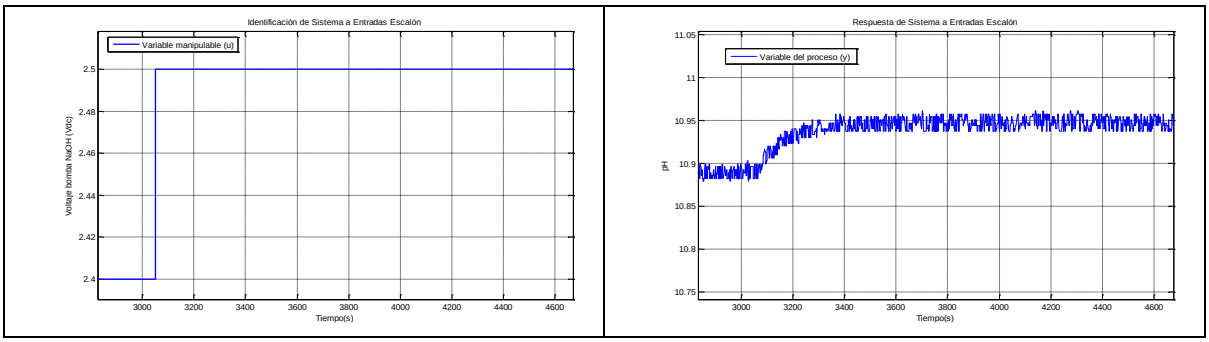


Fig. 5.25 Respuesta del sistema ante entrada escalón de 2.4 – 2.5 Vdc

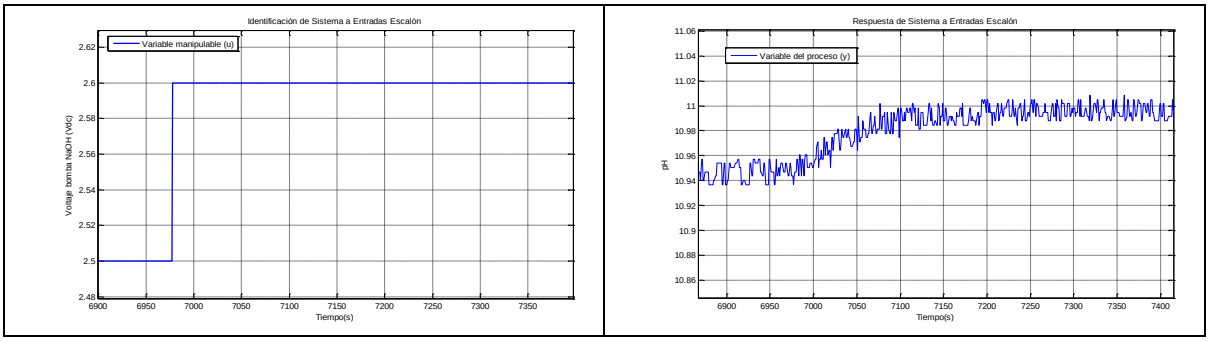


Fig. 5.26 Respuesta del sistema ante entrada escalón de 2.5 – 2.6 Vdc

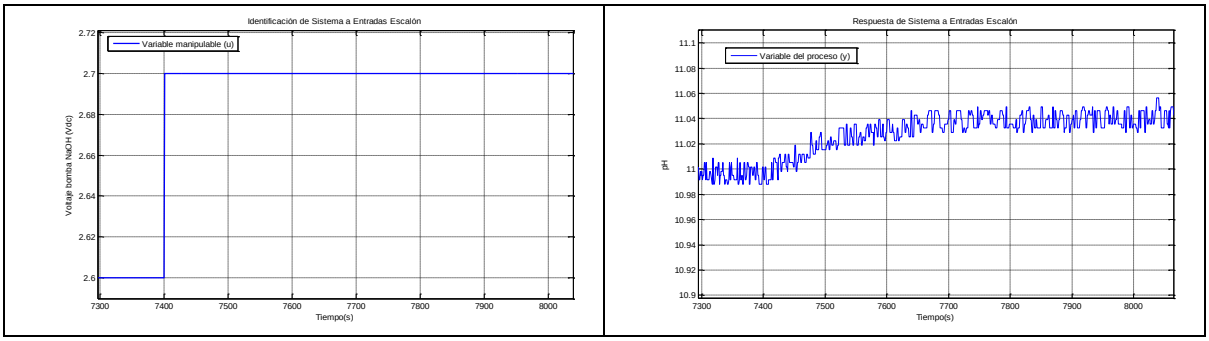


Fig. 5.27 Respuesta del sistema ante entrada escalón de 2.6 – 2.7 Vdc

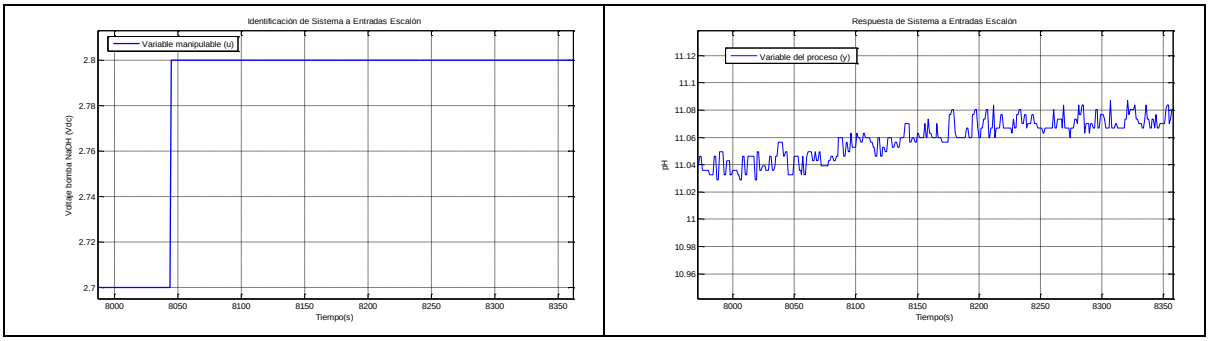


Fig. 5.28 Respuesta del sistema ante entrada escalón de 2.7 – 2.8 Vdc

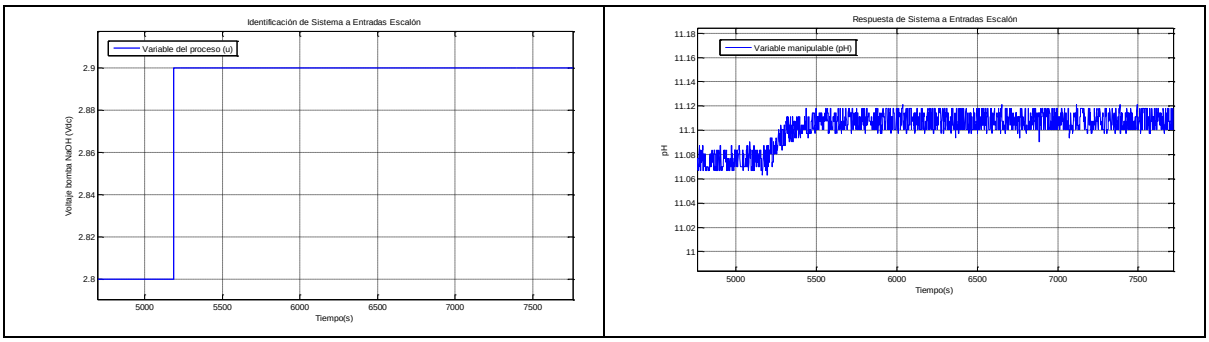


Fig. 5.29 Respuesta del sistema ante entrada escalón de 2.8 – 2.9 Vdc

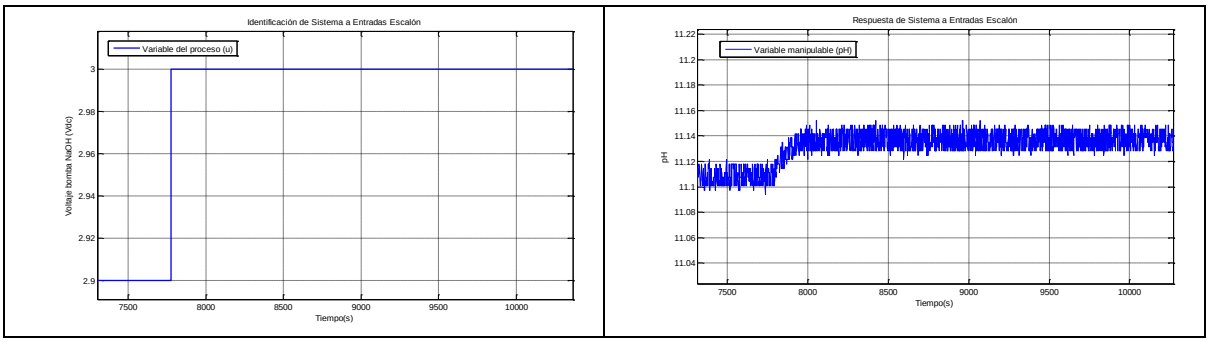


Fig. 5.30 Respuesta del sistema ante entrada escalón de 2.9 – 3 Vdc

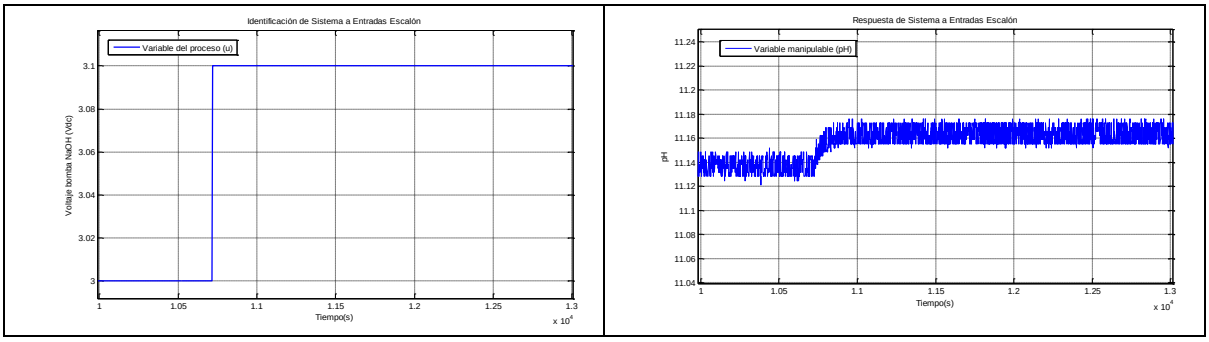


Fig. 5.31 Respuesta del sistema ante entrada escalón de 3 – 3.1 Vdc

5.1.2 Estudio de la ganancia estática

La información del experimento de identificación se ha almacenado en una base de datos, la cual se configura desde el sistema *SCADA*.

A continuación se muestra en la Tabla 5.1 las ganancias estáticas del sistema.

Tabla 5.1 Ganancias estáticas del proceso

u_i	u_f	Δu	y_i	y_f	Δy	K
0	0.1	0.1	2.33	2.38	0.05	0.5
0.1	0.2	0.1	2.38	2.43	0.05	0.5
0.2	0.3	0.1	2.43	2.475	0.045	0.45
0.3	0.4	0.1	2.475	2.525	0.05	0.5
0.4	0.5	0.1	2.525	2.575	0.05	0.5
0.5	0.6	0.1	2.575	2.63	0.055	0.55
0.6	0.7	0.1	2.63	2.69	0.06	0.6
0.7	0.8	0.1	2.69	2.74	0.05	0.5
0.8	0.9	0.1	2.74	2.8	0.06	0.6
0.9	1	0.1	2.8	2.87	0.07	0.7
1	1.1	0.1	2.87	2.94	0.07	0.7
1.1	1.2	0.1	2.94	3.02	0.08	0.8
1.2	1.3	0.1	3.02	3.11	0.09	0.9
1.3	1.4	0.1	3.11	3.22	0.11	1.1
1.4	1.5	0.1	3.22	3.35	0.13	1.3
1.5	1.6	0.1	3.35	3.53	0.18	1.8
1.6	1.7	0.1	3.53	3.82	0.29	2.9
1.7	1.8	0.1	3.82	4.7	0.88	8.8
1.8	1.9	0.1	4.7	10.1	5.4	54
1.9	2	0.1	10.1	10.41	0.31	3.1
2	2.1	0.1	10.43	10.6	0.17	1.7
2.1	2.2	0.1	10.6	10.73	0.13	1.3
2.2	2.3	0.1	10.73	10.81	0.08	0.8
2.3	2.4	0.1	10.81	10.89	0.08	0.8
2.4	2.5	0.1	10.89	10.95	0.06	0.6
2.5	2.6	0.1	10.95	11	0.05	0.5
2.6	2.7	0.1	11	11.04	0.04	0.4
2.7	2.8	0.1	11.04	11.075	0.035	0.35
2.8	2.9	0.1	11.075	11.11	0.035	0.35
2.9	3	0.1	11.11	11.14	0.03	0.3
3	3.1	0.1	11.14	11.165	0.025	0.25

En la figura siguiente graficamos la ganancia estática (K) respecto a la variable manipulable *(MV);* esto para analizar la no-linealidad del sistema.

En la Fig. 5.32, se aprecia que el sistema tiene un comportamiento no-lineal, con una marcada característica no-lineal entre valores de la variable manipulable de 1.8, que corresponden en torno de *pH* de 7.

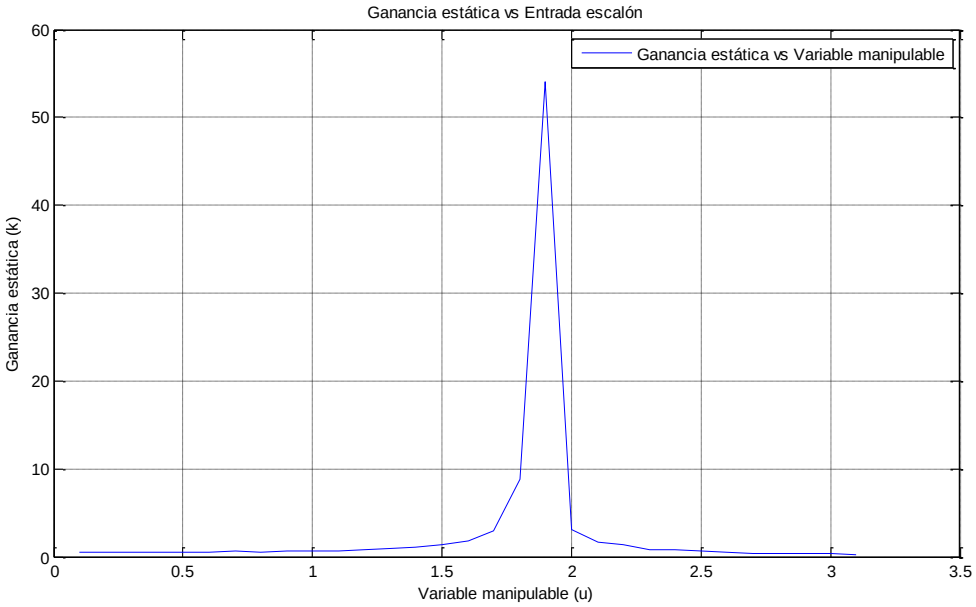


Fig. 5.32 Ganancia estática respecto a la variable manipulable

5.1.3 Obtención de modelos

Los modelos obtenidos a continuación corresponden a los rangos del experimento de identificación.

En la Tabla 5.2 se presentan las funciones de transferencia de los diferentes rangos.

Tabla 5.2 Funciones de transferencia

Tramo	Rango	T(s)
1	0-0.1	$\frac{0.5}{253s+1}$
2	0.1-0.2	$\frac{0.5}{261s+1}$
3	0.2-0.3	$\frac{0.45}{276s+1}$
4	0.3-0.4	$\frac{0.5}{116s+1}$
5	0.4-0.5	$\frac{0.5}{111s+1}$
6	0.5-0.6	$\frac{0.55}{113s+1}$
7	0.6-0.7	$\frac{0.6}{117s+1}$
8	0.7-0.8	$\frac{0.5}{94s+1}$
9	0.8-0.9	$\frac{0.6}{89s+1}$
10	0.9-1	$\frac{0.7}{87s+1}$

11	1-1.1	$\frac{0.7}{65s+1}$
12	1.1-1.2	$\frac{0.8}{60s+1}$
13	1.2-1.3	$\frac{0.9}{131s+1}$
14	1.3-1.4	$\frac{1.1}{154s+1}$
15	1.4-1.5	$\frac{1.3}{157s+1}$
16	1.5-1.6	$\frac{1.8}{98s+1}$
17	1.6-1.7	$\frac{2.9}{111s+1}$
18	1.7-1.8	$\frac{8.8}{244s+1}$
19	1.8-1.9	$\frac{54}{33s+1}$
20	1.9-2	$\frac{3.1}{58s+1}$
21	2-2.1	$\frac{1.7}{58s+1}$
22	2.1-2.2	$\frac{1.3}{79s+1}$
23	2.2-2.3	$\frac{0.8}{75s+1}$
24	2.3-2.4	$\frac{0.8}{91s+1}$
25	2.4-2.5	$\frac{0.6}{90s+1}$
26	2.5-2.6	$\frac{0.5}{93s+1}$
27	2.6-2.7	$\frac{0.4}{80s+1}$
28	2.7-2.8	$\frac{0.35}{79s+1}$
29	2.8-2.9	$\frac{0.35}{93s+1}$
30	2.9-3	$\frac{0.3}{94s+1}$
31	3-3.1	$\frac{0.25}{84s+1}$

La tabla 5.2 nos muestra las funciones de transferencia obtenidas en diferentes regiones. El experimento de identificación ha sido desarrollado dando escalones con incrementos de 0.1 Vdc. Hemos adquirido 31 tramos en total.

Se observa en las funciones de transferencia obtenidas, que las ganancias estáticas se mantienen entre el tramo 1 al 8, luego van aumentando hasta llegar a un máximo, en el tramo 19, para luego disminuir.

Mientras que las constantes de tiempo, *TAO,* tienen un comportamiento diferente, en los tramos 1 al 3 las constantes de tiempo van aumentando, luego disminuyen; pasando el rango de alta ganancia estática, tramo 19, estos podrían considerarse como constantes.

El sistema, por lo tanto, tiene un comportamiento altamente no lineal. Con una no-linealidad estática y dinámica. Por lo que la estrategia de control adaptativa como un controlador *PI* con ganancia programable *(Gain Scheduling)* daría buenos resultados.

La implementación de este tipo de regulador en controladores existentes, así como en PLC's, es factible, siendo una estrategia de control automático para procesos no lineales.

En el tramo que la ganancia es altamente no-lineal, se ha preparado un experimento de identificación con incrementos, en la variable manipulable mucho más pequeños. Estas pruebas junto con su desarrollo están presentes en el **Anexo D**.

5.2 Sintonización de controlador PID

Para la sintonización de los parámetros del regulador *PI*, en diferente rango de operación, se utilizó el método de asignación de polos.

En la figura a continuación se muestra la simulación del sistema a lazo cerrado con las funciones de transferencia encontradas.

Para ello se ha trabajado en el software *MATLAB - SIMULINK*.

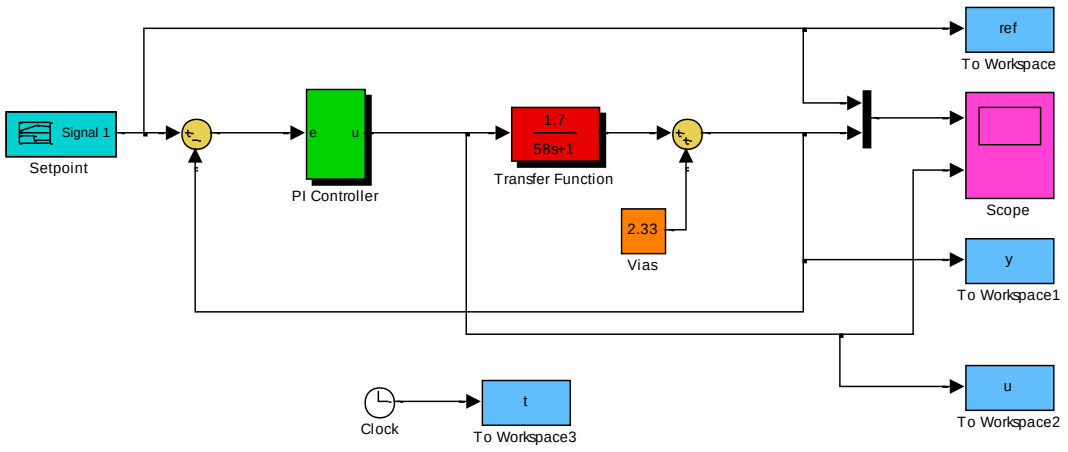


Fig. 5.33 Simulación en MatLab - Simulink

Las gráficas a continuación muestran las simulaciones del sistema a lazo cerrado, se han sintonizado parámetros del regulador PI para cada tramo.

Estos parámetros serán posteriormente embebidos dentro de la tarjeta de control FPGA.

La simulación obtenida en las gráficas siguientes representan los parámetros de control con mejores resultados, tanto en tiempo de establecimiento como en porcentaje de sobre oscilación.

Tramo 1:

En este tramo correspondiente a valores comprendidos de pH entre 2.33 a 2.47. Se consideró la siguiente función de transferencia:

$$T(s) = \frac{0.5}{261s + 1}$$

Los parámetros del controlador son los siguientes:

Kp = 8.6 Ti = 169.5

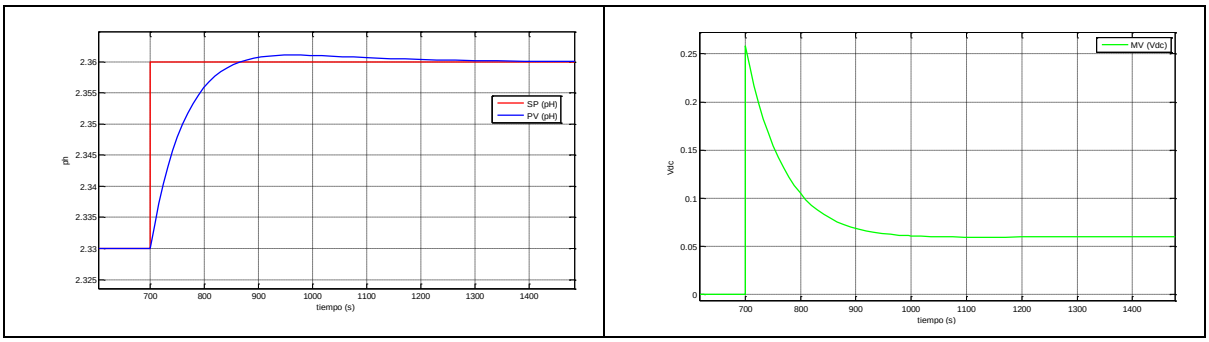


Fig. 5.34 Respuesta del sistema ante un cambio de setpoint de pH de 2.36

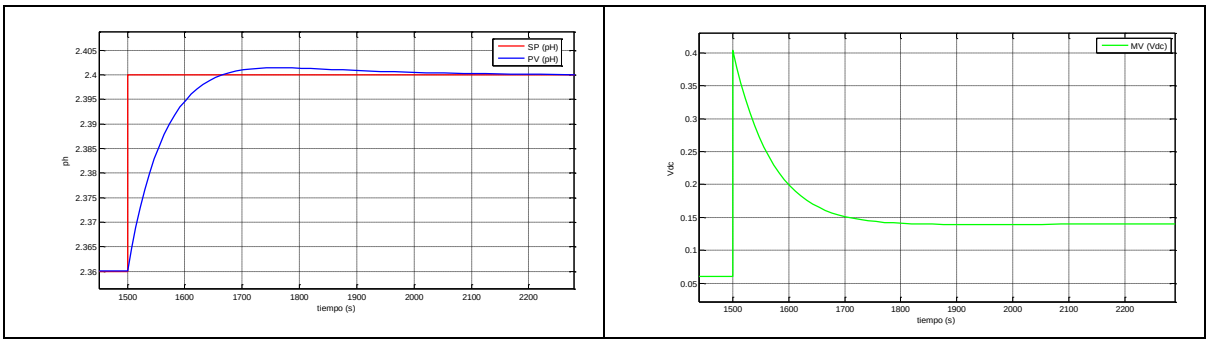


Fig. 5.35 Respuesta del sistema ante un cambio de setpoint de pH de 2.4

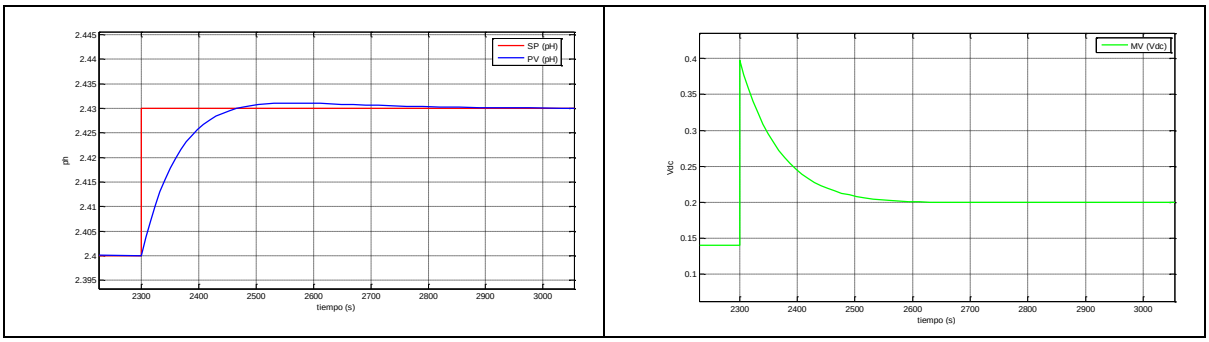


Fig. 5.36 Respuesta del sistema ante un cambio de setpoint de pH de 2.43

Tramo 2:

En este tramo correspondiente a valores comprendidos de pH entre 2.47 a 2.69. Se consideró la siguiente función de transferencia:

$$T(s) = \frac{0.5}{111s + 1}$$

Los parámetros del controlador son los siguientes:

Kp = 7.22 Ti = 76.05

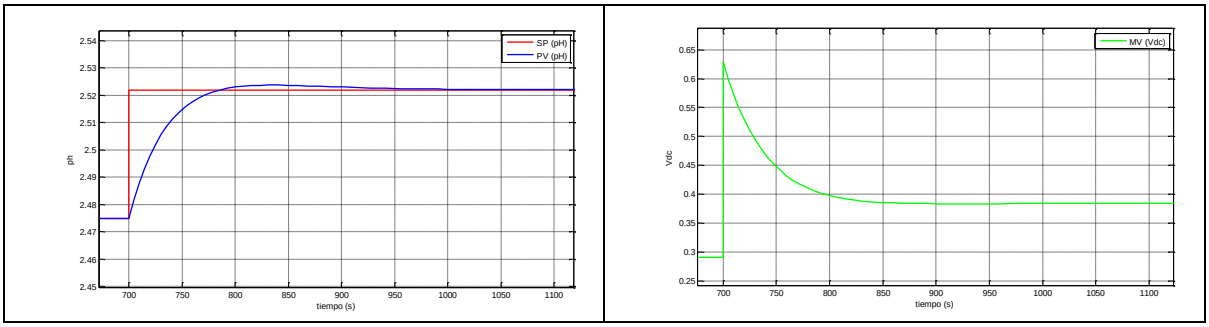


Fig. 5.37 Respuesta del sistema ante un cambio de setpoint de pH de 2.522

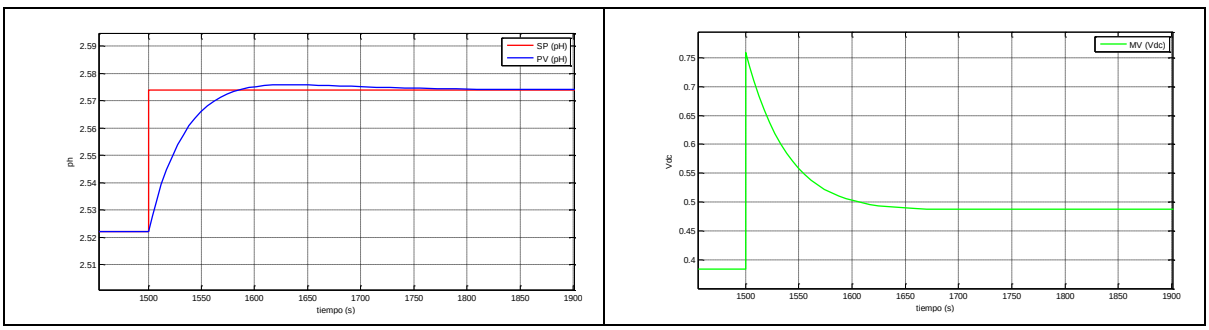


Fig. 5.38 Respuesta del sistema ante un cambio de setpoint de pH de 2.575

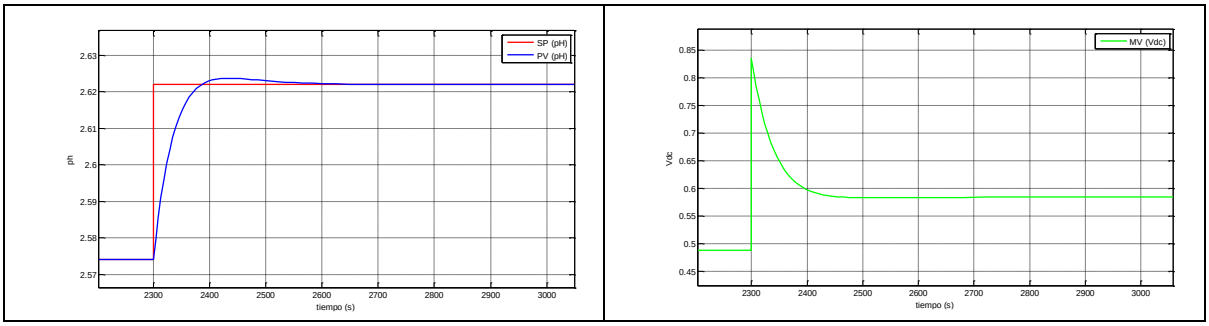


Fig. 5.39 Respuesta del sistema ante un cambio de setpoint de pH de 2.622

Tramo 3:

En este tramo correspondiente a valores comprendidos de pH entre 2.69 a 2.87. Se consideró la siguiente función de transferencia:

$$T(s) = \frac{0.6}{90s + 1}$$

Los parámetros del controlador son los siguientes:

Kp = 7.22 Ti = 57.85

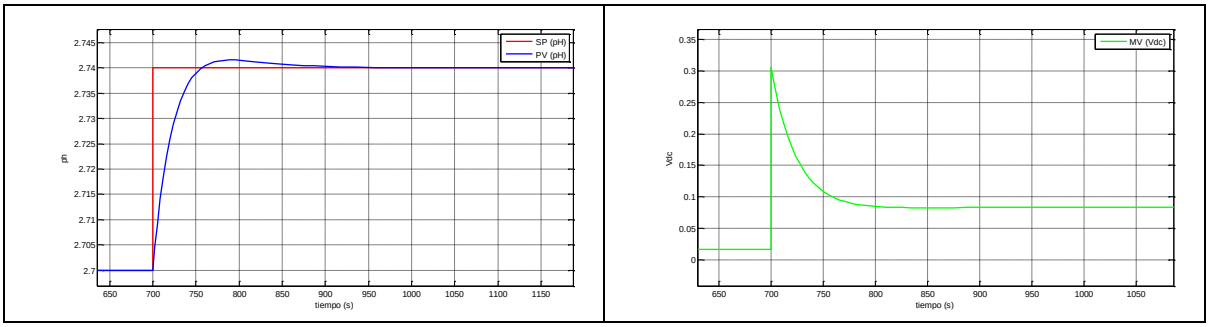


Fig. 5.40 Respuesta del sistema ante un cambio de setpoint de pH de 2.74

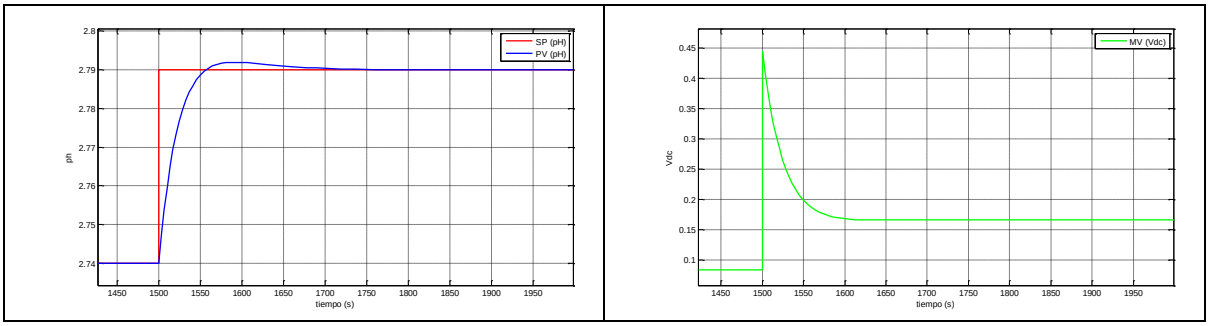


Fig. 5.41 Respuesta del sistema ante un cambio de setpoint de pH de 2.79

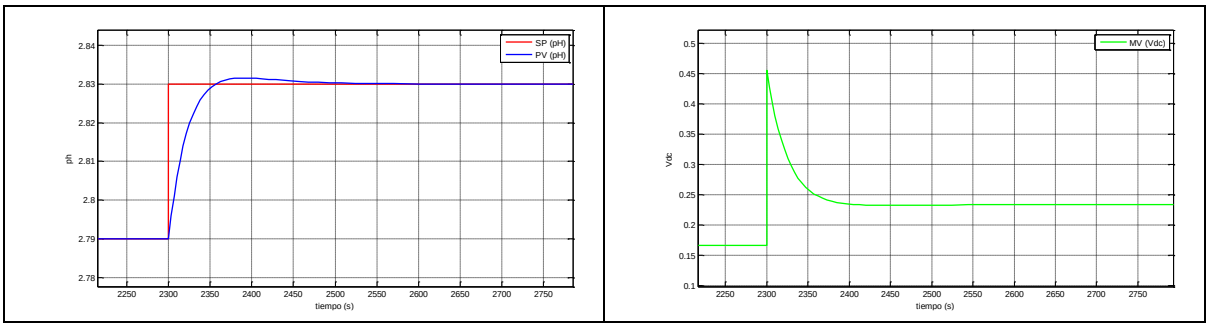


Fig. 5.42 Respuesta del sistema ante un cambio de setpoint de pH de 2.83

Tramo 4:

En este tramo correspondiente a valores comprendidos de pH entre 2.87 a 3.02. Se consideró la siguiente función de transferencia:

$$T(s) = \frac{0.8}{60s + 1}$$

Los parámetros del controlador son los siguientes:

Kp = 5.42 Ti = 39

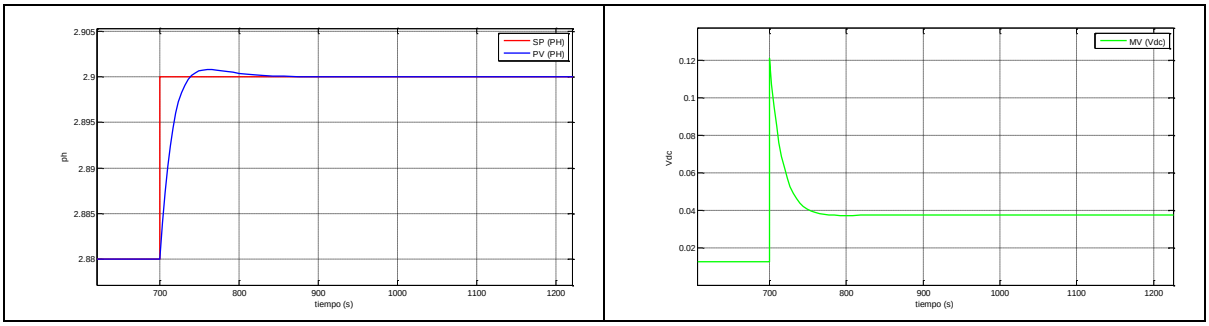


Fig. 5.43 Respuesta del sistema ante un cambio de setpoint de pH de 2.9

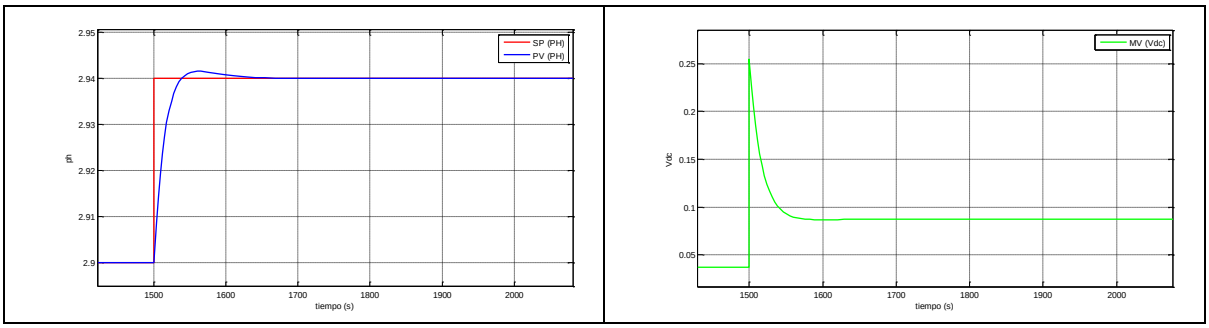


Fig. 5.44 Respuesta del sistema ante un cambio de setpoint de pH de 2.94

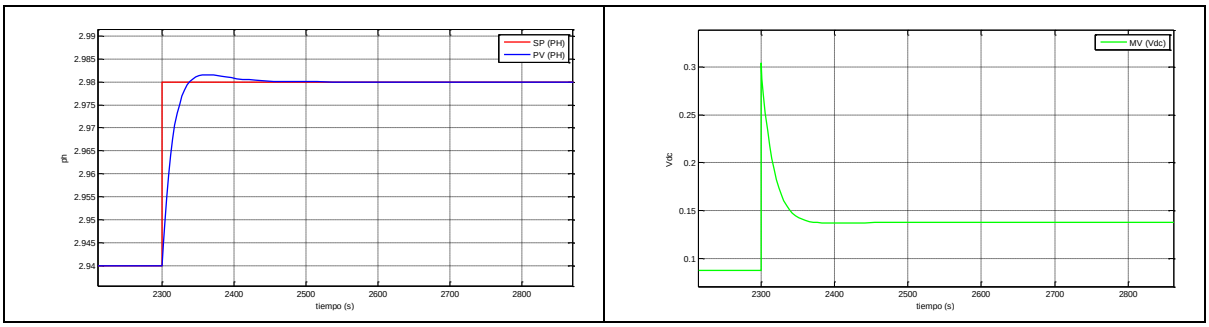


Fig. 5.45 Respuesta del sistema ante un cambio de setpoint de pH de 2.98

Tramo 5:

En este tramo correspondiente a valores comprendidos de pH entre 3.02 a 3.35. Se consideró la siguiente función de transferencia:

$$T(s) = \frac{1.1}{154s + 1}$$

Los parámetros del controlador son los siguientes:

Kp = 3.94 Ti = 100.1

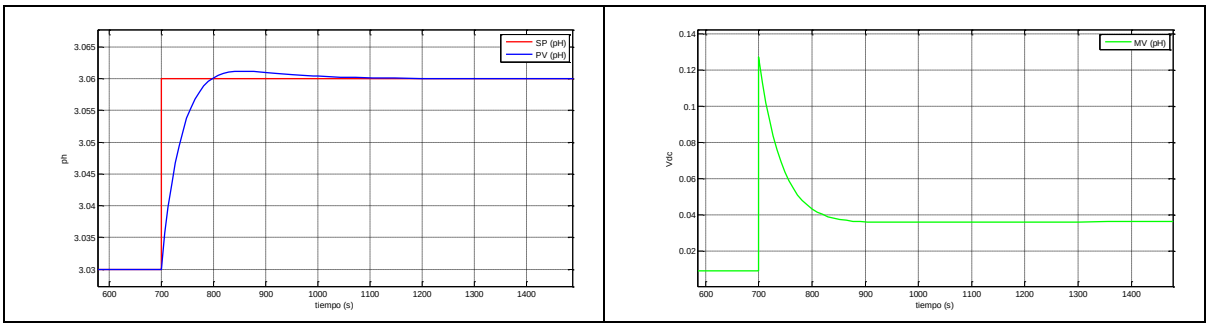


Fig. 5.46 Respuesta del sistema ante un cambio de setpoint de pH de 3.06

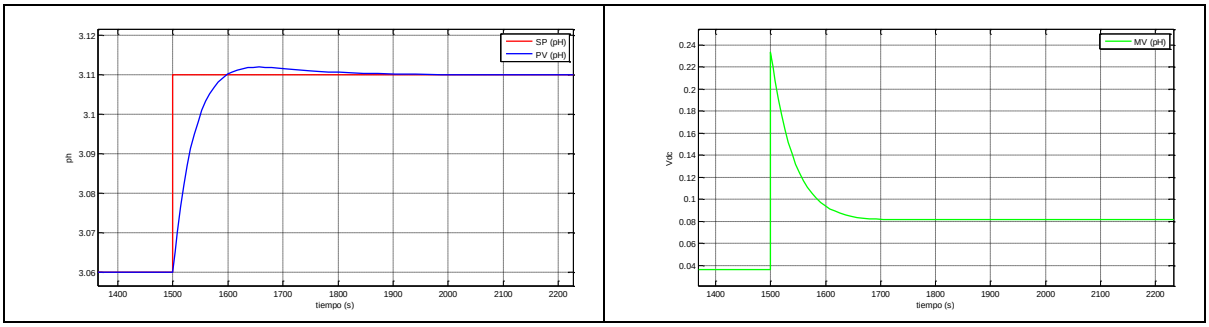


Fig. 5.47 Respuesta del sistema ante un cambio de setpoint de pH de 3.11

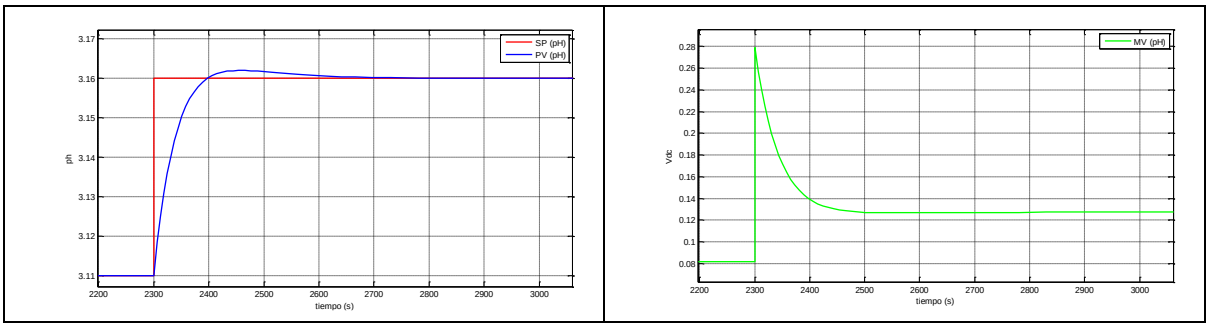


Fig. 5.48 Respuesta del sistema ante un cambio de setpoint de pH de 3.16

Tramo 6:

En este tramo correspondiente a valores comprendidos de pH entre 3.35 a 3.53. Se consideró la siguiente función de transferencia:

$$T(s) = \frac{1.8}{98s + 1}$$

Los parámetros del controlador son los siguientes:

Kp = 2.41 Ti = 63.7

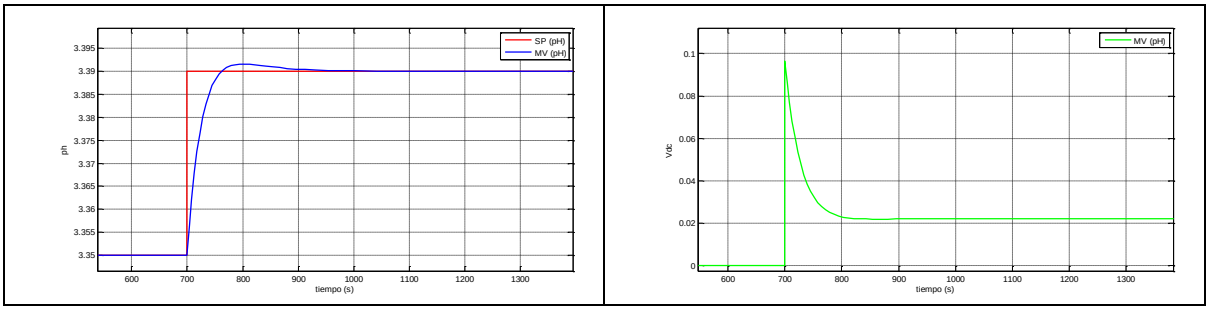


Fig. 5.49 Respuesta del sistema ante un cambio de setpoint de pH de 3.39

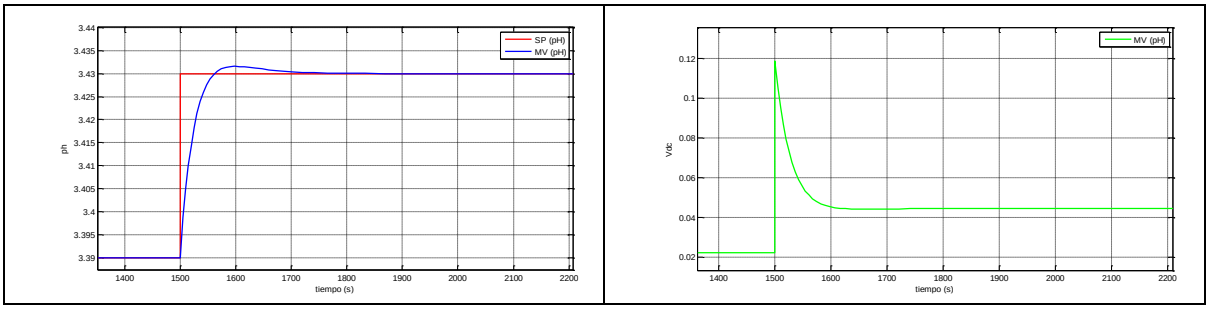


Fig. 5.50 Respuesta del sistema ante un cambio de setpoint de pH de 3.43

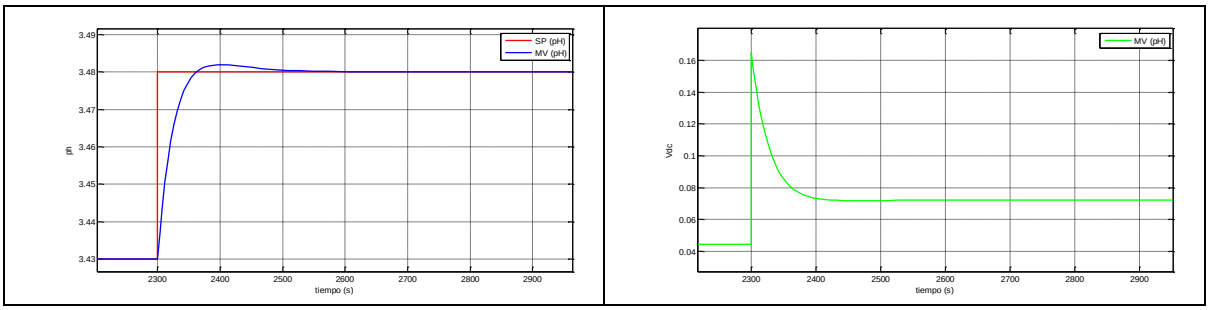


Fig. 5.51 Respuesta del sistema ante un cambio de setpoint de pH de 3.48

Tramo 7:

En este tramo correspondiente a valores comprendidos de pH entre 3.53 a 3.82. Se consideró la siguiente función de transferencia:

$$T(s) = \frac{2.9}{111s + 1}$$

Los parámetros del controlador son los siguientes:

Kp = 1.5 Ti = 72.15

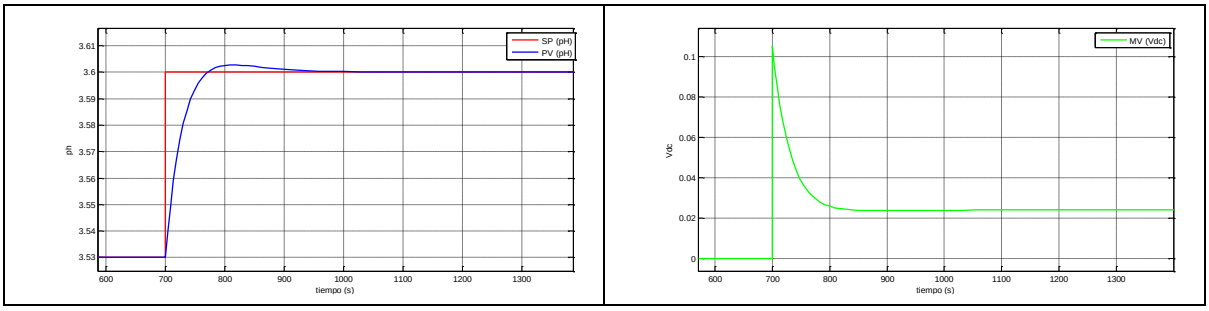


Fig. 5.52 Respuesta del sistema ante un cambio de setpoint de pH de 3.6

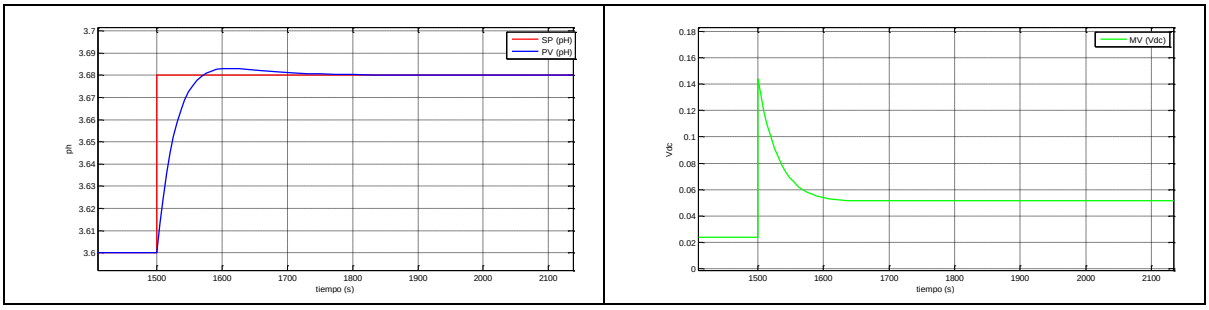


Fig. 5.53 Respuesta del sistema ante un cambio de setpoint de pH de 3.68

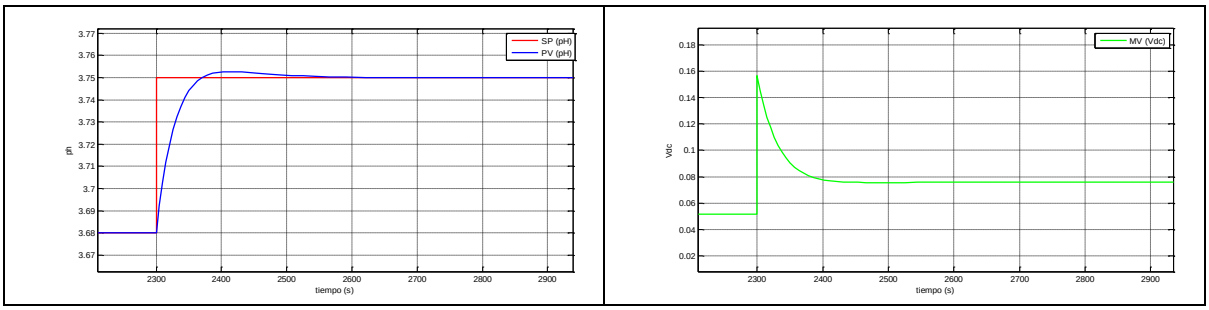


Fig. 5.54 Respuesta del sistema ante un cambio de setpoint de pH de 3.75

Tramo 8:

En este tramo correspondiente a valores comprendidos de pH entre 3.82 a 4.7. Se consideró la siguiente función de transferencia:

$$T(s) = \frac{8.8}{244s + 1}$$

Los parámetros del controlador son los siguientes:

Kp = 0.49 Ti = 158.6

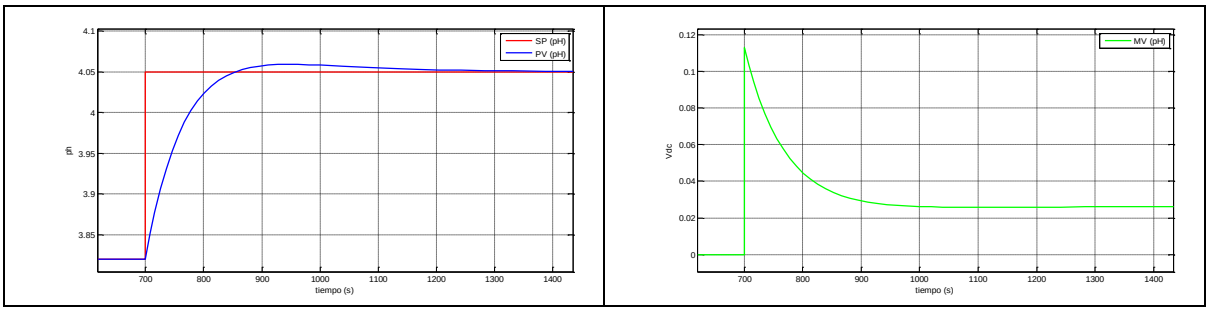


Fig. 5.55 Respuesta del sistema ante un cambio de setpoint de pH de 4.05

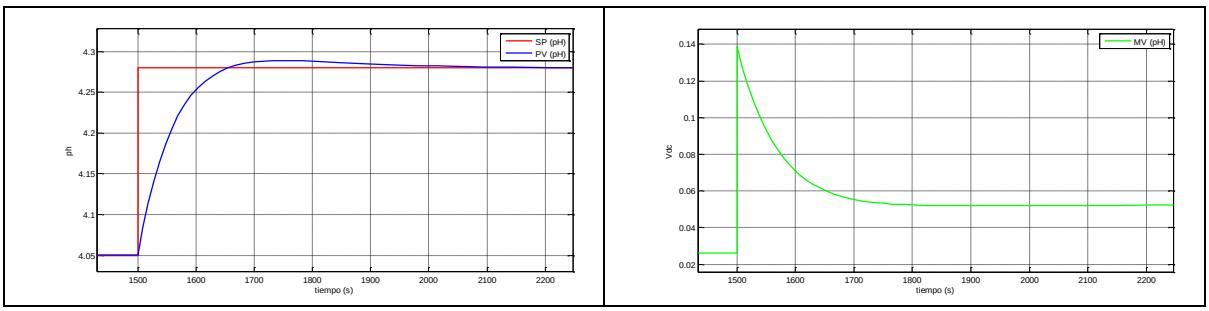


Fig. 5.56 Respuesta del sistema ante un cambio de setpoint de pH de 4.27

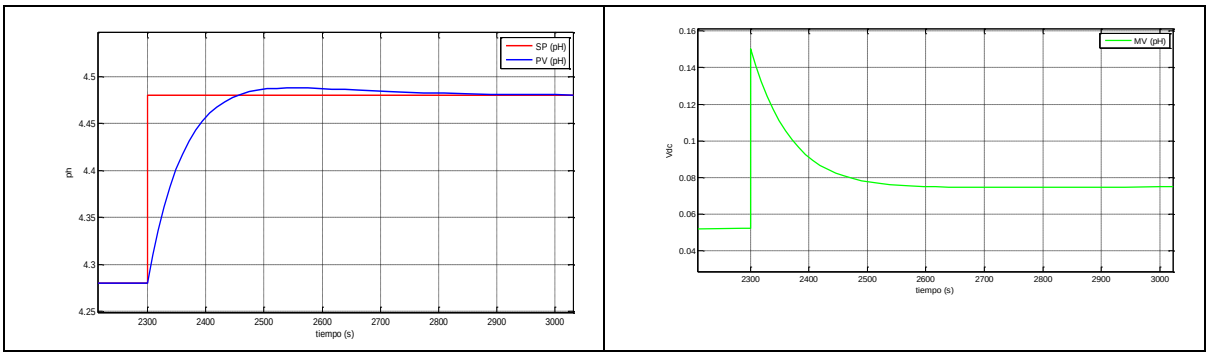


Fig. 5.57 Respuesta del sistema ante un cambio de setpoint de pH de 4.48

5.3 Control *Gain Scheduling* de módulo de pH

La implementación del regulador *PI* se realizó en un sistema embebido desarrollado en el laboratorio de SAC explicado anteriormente.

A continuación se muestran las ventanas del sistema *SCADA* desarrollado en el laboratorio, el cual cuenta ya con una protección por derecho de autor ante INDECOPI.

Las figuras a continuación muestran el proceso de regulación de pH a lazo cerrado, la ley de control embebida en el dispositivo FPGA responde a un controlador *PI* ideal con ganancia programable, siendo este un tipo de controlador adaptativo programado.

En la Fig. 5.58 se muestra el sistema a lazo cerrado con un setpoint de 2.6 de pH. Como se aprecia el sistema responde ante el cambio del valor de referencia, llegando en 30 segundos aproximadamente.

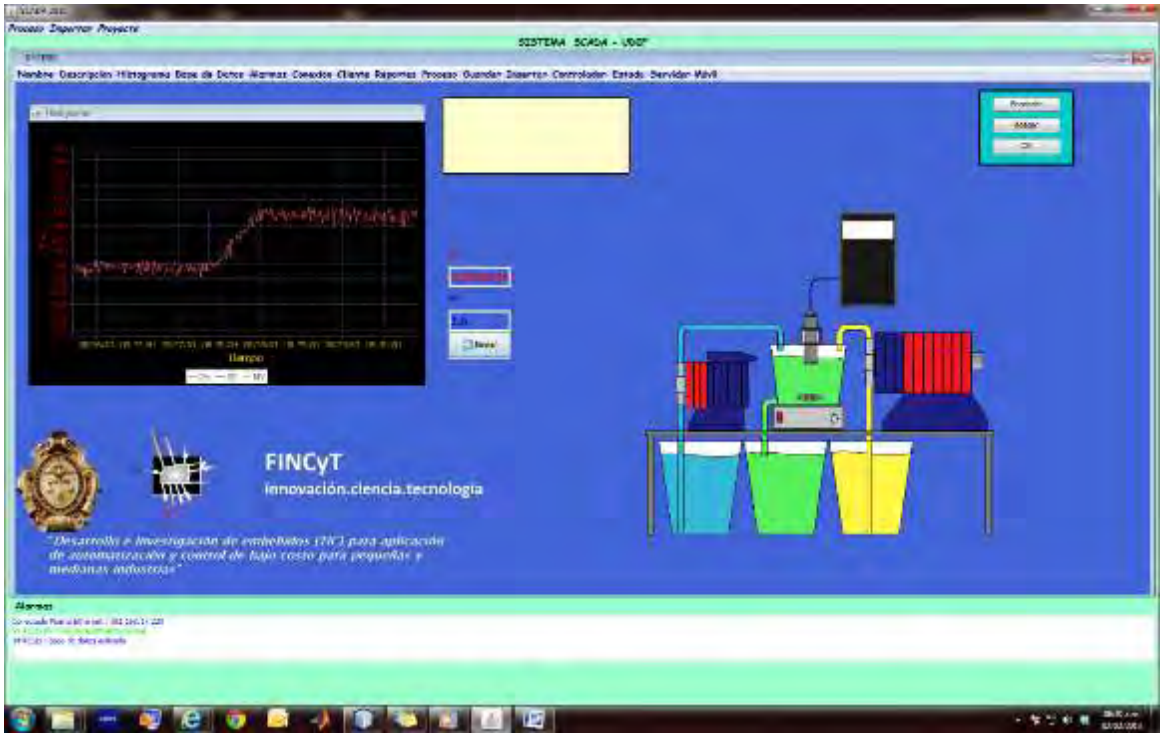


Fig. 5.58 Interface de sistema SCADA con setpoint de 2.6 de pH

En la figura siguiente se muestra el sistema a lazo cerrado con un setpoint de 2.8 de pH. El sistema responde ante el cambio del valor de referencia, llegando en 30 segundos.

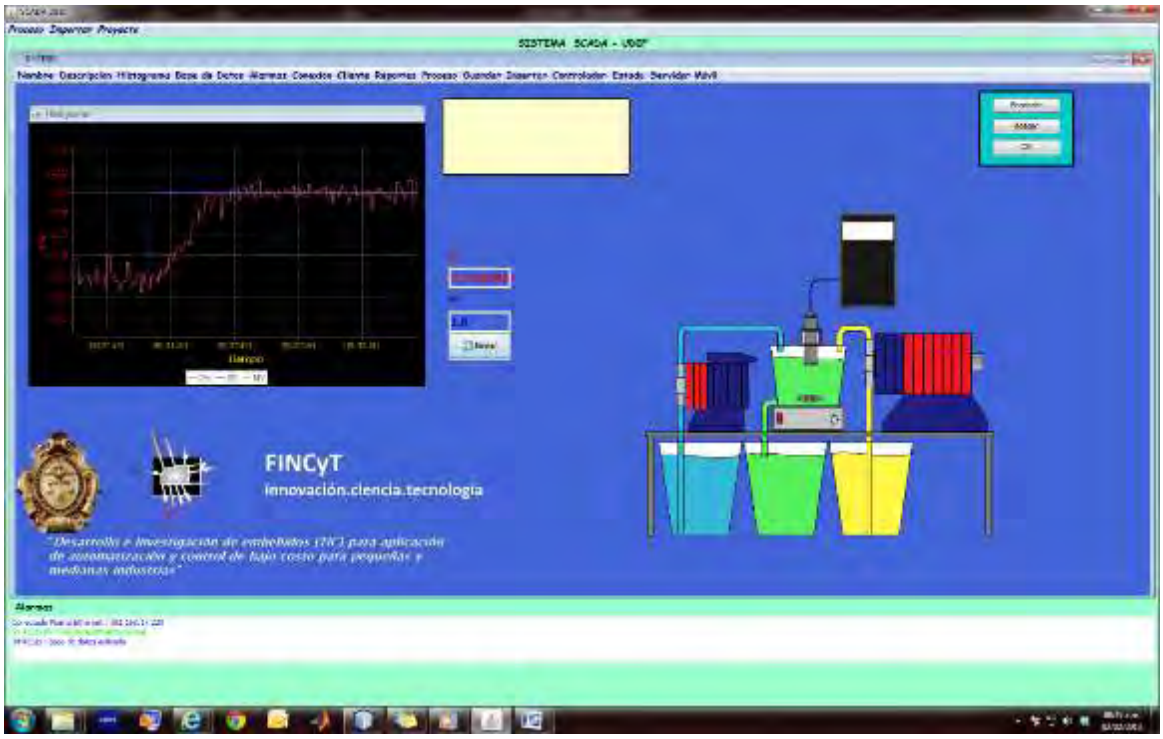


Fig. 5.59 Interface de sistema SCADA con setpoint de 2.8 de pH

En la Fig. 5.60 se aprecia un cambio de setpoint a pH 3. El sistema responde ante el cambio del valor de referencia de la misma forma anterior, llegando a este en 30 segundos aproximadamente.

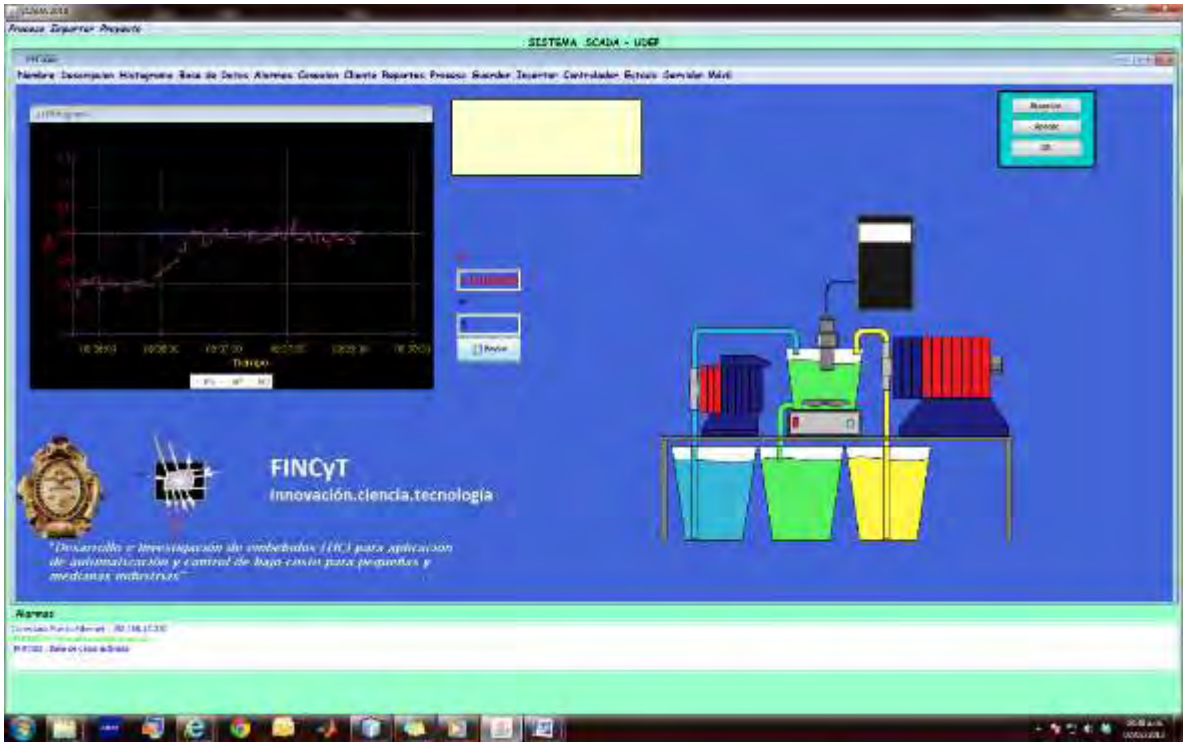


Fig. 5.60 Interface de sistema SCADA con setpoint de 3 de pH

En la Fig. 5.61 se muestra el sistema a lazo cerrado con un setpoint de 3.2 de *pH*. El sistema responde ante el cambio del valor de la misma forma.

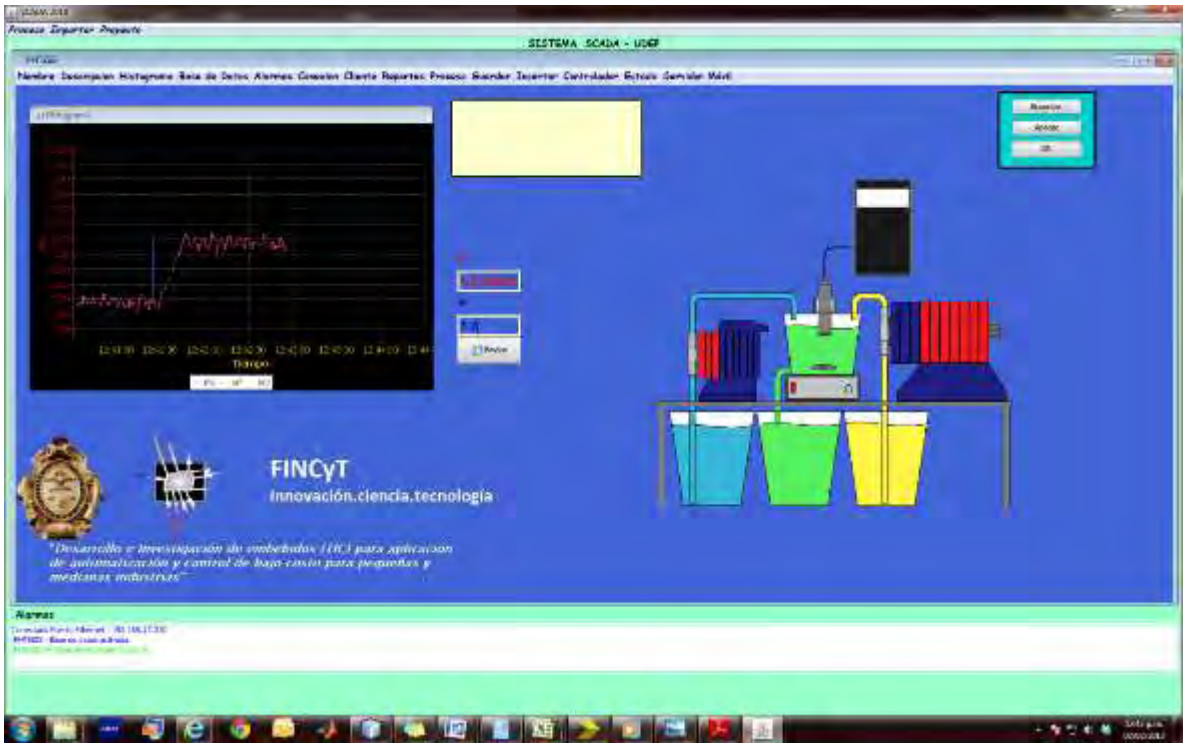


Fig. 5.61 Interface de sistema SCADA con setpoint de 3.2 de pH

En la Fig. 5.62 se muestra el sistema a lazo cerrado con un setpoint de 3.4 de pH.

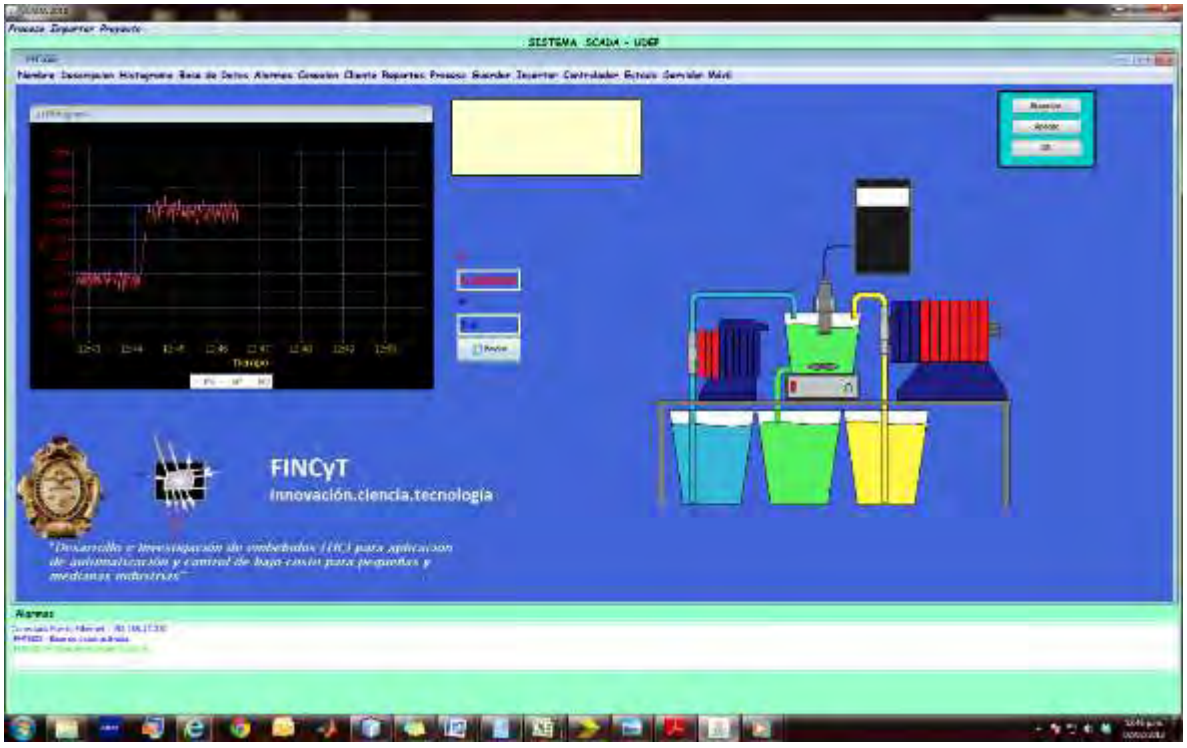


Fig. 5.62 Interface de sistema SCADA con setpoint de 3.4 de pH

En la figura a continuación se aprecia un cambio de setpoint a pH 3.6. El sistema responde ante el cambio del valor de referencia, llegando a este en 30 segundos aproximadamente.

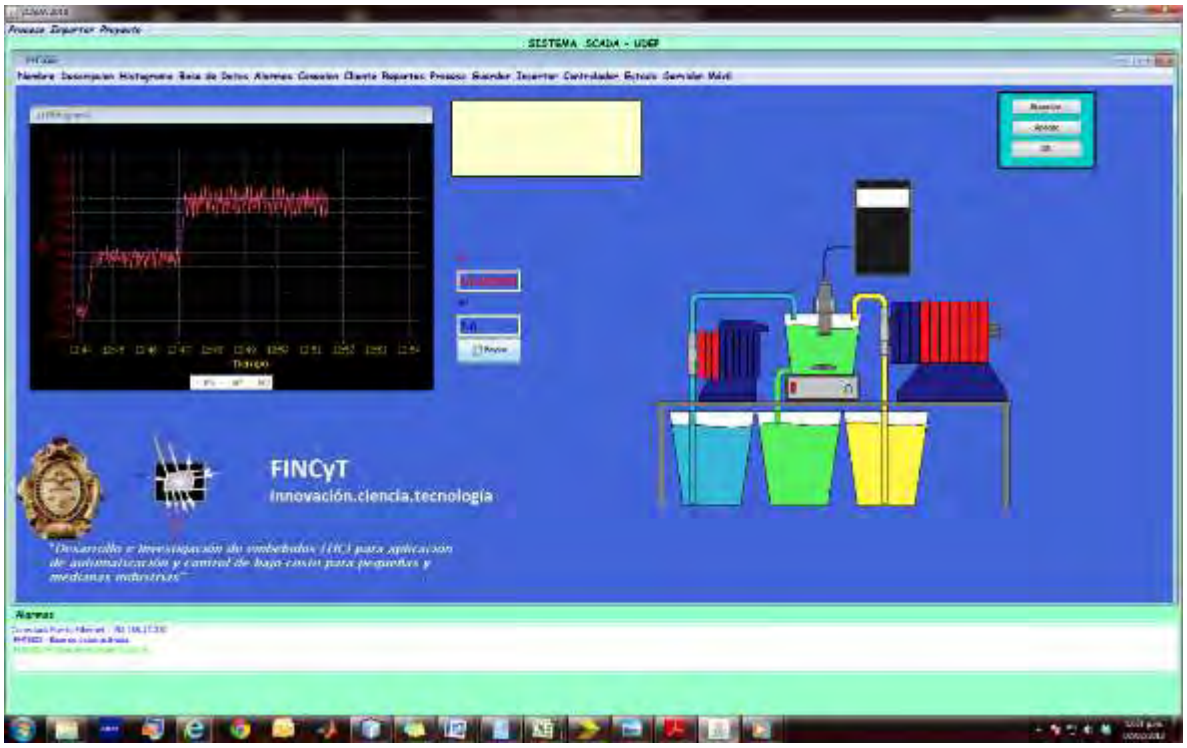


Fig. 5.63 Interface de sistema SCADA con setpoint de 3.6 de pH

En la Fig. 5.64 se muestra el sistema a lazo cerrado con un setpoint de 3.8 de pH.

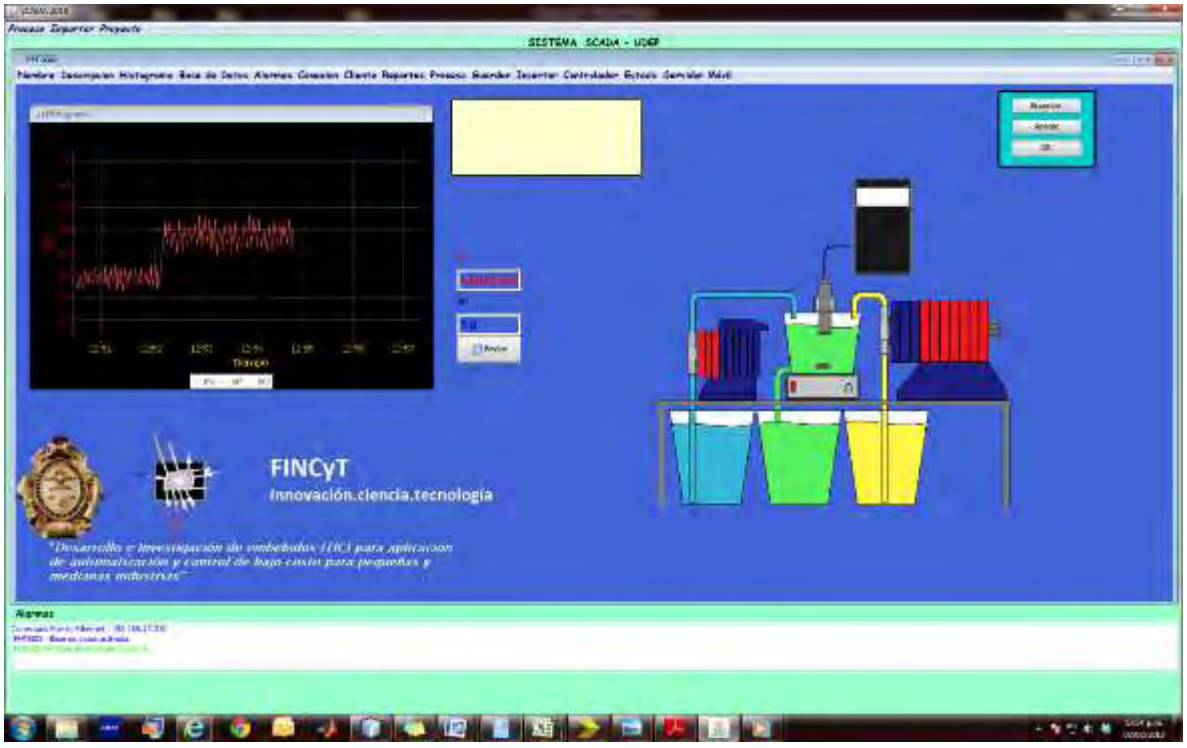


Fig. 5.64 Interface de sistema SCADA con setpoint de 3.8 de pH

En la Fig. 5.65 se aprecia un cambio de setpoint a pH 4. Se aprecia también que a medida que aumentamos en pH, el ruido se hace presente con mayor intensidad. Esto debido que va siendo presente la influencia de la no-linealidad dinámica, por otro lado la no implementación de un filtro.

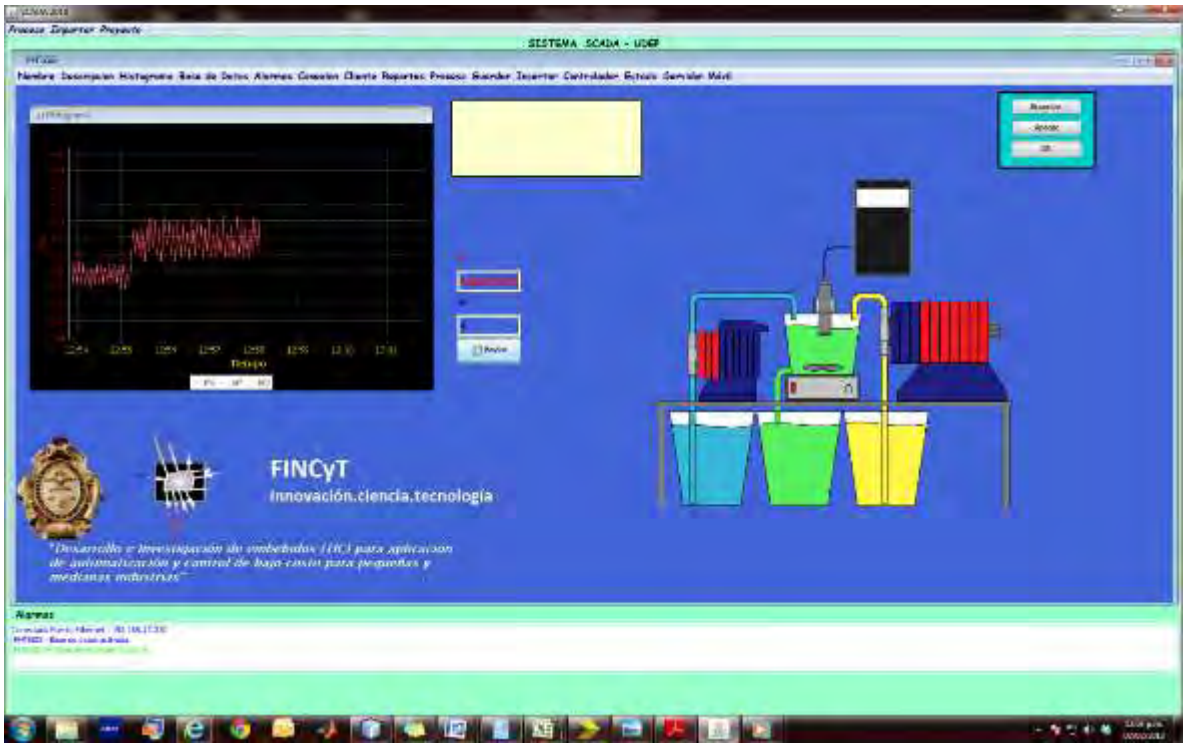


Fig. 5.65 Interface de sistema SCADA con setpoint de 4 de pH

En la Fig. 5.66 se muestra un cambio de setpoint de 4.2 de pH.

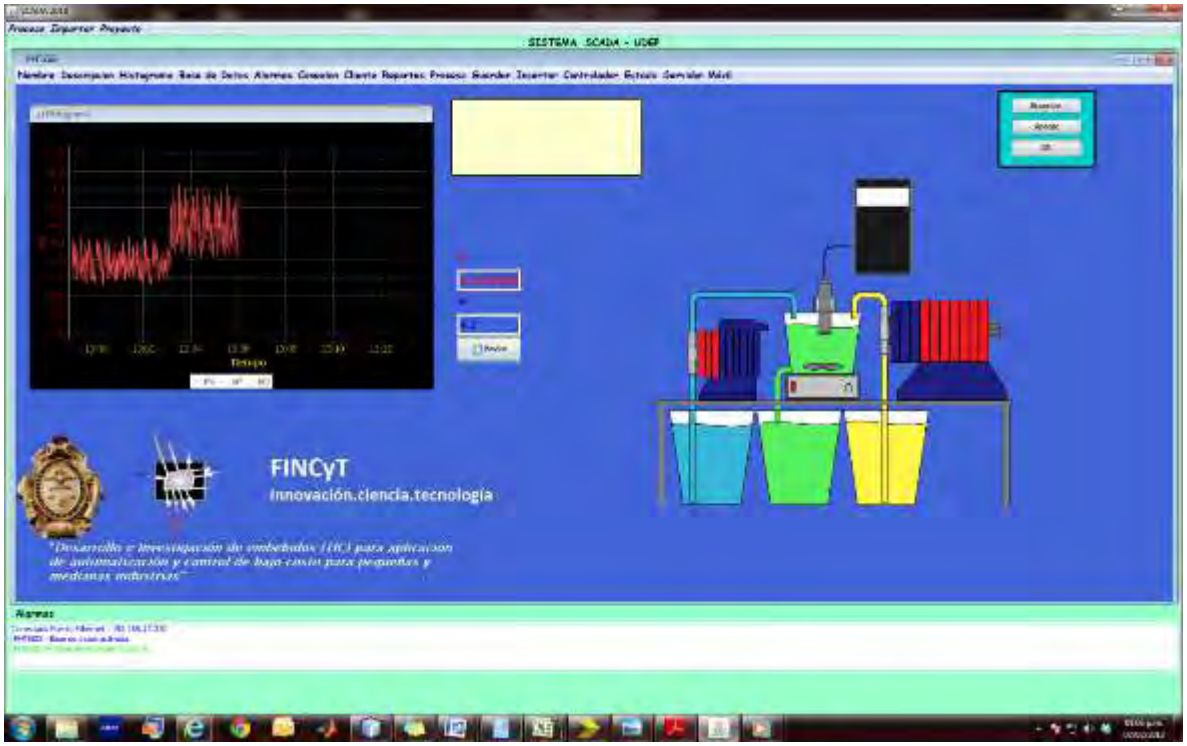


Fig. 5.66 Interface de sistema SCADA con setpoint de 4.2 de pH

En la Fig. 5.67 se muestra la pantalla del sistema SCADA con cambio de setpoint en 4.4 de pH.

El sistema responde ante el cambio de referencia de buena forma. A partir de valores superiores a este en pH, la alta no linealidad, tanto estática y dinámica se hace presente con mucha más fuerza. Aumentando la amplitud del ruido con bastante intensidad.

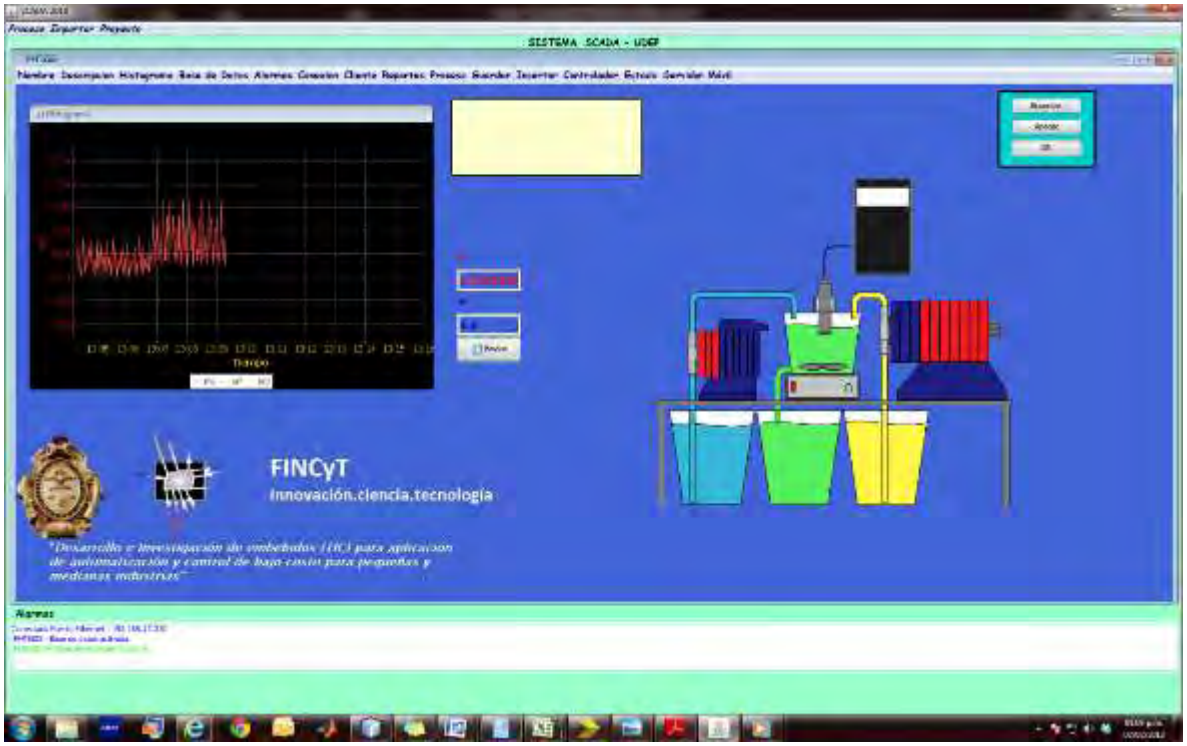


Fig. 5.67 Interface de sistema SCADA con setpoint de 4.4 de pH

Conclusiones

Esta tesis propone un controlador adaptativo programado por medio de un algoritmo de control *PI con Gain Scheduling*, en rango donde solo es afectada la no linealidad estática.

De esta forma se procede a linealizar por tramos encontrando los parámetros del controlador *PI* por medio de sintonización por el método de asignación de polos.

Propone una alternativa de control utilizando un sistema que cuenta con un alto grado de paralelismo, Los FPGAs son una arquitectura adecuada para la implementación de controladores como el *PID, Predictivo, Fuzzi Logic,* entre otros. En este caso se ha implementado un algoritmo de control *PI con Gain Scheduling* debido a la gran no linealidad del proceso, controlando en regiones bajas donde no hay mucho efecto de la no linealidad dinámica.

Los procesos industriales en su mayoría presentan un comportamiento no lineal, y al aproximarlo a uno lineal causa deficiencias en el control automático. Por lo que se busca implementar nuevas estrategias de control avanzado, así como la técnica presente en esta tesis, control adaptativo programado por medio de un algoritmo *PI con Gain Scheduling,* permitiendo afrontar el problema.

Se ha implementado un control automático sobre un módulo existente en el laboratorio de electrónica y automática de la universidad de Piura. Además se ha desarrollado, fabricado y programado software y hardware de última generación como es la tarjeta FPGA diseñada.

En procesos donde la no linealidad dinámica unida a la gran no linealidad estática, hace que no se pueda representar su ganancia estática utilizando una recta; para este caso se hace necesaria la aproximación de la no linealidad.

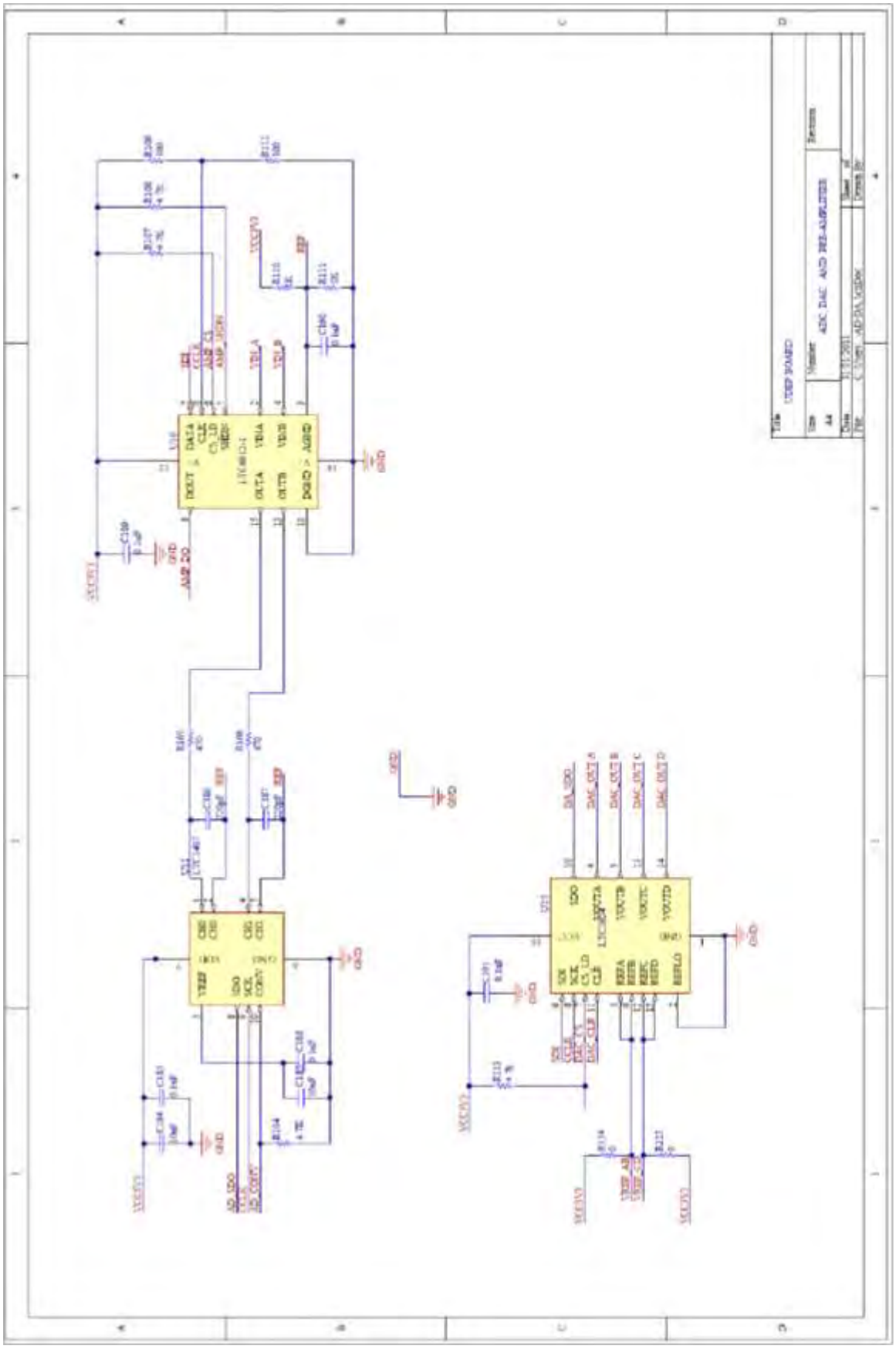
El alto grado de paralelismo de los FPGAs hace que el tiempo real sea posible, así también la reducción de coste computacional.

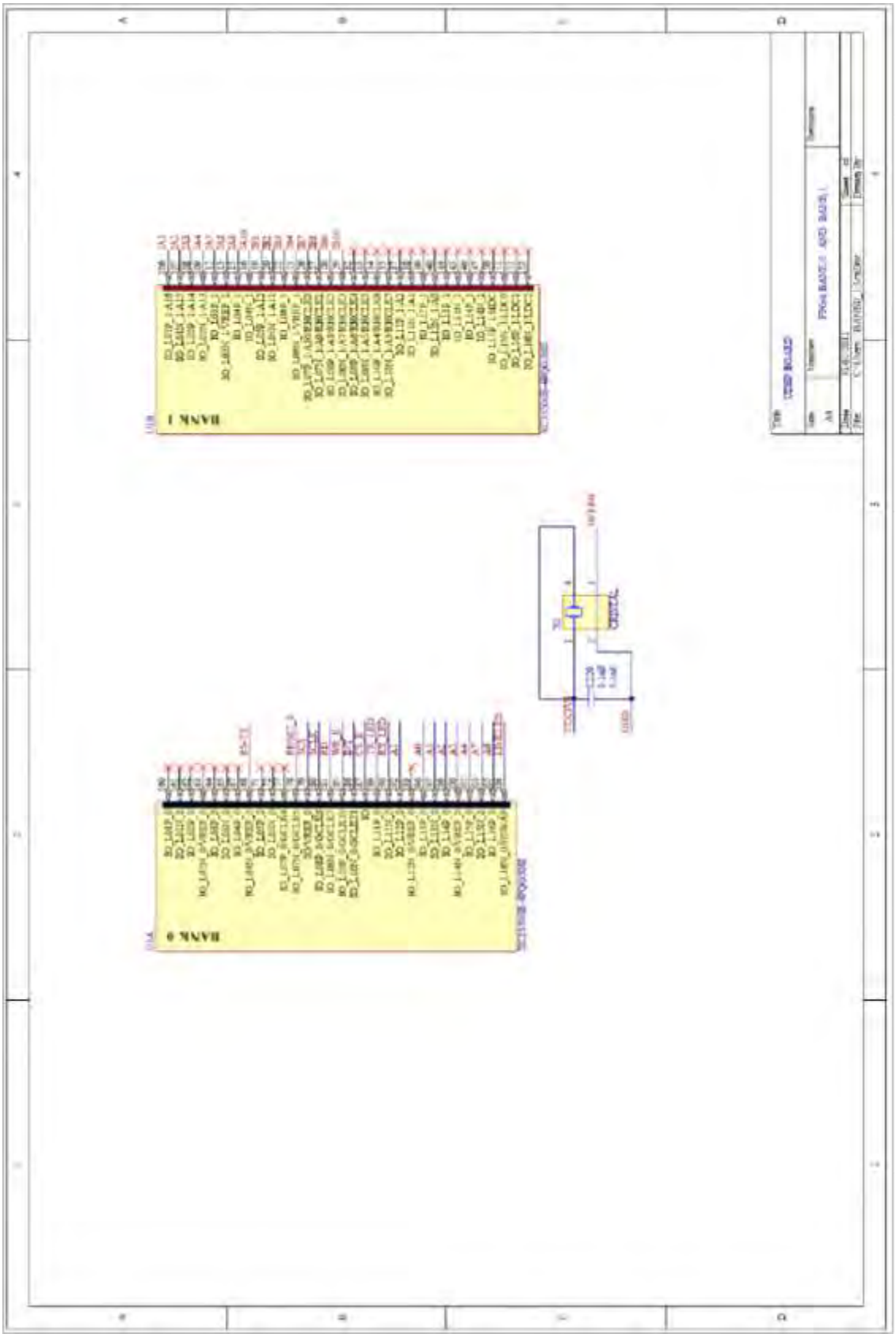
Los resultados experimentales han sido muy satisfactorios, el pH se llega a controlar de buena forma.

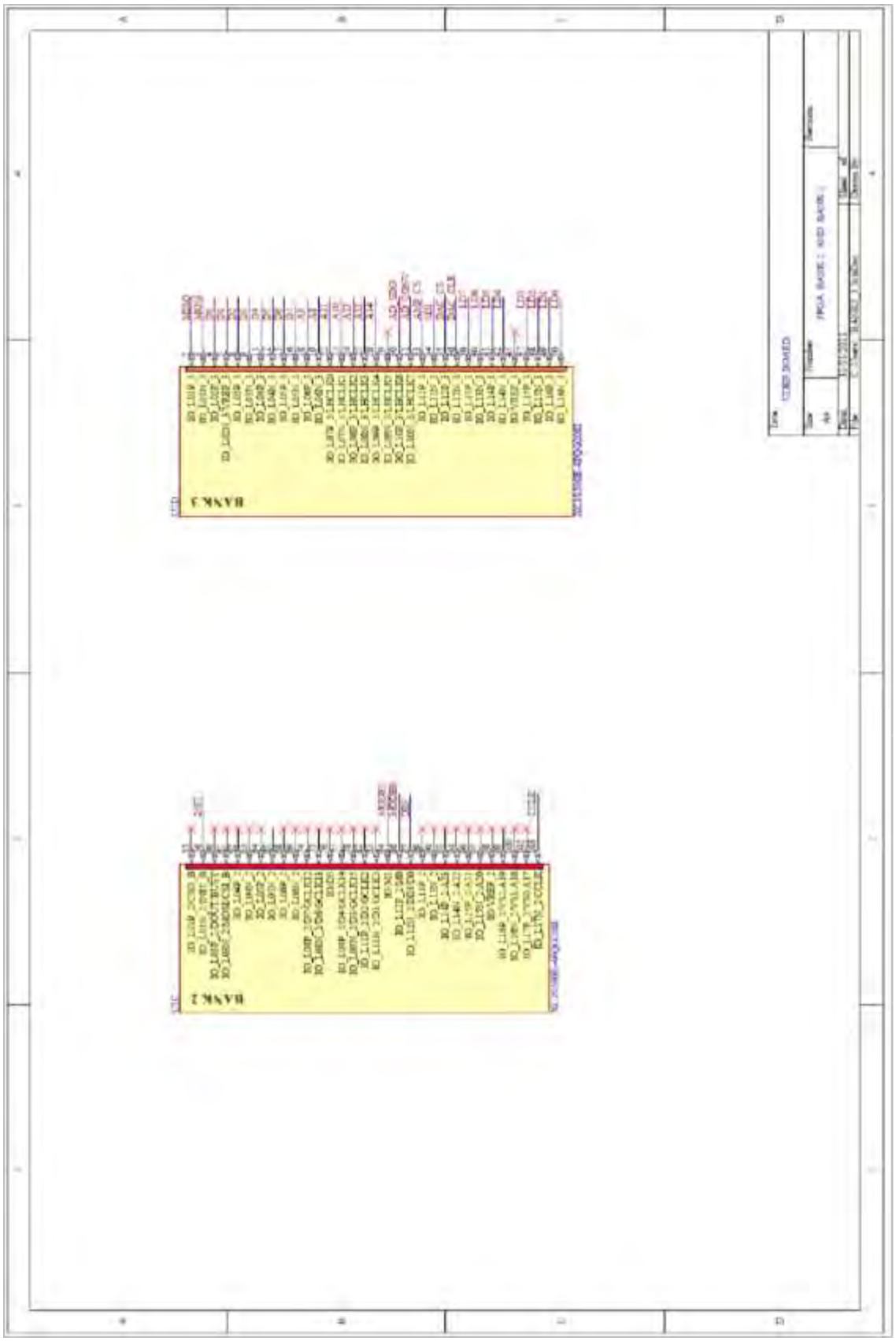
La velocidad de procesamiento de la tarjeta FPGA, 50 Mhz, hacen a este dispositivo sea capaz de controlar procesos de dinámica lenta como procesos de dinámica rápida.

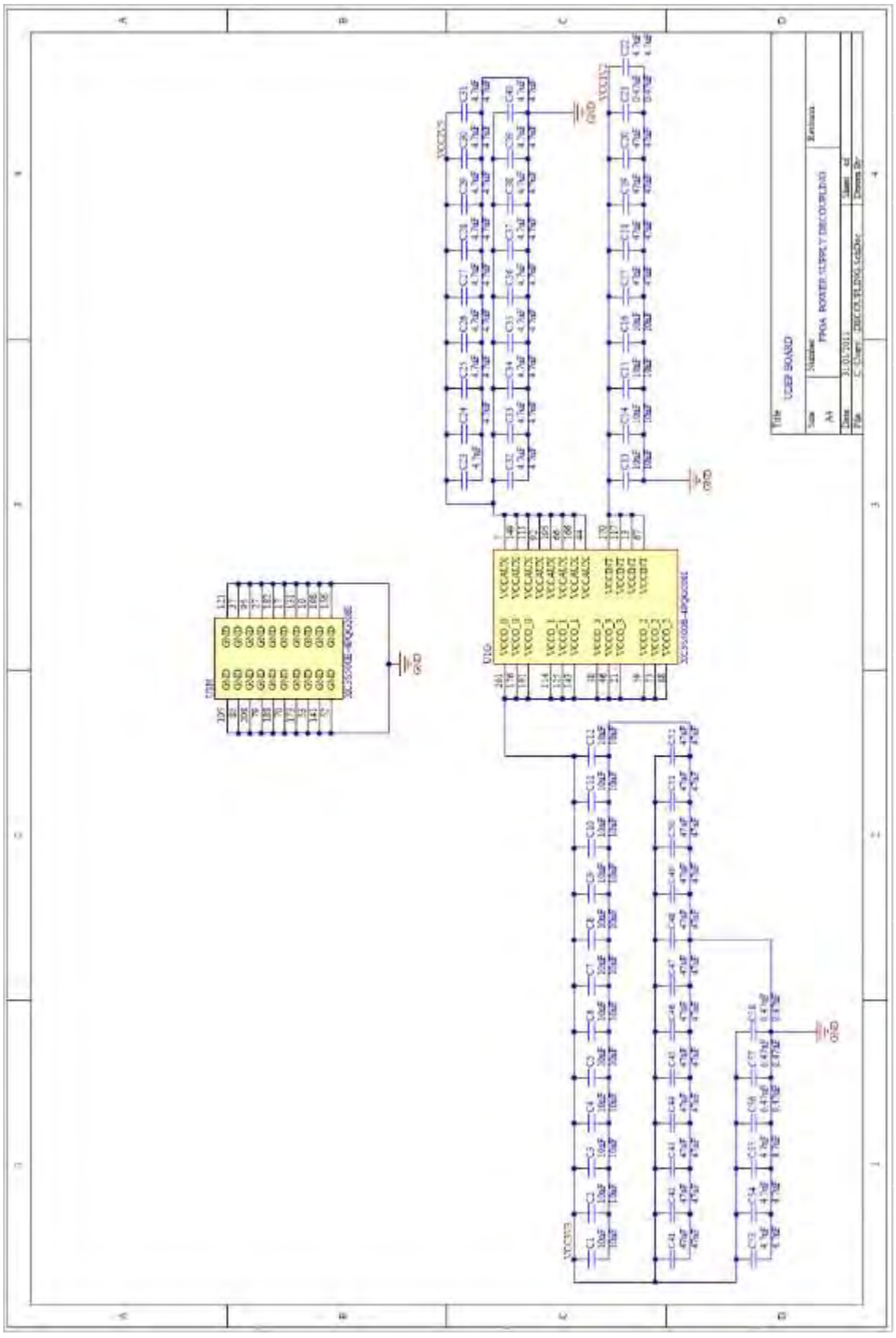
Anexo A

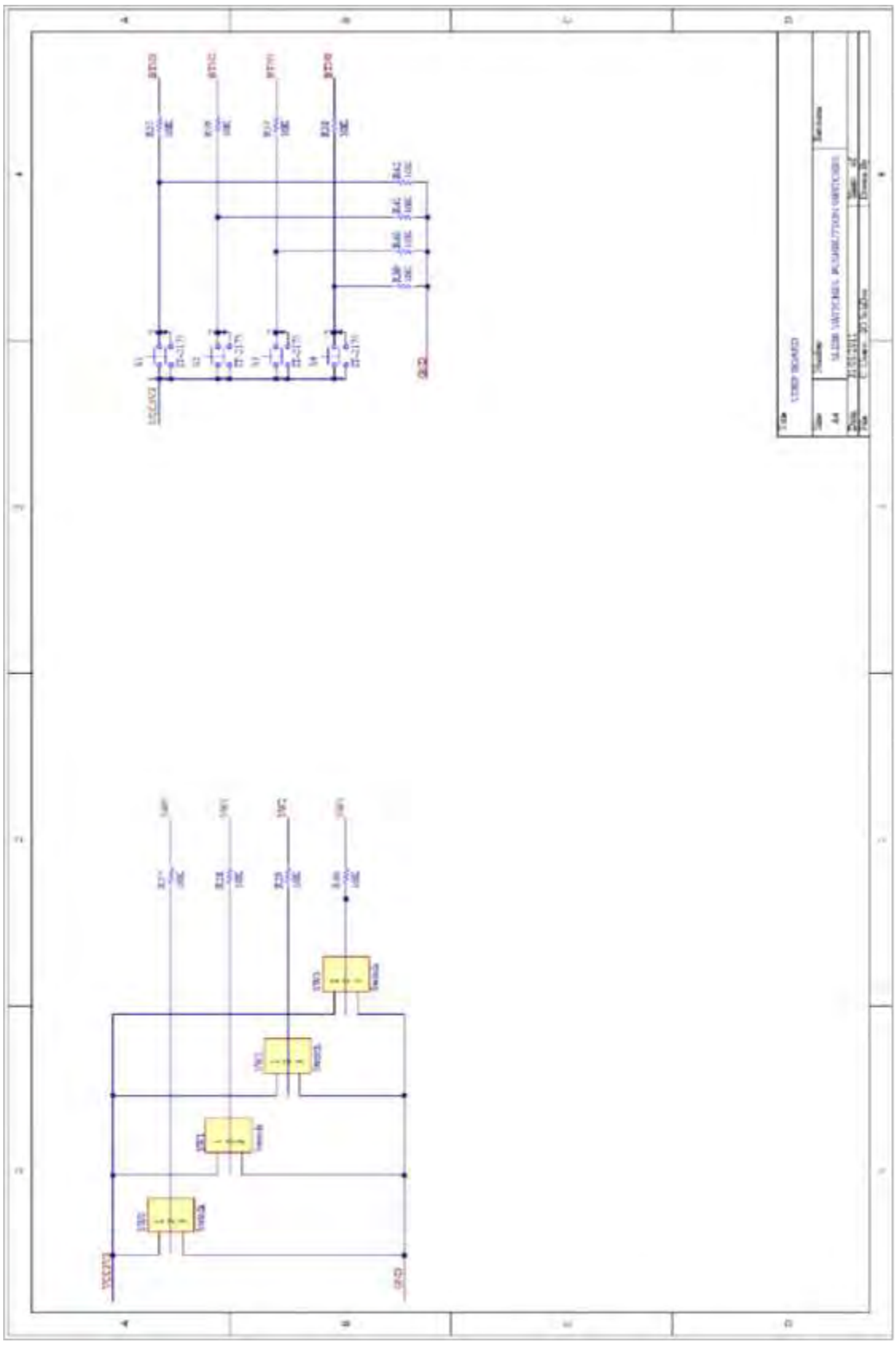
Archivos esquemáticos y gerber correspondiente a la tarjeta FPGA

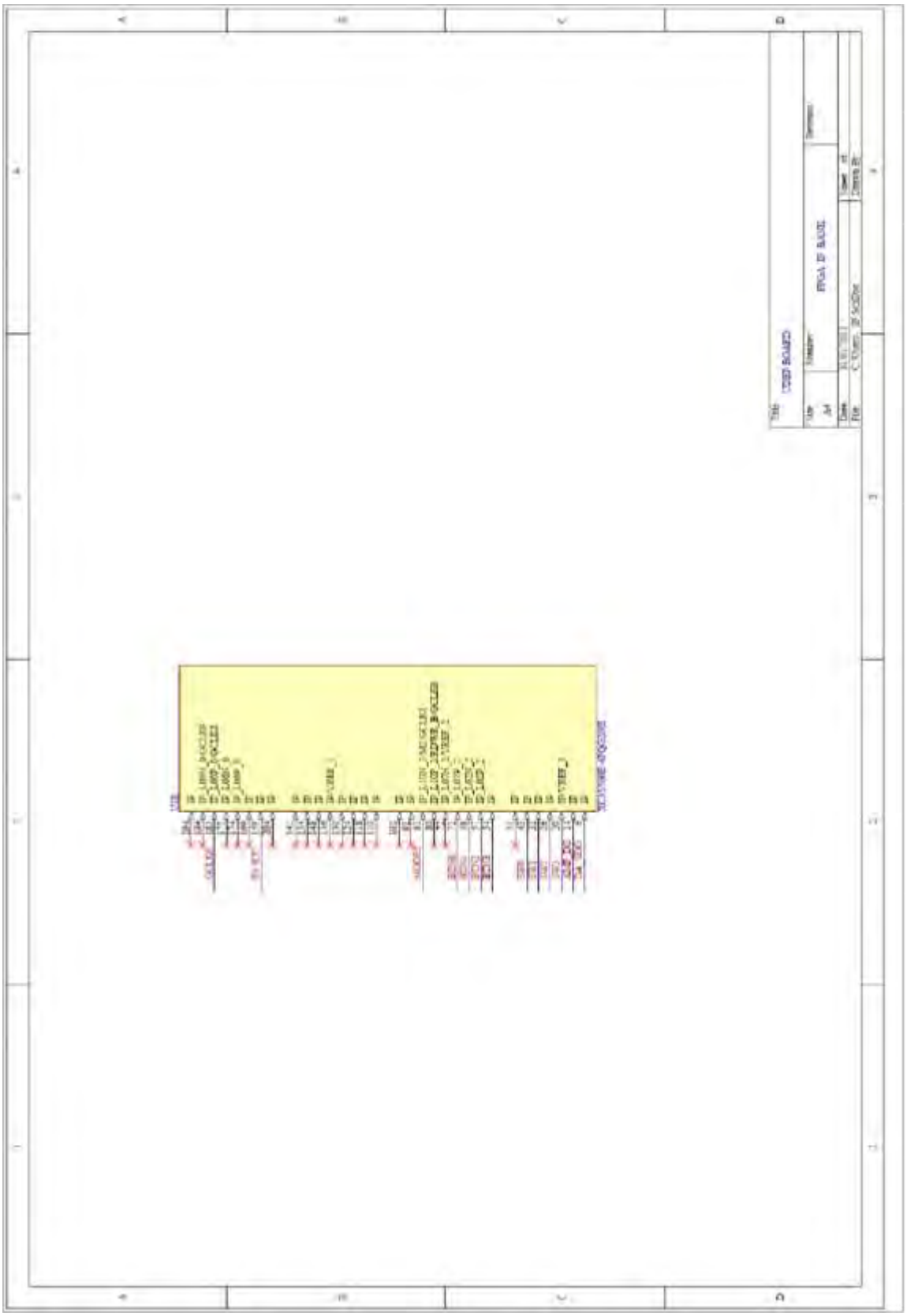


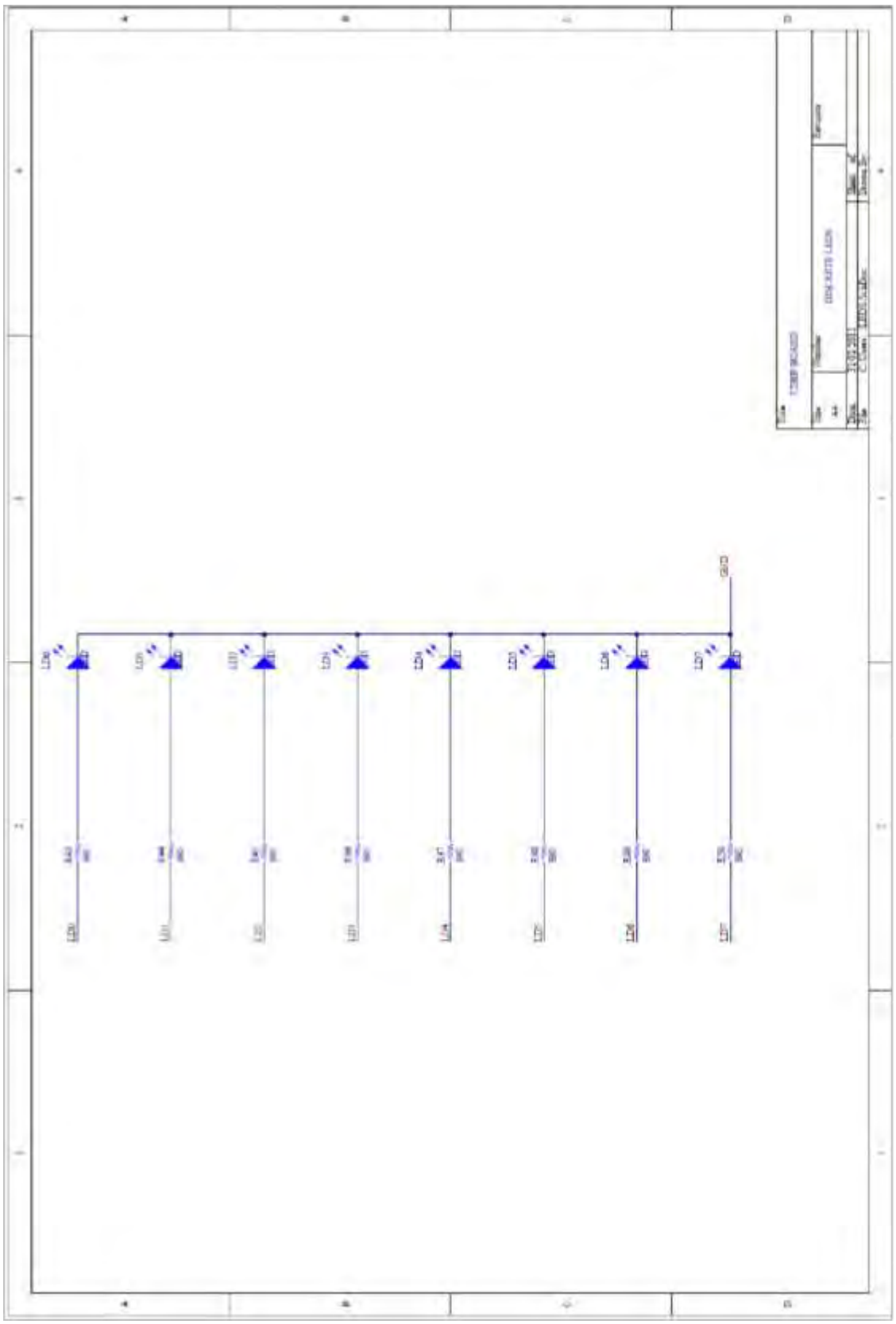


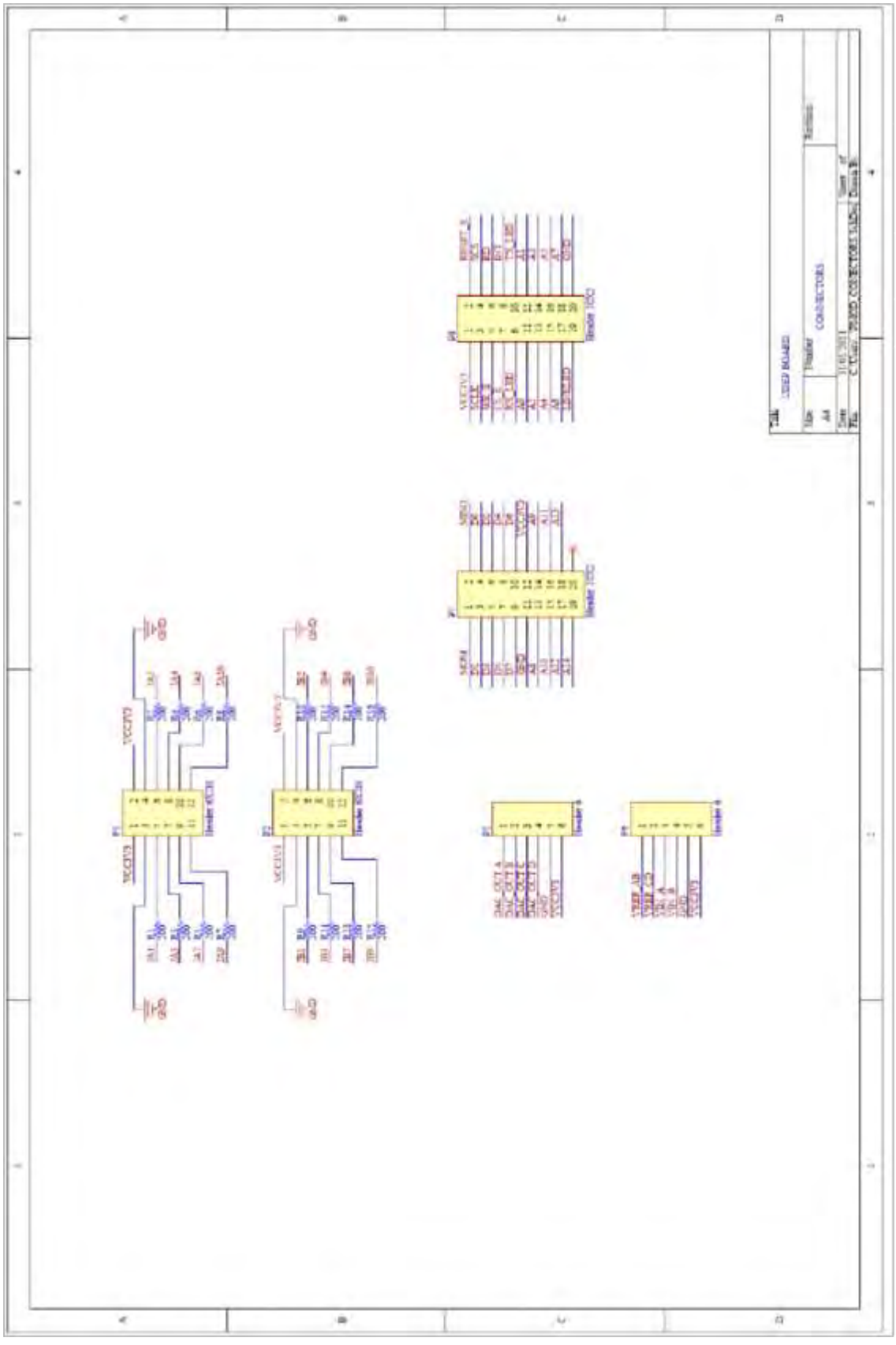


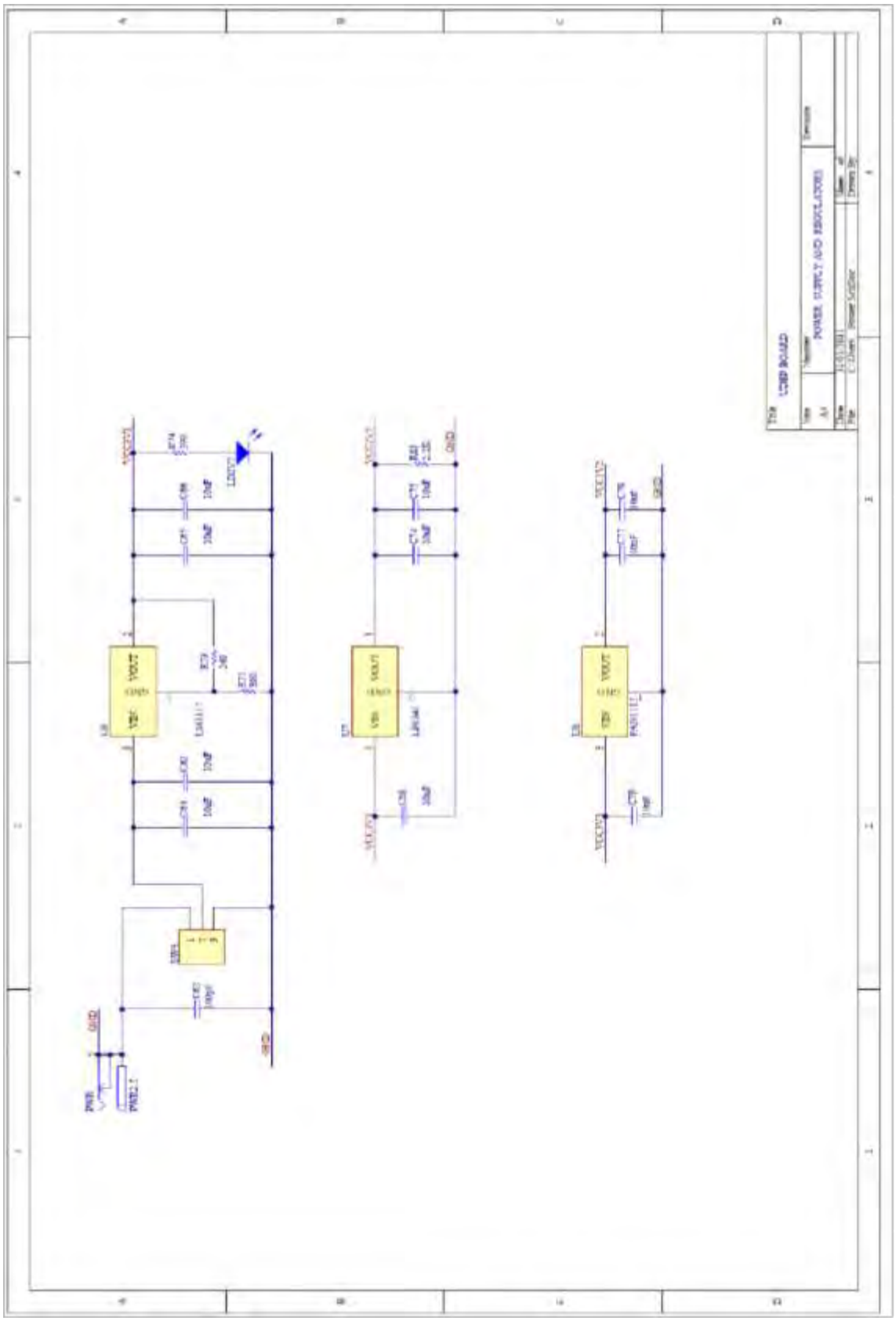


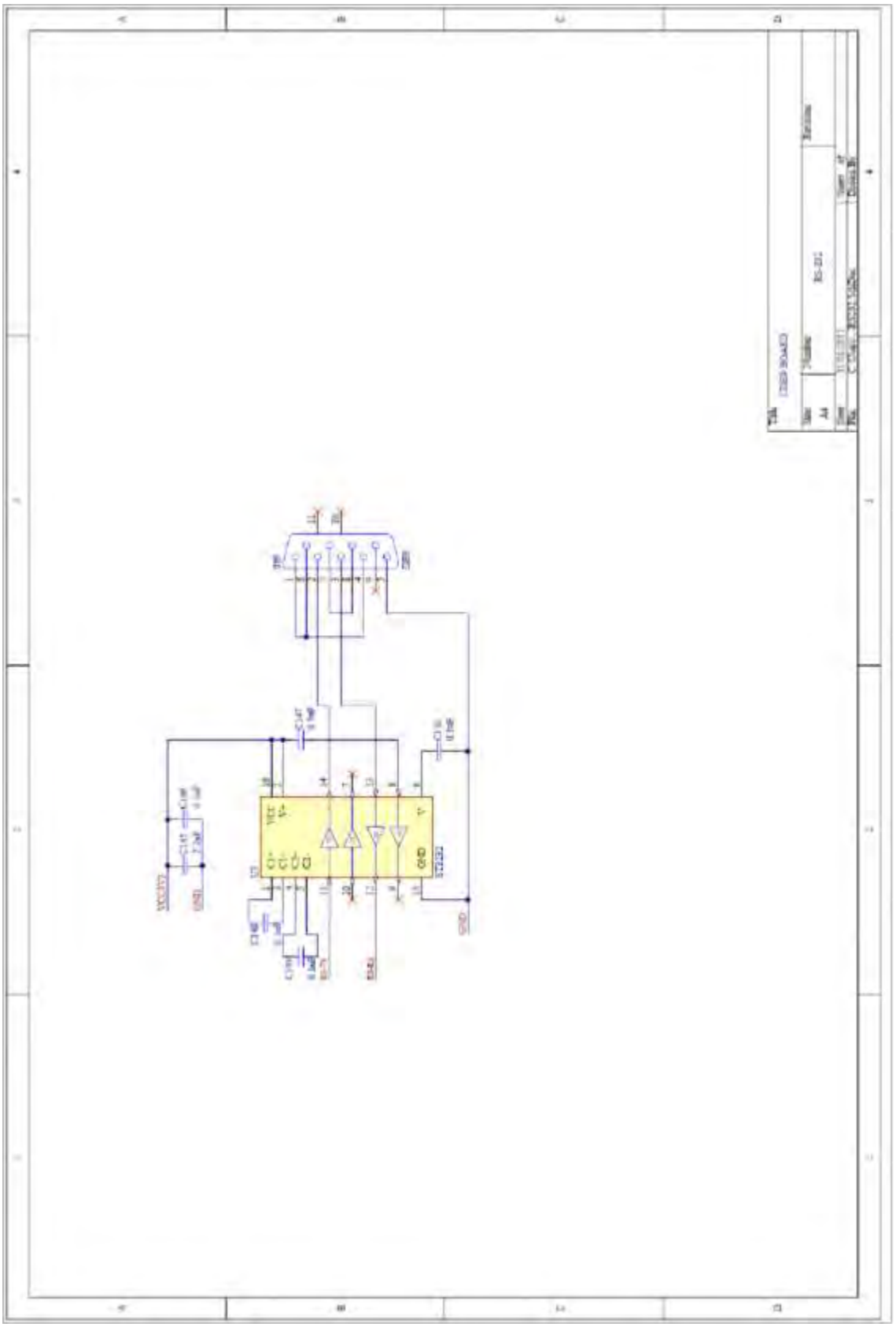


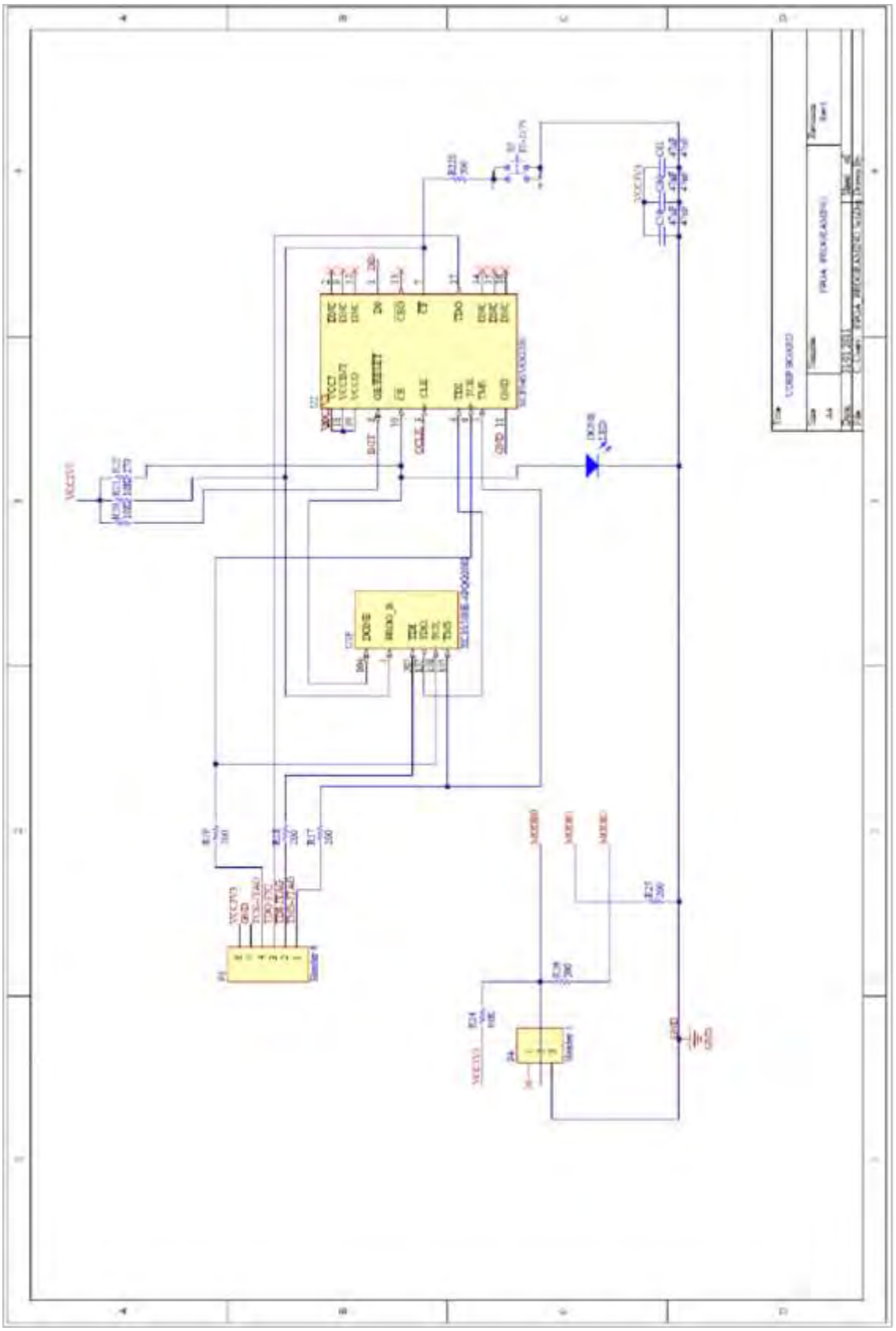












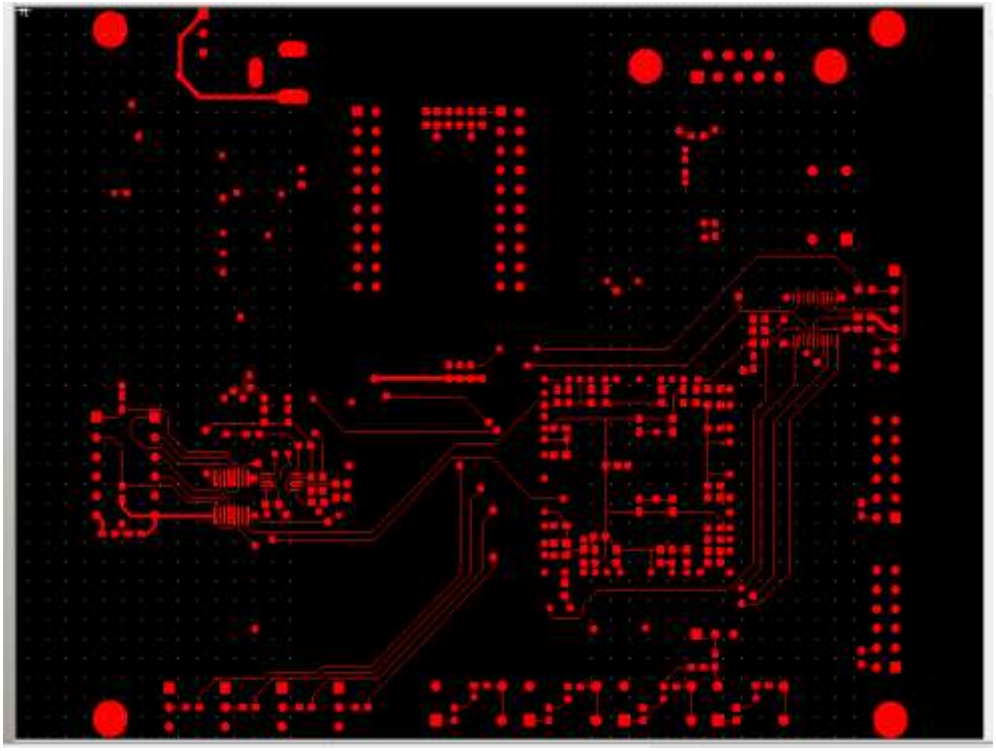


Fig. A.1 Archivo Gerber GBL

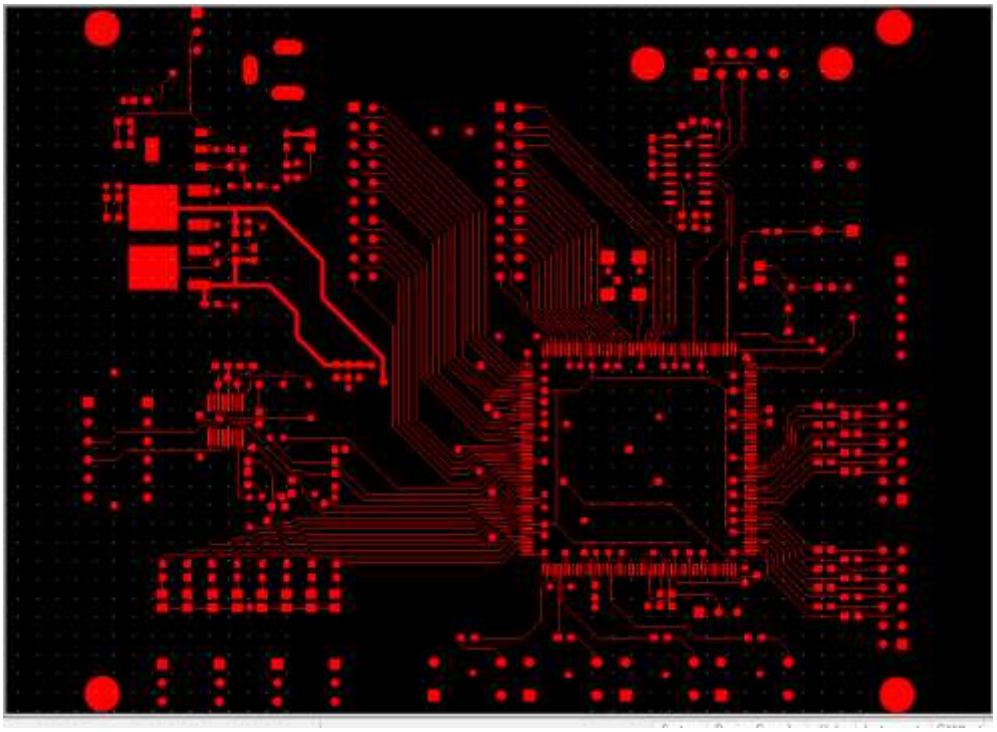


Fig. A.2 Archivo Gerber GTL

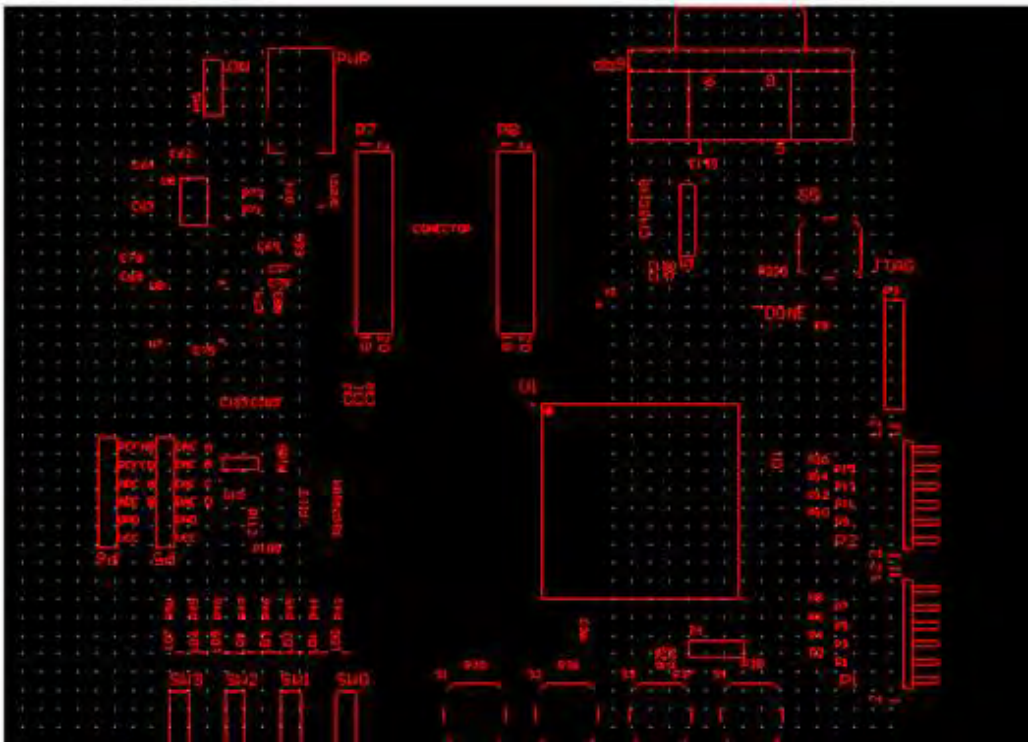


Fig. A.3 Archivo Gerber GTO

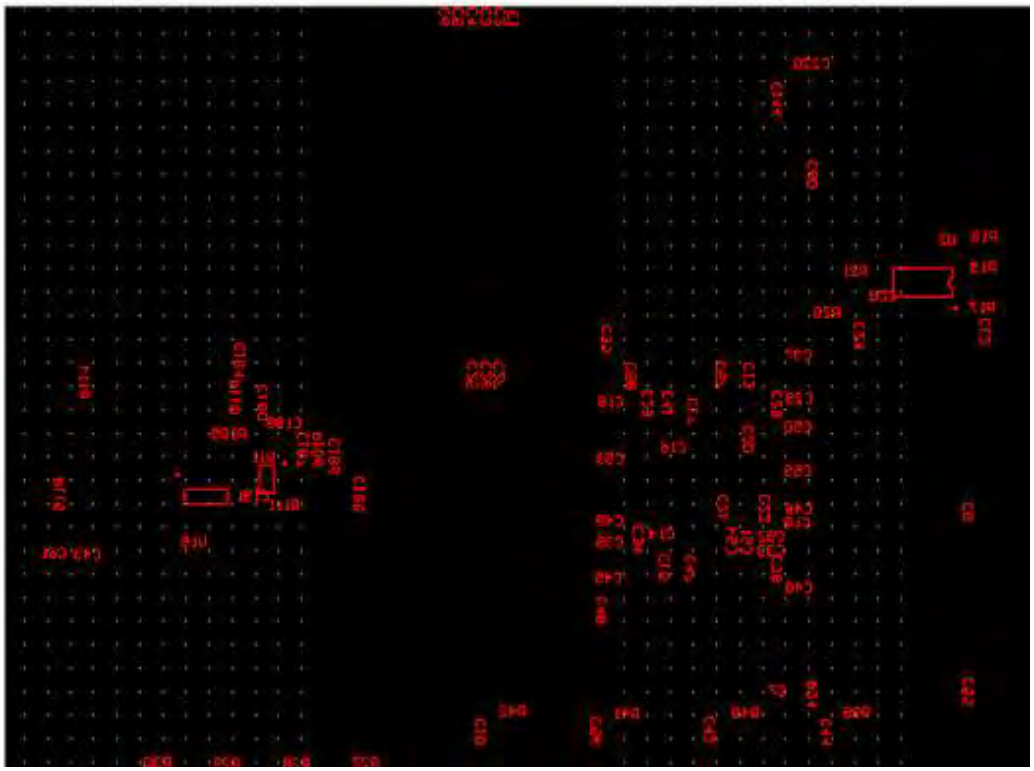


Fig. A.4 Archivo Gerber GBO

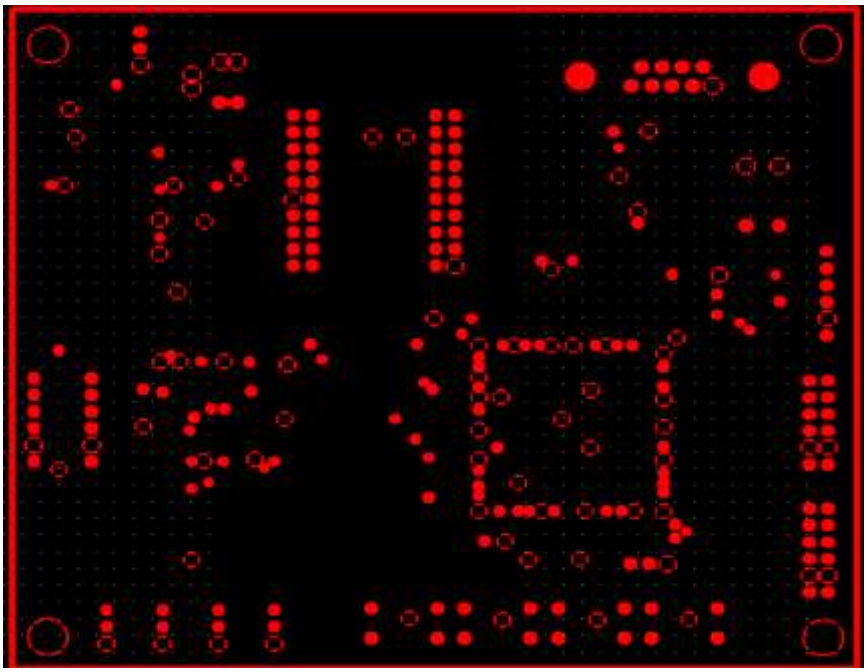


Fig. A.5 Archivo Gerber GP1

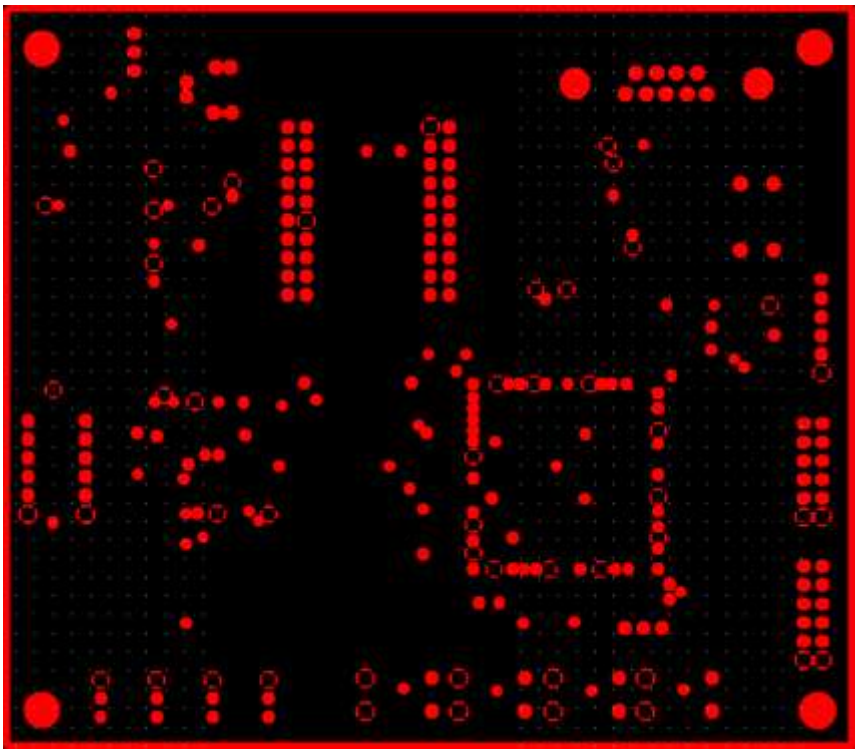


Fig. A.6 Archivo Gerber GP2

Anexo B

Configuración de circuito electrónico WIZNET 110S

El desarrollo de la comunicación Ethernet en nuestros prototipos construidos, se ha llevado a cabo a través de la configuración del módulo *WizNet110S* (Fig. B.1), el cual está basado en el chip W5100, el cuál integra tanto la pila TCP/IP como la MAC y capa física de Ethernet.

Este módulo funciona como un Gateway que convierte el protocolo RS-232 a protocolo *TCP/IP* y viceversa. Permite entre otras cosas dar conectividad Ethernet a los equipos que no lo poseen.

Fig. B.1 Board WizNet110S

Entre las características principales del módulo *WizNet110S* se tienen:

- Módulo todo incluido para RS-232 y Ethernet.
    - ✓ Simple y rápida implementación de la red
    - ✓ Disponibilidad de firmware para varios dispositivos seriales.
- Programa de configuración por medio del puerto serial y ethernet.
- Interfaz Ethernet 10/100 Mbps, Interfaz Serial Max 230 Kbps
- Compatible con RoHS

Las especificaciones generales son las mostradas a continuación, ver Fig. B.2:

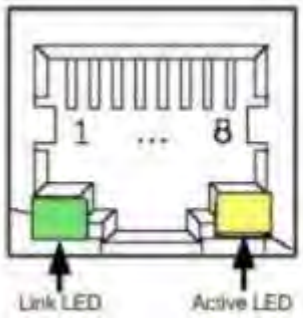
Arquitectura	TCP / IP	W5100
	PHY	Incluido en W5100
	Serial	RS-232
	MCU	GC89L591A0-MQ44I (compatible con 80C52) 62K Bytes in-system programmable(ISP) flash
Network	Interfaz	10/100 Mbps con detección automático. Conector RJ-45.
	Protocolo	TCP, UDP, IP, ARP, ICMP, MAC, PPPoE
Serial	Señales	TXD, RXD, RTS, CTS, GND
	Parametros	Parity : None, Even, Odd Data bits : 7,8 Flow Control : RTS/CTS, XON/XOFF
	Velocidad	Up to 230Kbps
Dimensiones		75mm X 45mm (PCB ancho X largo)
Voltaje de entrada		5V
Consumo de Potencia		DC 5V Under 180mA
Temperatura		0°C ~ 80°C (Operando) -40°C ~ 85°C (Embalado)
Humedad		10 ~ 90%

Tabla B.2 Especificaciones generales del módulo WizNet110S

Especificaciones técnicas:

- MCU 8051
- FLASH 62 KB (MCU interna)
- SRAM 16 KB (MCU interna)
- EEPROM 2KB (MCU interna)

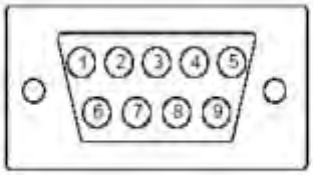
El diagrama de conexión del conector RJ-45 es el siguiente, ver Fig. B.3:



Pin	Signal
1	TX+
2	TX-
3	RX+
6	RX-

Fig. B.3 Conector RJ-45

El diagrama del conector RS-232 es el siguiente, ver Fig. B.4:



Pin Number	Signal	Description
1	NC	Not Connected
2	RxD	Receive Data
3	TxD	Transmit Data
4	DTR	Data Terminal Ready
5	GND	Ground
6	DSR	Data Set Ready
7	RTS	Request To Send
8	CTS	Clear To Send
9	NC	Not Connected

Fig. B.4 Conector RS-232

La configuración del módulo *WizNet*, se realiza a través del puerto Ethernet. Esta aplicación posee todo lo necesario para poder modificar los parámetros de red, velocidad de transmisión del puerto serial y opciones en general, ver Fig. B.5.

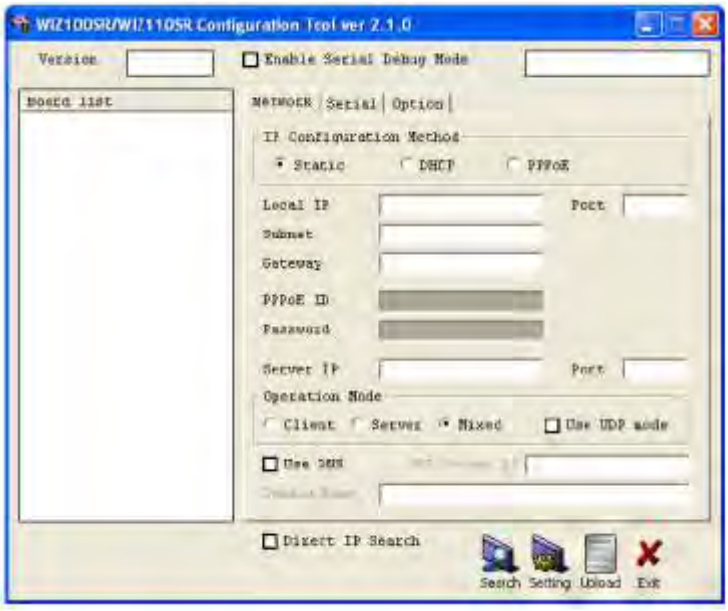


Fig. B.5 Ventana de configuración de módulo WiZnet

A través de la ventana mostrada en la Fig. B.6, se puede realizar diferentes configuraciones las cuales son mostradas a continuación.

Configuración de la red

A continuación se describirán los diferentes parámetros de configuración de la red:

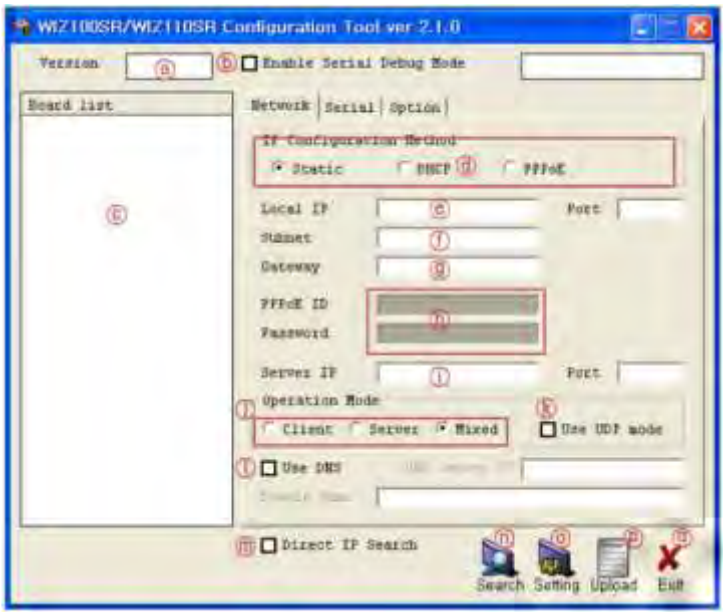


Fig. B.6 Ventana de configuración de la red

a) ***Versión:*** Muestra la versión del firmware.

b) ***Enable Serial Debug Mode:*** si está activado, se pueden monitorear el estado y los mensajes del socket del WIZ110SR (listen OK, connect fail etc) a través del puerto serial.

c) ***Board list:*** si se presiona el botón "**Search**" {n)}, todas las direcciones MAC de la subred serán mostradas.

d) ***IP configuration method:*** Selecciona el modo de configuración de la IP.

e) ***Server IP / Port:*** Cuando el WIZ110SR se encuentra en el modo **Cliente** o **Mixed**, la IP y puerto del servidor deben ser configurados aquí. Así el WIZ110SR se conectará a esta dirección IP, y al puerto indicado.

f) ***Operation Mode:*** Indica el modo de operación de la red. Se permiten cuatro modos. Modo Cliente; Modo Servidor; Modo Mixed; y Modo UDP.

g) ***Use DNS:*** Si se habilita esta opción, se debe ingresar el nombre de dominio del servidor DNS.

h) ***Direct IP Search:*** Esta opción se usa para buscar otros WIZ110SR no instalados, dentro de la misma subred.

i) ***Boton Search:*** sirve para buscar todos los módulos *WIZ110SR* dentro de la misma LAN. Los dispositivos encontrados serán mostrados por su MAC dentro de *"**Board List**"*.

j) ***Botón Setting:*** completa y activa los cambios realizados. Al presionarlo el módulo se reiniciará con los cambios, los cuales son grabados en la EEPROM.

k) ***Botón Upload:*** Con este botón, el firmware del módulo se actualizará a través de la red. Se necesitan de 20 a 30 segundos para que el equipo se inicialice.

l) ***Exit:*** Cierra el programa de configuración.

Configuración del puerto serial

Esta ventana (Fig. B.7), permite configurar la salida del puerto serial. Posee lo básico para ajustar un puerto serial:

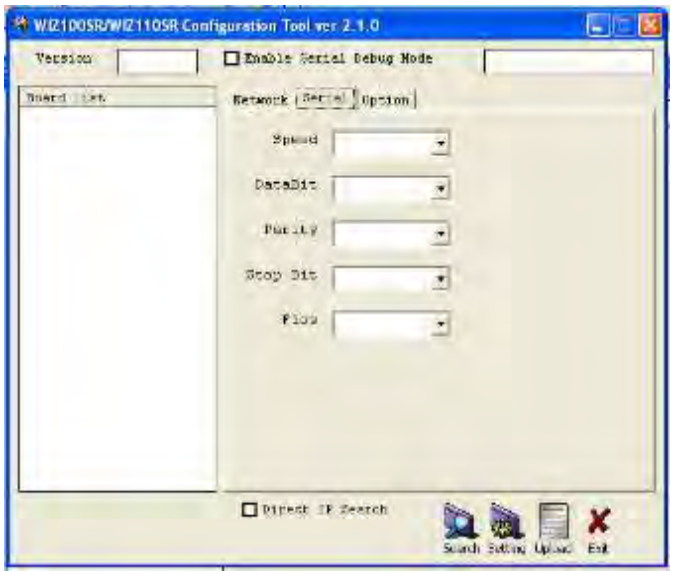


Fig. B.7 Ventana de configuración del puerto serial

a) Speed: Velocidad de conexión.

b) Databit: número de bit de datos.

c) Parity: Paridad.

d) Stop bit: bit de parada de los datos.

e) Flow: Selecciona el control de flujo por hardware, software o ninguno de los dos.

Una vez configurado los valores se debe presionar *"Setting"* para que los cambios tengan efecto y sean almacenados en el equipo, estos cambios quedan incluso después de apagar el módulo.

Opciones de configuración

Permite ajustar la configuración de la conexión y de los paquetes de datos, ver Fig. B.8.

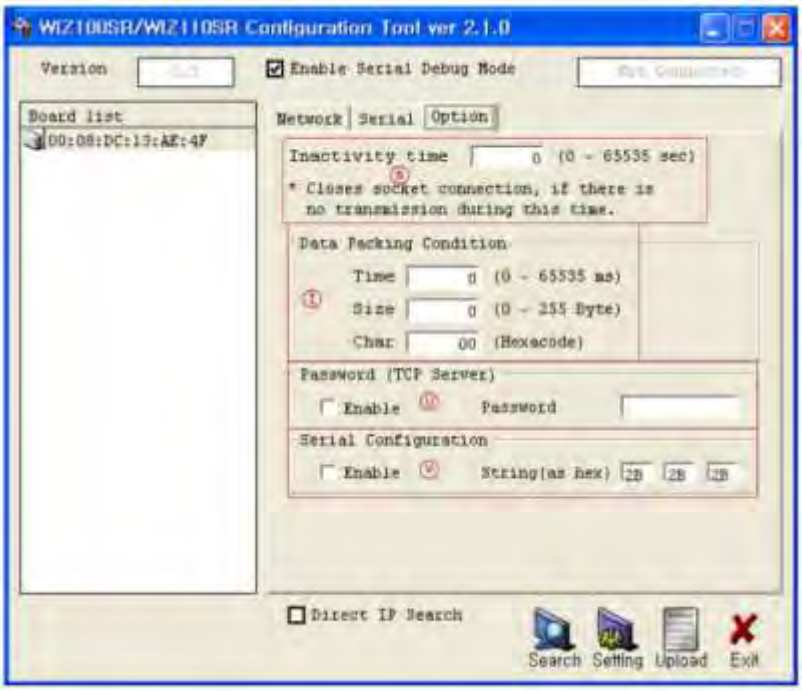


Fig. B.8 Ventana de opciones de configuración

(s) Inactivity time: Después de que se ha establecido la conexión, si ha pasado el tiempo indicado aquí, la conexión se cerrará. El valor por defecto es '0', para el cual esta función estará desactivada.

(t) Data Parking Condition: Indica cómo los datos son empaquetados para enviarlos por la salida Ethernet.

- ***Time:*** Si no hay datos nuevos por un tiempo dado por este campo, entonces toda la información guardada en el buffer de entrada será enviada por Ethernet.
- ***Size:*** cuando llegan los suficientes datos como para alcanzar este valor, se envían inmediatamente.
- ***Char:*** Los datos son recibidos hasta que se reciba un carácter especial, para el cual la información serán enviados.

(u) Password: sirve para bloquear el acceso de clientes a los cuales no se les permite la conexión. Esta función sólo puede operar en el modo Servidor TCP.

(v) Serial Configuration: Cuando se habilita esta opción, el módulo se puede configurar no utilizando la red, sino que el puerto serial.

Pruebas de configuración:

Conexión serial transparente entre 2 equipos:

Utilizamos esta configuración cuando se tiene equipos seriales que deben ser comunicados entre sí, utilizando la red Ethernet. Este tipo de conexión, permite tener el equipo serial a la misma distancia que la red de área local lo admita utilizando cableados Ethernet pre existentes.

Para crear esta conexión necesitamos 2 conversores conectados a la red local, uno en el PC y el otro en el equipo de conexión serial que se desea conectar.

La prueba fue hecha para la comunicación serial entre dos PC's. El esquema de la conexión es la siguiente, ver Fig. B.9:

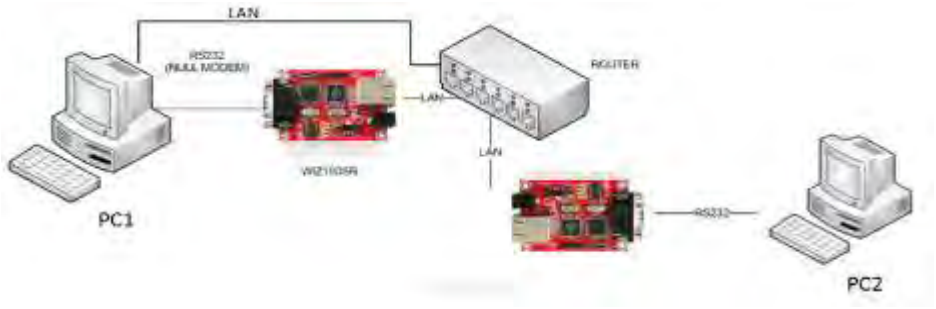


Fig. B.9 Esquema de conexión serial entre 2 pc's

En la Fig. B.9, se puede reemplazar la PC2 por el prototipo FPGA construido, habiéndoles dotado de comunicación alámbrica Ethernet.

Configuración de los equipos

Se deben conectar los equipos a la red de área local y alimentarlos con fuente de 5v, luego ejecutamos el programa de configuración de módulo desde cualquier PC que esté dentro de la misma red. En este caso la configuración se hará desde la PC1.

Una vez abierto el programa y conectados los equipos respectivamente, presionamos *"Search"* y aparecen en *"board list"* los dos equipos conectados a la red, ver Fig. B.10.

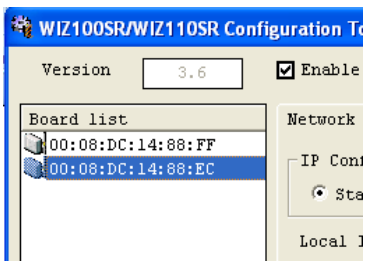


Fig. B.10 Ventana de configuración

La configuración que se llevo a cabo es la siguiente, ver Fig. B.11:

- ✓ **WIZ 1 (PC1)**

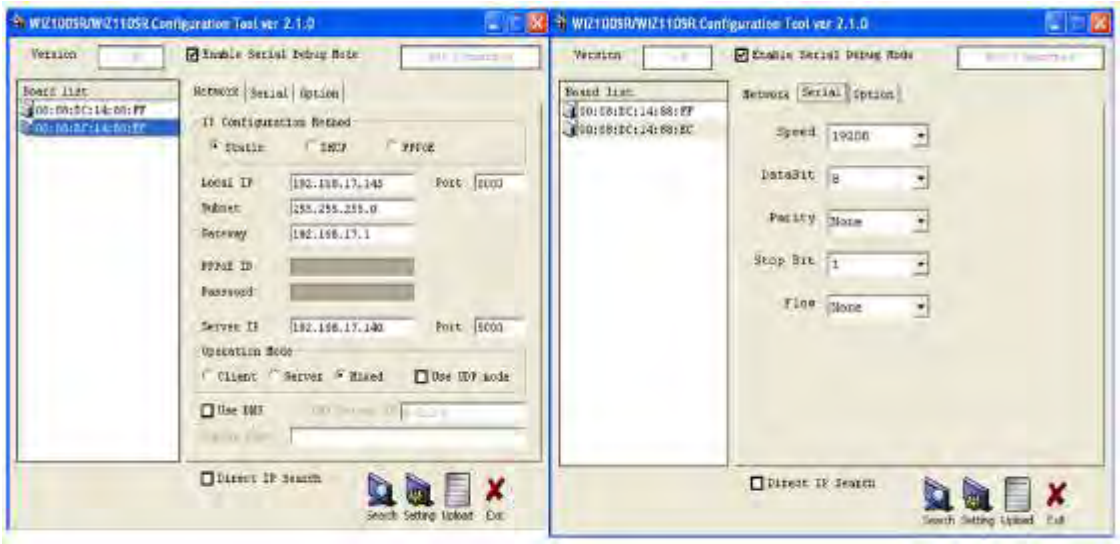


Fig. B.11 Ventana de configuración PC 1

Parámetros de configuración para PC 1, ver Fig. B.12:

PC1 ***(ipconfig)***

- ➢ **Ip : 192.168.17.137**
- ➢ **Subred : 255.255.255.0**
- ➢ **Mascara : 192.168.17.1**

- ✓ **WIZ 2 (PC2)**

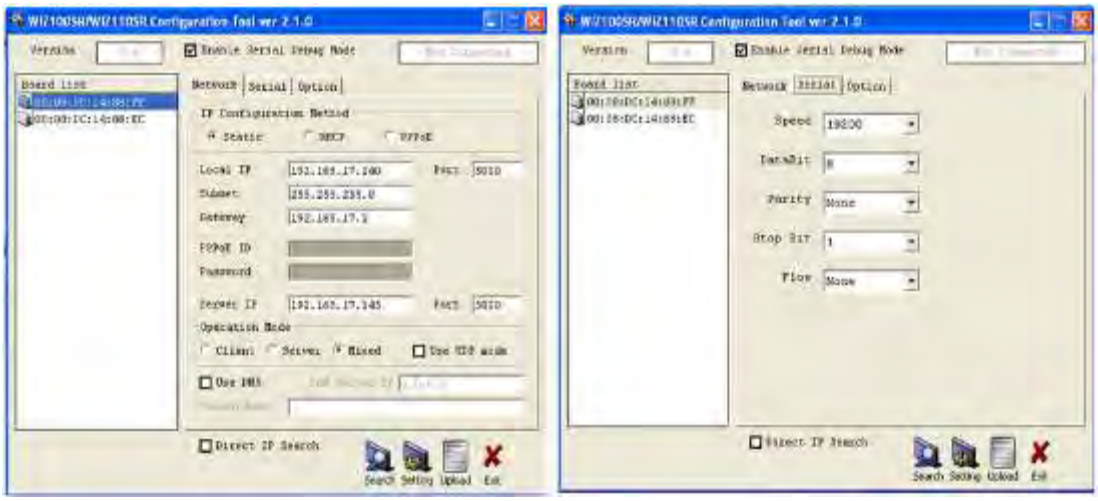


Fig. B.12 Ventana de configuración PC 2

Parámetros de configuración para PC2, ver Fig. B.13:

PC2 ***(ipconfig)***

- Ip : 192.168.17.142
- Subred : 255.255.255.0
- Mascara : 192.168.17.1

Lo que se hace al configurar cada equipo RS232 es asignarles a cada uno un IP el cual tiene que ser reconocido por la red local. Tomando los datos de la PC1, sabemos que su IP es 192.168.17.137, por lo tanto para asignar el IP al convertidor, ésta será de la forma 192.168.17.X, donde en este caso X = 130, de igual forma para el segundo convertidor, para el cual se le asigno la IP 192.168.17.145

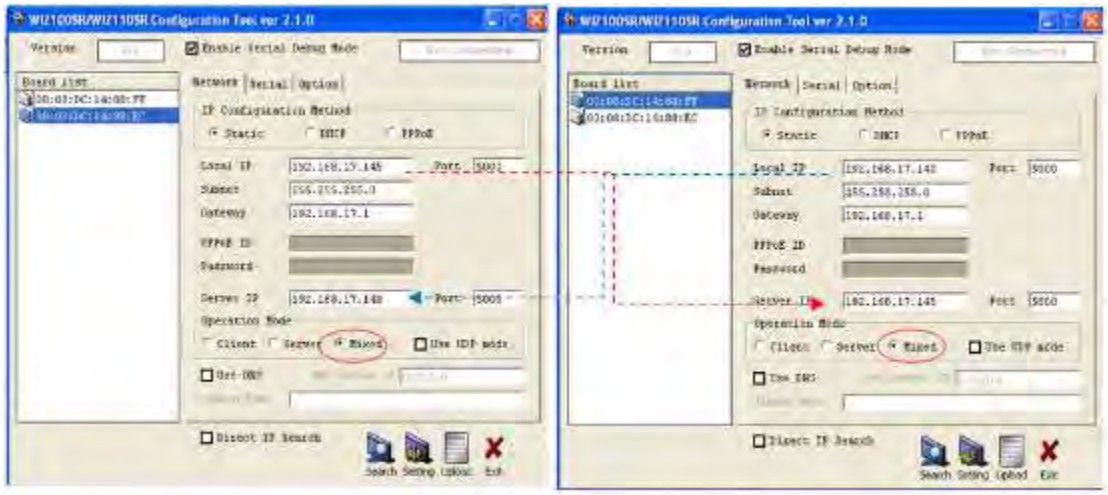


Fig. B.13 Ventana de configuración de ambos convertidores

Hay que tener en cuenta que en la configuración de ambos convertidores, se debe de señalar el modo de operación *"Mixed"* para que ambos dispositivos transmitan y reciban información por los puertos seriales, ver Fig. B.13.

Para la comprobación de esta prueba, se utilizó la aplicación Hyper Terminal, tras su previa configuración (Ver Fig.s B.14-B.16).

Fig. B.14 Ventana de configuración de aplicación Hyper terminal

Fig. B.15 Ventana de configuración de aplicación Hyper terminal

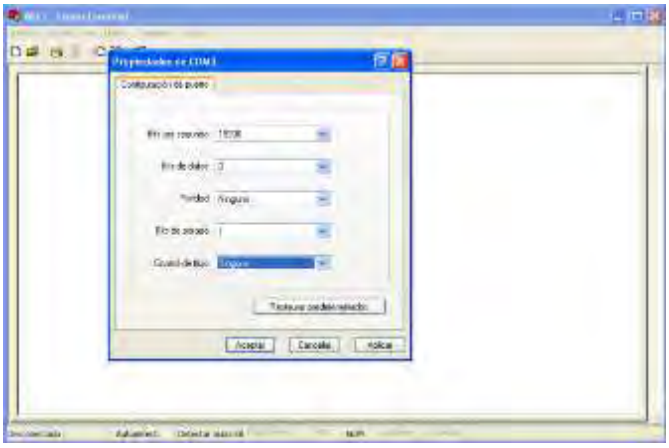


Fig. B.16 Ventana de configuración de aplicación Hyper terminal

Abriendo una ventana de comunicación en la PC1, se estableció la comunicación con el WIZ 2 mediante el puerto COM3 como se muestra en la Fig. B.17:

De manera análoga se realizan los mismos pasos para la configuración de la aplicación Hyper terminal en la PC2, para de esta manera establecer la comunicación con el WIZ 2.

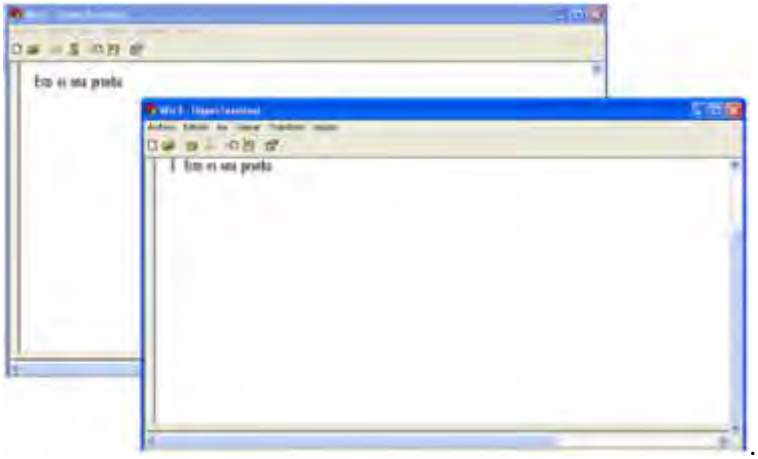


Fig. B.17 Ventanas de comunicación establecidas entre los WIZ

Conexión seria vía *WinSock*

Esta configuración se realiza cuando tenemos un equipo con conectividad Ethernet (PC) y otro con conectividad serial y queremos comunicarlos entre sí.

Para reconocer el *WIZ110SR*, se debe conectar el módulo con un cable de red a la red de área local en la cual estamos trabajando (Fig. B.18).

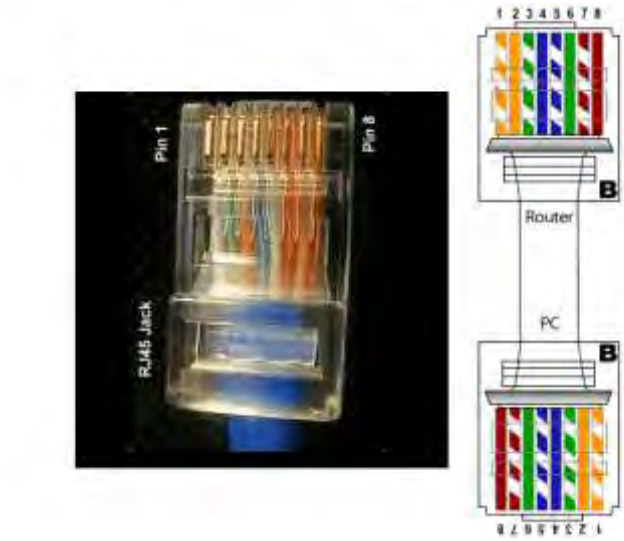


Fig. B.18 Cable de red (conexiones)

Ejecutamos el programa y presionamos el botón *SEARCH* con lo cual aparecerá la siguiente ventana, ver Fig. B.19:

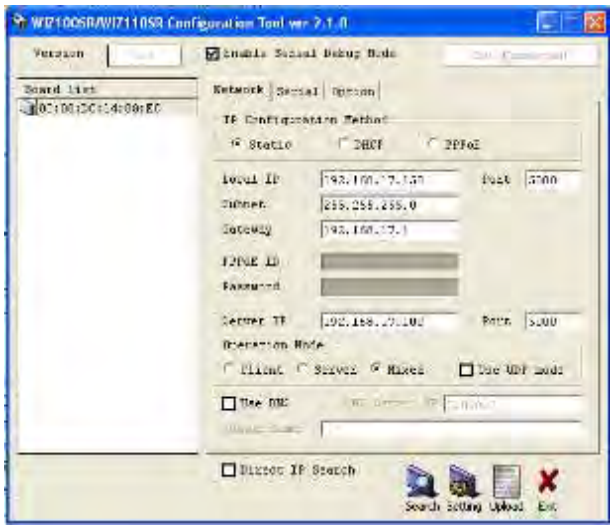


Fig. B.19 Ventana de configuración del WIZ110SR

Por defecto el módulo tendrá una dirección IP, subred y puerta de enlace (*Local IP, Subnet* y *Gateway* respectivamente).

Por lo tanto necesitamos configurar nuestro módulo para que pertenezca a la red local en la cual estamos trabajando.

La información que se utilizará será la dirección IP y la máscara de subred. La máscara indicará las direcciones IP que se encuentran dentro de la misma subred.

Para la configuración de la PC1 necesitamos la información de su IP, máscara de subred. Dicha información se puede obtener tras ingresar el comando *ipconfig* en la ventana de comandos de Windows, ver Fig. B.20:

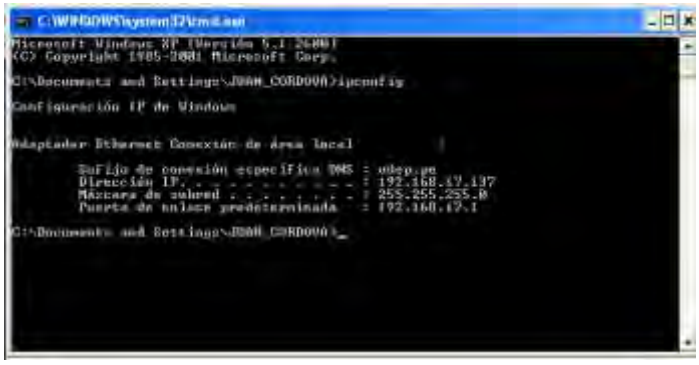


Fig. B.20 Ventana de comandos de Windows

Con esta información configuramos el dispositivo WIZ de la siguiente manera:

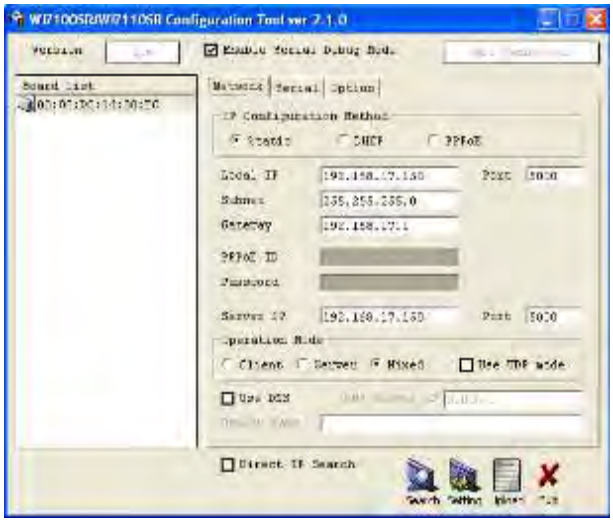


Fig. B.21 Configuración de dispositivo WIZ

Para dicha configuración simplemente se selecciona *BOARD LIST*, ver Fig. B.21, se escriben los nuevos valores y se presiona *SETTING.*

Para comprobar el funcionamiento se utilizó la aplicación *Hyper Terminal* anteriormente mencionada. Para ello se debe conectar el módulo al puerto serial por medio de una conexión *NULL MODEM*. Esta conexión se muestra en la Fig. B.22:

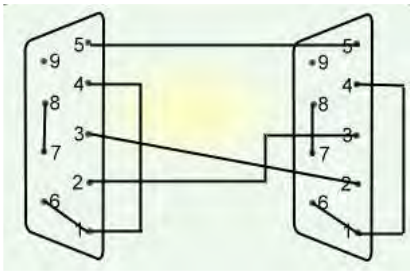


Fig. B.22 Conexión Null Modem

El esquema de la prueba a realizar es el siguiente, ver Fig. B.23:

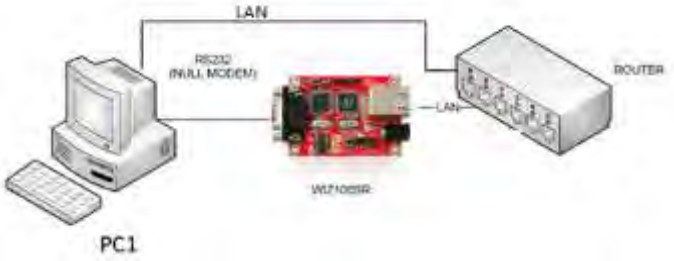


Fig. B.23 Esquema de conexión

Abrimos la aplicación *Hyper terminal* y creamos sesiones para la comunicación por el puerto serial COM3 y otra para la comunicación vía Ethernet como se muestra a continuación en las Fig. B.24, B.25 y B.26.

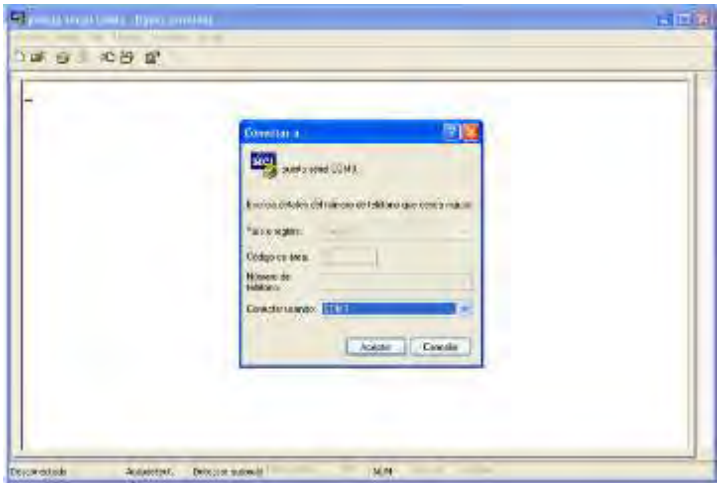


Fig. B.24 Configuración de comunicación serial

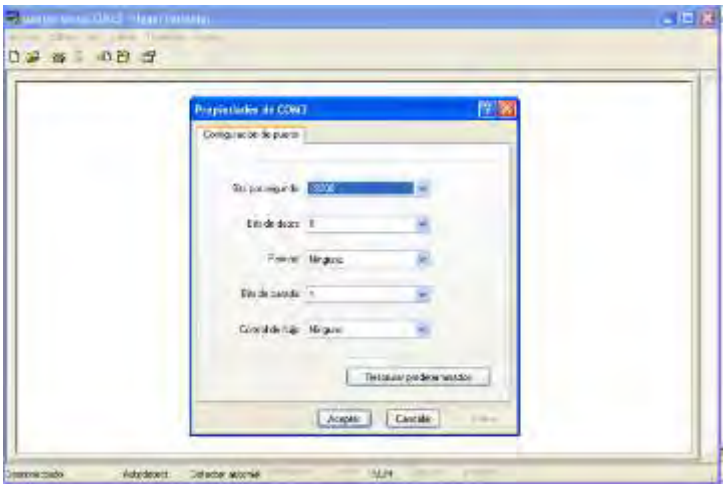


Fig. B.25 Configuración de comunicación serial

Fig. B.26 Configuración de comunicación Ethernet

Tener en cuenta que la dirección del host es la dirección IP asignada al módulo y con el mismo port (5000). La comunicación es de serial a Ethernet y viceversa (Figura B.27).

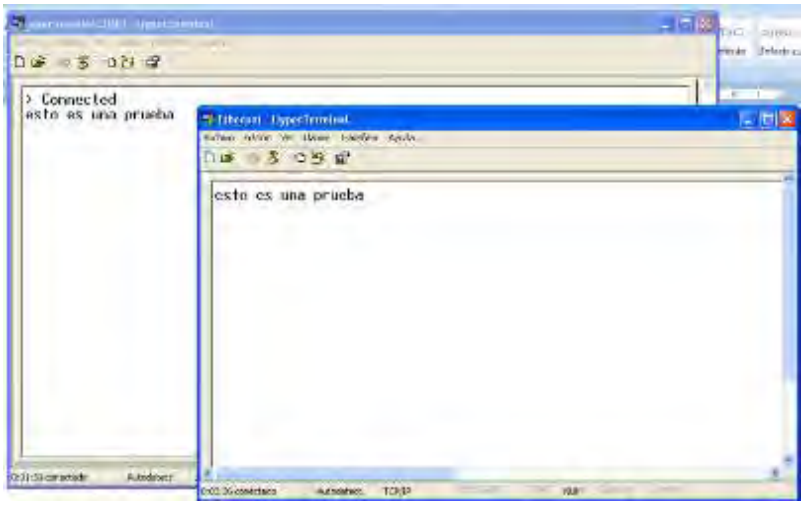


Fig. B.27 Ventanas de comunicación establecidas entre los Wiz

Anexo C

Manual FPGA board

Fig. C.1 Tarjeta PCB desarrollada

Este manual provee una descripción de la tarjeta prototipo *FPGA – UDEP,* Fig. C.1, de la familia *Spartan3E*, el cual cuenta con el dispositivo *XC3S500E-4pq208.*

Contiene las especificaciones técnicas de la PCB, así como los circuitos esquemáticos.
Es totalmente compatible con todas las versiones de las herramientas *Xilinx ISE WebPack* incluida la versión libre.

El módulo *FPGA – UDEP* dispone del FPGA *SPARTAN 3E* de la compañía *Xilinx*, tiene las siguientes características:

- capacidad de procesamiento de 2.7 ns.
- 500,000 puertas con una equivalencia de 5,508 Celdas,
- 500Kb de memoria en bloque.
- 73Kb de memoria distribuida,

- 20 multiplicadores dedicados.
- 4 unidades de control de reloj DCM
- Puerto de programación puerto JTAG.
- El consumo de Potencia máxima es de 5W.
- Los niveles de tensión de las señales son de 3.3 VDC.

La alimentación, viene dada por 5V DC, el cual por medio de un circuito integrado en la PCB, baja este voltaje de alimentación a 3 VDC, 2.5 VDC y 1.7 VDC, estos 3 niveles de tensión son aprovechados por el *FPGA* para su respectivo funcionamiento.

La Potencia del circuito alimentación es de 5W máximo, esto cuando todos los dispositivos periféricos están en funcionamiento.

La PCB diseñada, consta de los siguientes periféricos, Fig. C.2, los cuales van a llevar las distintas señales como son de comunicación, señal de sensores, señal de actuación.

La PCB cuenta con los siguientes periféricos:

- 2 canales de entrada Analógica.
- 4 canales de salida Analógica.
- 1 puerto de comunicación UART para PC.
- 4 entradas de Interruptores – entradas digitales.
- 5 entradas de pulsadores - entradas digitales.
- 8 salidas indicadores Led’s – salidas digitales.
- 4 conectores PMOD.
- 2 conectores de Propósito General – GPIO´s digitales.

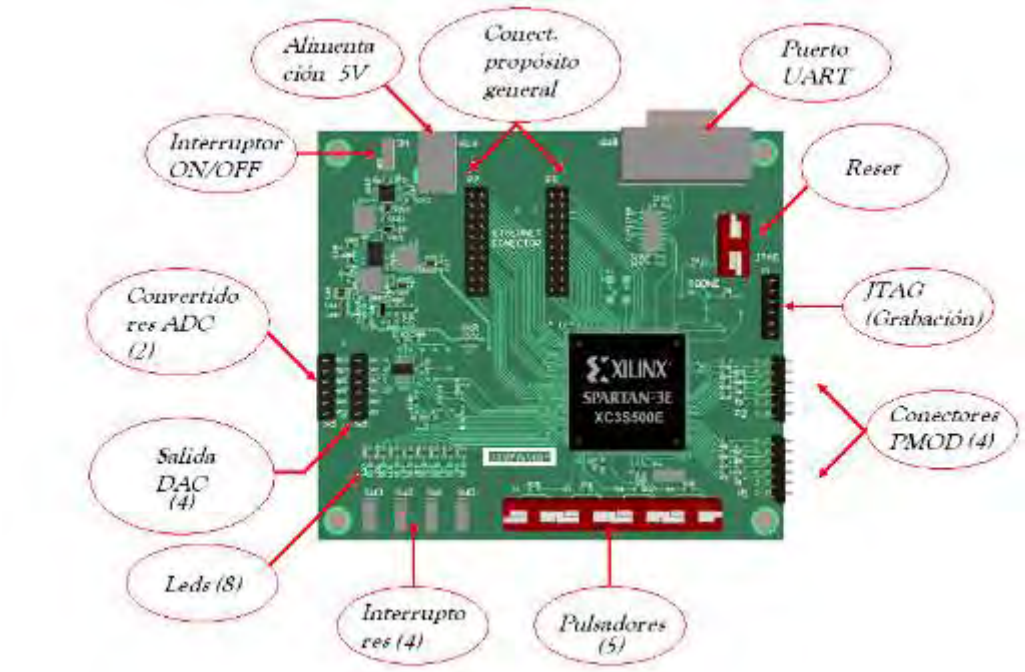


Fig. C.2 Interfaces de entrada y salida de la PCB

Periféricos

a. Pulsadores:

El prototipo **PCB**, tiene cuatro pulsadores, como se muestra en la Fig. C.3.

Estos pulsadores se encuentran en la parte inferior de la PCB, y se etiquetan desde S1 hasta S4. En la figura siguiente se tiene la dirección de pines en el FPGA.

Fig. C.3 Pulsadores de la tarjeta PCB

Dirección en el Integrado FPGA

```
Net "S1" loc = "P54" ;
Net "S2" loc = "P57" ;
Net "S3" loc = "P58" ;
Net "S4" loc = "P71" ;
```

b. Interruptores:

El prototipo **PCB,** tiene cuatro interruptores deslizantes, como se muestra en la Fig. C.4. Los interruptores deslizantes se encuentran en la parte inferior izquierda de la PCB, y se etiquetan desde SW0 hasta SW3.

Fig. C.4 Interruptores deslizantes

Cuando el interruptor deslizante se encuentre en el estado de ON (High = '1'), el interruptor conecta el pin del FPGA a 3.3V.

Cuando el interruptor deslizante se encuentre en la posición OFF (Low = '0'), el interruptor conecta este pin del FPGA a referencia.

Los interruptores suelen tener unos 2ms de rebote mecánico. Este diseño no presenta un circuito para eliminación de rebote, aunque circuitos como este tipo podría ser programado en el FPGA.

Dirección en el Integrado *FPGA*

```
Net "SW3" loc = "P20" ;
Net "SW2" loc = "P26" ;
Net "SW1" loc = "P32" ;
Net "SW0" loc = "P43" ;
```

c. Convertidores ADC:

El prototipo ***PCB***, incluye circuito de captura de señal analógica de dos canales, consiste en una amplificación programable pre-amplificador y un convertidor analógico-digital (ADC), como se muestra en la Fig. C.5.

Fig. C.5 Conector del ADC

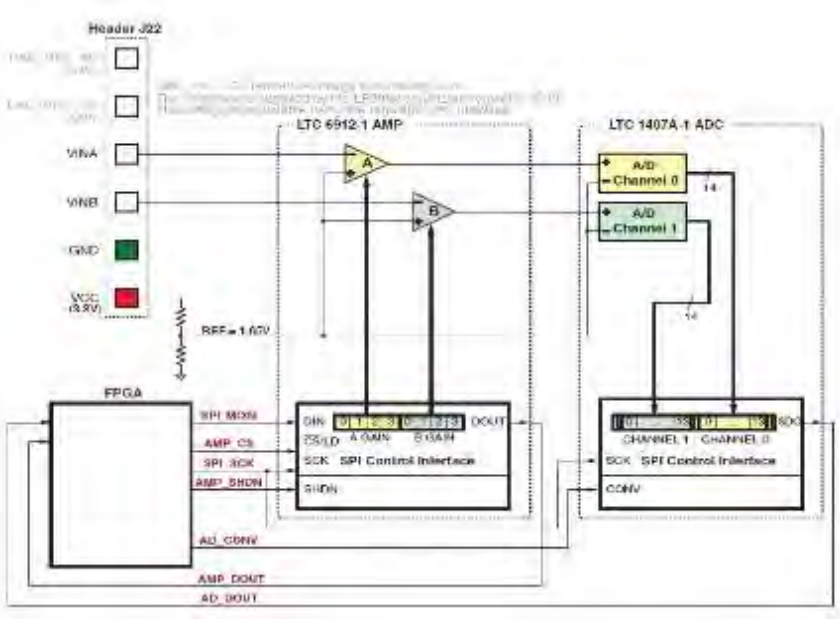


Fig. C.6 Esquema del funcionamiento del ADC

EL circuito de captura analógica, Fig. C.6, convierte la señal de ADC_A o ADC_B en una representación digital de 14 bits, D[13:0], según lo expresado en la siguiente ecuación:

$$D[13:0] = GAIN * \frac{(V_{in} - 1.65V)}{1.25V} * 8192$$

La ganancia es configurada en la programación del pre-amplificador. Los diversos ajustes permitidos para GAIN y tensiones admisibles aplicados al ADC_A y ADC_B entradas aparecen en la Tabla C.1.

La tensión de referencia para el amplificador y el ADC es 1.65V, generados a través de un divisor de tensión En consecuencia, 1.65V se resta de la tensión de entrada en ADC_A o ADC_B.

El alcance máximo del ADC es de ± 1,25 V, en torno a la tensión de referencia, 1.65V. Por lo tanto, 1.25V aparece en el denominador de la escala de entrada analógica.

Por último, este producto representa una salida digital en complemento a dos en 14 bits. Esta señal de complemento a dos, representa valores entre -2^{13} a $2^{13} - 1$.

El rango máximo de tención de entrada para circuito ADC es hasta 3.3 VDC, no se debe de superar este valor.

Amplificador Pre-Programable:

El LTC6912-1 dispone de dos amplificadores independientes, invirtiendo con ganancia programable.

La ganancia del amplificador es programada desde -1 a -100, ver tabla C.1.

Tabla C.1

Gain	A3	A2	A1	A0	Input Voltage Range	
	B3	B2	B1	B0	Minimum	Maximum
0	0	0	0	0		
-1	0	0	0	1	0.4	2.9
-2	0	0	1	0	1.025	2.275
-5	0	0	1	1	1.4	1.9
-10	0	1	0	0	1.525	1.775
-20	0	1	0	1	1.5875	1.7125
-50	0	1	1	0	1.625	1.675
-100	0	1	1	1	1.6375	1.6625

Control de interface *SPI*:

La Fig. C.7 destaca la interfaz de comunicaciones basada en *SPI* con el amplificador. La ganancia para cada amplificador se envía como una palabra de comando de ocho bits, que consta de dos campos de cuatro bits. El bit más significativo, B3, se envía primero.

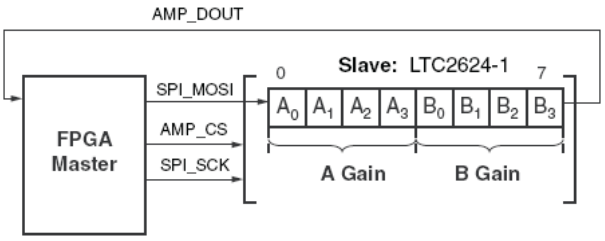


Fig. C.7 Control de interface *SPI*

La señal *AMP_DOUT* es una señal eco, puede ser ignorada para la mayoría de las aplicaciones.

La transacción bus SPI se inicia cuando la señal *AMP_CS* del *FPGA* cambia a bajo (vea la Fig. C.8). El amplificador captura los datos seriales del *SPI_MOSI* en cada flanco de subida del *SPI_CLK*.

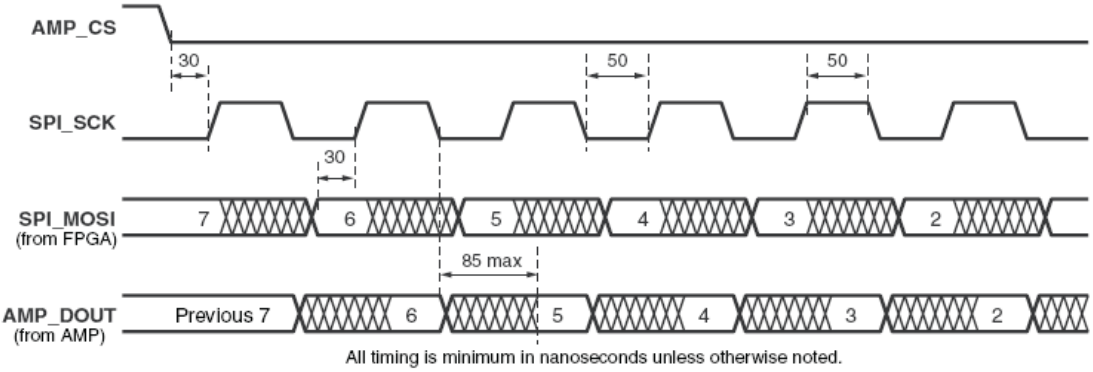


Fig. C.8 Configuración de señales de interface SPI

La interfaz del amplificador es relativamente lenta, apoyando sólo una frecuencia de reloj de 10 MHz.

Dirección en el Integrado FPGA

```
Net "AMP_CS" loc  =  "P33";
Net "SPI_CLK" loc = "P103";
Net "SPI_MOSI" loc = "P34";
Net "AMP_OUT" loc = "P14";
Net "SPI_SCK" loc = "P103";
Net "AD_CONV" loc = "P31";
Net "AD_DOUT" loc = "P30";
```

d. Convertidores DAC:

El prototipo **PCB** incluye un convertidor analógico digital (DAC) de cuatro canales de comunicación SPI. Es un dispositivo de *Linear Technology* LTC2624 DAC, con una resolución de 12-bits sin signo. Las cuatro salidas de la DAC aparecen en el conector P5, ver Fig. C.9.

Fig. C.9 Conector del DAC

La **PCB** utiliza una interfaz periférica serial *(SPI),* Fig. C.10, para comunicar valores digitales para cada uno de los cuatro canales del *DAC.* La comunicación SPI es *full-duplex*.

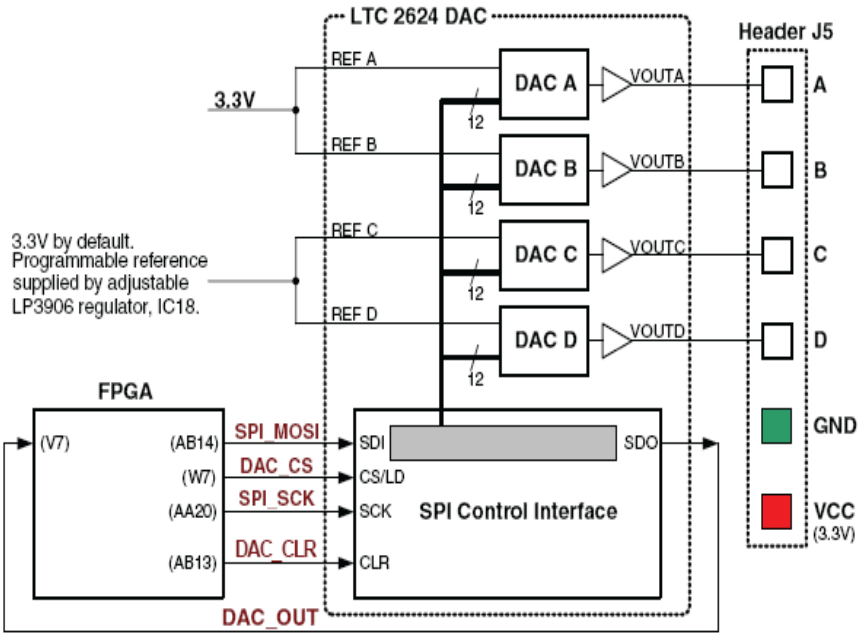


Fig. C.10 Esquema del funcionamiento del DAC

La tabla siguiente enumera las señales de interfaz entre *la FPGA* y el *DAC*. El *SPI_MOSI*, *DAC_OUT*, y las señales de SPI_SCK se comparten con otros dispositivos en el bus SPI.

La señal *DAC_CS* se activa en bajo. La señal es *DAC_CLR* se activa en bajo, esta es una entrada asíncrona de reset para el *DAC*.

Tabla C.2

Signal	FPGA Pin	Direction	Description
SPI_MOSI	AB14	FPGA→DAC	Serial data: Master Output, Slave Input
DAC_CS	W7	FPGA→DAC	Active-Low chip-select. Digital-to-analog conversion starts when this signal returns High.
SPI_SCK	AA20	FPGA→DAC	Clock
DAC_CLR	AB13	FPGA→DAC	Asynchronous, active-Low reset input
DAC_OUT	V7	FPGA←DAC	Serial data from the DAC

La Fig. C.11 muestra un ejemplo detallado del bus *SPI*. Cada bit se transmite o recibe en relación con la señal de reloj *SPI_SCK*.

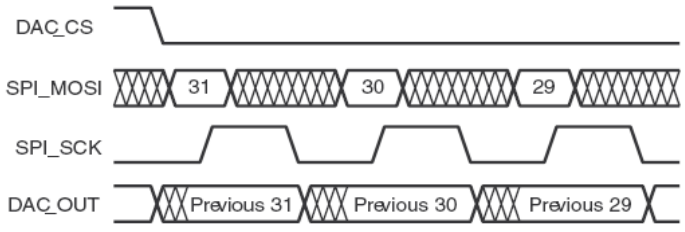


Fig. C.11 Configuración de señales de interface SPI

Cuando la señal *DAC_CS* se activa en bajo, la FPGA transmite datos sobre la señal SPI_MOSI, primero MSB.

El integrado *LTC2624* captura datos de entrada del *(SPI_MOSI)* en el flanco de subida del *SPI_SCK,* los datos deben ser validados durante al menos 4ns con respecto al flanco de subida del reloj.

El integrado *DAC LTC2624* transmites los datos por medio de la señal *DAC_OUT* en el flanco de bajada del *SPI_CLK*. El FPGA captura estos datos en el siguiente flanco de subida de *SPI_SCK*. El FPGA necesita leer primero el valor de *DAC_OUT* en el flanco de subida *SPI_CSK* después de que la señal *DAC_CS* se pone en bajo de lo contrario, el bit 31 se pierde.

Después de haber transmitido todos los 32 bits de datos, la FPGA se completa la transmisión del bus SPI cuando retorna la señal del DAC_CS a alto.

El rango máximo de tención de salida para circuito DAC es hasta 3.3 VDC.

Protocolo de comunicación:

La Fig. C.12, muestra el protocolo de comunicaciones necesaria para conectarse con el integrado *DAC LTC2624*. Es compatible con 24-bits y 32-bit. La interfaz *SPI* está formada por un registro de desplazamiento de 32-bit. Cada trama consta de una palabra de comando, una dirección, seguida por un valor de datos.

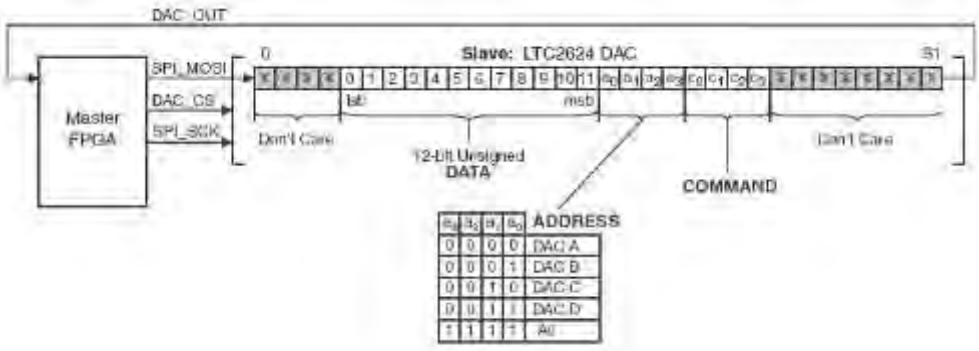


Fig. C.12 Control de interface SPI

Los primeros ocho bits de envío son ficticios, seguido por cuatro bits de comando. En la mayoría se utiliza el mando [3:0] = 0011, que actualiza inmediatamente la salida del *DAC* seleccionado con el valor de datos especificado. A raíz de la comando, *el FPGA* selecciona uno o todos los canales de salida del *DAC* a través de una dirección de cuatro bits. La *FPGA* envía un valor sin signo de 12-bit de datos que el DAC se convierte en un valor analógico en la salida seleccionada(s).

$$V_{out} = \frac{D[11:0]}{4096} * V_{ref}$$

Dirección en el Integrado FPGA

```
Net "DAC_OUT" loc = "P14";
Net "SPI_CLK" loc = "P103";
Net "SPI_MOSI" loc = "P34";
Net "DAC_CS" loc = "P35";
Net "DAC_CLR" loc = "P36";
```

e. Puerto JTAG:

Es el puerto por el cual grabaremos el programa realizado al FPGA, ver Fig. C.13, el archivo de descripción de hardware (.bit) previamente realizado en VHDL y luego sintetizado, se grabará al FPGA por este puerto. Posee una memoria RAM para asegurar que el programa no se pierda al dejar sin alimentación la Board.

Fig. C.13 Conector JTAG

f. Reset:

Este consta de un pulsador el cual da un Reset (S5), ver Fig. C.14. Si el programa descargado en la FPGA no ha sido grabado en memoria, la descripción de hardware programada en el FPGA desaparece por completo.

Fig. C.14 Interruptor de Reset de programa

g. Puerto UART:

La Fig. C.15 muestra el circuito para la comunicación UART, por protocolo RS-232, para la comunicación con el SCADA.

Fig. C.15 Conector DB9 Female

El puerto UART, ver Fig. C.16, se utilizará para una comunicación RS-232, a algún dispositivo externo o a la PC.

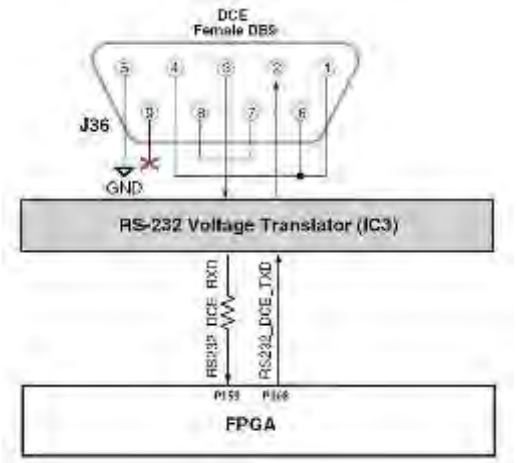


Fig. C.16 Control de datos comunicación UART

Por medio de este puerto se conectara el dispositivo Gateway *WizNet110RS*, Fig. C.17*,* el cual nos permitirá acceder a Ethernet.

Fig. C.17 Tarjeta WizNet110RS

h. Conectores PMOD:

Este tipo de conectores los utilizaremos para conectar algún dispositivo externo como una tarjeta PMOD ADC ó PMOD de puente H, estos periféricos son hardwares externos que se acoplan a la tarjeta para algún propósito específico, ver Fig. C.18. En esta Board contamos con 4.

Fig. C.18 Conectores PMOD

i. Conectores de Propósito General:

Fig. C.19 Conectores de propósito general

Este tipo de conectores nos proporciona una serie de I/O digitales al FPGA configurables por el usuario, ver Fig. C.19.

En la tabla C.3 y tabla C.4 tenemos la configuración de pines del dispositivo Ethernet WIZ812MJ hacia los pines del dispositivo FPGA XC500E-PQG208C.

Tabla C.3

CONECTOR P7				
PIN CONEC-P7	TIPO	SEÑAL	DESCRIPCIÓN	PIN FPGA
1	I	MOSI	Pin usado para configuración MOSI (Master Out Slave In)	"P32"
2	I/O	MISO	Pin usado para configuración MOSI (Slave In Master Out)	"P2"
3	I/O	D1	Pin usado para dirección [D7-D0]	"P5"
4	I/O	D0	Pin usado para dirección [D7-D0]	"P4"
5	I/O	D3	Pin usado para dirección [D7-D0]	"P9"
6	I/O	D2	Pin usado para dirección [D7-D0]	"P8"
7	I/O	D5	Pin usado para dirección [D7-D0]	"P12"
8	I/O	D4	Pin usado para dirección [D7-D0]	"P11"
9	I/O	D7	Pin usado para dirección [D7-D0]	"P16"
10	I/O	D6	Pin usado para dirección [D7-D0]	"P15"
11	P	GND	Pin de tierra	"GND"
12	P	3V3	Pin de alimentación de 3.3V	"3V3"
13	I	A8	Pin usado para dirección [D14-D8]	"P19"
14	I	A9	Pin usado para dirección [D14-D8]	"P18"
15	I	A10	Pin usado para dirección [D14-D8]	"P23"
16	I	A11	Pin usado para dirección [D14-D8]	"P22"
17	I	A12	Pin usado para dirección [D14-D8]	"P25"
18	I	A13	Pin usado para dirección [D14-D8]	"P24"
19	I	A14	Pin usado para dirección [D14-D8]	"P28"
20		NC	Pin no conectado	N/C

Tabla C.4

CONECTOR P8				
PIN CONEC-P8	TIPO	SEÑAL	DESCRIPCIÓN	PIN FPGA
1	P	3V3	Pin de alimentación de 3.3V	3V3
2	I	RESET	Pin de reset	"P178"
3	I	SCLK	Pin usado para señal de SPI clock	"P180"
4	I	SCS	Pin usado para habilitar el modo SPI	"P179"
5	I	WR	Pin para habilitar modo de escritura de datos del W5100	"P185"
6	I	RD	Pin para habilitar modo de lectura de datos del W5100	"P181"
7	I	CS	pin habilitar modulación de señal del W5100	"P187"
8	O	INT	Después de la recepción o transmisión indica que el WC5100 requiere atención. Se activa en bajo	"P186"
9	O	RX_LED	Indica actividad de señal de recepción de datos	"P190"
10	O	TX_LED	Indica actividad de señal de transmisión de datos	"P189"
11	I	A0	Pin usado para dirección [A7-A0]	"P196"
12	I	A1	Pin usado para dirección [A7-A0]	"P192"
13	I	A2	Pin usado para dirección [A7-A0]	"P199"
14	I	A3	Pin usado para dirección [A7-A0]	"P197"
15	I	A4	Pin usado para dirección [A7-A0]	"P202"
16	I	A5	Pin usado para dirección [A7-A0]	"P200"
17	I	A6	Pin usado para dirección [A7-A0]	"P205"
18	I	A7	Pin usado para dirección [A7-A0]	"P203"
19	O	LINKLED	Indica un buen estado del enlace para 10/100M	"P206"
20	P	GND	Pin de tierra	GND

Anexo D

Identificación y control de tramo altamente no lineal

Las figuras siguientes muestran la respuesta del proceso ante una señal de tipo escalón.

Este experimento de identificación se ha realizado en un tramo donde el proceso presente una no-linealidad bastante fuerte. A continuación se muestras las respuestas de la variable manipulable (entrada) y la variable del proceso (salida) respectivamente.

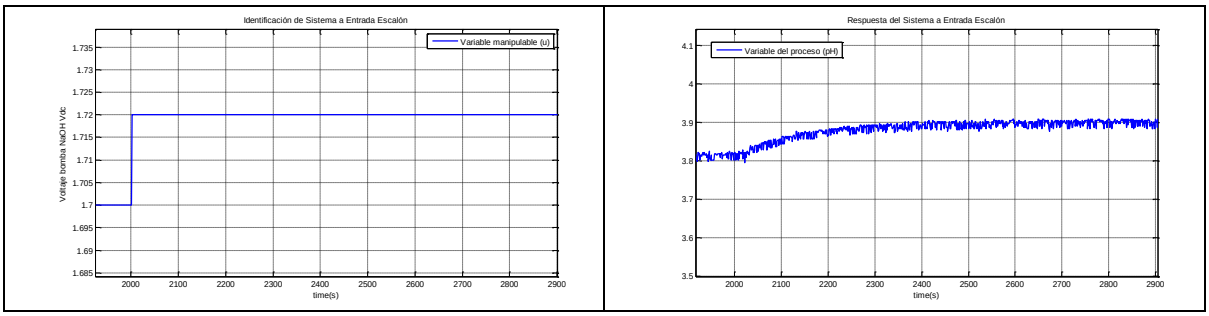


Fig. D.1 Respuesta del sistema ante entrada escalón de 1.7 – 1.72 Vdc

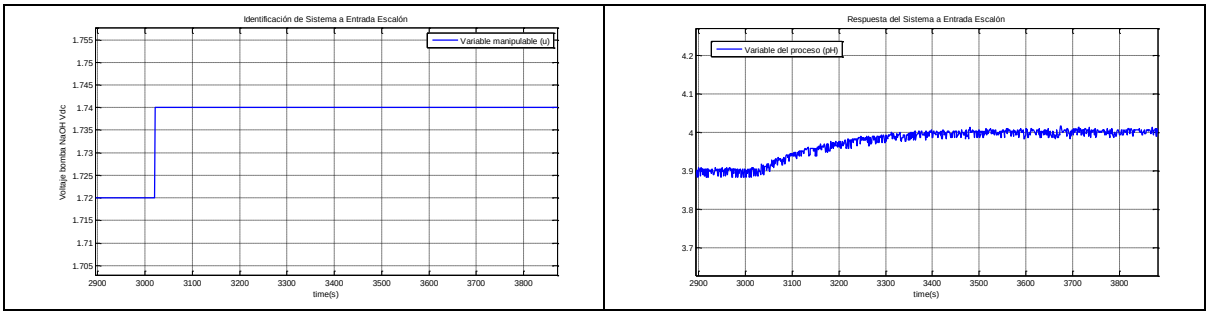


Fig. D.2 Respuesta del sistema ante entrada escalón de 1.72 – 1.74 Vdc

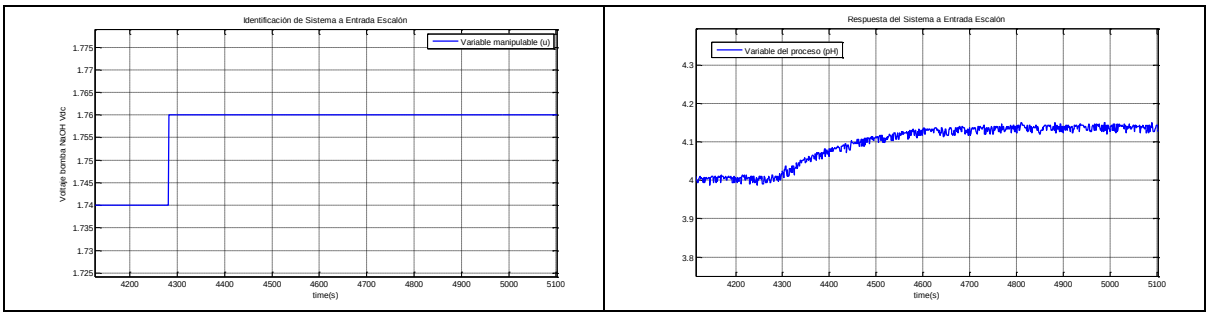


Fig. D.3 Respuesta del sistema ante entrada escalón de 1.74 – 1.76 Vdc

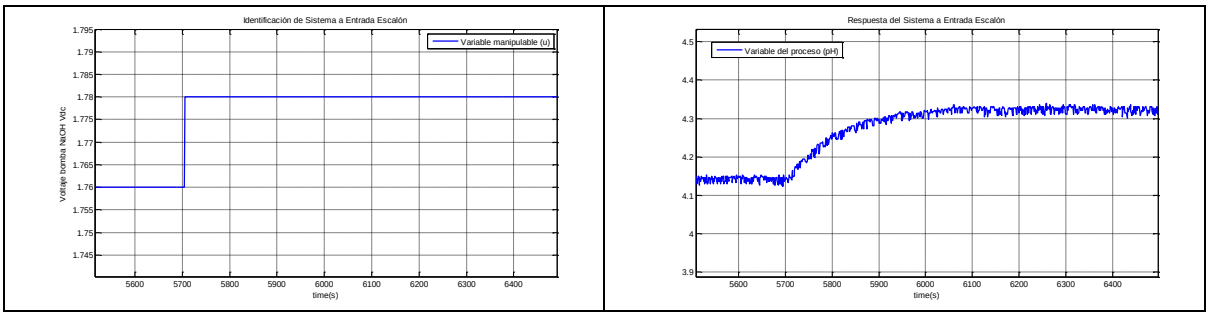


Fig. D.4 Respuesta del sistema ante entrada escalón de 1.76 – 1.78 Vdc

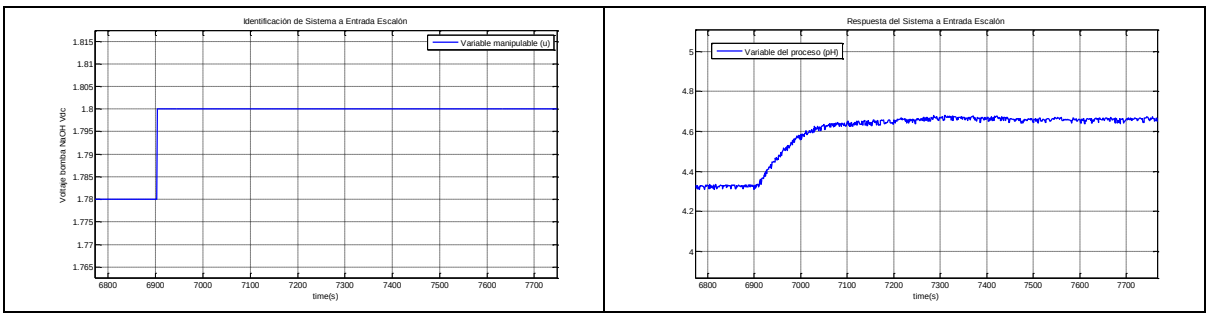


Fig. D.5 Respuesta del sistema ante entrada escalón de 1.78 – 1.8 Vdc

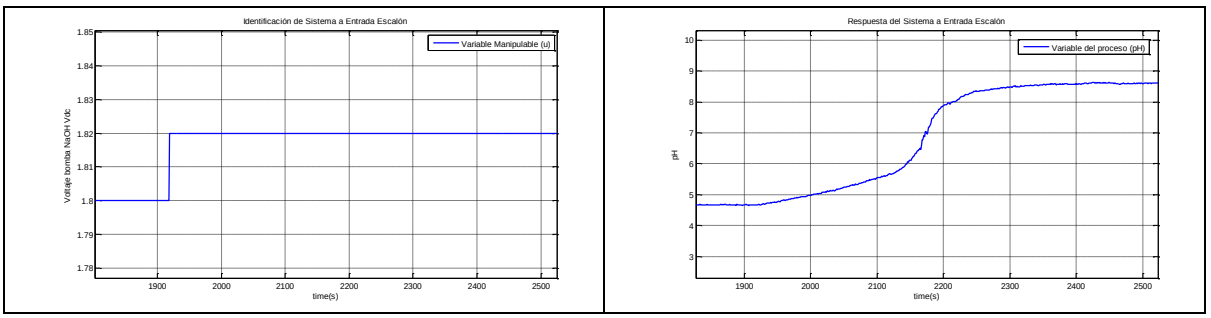


Fig. D.6 Respuesta del sistema ante entrada escalón de 1.8 – 1.82 Vdc

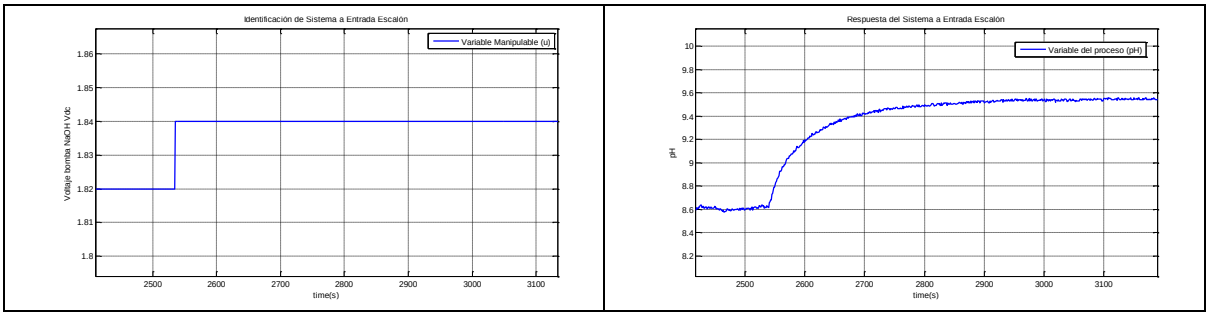


Fig. D.7 Respuesta del sistema ante entrada escalón de 1.82 – 1.84 Vdc

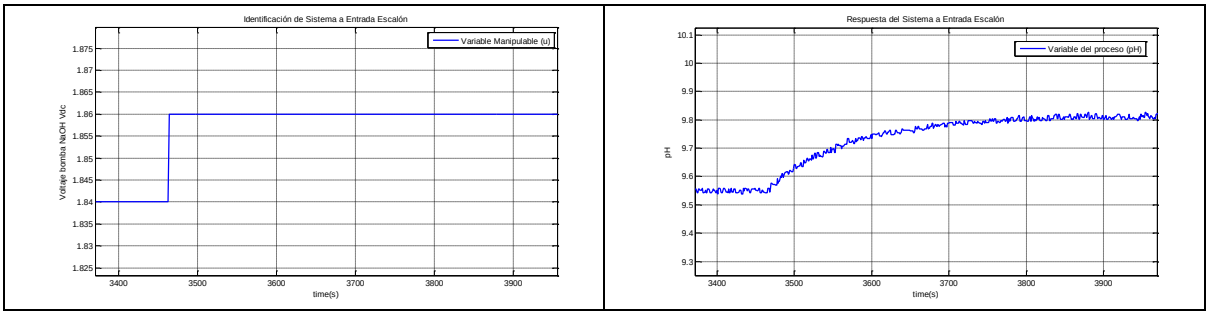


Fig. D.8 Respuesta del sistema ante entrada escalón de 1.84 – 1.86 Vdc

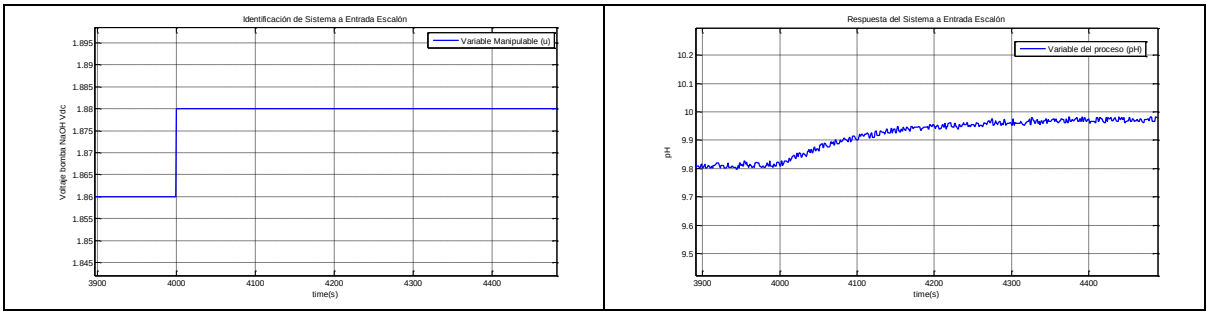


Fig. D.9 Respuesta del sistema ante entrada escalón de 1.86 – 1.88 Vdc

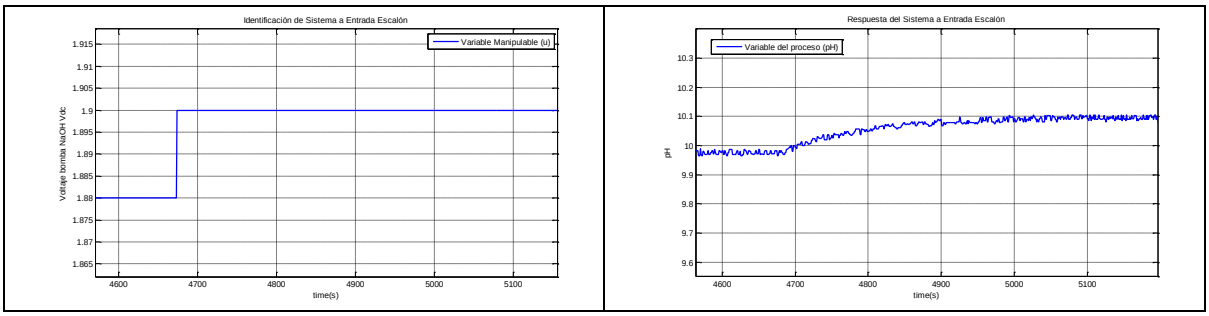


Fig. D.10 Respuesta del sistema ante entrada escalón de 1.88 – 1.9 Vdc

La información del experimento de identificación se ha almacenado en una base de datos, la cual se configura desde el sistema SCADA.

A continuación se muestra en la Tabla D.1 las ganancias estáticas del sistema.

Tabla D.1 Ganancias estáticas del proceso

u_i	u_f	Δu	y_i	y_f	Δy	K
1.7	1.72	0.02	3.81	3.9	0.09	4.5

1.72	1.74	0.02	3.9	4	0.1	5
1.74	1.76	0.02	4	4.14	0.14	7
1.76	1.78	0.02	4.14	4.32	0.18	9
1.78	1.8	0.02	4.32	4.66	0.34	17
1.8	1.82	0.02	4.66	8.6	3.94	197
1.82	1.84	0.02	8.6	9.54	0.94	47
1.84	1.86	0.02	9.54	9.81	0.27	13.5
1.86	1.88	0.02	9.81	9.94	0.13	6.5
1.88	1.9	0.02	9.97	10.1	0.13	6.5

En la Fig. D.11 graficamos la ganancia estática respecto a la variable manipulable (MV); esto para analizar la no-linealidad del sistema. En esta gráfica se aprecia que el sistema tiene un comportamiento no-lineal, con una marcada característica no-lineal, que corresponden a valores de en torno a 7 de pH.

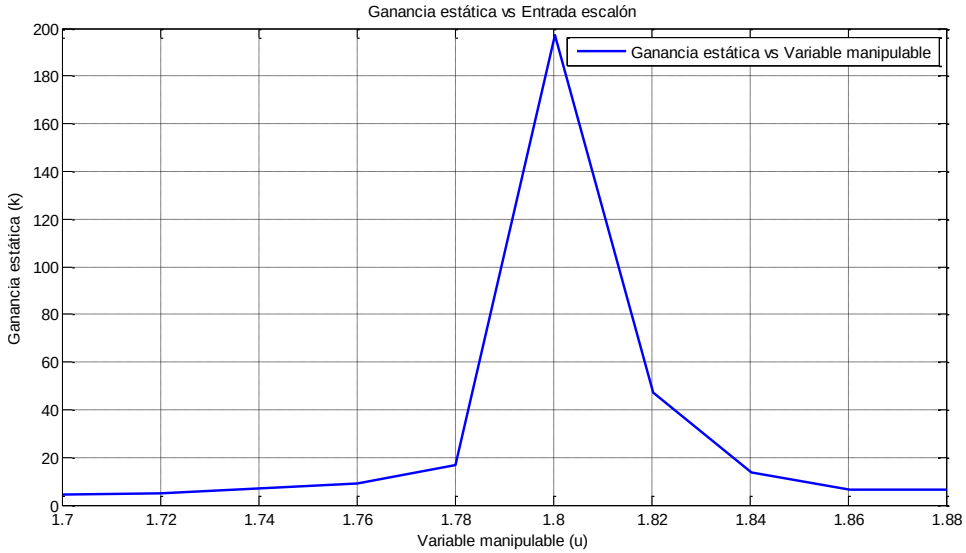


Fig. D.11 Ganancia estática respecto a la variable manipulable

Los modelos obtenidos a continuación corresponden a los rangos obtenidos mediante el experimento de identificación.

En la Tabla D.2 se presentan dicho modelos de funciones de transferencia.

Tabla D.2 Funciones de transferencia

Tramo	Rango	T(s)
1	1.7-1.72	$\frac{4.5}{120s+1}$
2	1.72-1.74	$\frac{5}{140s+1}$

3	1.74-1.76	$\frac{7}{170s+1}$
4	1.76-1.78	$\frac{9}{96s+1}$
5	1.78-1.8	$\frac{17}{57s+1}$
6	1.8-1.82	$\frac{197}{259s+1}$
7	1.82-1.84	$\frac{47}{65s+1}$
8	1.84-1.86	$\frac{13.5}{97s+1}$
9	1.86-1.88	$\frac{6.5}{80s+1}$
10	1.88-1.9	$\frac{6.5}{107s+1}$

La tabla D.1 nos muestra las ganancias estáticas. El experimento de identificación ha sido desarrollado dando escalones con incrementos de 0.02 Vdc. Hemos adquirido 10 tramos en esta región.

Se observa en las funciones de transferencia obtenidas, que las ganancias estáticas van aumentando hasta llegar a un máximo para luego disminuir nuevamente.

Mientras que las constantes de tiempo, TAO, tienen un comportamiento que varía en cada tramo.

Se puede concluir de esto, que el sistema tiene un comportamiento altamente no lineal en esta región.

Tramo 1:

En este tramo correspondiente a valores de la variable manipulable (VM), comprendidos entre 1.72-1.74. Se consideró la siguiente función de transferencia:

$$T(s) = \frac{4.5}{120s+1}$$

Los parámetros del controlador son los siguientes:

Kp = 8.6 Ti = 169.5

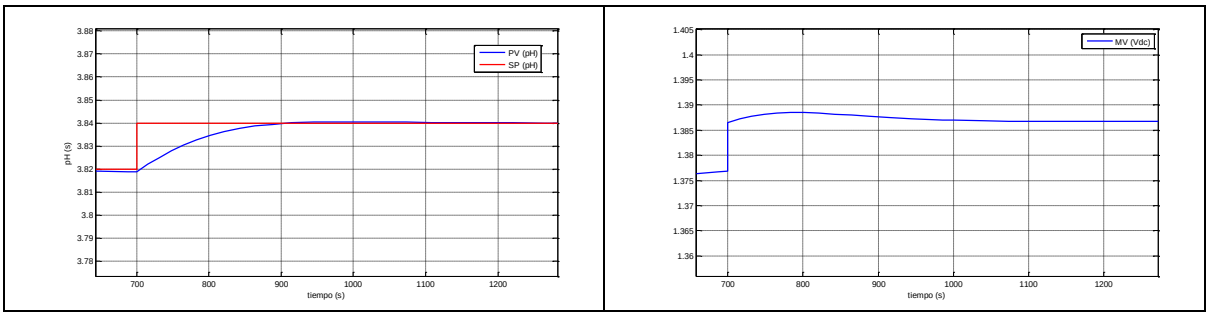


Fig. D.12 Respuesta del sistema ante un cambio de setpoint de pH de 3.84

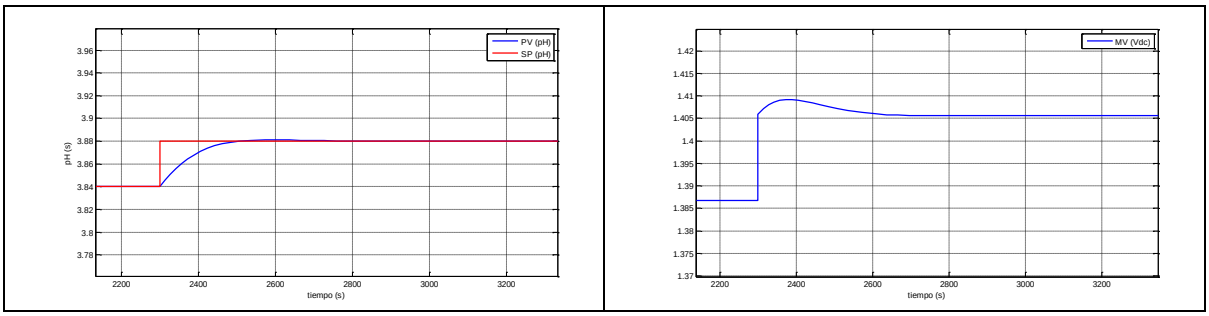


Fig. D.13 Respuesta del sistema ante un cambio de setpoint de pH de 3.88

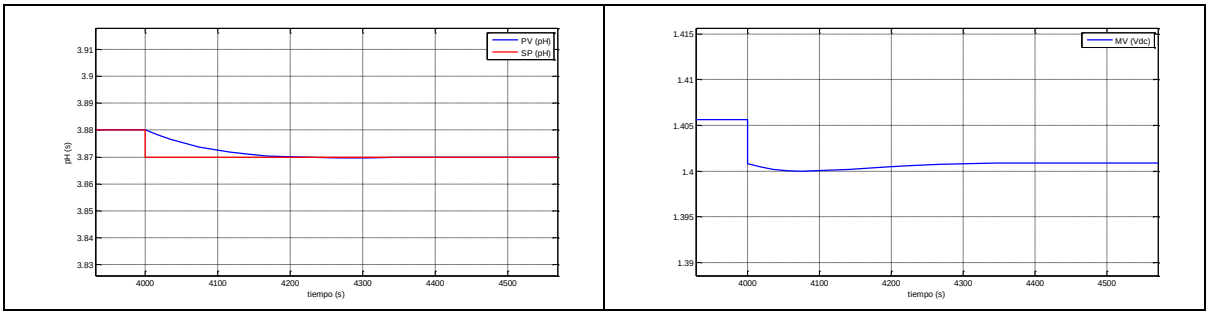


Fig. D.14 Respuesta del sistema ante un cambio de setpoint de pH de 3.87

Tramo 2:

En este tramo correspondiente a valores de la variable manipulable (VM), comprendidos entre 2.33 a 2.47. Se consideró la siguiente función de transferencia:

$$T(s) = \frac{5}{140s + 1}$$

Los parámetros del controlador son los siguientes:

$$\text{Kp} = 0.87 \qquad \text{Ti} = 91$$

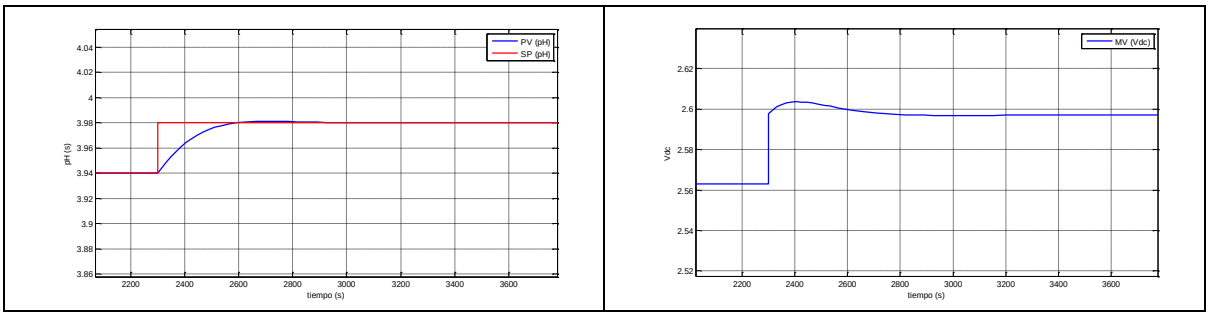


Fig. D.15 Respuesta del sistema ante un cambio de setpoint de pH de 3.98

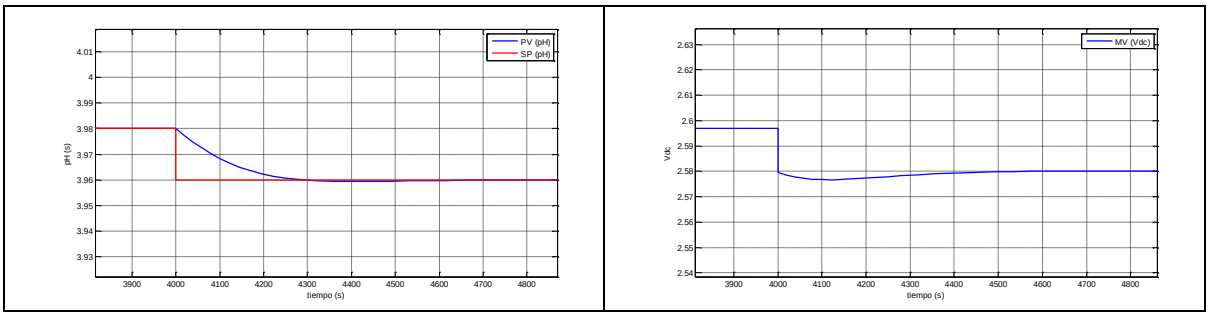


Fig. D.16 Respuesta del sistema ante un cambio de setpoint de pH de 3.96

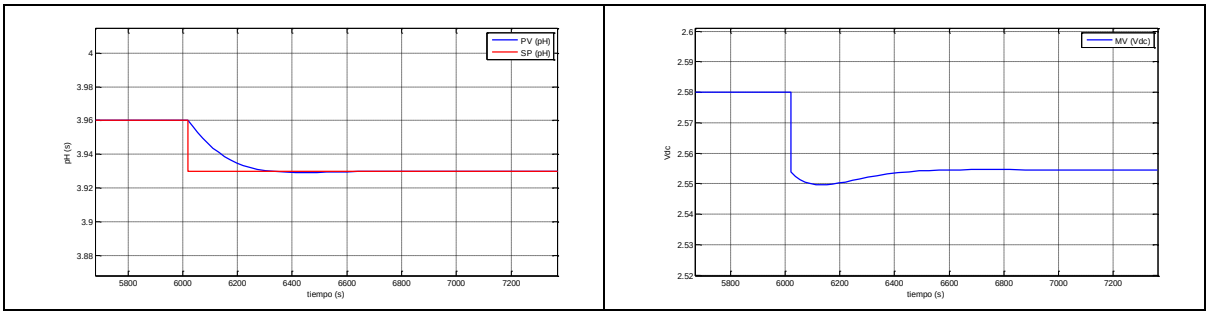


Fig. D.17 Respuesta del sistema ante un cambio de setpoint de pH de 3.93

Tramo 3:

En este tramo correspondiente a valores de la variable manipulable (VM), comprendidos entre 1.74-1.76. Se consideró la siguiente función de transferencia:

$$T(s) = \frac{7}{261s + 1}$$

Los parámetros del controlador son los siguientes:

Kp = 0.62 Ti = 110

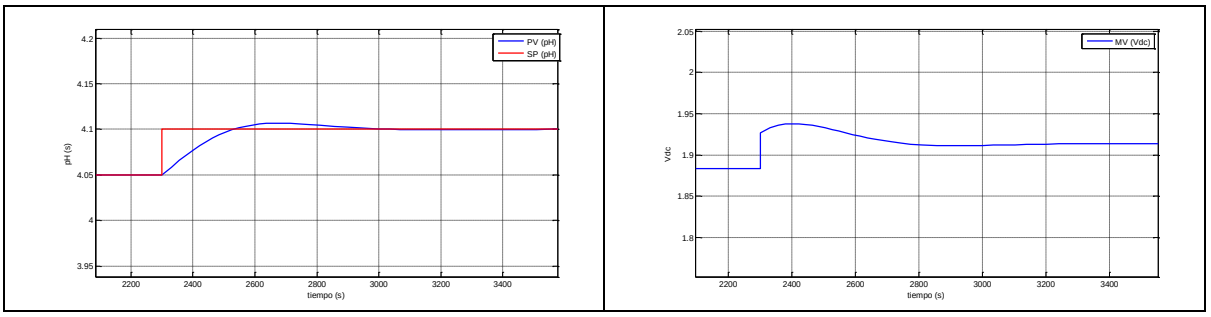


Fig. D.18 Respuesta del sistema ante un cambio de setpoint de pH de 4.1

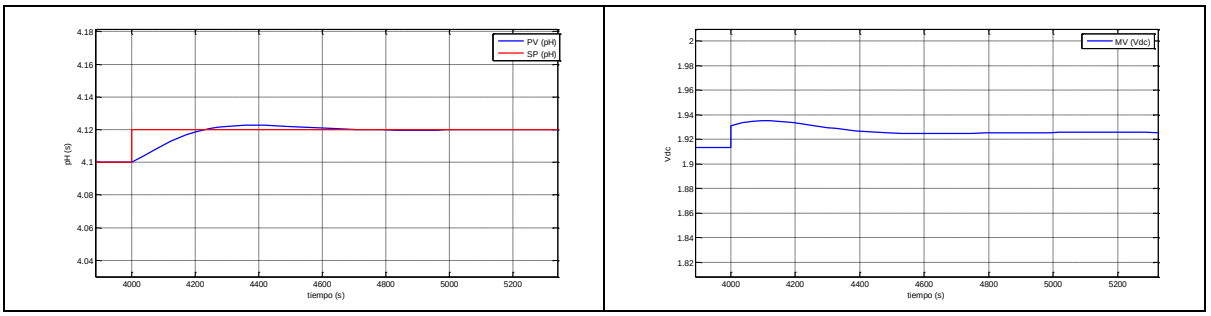


Fig. D.19 Respuesta del sistema ante un cambio de setpoint de pH de 4.12

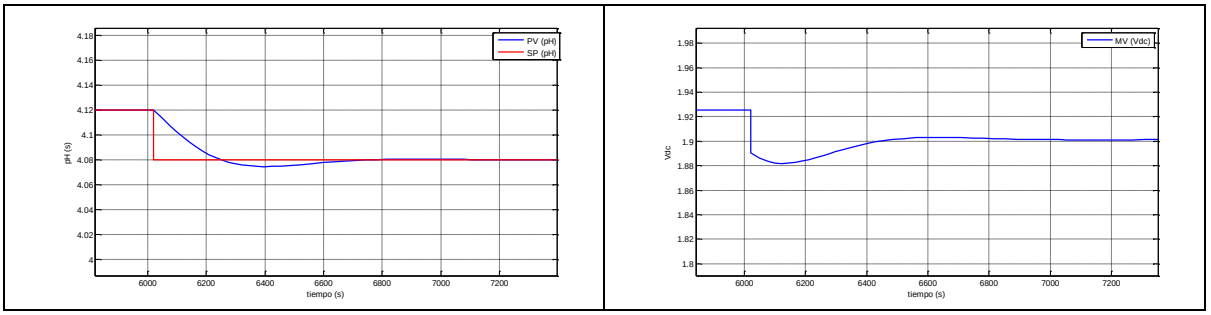


Fig. D.20 Respuesta del sistema ante un cambio de setpoint de pH de 4.08

Tramo 4:

En este tramo correspondiente a valores de la variable manipulable (VM), comprendidos entre 1.76-1.78. Se consideró la siguiente función de transferencia:

$$T(s) = \frac{9}{96s + 1}$$

Los parámetros del controlador son los siguientes:

Kp =0.48 Ti = 62.5

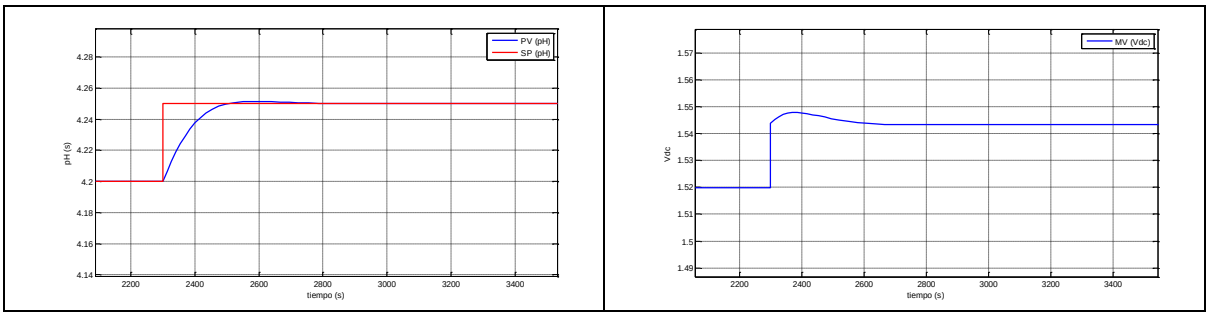


Fig. D.21 Respuesta del sistema ante un cambio de setpoint de pH de 4.25

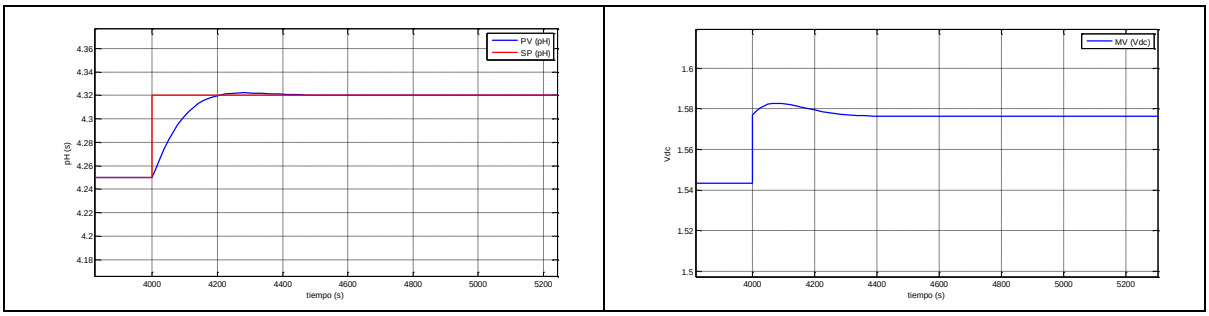


Fig. D.22 Respuesta del sistema ante un cambio de setpoint de pH de 4.32

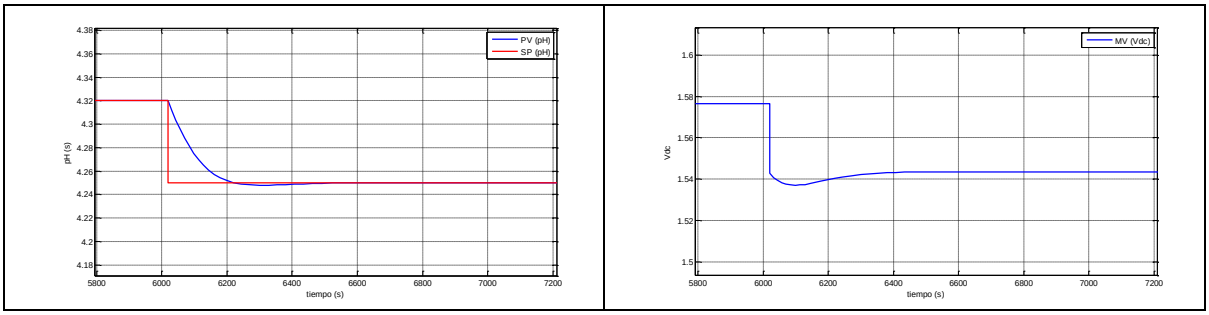


Fig. D.23 Respuesta del sistema ante un cambio de setpoint de pH de 4.25

Tramo 5:

En este tramo a valores de la variable manipulable (VM), comprendidos entre 1.78-1.8. Se consideró la siguiente función de transferencia:

$$T(s) = \frac{17}{57s + 1}$$

Los parámetros del controlador son los siguientes:

Kp = 0.25 Ti = 37

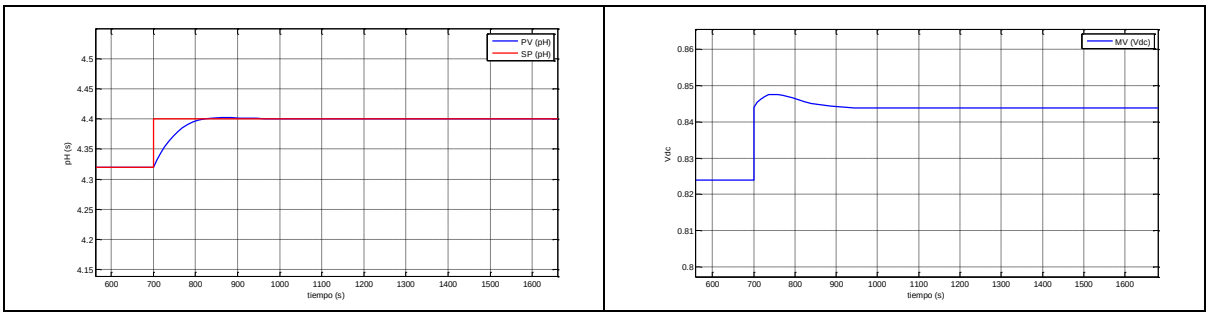


Fig. D.24 Respuesta del sistema ante un cambio de setpoint de pH de 4.4

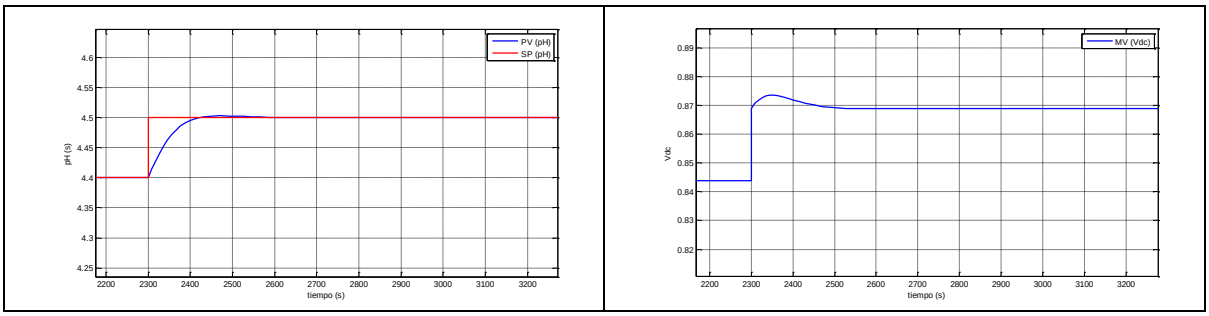


Fig. D.25 Respuesta del sistema ante un cambio de setpoint de pH de 4.5

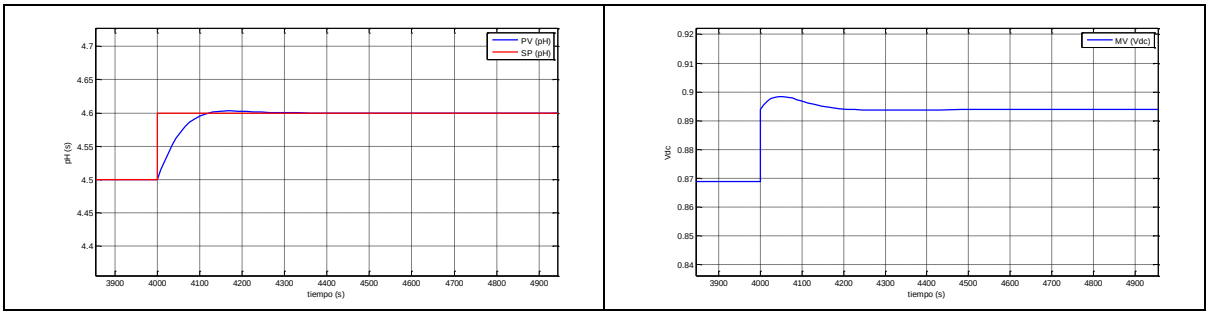


Fig. D.26 Respuesta del sistema ante un cambio de setpoint de pH de 4.6

Tramo 6:

En este tramo correspondiente a valores de la variable manipulable (VM), comprendidos entre 1.8-1.82. Se consideró la siguiente función de transferencia:

$$T(s) = \frac{197}{259s + 1}$$

Los parámetros del controlador son los siguientes:

Kp = 0.022 Ti = 168.3

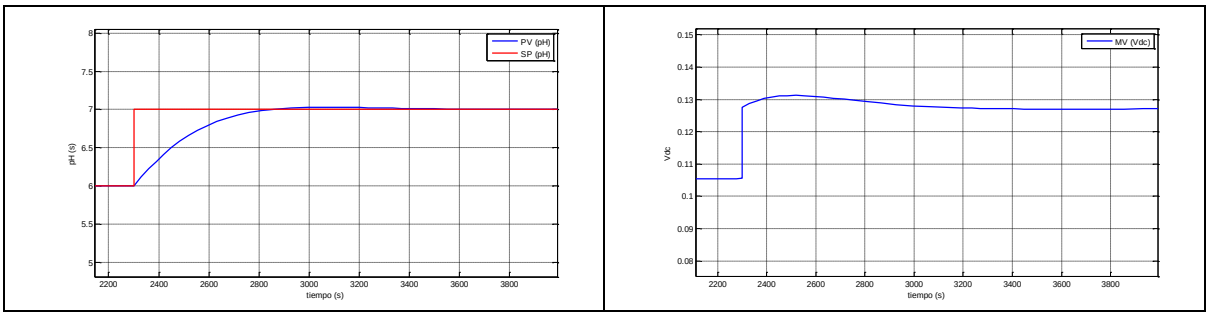


Fig. D.27 Respuesta del sistema ante un cambio de setpoint de pH de 7

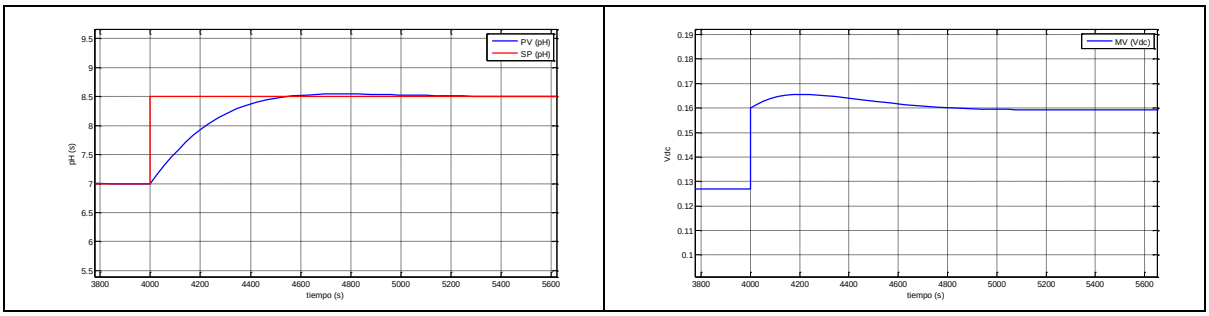


Fig. D.28 Respuesta del sistema ante un cambio de setpoint de pH de 8.5

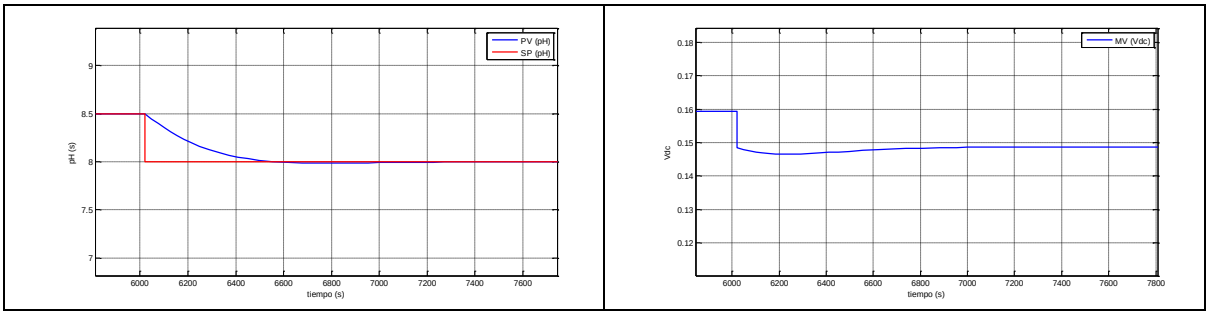


Fig. D.29 Respuesta del sistema ante un cambio de setpoint de pH de 8

Tramo 7:

En este tramo correspondiente a valores de la variable manipulable (VM), comprendidos entre 1.82-1.84. Se consideró la siguiente función de transferencia:

$$T(s) = \frac{47}{65s + 1}$$

Los parámetros del controlador son los siguientes:

Kp = 0.09 Ti = 42.2

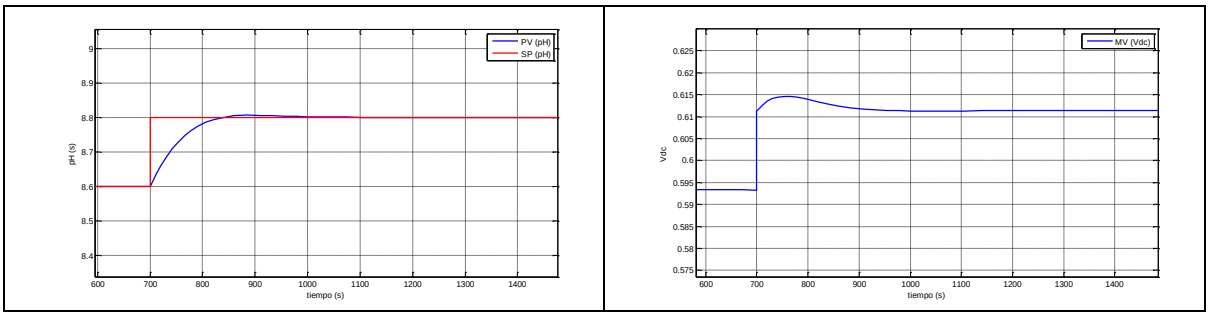


Fig. D.30 Respuesta del sistema ante un cambio de setpoint de pH de 8.8

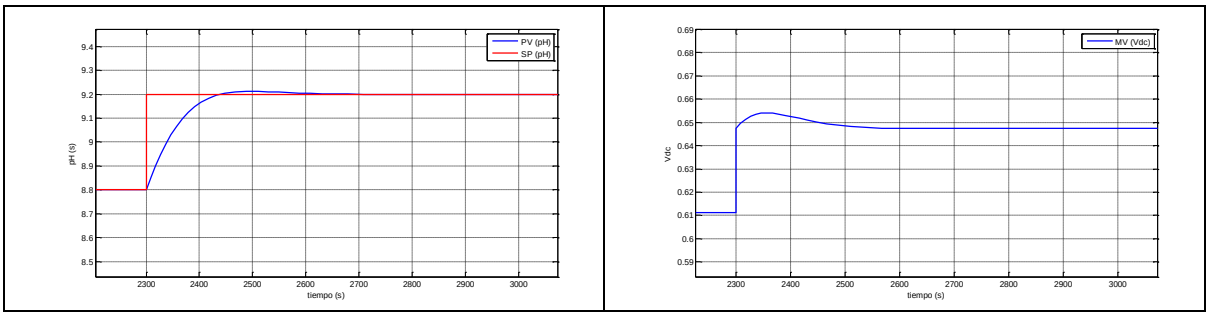


Fig. D.31 Respuesta del sistema ante un cambio de setpoint de pH de 9.2

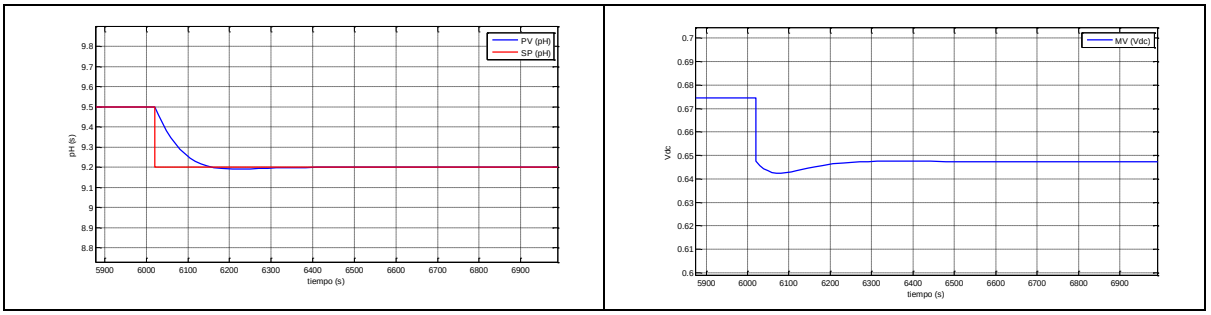


Fig. D.32 Respuesta del sistema ante un cambio de setpoint de pH de 9.2

Tramo 8:

En este tramo correspondiente a valores de la variable manipulable (VM), comprendidos entre 1.84-1.86. Se consideró la siguiente función de transferencia:

$$T(s) = \frac{13.5}{97s + 1}$$

Los parámetros del controlador son los siguientes:

Kp =0.32 Ti = 63

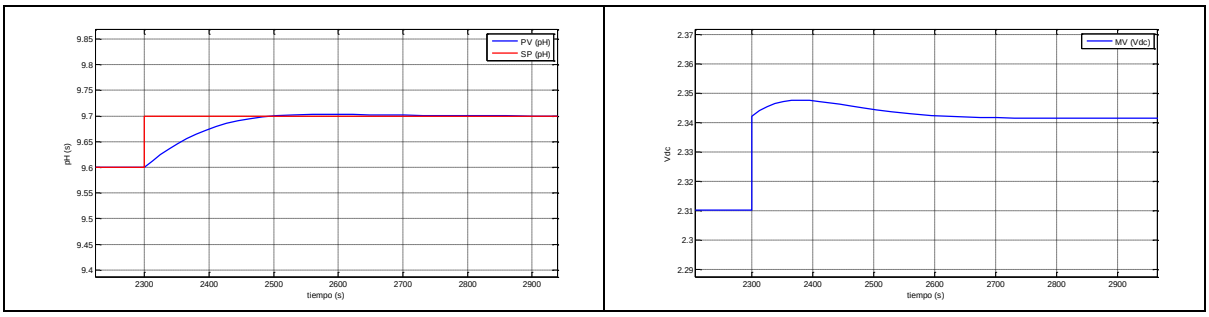


Fig. D.33 Respuesta del sistema ante un cambio de setpoint de pH de 9.7

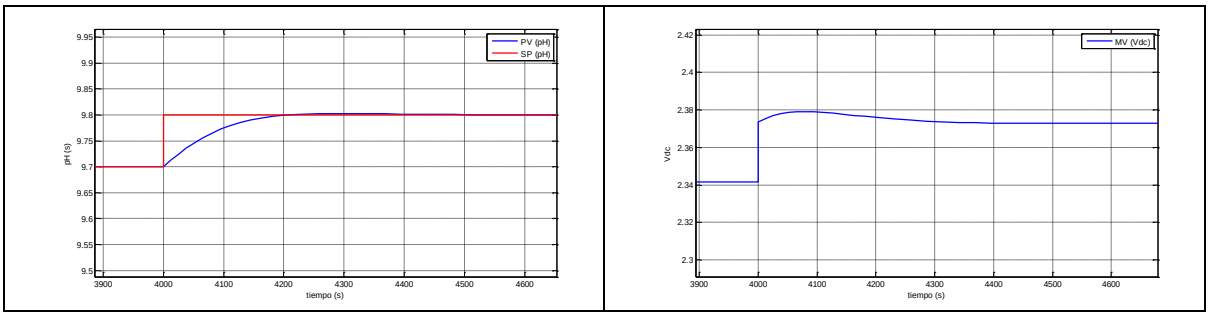


Fig. D.34 Respuesta del sistema ante un cambio de setpoint de pH de 9.8

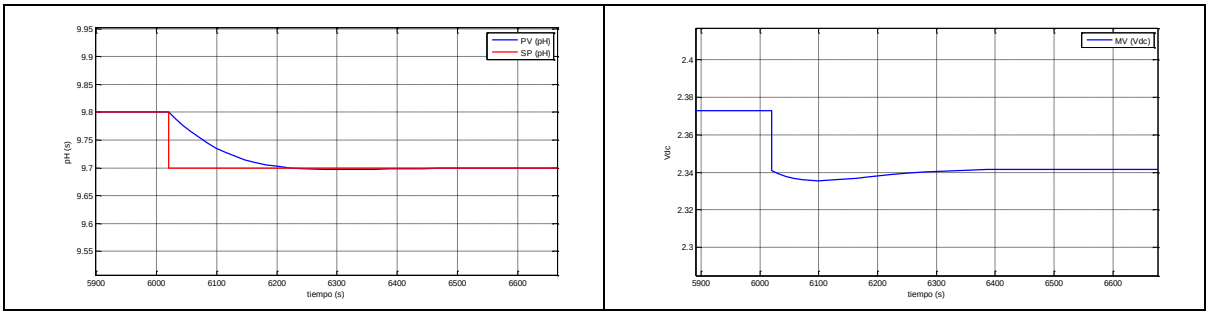


Fig. D.35 Respuesta del sistema ante un cambio de setpoint de pH de 9.7

Tramo 9:

En este tramo correspondiente a valores de la variable manipulable (VM), comprendidos entre 1.86-1.88. Se consideró la siguiente función de transferencia:

$$T(s) = \frac{6.5}{80s + 1}$$

Los parámetros del controlador son los siguientes:

Kp = 0.67 Ti = 52

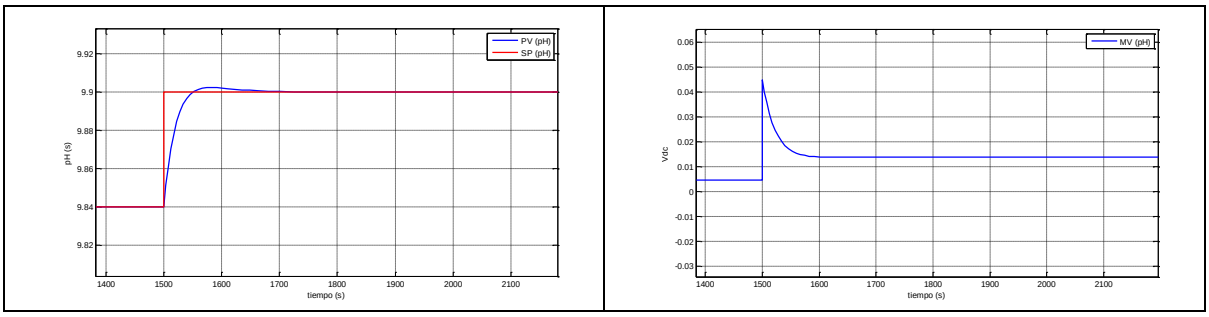


Fig. D.36 Respuesta del sistema ante un cambio de setpoint de pH de 9.9

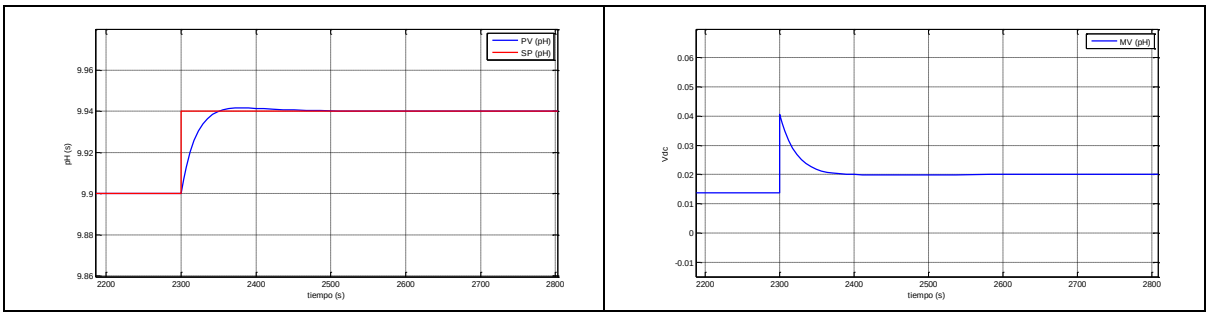


Fig. D.37 Respuesta del sistema ante un cambio de setpoint de pH de 9.94

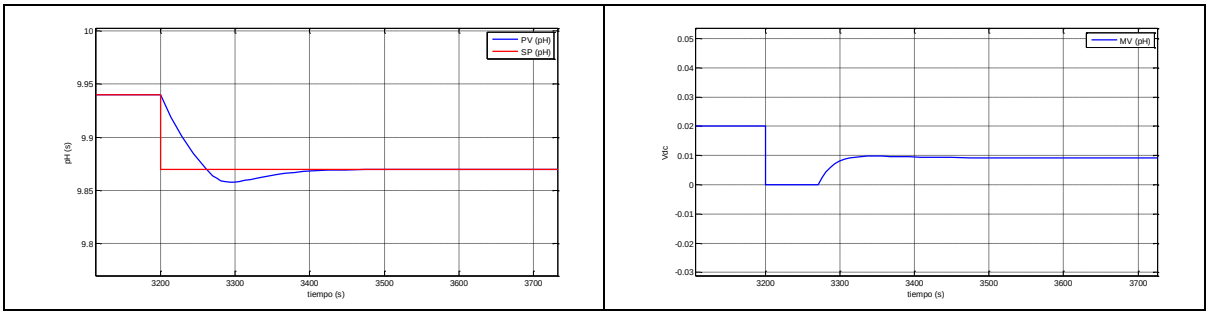


Fig. D.38 Respuesta del sistema ante un cambio de setpoint de pH de 9.87

Tramo 10:

En este tramo correspondiente a valores de la variable manipulable (VM), comprendidos entre 1.88-1.9. Se consideró la siguiente función de transferencia:

$$T(s) = \frac{6.5}{107s + 1}$$

Los parámetros del controlador son los siguientes:

Kp = 0.67 Ti = 69.5

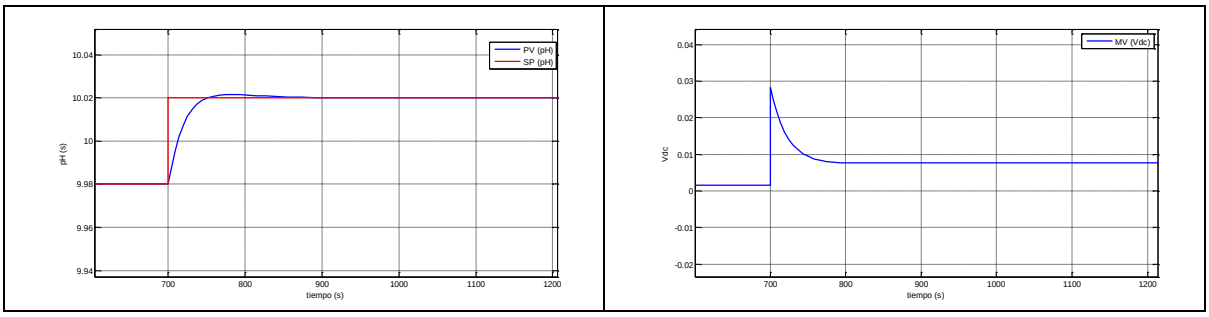


Fig. D.39 Respuesta del sistema ante un cambio de setpoint de pH de 10.02

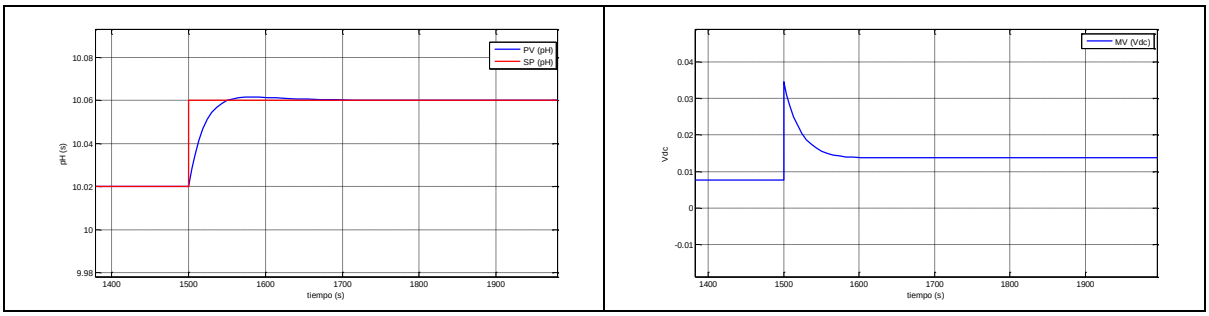


Fig. D.40 Respuesta del sistema ante un cambio de setpoint de pH de 10.06

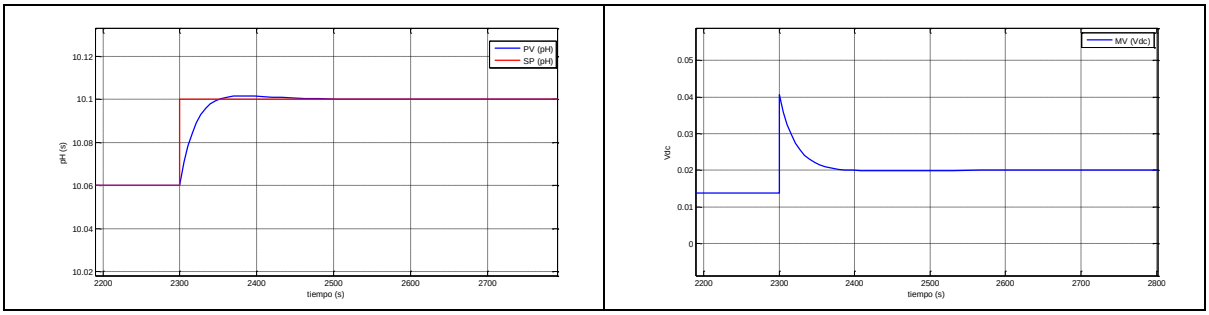


Fig. D.41 Respuesta del sistema ante un cambio de setpoint de pH de 10.1

Se observa en estas pruebas que hay un efecto muy fuerte tanto en no linealidad estática como en la estática. Esto hace que no se pueda representar su ganancia estática utilizando una recta; para este caso se hace necesaria la aproximación de la no linealidad.

Una de las técnicas de control que permiten enfrentar esta problemática ha sido probada en este módulo arrojando buenos resultados. Esto deja entrever que las técnicas de control no lineal se están desarrollando rápidamente con resultados satisfactorios.

La implementación de este tipo de regulador en controladores existentes, así como en PLC's, es factible, siendo una estrategia de control automático para procesos no lineales.

Durante el desarrollo de la tesis en el laboratorio de sistemas automáticos de control, nos permitió la publicación de los siguientes artículos relacionados:

- Control con ganancia programable en proceso no lineal con sistema embebido *FPGA*, Congreso Latinoamericano de Control Automático CLCA-2012.

- Diseño, Construcción y puesta en funcionamiento de tarjeta PCB basada en FPGA aplicada a modulo experimental de laboratorio, INTERCON 2010.

- Estrategia de Control con ganancia variable en Sistemas Embebidos basado en FPGA aplicado a un motor DC, INTERCON 2009.

Así también debido a la actividad de investigación y desarrollo (I+D) en el laboratorio de sistemas automáticos de control, y gracias a las competencias adquiridas, permitieron participar en este otro artículo:

- Desarrollo de Software para control de módulo *WorkStation* de Festo con PLC *Siemens Simatic S7 314C* para aplicaciones industriales, INTERCON 2010, Juan Carlos Soto Bohórquez, Juan Fidel Córdova Rosales, William Ipanaqué Alama, Puno, Perú.

Así también ha permitido la presentación de una solicitud de patente por modelo de utilidad ante INDECOPI:

- Tarjeta Controlador para procesos automáticos con tecnología FPGA.

Referencias

[1] **Amaya, Ferney.** Taller de comunicaciones digitales basadas en FPGAs. (2007). Asamblea XV ISTEC-UTPL, Loja - Ecuador.

[2] **Canto, E; Sutter, G; Deschamps, J.** Guide to FPGA Implementation of Arithmetic Functions. (2012). Editorial Springer Science + Business Media. ISBN 978-94-007-2986-5.

[3] **Dewei Li;Nan Yang; Ran Niu; Hai Qiu; Yugeng Xi**. FPGA based QDMC control for reverse-osmosis water desalination system. (2012). Original Research Article Desalination. Editorial Elsevier. Volume 285, Pages 83–90.

[4] **Furber, Steve.** ARM: System-on-Chip Architecture. (2008). Editorial Pearson Education Limited. ISBN–13:978-0-201-67519-1.

[5] **Garces, Sergio.** Diseño e implementación de un módulo de laboratorio para el control PID de pH. (2008). Tesis de grado. Universidad de Piura, Perú.

[6] **Monmasson, E; Cirstea M.** FPGA design methodology for industrial control systems - a review. (2007). IEEE Transactions on Industrial Electronics, Vol. 54, No. 4, Pages 1824–42.

[7] **Noergaard, T.** Embedded system architecture, A comprehensive guide for engineers and programmers. (2005), Elsevier Inc. ISBN: 0-7506-7792-9.

[8] **O'Dwyer, Aidan.** Handbook of PI and PID Controller Tuning Rules. (2009). Editorial Imperial College Press.

[9] **Pardo, Fernando; Boluda, José.** Lenguaje para Síntesis y Modelado de Circuitos. (1999). Editado por RA-MA Editorial. ISBN: 84-7897-351-6.

[10] **Perez, Carlos; Gracia, Luis; Reinoso, O.** Fusión borrosa de estimadores para aplicaciones de control basado en imagen. (Abril 2010). Revista Iberoamericana de automática e informática industrial. Vol. 7, Num. 2, pp. 81-90.

[11] **Pérez, D.** Sistemas Embebidos y Sistemas Operativos Embebidos. (2009). Universidad Central de Venezuela, Facultad de Ciencias - Escuela de Computación. ISSN 1316-6239.

[12] Pong P. Chu. FPGA Prototyping by VHDL Examples. (2008). Editorial John Wiley & Sons, INC.

[13] Rashid, M. Circuitos microelectrónicos, análisis y diseño. (2000). Editorial International Thomson Editores.

[14] Silage, Dennis. Embedded Design using Programmable Gate Arrays. (2008). Published by Bookstand Publishing. ISBN 978-1-58909-486-4.

[15] Soto, J; Ipanaqué, W. Control con ganancia programable en proceso no lineal con sistema embebido FPGA. (2012). Memoria XV Congreso Latinoamericano de Control Automático CLCA-2012. Pag. 74.

[16] Soto, J; Ipanaqué, W. Diseño, Construcción y puesta en funcionamiento de tarjeta PCB basada en FPGA aplicada a modulo experimental de laboratorio, INTERCON 2010.

[17] Soto, J; Ipanaqué, W. Estrategia de Control con ganancia variable en Sistemas Embebidos basado en FPGA aplicado a un motor DC, INTERCON 2009.

[18] Xilinx. Guia Spartan-3E Starter Kit v1.0 (2006). Xilinx, Inc.

[19] Xilinx. Guia Spartan-3E FPGA Family: Complete Data Sheet. (DS312 March 21, 2005). Xilinx, Inc.

[20] Yi-Wei Tu; Ming-Tzu Ho. Design and implementation of robust visual servoing control of an inverted pendulum with an FPGA-based image co-processor. (October 2011). Original Research Article Mechatronics, Volume 21, Pages 1170-1182.