DISEÑO Y CONSTRUCCIÓN DE UN CONTROLADOR DIGITAL BASADO EN UN MICROCONTROLADOR DE LA FAMILIA 68HC908[1]

Cayetano Antonio Timoteo
Rivera Rodríguez Martín
Gómez Noguera David

Octubre 2009

[1]Tesis dirigida por: M.I. Antonio Salvá Calleja

Esta tesis está dedicada a Ubaldo y Filadelfa, amorosos padres.

AGRADECIMIENTOS

Este trabajo representa la culminación de un proceso que empezó cuando decidí emprender la aventura de salir de mi casa con una maleta llena de esperanzas y sueños, para tratar de rebelarme ante la terrible realidad que las condiciones históricas y los prejuicios sociales habían impuesto sobre mí: un joven mixe de Oaxaca.

Tal vez esto no ha sido como lo soñé, pero no me cabe ninguna duda que todo ha valido la pena. Me gustaría decir *veni, vidi, vici* pero la última palabra se sabrá, como siempre, al final del día.

Durante esta *jamás vista* aventura he tenido siempre el apoyo de mi familia, a la cuál agradezco profundamente. Todo el crédito lo tienen ellos. Así que:

> *Agradezco la mirada de mi padre, la sonrisa de mi madre, el '¿y tu casa?' de lincito y el 'ala' de cory. También, agradezco los insustituibles momentos compartidos con mis dos hermanos y mi cuñado.*

Para validar el funcionamiento del controlador digital construido en este proyecto de tesis fueron necesarias varias pruebas. Dichas pruebas se hicieron en el Laboratorio de Control de la Facultad de Ingeniería de la UNAM, por esa razón, doy las gracias al personal que trabaja en dicho laboratorio.

Deseo expresar un agradecimiento especial al M. en I. Antonio Salvá-Calleja, nuestro director de tesis, por su paciencia y sus palabras de aliento en los momentos complicados.

Por último, agradezco a la UNAM en general, y a la Facultad de Ingeniería en particular por todas las enseñanzas brindadas. A su excelente planta de profesores, en particular a aquellos de cuyas cátedras disfruté en estos años.

ayuuk jaaya'ay adaa tesis jaa'y

Timoteo Cayetano Antonio

Agradezco a mis hermanos por la compañía y el apoyo que me brindan. Se que cuento con ellos siempre.

Agradezco a Dios por llenar mi vida de dicha y bendiciones.

Agradezco haber encontrado el amor y compartir mi existencia con ella.

Agradezco a los amigos por su confianza y lealtad.

Agradezco a mi país porque espera lo mejor de mí.

Agradezco a mis maestros por su disposición y ayuda brindadas.

Martín Rodríguez Rivera

ÍNDICE GENERAL

Capítulo 1

INTRODUCCIÓN

En nuestra vida cotidiana, las personas están en contacto directo con diversos sistemas de control. Aunque la mayoría de las veces no se percaten de ello, diversos mecanismos y dispositivos están siendo controlados o están controlando algo. El cuerpo humano mismo es un complejo sistema de control con una infinidad de sensores donde constantemente se está enviando información en forma de señales eléctricas por parte del cerebro al resto del cuerpo para actuar de una determinada forma u otra.

Un refrigerador, una plancha, el caminar, el andar en bicicleta, son sólo algunos ejemplos de la infinidad de sistemas de control que nos rodean en nuestra vida diaria. Es por eso y por otras razones, que la Ingeniería ha dedicado una de sus ramas a estudiar este tema que a todas luces parece apasionante y maravillosa.

Por otro lado, en el estudio de cualquier disciplina se hace necesario el uso de instrumentos, mecanismos o dispositivos que ayuden a facilitar la labor de análisis o investigación. El uso de estos instrumentos, mecanismos o dispositivos se hace más evidente en cualquier rama de la Ingeniería; por ejemplo, en la Electrónica se usan instrumentos de medición como el osciloscopio, el multímetro y otros dispositivos, como las computadoras, que ayudan a facilitar la labor del estudiante; o que también sirven al diseñador de circuitos electrónicos para simular o verificar sus diseños. Dentro del ámbito del Control, al ser ésta una rama de la Ingeniería, también se necesita esta clase de mecanismos de ayuda. Por ejemplo, algunas veces puede resultar conveniente el uso de un dispositivo que ayude a entender lo que puede estar sucediendo con un diseño de un sistema de control particular. Esto es cierto sobre todo para los principiantes en el estudio del Control, para que puedan sentirse seguros en sus primeras incursiones en esta disciplina.

Es por eso que se ha desarrollado este proyecto de tesis, que consiste en la construcción de un dispositivo para que los principantes en el Control se sientan convencidos de lo que sus diseños están haciendo y, al mismo tiempo, sean atraídos hacia esta importante rama del conocimiento.

Para aclarar la función de este dispositivo, a continuación se muestra el diagrama de

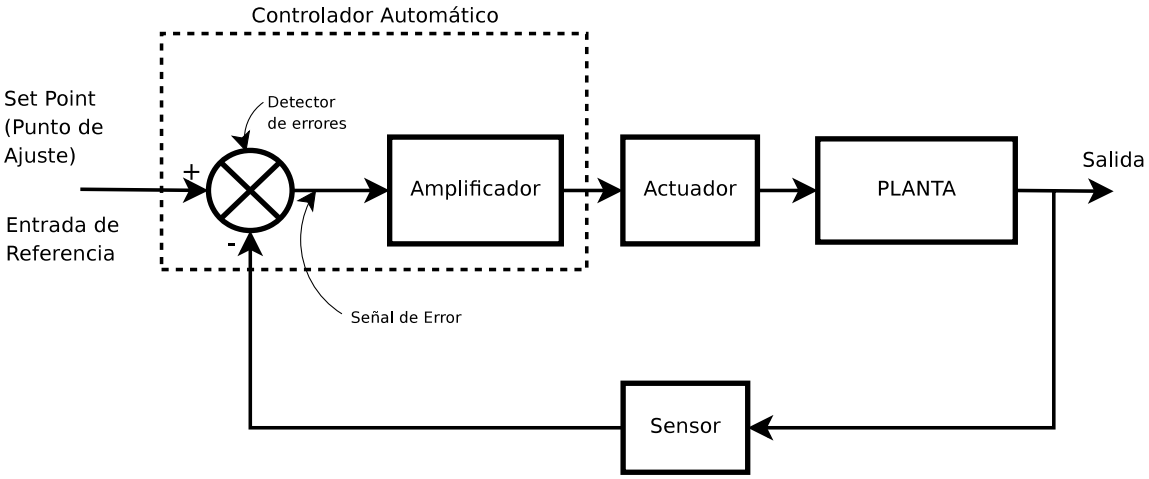


Figura 1.1: Diagrama de bloques básico de un sistema de control.

bloques básico de un sistema de control. Esto se hace en la figura 1.1. Este diagrama se encuentra en cualquier libro de texto de esta disciplina. De dicha figura se resaltan los siguientes bloques:

- Planta
- Controlador
- Actuador
- *Set Point* o Entrada de Referencia
- Salida

De todos los bloques mencionados anteriormente, este proyecto de tesis reproduce y/o simula el conjunto de bloques etiquetado como *controlador automático*. Se observa que la señal de entrada de dicho conjunto es el *set point* y la señal de salida (conocida como señal de control) sirve para excitar al actuador con objeto de reducir el error que se tendrá entre la salida de la planta y el *set point*.

1.1. Objetivo final del proyecto de tesis

El objetivo final de este proyecto de tesis es:

> Diseñar y construir un controlador digital, empleando para ello un microcontrolador de la familia 68HC908, aplicando en el proceso conceptos relacionados

con la teoría de control, así como aspectos tanto de software como de hardware asociados con el microcontrolador empleado.

El uso del dispositivo se resume de la siguiente forma:

1. El usuario hace el diseño de un controlador básico, es decir, encuentra los parámetros del controlador.
2. Conecta el dispositivo a la computadora PC y a la planta.
3. Por medio de la interfaz, ingresa los valores de los parámetros calculados.
4. El dispositivo simula el controlador elegido y cuyos valores se han calculado; y
5. El usuario, después de ver los resultados de la simulación, decide si sus diseño es correcto y a partir de esto, repetir o no la simulación.

En este proyecto de tesis se decidió construir el controlador con componentes electrónicos, debido a que es la rama de la ingeniería donde los autores se pueden desenvolver con mayor libertad. Aunque para realizar esta tarea se pueden usar componentes analógicos, se ha usado el componente digital conocido como microcontrolador debido a su versatilidad y facilidad de uso.

1.2. Organización de la tesis

Para terminar, a continuación se menciona cómo está organizada esta tesis. En los capítulos 2 y 3 se explican todos los conceptos relacionados con la teoría de control y la discretización de controladores, a saber, los controladores básicos, el control digital, el control analógico, la teoría del muestreo, los distintos tipos de discretización, etcétera. En el capítulo 4 se explican los circuitos de adecuación de entrada y de adecuación de salida, los cuales se usaron para el acondicionamiento de señales entre distintos bloques. En el capítulo 5 se desglosa y analiza el programa que debe correr en el microcontrolador. En el capítulo 6 se habla sobre el programa interfaz gráfico de usuario que corre en la PC. En el capítulo 7 se presentan ejemplos de diseño de controladores básicos para una planta consistente en un circuito RC de segundo orden. El capítulo 8 habla sobre las pruebas piloto hechas en el dispositivo y los resultados logrados, contrastando los resultados obtenidos teóricamente en el capítulo 7 con los obtenidos con el dispositivo construido. Las conclusiones se presentan en el capítulo 9. En el apéndice A se muestra el código escrito en lenguaje C en el microcontrolador y el código escrito en lenguaje Java para la interfaz gráfica de usuario. En el apéndice B se incluye el manual de usuario del dispositivo y, por último, en el apéndice C se coloca un glosario con los términos más usados en la teoría de control.

Capítulo 2

TEORÍA BÁSICA ACERCA DE CONTROLADORES *ON-OFF*, P, PI, PD Y PID

2.1. Introducción

En el estudio de los sistemas de control, se hacen distintas clasificaciones de los controladores atendiendo a diversos criterios. Uno de esos criterios es la forma de operar que tienen estos controladores. En este capítulo se usa este criterio para describir las distintas acciones de control y sus principales ventajas y desventajas.

Con objeto de facilitar la explicación y el entendimiento de los conceptos, en este capítulo se usa un lenguaje sencillo con ejemplos tomados de la cotidianidad. Sin embargo, se ha tenido cuidado de no perder la sobriedad y la formalidad que exige un documento de este tipo.

Cabe aclarar que a partir de aquí se usarán términos usados en Sistemas, Señales y Control para explicar varios conceptos. Algunos de los términos usados comúnmente se definen en el apéndice C.

Por último, se menciona que en la sección 2.2, se habla de la clasificación que se hace de los controladores básicos tomando en cuenta sus acciones de control; mientras que en la sección 2.3 se explica el funcionamiento de estos controladores.

2.2. Clasificación de los controladores

Los controladores básicos se clasifican, de acuerdo con sus acciones de control, en:

- De dos posiciones o de encendido y apagado (*ON-OFF.*)
- Proporcional.

- Proporcional Integral.
- Proporcional Derivativo.
- Proporcional Integral Derivativo.

Existen otras formas de clasificarlos, por ejemplo, de acuerdo a la energía que utilizan para su operación se dividen en: neumáticos, hidráulicos o electrónicos. Es importante resaltar aquí que la elección de uno u otro controlador depende de varios factores, a saber: la naturaleza de la planta, la precisión deseada, el peso, el tamaño y otros que no tienen una base estrictamente técnica como el costo de su diseño o de su realización física. Estos últimos, aunque se podría pensar lo contrario, algunas veces pesan más en la decisión de escoger un controlador determinado[1].

Un ejemplo de la elección de un controlador, basándose en cuestiones estrictamente técnicas, sobre una planta que maneja un determinado tipo de gas, podría ser uno neumático, ya que uno electrónico podría resultar peligroso.

2.3. Acciones de control

En esta sección se hará una explicación de cada uno de los controladores que se enumeraron en la sección anterior de acuerdo a sus acciones de control. Se ha considerado que esta tarea se volverá más sencilla si se hace mediante un ejemplo, el cuál se presenta a continuación:

Un patinador está andando en la calle de su colonia (o de cualquier otra). Este patinador tiene un medio de medir la rapidez a la que anda, es decir, una suerte de velocímetro que le indica a qué velocidad va en la patineta. Se ha dado cuenta de que se le está haciendo tarde para llegar al lugar a donde va, y calcula que si va a una velocidad de 10 kilómetros por hora llegará justo a tiempo. Es entonces que decide "apretar el paso", decide impulsarse sobre el suelo con uno de sus pies, para alcanzar la velocidad deseada, y llegar a tiempo a su importante cita.

Obsérvese que este escenario —aunque bastante improbable— es otra muestra de un sistema de control. A manera de puntualización, a continuación se identifica cada bloque del diagrama de bloques general de un sistema de control de lazo cerrado de la figura 1.1, presentado en el capítulo 1: El *set point*, o valor de referencia, es la velocidad que el patinador desea alcanzar, en este caso, 10 kilómetros por hora; el sensor, es el velocímetro ficticio que tiene en la mano, el cuál le indica qué velocidad lleva en el instante en qué la mira; el restador es obviamente su cerebro que hace la operación matemática de la sustracción entre la velocidad deseada y la velocidad mostrada por el velocímetro, hallando

[1]Como en la mayoría de las obras de ingeniería, el factor costo, desempeña un papel importante en la toma de decisiones de un proyecto determinado.

con esto el error; el actuador es su pie impulsándose sobre el suelo; por último, la planta es el mismo patinador desplazándose.

Cada vez que el patinador mira el velocímetro y se da cuenta de que no lleva los 10 kilómetros por hora que debería llevar, debe tomar una decisión sobre ese dato e impulsarse con más o menos fuerza para alcanzar la velocidad deseada. A esa decisión (y a la acción que resulta de esa decisión) es a lo que se conoce como *acción de control*.

En las siguientes subsecciones se analizan y estudian las acciones básicas de control, sus ventajas e inconvenientes para su realización y sus modelos matemáticos. El ejemplo del patinador ayudará a explicar y entender mejor sobre cómo se comportan estas acciones de control con un lenguaje claro y sencillo; sin dejar de lado las ecuaciones que caracterizan a estas acciones de control.

2.3.1. Acción de control de dos posiciones o de encendido y apagado (*ON-OFF*)

En un sistema de control de dos posiciones, como su nombre lo indica, el controlador solamente puede tomar cualquiera de dos estados en un momento determinado, ningún otro. El estado en el que se encuentre será determinado por el error o la diferencia entre el valor deseado (*set point*) y el valor actual de la salida.

El ejemplo del patinador no puede aplicarse aquí debido a que, en lugar de simplificar la explicación, la complicaría, por tanto, para esta acción de control se usará el ejemplo de un aparato doméstico muy conocido y que usa esta misma acción de control para realizar su trabajo: el refrigerador[2].

El refrigerador es el modelo de control *ON-OFF* por excelencia. Se sabe que este aparato produce un ruido cuando está trabajando y que en algunos casos puede llegar a ser molesto (dependiendo del modelo y la antigüedad). Este ruido no es más que el provocado por el motor que comprime el refrigerante. La unidad básica de control del refrigerador es un termostato[3]. Este termostato se abre o se cierra dependiendo de la temperatura requerida en comparación con la temperatura que se tiene en el refrigerador. El usuario fijará la temperatura deseada y el termostato se abrirá o se cerrará dependiendo de si se ha alcanzado o rebasado dicha temperatura.

Entonces, en palabras llanas y simples, el funcionamiento del sistema de control del refrigerador es: se enfría hasta alcanzar la temperatura deseada, una vez que esto se logra, se apaga el motor. Debido al ambiente, la temperatura del sistema se eleva rebasando la temperatura deseada; una vez que esto sucede, el termostato cierra el circuito y el motor nuevamente comienza a funcionar bajando la temperatura del sistema. Al alcanzarse la temperatura deseada, se abre el termostato, apagándose el motor. Este ciclo se repite

[2]En realidad, el refrigerador usa la acción de control *ON-OFF* con histéresis. Aquí se simplifica la explicación considerando la acción de control *ON-OFF* simple.

[3]Un termostato es un componente de un sistema de control que se abre o se cierra en función de la temperatura.

continuamente.

Extrapolando el funcionamiento del refrigerador a una acción de control *ON-OFF* en general, éste toma el error como variable de entrada y verifica si este último es mayor o menor que cero. Dependiendo del resultado de esta comparación, la salida del controlador estará en una posición o en otra (será encendida o apagada).

Si se considera que la señal de salida del controlador es $u(t)$ y que la señal de error es $e(t)$; el funcionamiento del control de dos posiciones o control *ON-FF* se puede expresar de la siguiente manera:

$$u(t) = \begin{cases} U_1 & \text{para } e(t) > 0 \\ U_2 & \text{para } e(t) < 0 \end{cases} \tag{2.1}$$

en donde U_1 y U_2 son constantes. Por lo general, el valor mínimo de U_2 es cero o $-U_1$.

Obsérvese que la realización física de este controlador, aunque posible, no es la más conveniente, porque, según el modelo matemático de este controlador dado por la ecuación 2.1, la salida siempre estará en un estado o en otro dependiendo del error; y en sistemas en donde la constante de tiempo sea muy pequeña, es decir, donde la velocidad de respuesta sea rápida, el actuador estará encendiéndose o apagándose con una frecuencia muy alta, lo que conllevaría a una destrucción rápida de dicho actuador o su tiempo de vida útil se vería drásticamente reducida[4]. Es por eso que se acostumbra poner una *ventana* o *histéresis*.

La histéresis es el espacio o zona muerta alrededor del error nulo dentro de la cuál no ocurrirá ningún cambio de estado por parte del controlador; es decir, si el error $e(t)$ está dentro de la ventana o *histéresis*, la acción de control no variará. La inclusión de la histéresis permite que el controlador *ON-OFF* no conmute indiscriminadamente ante pequeñas perturbaciones.

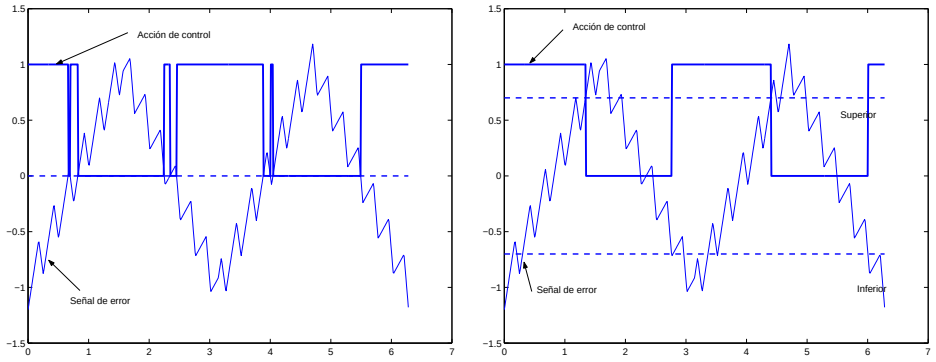
En la figura 2.1 se muestra una comparación entre la salida de dos controladores *ON-OFF* sin y con histéresis para una entrada de error ficticia que cambia con brusquedad. Se puede observar que cuando el controlador no tiene una histéresis, los cambios de estado en la salida se presentan con una frecuencia alta, siendo esto muy evidente en 2.1a. En la figura 2.1b se muestra la acción del controlador *ON-OFF* con histéresis sobre la misma señal de error, donde los límites de la histéresis están señaladas con líneas discontinuas. Se puede observar que los cambios de estado no son tan frecuentes en comparación con el mismo controlador sin histéresis.

La figura 2.2 muestra el diagrama de bloques para un controlador *ON-OFF* sin y con histéresis.

Como un dato al margen, algunos autores denominan a la *histéresis* o *ventana* como *brecha diferencial*. Dicha designación no es muy popular, por lo que en este texto no se usa.

Por lo visto hasta ahora, se podría pensar que el diseño de este tipo de controladores es muy sencillo. Sin embargo, eso no es del todo cierto. El diseño y posterior construcción de un controlador *ON-OFF* se puede complicar debido a la falta de un método o algoritmo

[4]Piénsese, por ejemplo, en un relevador que se esté abriendo y cerrando constantemente, las laminillas se gastarían muy rápido



(a) (b)

Figura 2.1: Acción de un controlador *ON-OFF* sin y con histéresis.

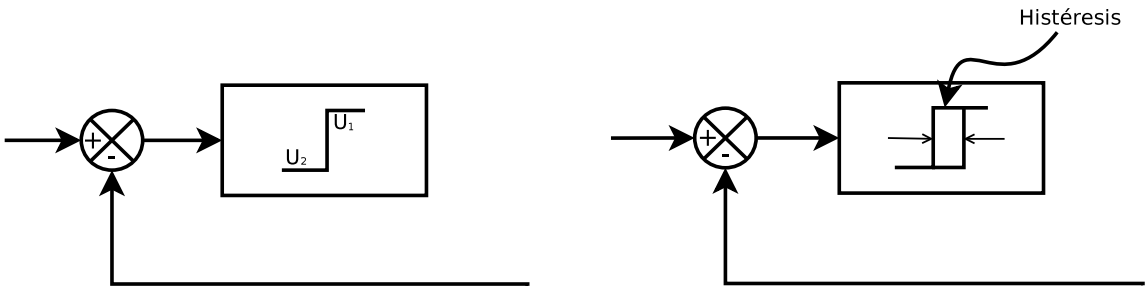


Figura 2.2: Diagrama de bloques de un controlador ON-OFF sin y con histéresis.

para determinar el tamaño de la histéresis para un proceso determinado; por lo que el diseñador deberá tener especial cuidado en este aspecto.

2.3.2. Acción de control Proporcional

El término proporcional se refiere a la relación entre la variable de control y el error; dicho con otras palabras, la salida del controlador será proporcional a la señal de error.

Para una mejor explicación de este concepto se tomará el ejemplo del patinador planteado al principio de esta sección. Piénsese, junto a todo el escenario planteado anteriormente, que el patinador se impulsará de acuerdo a la velocidad deseada y a la velocidad que mira en su velocímetro por medio de un control proporcional; por lo que, éste mirará el velocímetro y se impulsará tan rápido o tan despacio como la diferencia entre la velocidad deseada y su velocidad actual, es decir, al error.

Lo que sucede es lo siguiente:

Al principio, cuando el patinador está en reposo, su velocidad es de cero; la diferencia entre la velocidad deseada (10 kilómetros por hora) y su velocidad actual es de 10 kilómetros por hora, entonces el patinador se impulsará con todas sus fuerzas para tratar de reducir este error a cero. Después de haber pasado cierto tiempo, el patinador mira su velocímetro y se da cuenta de que ha conseguido desplazarse a 5 kilómetros por hora. Hace cálculos mentalmente y concluye que el nuevo error es de 5 kilómetros por hora; es decir, el error se ha reducido a la mitad; entonces, de acuerdo a lo establecido, el patinador ya no usará la misma fuerza para impulsarse; para este instante, usará la mitad de fuerza usada para el primer caso. Si continúa así, llegará un momento en que el error se habrá reducido a la tercera parte del original, instante en el cual ya no se impulsará con la misma fuerza que para el caso anterior sino con la tercera parte de la fuerza original.

En resumen, la fuerza utilizada por el patinador para impulsarse es directamente proporcional a la diferencia entre la velocidad deseada y la velocidad reportada en su velocímetro.

Es importante aclarar que la constante de proporcionalidad, conocida como K_C, es la misma durante todo este proceso.

El escenario anterior ejemplifica de manera clara lo que sucede con un sistema en donde el controlador es de tipo proporcional. Para una mayor aclaración, en la figura 2.3 se muestra una gráfica que contiene varias respuestas a escalón unitario de un sistema de control de lazo cerrado, en donde el controlador es el explicado en este apartado. Estas respuestas son para un planta ficticia de primer orden. En la figura 2.3 se observa que el error nunca logra llegar a cero por más que se aumente la ganancia o la constante de proporcionalidad K_C. Esto es así debido a que la acción de control depende exclusivamente del error, es decir, a mayor error la acción de control es mayor; conforme se va reduciendo el error, la acción de control va reduciéndose en la misma proporción. Si el error llega a cero, la acción de control desaparece. Esta es una de las grandes desventajas de este tipo de controladores. Otra forma de entender lo anterior es que el controlador proporcional no puede autodestruirse y por esa razón no puede "permitir" que el error sea cero.

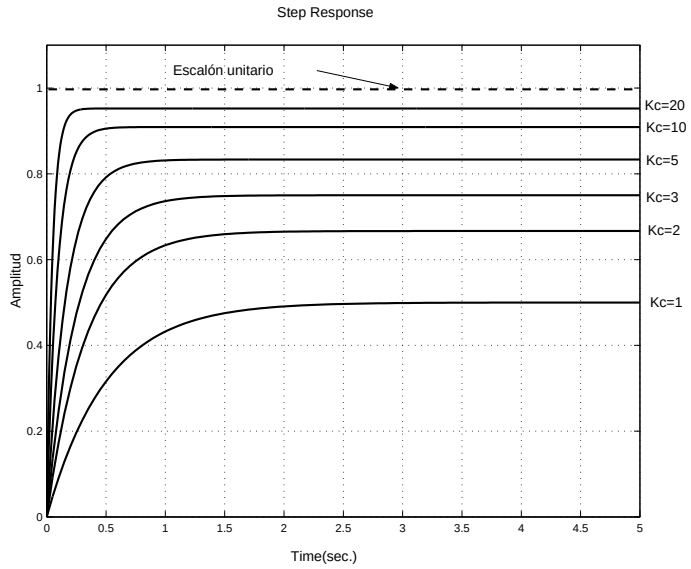


Figura 2.3: Respuesta a escalón unitario de un sistema de primer orden con un control proporcional y una K_C dada.

De acuerdo a la figura 2.3, es evidente que entre mayor es el valor de K_C para un sistema dado, la velocidad de respuesta aumenta. Otra conclusión que se puede obtener de las gráficas de la figura 2.3 es que conforme se aumenta la ganancia K_C el error en estado estacionario disminuye; sin embargo, esta conclusión solamente aplica para una planta de primer orden. Si el orden de una planta es de 3, por ejemplo, las respuestas a escalón unitario con distintos valores de ganancia se muestran en la figura 2.4. En dicha figura se observa que el sistema se vuelve inestable si la ganancia K_C rebasa un cierto límite.

Por todo lo anterior se puede concluir que aunque un aumento en la ganancia proporcional puede reducir el error en estado estacionario, ésta también puede tener consecuencias indeseables como la inestabilidad total del sistema.

A continuación se ponen los conceptos anteriores en términos algebraicos:

Sea $u(t)$ la acción de control, $e(t)$ el error y K_C la *Ganancia Proporcional*; la acción de control proporcional viene dado por la siguiente ecuación:

$$u(t) = K_C e(t) \tag{2.2}$$

o en términos de la transformada de Laplace

$$\frac{U(s)}{E(s)} = K_C \tag{2.3}$$

Algunos autores consideran al controlador proporcional como un amplificador con ganancia ajustable.

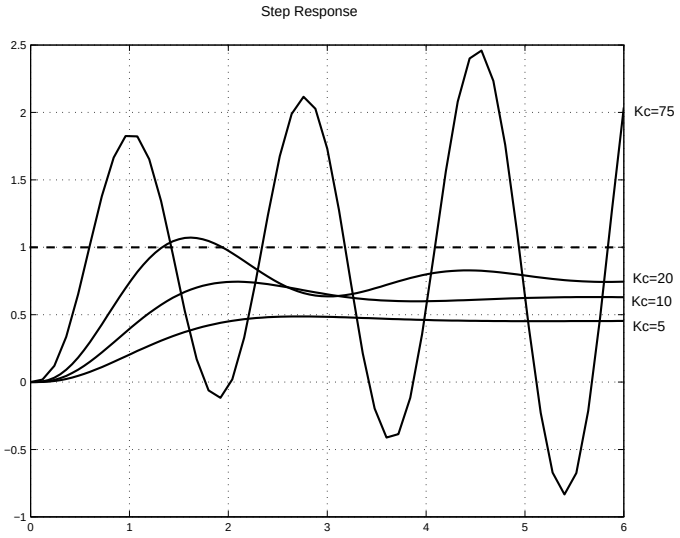


Figura 2.4: Respuestas para un sistema de tercer orden con diferentes valores de K_C. La función de transferencia de la planta simulada es $\frac{1}{s^3+6s^2+11s+6}$.

A manera de conclusión, a continuación se muestra una lista de las principales características del controlador proporcional:

1. Reduce el error sin llegar a eliminarlo por completo.
2. No toma en cuenta como cambia el error con el tiempo.
3. Se puede aumentar la velocidad del sistema, aumentando el valor de K_C.
4. Se puede disminuir el error en estado estable aumentando K_C.
5. Si K_C se aumenta demasiado en algunos sistemas, estos se pueden volver inestables.
6. Su construcción física es sencilla y económica, debido a la pequeña cantidad de componentes que se necesitan.

En la figura 2.5 se muestra el diagrama de bloques de un controlador proporcional.

2.3.3. Acción de control Proporcional Integral

Siguiendo con el ejemplo del patinador; obsérvese que si éste continúa utilizando la acción de control Proporcional nunca logrará mantener la velocidad deseada (10 kilómetros por hora) tal como se explicó en la subsección anterior; sin embargo, si se hace una modificación a dicha acción de control se puede lograr el objetivo de lograr llegar y mantener la velocidad deseada.

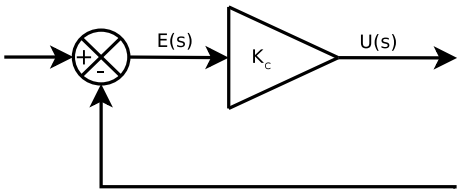


Figura 2.5: Diagrama de bloques de un controlador proporcional.

Dentro de las acciones de control mencionadas en la sección anterior, está la acción de control Proporcional Integral. Para esta acción de control, en lugar de que la señal de control sea solamente proporcional al error, esta última es también integrada. En el ejemplo del patinador, si éste usara la acción de control Proporcional Integral, se impulsaría de acuerdo a la integral del error y de manera proporcional a este último. Se hace énfasis en que la señal de error será la diferencia entre la velocidad deseada y la velocidad registrada en su velocímetro.

Debido a que la operación de integración de una señal no es tan evidente o sencilla de comprender como en el caso de la multiplicación por una constante (acción de control proporcional) mencionado anteriormente, en el siguiente apartado se explica como actúa solamente la parte integral del controlador. Inmediatamente después se explicarán las características de la acción de control integral y proporcional juntos.

La acción de control integral

En la figura 2.6, se pueden observar las gráficas de un sistema ficticio de primer orden. En dicha figura se observan simultáneamente las gráficas de la respuesta a escalón unitario del sistema ficticio, el error de esa respuesta y la integral del error. Se observa que la salida se estabiliza por un momento en 0.5, luego comienza a subir para después estabilizarse definitivamente en 1. En la última gráfica se muestra la integral de la señal de error. En esta gráfica se observa que la integral del error nunca se vuelve cero aún cuando el error mismo sí lo haga. Cuando el error se vuelve constante en 0.5, se observa que su integral es una rampa, es decir, cuanto más tiempo permanezca constante el error, su integral aumentará continuamente de forma proporcional. En términos de control, se dice que el controlador integral no cesa su acción cuando el error permanece contante.

Matemáticamente, en una acción de control integral la salida del controlador es expresada como sigue:

$$u(t) = K_i \int e(t)dt \tag{2.4}$$

donde K_i es una constante ajustable. Despejando $K_i e(t)$, se obtiene:

$$\frac{du(t)}{dt} = K_i e(t) \tag{2.5}$$

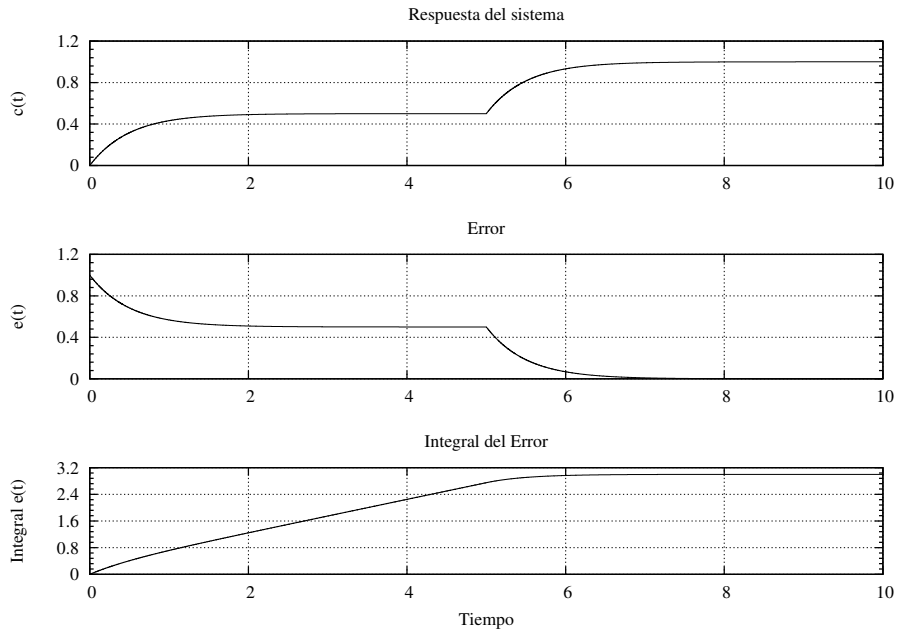


Figura 2.6: Funcionamiento de la acción integral.

De la ecuación 2.5 se deduce otra interpretación de la acción de control integral: *"La razón de cambio de la acción de control $u(t)$ con respecto del tiempo es proporcional al error $e(t)$"*. La explicación de esta ecuación es la siguiente: Si el error $e(t)$ se duplica, el valor de la salida $u(t)$ (la señal de control) variará dos veces más rápido; es decir, *la señal de control variará más rápido o más lento dependiendo de la magnitud del error*. Por lo anterior, se puede deducir que la acción de control integral aumenta el sobrepaso en la respuesta en el tiempo del sistema.

La función de transferencia del controlador Integral, debido a la ecuación 2.4, queda expresada como:

$$\frac{U(s)}{E(s)} = \frac{K_i}{s} \tag{2.6}$$

Con lo explicado hasta aquí y con la ayuda de la figura 2.6, a continuación se muestra una lista de las características de la acción de control integral:

1. La acción de control no desaparece aún cuando el error si lo haga, debido a que no actúa sobre el error en sí, sino sobre la integral de éste hasta un momento dado.

2. Cuando el error se vuelve constante, la acción de control no lo hace, ésta va incrementando su valor de forma proporcional.

3. La respuesta en el tiempo de un sistema sometido a un control integral, sufre un aumento en el sobrepaso.

Después de explicar la acción del controlador integral, a continuación se verá y analizará lo que sucede cuando la acción de control proporcional y la integral actúan simultáneamente para formar lo que se conoce como *Controlador Proporcional Integral*.

Como se mencionó en la subsección anterior, el principal defecto del control proporcional es su incapacidad para eliminar el error en estado estable. De igual manera, se ha mencionado que el control integral va incrementando su valor de forma proporcional sobre un error constante.

Cuando ambas acciones de control actúan de manera simultánea, se combinan las características anteriormente mencionadas. En efecto, en un controlador Proporcional Integral, el error en estado estable desaparece debido a la acción integral. Sin embargo, junto a esto, aparece lo que se podría denominar un *daño colateral*: el sobrepaso en la respuesta temporal aumenta.

Matemáticamente, el controlador Proporcional Integral se define como:

$$u(t) = K_C e(t) + \frac{K_C}{T_i} \int_0^t e(t)dt \tag{2.7}$$

o por medio de su función de transferencia:

$$\frac{U(s)}{E(s)} = K_C \left(1 + \frac{1}{T_i s}\right) \tag{2.8}$$

donde K_C es la ganancia proporcional y T_i se denomina *tiempo integral*. Ambas constantes son ajustables, por lo que la tarea del diseñador consiste en encontrar los valores apropiados de estas constantes.

Como se aprecia claramente en la ecuación 2.8, K_C afecta tanto a la parte proporcional como a la integral, mientras que T_i afecta solamente a la parte integral. Es necesario agregar que el inverso de la contante T_i es conocido como *velocidad de reajuste* y mide la cantidad de veces por unidad de tiempo que se duplica la parte proporcional de la acción de control total (Proporcional Integral).

A continuación se muestra una lista de las características del controlador Proporcional Integral:

1. El error en estado estable es cero. Lo anterior se cumple para sistemas con referencia escalón y función de transferencia de lazo abierto de tipo cero.

2. El sobrepaso en la respuesta en el tiempo aumenta.

En la figura 2.7 se observa el diagrama de bloques de un controlador proporcional integral.

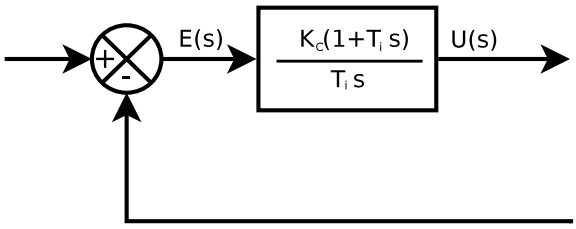


Figura 2.7: Diagrama de bloques de un controlador PI.

2.3.4. Acción de control Proporcional Derivativa

La siguiente acción de control que se estudia en este capítulo es conocida como Proporcional Derivativa. Esta acción de control es similar al estudiado anteriormente, con la diferencia de que en lugar de integrar la señal de error, ésta se deriva. Para analizar esta acción de control, se procederá de la misma forma a como se hizo en la acción de control Proporcional Integral; es decir, primero se analizará la acción derivativa actuando sola, después, se estudiará la acción de control Proporcional en conjunción con la acción de control Derivativa.

La Derivada

Antes de pasar a definir y explicar como actúa la acción derivativa, conviene definir y analizar primero lo que es la derivada.

La derivada es un operador unario que al aplicarse sobre una función real de variable real, comúnmente tiene la interpretación de la pendiente de la recta tangente al punto que se esté analizando, de ahí que uno de los usos comunes que se da a la derivada es la de hallar máximos y mínimos de una función real de variable real, sólo encontrando el punto donde la derivada sea igual a cero. La interpretación anterior, aunque correcta, es un poco simplista y no puede usarse para aclarar el funcionamiento de la acción de control derivativa.

La otra interpretación, se podría decir que es la segunda más popular, es que la derivada representa la razón de cambio de la variable dependiente con respecto de la variable independiente. La variable independiente puede ser de cualquier tipo, sin embargo, si ésta es el tiempo, la interpretación se hace aún más intuitiva, debido al contacto permanente que se tiene con este concepto y los evidentes cambios que se experimentan conforme transcurre ésta. Es así que a nadie se le hace raro, ver cambios en la estatura de algún niño, cambios en hábitos de vida, etcétera. Lo anterior, si se pudiera medir de alguna forma, se podría poner en términos de razones de cambio con respecto del tiempo.

En la figura 2.8 se presenta una gráfica que muestra la derivada de una función real de variable real en el punto x_0, con variable independiente denotada por x. Conforme el incremento, denotado por Δx, se vuelve más pequeño, la fracción $\frac{f(x_0+\Delta x)-f(x_0)}{\Delta x}$ se

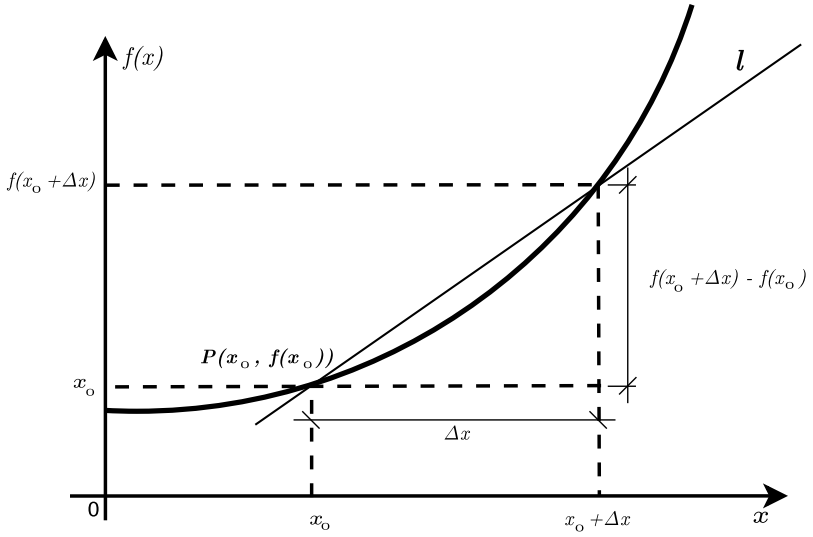


Figura 2.8: La derivada como la pendiente de la recta tangente en el punto x_0.

aproxima a la derivada, es decir, a la pendiente de la curva $f(x)$ en el punto x_0.

Ahora bien, conforme a lo definido,

$$\lim_{\Delta x \to 0} \frac{df(x)}{dx} = \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x} = \frac{\Delta f(x)}{\Delta x}$$

donde $\frac{df(x)}{dx}$ es la derivada de $f(x)$.

Ahora que ya se ha definido la derivada, a continuación se a definirá la acción derivativa.

Acción Derivativa

Para la explicación de esta acción de control, se considera un sistema de segundo orden, al cuál se le aplica una señal escalón unitario y presenta la salida mostrada en la figura 2.9. La salida de ese sistema se muestra en la primera gráfica, el error en la segunda gráfica y la derivada del error en la tercera gráfica de dicha figura.

Puede observarse que mientras la respuesta del sistema va creciendo, la derivada del error toma signo negativo; esto generará una acción de control opuesta al crecimiento de la señal de salida, es decir, se genera una señal de *frenado*. Después de que la salida ha alcanzado su valor máximo, comienza a descender; entonces la derivada del error toma signo positivo. En ambos casos, es posible darse cuenta de que la derivada del error se *opone* al crecimiento o decrecimiento de la señal de salida.

Por lo mencionado anteriormente, se puede deducir que en un control derivativo, el sobrepaso de la respuesta del sistema, será menor comparándolo con la respuesta a ese

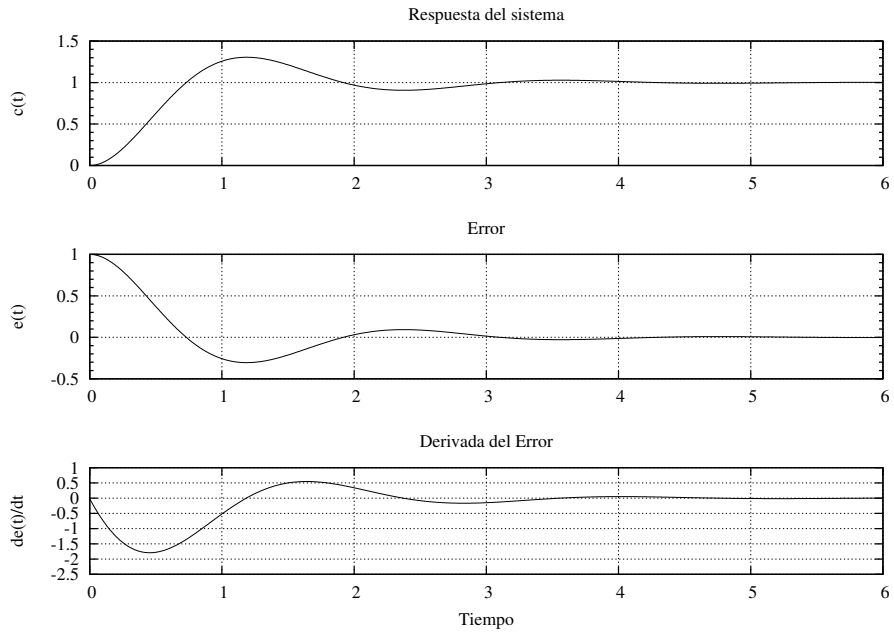


Figura 2.9: Acción de control derivativa.

mismo sistema sin control derivativo, debido al *frenado*. También se puede decir que el control derivativo (conforme a lo establecido en la subsección de derivada) responde a los cambios en la respuesta del sistema; es decir, solamente afecta a las variaciones de la respuesta, mientras que en una respuesta constante no tiene ningún efecto aún cuando está presente. Lo anterior significa que el control derivativo no afecta al error en estado estable.

Ahora que ya se conocen las características del control derivativo, a continuación se analizará el conjunto Proporcional Derivativo.

La acción de control de un controlador Proporcional Derivativo (PD) se define mediante la siguiente ecuación:

$$u(t) = K_C e(t) + K_C T_d \frac{de(t)}{dt} \tag{2.9}$$

Al analizar la ecuación 2.9, se puede ver que otra forma de escribir ésta es:

$$u(t) = K_C \left(e(t) + T_d \frac{de(t)}{dt} \right) \tag{2.10}$$

Partiendo de la ecuación 2.10 se puede trazar la gráfica 2.10. En esta gráfica se puede interpretar a la acción derivativa como una extrapolación del error T_d unidades de tiempo

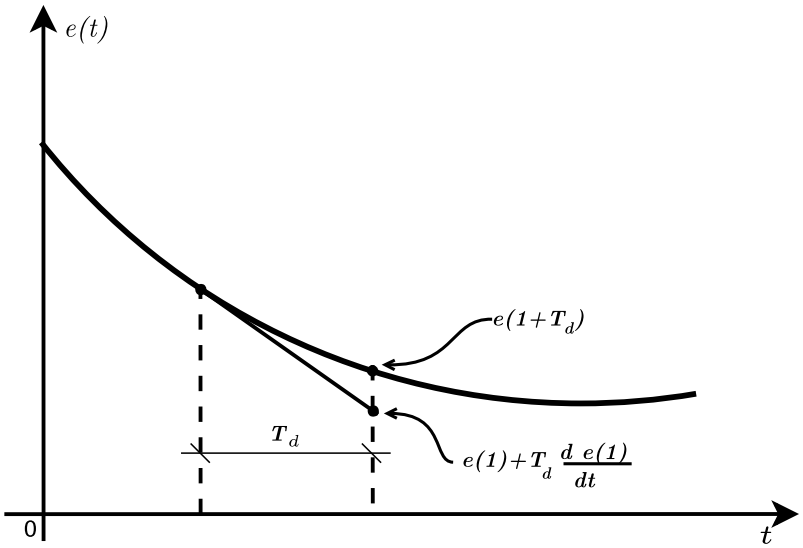


Figura 2.10: Aproximación del control proporcional derivativo.

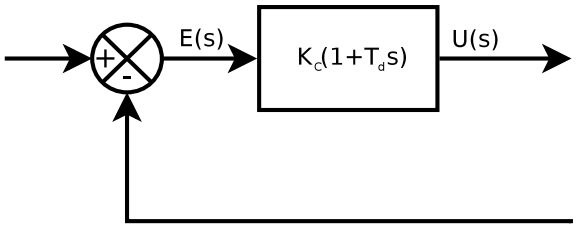


Figura 2.11: Diagrama de bloques del controlador PD.

hacia adelante. Dicho de otra manera, el control Proporcional Derivativo se *anticipa* al error para hacer la acción de control. Una visión alternativa de la acción de control en cuestión es que un controlador proporcional derivativo *predice* el error y a partir de dicha *predicción* actúa.

La función de transferencia de un control proporcional derivativo está dado por la ecuación 2.11. El diagrama de bloques se muestra en la figura 2.11.

$$\frac{U(s)}{E(s)} = K_C(1 + T_d s) \tag{2.11}$$

Al observar la ecuación 2.11 se puede inferir que un controlador Proporcional Derivativo introduce un cero en la función de transferencia de trayectoria directa. Por [12] se sabe que la introducción de un cero a una función de transferencia de lazo abierto tiene el efecto de *jalar* el lugar geométrico de las raíces hacia la izquierda.

Por otro lado, si se supone una diferenciación ideal, la función de transferencia es impropia (el grado del polinomio del numerador es mayor que el grado del polinomio en el denominador), por lo tanto, hay restricciones prácticas que prohíben una realización exacta del controlador PD ideal. La más evidente es que el ancho de banda es infinito. Además, en muchas aplicaciones prácticas, el *set point* es constante a tramos; esto significa que la derivada del *set point* será cero excepto para aquellos instantes de tiempo cuando éste cambie, donde la derivada será infinita. De igual manera, la extrapolación lineal no es exacta cuando la señal medida cambia más rápidamente comparada con el valor de T_d (véase figura 2.10).

Atendiendo a todo lo anterior, a continuación se muestra la función de transferencia del controlador Proporcional Derivativo práctico:

$$G_C(s) = K_C \left(1 + \frac{T_d s}{1 + \frac{T_d}{N} s}\right) \tag{2.12}$$

Como se observa en la ecuación 2.12, la diferencia esencial está en el filtro que se le agrega a la parte derivativa. La frecuencia de corte de este filtro es N/T_d, donde N es conocido como *ganancia derivativa máxima*. El objetivo del filtro es dejar pasar únicamente las frecuencias menores a N/T_d.

Las principales características del control proporcional derivativo son:

1. Se mejora el amortiguamiento de la respuesta del sistema, es decir, el sobrepaso disminuye.

2. No afecta al error en estado estable.

3. Nunca se aplica sola, debido a que solo actúa en estado transitorio.

Regresando al ejemplo del patinador, para que éste pueda controlar la velocidad deseada por medio de un control Proporcional Derivativo, necesita conocer *a priori* el error para saber la fuerza necesaria que debe aplicar y poder conservar la velocidad deseada. Lo anterior es a todas luces imposible en la realidad.

2.3.5. Acción de control Proporcional Integral Derivativa

En esta sección se estudiará el controlador más usado en la industria: la combinación de los controladores Proporcional, Integral y Derivativo.

Este controlador reúne las ventajas y los inconvenientes de todos los controladores estudiados anteriormente (con excepción del controlador *ON-OFF*).

Debido a que todos los controladores se han estudiado por separado ya, aquí solamente se expondrán las distintas formas de conexión de dichos controladores y la manera en que estos interactúan.

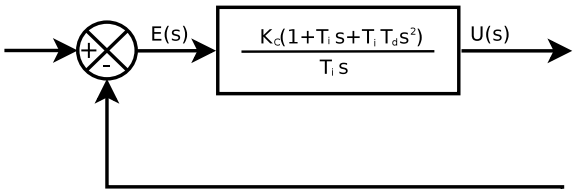


Figura 2.12: Diagrama de bloques del controlador PID.

Las acciones de control, pueden colocarse en serie o en paralelo, en la trayectoria directa o en la trayectoria de realimentación. Eso depende de la decisión del diseñador y, por supuesto, de las necesidades del sistema.

La primera forma analizada es conocida como forma estándar normalizada. Su función de transferencia es la siguiente:

$$\frac{U(s)}{E(s)} = K_C \left(1 + \frac{1}{T_i s} + T_d s\right) \tag{2.13}$$

Este arreglo es el que se utiliza con mayor frecuencia en el diseño de sistemas de control. Fue utilizado también en este proyecto de tesis.

La gráfica 2.12 muestra el diagrama de bloques de un controlador PID para este arreglo.

En las siguientes subsecciones se muestran otros arreglos de interés.

Control PI-D

En la figura 2.13 se muestra el diagrama de bloques de un sistema con controlador PID modificado. Se observa que el controlador Derivativo está en la trayectoria de realimentación y, por tanto, no afecta a la entrada de referencia. Con esta modificación se logra evitar el fenómeno conocido como *reacción del punto de ajuste*. Este fenómeno se presenta cuando la entrada de referencia es una función escalón y, debido a la presencia de la acción derivativa, la salida de la acción de control tendrá una función *impulso*.

Con la modificación presentada en la figura 2.13, la acción derivativa se aplica a la trayectoria de realimentación.

De la figura 2.13, la acción de control está dada por la siguiente expresión:

$$U(s) = K_C \left(1 + \frac{1}{T_i s}\right) R(s) - K_C \left(1 + \frac{1}{T_i s} + T_d s\right) B(s) \tag{2.14}$$

Donde se hace evidente que la entrada de referencia $R(s)$ no es afectada por la acción derivativa.

Control I-PD

En algunas ocasiones, cuando la señal de entrada de referencia es una señal escalón unitario, la acción de control $U(s)$ contiene también un escalón unitario. Es el caso del

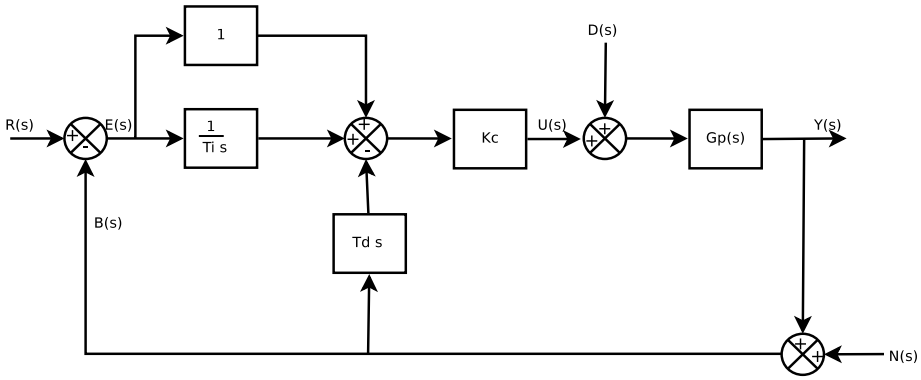


Figura 2.13: Ejemplo de un sistema $G_P(s)$ con un controlador PI-D, con presencia de perturbaciones.

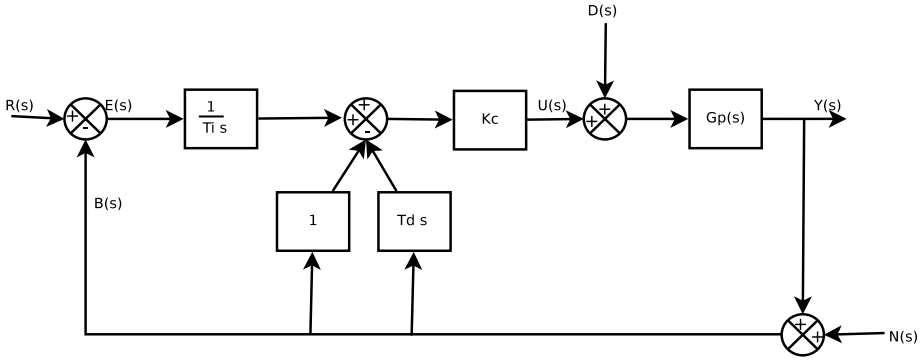


Figura 2.14: Ejemplo de un sistema $G_P(s)$ con un controlador I-PD, con presencia de perturbaciones.

control PID y el PI-D. Puede suceder que no sea conveniente ese cambio escalón en la señal de control. Para evitar que eso suceda se presenta el control I-PD, cuya acción dentro de un sistema se presenta en la figura 2.14.

En dicha figura, se observa que en el controlador I-PD, la acción de control proporcional se ha movido a la trayectoria de realimentación junto con el control derivativo. Por lo tanto, la única acción de control que afecta a la entrada de referencia es la integral.

La señal de control se obtiene mediante la siguiente expresión:

$$U(s) = K_C \frac{1}{T_i s} R(s) - K_C \left(1 + \frac{1}{T_i s} + T_d s\right) B(s) \tag{2.15}$$

Capítulo 3

TEORÍA BÁSICA ACERCA DE LA DISCRETIZACIÓN DE CONTROLADORES

3.1. Introducción

El enfoque tradicional para diseñar sistemas de control digital para plantas de tiempo continuo es diseñar primero un controlador analógico para la planta y después derivar su contraparte digital que se aproxime lo más posible al comportamiento del controlador analógico. En algunas ocasiones, sobre todo en la industria, por cuestiones de costo, es necesario usar este mismo enfoque cuando se decide sustituir un controlador analógico existente por uno digital. Al proceso de diseñar un controlador digital cuyo comportamiento sea lo más cercano posible a uno analógico, se le conoce como *discretización*.

Esta misma aproximación de un controlador analógico, resulta útil también cuando se desea simular éste por medio de una computadora digital, ya que estos solamente pueden interpretar datos discretos. Los programas especializados para simulación de sistemas, por ejemplo Matlab®, Octave® u otros similares tienen que auxiliarse del equivalente discreto de los sistemas analógicos que se desean simular para realizar su tarea. En este proyecto de tesis, la discretización se usa para obtener la función de transferencia discreta que ha de realizar el microcontrolador que valide el control.

Por las razones anteriores, este capítulo se dedica a estudiar y explicar varias técnicas para encontrar el equivalente discreto de un controlador analógico.

Antes de pasar a explicar los distintos métodos, conviene plantear y definir el problema de la discretización. Se puede describir así:

> Dado un controlador analógico con función de transferencia $G_C(s)$, ¿cuál es la función discreta equivalente $H(z)$, cuyo comportamiento se aproxime a $G_C(s)$?

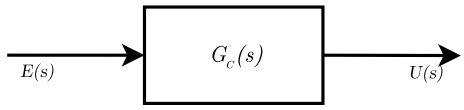


Figura 3.1: Diagrama de bloques de un controlador analógico.

En la figura 3.1 se muestra el diagrama de bloques de un controlador cuya función de transferencia es $G_C(s)$, la señal de entrada, es decir la señal de error, es $E(s)$ y la señal de salida (o señal de control) es $U(s)$. Si este controlador se define como un sistema lineal, causal e invariante en el tiempo, entonces su dinámica puede ser representada por medio de una ecuación diferencial ordinaria con coeficientes constantes.

Por lo anterior, se puede concluir que aproximar el comportamiento de un controlador analógico es similar a aproximar la solución de la ecuación diferencial ordinaria que modela a dicho controlador.

Los métodos que se analizan en este capítulo para aproximar la solución de una ecuación diferencial ordinaria son: *integración numérica* y *diferenciación numérica*. Estos dos métodos se explican en las siguientes dos secciones. El primero de ellos es el más usado por lo que se le dará mayor atención, además es el que se usa en esta tesis.

Aparte de las aproximaciones numéricas, existen otros métodos para aproximar controladores analógicos, a saber, *Coincidencia a la respuesta a escalón unitario y a otras respuestas* y *Coincidencia de polos y ceros*. Estos métodos se estudian en la sección 3.4.

3.2. Integración numérica

Uno de los métodos de aproximación de ecuaciones diferenciales ordinarias más conocidos es el de *integración numérica*.

La aproximación más común de integración numérica es dividir el intervalo de integración en muchos subintervalos T de igual tamaño y aproximar la integral total sumando la contribución de cada subintervalo, el cual será el producto de la base T por la altura $e(T)$; es decir, al área del rectángulo cuya base es T y cuya altura es $e(T)$.

A continuación se muestra cómo se deduce matemáticamente la expresión que relaciona la función de transferencia analógica, dada en términos de la variable s, con la función de transferencia discreta, dada en términos de la variable z, para el método de integración numérica.

Lo primero que se hace es considerar la función de transferencia $G_C(s)$ que corresponde a un integrador, con el objetivo de facilitar la deducción:

$$G_C(s) = \frac{F(s)}{E(s)} = \frac{1}{s} \tag{3.1}$$

Esta función de transferencia corresponde a la ecuación diferencial:

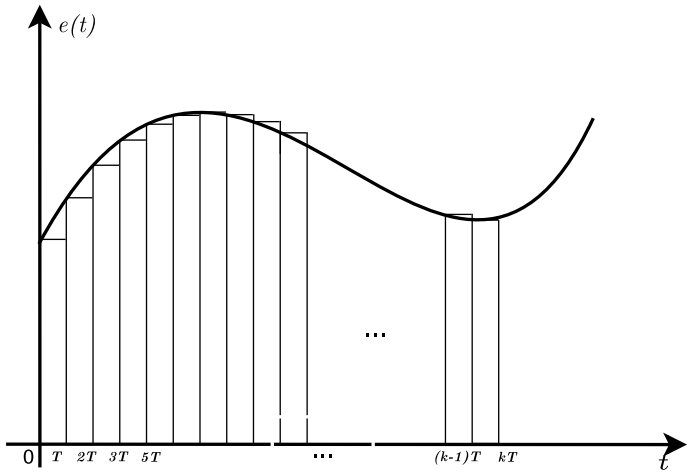


Figura 3.2: Aproximación de la integral por el método rectangular hacia adelante.

$$\frac{df}{dt} = e(t) \tag{3.2}$$

Al momento de integrar ambos miembros de la ecuación 3.2, desde t_0 hasta t, se tiene:

$$f(t) = f(t_0) + \int_{t_0}^{t} e(t)\,dt \qquad\qquad t \geq t_0 \tag{3.3}$$

Donde $f(t_0)$ es el valor de la integral hasta t_0.

Al hacer que el intervalo de integración sea en un periodo de muestreo, se tiene entonces que los límites de la integral son $t_0 = kT$ y $t = kT + T$; entonces,

$$f(kT + T) = f(kT) + \int_{kT}^{kT+T} e(t)\,dt \tag{3.4}$$

A continuación se detallan tres métodos para aproximar la integral 3.4.

3.2.1. Método de Euler hacia adelante

La forma más sencilla de resolver la integral de la ecuación 3.4 es simplemente aproximar el integrando por una constante igual al valor del integrando a la izquierda del punto final del subintervalo T y multiplicarlo por el valor de T. Lo anterior se observa mejor en la figura 3.2.

De esta manera, la ecuación 3.4 toma la siguiente forma:

$$\hat{f}(kT + T) = \hat{f}(kT) + Te(kT) \tag{3.5}$$

donde $\hat{f}$ representa el valor aproximado de f en el instante de tomar la muestra.
Al hallar la transformada z de la ecuación 3.5 se tiene:

$$z\hat{F}(z) = \hat{F}(z) + TE(z) \tag{3.6}$$

Y por consiguiente:

$$H(z) = \frac{\hat{F}(z)}{E(z)} = \frac{T}{z-1} \tag{3.7}$$

Como se pretende que los controladores analógico y digital correspondientes, se comporten de forma aproximada, se iguala la ecuación 3.1 con la 3.7, obteniendo con ello lo siguiente:

$$s = \frac{z-1}{T} \tag{3.8}$$

Esto significa que el equivalente en tiempo discreto de un controlador analógico puede ser determinado con el método de Euler hacia adelante, solamente reemplazando cada s en la función de transferencia del controlador analógico por $(z-1)/T$, esto es:

$$H(z) = G_C(s)|_{s=\frac{z-1}{T}} \tag{3.9}$$

3.2.2. Método de Euler hacia atrás

Este método es similar al anterior. La principal diferencia con aquel es que en lugar de aproximar el integrando en la ecuación 3.4 por su valor a la izquierda del punto final, el método de Euler hacia atrás aproxima el integrando por su valor a la derecha del punto final de cada subintervalo T y lo multiplica por el intervalo de muestreo, como en la figura 3.3.

Entonces, la ecuación 3.4 se convierte en:

$$\hat{f}(kT+T) = \hat{f}(kT) + Te(kT+T) \tag{3.10}$$

Siguiendo un procedimiento similar al seguido anteriormente para encontrar la relación de s con z, a continuación se encuentra la transformada de z de la ecuación anterior:

$$z\hat{F}(z) = \hat{F}(z) + TzE(z) \tag{3.11}$$

Por lo tanto, la función de transferencia discreta es:

$$H(z) = \frac{\hat{F}(z)}{E(z)} = \frac{Tz}{z-1} \tag{3.12}$$

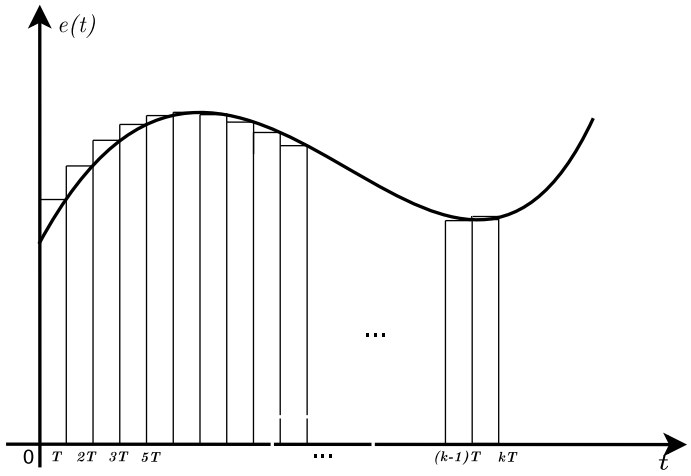


Figura 3.3: Aproximación de la integral por el método rectangular hacia atrás.

Igualando las ecuaciones 3.1 y 3.12, se encuentra que:

$$s = \frac{z-1}{Tz} \tag{3.13}$$

Con la ecuación anterior se deduce que la función de transferencia discreta equivalente del controlador analógico puede ser obtenida remplazando cada s en $G_C(s)$ con $(z-1)/Tz$, es decir:

$$H(z) = G_C(s)|_{s=\frac{z-1}{Tz}} \tag{3.14}$$

3.2.3. Método trapezoidal

Los métodos de Euler hacia adelante o hacia atrás, son conocidos como métodos rectangulares, porque durante el intervalo de muestreo el área bajo la curva es aproximada por un rectángulo. Adicionalmente, los métodos de Euler son conocidos también como de primer orden porque únicamente usan una muestra durante cada intervalo de muestreo.

Existe un tercer método para la discretización de controladores, que es conocido como *método trapezoidal* o de *Tustin*. Este método toma dos muestras en lugar de una como sucede con los métodos rectangulares. La idea subyacente detrás de este método es que la curva se aproxima por medio de una línea recta en el subintervalo estudiado. De esta manera, la aproximación del área bajo la curva en dicho intervalo será el área bajo la línea recta. Esto se muestra en la figura 3.4.

Al observar la figura 3.4, se entiende el nombre de "trapezoidal", ya que el área bajo la curva en el subintervalo en estudio se aproxima con el área de un trapecio.

Hablando en términos de control, resulta evidente que son necesarios dos muestras para discretizar un controlador con este método (muestra anterior y muestra presente); en lugar

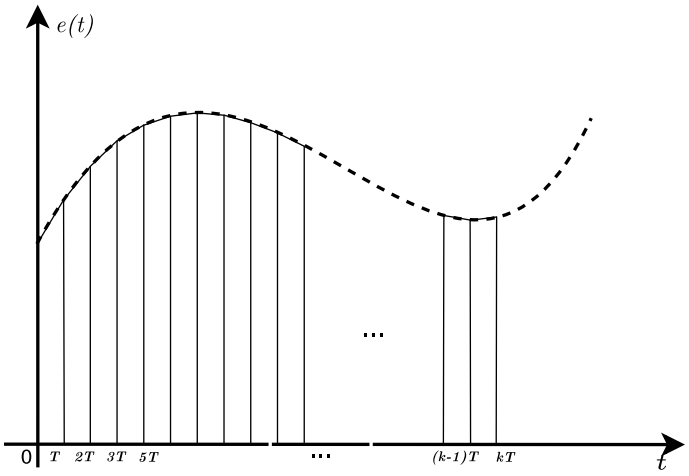


Figura 3.4: Aproximación de la integral por el método trapezoidal.

de uno como en los dos métodos anteriores. Por esta razón esta clase de aproximaciones es mas exacta en comparación con los métodos rectangulares.

Escribiendo todo esto en términos matemáticos, se tiene que:

$$\hat{f}(kT+T) = \hat{f}(kT) + \frac{T}{2}\left[e(kT) + e(kT+T)\right] \tag{3.15}$$

cuya transformada z es:

$$(z-1)\hat{F}(z) = \frac{T}{2}(z+1)E(z) \tag{3.16}$$

Reescribiendo 3.16:

$$H(z) = \frac{\hat{F}(z)}{E(z)} = \frac{T}{2}\left(\frac{z+1}{z-1}\right) \tag{3.17}$$

Comparando la ecuación 3.17, con la ecuación 3.1, la función de transferencia del controlador digital puede ser obtenido de la función de transferencia del controlador analógico, sustituyendo la s por $\frac{2}{T}(\frac{z-1}{z+1})$, esto es:

$$H(z) = G_C(s)|_{s=\frac{2}{T}(\frac{z-1}{z+1})} \tag{3.18}$$

El método trapezoidal también es conocido como *transformación bilineal*.

Como se había dicho, a este método se le considera como de segundo orden, debido a que se necesitan dos muestras en un intervalo de muestreo. Es posible aproximar la integral con más muestras en un intervalo, lo cual resultaría en métodos de mayor orden; por ejemplo, es posible aproximar la curva con un polinomio de orden 2 o superior. Es evidente que a mayor orden de aproximación , mejor aproximación de la integración analógica y la salida

del controlador digital resultante será más exacta, es decir, se aproximará mejor a la salida del controlador analógico del cuál se derivó.

Sin embargo, hay que recordar que la exactitud de la salida del controlador digital, no depende solamente del orden de aproximación sino también de la tasa de muestreo. Así, puede ocurrir que se tenga un controlador digital que haya sido aproximado por el método rectangular, pero cuya salida puede ser muy exacta si se tiene una tasa de muestreo apropiada.

Debido a que tanto los métodos de Euler como el de Tustin resultan en controladores digitales del mismo orden[1], los diseñadores generalmente escogen este último método para aproximar los controladores analógicos.

Hasta aquí se ha dicho que para encontrar la función de transferencia discreta equivalente de un controlador analógico se sustituye la s por su correspondiente transmitancia z, lo cual, matemáticamente significa un mapeo del plano s al plano z. En lo que resta de esta sección se analizará qué sucede durante este mapeo, con ayuda de gráficas para una mejor comprensión.

3.2.4. Mapeo del plano s al plano z por distintos métodos de discretización.

En las figuras 3.5, 3.6 3.7 se muestran gráficamente los mapeos del plano s al plano z para los métodos de discretización analizados hasta aquí.

En la figura 3.5 se puede apreciar que el método de Euler hacia adelante puede mapear polos que se encuentran en el semiplano izquierdo del plano s a puntos que se encuentren fuera del círculo unitario en el plano z. Dicho con otras palabras, puede suceder que dado un controlador analógico estable, su contraparte digital aproximado con el método de Euler hacia adelante sea inestable.

El método de Euler hacia atrás mapea el semiplano izquierdo del plano s a una región dentro del círculo unitario del plano z como se muestra en la figura 3.6. Se puede observar que en este mapeo no existen problemas de estabilidad en el controlador digital siempre y cuando el controlador analógico del cuál se derivó sea estable. Sin embargo, la respuesta del controlador digital siempre será oscilatoria debido a que la parte real de los polos del controlador digital serán siempre positivos y menores a uno.

Con respecto al método de Tustin, la figura 3.7 muestra qué sucede con el mapeo del plano s al plano z. 3.7.

Se puede ver que de los tres métodos estudiados en esta sección, el método de Tustin es el que hace un mapeo más "fiel", es decir, el semiplano izquierdo del plano s es mapeado dentro del círculo unitario en el plano z; el semiplano derecho de s es mapeado fuera del círculo unitario y el eje imaginario es mapeado en la circunferencia unitaria en el plano z. Esa es otra de las razones por la que los diseñadores prefieren la discretización de controladores analógicos por el método de *Tustin*.

[1] Es decir, el orden de la ecuaciones en diferencias resultante es el mismo, como puede observarse en las ecuaciones 3.14 y 3.18.

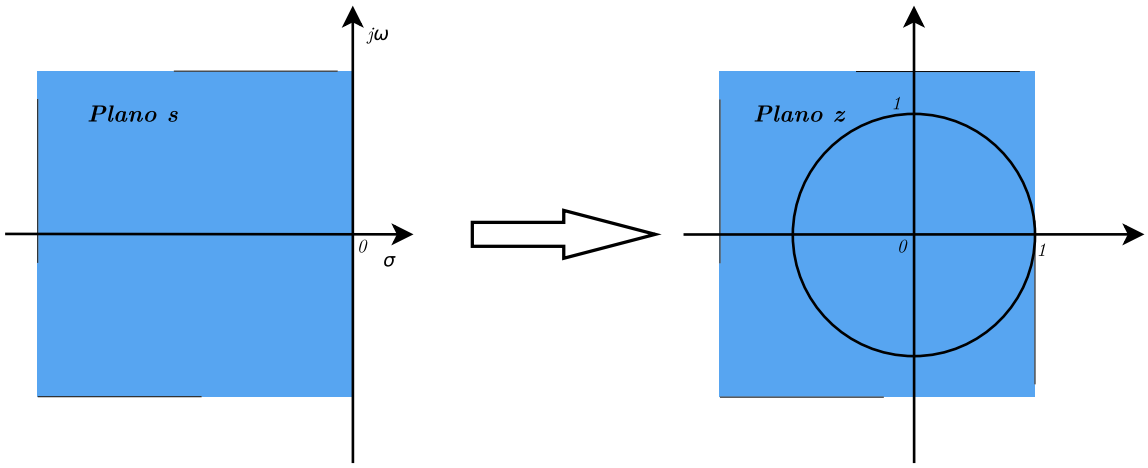


Figura 3.5: Mapeo del plano s al plano z del método de Euler hacia adelante.

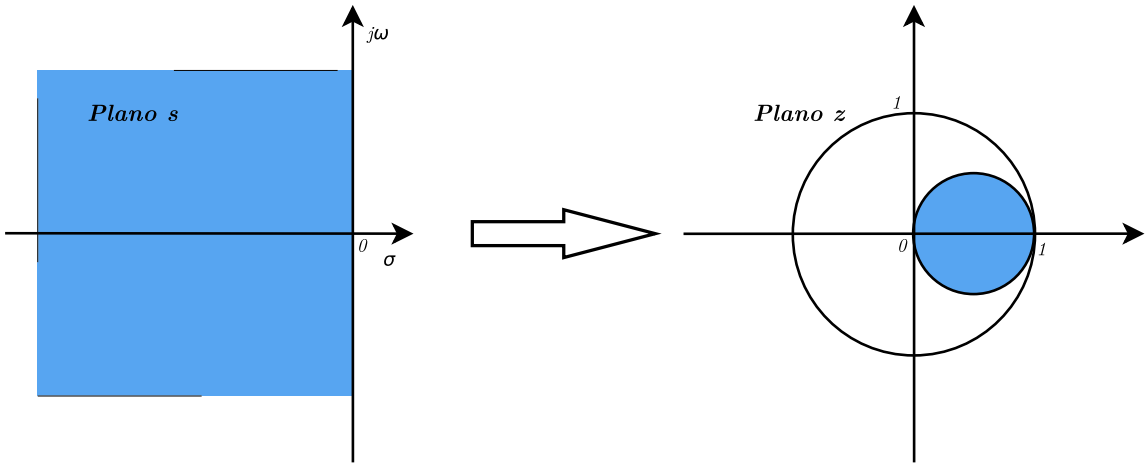


Figura 3.6: Mapeo del plano s al plano z del método de Euler hacia atrás.

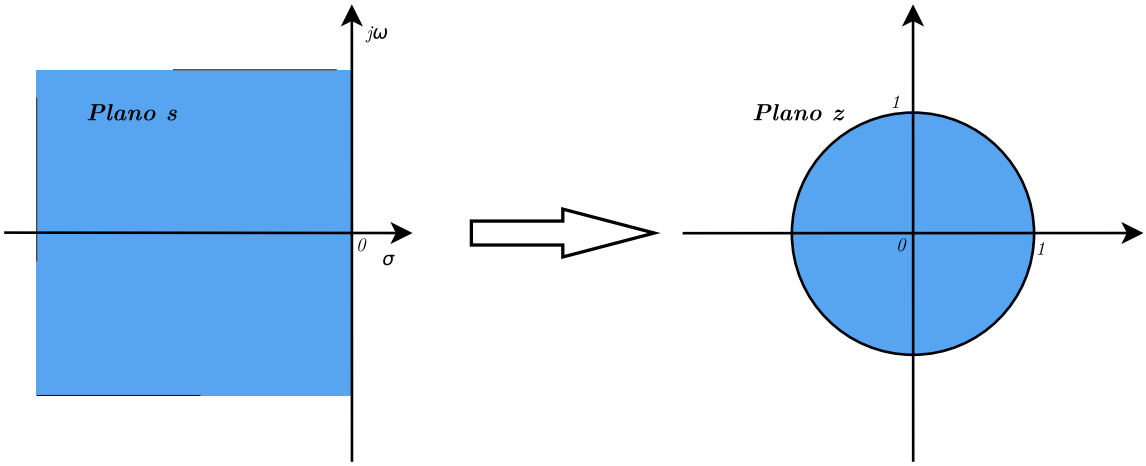


Figura 3.7: Mapeo del plano s al plano z del método de Tustin.

3.3. Diferenciación numérica

En esta sección se estudia otra forma de aproximar la solución de una ecuación diferencial: *la diferenciación numérica*.

En este método la solución aproximada de la ecuación diferencial es obtenida remplazando el controlador puramente integral por uno puramente derivativo. Después, este término se sustituye por un término de diferencia finita. La ecuación en diferencias resultante se resuelve entonces numéricamente.

La manera de encontrar la relación entre s y z será la misma que la utilizada en la integración numérica; de esta manera, ya no será necesario ser tan explícito en el procedimiento.

Si consideramos la función de transferencia:

$$G_C(s) = \frac{F(s)}{E(s)} = s \tag{3.19}$$

Cuya ecuación diferencial es:

$$f(t) = \frac{de(t)}{dt} \tag{3.20}$$

La forma más común de aproximar la ecuación diferencial anterior es por medio de incrementos, es decir:

$$f(t) = \frac{e(t + \Delta t) - e(t)}{\Delta t} \tag{3.21}$$

que es la aproximación de la derivada (la pendiente de la recta tangente) en el punto $e(t)$. Al hacer que el incremento Δt sea igual al intervalo de muestreo, se tendrá una función $f(t)$ que será una aproximación del controlador digital para un controlador analógico dado. A continuación se muestran dos formas de encontrar esta aproximación.

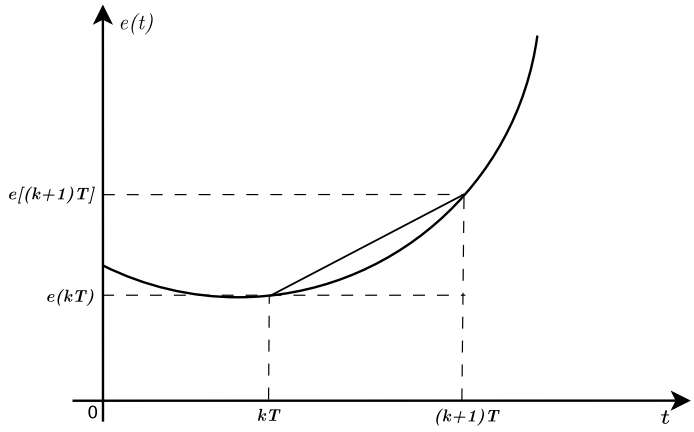


Figura 3.8: Aproximación de la diferencial. Diferencia hacia adelante.

3.3.1. Diferencia hacia delante

En la gráfica 3.8 se puede observar cómo se aproxima la curva por una recta en el intervalo de muestreo. Matemáticamente se puede expresar como:

$$\hat{f}(t) = \frac{e[(k+1)T] - e(kT)}{T} \tag{3.22}$$

Al hallar la transformada z de 3.22

$$\hat{F}(z) = \frac{E(z)(z-1)}{T} \tag{3.23}$$

De donde:

$$\frac{\hat{F}(z)}{E(z)} = \frac{z-1}{T} \tag{3.24}$$

Igualando 3.19 y 3.24 se tiene:

$$s = \frac{z-1}{T} \tag{3.25}$$

Por lo tanto, para encontrar la función de transferencia de un controlador digital dado uno analógico, se sustituye la s por $(z-1)/T$; es decir:

$$H(z) = G_C(s)|_{s=\frac{z-1}{T}} \tag{3.26}$$

3.3.2. Diferencia hacia atrás

Para este caso se procede de la misma manera que en la diferencia hacia adelante. Se toma en cuenta la figura 3.9. De esta manera,

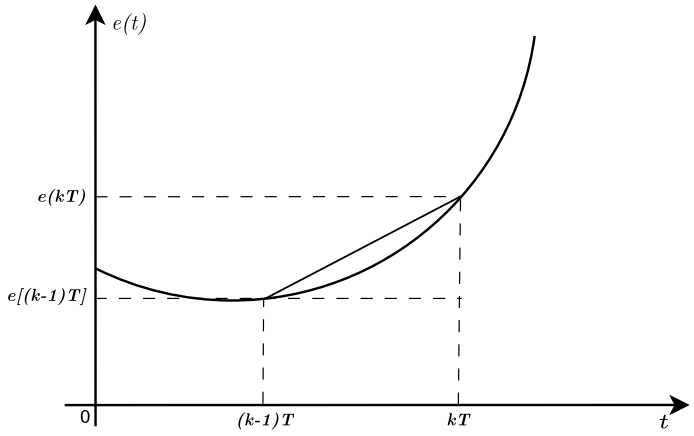


Figura 3.9: Aproximación de la diferencial. Diferencia hacia atrás.

$$\hat{f}(t) = \frac{e(kT) - e[(kT-1)T]}{T} \tag{3.27}$$

Al encontrar la transformada z de 3.27 y encontrar la función de transferencia:

$$\frac{\hat{F}(t)}{E(z)} = \frac{z-1}{Tz} \tag{3.28}$$

Por lo tanto, dado un controlador analógico, su aproximación digital correspondiente se puede encontrar sustituyendo la s por $(z-1)/Tz$; es decir:

$$H(z) = G_C(s)|_{s=\frac{z-1}{Tz}} \tag{3.29}$$

3.4. Otros métodos

Existen otros métodos para simular un controlador analógico. En esta sección se estudian algunos, a saber: Coincidencia a la respuesta a escalón unitario y a otras respuestas y Coincidencia de Polos y Ceros.

3.4.1. Coincidencia a la respuesta a escalón unitario y a otras respuestas

Otra forma de aproximar el comportamiento de un controlador analógico con uno digital es hacer que en las muestras (en el instante de tomar la muestra) la respuesta a escalón unitario del controlador digital coincida con la respuesta a escalón unitario del controlador analógico. En las figuras 3.10 y 3.11 se observa mejor esto.

Considerando la respuesta a escalón unitario del controlador analógico con función de transferencia $G_C(s)$ como en la figura 3.10a, el método en cuestión tiene como objetivo

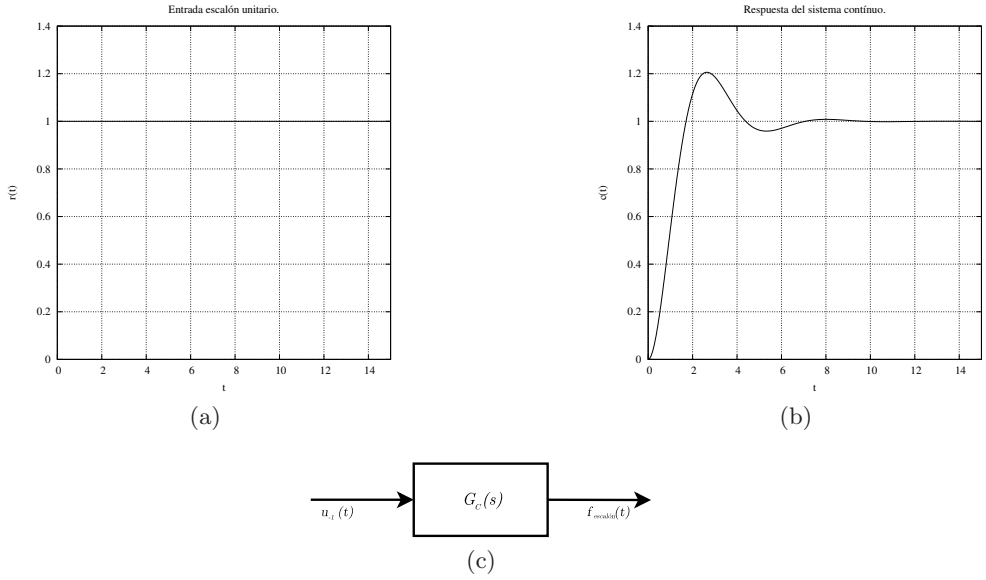


Figura 3.10: Respuesta a escalón unitario de un sistema analógico.

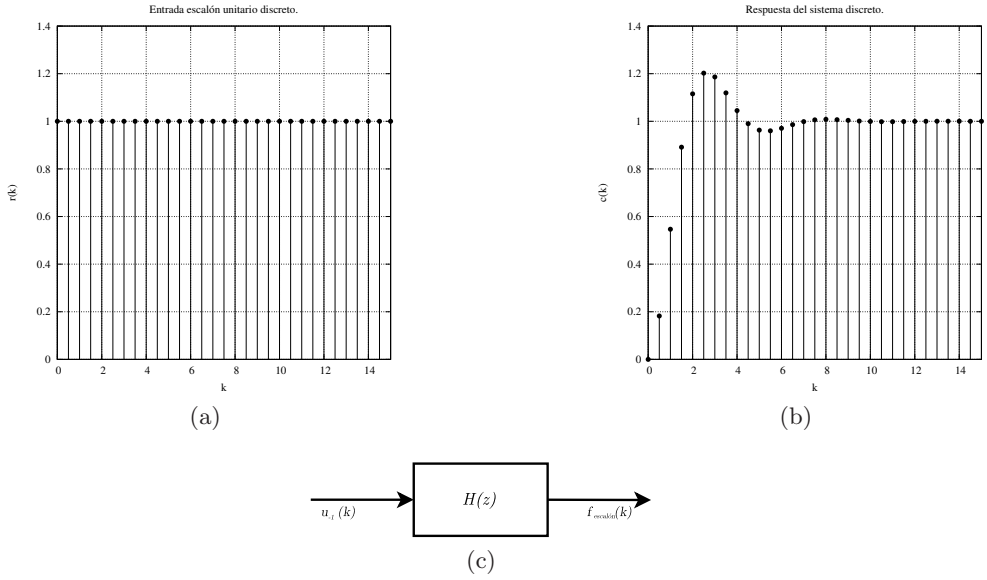


Figura 3.11: Respuesta a escalón unitario de un sistema discreto.

diseñar un controlador digital $H(z)$ como en la figura 3.11b, de tal manera que su respuesta a escalón unitario discreto consista en muestras de la respuesta a escalón unitario del referido controlador analógico. De esta manera, como se muestra en la figura 3.11a, el controlador digital tiene una respuesta a escalón que iguala a la respuesta del controlador analógico en los tiempos de muestreo.

Matemáticamente:

Dado un controlador analógico $G_C(s)$, su respuesta a escalón unitario es:

$$F_{escalon}(s) = \frac{1}{s} G_C(s) \tag{3.30}$$

Que en el dominio del tiempo es:

$$f_{escalon}(s) = u_{-1}(t) * g_c(t) \tag{3.31}$$

Donde $*$ es el símbolo de *convolución*. Al discretizar 3.31 con un periodo de de T:

$$f_{escalon}(kT) = u_{-1}(kT) * g_c(kT) \tag{3.32}$$

La transformada z de 3.32 es:

$$F_{escalon}(z) = \left(\frac{z}{z-1}\right) G_C(z) \tag{3.33}$$

La ecuación 3.33 representa la transformada z de la salida muestreada del sistema $G_C(s)$, es decir, la que se observa en la figura 3.11. Por otro lado, la transformada z de la respuesta a escalón unitario de un sistema discreto $H(z)$ se puede escribir como:

$$F_{escalon}(z) = \left(\frac{z}{z-1}\right) H(z) \tag{3.34}$$

De donde se concluye que la función de transferencia de un controlador discreto puede ser obtenida por medio del muestreo de la respuesta a escalón unitario de un controlador analógico por medio de la siguiente relación:

$$H(z) = \left(\frac{z-1}{z}\right) F_{escalon}(z) \tag{3.35}$$

Algunas veces, también es deseable diseñar un controlador digital cuya respuesta sea la respuesta muestreada a una rampa o al impulso de un controlador analógico. Se procederá de la misma forma que como se hizo con la respuesta a escalón unitario, en caso de que se desee.

3.4.2. Coincidencia de polos y ceros

Otro método de aproximar un controlador analógico por uno digital es mapear los polos y ceros de la función de transferencia del controlador analógico $G_C(s)$ a aquellos correspondientes del controlador digital $H(z)$ de la siguiente manera:

$$(s+a) \longrightarrow z - e^{-aT} \tag{3.36}$$

Para raices reales, y

$$(s+a)^2 + b^2 \longrightarrow z^2 - 2(e^{-aT}\cos bT)z + e^{-2aT} \tag{3.37}$$

Para raíces complejas conjugadas.

Generalmente, un controlador analógico tiene más polos finitos que ceros. En este caso su respuesta a altas frecuencias tiende a cero conforme ω_C tiende a infinito. Debido a que el eje imaginario $j\omega$ del plano s es mapeado completamente dentro de una revolución completa del círculo unitario en el plano z, la frecuencia más alta posible sobre el eje $j\omega$ es de $\omega_C = \frac{\pi}{T}$. Por lo tanto,

$$z = e^{sT} = e^{j(\pi/T)T} = -1 \tag{3.38}$$

Y, entonces, los ceros que estén en el infinito del controlador analógico mapean en ceros finitos localizados en $z = -1$ en el controlador digital equivalente. La función de transferencia resultante del controlador digital siempre tendrá un número de polos igual al números de ceros. Todos finitos.

Si el controlador digital tiene polos y ceros en el origen del plano s la ganancia K del controlador digital es seleccionada de tal forma que coincida con la ganancia del controlador analógico a determinadas frecuencias.

A bajas frecuencias,

$$H(z)|_{z=1} = G_C(s)|_{s=0} \tag{3.39}$$

Y a altas frecuencias,

$$H(z)|_{z=-1} = G_C(s)|_{s\to\infty} \tag{3.40}$$

3.5. Consideraciones sobre el periodo de muestreo

En los métodos que se han estudiado en este capítulo para discretizar un controlador analógico, se ha dicho que tiene que hacerse sobre un determinado periodo de muestreo, el cual se ha denotado por la letra T. En la discretización, el concepto de periodo de muestreo es sumamente importante, ya que de él dependerá, en gran medida, qué tanto se aproxima el controlador digital al analógico del cuál se derivó.

3.5.1. El problema del *aliasing*

Una mala elección del periodo de muestreo puede provocar *aliasing*.

El *aliasing* es un fenómeno que se presenta cuando se hace una mala elección del periodo de muestreo. El concepto de *aliasing* puede ser comprendido si se hace un experimento mental. Un observador frente a un movimiento pendular, puede reproducir y explicar este fenómeno. Si el observador "toma muestras" de la posición del péndulo en un instante determinado cerrando y abriendo los ojos con una frecuencia constante, entonces verá cómo el péndulo va cambiando de posición. Sin embargo, si la frecuencia de parpadeo del observador iguala la frecuencia del péndulo, entonces verá que éste se mantiene estático. Evidentemente, ésta información es incorrecta, ya que se supuso en primer lugar que el péndulo estaba en movimiento, pero esto no puede saberlo el observador y creerá que, en realidad el péndulo se mantiene en reposo en una determinada posición, ya que no tiene forma de saber que la información es errónea. Sobra decir que esta misma información puede ser obtenida si el observador ve un péndulo estático. A partir de lo anterior, un observador no podrá diferenciar si lo que ve es en realidad un péndulo estático o no.

Algo similar ocurre en el campo de las señales. Dos señales continuas distintas pueden producir la misma señal discreta después de que han sido muestreadas con frecuencias distintas. A este fenómeno se le conoce como *aliasing*.

3.5.2. El teorema de Nyquist-Shannon

El teorema de Nyquist, se refiere a la mínima frecuencia de muestreo que es posible hacer sobre una señal sin que haya pérdida de información, es decir, que sea posible reconstruir la señal a partir de sus muestras.

El teorema de Nyquist-Shannon dice:

> Si una señal no contiene frecuencias mayores a f_S Hertz, es completamente caracterizado por los valores de la señal en instantes de tiempo separados por $\frac{1}{2f_S}$ segundos.

Dicho con otras palabras, el teorema de muestreo de Nyquist sugiere que la mínima frecuencia que se tiene que usar en un muestreo debe ser dos veces la frecuencia máxima que contiene dicha señal. De otra forma, es imposible reconstruir la señal a partir de sus muestras.

Matemáticamente:

$$f_{S_{min}} \geq 2BW \tag{3.41}$$

Donde BW es el ancho de banda de la señal continua.

De aquí se puede concluir que la elección del periodo de muestreo depende de la máxima frecuencia que contiene la señal a discretizar.

3.5.3. Velocidad de procesamiento de la computadora

Dado que generalmente el objetivo de muestrear o discretizar una señal analógica es que ésta sea reconocida o tratada por una computadora digital, la elección del periodo de muestreo dependerá también de la velocidad de procesamiento de la computadora en el que se va a implantar el algoritmo o programa que manipulará la señal digitalizada.

La velocidad de procesamiento de la computadora es un parámetro muy importante ya que el periodo de muestreo escogido debe estar acorde a esta velocidad para evitar la pérdida de datos, es decir, la computadora debe de haber podido terminar de procesar la señal muestreada en un instante t determinado antes de que la siguiente muestra aparezca en el instante $t + T$, donde T es el periodo de muestreo. De no ser así, el sistema tendrá un comportamiento errático.

Capítulo 4

CIRCUITOS ANALÓGICOS

4.1. Introducción

En los capítulos precedentes se ha visto la teoría básica relacionada con los sistemas de control analógico y digital. Se han explicado los distintos tipos de controladores y las diferentes formas de discretización para encontrar los controladores digitales, así como las consideraciones que se deben tomar en cuenta para evitar fenómenos como el *aliasing*.

En este capítulo se inicia la explicación sobre cómo fue realizado el prototipo objeto de esta tesis. En particular, en este capítulo se explica cómo se implementaron físicamente los distintos bloques de un sistema básico de control. Tal como se aclaró en la introducción de esta tesis se usó un microcontrolador para realizar el algoritmo de control. A manera de ilustración, en la figura 4.1 se reproduce el diagrama de bloques conceptual de cómo se ideó este proyecto.

En este capítulo se hace hincapié sobre cómo se realizaron físicamente los siguientes bloques:

- La fuente de alimentación.
- El *Set Point*.
- El restador.
- El circuito de adecuación de entrada.
- El circuito de salida.

El bloque del microcontrolador, junto con el bloque de adecuación de niveles TTL, se estudia en el siguiente capítulo.

Conviene resaltar aquí la posición preponderante que tiene el bloque del microcontrolador en la figura 4.1, debido a que se le considera el *cerebro* de todo el proyecto. El

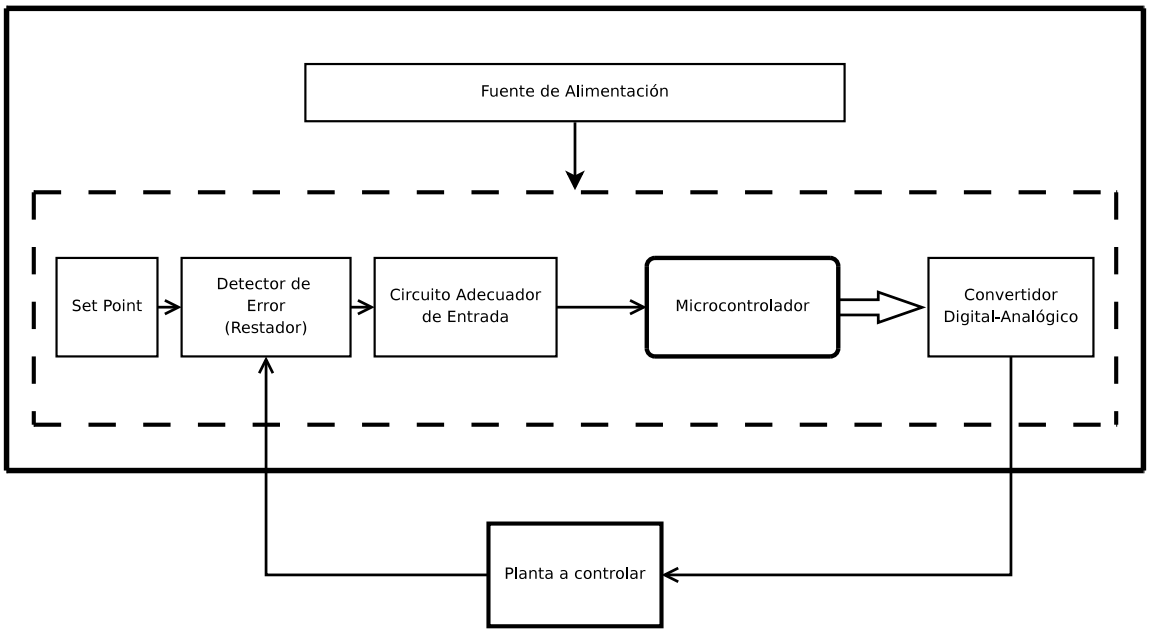


Figura 4.1: Diagrama de bloques conceptual del controlador digital.

microcontrolador empleado fue el 68HC908GP32 fabricado por Freescale™. En la referencia [9] se pueden leer las características técnicas de este microcontrolador.

En la sección 4.2 se explica cómo fue construida la fuente de alimentación. En la sección 4.5 denominada *Circuito adecuador de entrada*, se explica y analiza cómo fue construido el circuito que acondiciona la señal que entra al módulo de conversión analógico a digital del microcontrolador; mientras que en la sección 4.6 se explica cómo se construyó el circuito de conversión digital a analógico.

Cada una de las secciones que corresponde a cada circuito se divide en subsecciones que hablan sobre las características de diseño, la explicación de su funcionamiento y el porqué de la necesidad de éstos.

4.2. La fuente de alimentación

La fuente de alimentación es probablemente uno de los componentes menos valorados en un circuito electrónico y en cuyo diseño no se pone el cuidado que debería ponerse. Sin embargo, pese a esto, la fuente de alimentación constituye una de las partes más importantes de todo el circuito, ya que sin ésta, resultaría imposible que los demás componentes hicieran su trabajo.

El diseño de una fuente de alimentación es una tarea que, aunque a primera vista parece

sencilla, se debe tomar muy en serio, ya que una fuente mal diseñada podría estropear el diseño de las otras partes del sistema.

Existen diversos parámetros que definen el diseño de una fuente, entre los cuales, la regulación, el voltaje de salida y la corriente son los más comunes.

De igual manera, existen diversas formas de enfrentar la tarea del diseño de una fuente de alimentación; desde los más simples, que consisten en pocos componentes, hasta los más sofisticados que constituyen por sí mismos un sistema de control. La elección de una forma de diseño depende en gran medida del circuito que se pretende alimentar con dicha fuente. Así, un diseñador puede optar por construir una fuente consistente en pocos componentes si, con un previo análisis, se concluye que dicha fuente funcionará bien, sin afectar negativamente el desempeño total del sistema; sin embargo, no podrá hacerlo, si el circuito que se pretende alimentar es, por ejemplo, un sistema de comunicaciones, en donde una fuente mal diseñada puede introducir ruido electrónico al circuito y provocar que el sistema completo no funcione.

Una de las características que debe reunir una fuente de alimentación es la capacidad de regulación, es decir, el voltaje de salida con carga debe ser la misma que la salida sin carga. De ahí la importancia de conocer *a priori* al circuito que se pretende alimentar con la fuente. La forma de regulación es otra de las decisiones que debe tomar el diseñador dependiendo del circuito total. Entre las opciones que existen en la actualidad están los más sencillos como el diodo zener o un transistor TBJ en configuración base común. De igual manera, se venden en el mercado circuitos reguladores lineales encapsulados en un *chip*.

4.2.1. Requerimientos de la fuente de alimentación

Para saber los requerimientos de la fuente es necesario conocer, como se ha dicho, al circuito que se pretende alimentar.

En este proyecto, el circuito consiste en dos tarjetas impresas, que se van a alimentar con la fuente en cuestión. La primera tarjeta contiene el adecuador de entrada, el convertidor digital a analógico, el restador y el circuito que fija la referencia o *set point*. La segunda tarjeta contiene el microcontrolador y la interfaz de comunicación RS-232, es decir la que convierte las señales digitales con lógica TTL a niveles que pueda reconocer el computador anfitrión. Cabe resaltar aquí, que esta fuente no alimentará a la planta a controlar, ésta deberá tener su propia fuente de alimentación.

De acuerdo a sus hojas de datos, el convertidor digital a analógico [13] y los amplificadores operacionales encapsulados [4] se alimentan con una fuente de ± 15 [V]. El microcontrolador y el circuito de conversión de niveles TTL a niveles RS-232 y viceversa se alimentan con un voltaje de 5 [V].

Con un análisis[1] previo sobre las necesidades de los circuitos de todo el sistema se

[1] Se tomaron en cuenta los rangos máximos absolutos de los circuitos integrados así como mediciones hechas en el laboratorio.

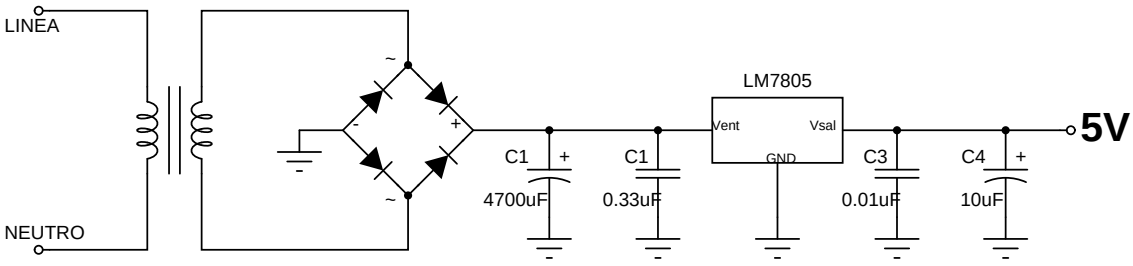


Figura 4.2: Circuito de la fuente de alimentación de 5 [V].

decidió usar un regulador lineal encapsulado existente en el mercado, ya que, según su hoja de características eléctricas (véase [14] y [15]), cumple con las necesidades del proyecto.

Por todo lo anterior, los requerimientos de la fuente de alimentación son los siguientes:

Voltaje de Salida:	Constante: 15 [V], -15 [V] y 5 [V].
Corriente mínima de salida:	500 [mA]
Regulación:	Buena.
Estabilidad:	Buena, debido a que el rango de voltajes de alimentación del microcontrolador es pequeño.

En la figura 4.2 se muestra el circuito de la fuente de alimentación de 5 [V]. Se observa, que el regulador utilizado es el circuito LM7805. Con dicho circuito, se logra tener una buena regulación que cumple con los requerimientos del sistema.

En la figura 4.3 se muestra el circuito de la fuente de alimentación de 15[V] y -15 [V]. De igual manera, se utilizan reguladores lineales encapsulados para tener la regulación requerida. Con los valores de los capacitores mostrados se logró tener un filtrado excelente, según pruebas hechas en el laboratorio.

4.3. El *set point* o punto de ajuste

El *Set Point* o punto de ajuste es la parte del circuito que permite ajustar la entrada de referencia. Con este circuito, el usuario puede manipular cómo quiere que sea la salida de la planta a controlar. Para este caso particular, la referencia es ajustada por medio de un potenciómetro de vuelta sencilla que varía en un rango de -10 volts a 10 volts. De esta manera, el usuario le indicará al sistema de control, la salida deseada.

De acuerdo a lo anterior, los requerimientos para este circuito son:

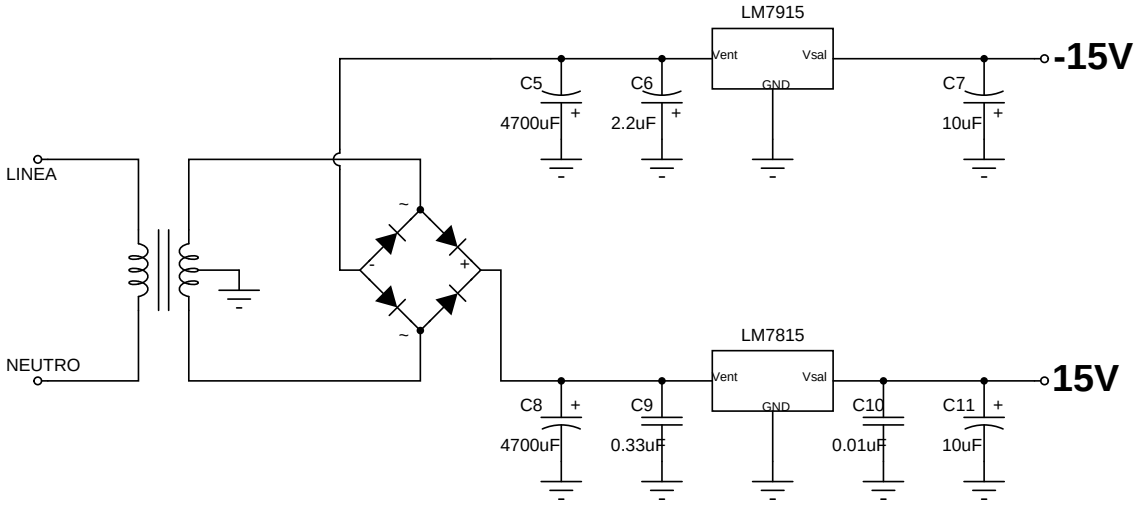


Figura 4.3: Circuito de la fuente de alimentación de 15 [V].

Voltaje de Salida:	Variable linealmente en un rango de -10[V] a 10 [V].
Impedancia de Salida:	Baja, para evitar cualquier desajuste provocado por un cambio en la carga del sistema.

Dado que se cuenta con una fuente de -15[V] a 15 [V], el circuito propuesto es un divisor de voltajes por medio de un arreglo de resistencias en serie y en paralelo cómo se muestra en la figura 4.4. El arreglo de resistencias mostrado en dicha figura consiste en dos resistencias conectadas en paralelo, los cuales, a su vez están conectados en serio con otra resistencia, todas de igual valor. Los extremos de este arreglo están conectados a la referencia del sistema (GND) y a un voltaje V. La salida de este arreglo en el punto a es:

$$V_a = \frac{2}{3}V$$

dado que todas las resistencias tienen igual valor.

En este caso particular, si la fuente V es de 15 volts, la salida es de 10 volts, tal como se desea.

Sin embargo, en la práctica, los valores reales de las resistencias suelen diferir ligeramente de sus valores nominales. De esta manera, un arreglo como el propuesto en la figura 4.4, puede ser diferente en la salida del pronosticado.

Por lo anterior, un circuito que cumpla cabalmente con los requerimientos mencionados anteriormente, se muestra en la figura 4.5.

Se resalta la inclusión de las resistencias variables POT1 y POT2. La función de estas resistencias es la de ajustar la salida desajustada provocada por las diferencias en los valores reales de las resistencias usadas. Así, cualquier diferencia en las resistencias puede ser

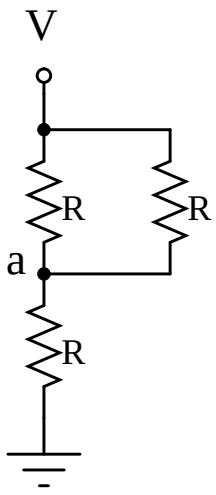


Figura 4.4: Primera aproximación al circuito del punto de ajuste.

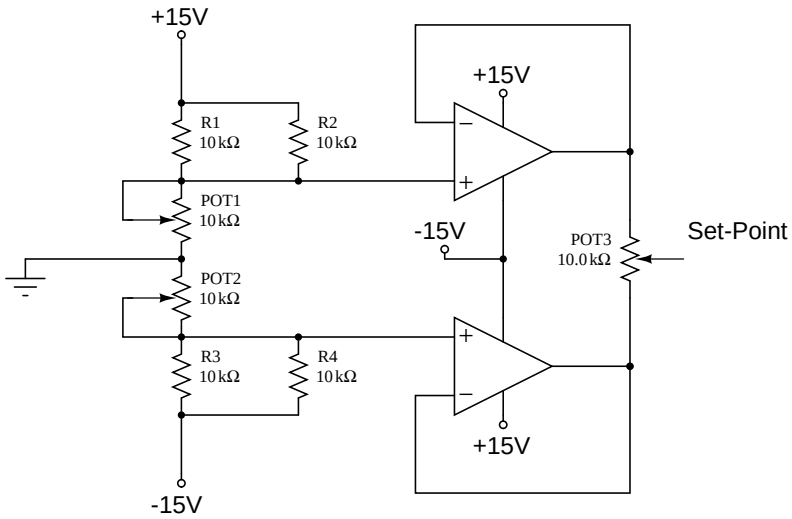


Figura 4.5: Circuito del punto de ajuste.

compensada o neutralizada ajustando solamente con el cursor de los potenciómetros POT1 y POT2 y hacer que la salida sea la más próxima a la deseada, en este caso de 10 volts y -10 volts.

Como se observa, el arreglo de resistencias en la figura 4.5, produce en la salida un voltaje variable, gracias al potenciómetro POT3, en el rango de -10 a 10 volts. Los dos amplificadores operacionales en su configuración seguidor de voltaje se usan para que el voltaje de salida sea la misma no importando las condiciones de carga en el circuito que está afuera del *set point* y al cuál es conectado éste.

4.4. El restador

El circuito restador surge de la necesidad de requerir de un circuito que detecte el error entre la salida de la planta y la entrada de referencia. La salida de este circuito será el error, misma que será la entrada del circuito controlador para que en función de ella se produzca la señal de control deseada.

Los requerimientos del circuito son:

Entradas:	La salida de la planta y el *set point*
Salida:	La diferencia entre el *set point* y la salida de la planta.
	La corriente que circule por el circuito debe ser tal que
	el amplificador operacional no se queme y sea suficiente para que
	sea leída correctamente por el adecuador de entrada.

El circuito propuesto para el restador es un amplificador operacional en su configuración de restador, el cual se muestra en la figura 4.6. Las entradas son la salida de la planta a controlar y el voltaje deseado, es decir, la salida del *set point*.

Los valores de resistencias mostradas tienen que ser de igual valor para que la operación efectuada por el circuito sea la correcta, en caso contrario, habrá una ganancia positiva o negativa de la diferencia.

Para corroborar el funcionamiento del circuito restador, se hizo una simulación con ayuda del program PSpice®. Los resultados de dichas simulaciones se muestran en la figura 4.7. En dicha figura se puede ver que, en efecto, el circuito del restador funciona. Se puede observar que la salida del circuito es la diferencia entre el *set point* y la salida de la planta. La línea sólida más gruesa representa la salida del circuito, la línea discontinua delgada representa la entrada en la entrada inversora del amplificador operacional y la línea discontinua gruesa representa la entrada en la entrada no inversora del amplificador operacional.

Como se dijo anteriormente, para que el restador funcione correctamente, los valores de las resistencias deben ser iguales. En la figura 4.6 se observa que las resistencias utilizadas tienen un valor de 4.7 [kΩ]. Este valor fue escogido utilizando como restricción la corriente

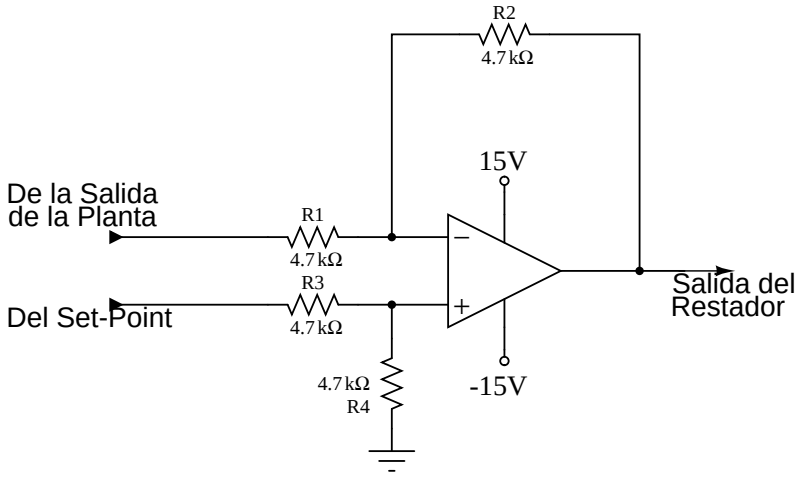


Figura 4.6: Circuito del detector de error.

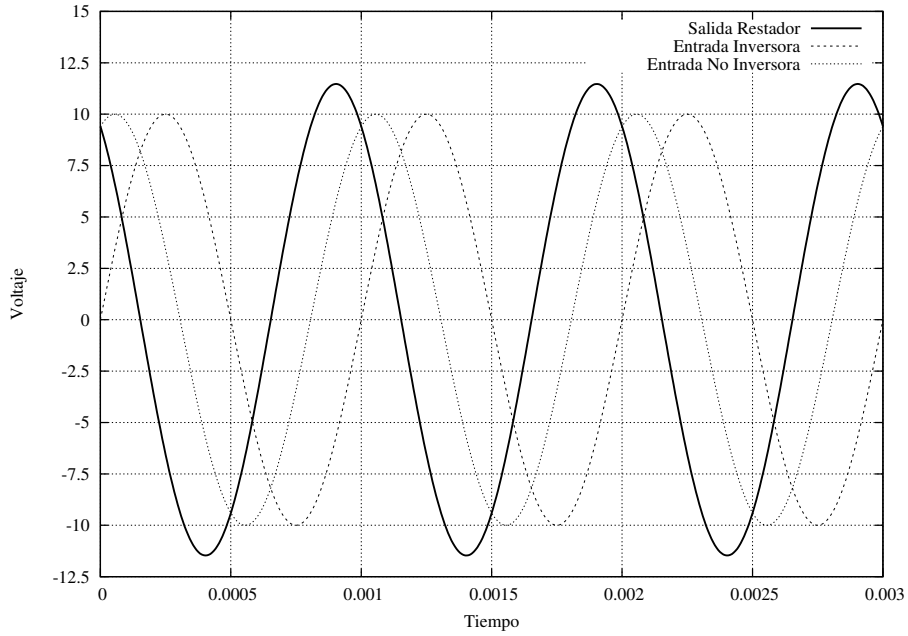


Figura 4.7: Resultados de la simulación del restador.

que debe circular por el amplificador operacional. Esta corriente debe ser menor al valor máximo permitido según [4] y también debe ser suficiente para que el voltaje de salida pueda ser reconocida por el circuito de la siguiente etapa.

4.5. El circuito de adecuación de entrada

Esta sección trata sobre la parte del sistema que sirve para el acondicionamiento de la señal de entrada y su implementación en un circuito físico.

Muchos dispositivos digitales de alta escala de integración como los microcontroladores, suelen contener como un módulo funcional, un convertidor analógico a digital (módulo ADC, de aquí en adelante). Ese es el caso del microcontrolador 68HC908GP32. Aunque la descripción completa de este dispositivo, junto con su modelo de programación, se hace en el capítulo siguiente, aquí se hará una breve descripción sobre el módulo de conversión ADC para fines de la explicación del diseño presentado en esta sección.

El módulo ADC del MCU68HC908GP32, de acuerdo con su hoja de características eléctricas[9], tiene las siguientes características:

- Voltaje de alimentación del módulo ADC V_{DDAD}: de 2.7 a 5.5 volts.
- Rango de voltaje de entrada al módulo ADC: 0 a V_{DDAD}(0 a 5[V] típico).
- La corriente máxima que puede soportar cada pin del microcontrolador (incluyendo el pin de entrada al convertidor analógico a digital) es $\pm$15 [mA].

Por otro lado, por las características de los circuitos que componen al sistema total, la salida del restador está en el intervalo de -10 [V] a 10 [V]. Por lo que, se hace evidente la necesidad de un sistema que convierta la señal de salida del restador al rango de entradas analógicas del microcontrolador, es decir, a un rango de 0[V] a 5[V].

4.5.1. Características técnicas del diseño.

Las características técnicas del diseño son:

Voltajes de Entrada	-10 a 10 [V]
Voltajes de Salida	0 a 5 [V]
Corriente máxima de salida	15 [mA]

La conversión debe ser lineal. Tal como lo muestra la figura 4.8.

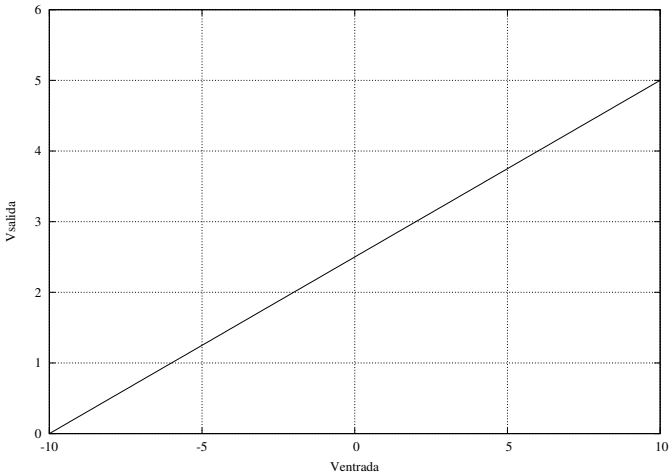


Figura 4.8: Relación entre el voltaje de entrada y el de salida en el circuito adecuador.

4.5.2. Diseño del circuito

Debido a que el sistema necesita ser lineal en un determinado rango, se decidió usar amplificadores operacionales, ya que estos componentes electrónicos tienen un amplio rango de linealidad, son muy económicos y de fácil uso.

En la referencia [4] se tienen las características eléctricas de los circuitos TL084. Estos circuitos tienen 4 amplificadores operacionales encapsulados dentro de un solo *chip*. Se decidió usar este circuito con el objeto de ahorrar espacio dentro de la tarjeta electrónica y por su bajo precio en el mercado.

En la figura 4.11 se muestra el circuito adecuador de entrada, es decir, el que convierte los voltajes de -10 [V] a +10 [V] al intervalo de 0 [V] a 5 [V]. A continuación se explica su funcionamiento:

La relación lineal que se presenta en la figura 4.8 se modela matemáticamente con la siguiente función:

$$V_{salida} = 0.25V_{entrada} + 2.5 \tag{4.1}$$

Esta función puede ser fácilmente implementada físicamente mediante el uso de un amplificador operacional, tal como se muestra en la figura 4.9[2].

Sin embargo, esta aproximación, aunque sencilla, presenta el inconveniente práctico de que los valores de las resistencias R_F, R_{i1} y R_{i2} y la fuente V_{dc} no aceptan ninguna tolerancia, sus valores reales deben ser idénticos a los calculados.

En la figura 4.11 se presenta el circuito mejorado de la primera versión, el cual fue implementado en este proyecto. Este circuito elimina la restricción de depender de los

[2]Aunque se hace evidente la necesidad de un amplificador inversor para representar fielmente la ecuación 4.1 debido a que la señal de salida es invertida.

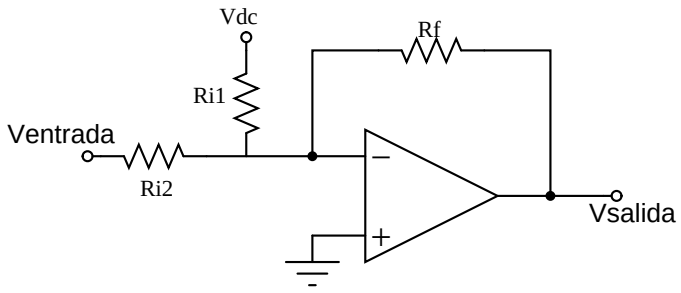


Figura 4.9: Primera aproximación para el circuito adecuador de entrada.

valores reales de los componentes debido a la presencia de los potenciómetros POT1 y POT2. Con estos potenciómetros se logran ajustar los desajustes que pudieran presentarse debido a los valores reales de las resistencias.

Los dos diodos de la figura 4.11 tienen la función de limitar el voltaje de salida en el rango de 0 a 5 volts.

El proceso de ajuste del adecuador de entrada se presenta en el siguiente apartado.

Calibración del adecuador de entrada

Se puede dividir el circuito del adecuador de entrada (figura 4.11) en dos subcircuitos: el sumador que ajusta el *offset* (representado por el amplificador operacional 1) y el amplificador que ajusta la ganancia (que contiene el amplificador operacional 2).

El funcionamiento del subcircuito 1 es como sigue: la configuración del circuito con amplificador operacional es un sumador inversor, cuyas entradas son 15 [V] y la señal de error. La salida será la suma de estas dos señales multiplicadas por sendas ganancias. Estas ganancias están dadas por $-\frac{R_3+R_4}{R_2}$ para la señal de error y $-\frac{R_3+R_4}{R_1+POT_1}$ para la señal de DC. Se observa que la ganancia de la entrada de 15 [V] depende del potenciómetro POT1. El resultado del movimiento del cursor de POT1 es una señal cuyo valor de *offset* va cambiando.

El subcircuito 2 se explica como sigue:

La configuración del circuito con amplificador operacional es un amplificador inversor. La ganancia de este amplificador está dado por $-\frac{R_6+POT_2}{R_5}$. Esta ganancia está en función de la posición del cursor del potenciómetro POT2. De esta manera, este subcircuito modifica la ganancia de todo el circuito adecuador de entrada, ya que la salida del subcircuito 2 depende de la salida del subcircuito 1.

Como ha quedado dicho, el segundo subcircuito invierte la señal de entrada. De igual manera, el subcircuito 1, invierte su entrada. De esta manera, dos inversiones a la señal de entrada, se anulan, es decir, la señal de salida tendrá la misma polaridad que la señal de entrada.

Antes de pasar al proceso de calibración se debe tomar en cuenta que para ajustar el *offset*, se tiene que mover el cursor del potenciómetro 1, es decir POT1, mientras que para ajustar la ganancia se debe mover el cursor del potenciómetro 2, es decir POT2.

Para calibrar el sistema adecuador de entrada, se tiene que seguir el siguiente proceso:

1. AJUSTAR EL *OFFSET*. Este paso consiste en mover el cursor de POT1 para que cumpla la siguiente condición: con un voltaje de entrada de -10 volts, la salida del adecuador debe ser de 0 volts.

2. AJUSTAR LA GANANCIA. Este paso consiste en mover el cursor de POT2 para que a una entrada de 10 [V] la salida del adecuador sea de 5 [V].

3. Verificar el paso 1, es decir, que el adecuador nuevamente cumpla la condición. En caso de no ser así se procederá a ajustar nuevamente el *offset*. Cuando se haya ajustado, se verificará que se cumple la condición del paso 2.

4. La calibración termina cuando se cumplan las dos condiciones anteriores.

4.5.3. Simulaciones

Las simulaciones para el diseño del circuito adecuador de entrada se hicieron en PSpice®. En la figura 4.10 se pueden observar los resultados de dichas simulaciones.

Para esta simulación se utilizó como señal de entrada, una señal senoidal con 1 [kHz] de frecuencia y una amplitud de 10 [V]. El resultado gráfico de esta simulación se muestra en la figura 4.10. En dicha figura se puede observar que la salida también es una señal senoidal con la misma frecuencia y la misma fase que la entrada pero cuya amplitud se relaciona con la entrada de acuerdo a la expresión 4.1. En la figura 4.10 la línea continua gruesa representa la salida del circuito adecuador, mientras que la línea discontinua representa la entrada a dicho adecuador.

En la figura 4.11 se muestra el circuito simulado.

4.6. Circuito de salida: Convertidor digital a analógico

El convertidor digital a analógico (DAC de aquí en adelante) es un circuito cuya función es servir de interfaz entre la planta o el actuador y el microcontrolador que genera la acción de control, previa resolución de la ecuación en diferencias por parte del microcontrolador. Como la mayoría de las plantas a controlar en el mundo real son analógicas, siempre se incluye este componente en un sistema de control digital. Antes de mostrar el diseño del circuito DAC, conviene mostrar el funcionamiento interno de éste.

En principio, se debe hacer notar que el DAC0800 (el cuál fue usado en el dispositivo prototipo, objeto de esta tesis) es un convertidor digital a analógico que internamente tiene un circuito R-2R, conocido más comúnmente como *circuito en escalera*. Este circuito tiene

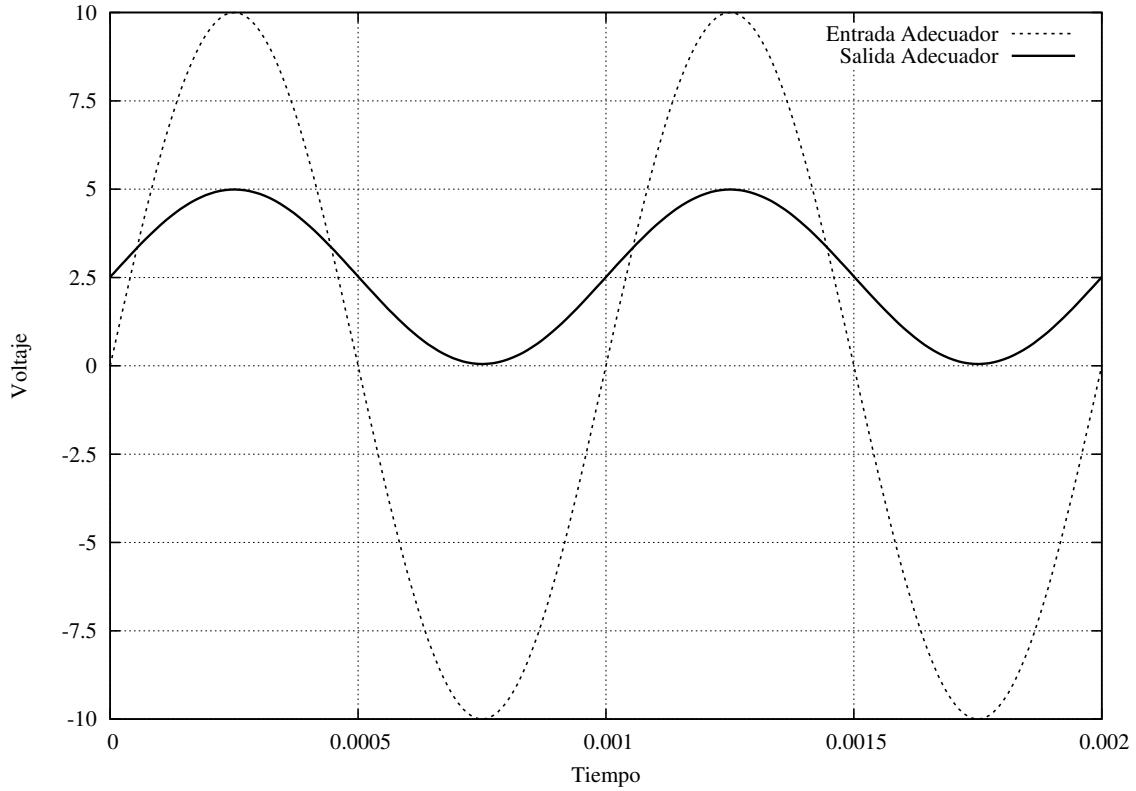


Figura 4.10: Resultado de la simulación del adecuador de entrada.

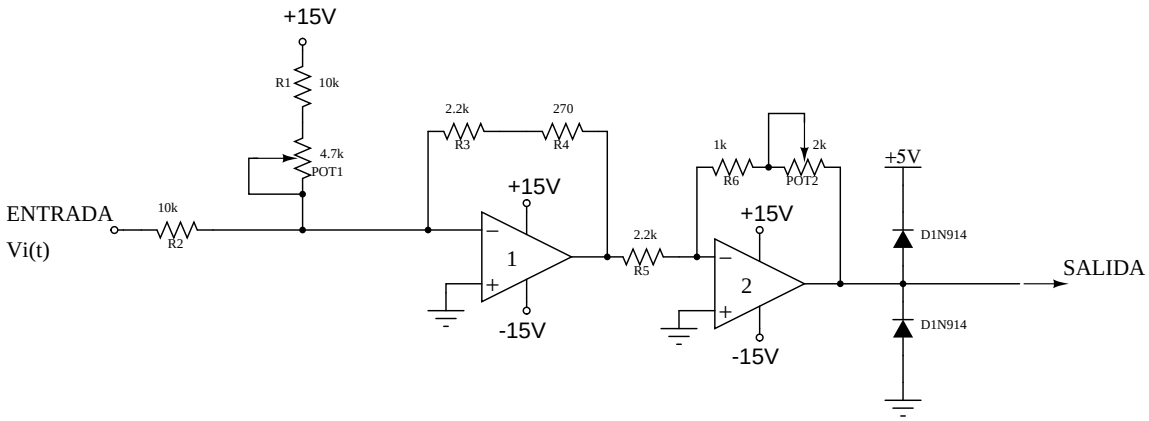


Figura 4.11: Circuito empleado para el adecuador de entrada.

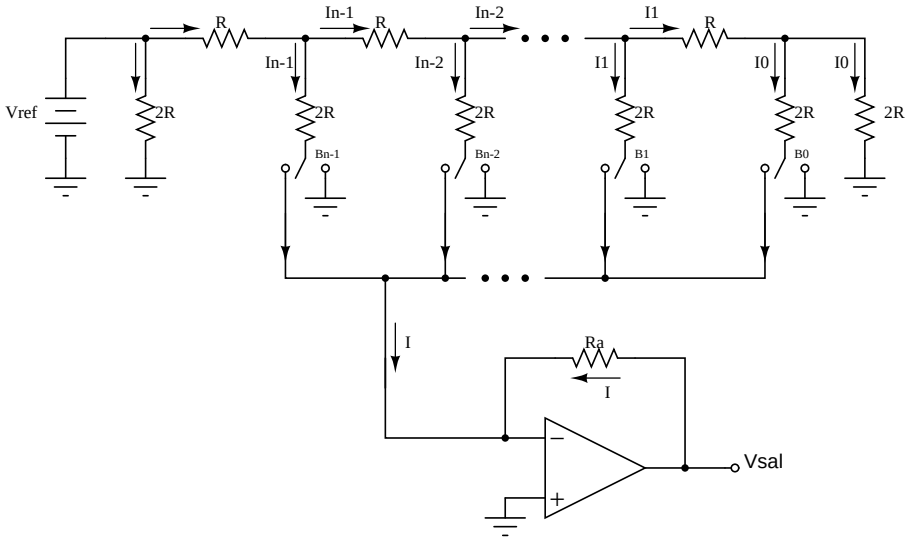


Figura 4.12: Circuito R-2R.

la particularidad de producir una corriente a su salida que es proporcional a un código binario (generado por un conjunto de interruptores) a la entrada del circuito, como se muestra en la figura 4.12.

En la figura 4.12 se observa el circuito R-2R de n interruptores. El arreglo de este circuito es tal que todas las resistencias que lo conforman son de valores R y $2R$. La particularidad de este circuito es que el equivalente del arreglo de resistencias es R. Con el arreglo del circuito R-2R, se logra que las corrientes que fluyen en las ramas sean proporcionales, es decir, que la corriente que fluye en una rama siempre sea el doble del que fluye en la rama precedente y será siempre la mitad que la que fluye en la rama siguiente.

El funcionamiento del circuito en escalera es el siguiente:

Un código binario de entrada acciona los interruptores (uno por cada dígito binario) conectado a la red de resistencias. En la figura 4.12 existen n interruptores, por lo tanto, el número binario de entrada puede ser de hasta n bits. Es conveniente señalar que el orden de los interruptores es importante, tal como se observa en la figura 4.12: el bit menos significativo es el de la derecha. La salida de esta red se da en forma de una corriente, la cuál será la suma total de las corrientes que circulan en cada rama cuando todos los interruptores están cerrados. En caso de que ningún interruptor esté cerrado, es decir, que todos estén puestos a tierra, la corriente que sale de la red será nula. Cuando no todos los interruptores se encuentran cerrados (o no todos están abiertos) la corriente de salida estará dada en forma proporcional a la cantidad binaria de entrada. De esta manera, la corriente de salida del circuito R-2R será proporcional al número binario dado.

De la figura 4.12

$$I = I_0 + I_1 + I_2 + ... + I_{n-2} + I_{n-1}$$

A su vez, por el arreglo de las resistencias y por sus valores:

$$\begin{aligned} I_0 &= \tfrac{I_1}{2} \\ I_1 &= \tfrac{I_2}{2} \\ I_2 &= \tfrac{I_3}{2} \\ &\dots \\ I_{n-2} &= \tfrac{I_{n-1}}{2} \end{aligned}$$

Debido a que la salida de este circuito es una corriente, es necesario usar un circuito que la convierta a un voltaje, dado que este último es más fácil de manipular que la corriente. La conversión de corriente a voltaje se hace por medio de un amplificador operacional. En el caso de la figura 4.12, se debe observar que cuando todos los interruptores estén colocados a tierra, la salida del circuito será de 0 [V]. Cuando todos los interruptores estén cerrados la salida será un voltaje negativo, que depende de la resistencia R_a. En caso de que se desee generar un voltaje positivo cuando todos los interruptores estén cerrados, la salida del circuito R-2R deberá ser conectado a la entrada no inversora del amplificador operacional.

Para lograr que un circuito en escalera R-2R tenga una salida bipolar se requiere que se haga una ligera modificación al circuito de la figura 4.12. El circuito modificado se muestra en la figura 4.13. Como se puede observar en dicha figura, se tienen dos líneas para el flujo de corriente. Cuando se accionan los interruptores por medio del código binario (dependiendo del valor de cada bit, el interruptor se cerrará hacia uno u otro lado) comenzará un flujo de corriente, el cual estará en alguna de las dos líneas. Por tanto, la suma de las corrientes en ambas líneas será siempre constante ya que no importando la posición en que se cierre el interruptor siempre habrá una corriente fluyendo en cualquiera de las dos líneas. Nuevamente, a la salida, se necesita un circuito convertidor de corriente a voltaje; éste puede ser implementado por medio de un amplificador operacional como el mostrado en la figura 4.13. Se debe observar que el voltaje de salida estará en el rango de $-V$ a $+V$, que depende del valor de las resistencias Ra. Ambas resistencias deben tener el mismo valor para lograr que el voltaje de salida sea simétrica con respecto a la referencia o tierra.

4.6.1. Diseño del circuito

El circuito de salida usado consiste en un *chip* DAC de 8 bits (por lo tanto, su resolución es de 256 estados).

Las características eléctricas de este dispositivo se muestran en su hoja de datos correspondiente [13]; sin embargo a continuación se mostrarán las más relevantes:

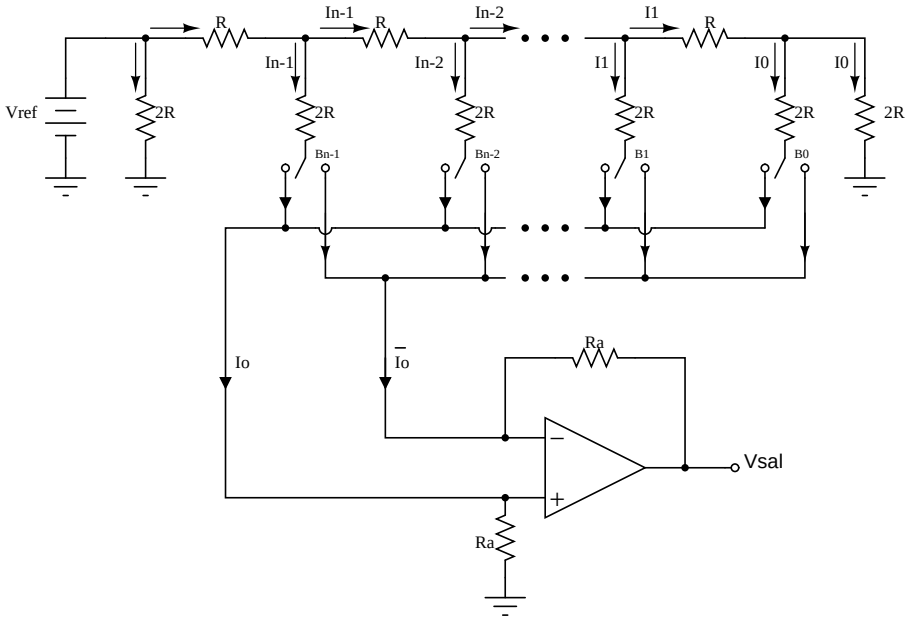


Figura 4.13: Circuito R-2R bipolar.

Error sobre escala completa:	± 1 LSB
Consumo de potencia:	33 mW a ± 5[V]
Bajo Costo	
Salidas de corriente complementarias.	
Interfaz directa con salidas TTL, CMOS, PMOS y otros.	

El diseño del circuito se basó en las características eléctricas del DAC y los requerimientos del circuito a la salida del controlador, es decir, que el voltaje a la salida esté dentro del intervalo de -10[V] a +10 [V]. Por otro lado, el voltaje de salida del microcontrolador es un valor binario de 256 estados, desde 0x00 hasta 0xFF en notación hexadecimal.

De esta manera, el DAC convertirá el voltaje digital a analógico de la siguiente manera:
0x00 - -10 [V]
0xFF - 10 [V]

El circuito convertidor digital a analógico se presenta en la figura 4.14. A continuación se explica cómo se llegó a dicha figura, haciendo un análisis del circuito.

El principio de funcionamiento del DAC0800 es el mismo que para el circuito en escalera de 8 interruptores. Así que la salida del DAC también será una corriente proporcional a la entrada en números binarios. Es por esa razón que a la salida, se coloca un amplificador operacional en su configuración de amplificador inversor, que convierte la corriente de salida

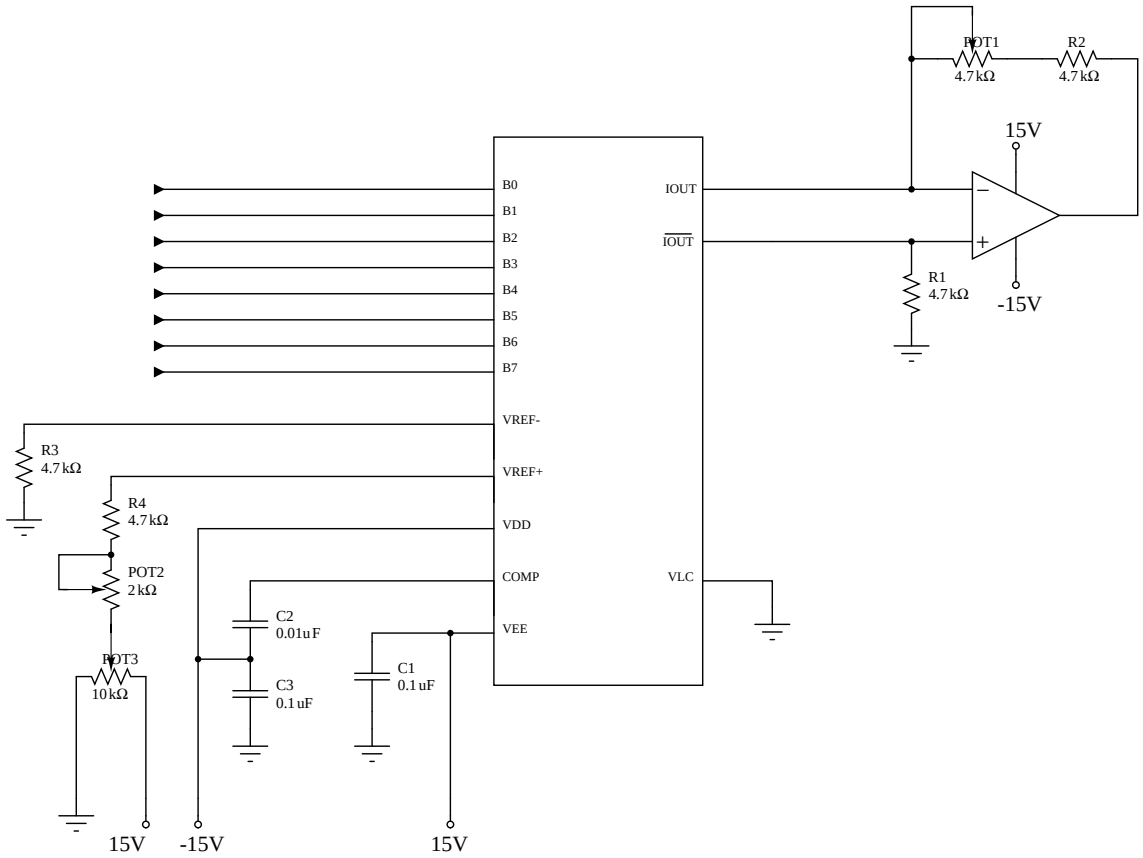


Figura 4.14: Circuito del convertidor digital a analógico.

en a un voltaje en el rango de -10[V] a +10[V].

La referencia del DAC0800 (V_{ref}, en el circuito R-2R) debe ser de 10 volts, el cuál se logra con el potenciómetro POT3. Para que la conversión se haga correctamente, es necesario que en la terminal V_{REF+} del DAC fluya una corriente de 2 [mA], es por esa razón que se colocan las resistencias R4 y POT2. La suma de estas resistencias debe ser de 5 [kΩ], para lograr este objetivo. Esta resistencia se ajusta con POT2.

Una de las características que debe tener el convertidor es que tiene que ser simétrico con respecto a la referencia del circuito, es decir, a tierra.

Se observa que en el amplificador de salida, se conecta una resistencia (R1) a la entrada no inversora y a tierra. El objetivo de esta resistencia es hacer que la salida del amplificador sea simétrica con respecto a la referencia (tierra).

Capítulo 5

SOFTWARE EJECUTABLE EN EL MICROCONTROLADOR ASOCIADO CON EL CONTROLADOR

5.1. Introducción

En este capítulo se explica, el *cerebro* de todo el proyecto de tesis: el circuito y el programa en el microcontrolador que resuelve la ecuación en diferencias que ejecuta el controlador escogido.

El controlador se elige desde una interfaz en la PC (esta interfaz se explica en el siguiente capítulo), misma que se comunica de forma serial al microcontrolador, pasándole los siguientes datos: un identificador para indicar el tipo de controlador y los parámetros de control que el usuario ha introducido. El microcontrolador, una vez que conoce esos datos, discretiza el controlador por medio del método de Tustin y resuelve la ecuación en diferencias correspondiente. Después, la salida (acción de control) se manda a través de un puerto del microcontrolador.

El algoritmo para resolver la ecuación en diferencias usado es muy sencillo; se basa únicamente en sumas y multiplicaciones que el microcontrolador es capaz de ejecutar con mucha precisión y con relativa facilidad. A este algoritmo se le conoce como *método de recursión para resolver ecuaciones en diferencias*.

A grandes rasgos, esa es la tarea que realiza el microcontrolador. En este capítulo, se explica con más detalle.

En la sección 5.2 se explican algunas de las características del microcontrolador 68HC908GP32[1]

[1]Aunque a lo largo de esta tesis se han hecho referencias y alusiones a dicho dispositivo, es en este

que se usaron en el desarrollo del programa.

En la sección 5.5 se explica, de forma detallada, el algoritmo usado para la resolución de la ecuación en diferencias asociado a los controladores básicos, a saber, *ON-OFF*, P, PI, PD y PID.

En este capítulo también se muestra el circuito usado para la comunicación entre la PC y el microcontrolador a través de una interfaz para convertir la lógica TTL a niveles RS-232. Eso se explica en la sección 5.3. De igual manera, en la sección 5.4 se muestra el circuito del cuál forma parte el microcontrolador.

El programa, como se observa en el apéndice A, fue hecho en lenguaje C ANSI. El compilador utilizado fue el ICC08, un producto de ImageCraft®, en su versión de evaluación.

La elección del lenguaje C se debe en parte a que el código del programa es más legible, además, el MCU68HC908GP32 está optimizado para este lenguaje. La desventaja de usar un lenguaje de alto nivel es que el uso de memoria de programa es mayor en comparación con un código hecho en ensamblador. Sin embargo, el microcontrolador escogido, tiene la suficiente cantidad de memoria para albergar la cantidad de código generado por un programa hecho en el lenguaje C.

Por último, es necesario mencionar que la forma de grabar el programa diseñado —el cuál se analiza en este capítulo— en el microcontrolador 68HC908GP32 fue por medio de un programa interfaz conocido como PUMMA08, el cuál fue desarrollado por el M. en I. Antonio Salvá Calleja en la Facultad de Ingeniería de la UNAM.

5.2. Arquitectura del microcontrolador 68HC908GP32

En la hoja de datos del microcontrolador 68HC908GP32 ([9]), se menciona que éste es un microcontrolador de 8 bits, con 32 kilobytes de memoria y varios módulos internos. Entre las características más importantes, se mencionan las siguientes:

- Arquitectura M68HC08 optimizada para compiladores C.
- Frecuencia de bus interno de 8 MHz
- Modos de operación de bajo consumo de potencia: *wait mode* y *stop mode*.
- 32 KBytes de memoria FLASH, con capacidad de programación *In-circuit*.
- 512 bytes de memoria RAM.
- Módulo Interfaz periférico serial (SPI).
- Módulo Interfaz de comunicación serial (SCI).
- 2 Módulos Interfaz de Temporización de 16 bits.

capítulo donde se detalla más ampliamente.

- 8 canales de conversión analógico digital de 8 bits por aproximaciones sucesivas.

En los siguientes apartados se estudian a detalle los módulos de interfaz de temporización, el módulo del convertidor analógico a digital y el módulo interfaz de comunicación serial, para ver como se configuraron dichos módulos, ya que fueron éstos los que se usaron en el desarrollo de este proyecto de tesis. Las abreviaturas y notaciones usadas en este capítulo son explicadas en [9].

5.2.1. Módulo interfaz de temporización

El microcontrolador 68HC908GP32 contiene un módulo para manejo de tiempos, conocido como Módulo Interfaz de Temporización (TIM, por sus siglas en inglés Timer Inteface Module).

El componente central de este módulo es un registro de 16 bits que puede actuar como un contador de corrida libre o como un módulo contador ascendente. En el primer caso, este registro proporciona el tiempo de referencia para las funciones de captura de entrada, salida de comparación y PWM. En el segundo caso, los registros TMODH y TMODL controlan, el valor del módulo contador del TIM.

Captura de entrada

Cuando el TIM se configura para funcionar como captura de entrada, uno de los pines del microcontrolador se configura como entrada de alguna señal digital, mientras que el contador del TIM se configura para que actúe como un registro de 16 bits de corrida libre. Cuando en el pin de entrada existe un cambio de flanco, ya sea positivo o negativo, dependiendo de la configuración, el contador se detiene. Con esta función se puede medir el tiempo en que sucede algún evento.

Salida de comparación

Cuando el TIM se configura para funcionar como salida de comparación, el contador inicia una cuenta de corrida libre. Cuando este contador alcanza el valor del registro de un canal de comparación destinado para ello, el TIM prende, apaga o intercambia el estado de uno de los pines del microcontrolador. Con esta función se puede hacer que un evento ocurra en un determinado tiempo. Generalmente este módulo es usado cuando el evento que se pretende controlar es externo.

Modulación por ancho de pulso (PWM)

Gracias al módulo interfaz de temporización, el microcontrolador puede generar una señal de modulación por ancho de pulso. Esta función es posible usando el módulo de comparación de salida junto con la característica de cambiar el estado de un pin cuando hay un desbordamiento en el registro de 16 bits.

El cambio de estado de un pin por desbordamiento determina el periodo de la señal PWM y la comparación de salida determina el ancho del pulso.

Módulo contador ascendente

Cuando el TIM se configura para funcionar como un módulo contador ascendente, el tiempo es manipulado por los registros TMODH y TMODL. EL valor de estos registros indica el valor final del temporizador. Esta forma de configurar el TIM es la forma más simple de las tres y su función es manejar los tiempos en software. Cuando hay un desbordamiento del registro de 16 bits (TMODH:TMODL[2]), esta forma de configuración puede provocar una interrupción por software.

Cuando el TIM se configura para funcionar como un contador ascendente se pretende generar un tiempo determinado. Para lograrlo, se coloca en los registros adyacentes TMODH y TMODL un valor que será el valor final del registro contador de 16 bits.

Una vez que el contador comienza a incrementar con la frecuencia configurada, ésta continúa así hasta que se alcanza el valor colocado en el registro TMODH:TMODL, después, el siguiente valor de la cuenta es 0x0000. Cuando no hay ningún valor escrito en el registro TMODH:TMODL, el contador incrementa hasta que se alcanza el valor máximo, es decir, 0xFFFF. Es necesario enfatizar que el desbordamiento del temporizador estará determinado por el registro TMODH:TMODL, en caso de que exista un valor escrito; y no del valor máximo que puede tomar el registro contador de 16 bits.

El tiempo logrado por el microcontrolador es, entonces, medido desde el valor de inicio (0x0000) hasta que ocurre el desbordamiento, es decir, cuando se alcanza el valor de TMODH:TMODL. Resulta evidente, que se pueden lograr distintos valores de tiempo, colocando distintos valores en el registro TMODH:TMODL.

Se ha dicho que los tiempos que maneja el MC68HC908GP32 está determinado por el registro contador de 16 bits del TIM. Este contador se incrementa de acuerdo a la frecuencia del bus interno del microcontrolador en una unidad. Sin embargo, es posible modificar esta frecuencia por medio del contenido de un registro conocido como *prescaler*. Dependiendo del contenido de este registro, el periodo de incremento del contador de 16 bits, será un múltiplo del periodo de reloj del bus interno. En total se pueden obtener 7 periodos distintos.

Para propósitos de este proyecto, el módulo interfaz de temporización fue configurado para que actuara como un módulo contador ascendente.

5.2.2. Módulo convertidor analógico a digital

El módulo de conversión analógico a digital que está incluido en el microcontrolador 68HC908GP32 tiene una resolución de 8 bits y un método de conversión conocido como *de*

[2]TMODH:TMODL es la notación de un registro de 16 bits, resultado de la yuxtaposición de los registros de 8 bits cada uno TMODH y TMODL

aproximaciones sucesivas. En total, este módulo incluye 8 canales de conversión los cuáles son seleccionables por medio de un multiplexor analógico interno. En este proyecto de tesis solamente se usó un canal.

El modo de conversión puede ser de dos tipos: modo continuo y modo simple. El modo simple solamente realiza una conversión y se detiene cuando se completa la conversión y coloca el valor binario en el registro correspondiente, en tanto que el modo continuo realiza la conversión continuamente y sigue así hasta que se le indique lo contrario. En este último modo, el dato se sobreescribirá aún cuando no se haya leído el valor binario.

Los voltajes de alimentación para el módulo convertidor, se toman de los pines V_{DDAD} y V_{SSAD}. Generalmente estos voltajes son tomados de los voltajes de alimentación del microcontrolador. Así que los valores de voltaje en los pines V_{DDAD} y V_{SSAD} serán 5 [V] y 0 [V] respectivamente. Internamente los voltajes de referencia del convertidor, V_{REFH} y V_{REFL}, son tomados de los pines V_{DDAD} y V_{SSAD}, respectivamente. Por lo anterior, cuando el voltaje de entrada es igual al voltaje en el pin V_{REFH}, el valor binario leído en el registro correspondiente es $0xFF$. Cuando el voltaje de entrada es igual al voltaje en el pin V_{REFL}, el valor leído es $0x00$. De esta manera, el voltaje de entrada al convertidor no puede exceder los voltajes V_{REFH} y V_{REFL}. En lo concerniente a este proyecto, para asegurar esto, se ideó el circuito de adecuación de entrada al microcontrolador que se explicó en el capítulo 4 de esta tesis.

La conversión inicia en el momento en que se escribe un dato en el registro de configuración del ADC y termina hasta que se haya completado y obtenido el valor equivalente binario de la señal analógica. Dependiendo del modo elegido, este proceso continuará después de hacer la primera conversión o no.

El tiempo de conversión está dada por la siguiente relación:

$$Tiempo_{conversion} = \frac{16\text{—}17 ciclos ADC}{frecuencia_{reloj} ADC}$$

Según la hoja de datos del fabricante([9]), la frecuencia de reloj del ADC, debe ser configurado a 1 [MHz] para el correcto funcionamiento de éste. De modo que, según la relación anterior, el tiempo de conversión es:

$$Tiempo_{conversion} = 16\text{—}17[\mu s]$$

Otro punto importante a aclarar es que el módulo de conversión analógico digital puede tomar la entrada de reloj de forma externa (cristal oscilador externo) o del bus interno para formar el reloj del ADC. Cualquiera que sea la fuente de reloj elegido, éste puede ser dividido entre 1, 2, 4, 8 o 16 para formar el reloj del ADC.

5.2.3. Módulo interfaz de comunicación serial (SCI)

El módulo de interfaz de comunicación serial del MCU68HC908GP32 es el encargado de configurar, enviar y recibir los datos en forma serial asíncrona a un dispositivo externo.

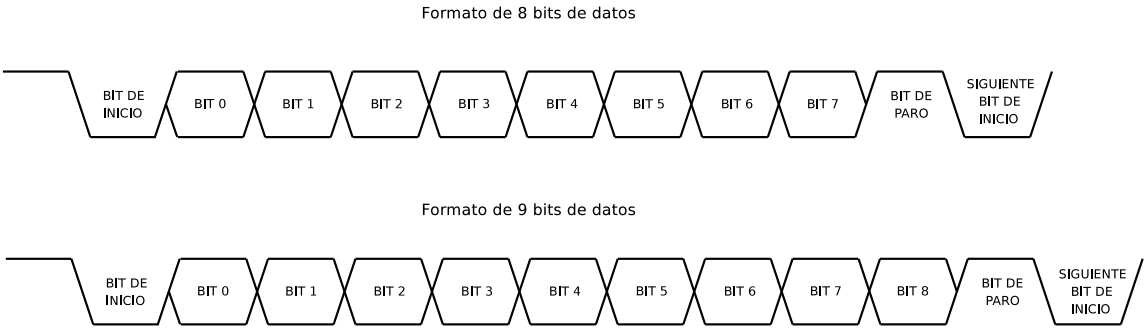


Figura 5.1: Formato SCI usando formato NRZ, con 8 y 9 bits de datos.

Las principales características del módulo SCI en el MCU68HC908GP32, son:

- Operación *full-duplex*, es decir, mandar y enviar datos al mismo tiempo.
- El formato utilizado es No Retorno a Cero (NRZ).
- 32 velocidades programables de envío y recepción.
- Longitud de caracter programable a 8 o 9 bits.

La transmisión y recepción de datos del MCU68HC908GP32 opera de forma independiente, aunque comparten el mismo registro de envío y recepción. La longitud de la palabra a recibir o transmitir puede ser de 8 o 9 bits.

En la figura 5.1 se muestra el formato de datos que usa el módulo SCI del microcontrolador 68HC908GP32, para una palabra de datos de 8 y 9 bits. A continuación se explica como se realiza la transmisión y la recepción de un caracter, usando el formato NRZ mostrado.

Transmisión de un caracter

Cuando un dato es escrito en el registro-*buffer* correspondiente a través del bus de datos, inmediatamente, se inicia la transmisión pasando la línea de transmisión del estado *ocioso*(estado alto) al estado bajo, que es el indicador del bit de inicio. Después, comienza la transmisión en forma serial comenzando por el bit menos significativo de la palabra a transmitir. En el MCU68HC908GP32 esta operación está implementada por *hardware*, así que todo lo realiza un registro de corrimiento (*shift register*) siendo esto transparente para el programador. Cada bit transmitido tiene una duración que depende de la velocidad de transmisión (mejor conocido como *baudaje*). Después de la transmisión de la palabra de 8 o 9 bits se transmite un bit de paro. Si el módulo interfaz de comunicación serial se configura para transmitir un bit de paridad, éste se transmite antes del bit de paro, ocupando el lugar

del último bit (más significativo). Cuando se han transmitido todos los bits, un bit-bandera o bit indicador cambia de estado para indicar que el módulo interfaz está listo para hacer la transmisión de una nueva palabra.

Recepción de un caracter

La recepción de un caracter en el módulo de interfaz de comunicación serial se realiza de forma análoga a la transmisión. La longitud de los datos puede ser de 8 o 9 bits. Existe un registro de corrimiento para la recepción, el cuál va recibiendo los bits y los va recorriendo una posición en la palabra hasta que la recibe por completo. El bit que se recibe primero es el de inicio (que es un estado bajo) después se recibe el bit menos significativo de la palabra que se está recibiendo. Para finalizar se recibe un bit de paro. Cuando esto ocurre, un bit-bandera o bit indicador cambia de estado para indicar que se ha recibido por completo la palabra y que el microcontrolador está listo para recibir una nueva palabra.

El microcontrolador incluye algunas características para la detección de errores en la recepción de los datos. Algo a subrayar en el módulo interfaz de comunicación serial asíncrona de este MCU, es que tanto en la recepción de los datos como en la transmisión se tiene que utilizar el mismo *baudaje*.

5.3. Circuito interfaz entre la PC y el microcontrolador 68HC908GP32

Como quedó dicho en la introducción de este capítulo, la comunicación entre la PC y el microcontrolador se realiza de forma serial. Las razones de esta elección, frente a la comunicación paralela, se debieron principalmente al número de pines usado, a la fiabilidad de los datos transmitidos y recibidos, y, sobre todo, a la disposición de los puertos en los dispositivos involucrados.

Debido a que los niveles de voltaje que maneja la PC es distinta a la manejada por el microcontrolador para la misma lógica binaria, surge la necesidad de disponer de una interfaz entre ambos dispositivos. Así, mientras el puerto serial de la PC maneja la norma RS-232 en los niveles de voltaje, ese mismo puerto en el microcontrolador maneja niveles TTL.

Para solucionar este problema, es necesaria la inclusión de un circuito entre la PC y el microcontrolador que convierta los distintos niveles de voltaje. Existen diversas opciones para lograr este objetivo. Desde los más sencillos como un juego de transistores hasta los más complejos, integrados dentro de un circuito. En el mercado existen diversas opciones en forma de circuito integrado. Por ejemplo, los circuitos MC1488 Y MC1489 de Motorola®. Estos circuitos fueron concebidos para lograr el entendimiento en la comunicación serial, de dos equipos que manejan distintos niveles de voltaje en su lógica. Sin embargo, estos circuitos precisan de una fuente de 12 volts y -12 volts para lograr la conversión de voltajes.

Esa es la principal desventaja de los mismos. Afortunadamente, también existen otros con menos requerimientos de alimentación. Ese es el caso del CI MAX232 que solamente precisa de una fuente de 5 volts para lograr su objetivo. Por esa razón, se escogió este último.

Antes de mostrar el circuito interfaz, conviene mostrar las características del estándar RS-232.

El estándar RS-232 cumple con los siguientes niveles de voltaje:

- Un 1 lógico es un voltaje que puede estar en el intervalo comprendido entre -3[V] y -15[V].
- Un 0 lógico es un voltaje que puede estar en el intervalo comprendido entre +3[V] y +15 [V].
- Los niveles de voltaje más comunes son: $\pm 5[V]$, $\pm 10[V]$, $\pm 12[V]$ y $\pm 15[V]$.

Como el puerto serie de una PC trabaja mediante el estándar RS-232, éste maneja niveles de voltaje que están dentro de los rangos mencionados anteriormente, a saber, -12[V] para el 1 lógico y 12[V] para el 0 lógico.

En la figura 5.2 se muestra el circuito interfaz que convierte los niveles de voltaje TTL a niveles RS-232 por medio del CI MAX232. El CI MAX232, como se dijo anteriormente, es un circuito que se caracteriza por no necesitar de fuentes dobles para hacer la conversión de voltajes. Solamente precisa de una fuente de voltaje de 5 volts unipolar.

Los capacitores usados en el circuito interfaz son electrolíticos con un valor de 1 [μF]. El valor de estos capacitores, es tal que, asegura el correcto funcionamiento del circuito interfaz según la hoja de datos del fabricante del CI MAX232.

Como se puede apreciar en la figura 5.2, las tierras son comunes y solamente se necesitan de 2 líneas de comunicación entre ambos dispositivos. Una de las líneas es para el envío de datos y el otro es para la recepción de los mismos. Esto es así porque la comunicación serial, en este caso, es asíncrona, por lo que no se necesita de una línea de reloj como sucede en la comunicación serial síncrona.

Para lograr que la comunicación se pueda realizar, aparte del circuito interfaz, es necesario configurar ambos dispositivos con los siguientes parámetros, propios de la comunicación serial asíncrona:

- Número de bits: pueden ser 5, 6, 7 u 8.
- Velocidad de transmisión (*baudaje*): Generalmente de 9600 bits/segundo.
- Bit de paridad.
- Bit de paro.

Evidentemente, estos valores deben ser los mismos en ambos dispositivos.

Para el proyecto en cuestión, el MCU 68HC908GP32 y la PC en donde reside la interfaz de usuario, se configuraron para que tuvieran las siguientes características:

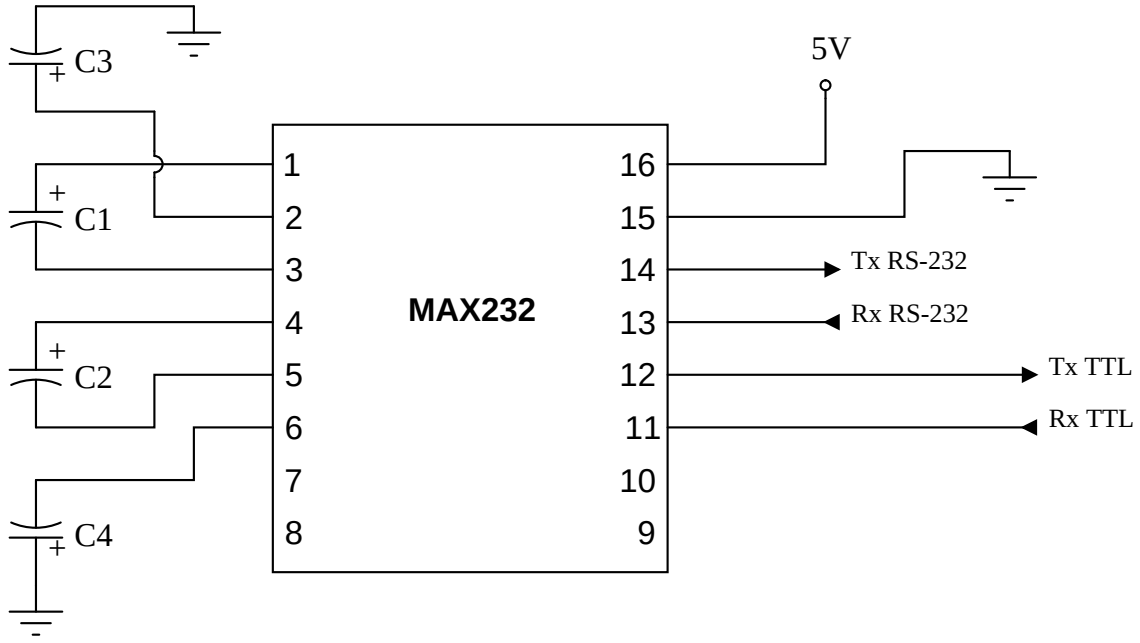


Figura 5.2: Circuito interfaz para comunicación serial entre PC y el microcontrolador.

- Número de bits: 8
- Velocidad de transmisión (*baudaje*): 9600 bits/segundo
- Bit de paridad: Ninguno.
- Bit de paro: No.

5.4. Circuito digital del microcontrolador

En esta sección se muestra el circuito del cuál forma parte el microcontrolador. Ese circuito se puede observar en la figura 5.3, misma que se describe a continuación.

El *Reset*

Como se puede observar en la figura 5.3, el circuito del *reset* se encuentra conectado al pin 6 del MCU 68HC908GP32. Este circuito es una red RC junto con un interruptor tipo *push-botton*. Cuando se presiona este interruptor, el circuito sufre un reinicio, y en el momento en que se suelta, el contador de programa del microcontrolador se coloca en la dirección de inicio.

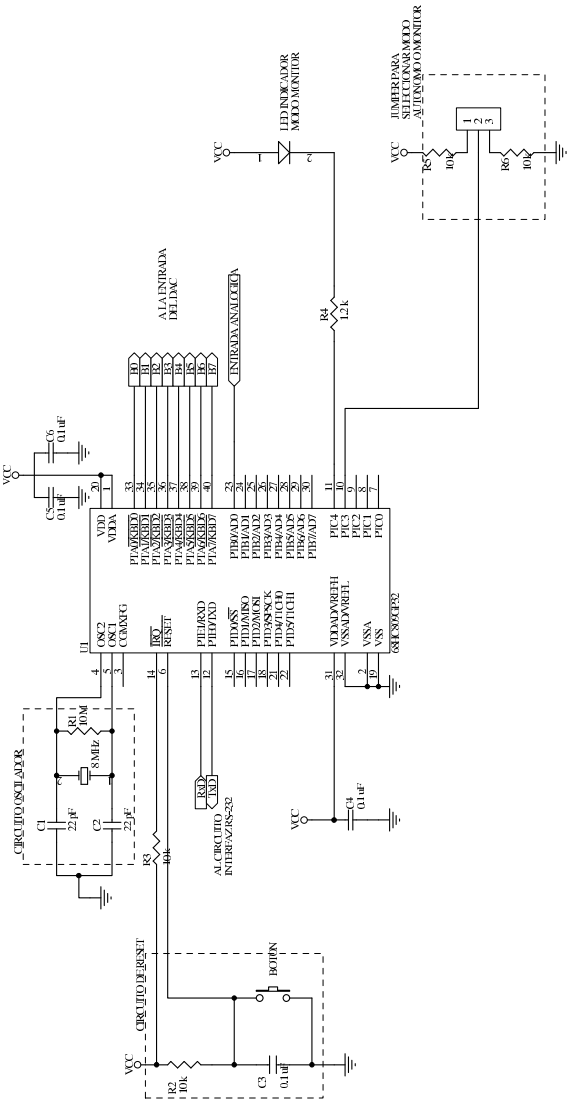


Figura 5.3: Circuito del microcontrolador.

El circuito RC tiene la función de evitar el fenómeno conocido como *rebote*. Este fenómeno se presenta debido a que se trata de un interruptor mecánico y consiste en que cuando se presiona el interruptor, en realidad se registran varios contactos en lugar de uno. Aunque este fenómeno puede ser evitado también por software, en este proyecto se ha preferido usar este circuito con el objeto de evitar complicaciones innecesarias en el programa.

El circuito oscilador

El circuito oscilador consiste en un cristal de cuarzo con un par de capacitores conectados a referencia (tierra). Este cristal oscila a una frecuencia de 8 [MHz] produciendo un periodo de reloj de 125 [nanosegundos].

Puertos de Entrada y Salida

De acuerdo a la figura 4.1 del capítulo 4, el microcontrolador procesa la señal de error, y manda la salida a un puerto. En la figura 5.3, la entrada es el error que existe en la salida del circuito y la señal de referencia, esta entrada se conecta al módulo de conversión analógico digital del microcontrolador por el pin 23 (puerto B). La salida, se manda al puerto A, misma que está conectada al circuito de conversión digital a analógico que fue explicado en el capítulo 4.

En el pin 10 (puerto C) se conecta un *jumper*. Este *jumper* permite elegir el modo de funcionamiento del microcontrolador: modo monitor o modo autónomo, dependiendo de si está conectado a tierra (referencia) o a V_{DD}, respectivamente. Aunque para este dispositivo, el usuario solo tendrá que usar el modo autónomo porque no tendrá necesidad de modificar el programa que estará corriendo dentro del microcontrolador.

En el pin 11 (puerto C) se conecta un LED indicador. Este LED parpadeará en caso de que el microcontrolador se encuentre en modo monitor.

El pin 14 se conecta al nivel V_{DD}, por medio de una resistencia de *pull-up*. Este pin sirve para la entrada de interrupciones externas, aunque en este proyecto no se necesitaron de éstas.

Los pines 12 y 13 sirven para la comunicación serial asíncrona. Estos pines se conectan al circuito interfaz que fue explicada en la sección 5.3.

Es importante mencionar que en los pines de alimentación (tanto del microcontrolador en general y del módulo de conversión analógico digital en particular) se conectaron capacitores en paralelo. Estos capacitores se conocen como *capacitores de desacoplamiento*[3] y tienen la función de evitar que el ruido producido por una parte del circuito no sea transmitida a otra parte del mismo. En este caso, se trata de evitar que el ruido producido en la fuente sea propagada hacia los pines de alimentación del microcontrolador, causando con ello un comportamiento errático del mismo.

[3]En la literatura, se conocen también como capacitores de *bypass*.

En la placa del circuito impreso, los capacitores de desacoplamiento van lo más cerca posible de los pines de alimentación del microcontrolador para que puedan cumplir su función.

5.5. Diagramas de flujo

En esta sección se explica el algoritmo del programa usado que se ejecuta en el microcontrolador para la implementación del controlador digital. En primer lugar se muestra el diagrama de flujo general del programa para, posteriormente, detallar cada bloque. El código completo del programa se anexa en el apéndice A.

En la figura 5.4 se muestra el diagrama de flujo general del programa que ejecuta el microcontrolador. Se pueden observar los bloques principales del diagrama de flujo, los cuáles son: Configuración del microcontrolador y la configuración del controlador elegido. Estos dos bloques se explican en las siguientes subsecciones. Los demás bloques tienen la función indicada dentro de ellas, de modo tal que la explicación del programa sería la siguiente:

1. El microcontrolador es encendido y, automáticamente, el contador de programa se coloca en el vector de *RESET*, el cuál contiene la dirección de inicio del programa principal.

2. Se configuran los registros apropiados del microcontrolador.

3. El microcontrolador se queda en la espera de un dato en el puerto serial.

4. En el momento de encontrar un dato en el puerto serial, se deshabilita la interrupción por temporizador[4], éste se debe deshabilitar, para que no ocurra la interrupción en el momento en que se estén leyendo los datos y/o en el momento de la configuración del controlador elegido.

5. El microcontrolador comienza a leer los datos enviados por el puerto serial que el usuario ha indicado por medio de una interfaz en la PC. Estos datos son el tipo de controlador, los parámetros de dicho controlador y el periodo de muestreo.

6. Cuando ha terminado el proceso anterior nuevamente se habilita el *timer*, y el microcontrolador queda otra vez a la espera de que algún dato sea puesto en el puerto serial, el cuál ocurrirá cuando el usuario decida interrumpir el proceso o cambiar el tipo de controlador.

[4]Evidentemente, cuando ocurre esto por primera vez, aún no se ha habilitado dicha interrupción, y este bloque carece de razón de estar ahí.

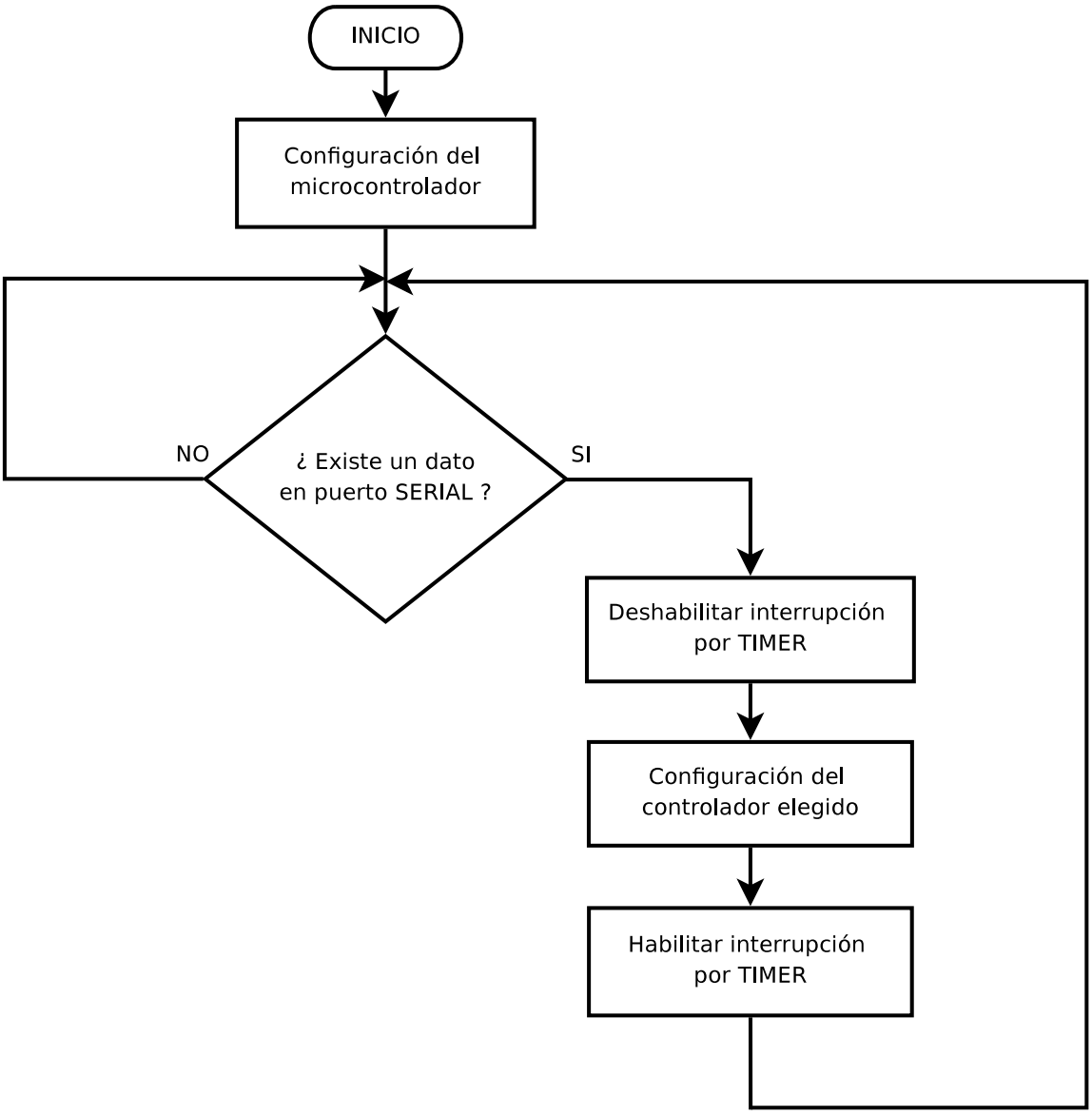


Figura 5.4: Diagrama de flujo general.

5.5.1. Configuración del microcontrolador

En el bloque marcado en la figura 5.4, con la etiqueta *configuración del microcontrolador* se hicieron las modificaciones necesarias para que el microcontrolador ejecutara correctamente el programa. En dicho bloque se configura el convertidor analógico digital y la interrupción por *overflow* o desbordamiento del temporizador. De igual manera, se declaran las variables utilizadas en el programa. En ese bloque también se configura la comunicación de forma serial, indicando la velocidad de transmisión, el número de bits y el tipo de transmisión. Por último, se le indica al microcontrolador que la ejecución del programa será de forma autónoma.

Configuración del convertidor analógico a digital

Para este proyecto de tesis, la configuración del convertidor analógico digital, se hizo de la siguiente manera:

- Utilizar el canal 0 del convertidor.
- Usar un reloj de conversión con una frecuencia de 1 [MHz] tomando como fuente de reloj un cristal resonador externo de 8 [MHz].
- Usar el modo continuo de conversión.

Configuración del temporizador

Para poder obtener una interrupción cada 10 milisegundos, que es el tiempo base utilizado, se configuró el temporizador TIMER1 de tal manera que el mismo sufriera un desbordamiento en dicho tiempo. Además, que el mencionado desbordamiento produjera una interrupción. Para eso se configuró el módulo de interfaz de temporización en su modo contador ascendente. Cada incremento de cuenta se realiza en cada ciclo máquina o un múltiplo de éste dependiendo de la configuración del registro *prescaler*. Por otro lado, cada ciclo máquina o ciclo de CPU para el MCU68HC908GP32 es igual a 4 ciclos de reloj.

Como se utiliza un cristal resonador con una frecuencia de 8 MHz, cada incremento de cuenta, con un *prescaler* de 1, se realiza en:

$$t_{CPU} = 4 * t_{RELOJ} = 4 * \frac{1}{8[MHz]}$$

Por lo que cada ciclo de CPU es de 500 nanosegundos.

Para lograr que cada 10 milisegundos se genere una interrupción, se necesitan 20,000 incrementos de cuenta del temporizador, ya que:

$$num_{incrementos} = \frac{10[ms]}{500[ns]} = 20,000$$

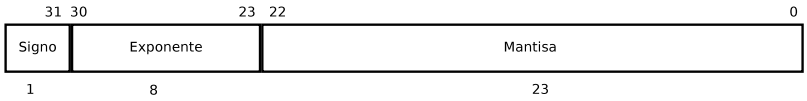


Figura 5.5: Formato para la representación de un número en punto flotante de simple precisión según el estándar IEEE 754.

Por lo explicado en la sección 5.2.1, en los registros TMODH y TMODL se deberá colocar el número 20,000 en notación hexadecimal para que el temporizador incremente su cuenta desde 0 hasta 20,000 y cuando alcance este número se produzca un desbordamiento y, por consiguiente, una interrupción.

También, en el registro correspondiente se deberá indicar que el *prescaler* a utilizar, es de 1, y, por último, se deberán habilitar las interrupciones por módulo de temporización y las interrupciones en general.

En este proyecto, solamente se necesitó del uso del temporizador 1.

5.5.2. Configuración del controlador elegido

En esta sección se explica la forma de configuración del controlador elegido. Sin embargo, se inicia con la explicación de cómo se logran pasar los parámetros de dicho controlador, desde la PC al microcontrolador. Este proceso se realiza cada vez que se hace una configuración, antes del inicio del proceso de control; es por esa razón que se explica en este apartado.

Como los parámetros de control generalmente son números no enteros, es decir, números con parte decimal, la transmisión de los datos no se hace de forma directa como sucede con un caracter alfanumérico. En estos últimos, solo se necesitan transmitir 7 u 8 bits por cada caracter, mismos que pueden caber en un solo registro del microcontrolador y tienen una representación en código ASCII. Sin embargo, los números decimales precisan de una mayor cantidad de bits (o de bytes) para ser representados e interpretados correctamente. Esa representación es conocida como *representación en punto flotante*.

El estándar IEEE-754 define el formato, el número de bits, la precisión y un conjunto de operaciones sobre números de punto flotante para representar los números decimales en las computadoras. Según este estándar, un número de punto flotante de simple precisión usa 4 bytes, en los cuáles, la distribución de bits se muestra en la figura 5.5. En este proyecto se usó este estándar.

El proceso de envío de los números flotantes desde la PC y la recepción en el microcontrolador es como sigue:

La PC envía el número flotante en formato de 4 bytes, un byte detrás de otro; el microcontrolador recibe cada uno de los bytes y los coloca en cuatro registros de memoria contiguos.

En el programa que se ejecuta dentro del microcontrolador, cuando éste tiene que leer el número en punto flotante, únicamente necesita la dirección de la primera localidad de memoria de los 4 bytes donde se guardó el número recibido. La identificación del número se hace correctamente debido a que el microcontrolador interpreta el dato de acuerdo al formato IEEE-754. Aquí es necesario subrayar, que esta característica es posible gracias al compilador usado.

Una vez que el microcontrolador ha recibido los datos (parámetros de control en punto flotante, periodo de muestreo y tipo de controlador), se inicia la configuración en sí del controlador elegido. La configuración del controlador elegido se refiere al proceso que se sigue para que el algoritmo de control pueda realizarse con éxito. Por lo tanto, en la configuración del controlador se tienen que hallar los coeficientes de la ecuación en diferencias asociado para algunos tipos de controladores y el periodo de muestreo requerido. Evidentemente, en el caso de que el controlador elegido sea *ON-OFF* o Proporcional no existe una ecuación en diferencias por lo que su configuración es distinta.

Antes de pasar a las subsecciones donde se explica la configuración de cada controlador, es importante explicar la forma de hallar los coeficientes de la función de transferencia asociada a dicho controlador.

Como quedó dicho en el capítulo 3, existen diversas técnicas y métodos para discretizar un controlador analógico. En este proyecto se optó por utilizar el método del trapecio, mejor conocido como *método de Tustin*. La elección de este método se debió principalmente a que éste es más fiel al controlador analógico del cual se deriva. De acuerdo a este método, la relación entre un controlador digital y su contraparte analógica está dada por:

$$H(z) = G_C(s)\Big|_{s=\frac{2}{T}\left(\frac{z-1}{z+1}\right)}$$

En el capítulo 3 se explicó como se dedujo esta ecuación. El significado práctico de éste es que para encontrar el modelo matemático de la contraparte digital de los controladores básicos vistos en el capítulo 2, bastará con sustituir la variable s por la expresión $\frac{2}{T}\left(\frac{z-1}{z+1}\right)$.

Al hacer la sustitución anterior se obtiene la función de transferencia discreta del controlador. En esta función de transferencia se tienen los coeficientes de la ecuación en diferencias asociado a dicho controlador.

Por otro lado, el diseño de un controlador analógico, se hace por medio de sus parámetros, a saber, K_C, T_i y T_d dependiendo de su tipo. Cuando se hace la sustitución referida anteriormente, se obtiene una expresión en términos de la variable z cuyos coeficientes están, a su vez, en términos de los parámetros K_C, T_i y T_d. De esta manera, cuando el microcontrolador resuelve la ecuación en diferencias asociado, cuyos coeficientes están en términos de K_C, T_i y T_d, se logra que el controlador discreto obtenido simule el comportamiento de un controlador analógico.

Dependiendo del tipo de controlador, la ecuación en diferencias será de primero o de segundo orden; y, por consiguiente, la función de transferencia discreta tendrá un polinomio característico de primer o segundo grado, respectivamente. De esta manera, las funciones de transferencia tendrán las siguientes formas:

Primer orden:

$$H(z) = \frac{b_0 z + b_1}{z - a_1} \tag{5.1}$$

Segundo orden:

$$H(z) = \frac{b_0 z^2 + b_1 z + b_2}{z^2 - a_1 z - a_2} \tag{5.2}$$

Se hace notar que las ecuaciones en diferencias están normalizadas, es decir, el coeficiente de la variable de mayor orden del polinomio característico es la unidad ($a_0 = 1$).

En resumen, una vez que se tiene la función de transferencia discreta de un controlador, implícitamente se tendrá su ecuación en diferencias, que el microcontrolador (o cualquier otra computadora digital) podrá resolver sin mayor problema, haciendo lo cuál, estará reproduciendo el comportamiento del controlador analógico.

Una vez analizada la forma de hallar los coeficientes de las ecuaciones en diferencias, a continuación se explicarán, la configuración de los distintos controladores.

Controlador *ON-OFF*

El controlador *ON-OFF* es el más sencillo de todos, como quedó explicado en el capítulo 2. La configuración de este controlador es recibir el periodo de muestreo T, la amplitud del voltaje de salida y el tamaño de la ventana de histéresis en volts.

La amplitud del voltaje de salida puede estar en el intervalo de -10 [V] a +10 [V] volts y será el voltaje que presente el controlador en uno de los estados (estado ALTO); el otro estado es de -10 volts (estado BAJO).

La ventana de histéresis está en el intervalo de 0 [V] a 10 [V].

Controlador P

El controlador P no es dinámico, es decir, la salida no depende de los estados anteriores (entradas anteriores y salidas anteriores) por lo que su modelo matemático no está dado en términos de ecuaciones diferenciales.

Debido a esto, tampoco existe una ecuación en diferencias asociado al controlador P.

Por lo tanto, la configuración del controlador P es únicamente recibir el parámetro de control K_C y el periodo de muestreo T.

Controlador PI

La configuración del controlador PI consiste en hallar los coeficientes de su ecuación en diferencias asociado en función de las parámetros de control K_C y T_i, además del periodo de muestreo T. Para hallar los coeficientes, se deberá evaluar la relación $s = \frac{2}{T}\left(\frac{z-1}{z+1}\right)$ en la función de transferencia analógica de este controlador, es decir,

$$H(z)_{PI} = G_{C_{PI}}(s)\Big|_{s=\frac{2}{T}\left(\frac{z-1}{z+1}\right)}$$

Por lo tanto:

$$H(z)_{PI} = K_c\left(1+\frac{1}{T_i s}\right)\Big|_{s=\frac{2}{T}\left(\frac{z-1}{z+1}\right)} \tag{5.3}$$

Como la función de transferencia de este controlador es de primer orden, la ecuación en diferencias resultante será también de primer orden.

$$H(z)_{PI} = K_c\left(1+\frac{1}{T_i\left(\frac{2}{T}\left(\frac{z-1}{z+1}\right)\right)}\right)$$

Al hacer las operaciones necesarias, se tiene que:

$$H(z)_{PI} = \frac{\left[K_c\left(\frac{T+2T_i}{2T_i}\right)\right]z + \left[K_c\left(\frac{T-2T_i}{2T_i}\right)\right]}{z-1}$$

De donde se concluye que los coeficientes de la ecuación en diferencias son:

$$\begin{aligned} a_1 &= 1 \\ b_0 &= K_c\left(\tfrac{T+2T_i}{2T_i}\right) \\ b_1 &= K_c\left(\tfrac{T-2T_i}{2T_i}\right) \end{aligned}$$

Obsérvese que $a_0 = 1$ debido a que se supone que la ecuación en diferencias está normalizada.

Volviendo al bloque de configuración del controlador elegido, en el caso de que el usuario elija simular el controlador PI, entonces, éste deberá introducir los parámetros de control, a saber K_C y T_i de acuerdo a su diseño, además del periodo de muestreo T; y el microcontrolador se encargará de calcular los coeficientes de la ecuación en diferencias asociado al controlador de acuerdo a las funciones encontradas anteriormente.

Para el controlador PD

Al igual que para el controlador PI, la configuración del controlador PD, consiste en hallar los coeficientes de la ecuación en diferencias asociado a la función de transferencia

analógica. Esta vez, dichos coeficientes estarán en términos de K_C, T_d y el periodo T. Para hallar estos coeficientes, se procederá de la misma forma que para el controlador PI:

$$H(z)_{PD} = G_{C_{PD}}(s)\Big|_{s=\frac{2}{T}\left(\frac{z-1}{z+1}\right)}$$

Por lo tanto:

$$H(z)_{PD} = K_c\left(1 + T_d s\right)\Big|_{s=\frac{2}{T}\left(\frac{z-1}{z+1}\right)} \tag{5.4}$$

Dado que la función de transferencia del controlador PD es de primer orden, la ecuación en diferencias resultante será de primer orden:

$$H(z)_{PD} = K_c\left(1 + T_d\left[\frac{2}{T}\left(\frac{z-1}{z+1}\right)\right]\right)$$

Al hacer las operaciones necesarias, se tiene que:

$$H(z)_{PD} = \frac{\left[K_c\left(\frac{T+2T_d}{T}\right)\right]z + \left[K_c\left(\frac{T-2T_d}{T}\right)\right]}{z+1}$$

De donde se concluye que los coeficientes de la ecuación en diferencias son:

$$\begin{aligned} a_1 &= -1 \\ b_0 &= K_c\left(\tfrac{T+2T_d}{T}\right) \\ b_1 &= K_c\left(\tfrac{T-2T_d}{T}\right) \end{aligned}$$

Para el controlador PID

La configuración del controlador PID consiste, nuevamente, en hallar los coeficientes de la ecuación en diferencias asociado en términos de los parámetros K_C, T_i y T_d, además del periodo de muestreo T. Para lograr esto, se deberá evaluar la función de transferencia de este controlador en $s = \frac{2}{T}\left(\frac{z-1}{z+1}\right)$.

Matemáticamente:

$$H(z)_{PID} = G(s)_{PID}\Big|_{s=\frac{2}{T}\left(\frac{z-1}{z+1}\right)}$$

$$H(z)_{PID} = K_c\left(1 + \frac{1}{T_i s} + T_d s\right)\Big|_{s=\frac{2}{T}\left(\frac{z-1}{z+1}\right)}$$

$$H(z)_{PID} = K_c\left(1 + \frac{1}{T_i\left(\frac{2}{T}\left(\frac{z-1}{z+1}\right)\right)} + T_d\left(\frac{2}{T}\left(\frac{z-1}{z+1}\right)\right)\right)$$

Al hacer las operaciones necesarias, se tiene que:

$$H(z)_{PID} = \frac{K_c\left(\frac{4T_iT_d+2T_iT+T^2}{2TT_i}\right)z^2 + K_c\left(\frac{T^2-4T_iT_d}{T_iT}\right)z + K_c\left(\frac{4T_iT_d-2T_iT+T^2}{2TT_i}\right)}{z^2-1} \tag{5.5}$$

De donde se concluye que los coeficientes de la ecuación en diferencias son:

$$\begin{aligned} a_1 &= 0 \\ a_2 &= 1 \\ b_0 &= K_c\left(\frac{4T_iT_d+2T_iT+T^2}{2TT_i}\right) \\ b_1 &= K_c\left(\frac{T^2-4T_iT_d}{T_iT}\right) \\ b_2 &= K_c\left(\frac{4T_iT_d-2T_iT+T^2}{2TT_i}\right) \end{aligned}$$

Aquí se recalca también que $a_0 = 1$, debido a que se supone que la ecuación en diferencias está normalizada.

Una vez que se tienen los coeficientes, se podrá resolver la ecuación en diferencias asociado al controlador PID.

En la tabla 5.1, se coloca el resumen de los valores de los coeficientes de la ecuación en diferencias asociado a los distintos controladores. Estas relaciones son las que usa el microcontrolador para calcular los coeficientes de la ecuación en diferencias asociado al controlador elegido.

5.5.3. Solución de la ecuación en diferencias asociado al controlador elegido. Subrutina de servicio de interrupción

Cuando se desea simular el comportamiento de un controlador analógico por medio de un microcontrolador (o cualquier otra computadora digital), se tiene que resolver su ecuación en diferencias asociada [5]; es decir, en un momento determinado, se tiene que calcular la salida a partir de la entrada en ese momento y de las entradas y salidas anteriores. Este cálculo se hace cada T segundos (cada intervalo de periodo de muestreo) de manera indefinida. Además, el cálculo se hace en tiempo real, es decir, se tiene que procesar la entrada actual antes de que aparezca la siguiente entrada. Esto significa que la duración del tiempo T debe ser tal que se puedan resolver todas las operaciones involucradas en ese tiempo.

Para generar el tiempo de cada periodo de muestreo T se usó la característica del microcontrolador conocida como *interrupción por desbordamiento del temporizador*. Su explicación y configuración fue abordada en la sección anterior. Entonces, el cálculo de la

[5] Se aclara que cuando el controlador no es dinámico, no posee ecuación en diferencias asociada. En su lugar tendrá otro tipo de modelo matemático.

Tipo de Controlador	Coeficientes
PI	$a_1 = 1$ $b_0 = K_c\left(\frac{T+2T_i}{2T_i}\right)$ $b_1 = K_c\left(\frac{T-2T_i}{2T_i}\right)$
PD	$a_1 = -1$ $b_0 = K_c\left(\frac{T+2T_d}{T}\right)$ $b_1 = K_c\left(\frac{T-2T_d}{T}\right)$
PID	$a_1 = 0$ $a_2 = 1$ $b_0 = K_c\left(\frac{4T_iT_d+2T_iT+T^2}{2TT_i}\right)$ $b_1 = K_c\left(\frac{T^2-4T_iT_d}{T_iT}\right)$ $b_2 = K_c\left(\frac{4T_iT_d-2T_iT+T^2}{2TT_i}\right)$

Cuadro 5.1: Coeficientes de la ecuación en diferencias asociado a los controladores digitales. Se usó el método de Tustin

salida en un tiempo determinado se encuentra dentro de una subrutina de servicio de interrupción.

La interrupción se hace cada 10 milisegundos debido a que es el mínimo tiempo que el microcontrolador necesita para resolver una ecuación en diferencias de segundo orden[6]. El tiempo necesario para resolver uno de primer orden bajo las mismas condiciones, es de 6.5 milisegundos.

En la figura 5.6 se muestra el diagrama de flujo de la rutina de servicio de interrupción. Esta rutina se ejecuta cada vez que ocurre una interrupción por desbordamiento del temporizador, es decir, cada 10 milisegundos.

El diagrama de flujo presentado en la figura 5.6 muestra de forma generalizada lo que ocurre en la subrutina de servicio de interrupción, que puede desglosarse de la siguiente manera:

1. Verifica si se ha cumplido con el periodo de muestreo, de ser así continúa en el siguiente paso. De lo contrario incrementa el valor de una variable y se sale de la subrutina.

2. Verifica cuál es el tipo de control requerido.

3. Toma una muestra de la entrada analógica, es decir, del error.

4. A partir de la muestra, resuelve la ecuación en diferencias asociado con el tipo de controlador requerido.

5. Manda la solución de la ecuación en diferencias a un puerto.

6. Fin de la rutina de servicio de interrupción.

Los pasos 3, 4 y 5 se encuentran englobados dentro del bloque etiquetado con “Resolver Controlador”. A continuación se explica de forma detallada cada bloque.

Verificación del periodo de muestreo y del controlador requerido

Antes de hacer la acción de control se tiene que verificar si el periodo de muestreo corresponde al pedido mediante la interfaz. Para lograr esto, se involucra una variable que lleva el control de cuantas veces se ha ejecutado la interrupción. Los valores válidos para el periodo de muestreo son los múltiplos de 10 milisegundos. Entonces, para verificar si el tiempo transcurrido es el periodo de muestreo, se compara el valor de una variable, que indica cuantas veces ha ocurrido la interrupción, con el número de veces que tiene que suceder la misma. Cuando ambos valores son iguales, significa que se ha alcanzado el periodo de muestreo y se debe ejecutar la acción de control. En caso contrario, se concluye

[6]Con un cristal de 8 MHz.

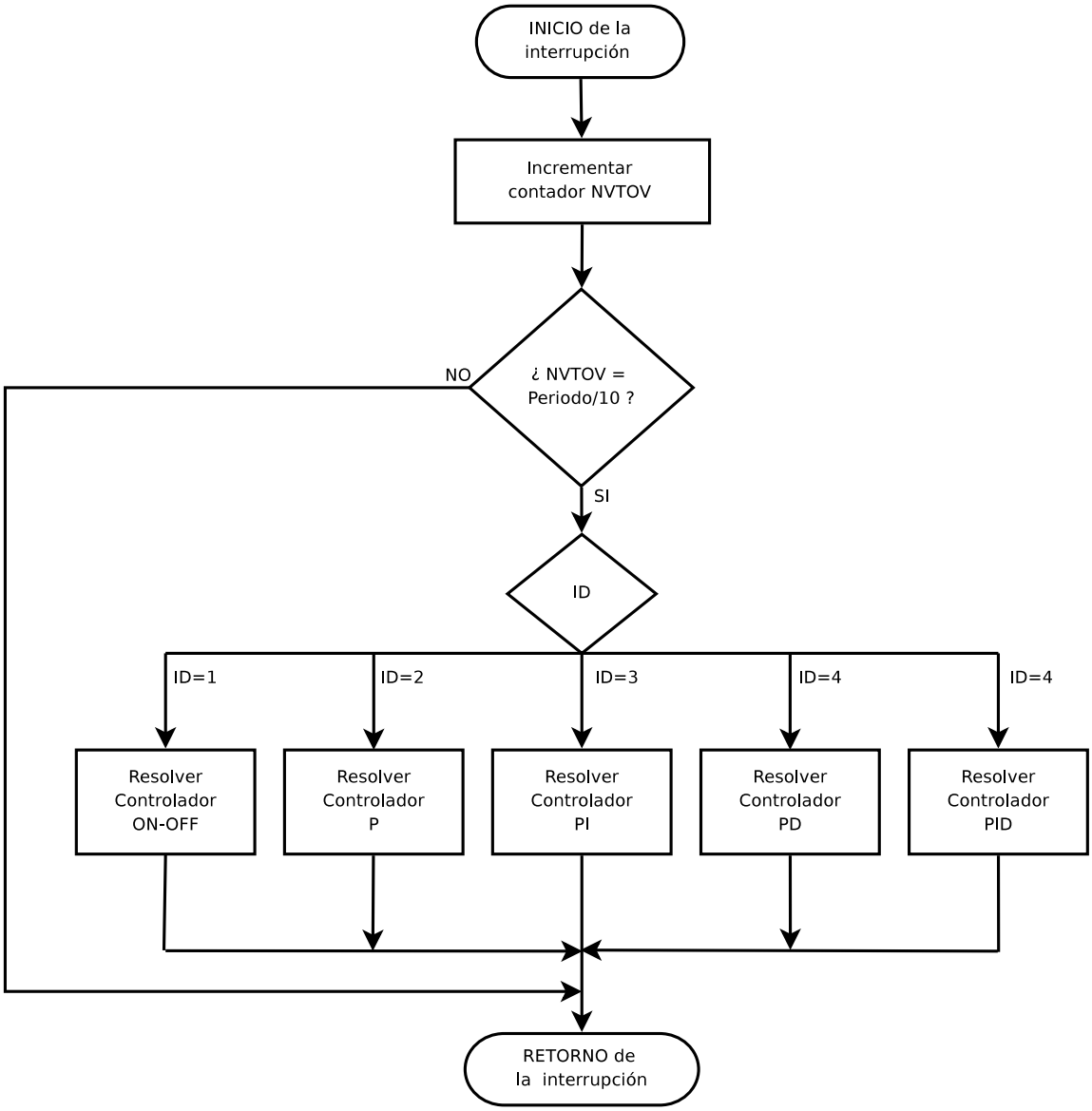


Figura 5.6: Diagrama de flujo de la rutina de servicio de interrupción.

que no ha transcurrido el tiempo necesario para alcanzar el periodo de muestreo definido y la subrutina de interrupción termina, en espera de la próxima vez que ocurra.

Cuando se ha cumplido este tiempo; es decir, cuando se ha alcanzado el periodo de muestreo, se inicia la verificación del tipo de controlador a simular. Este proceso involucra una variable que contiene un identificador para cada controlador. Por lo tanto, dentro de una condicional, se verifica qué tipo de controlador básico se desea simular por medio de ese identificador.

Una vez que se conocen estos datos (el tiempo de muestreo y el tipo de controlador) se inicia la acción de control, que es la parte medular de la subrutina de servicio de interrupción y cuyos detalles se explican a continuación.

Como se ha insistido anteriormente, la simulación de un controlador se hace por medio de su ecuación en diferencias asociada. Esta ecuación está en términos de la señal de entrada (entrada presente y entradas anteriores) y del estado anterior (salidas anteriores). Para este caso particular, la señal de entrada es la señal de error y la señal de salida es la señal de control. Resulta claro que la acción de control (salida de la ecuación en diferencias) está en función del error.

Para los controladores que no son dinámicos, es decir, que no pueden modelarse por una ecuación diferencial, tampoco tendrán un modelo matemático representado por una ecuación en diferencias. En esos casos la salida estará dada únicamente por la señal de error presente, tal como sucede en los controladores P y *ON-OFF*.

En lo que resta de esta sección y de este capítulo se explicará el algoritmo seguido para la resolución de la ecuación en diferencias asociado a los distintos controladores. Para aquellos controladores que no presentan una ecuación en diferencias asociado, se presentará el algoritmo para resolver su modelo matemático correspondiente.

Algoritmo para las acciones de control

Antes de explicar el algoritmo, para una mejor comprensión, enseguida se colocan los nombres de las variables manejadas con sus respectivos significados:

EP	Error presente
EANT	Error anterior
EANT2	Error antes del anterior
MP	Salida presente
MANT	Salida anterior
MANT2	Salida antes de la anterior
BL	Byte leído del convertidor analógico digital
BYSAL	Byte de salida que se envía a un puerto del MCU

El algoritmo general es el siguiente:

El microcontrolador toma la señal de error actual o presente por medio de uno de sus puerto. Esa señal es analógica, por lo que se convierte en una señal digital por medio del módulo ADC del MCU68HC908GP32. Por todo lo descrito en el capítulo 4 y en la sección referente al módulo de conversión analógico digital, los valores que puede tomar el error es de $0x00$ a $0xFF$ para los voltajes de -10 [V] y +10 [V], respectivamente. Debido a que en la subrutina de servicio de interrupción se manejan valores decimales, se tiene que hacer la conversión de hexadecimal (valor leído del registro ADC) a decimal, por medio de la siguiente relación lineal:

$$EP = 0.07843 * BL - 10.0 \tag{5.6}$$

El microcontrolador calcula la salida de la acción de control correspondiente, por medio de su modelo matemático. La salida de la acción de control (un número decimal) se tiene que mandar a un puerto, sin embargo antes se tiene que hacer una conversión de decimal a hexadecimal para que la acción de control pueda ser interpretada correctamente; por medio de la siguiente relación lineal, el cuál convierte el rango de voltajes -10 [volts] a +10 [volts] al rango hexadecimal $0x00$ a $0xFF$:

$$BYSAL = 12.75 * (MP + 10) \tag{5.7}$$

Después de hacer esto, el microcontrolador se queda en la espera de otra interrupción por desbordamiento del temporizador, para repetir el proceso anterior de manera indefinida.

Con todo esto, se estará simulando el comportamiento del controlador analógico por medio de un microcontrolador.

Lo anterior, es el algoritmo genérico seguido por los controladores simulados. Sin embargo, de manera más específica, se explican en las siguientes subsecciones los algoritmos seguidos por cada controlador en particular.

Control *ON-OFF*

La rutina para el control *ON-OFF* contiene el código para un control de dicho tipo.

Este es el control más simple de los cinco programados. Consiste en la comparación del valor de entrada (el error) apropiadamente escalado contra el tamaño de la ventana configurado por medio de el programa interfaz que se ejecuta en la PC conectado al dispositivo en cuestión.

Si este valor excede el umbral alto de la histéresis (ventana/2), entonces la salida generada es un 0 en el convertidor Digital a Analógico. Si el valor de entrada es menor que el umbral bajo de la histéresis (-Ventana/2), entonces la salida es la amplitud del voltaje de salida que el usuario indicó. En caso de que la entrada esté entre Ventana/2 y -Ventana/2, se tendrá que ver el estado anterior y mantenerse en dicho estado.

Para mostrar el estado en que se encuentre la salida, se ha dispuesto un LED indicador que en estado alto está prendido y en bajo se encuentra apagado.

Al terminar, se regresa el control de la rutina de interrupción al flujo del programa principal.

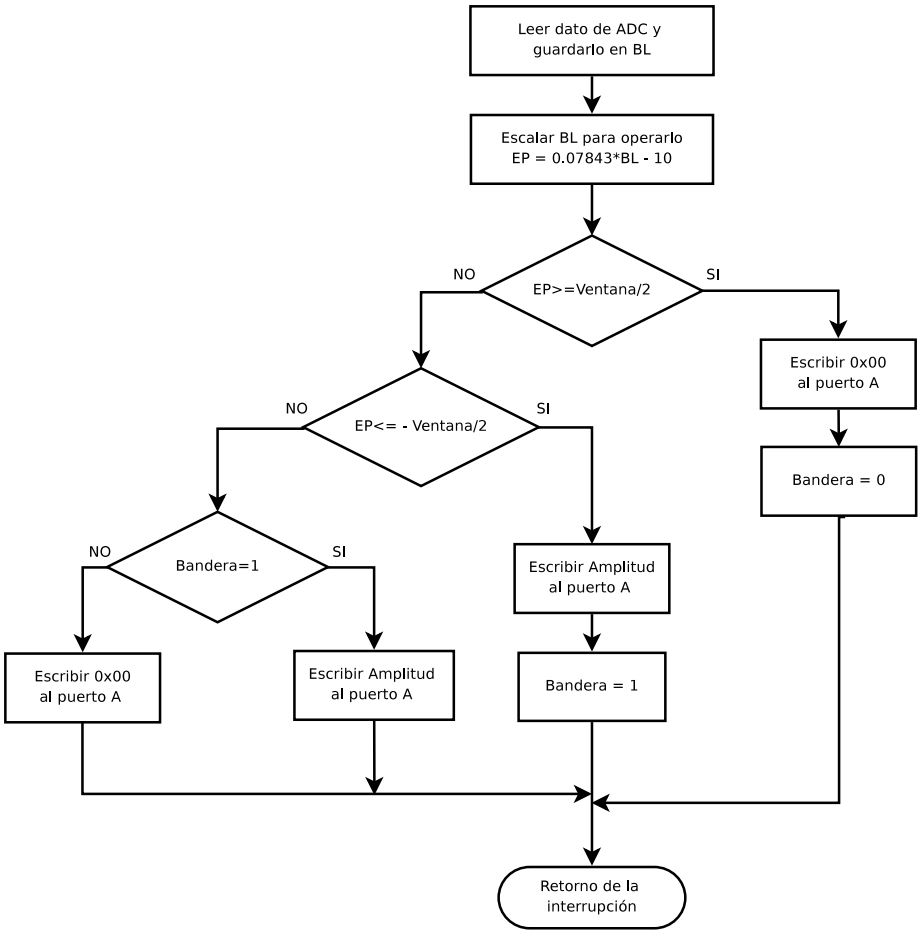


Figura 5.7: Diagrama de flujo del controlador *ON-OFF*.

En la figura 5.7 se muestra el diagrama de flujo para el control *ON-OFF*.

Control Proporcional

En la figura 5.8 se muestra el diagrama de flujo para la rutina de control proporcional. El algoritmo es el siguiente:

- Se lee el Error en el canal 0 del ADC.
- Se convierte el valor hexadecimal a decimal por medio de la ecuación 5.6.
- Se calcula la salida, por medio del modelo matemático del controlador P:

$$EP = K_C * EP$$

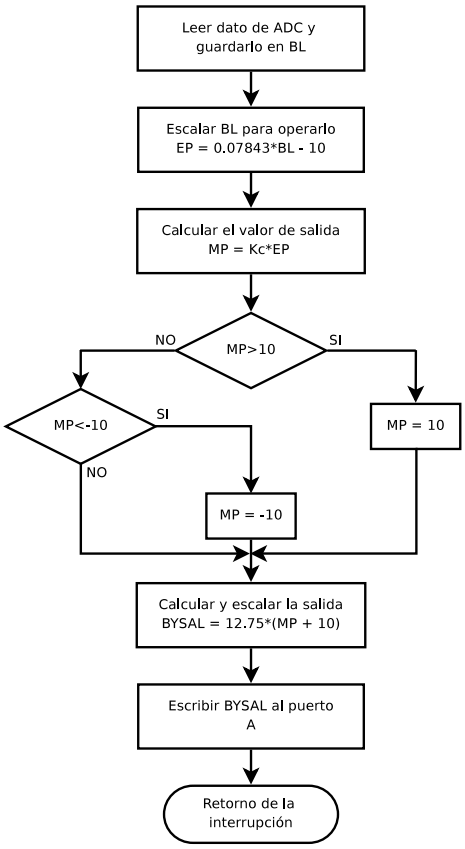


Figura 5.8: Diagrama de flujo del controlador P.

- Se verifica que la salida no rebase los límites impuestos por las características físicas del dispositivo, es decir, que se mantenga dentro del intervalo -10 [V] a + 10 [V].

- El valor de salida se convierte de decimal a su equivalente valor hexadecimal, por medio de la ecuación 5.7.

- Se manda la salida al puerto A.

- Termina la subrutina de interrupción y sale.

Control Proporcional Integral

En la figura 5.9, se observa el diagrama de flujo que resuelve la ecuación en diferencias asociado a un controlador Proporcional Integral.

El algoritmo presentado en el diagrama es el siguiente:

- Se lee el Error en el canal 0 del ADC.

- Se convierte el valor hexadecimal a decimal por medio de la ecuación 5.6.

- Se calcula la salida, por medio de la ecuación en diferencias del controlador PI, en términos del Error Presente, el Error Anterior y la Salida Anterior:

$$MP = b_0 * EP + b_1 * EANT + a_1 * MANT$$

  Los coeficientes a_1,b_0 y b_1, son los calculados por medio de las ecuaciones que aparecen en la tabla 5.1.

- Se verifica que la salida no rebase los límites impuestos por las características físicas del dispositivo, es decir, que se mantenga dentro del intervalo -10 [V] a + 10 [V].

- El valor de salida se convierte de decimal a su equivalente valor hexadecimal, por medio de la ecuación 5.7.

- Se manda la salida al puerto A.

- Se guardan los valores de Salida Actual(MP en $MANT$) y Error Actual (EP en $EANT$), para la siguiente iteración.

- Termina la subrutina de interrupción y sale.

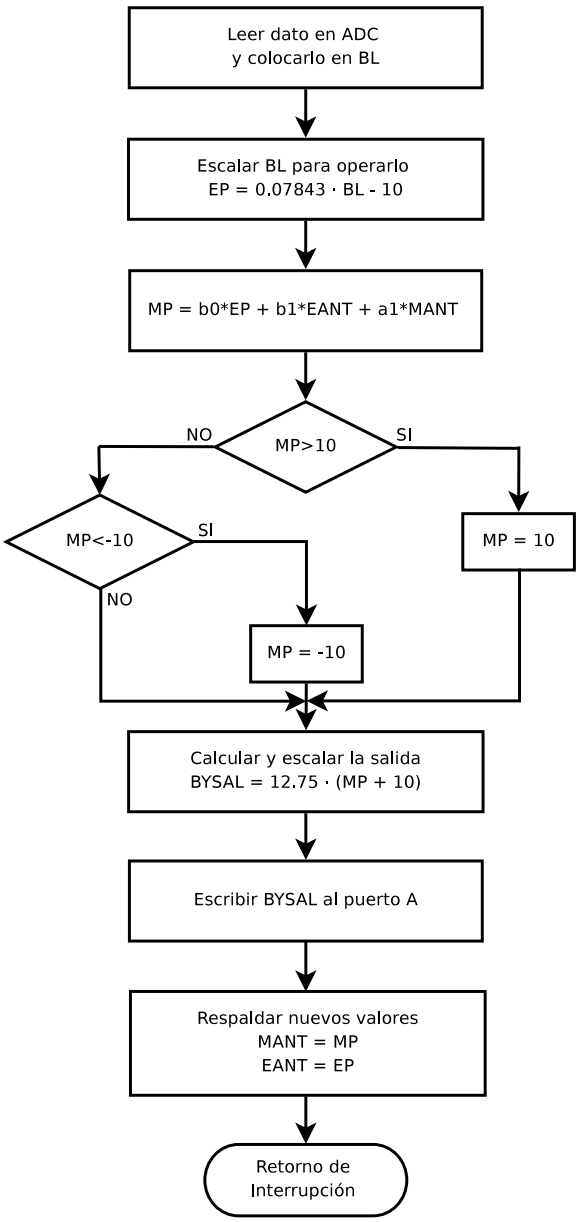


Figura 5.9: Diagrama de flujo para los controladores PI y PD.

Control Proporcional Derivativo

La ecuación en diferencias para el controlador PD es de primer orden, al igual que el del PI, por lo que comparte el algoritmo con este último para la solución de dicha ecuación. Por esa misma razón, el diagrama de flujo del control PD es el mostrado en la figura 5.9, que es la misma para el control PI.

La diferencia en los algoritmos de control del PD y del PI radica en los valores de los coeficientes de la ecuación en diferencias asociada a cada controlador.

A continuación se presenta el algoritmo de control del Proporcional Derivativo:

- Se lee el Error en el canal 0 del ADC.
- Se convierte el valor hexadecimal a decimal por medio de la ecuación 5.6.
- Se calcula la salida, por medio de la ecuación en diferencias del controlador PD, en términos del Error Presente, el Error Anterior y la Salida Anterior:

  $$MP = b_0 * EP + b_1 * EANT + a_1 * MANT$$

  Los coeficientes a_1,b_0 y b_1, son los calculados por medio de las ecuaciones que aparecen en la tabla 5.1.
- Se verifica que la salida no rebase los límites impuestos por las características físicas del dispositivo, es decir, que se mantenga dentro del intervalo -10 [V] a + 10 [V].
- El valor de salida se convierte de decimal a su equivalente valor hexadecimal, por medio de la ecuación 5.7.
- Se manda la salida al puerto A.
- Se guardan los valores de Salida Actual(MP en $MANT$) y Error Actual (EP en $EANT$), para la siguiente iteración.
- Termina la subrutina de interrupción y sale.

Control Proporcional Integral Derivativo

Cuando el controlador a ejecutar es el Proporcional Integral Derivativo, se ejecuta el algoritmo representado en el diagrama de flujo de la figura 5.10.

Ese algoritmo es el siguiente:

- Se lee el Error en el canal 0 del ADC
- Se convierte el valor hexadecimal a decimal por medio de la ecuación 5.6.

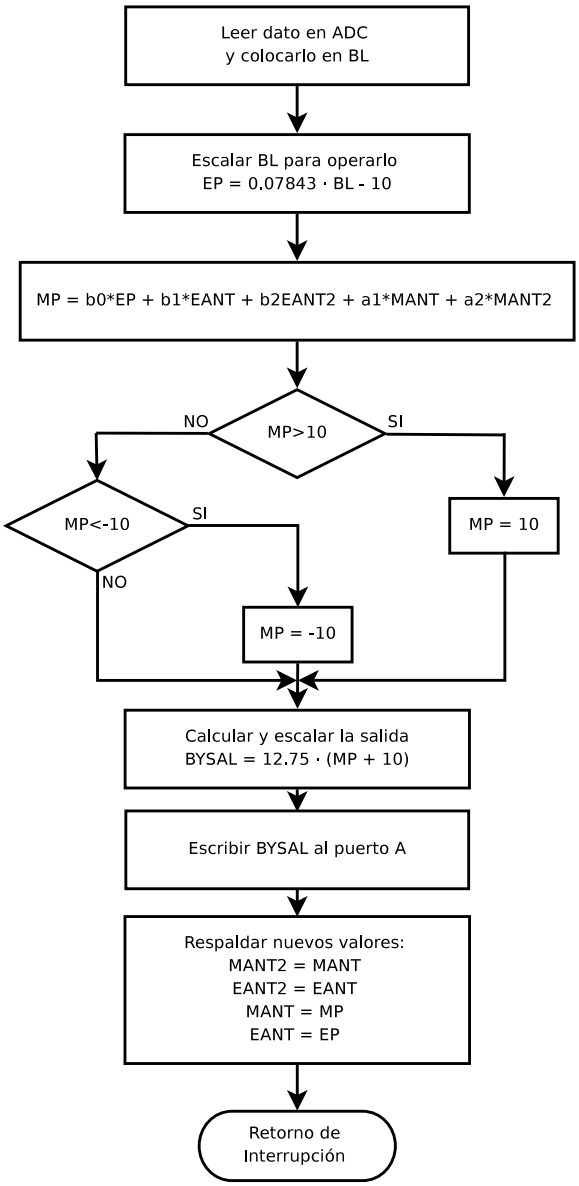


Figura 5.10: Diagrama de flujo para el controlador PID.

- Se calcula la salida, por medio de la ecuación en diferencias del controlador PID, en términos del Error Presente, el Error Anterior, el Error Antes del Anterior, la Salida Anterior y la Salida Antes de la Anterior.

  $$MP = b_0 * EP + b_1 * EANT + b_2 * EANT2 + a_1 * MANT + a_2 * MANT2;$$

  Los coeficientes a_1, a_2, b_0, b_1 y b_2 son los calculados por medio de las ecuaciones que aparecen en la tabla 5.1.

- Se verifica que la salida no rebase los límites impuestos por las características físicas del dispositivo, es decir, que se mantenga dentro del intervalo -10 [volts] a + 10 [volts].

- El valor de salida se convierte de decimal a su equivalente valor hexadecimal, por medio de la ecuación 5.7.

- Se manda la salida al puerto A.

- Se guardan los valores de Salida Anterior ($MANT$ en $MANT2$), Salida Actual(MP en $MANT$), Error Anterior ($EANT$ en $EANT2$) y Error Actual (EP en $EANT$), para la siguiente iteración.

- Termina la subrutina de interrupción y sale.

Capítulo 6

INTERFAZ GRÁFICA DE USUARIO

6.1. Introducción

Se ha dicho en capítulos anteriores que la forma en que el usuario puede indicarle los parámetros de control al controlador digital construido es por medio de una computadora PC. De ahí que sea necesaria la presencia de un agente que pueda servir de interfaz entre la PC y el dispositivo controlador digital. Ese agente es la interfaz de usuario.

El propósito de este capítulo es describir el programa interfaz que se ejecuta en la computadora personal, es decir, el lenguaje utilizado para dicho programa así como el algoritmo.

A continuación se presenta una figura que muestra la necesidad de un agente interfaz que permita al usuario una comunicación unidireccional con la computadora personal y ésta tenga una comunicación directa con el microcontrolador; ya que de no contarse con este agente, el usuario no podría comunicarse con el microcontrolador que ejecuta las distintas acciones de control y no podría variar los diversos parámetros involucrados.

Lo anterior significa que la interfaz de usuario con el controlador digital está compuesta de dos partes que pueden ser programadas de manera independiente siguiendo algún estándar; esas dos partes son:

- Interfaz gráfica de usuario con la computadora.
- Interfaz de la computadora con el dispositivo controlador.

En la figura 6.1 se puede apreciar un diagrama conceptual que ejemplifica este concepto.

Dada la gran versatilidad que tienen las computadoras personales de la actualidad, se tienen una gran gama de opciones para implementar la interfaz deseada.

En este capítulo se explica la alternativa escogida y el porqué de esa elección.

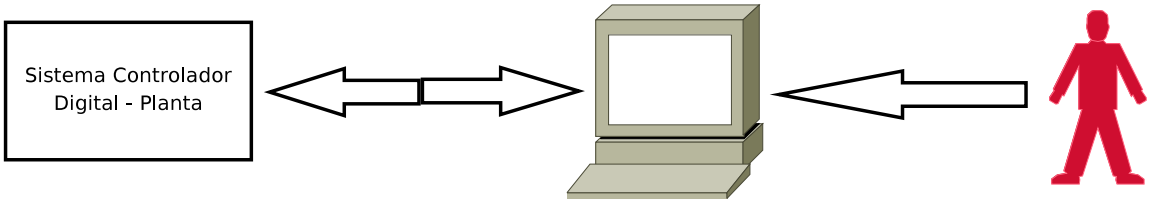


Figura 6.1: Diagrama conceptual de interacción usuario - PC - Controlador Digital.

6.2. La interfaz gráfica de usuario

6.2.1. Definición de interfaz

Se entiende por interfaz con una computadora al programa o aplicación que permite a la computadora comunicarse con un dispositivo externo así como el hardware dedicado al mismo propósito.

6.2.2. Diseño de la interfaz

Las necesidades que debe cubrir la interfaz de usuario a implementar son las siguientes:

- Debe ser una GUI, es decir una interfaz gráfica de usuario. Se prefiere la GUI a una interfaz de texto, por ser más intuitiva.
- Debe ser portable. Dado que el propósito de este proyecto de tesis, es la construcción de un dispositivo para apoyo didáctico, la interfaz debe ser portable, para que la misma pueda ser usada en diversas plataformas y no obligar al usuario a permanecer en una plataforma para usar la aplicación.
- Debe ser de fácil manejo.
- Debe ser robusto.

De acuerdo a los requerimientos mencionados anteriormente, con respecto a la interfaz de usuario, se puede observar que ésta puede ser programada en una amplia gama de lenguajes (como C, C++, Java, Visual Basic, C#, etcétera) utilizando una gran variedad de librerías disponibles (librerías nativas de Windows o Unix u otro sistema operativo, .Net, Gtk, Qt, etc.), cada cual con sus ventajas y desventajas, como lo son la sencillez y simplicidad de desarrollo para una plataforma, o la posibilidad de ejecutarse en una variedad de sistemas operativos.

6.2.3. Elección del lenguaje de programación

Para programar la interfaz con el usuario y la comunicación con la computador personal se ha optado por el lenguaje *java*, que al ser un lenguaje multiplataforma funciona especialmente bien para propósitos didácticos en los que se tiene una población de usuarios con requerimientos diversos.

También influyó en la selección del lenguaje su amplio uso, desarrollo y soporte.

La versión de java con la que se ha escrito la interfaz es la 1.6 de Sun [1].

Java es un lenguaje de programación orientado a objetos, (OOP por sus siglas en inglés Object-Oriented Programming), desarrollado por Sun Microsystems a principios de los años noventa. Es un lenguaje de código intermedio que por dicha característica requiere de una máquina virtual para ejecutarse y que al mismo tiempo le permite ejecutarse en cualquier arquitectura que cuente con tal máquina virtual. Esta característica le dio al lenguaje una gran popularidad en sus inicios, lo que redundó en su soporte para una gran variedad de tareas.

Aquellos lugares donde el lenguaje y sus especificaciones quedan cortos para realizar ciertas tareas (como la comunicación mas directa con el hardware del sistema) pueden ser realizadas por rutinas escritas en otros lenguajes (C o C++ principalmente) y agregadas al programa en java mediante JNI (Java Native Interface o Interfaz Nativa con Java) que es una especificación del lenguaje que le permite cargar y ejecutar rutinas en librerías nativas al sistema operativo (por ejemplo, Windows, Linux, Unix o MacOS). Dichas librerías requieren, sin embargo, estar presentes en el sistema operativo en cuestión y conformarse con los requerimientos dictados por el programa que las utiliza y las especificaciones de JNI para la versión de java de que se trate.

6.2.4. Ambiente integrado de desarrollo (IDE)

Un programa en su forma más básica no es mas que un archivo de texto en una codificación tal que el compilador encargado de generar el binario pueda entender. Esto significa que para escribir un programa no es necesario más que un simple editor de texto. Sin embargo, se puede facilitar enormemente la tarea del desarrollo del programa si se cuenta herramientas para el propósito. El conjunto de estas herramientas cuando pueden interoperar se conoce como Ambiente Integrado de Desarrollo (o IDE por sus siglas en inglés), y resulta valioso para la programación.

El IDE en el cual se desarrolló la interfaz con el usuario y la comunicación con el controlador digital es eclipse [2]. La elección de eclipse se hizo por familiaridad la previa del autor del programa, por su amplio soporte y por ser una herramienta muy completa y de código abierto. Esta interfaz cuenta con una gran cantidad de *plug-ins* o herramientas integrables extras que permiten tener acceso a varias utilidades no incluidas en su dis-

[1] http://java.sun.com
[2] http://www.eclipse.org

tribución estándar. De esta forma es posible, por ejemplo, agregar soporte al IDE para diversos lenguajes en añadidura de Java, o para metodologías de desarrollo alternativas, como diagramas UML, o programación gráfica de interfaces.

La interfaz gráfica de usuario se diseñó con el *plug-in* conocido como *jigloo* [3] de eclipse. Se eligió este *plug-in* porque permite un diseño rápido de la interfaces gráficas en lenguaje Java, es decir, la programación de interfaces gráficas se reduce al diseño de componentes como ventanas, botones, cajas de texto, etcétera y la escritura de funciones o subrutinas cada vez que suceda un evento con estos componentes.

Con este *plug-in* se logran dos cosas al mismo tiempo: sencillez y robustez, ya que permite enfocarse en el diseño del algoritmo asociado a los controles gráficos y no a la programación de dichos controles.

Para el diseño de la interfaz gráfica de usuario en sí, se consideraron los cinco tipos de control programados:

- *ON-OFF*
- Proporcional (P)
- Proporcional - Derivativo (PD)
- Proporcional - Integrador (PI)
- Proporcional - Integrador - Derivativo (PID)

Estos cinco controladores pueden configurarse mediante 2 pestañas, En la primera pestaña se configura el controlador *ON-OFF*, mientras que en la segunda se configuran cualquiera de los controladores P, PI, PD o PID. En esta última pestaña, el controlador se elige mediante una serie de opciones excluyentes para saber si se trata de un controlador P, PI, PD o PID.

En las figuras 6.2 y 6.3 se pueden observar el diseño de la ventana junto con las opciones anteriormente mencionadas.

Los valores permitidos para las variables de entrada son los siguientes:

- Periodo de muestreo: cualquier valor mayor a cero. Es el periodo con que se muestrea la señal de entrada del controlador.
- Ventana: Un valor entre 0 y 10, ambos inclusive.
- Amplitud: Un valor entre -10 y 10, ambos inclusive.
- Constantes K_P, T_d y T_i: Valores mayores a 0.

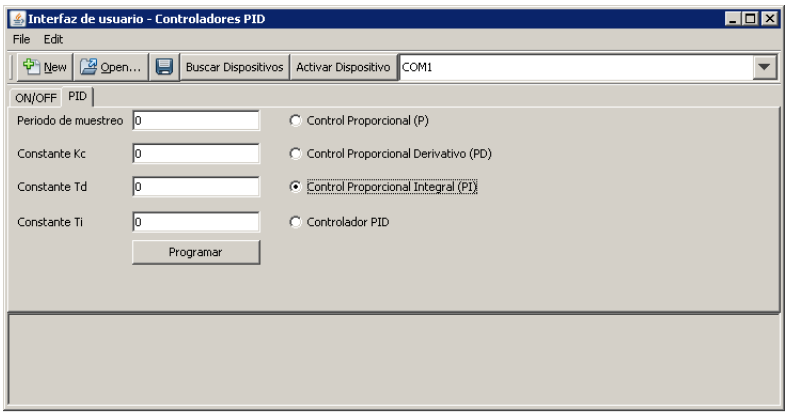


Figura 6.2: Interfaz gráfica para introducir valores de los parámetros de los controladores P, PI, PD o PID.

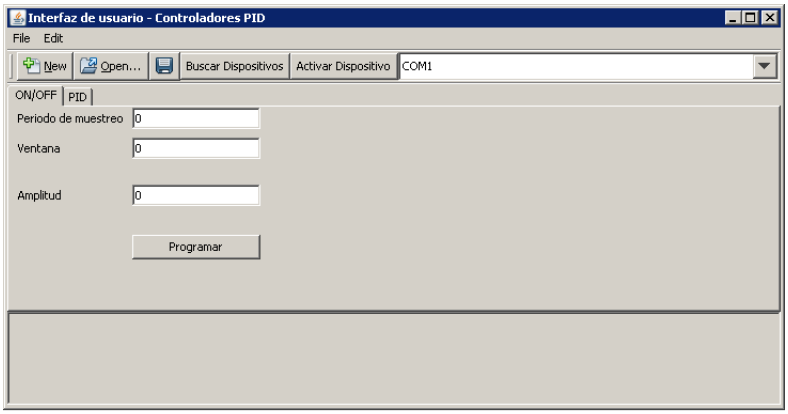


Figura 6.3: Interfaz gráfica para introducir valores de los parámetros del controlador *ON-OFF*.

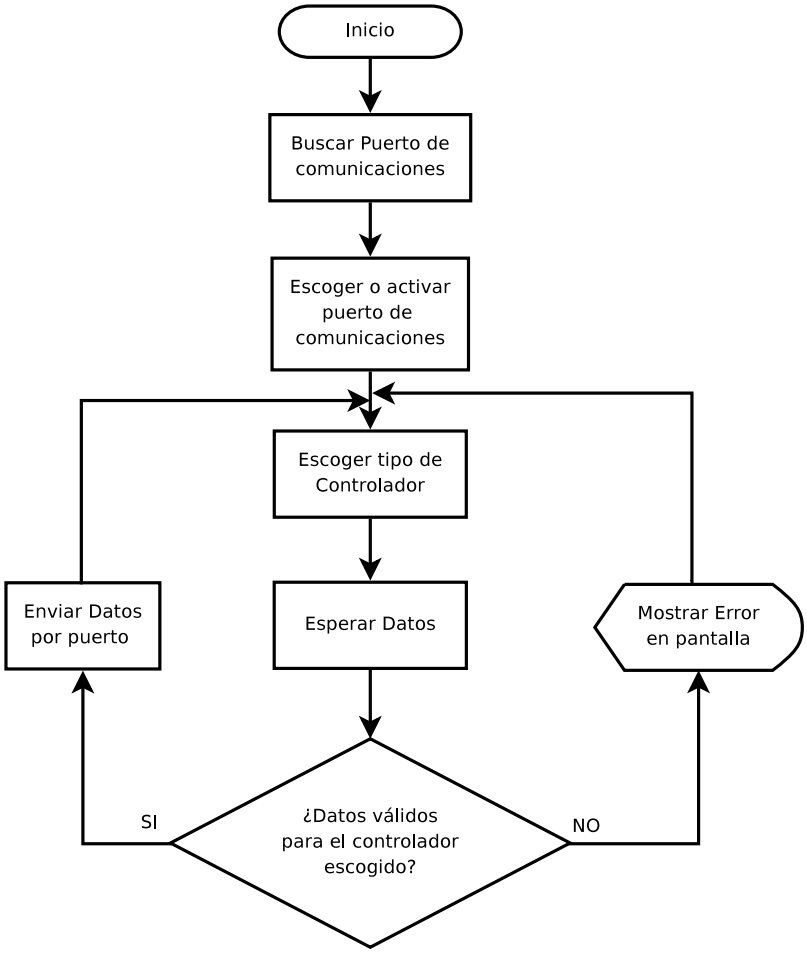


Figura 6.4: Diagrama de flujo de la interfaz gráfica de usuario.

A continuación se muestra un diagrama de flujo que indica el proceso que sigue la interfaz gráfica de usuario para mandar los datos a través del puerto serial. Ese diagrama puede observarse en la figura 6.4.

El diagrama de flujo que se encuentra en la figura 6.4 puede explicarse de la siguiente manera:

- Al inicio, la aplicación queda a la espera de que el usuario inicie la búsqueda de puertos serie en la computadora.
- Después de la búsqueda, se escoge el puerto serie de comunicaciones a usar y al cuál está conectado el dispositivo controlador digital.
- Se elige el tipo de controlador a simular.
- La aplicación se queda a la espera de los parámetros de control para el controlador elegido.
- Se hace la validación de datos. Si los datos son correctos se envían por el puerto de comunicaciones en uso. En caso contrario, se manda un mensaje de error en la pantalla.
- La aplicación queda a la espera de datos. Se puede elegir el tipo de controlador en cualquier momento.

El código en Java escrito para la interfaz gráfica de usuario se puede observar en el apéndice A.

6.3. Interfaz computadora - Dispositivo controlador digital

6.3.1. Comunicación con el controlador digital

La interfaz del controlador digital con una computadora personal se lleva a cabo mediante comunicación serial y una programación en el lado de la computadora personal para procesar los datos de la comunicación. La comunicación serial requiere un puerto con el protocolo RS-232 o para el caso de aquellas PCs más recientes que ya no cuentan con dichos puertos será necesario un convertidor serial DB-9 a USB.

La comunicación con el controlador digital por medio de una interfaz serial real o virtual (transparente para el programa) está integrada en la interfaz gráfica mediante las clases y librerías incluidas con el proyecto libre *rxtx* [4].

El proyecto *rxtx* consiste en un conjunto de clases de Java que permite la comunicación de una computadora por puerto serie y paralelo con otro dispositivo, independientemente

[3] *http://www.cloudgarden.com/jigloo/*
[4] http://www.rxtx.org

del sistema operativo que utilice dicha computadora y siempre y cuando el proyecto cuente con las librerías JNI para soporte de dicho sistema operativo.

Formato de la transmisión de datos

En el capítulo 5 de esta tesis, se menciona que la transmisión de datos de la computadora en donde corre la interfaz gráfica de usuario y el dispositivo controlador construido, se hace de manera serial asíncrona. También se menciona que debido a las limitaciones físicas del microcontrolador que solo cuenta con un registro de recepción de datos de 8 bits, la transmisión de datos de números en punto flotante se hace de grupos de 8 bits.

En el programa en Java de la interfaz de usuario que se muestra en el apéndice A se puede observar que el envío de datos es, efectivamente en grupos de 8 bits.

Por su parte, en el programa que se ejecuta en el microcontrolador, existe una subrutina especial que recibe los datos que se le envía por su puerto serie y los va colocando en localidades de memoria contiguas.

En el capítulo 5 se puede encontrar información detallada sobre la forma en que se reciben y almacenan los datos recibidos de la transmisión serial.

Capítulo 7

EJEMPLOS DE DISEÑO DE CONTROLADORES

7.1. Introducción

En este capítulo se muestra un caso práctico para el diseño de controladores básicos. Se presenta una planta sencilla de segundo orden (un circuito RC) y se proponen unas especificaciones de desempeño. Después se hace el diseño de los principales controladores básicos a partir de dichas especificaciones.

Debido a que la función de transferencia de la planta en cuestión es conocida, el diseño se simplifica bastante. Por esta razón se hace otro estudio para el caso en que no se conoce la función de transferencia de la planta, con un método conocido como de "Ziegler-Nichols". Este método es ampliamente utilizado en la industria.

El capítulo está dividido en las siguientes secciones: la planta, donde se hace la presentación del sistema a controlar, con la deducción de su función de transferencia y las características de su respuesta en el tiempo a una entrada escalón unitario. La sección 7.3 presenta varios métodos analíticos y gráficos de diseño de los controladores básicos. En la sección 7.4 se obtienen los coeficientes de la ecuación en diferencias asociado al controlador que resuelve el microcontrolador. En la última sección se presenta una introducción al método de Ziegler-Nichols para la sintonización de controladores PID para plantas cuya función de transferencia no se conoce.

7.2. La planta

La planta propuesta para este estudio es un circuito RC de segundo orden. Dicha planta se muestra en la figura 7.1. A continuación se presenta la deducción de su función de transferencia:

Como la planta contiene dos subcircuitos RC, separados por un *seguidor de voltaje*, ésta

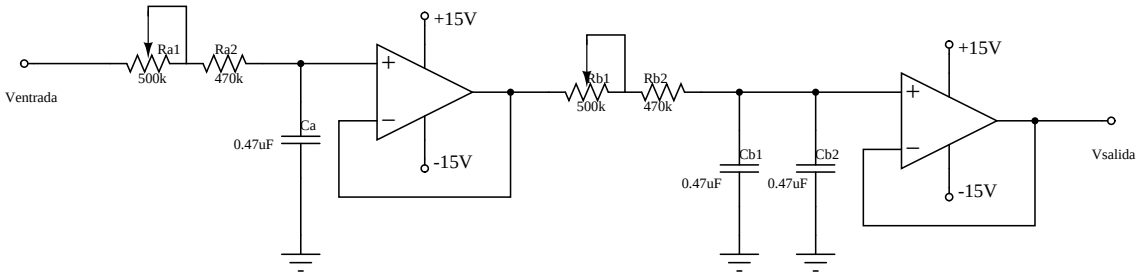


Figura 7.1: Planta de prueba usada.

puede representarse como $G_P = (g1)(g2)$ donde $g1$ es el primer subcircuito representado por los componentes R_{a_1},R_{a_2} y C_a; mientras que $g2$ es el segundo subcircuito representado por los componentes R_{b_1}, R_{b_2}, C_{b_1} y C_{b_2}.

Para hallar la función de transferencia de la planta, se midieron los valores exactos de los componentes R_{a_1}, R_{a_2}, R_{b_1}, R_{b_2}, C_a, C_{b_1} y C_{b_2}, los cuáles fueron:

$$\begin{aligned} R_{a_1} &= 498\ [k\Omega] \\ R_{a_2} &= 464\ [k\Omega] \\ C_a &= 0.39\ [\mu F] \\ R_{b_1} &= 491\ [k\Omega] \\ R_{b_2} &= 469\ [k\Omega] \\ C_{b_1} &= 0.457\ [\mu F] \\ C_{b_2} &= 0.457\ [\mu F] \end{aligned}$$

Para $g1$:

Éste es un circuito de primer orden tipo RC, cuya dinámica puede modelarse con la siguiente ecuación diferencial:

$$(R_{a_1} + R_{a_2})C_a \frac{dV_A}{dt} + V_A = V_i$$

o en su forma normalizada:

$$\frac{dV_A}{dt} + \frac{1}{(R_{a_1} + R_{a_2})C_a} V_A = \frac{1}{(R_{a_1} + R_{a_2})C_a} V_i \tag{7.1}$$

Al hallar la transformada de Laplace de 7.1, considerando condiciones iniciales nulas, se tiene que:

$$sV_A(s) + \frac{1}{(R_{a_1} + R_{a_2})C_a} V_A(s) = \frac{1}{(R_{a_1} + R_{a_2})C_a} V_i(s)$$

A partir de la ecuación anterior, se tiene que la función de transferencia de g_1 es:

$$\frac{V_A(s)}{V_i(s)} = \frac{\frac{1}{(R_{a_1}+R_{a_2})C_a}}{s + \frac{1}{(R_{a_1}+R_{a_2})C_a}} \tag{7.2}$$

Para $g2$:

Al igual que para $g1$, éste es un circuito de primer orden tipo RC cuya dinámica puede modelarse con la siguiente ecuación diferencial, en su forma normalizada:

$$\frac{dV_o}{dt} + \frac{1}{(R_{b_1} + R_{b_2})(C_{b_1} + C_{b_2})} V_o = \frac{1}{(R_{b_1} + R_{b_2})(C_{b_1} + C_{b_2})} V_A \tag{7.3}$$

Al hallar la transformada de Laplace de 7.3, considerando condiciones iniciales nulas, se tiene que:

$$sV_o(s) + \frac{1}{(R_{b_1} + R_{b_2})(C_{b_1} + C_{b_2})} V_o(s) = \frac{1}{(R_{b_1} + R_{b_2})(C_{b_1} + C_{b_2})} V_A(s)$$

A partir de la ecuación anterior, se tiene que la función de transferencia de g_2 es:

$$\frac{V_o(s)}{V_A(s)} = \frac{\frac{1}{(R_{b_1}+R_{b_2})(C_{b_1}+C_{b_2})}}{s + \frac{1}{(R_{b_1}+R_{b_2})(C_{b_1}+C_{b_2})}} \tag{7.4}$$

Por lo anterior, la función de transferencia de la planta es $g1(s)g2(s)$ por la presencia de los *seguidores de voltaje*, ya que la cantidad de corriente que circula por $g2$ no afecta el comportamiento dinámico de $g1$. Es importante recalcar que si esta condición no se cumple, la forma de determinar la función de transferencia de la planta sería distinta ya que los bloques no estarían en cascada.

La función de transferencia de la planta es:

$$G_P(s) = \frac{V_A(s)}{V_i(s)} \frac{V_o(s)}{V_A(s)} = \frac{V_o(s)}{V_i(s)} \tag{7.5}$$

$$G_P(s) = \left(\frac{\frac{1}{(R_{a_1}+R_{a_2})C_a}}{s + \frac{1}{(R_{a_1}+R_{a_2})C_a}} \right) \left(\frac{\frac{1}{(R_{b_1}+R_{b_2})(C_{b_1}+C_{b_2})}}{s + \frac{1}{(R_{b_1}+R_{b_2})(C_{b_1}+C_{b_2})}} \right) \tag{7.6}$$

Sustituyendo los valores mostrados en la tabla 7.2:

$$G_P(s) = \frac{3.04}{(s + 2.67)(s + 1.14)} \tag{7.7}$$

Aquí es importante señalar que debido a que se tienen dos circuitos RC, siempre se tendrán polos reales para esta clase de plantas, de tal manera que su respuesta a escalón unitario siempre será el de un sistema subamortiguado tal como aparece en la figura 7.2.

El lugar geométrico de las raíces del sistema en lazo cerrado se presenta en la figura 7.3. Con respecto al dicho lugar geométrico es necesario recalcar que éste es para la planta cuando se cierra el lazo con realimentación unitaria, cómo se muestra en la figura 7.4 y donde el parámetro variable es la ganancia.

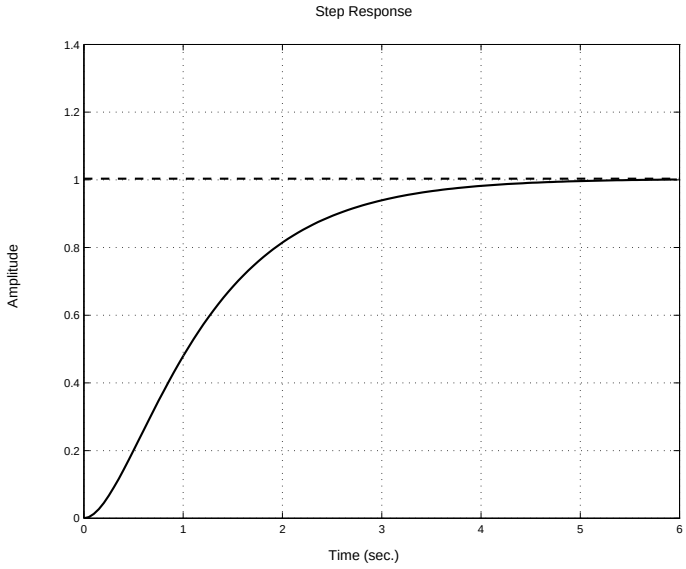


Figura 7.2: Respuesta a escalón unitario de la planta G_P.

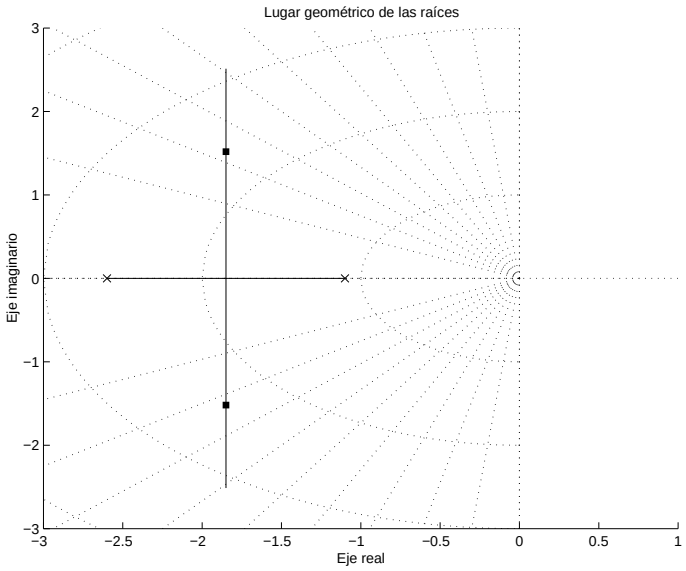


Figura 7.3: Lugar geométrico de las raíces de lazo cerrado.

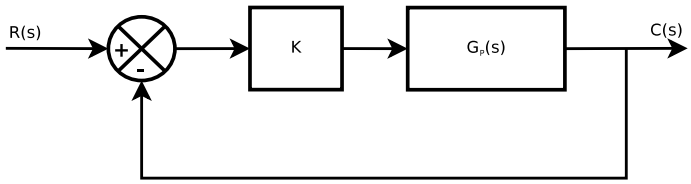


Figura 7.4: Diagrama de bloques cuyo lugar geométrico se muestra en la figura 7.3.

En este punto conviene resaltar que el sistema total no tiene el bloque conocido como *actuador*, esto es así debido a que las variables que involucran a la planta y a la de la salida del controlador son del mismo tipo. Por la misma razón, en el lazo de realimentación no se tiene un sensor o un transductor. Si la planta a controlar involucrara variables de distinto tipo al del controlador, sería necesaria la presencia de estos dos elementos. Es necesario aclarar estos dos puntos, ya que frecuentemente, existe una ligera confusión en esto.

7.3. Diseño de un controlador analógico para una planta con función de transferencia conocida

En esta sección se mostrará un procedimiento que se puede seguir para diseñar un controlador básico, a saber, P, PI, PD y PID. Este procedimiento es totalmente analítico, ayudándose de gráficas, y sirve solamente cuando se conoce el modelo matemático de la planta a controlar.

Cuando se desea diseñar un controlador analógico deben darse unas especificaciones que se deben cumplir una vez que el controlador sea incluido en el sistema. Estas especificaciones suelen darse en función de unos valores, ya sea en el dominio del tiempo o en el dominio de la frecuencia. Generalmente las especificaciones que se dan en el dominio del tiempo son:

- Porcentaje de sobrepaso (M_P).
- Tiempo de asentamiento (t_s).
- Porcentaje de error en estado estable(e_{ss}).

De la misma manera, las especificaciones que se dan en el dominio de la frecuencia son:

- Pico de resonancia (M_r).
- Frecuencia de resonancia (ω_r).
- Ancho de banda (BW).

Conviene aclarar que, para el diseño de un sistema de control, se prefieren las especificaciones en el dominio del tiempo, en particular, la respuesta transitoria, más que en el dominio de la frecuencia [1]. Es por esa razón, que en los sistemas de control, diseñar en el dominio de la frecuencia es una alternativa para dar las especificaciones en el dominio del tiempo, es decir, se dan las especificaciones de desempeño del dominio del tiempo en forma indirecta (a través de los del dominio de la frecuencia arriba listados).

Además de las especificaciones cuantitativas dadas, existen otros de carácter cualitativo, que se dan al momento de empezar el diseño, a saber, la estabilidad, la precisión y la velocidad, aunque esta última se puede dar en términos del tiempo de subida.

Para este caso, se darán las especificaciones de desempeño en el dominio del tiempo y se diseñarán los distintos controladores básicos, tratando de que la salida del sistema se ajuste al comportamiento indicado para la planta cuya función de transferencia es:

$$G_P(s) = \frac{C(s)}{R(s)} = \frac{3.04}{(s+2.67)(s+1.14)}$$

Las especificaciones de desempeño son:

Sobrepaso	$M_P = 20\,\%$
Tiempo de asentamiento	$t_s = 5$ segundos
El sistema debe ser estable.	

Para que la planta tenga una salida como la deseada; es decir $M_P = 20\,\%$ y $t_s = 5$ segundos, se necesita que los polos en lazo cerrado sean:

$$s_{p_{1,2}} = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2} \tag{7.8}$$

donde ω_n y ζ se relacionan con las especificaciones dadas, de las formas en que se muestran en las ecuaciones 7.9 y 7.10, para un sistema de segundo orden.

$$\zeta = \frac{1}{\sqrt{\left(\frac{\pi}{\ln M_P}\right)^2 + 1}} \tag{7.9}$$

$$t_s = \frac{3}{\zeta\omega_n} \tag{7.10}$$

Es necesario, aclarar que la ecuación 7.10, considera al estado estable como los valores que se encuentren dentro de un rango del 5 % de su valor final. Si se necesita mayor exactitud, se puede utilizar un criterio del 2 %.

Sustituyendo los valores deseados de M_P y t_s en las ecuaciones 7.9 y 7.10 se encuentran los valores de ω_n y ζ.

[1] Contrario a lo que se hace en otras áreas, por ejemplo los sistemas de comunicaciones, en donde las especificaciones más importantes para el diseño son los del dominio de la frecuencia.

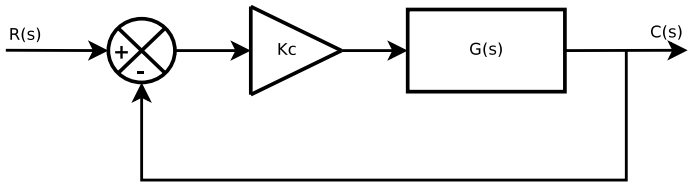


Figura 7.5: Diagrama de bloques del controlador proporcional empleado.

$$\zeta = 0.456$$
$$\omega_n = 1.316$$

Por lo anterior y por la ecuación 7.8, los polos del sistema en lazo cerrado deben ser :

$$s_{p_{1,2}} = -0.6 \pm j1.171 \tag{7.11}$$

7.3.1. El controlador P

El diseño de un controlador P, consiste en encontrar el valor de la ganancia K_C que hace que la salida de la planta sea igual al planeado con las especificaciones de desempeño. Debido a la característica del lugar geométrico de las raíces, una buena alternativa para encontrar el valor de K_C es trazar dicha gráfica y ubicar la posición de los polos deseados para encontrar el valor de dicha ganancia (K_C).

Se observa en el lugar geométrico de las raíces, mostrado en la figura 7.3, que éste nunca pasa por los polos deseados. Lo anterior significa que por más que se varíe la ganancia, nunca se logrará obtener las características de desempeño que se desean, es decir, $M_P = 20\,\%$ y $t_s = 5$ segundos.

Al no poder hacer pasar el lugar geométrico de las raíces por los polos deseados, se hará el diseño del controlador P para que los polos se encuentren lo más cercano posible a los polos deseados, considerando que la respuesta de la planta no será igual al comportamiento que se desea. Se tendrá que hacer un compromiso entre el sobrepaso máximo y el tiempo de asentamiento. Según [12], para una respuesta conveniente de un sistema de segundo orden, el factor de amortiguamiento relativo debe estar entre 0.4 y 0.8. Así que para este caso particular, se ha optado por mantener el valor de $\zeta = 0.456$ y a partir de ahí encontrar el valor de ω_n para que el polo se encuentre sobre el lugar geométrico de las raíces que se muestra en la figura 7.3.

Debido a que $\sigma = 1.9$ (es el valor absoluto donde el lugar geométrico de las raíces cruza al eje σ) y:

$$\sigma = \omega_n \zeta$$

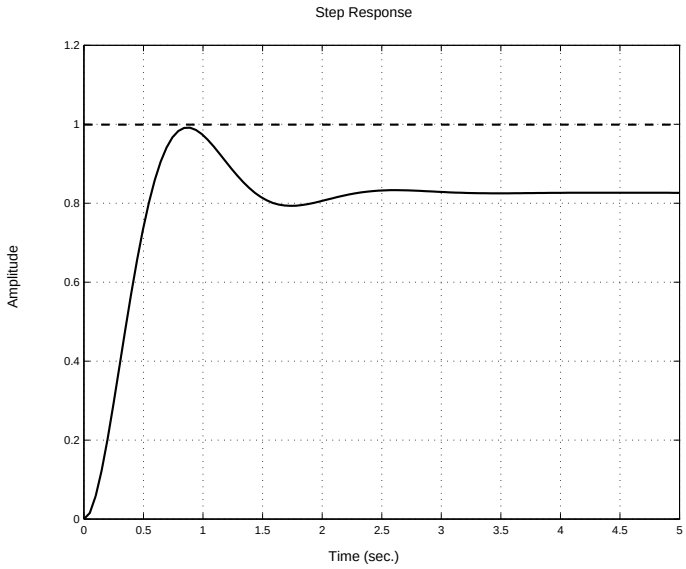


Figura 7.6: Respuesta del sistema al control proporcional.

Entonces,

$$\omega_n = \frac{\sigma}{\zeta} = \frac{1.9}{0.456} = 4.167$$

Con estas nuevas condiciones ($\omega_n = 4.167$ y $\zeta = 0.456$), los polos sobre el lugar geométrico de las raíces de la figura 7.3, de acuerdo a 7.8 son:

$$s'_{p_{1,2}} = -1.9 \pm j3.71$$

Para calcular la ganancia K_C en los polos $s'_{p_{1,2}}$, se puede usar un programa computacional o hacerlo de forma manual usando el álgebra. Para este caso y bajo las condiciones mencionadas, la ganancia es $K_C = 4.7$.

Con esos valores de frecuencia natural no amortiguada y coeficiente de amortiguamiento encontrados, la respuesta del sistema a un escalón unitario se muestra en la figura 7.6. En dicha figura se observa que existe un error en estado estable de aproximadamente 19 % y un sobrepaso de casi 20 %. Está claro que la ganancia K_C puede incrementarse con el objeto de reducir el error en estado estable, sin embargo para este caso, conviene dicho valor debido a las limitaciones físicas del circuito. Además de que un incremento en demasía de la ganancia K_C podría provocar una inestabilidad del sistema.

Las limitaciones físicas están dadas por el voltaje de la fuente. A continuación se hace un análisis de lo que sucede físicamente cuando se utiliza un controlador proporcional.

Aunque teóricamente se puede incrementar el parámetro K_C para reducir el error en estado estable, los voltajes que maneja el convertidor analógico digital impiden que se haga

físicamente. Para este caso particular, en que $K_C = 4.7$ y para un error de 2 volts, la salida en el controlador es de $K_C x2 = 9.4$ $[V]$, dicho valor cae dentro del intervalo de voltajes que puede manejar el convertidor digital a analógico, es decir, de -10 [V] a +10 [V]. Sin embargo, cuando el error es de 3 [V], la salida del controlador es de $K_C x3 = 14.1$, lo cuál queda fuera del intervalo mencionado anteriormente. Si se llega a dar el caso en que el error sobrepase los 3 volts, el sistema controlador presentará un comportamiento errático, ya que no se estará cumpliendo con la ley de control que rige al controlador P.

Por todas estas razones, el controlador proporcional, en este caso, tiene un comportamiento pobre, es decir, el error en estado estacionario es alto, aunque el tiempo de asentamiento está dentro del rango deseado, al igual que el sobrepaso.

Este controlador, tendría un buen funcionamiento si el sistema tuviera una respuesta lenta, tal cómo sucede en un sistema de control de temperatura en donde dicha variable, cambia lentamente, sin embargo, no se recomendaría su uso en un sistema en el que las variables involucradas variaran rápidamente.

7.3.2. El controlador PI

La función de transferencia de un controlador Proporcional Integral está dada por:

$$G_C(s) = K_C \left(1 + \frac{1}{T_i s}\right) \tag{7.12}$$

Por otro lado, para un sistema de control en lazo cerrado con una planta cuya función de transferencia es $G_P(s)$, y un controlador con función de transferencia $G_C(s)$, con realimentación unitaria,tiene la siguiente ecuación característica:

$$G_C(s)G_P(s) + 1 = 0 \tag{7.13}$$

De la ecuación 7.13 se tiene que:

$$G_C(s) = -\frac{1}{G_P(s)} \tag{7.14}$$

Por lo tanto:

$$G_C(s) = -\frac{1}{\frac{3.04}{(s+2.67)(s+1.14)}} \tag{7.15}$$

Como se desea que los polos del sistema en lazo cerrado, estén en

$$s_{p_{1,2}} = -0.6 \pm j1.171$$

éste se sustituye en la ecuación 7.15 y se obtiene:

$$G_C(s) = 0.0835 - j1.0055 \tag{7.16}$$

Por otro lado, al sustituir el polo deseado en la ecuación 7.12:

$$G_C(s) = K_C\left(1 + \frac{1}{T_i(-0.6 + j1.171)}\right)$$

se obtiene:

$$G_C(s) = \left(K_C - 0.3466\frac{K_C}{T_i}\right) - j0.6764\frac{K_C}{Ti} \tag{7.17}$$

Igualando las ecuaciones 7.16 y 7.17 se obtiene:

$$\left(K_C - 0.3466\frac{K_C}{T_i}\right) - j0.6764\frac{K_C}{Ti} = 0.0835 - j1.0055 \tag{7.18}$$

Al igualar la parte real y la parte imaginaria de los números complejos en la ecuación 7.18 y la ecuación 7.16 se forma un sistema de dos ecuaciones con dos incógnitas:

$$\begin{array}{l} K_C - 0.3466\frac{K_C}{T_i} = 0.0835 \\ 0.6764\frac{K_C}{Ti} = 1.0055 \end{array} \tag{7.19}$$

Del sistema de ecuaciones 7.19 se obtienen los parámetros requeridos:

$$\begin{array}{l} K_C = 0.5986 \\ T_i = 0.4027 \end{array}$$

Por lo tanto, la función de transferencia del controlador es:

$$G_C(s) = 0.5986\left(1 + \frac{1}{0.4027s}\right)$$

El sistema con realimentación unitaria y controlador Proporcional Integral con los parámetros calculados, tiene la respuesta a escalón unitario que se muestra en la figura 7.7.

Como se observa, la respuesta a escalón unitario del sistema con realimentación unitaria y controlador PI, es la deseada, es decir, la respuesta del sistema realimentado tiene un 20 % de sobrepaso y un tiempo de establecimiento de 5 segundos (con un criterio del 5 %).

7.3.3. El controlador PD

A continuación se muestra como hallar los valores de los parámetros de control, cuando el controlador usado es el Proporcional Derivativo y la planta de prueba es de segundo orden.

Dada la función de transferencia de la planta:

$$G_P(s) = \frac{3.04}{(s + 2.67)(s + 1.14)} \tag{7.20}$$

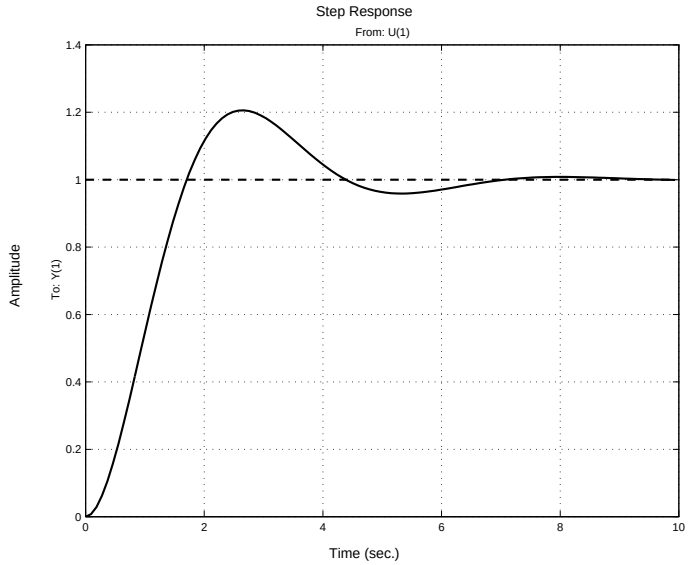


Figura 7.7: Respuesta a escalón unitario del sistema realimentado y controlador PI.

y la función de transferencia de un controlador PD:

$$G_C(s) = K_C\left(1 + T_d s\right) \tag{7.21}$$

Como el sistema de control es de lazo cerrado, la función de transferencia de lazo cerrado del sistema es:

$$G(s) = \frac{G_C(s)G_P(s)}{1 + G_C(s)G_P(s)} \tag{7.22}$$

Al sustituir 7.20 y 7.21 en 7.22, se tiene:

$$G(s) = \frac{3.04K_C\left(1 + T_d s\right)}{\left(s + 2.67\right)\left(s + 1.14\right) + 3.04K_C\left(1 + T_d s\right)} \tag{7.23}$$

Al simplificar 7.23 se tiene:

$$G(s) = \frac{3.04K_C\left(1 + T_d s\right)}{s^2 + \left(3.81 + 3.04K_C T_d\right)s + \left(3.0438 + 3.04K_C\right)} \tag{7.24}$$

Como se puede observar en la expresión 7.24, la función de transferencia corresponde a un sistema de segundo orden, por lo que su polinomio característico es de segundo grado. De [12], se sabe que el polinomio característico de un sistema de segundo orden está dada por:

$$s^2 + 2\zeta\omega_n + \omega_n^2 = 0 \tag{7.25}$$

Es evidente que el comportamiento dinámico del sistema total está dado por los valores de ζ y ω_n. De las especificaciones del sistema de control se tiene:

$$\zeta = 0.456$$
$$\omega_n = 1.316$$

Para que el comportamiento del sistema de control en cuestión sea igual al pedido ($M_P = 20\,\%$ y $t_s = 5$ segundos) se tendrá que igualar 7.25 con el polinomio característico dado en el denominador de 7.23. Al hacer lo anterior, se tiene:

$$\begin{aligned} 3.81 + 3.04K_C T_d &= 2\zeta\omega_n \\ 3.0438 + 3.04K_C &= \omega_n^2 \end{aligned} \tag{7.26}$$

La solución del sistema de ecuaciones dado por 7.26 resulta en una constante K_C negativa. Lo anterior significa que las condiciones mencionadas no se pueden cumplir para el sistema en cuestión. Por lo que se tiene que hacer un compromiso entre las especificaciones de desempeño (M_P y t_s) para hacer que el control proporcional derivativo pueda ser usado con la planta propuesta. A continuación se muestra como se resuelve esta dificultad.

Del sistema de ecuaciones anterior se sabe que para que las dos constantes sean positivas se tiene que cumplir que:

$$2\zeta\omega_n > 3.81 \tag{7.27}$$

Entonces:

$$\zeta\omega_n > 1.905$$

Si se escoge un valor arbitrario, por ejemplo $\zeta\omega_n = 2$ para que se cumpla la desigualdad 7.27, se tiene:

$$t_s = \frac{3}{\zeta\omega_n} = \frac{3}{2} = 1.5 \tag{7.28}$$

De la ecuación 7.28 se concluye que el tiempo de asentamiento máximo debe ser alrededor de 1.5 segundos para que se pueda usar el controlador Proporcional Derivativo. Tomando este tiempo como una nueva especificación de desempeño, junto con el sobrepaso $M_P = 0.2$, a continuación se muestra como hallar el valor de los dos parámetros de control.

Dado que $\zeta\omega_n = 2$ y $\zeta = 0.456$, entonces:

$$\omega_n = 4.39 \tag{7.29}$$

Por lo tanto, los nuevos valores de ω_n y ζ son:

$$\omega_n = 4.39$$
$$\zeta = 0.456$$

Por lo que al resolver el sistema de ecuaciones dado por 7.26, los valores de K_C y T_d son:

$$T_d = 0.0117$$
$$K_C = 5.338$$

Con estos valores, la respuesta a escalón unitario de la planta para un controlador PD se muestra en la figura 7.8.

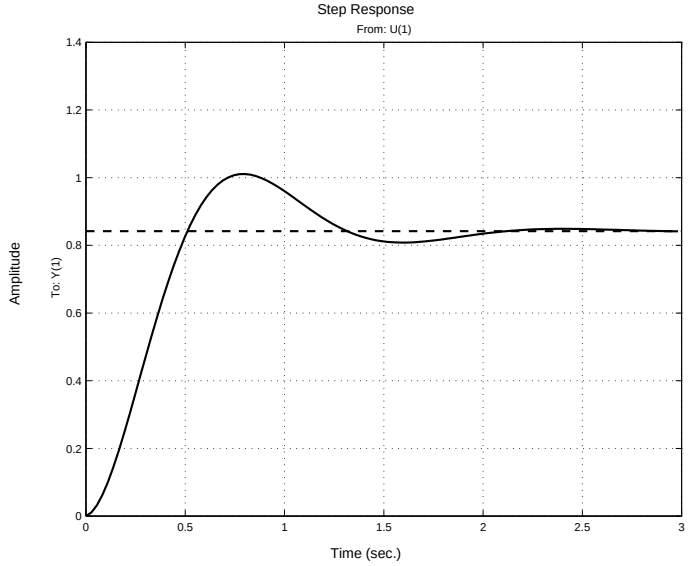


Figura 7.8: Respuesta a escalón unitario del sistema realimentado y controlador PD.

7.3.4. El controlador PID

El controlador PID tiene la siguiente función de transferencia:

$$G_C(s) = K_C \left(1 + T_d s + \frac{1}{T_i s}\right) \tag{7.30}$$

Los polos deseados del sistema en lazo cerrado son:

$$s_{p_{1,2}} = -0.6 \pm j1.171$$

Con un procedimiento similar al seguido para el cálculo de los parámetros del controlador PI, se sustituyen los polos deseados en la ecuación 7.15, que es la que representa la función de transferencia del sistema en lazo cerrado con un controlador G_C. Se obtiene que:

$$G_C = 0.0835 - j1.0055 \tag{7.31}$$

Por otro lado, al sustituir los polos deseados en la ecuación 7.30 se obtiene:

$$G_C(s)|_{s=-0.6+j1.171} = K_C \left(1 + T_d(-0.6 + j1.171) + \frac{1}{T_i(-0.6 + j1.171)}\right)$$

Al resolver:

$$G_C = K_C \left[\left(1 - 0.6T_d - \frac{0.3466}{T_i}\right) + j\left(1.171T_d - \frac{0.6746}{T_i}\right)\right] \tag{7.32}$$

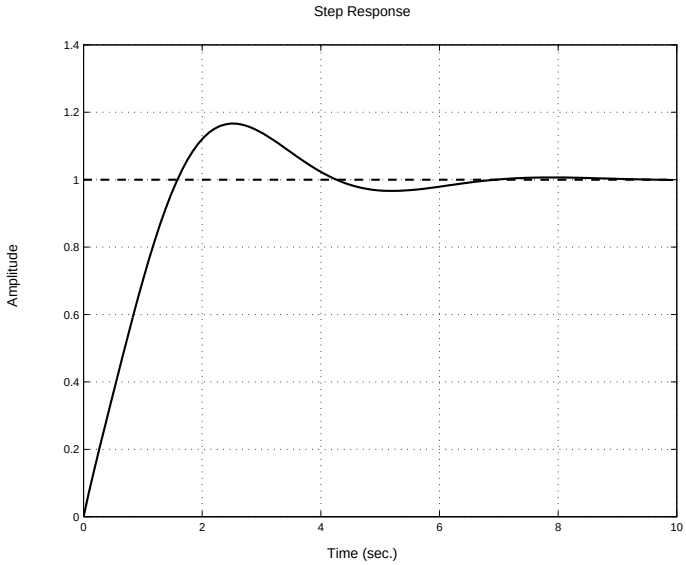


Figura 7.9: Respuesta a escalón unitario del sistema con controlador PID.

Igualando las ecuaciones 7.32 y 7.31 se obtiene un sistema de dos ecuaciones con tres incógnitas:

$$\begin{array}{l} K_C\left(1-0.6T_d-\frac{0.3466}{T_i}\right)=0.0835 \\ K_C\left(1.171T_d-\frac{0.6746}{T_i}\right)=1.0055 \end{array} \tag{7.33}$$

La ecuación 7.33 se puede resolver, fijando una de las incógnitas y resolviendo para las otras dos.

En este caso se fija el parámetro K_C en 1. Se acostumbra fijar este parámetro a un valor razonable, ya que éste depende de las limitaciones físicas del sistema tal como se ha explicado líneas atrás.

Al resolver el sistema de ecuaciones 7.33, se obtienen los siguientes valores para los parámetros:

$$\begin{array}{l} K_C=1 \\ T_i=0.4840 \\ T_d=0.3345 \end{array}$$

Con estos valores, la función de transferencia del controlador PID es:

$$G_C(s)=1+0.3345s+\frac{1}{0.4840s} \tag{7.34}$$

La respuesta del sistema en lazo cerrado a una entrada escalón unitario, se muestra en la figura 7.9.

Se observa en la figura 7.9 que el sobrepaso no llega a ser del 20 % como se había pronosticado. El tiempo de asentamiento es de aproximadamente 5 segundos con un criterio del 5 % tal como se desea. El tiempo de levantamiento es menor a 1 segundo por lo que la respuesta del sistema es buena en comparación con el controlador P.

7.4. Cálculo de los coeficientes de la ecuación en diferencias

Aunque los coeficientes de la ecuación en diferencias son calculados dentro del programa del microcontrolador, en esta sección se muestra como se hacen esos cálculos, con el objetivo de contrastar y comparar las salidas de los controladores diseñados en la sección anterior con la salida de los controladores discretos que se derivaron a partir de los analógicos.

A partir de los parámetros de control T_d, T_i,K_C, se calcularon los coeficientes de la ecuación en diferencias de primero o de segundo orden. Para la ecuación en diferencias de primer orden, se usan b_0, b_1 y a_1 ya que la ecuación se supone normalizada. En el caso de que la planta sea de segundo orden (por consiguiente su ecuación en diferencias es también de segundo orden), se usan los coeficiente: b_0, b_1, b_2, a_1 y a_2, porque también se supone que la ecuación está normalizada.

Para el controlador PI.

$$\begin{aligned} a_1 &= 1 \\ b_0 &= K_c\left(\tfrac{T+2T_i}{2T_i}\right) \\ b_1 &= K_c\left(\tfrac{T-2T_i}{2T_i}\right) \end{aligned}$$

Para el controlador PD.

$$\begin{aligned} a_1 &= -1 \\ b_0 &= K_c\left(\tfrac{T+2T_d}{T}\right) \\ b_0 &= K_c\left(\tfrac{T-2T_d}{T}\right) \end{aligned}$$

Para el controlador PID.

$$\begin{aligned} a_1 &= 0 \\ a_2 &= 1 \\ b_0 &= K_c\left(\tfrac{4T_iT_d+2T_iT+T^2}{2TT_i}\right) \\ b_1 &= K_c\left(\tfrac{T^2-4T_iT_d}{T_iT}\right) \\ b_2 &= K_c\left(\tfrac{4T_iT_d-2T_iT+T^2}{2TT_i}\right) \end{aligned}$$

La deducción de estas ecuaciones, se hizo en el capítulo 5. En esta sección se hacen los cálculos de los coeficientes de la ecuación en diferencias usando dichas ecuaciones, para hallar la función de transferencia discreta asociada a cada uno de los controladores básicos y que sería la ecuación que resolvería el microcontrolador.

Cabe recalcar que el periodo de muestreo usado es de $T = 10$ [milisegundos]

7.4.1. El controlador P

En este caso, no hay una ecuación en diferencias asociado al controlador, ya que la salida de la misma en un instante determinado solamente depende de las entradas en ese instante.

7.4.2. El controlador PI

Los coeficientes de la ecuación en diferencias del controlador PI son:

$$\begin{aligned} a_1 &= 1 \\ b_0 &= 0.6060 \\ b_1 &= -0.5912 \end{aligned}$$

La función de transferencia discreta del controlador PI es:

$$H(z) = \frac{0.6060z - 0.5912}{z - 1}$$

En la figura 7.10 se muestra una comparación entre las salidas de los controladores PI analógico y digital con distintos periodos de muestreo.

En la figura 7.10 se observan las respuestas de un sistema con controlador analógico y con controlador discreto. En la gráfica 7.10b se utilizó un periodo de muestreo de 10 milisegundos para discretizar el controlador analógico, mientras que en la figura 7.10c se usó un periodo de muestreo de 100 milisegundos, por lo que en la respuesta de esta última es evidente el *efecto de escalera*.

Con un periodo de muestreo adecuado (10 milisegundos, para este caso) las respuestas en ambos sistemas prácticamente son iguales.

7.4.3. El controlador PD

Los coeficientes de la ecuación en diferencias del controlador PD son:

$$\begin{aligned} a_1 &= -1 \\ b_0 &= 8.9145 \\ b_1 &= -3.5765 \end{aligned}$$

cuando se utiliza un periodo de muestreo de 10 milisegundos. Para este caso, la función de transferencia discreta de dicho controlador es:

$$H(z) = \frac{8.9145z - 3.5765}{z + 1}$$

En la figura 7.11 se muestran las respuestas del sistema a escalón unitario cuando se tiene un controlador PD analógico y un PD digital con distintos periodos de muestreo. Cuando el periodo de muestreo es de 10 milisegundos, la respuesta es prácticamente idéntica a la respuesta de un controlador analógico. Eso se observa en la subfigura 7.11b.

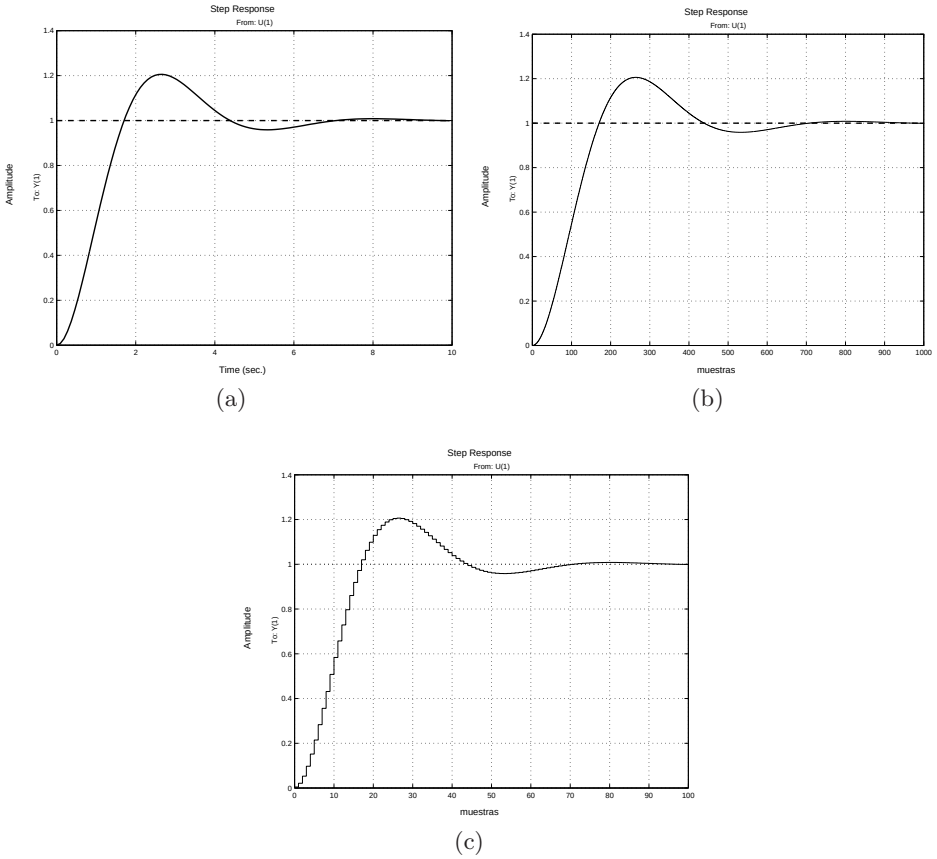


Figura 7.10: Comparación entre las respuestas de un sistema con controlador analógico y uno con controlador digital PI.

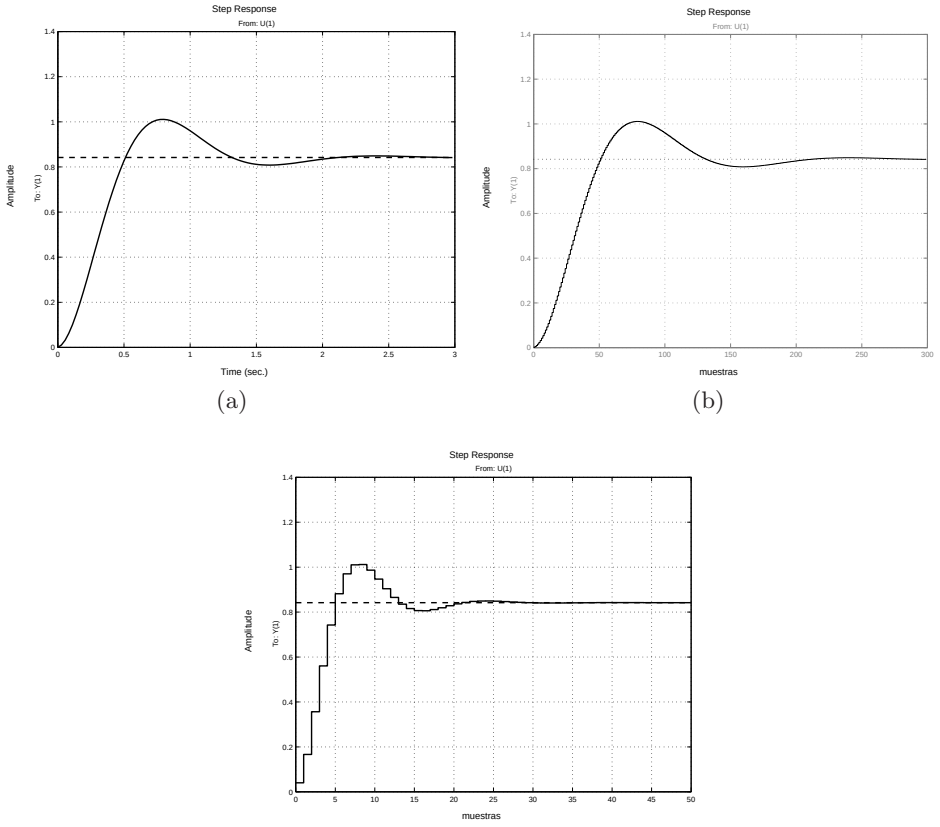


Figura 7.11: Comparación entre las respuestas de un sistema con controlador analógico y uno con controlador digital PD.

7.4.4. El controlador PID

Para el caso del controlador PID, se forma una ecuación en diferencias de segundo orden con los siguientes coeficientes, para un periodo de muestreo de 10 milisegundos:

$$\begin{aligned} a_1 &= 0 \\ a_2 &= 1 \\ b_0 &= 67.9103 \\ b_1 &= -133.7793 \\ b_2 &= 65.9103 \end{aligned}$$

Por lo que la función de transferencia del controlador discreto que se forma es:

$$H(z) = \frac{67.9103z^2 - 133.7793z + 65.9103}{z^2 - 1} \tag{7.35}$$

Si el periodo de muestreo para discretizar el controlador es de 100 milisegundos, la función de transferencia discreta del mismo es:

$$H(z) = \frac{7.7933z^2 - 13.1733z + 5.7933}{z^2 - 1} \tag{7.36}$$

En la figura 7.12 se muestran las salidas de los sistemas cuando se usa un controlador PID analógico y uno discreto con distintos periodos de muestreo.

7.5. Sintonización de controladores PID cuando no se conoce la función de transferencia de la planta. Método de Ziegler-Nichols

El método de Ziegler-Nichols es ampliamente utilizado en la industria. Por esa razón, en esta sección se hace una introducción sobre cómo se debe proceder para calcular los parámetros de un controlador PID para una planta cuya función de transferencia no se conoce, como a menudo ocurre en la realidad.

Es claro que el objetivo de este proyecto, es de tipo didáctico, por lo que casi siempre se trabajará con plantas cuyo modelo matemático esté perfectamente definido. Sin embargo, se subraya, que en algunas ocasiones se tendrá que trabajar con plantas cuyo modelo matemático no se pueda definir, por lo que se tendrá que hacer uso del método que se estudia en esta sección. Como será evidente posteriormente, el método de Ziegler-Nichols puede servir también cuando la función de transferencia de la planta en cuestión está definida con su modelo matemático, sin embargo el valor más apreciado de este método es cuando es imposible definir dicho modelo.

Aquí se presentan los dos métodos más conocidos y más usados, mismos que se usan bajo diferentes circunstancias.

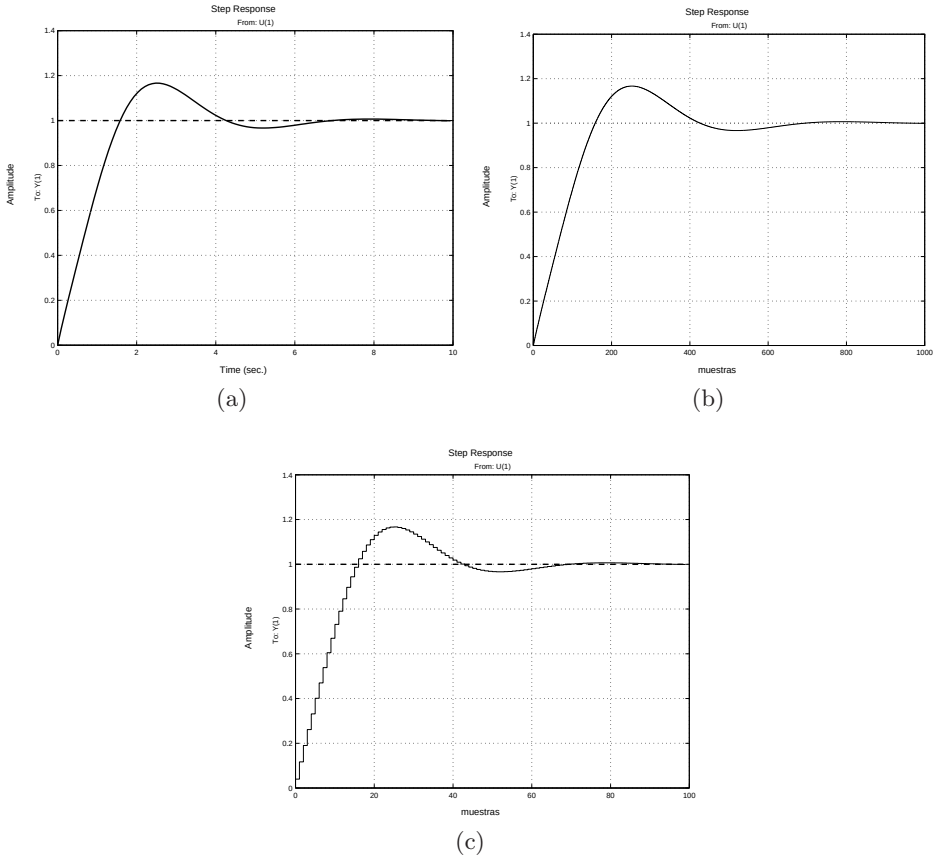


Figura 7.12: Comparación entre las respuestas de un sistema con controlador analógico y uno con controlador digital PID.

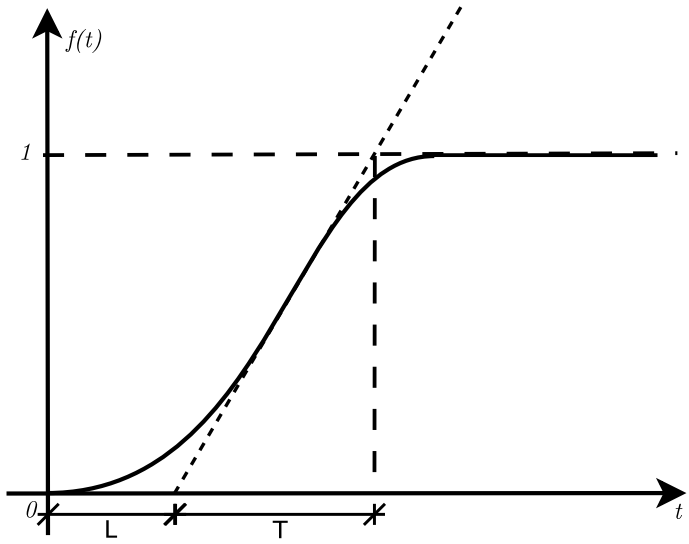


Figura 7.13: Respuesta a escalón unitario.

7.5.1. Primer método

El primer método de Ziegler-Nichols parte de la idea de que la respuesta escalón unitario de un sistema de cualquier orden tiene la forma de una S (sigmoidea). En caso de que dicha respuesta no tuviera esa forma, este método no puede aplicarse. La respuesta a escalón unitario de un sistema en lazo abierto, tiene la forma de S cuando dicho sistema no tiene integradores ni polos dominantes complejos.

Esta curva se presenta en la figura 7.13. En dicha figura, se pueden identificar dos parámetros: L y τ, los cuáles se conocen como el retardo y la constante de tiempo, respectivamente.

El primer método de Ziegler-Nichols, asegura que se puede obtener un sobrepaso del 25 % si los parámetros de un controlador P, PI o PID, se encuentran como se muestra en el cuadro 7.1 a partir de los valores de L y τ.

De acuerdo al cuadro 7.1, la función de transferencia de un controlador PID es:

$$G_C(s) = K_C\left(1 + T_d s + \frac{1}{T_i s}\right)$$

$$G_C(s) = 1.2\frac{\tau}{L}\left(1 + 0.5Ls + \frac{1}{2Ls}\right)$$

$$G_C(s) = 0.6\tau\frac{\left(s + \frac{1}{L}\right)^2}{s} \tag{7.37}$$

Tipo de Controlador	K_C	T_i	T_d
P	$\frac{T}{L}$	∞	0
PI	$0.9\frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{T}{L}$	$2L$	$0.5L$

Cuadro 7.1: Primer método de Ziegler-Nichols.

Se observa que el controlador PID sintonizado tiene un polo en el origen y un cero doble en $s = -1/L$.

7.5.2. Segundo método

El segundo método para sintonización de controladores PID, propuesto por Ziegler-Nichols, a diferencia del primero, analiza la respuesta del sistema de la planta con un controlador proporcional. El método indica que la ganancia K_C de dicho controlador se tiene que variar hasta que la salida del sistema tenga oscilaciones sostenidas. En el lugar geométrico de las raíces, esta ganancia se obtendría cuando los polos de lazo cerrado tengan su parte real sobre el eje $j\omega$.

La forma de la respuesta de dicho sistema se muestra en la figura 7.14.

Como se observa en la figura 7.14 el periodo de oscilación se le designa como P_{cr} y la ganancia a la cual ocurre dicha oscilación es K_{cr}.

De acuerdo con este segundo método, los parámetros del controlador P, PI y PID, se encuentran usando las relaciones que se presentan en el cuadro 7.2.

El controlador PID sintonizado mediante el segundo método, tiene la función de transferencia:

$$G_C(s) = K_C \left(1 + T_d s + \frac{1}{T_i s}\right)$$

$$G_C(s) = 0.6K_{cr} \left(1 + 0.125P_{cr}s + \frac{1}{0.5P_{cr}s}\right)$$

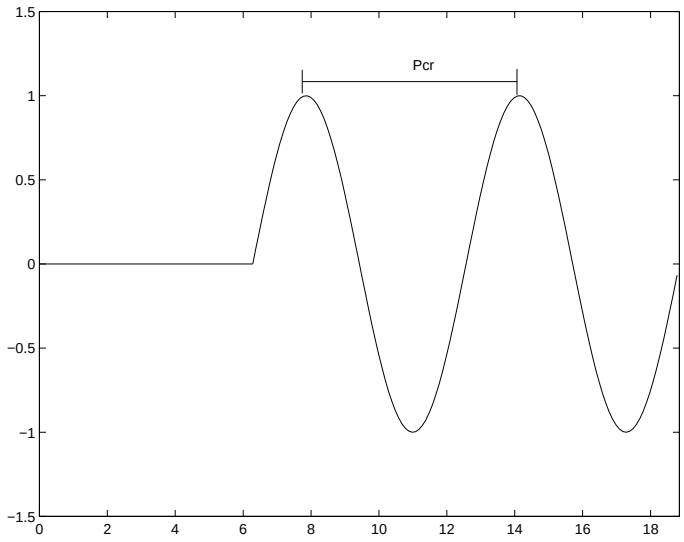


Figura 7.14: Respuesta de un sistema con oscilaciones sostenidas.

Tipo de Controlador	K_C	T_i	T_d
P	$0.5K_{cr}$	∞	0
PI	$0.45K_{cr}$	$\frac{1}{1.2}P_{cr}$	0
PID	$0.6K_{cr}$	$0.5P_{cr}$	$0.125P_{cr}$

Cuadro 7.2: Segundo método de Ziegler-Nichols.

$$G_C(s) = 0.075 K_{cr} Pcr \frac{\left(s + \frac{4}{P_{cr}}\right)^2}{s} \tag{7.38}$$

En la función de transferencia del controlador PID, se observa que éste tiene un polo en el origen y un cero doble en $s = -4/Pcr$.

7.5.3. Ejemplo de aplicación de los métodos de Ziegler-Nichols

En esta subsección, se hace uso de los dos métodos de Ziegler-Nichols explicados en esta sección para sintonizar un controlador PID para la planta que se explicó en la sección correspondiente.

Aunque debe aclararse que la bondad principal del método es la sintonización de los parámetros del PID para plantas cuyo modelo matemático se desconoce, aquí se aplica solo para ejemplificar.

Para este ejemplo, se ha utilizado el programa Matlab® para encontrar las variables requeridas.

Primer método

La respuesta a escalón unitario de la planta, se muestra en la 7.2. Se puede observar que dicha figura tiene forma *sigmoidal*, por lo que se puede usar el primer método para encontrar los valores de los parámetros del controlador PID.

Desde un punto de vista analítico, la recta que sirve para definir los valores de τ y L es la tangente de la curva en su punto de inflexión. Por lo que la ecuación que lo caracteriza puede encontrarse conociendo su pendiente y un punto de la misma. Una vez que se conozca dicha ecuación, se podrán determinar los valores de τ y L con simples operaciones de suma y resta. Siguiendo este algoritmo, se hizo un programa en Matlab® para encontrar los valores de τ y L, cuyo código se muestra en el apéndice A.

Los valores encontrados son:

$$\tau = 1.6564$$
$$L = 0.1540$$

Con estos valores, los parámetros del controlador PID son:

$$K_C = 12.9045$$
$$T_i = 0.3081$$
$$T_d = 0.0770$$

Por lo que la función de transferencia del controlador PID es:

$$G_C(s) = 0.9938 \frac{(s + 6.492)^2}{s} \tag{7.39}$$

Segundo método

Como se ha explicado en la subsección correspondiente, este método consiste en ir variando la ganancia de un controlador proporcional, hasta que la salida del sistema realimentado tenga oscilaciones sostenidas.

Para el caso de la planta en estudio podemos observar en el lugar geométrico de las raíces, que se aprecia en la figura 7.4, nunca cruza el eje $j\omega$, por lo que la salida de la planta nunca tendrá oscilaciones sostenidas por más que se varíe la ganancia, es decir no hay un valor de K_{cr}. Se concluye que el segundo método de Ziegler-Nichols no puede aplicarse en este sistema.

En la figura 7.15 se aprecia la salida del sistema realimentado cuando se usa un controlador PID diseñado por distintos métodos. La curva con líneas discontinuas es la salida del sistema cuando se usa el PID que se obtuvo en la sección por el método analítico. La curva con línea sólida es la salida del sistema cuando se usa el PID que se obtuvo con el primer método de Ziegler-Nichols.

Es evidente que la diferencia de sobrepasos es muy grande. Mientras que en el método analítico el sobrepaso es del 20 %, en el método de Ziegler-Nichols es aproximadamente del 40 %. Aunque en esta última, el tiempo de asentamiento es menor en comparación con el método analítico.

Si se necesita que el sobrepaso del sistema en cuestión sea menor que la que se observa en la figura 7.15, puede tomarse como base el diseño del PID por el método de Ziegler-Nichols para hacer un ajuste más fino.

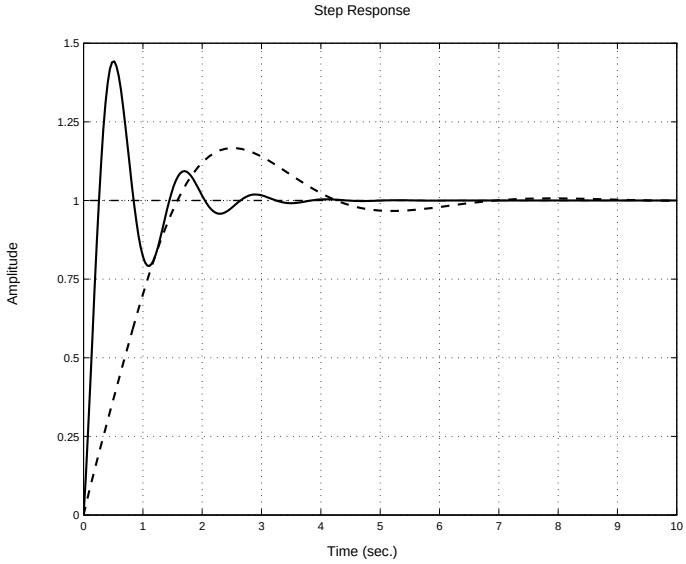


Figura 7.15: Comparación de salidas del sistema con controlador PID sintonizados con métodos distintos.

Capítulo 8

PRUEBAS PILOTO DEL DISPOSITIVO

8.1. Introducción

En este capítulo se muestran las pruebas realizadas en el controlador digital construido. En el capítulo 7 se mostró una forma de diseño de los controladores básicos para una planta de segundo orden. Aquí se usarán esos resultados para investigar el comportamiento del controlador digital desarrollado y hacer una comparación cuantitativa y cualitativa entre el diseño teórico y el resultado práctico.

8.2. Diagrama simplificado del dispositivo

En la figura 8.1 se muestra un diagrama de simplificado del dispositivo construido.

En los capítulos anteriores se mostraron los detalles de cada parte del circuito. Como se puede observar, la planta de prueba utilizada para validar el funcionamiento del dispositivo fue un circuito RC de segundo orden, cuya entrada es la salida del convertidor digital a analógico y cuya salida se conecta al circuito restador.

8.3. Pruebas realizadas

Las pruebas para garantizar el funcionamiento del controlador digital realizado en este proyecto de tesis se hicieron con una planta cuya función de transferencia fue mostrada en el capítulo 7, la cuál se reproduce a continuación:

$$G_P(s) = \frac{3.04}{(s + 2.67)(s + 1.14)}$$

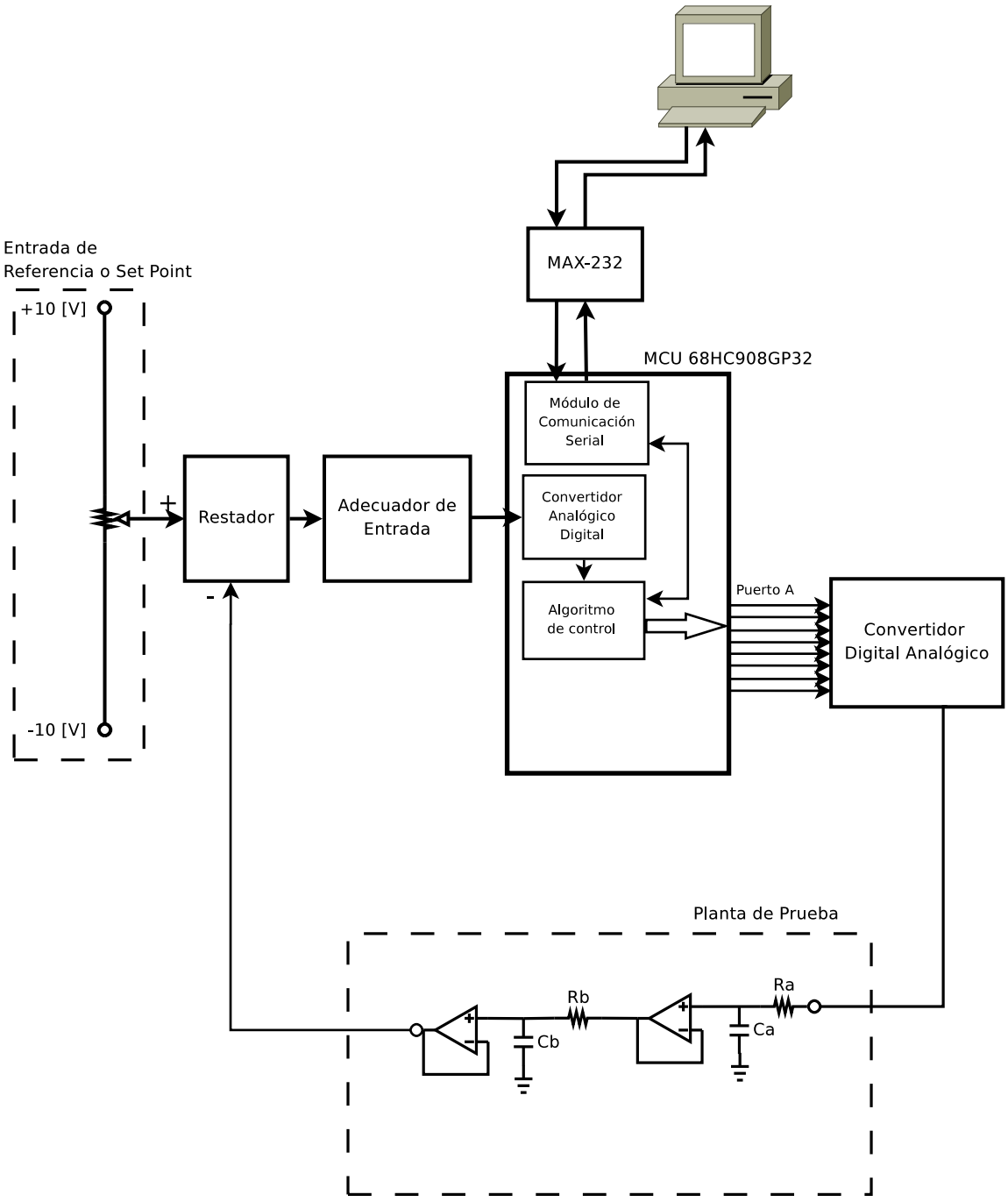


Figura 8.1: Diagrama de bloques simplificado del dispositivo construido.

Antes de mostrar los resultados de las pruebas realizadas, a continuación se exponen las circunstancias en las que se hicieron:

- La conexión del dispositivo construido, la planta y la PC, se muestra en la figura 8.1.
- Los datos se introdujeron por medio de la interfaz hecha en Java que se ejecutó en una laptop. Estos datos fueron: los parámetros de control, el periodo y el tipo de controlador.
- La laptop utilizada ejecutó el sistema operativo Windows XP. Esta laptop no contaba con un puerto serial, por lo que se se usó un cable convertidor de salida USB a serial.
- El *set point* se introdujo por medio de un potenciómetro de vuelta sencilla, para colocar el voltaje en el rango de -10 a +10 volts.
- Los resultados de la salida de la planta se mostraron en un osciloscopio digital de Agilent Technologies, modelo DSO3062A[1].

Aquí se reportan las pruebas hechas tomando como entrada una señal escalón. Las características de respuesta deseada son las que se mencionaron en el capítulo 7, a saber, porcentaje de sobrepaso = 20 % y tiempo de asentamiento de 5 segundos.

Con estas pruebas se investigó el comportamiento del dispositivo de control y la salida de la planta frente a una variación de la entrada.

El procedimiento seguido para hacer las pruebas fue el siguiente:

1. Se conectó el dispositivo, la PC, la planta de prueba y el osciloscopio.
2. Se introdujeron los parámetros de control y el periodo de muestreo por medio del programa interfaz que se ejecuta en la PC.
3. Se varió el *set point* con el potenciómetro de vuelta sencilla que se encuentra en el dispositivo para observar la respuesta de seguimiento del controlador digital construido.
4. Se observó el resultado en la pantalla del osciloscopio.
5. Se guardó el resultado en un dispositivo de almacenamiento.

Se aclara que en todas las pruebas realizadas, el canal 1 del osciloscopio se usó para visualizar la entrada de referencia (*set point*) mientras que el canal 2 se usó para visualizar la salida de la planta. Las figuras que se muestran en los siguientes apartados son exactamente las salidas que se vieron en la pantalla del osciloscopio usado. En dichas figuras se observan las escalas de división de tiempo y voltaje usadas.

[1]Este modelo tiene una entrada USB donde se puede conectar un dispositivo de almacenamiento para grabar los datos que se muestran en la pantalla. Se hizo uso de esta característica para almacenar la gráfica de salida que se muestra en las figuras presentadas en lo que resta de este capítulo

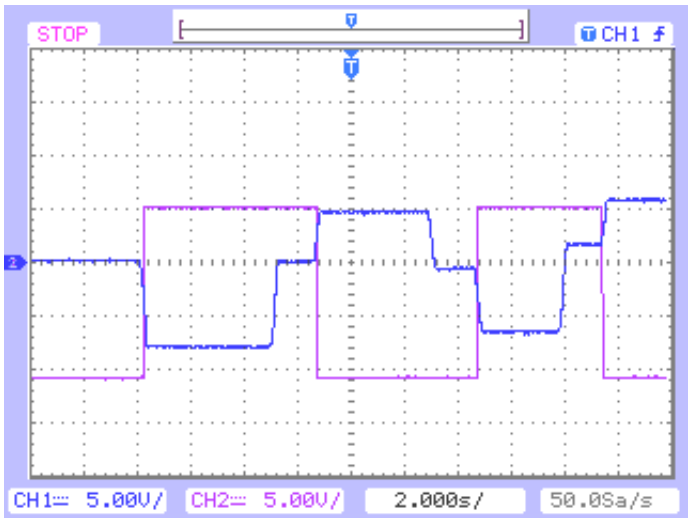


Figura 8.2: Resultados de la prueba realizada con el controlador ON-OFF

8.3.1. Controlador *ON-OFF*

Al ser éste el controlador más sencillo de los cinco programados, no se tuvo ningún problema al momento de hacer las pruebas. El controlador funciona a la perfección: se respeta la histéresis y la amplitud que se le introduce por medio de la interfaz.

Para hacer las pruebas para este controlador, no se usó ninguna planta, es decir, se desconectó. La salida que se midió fue la del DAC. Como no se contaba con ningún proceso o ninguna planta, el error permanecía constante. Para lograr que cambiara el error, se colocó un voltaje de 0 volts (aunque se pudo haber colocado un voltaje arbitrario) a la entrada negativa del restador y se movió el potenciómetro del *set point*, el cuál está conectado a la entrada positiva del restador.

Como el error cambiaba, en la salida del DAC se observaba cómo éste cambiaba de estado, es decir de estado *ON* a *OFF* o viceversa. El estado *OFF* manejado es de -10 volts, mientras que el estado *ON* es la amplitud que se coloca por medio de la interfaz de Java en la laptop.

El periodo de muestreo usado fue el mínimo, es decir, de 10 milisegundos.

El resultado observado en el osciloscopio se muestra en la figura 8.2. El canal 1 del osciloscopio muestra el error que varía en forma arbitraria, mientras que en el canal 2 se puede observar la salida del controlador.

Cabe mencionar que para el uso de este controlador en un sistema real, la salida del dispositivo construido debe ir conectado a un dispositivo actuador antes de conectarse a la planta. La salida de la planta debe conectarse a la entrada negativa del restador, para formar la realimentación.

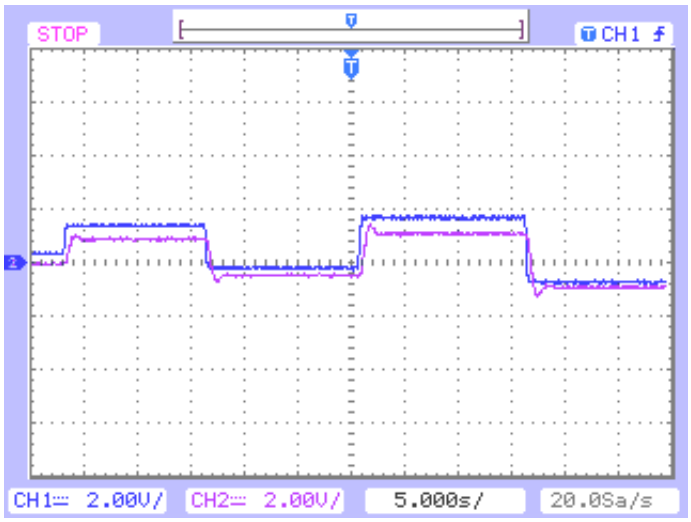


Figura 8.3: Resultados de la prueba realizada con el controlador Proporcional

8.3.2. Controlador P

Para hacer la prueba con este controlador, el periodo de muestreo introducido en la interfaz fue de 10 [ms], es decir, el mínimo y la constante $K_C = 4.7$, la cuál fue obtenida de los diseños del capítulo 7 para esta misma planta.

La prueba realizada para el controlador Proporcional fue la prueba de seguimiento, es decir, observar la salida del sistema frente a un cambio abrupto en la señal de entrada. El comportamiento de la salida de la planta junto a la señal de referencia (*set point*) puede observarse en la figura 8.3

Ya se había advertido, en el capítulo 7, del comportamiento errático en la salida del controlador para una entrada escalón con una amplitud mayor a 2 volts; esto debido a las limitaciones físicas del dispositivo. Sin embargo, se puede observar en la figura 8.3 que la salida sigue a la entrada cuando ésta es un escalón con una amplitud que está dentro del intervalo de valores que puede manejar el dispositivo con un controlador Proporcional.

Como se puede observar en la figura 8.3, la salida del sistema con un controlador proporcional se asemeja mucho a la salida teórica obtenida por medio del programa Matlab®, mostrada en la figura 7.6 en el capítulo 7.

Con respecto del error en estado estable, se puede observar que éste es mayor al pronosticado. La estabilización se logra en el tiempo indicado, mientras que el sobrepaso es mucho menor al 20 %.

Por último, se observó en las pruebas que cuando la amplitud de la señal escalón supera los 3 volts, la salida pasa a ser errática, incluso, la salida es completamente diferente a la entrada. Esto, evidentemente, se debe a las limitaciones físicas de los circuitos electrónicos,

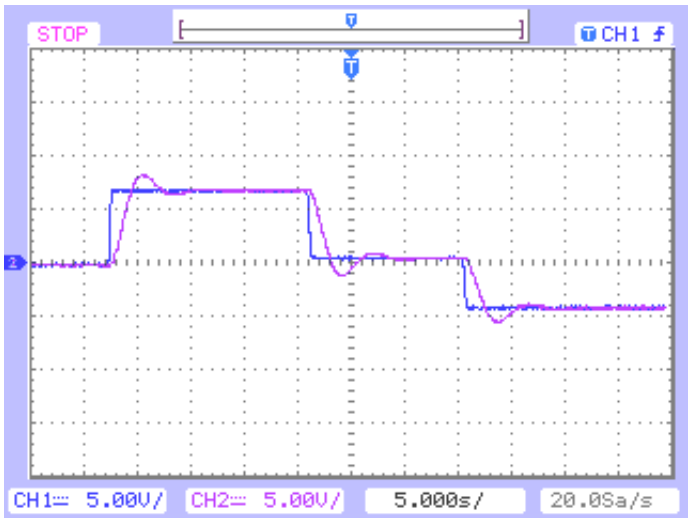


Figura 8.4: Resultados de la prueba realizada con el controlador Proporcional Integral

explicados en el capítulo 7.

8.3.3. Controlador PI

Las condiciones bajo las cuáles se probó el controlador Proporcional Integral fueron:

- Periodo de muestreo introducido en la interfaz: 10 [milisegundos].
- Los parámetros introducidos, fueron los que se calcularon en el capítulo 7, es decir, $K_C = 0.5986$ y $T_i = 0.4027$.

El resultado de la prueba de seguimiento puede observarse en la figura 8.4. De dicha figura se puede concluir que la respuesta del sistema con el controlador en cuestión, es excelente.

Se observa que las condiciones bajo las cuáles se diseñó el controlador se respetan, es decir:

- El error en estado estable es prácticamente nulo.
- El porcentaje de sobrepaso es aproximadamente 20 %.
- El tiempo de asentamiento prácticamente es de 5 segundos, con un criterio del 5 %.

Una comparación entre la figura 8.4 y la figura 7.7 del capítulo 7 muestra el parecido en las respuestas práctica y teórica respectivamente.

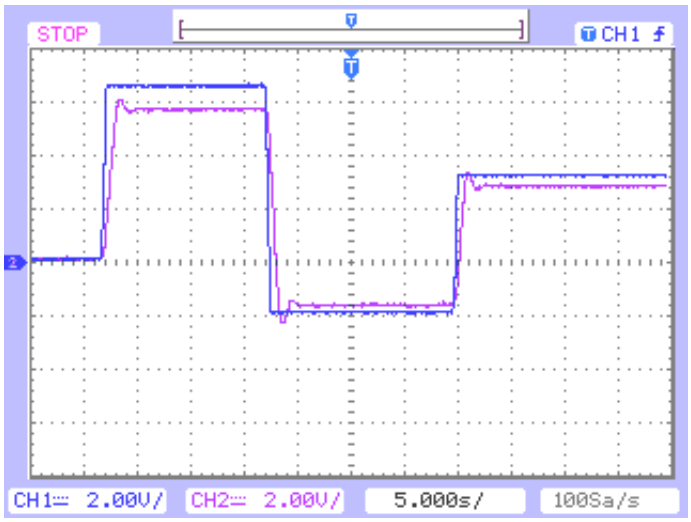


Figura 8.5: Resultados de la prueba realizada con el controlador Proporcional Derivativo

Por último, se menciona que la salida del controlador tuvo un comportamiento errático para incrementos en voltaje muy grandes, es decir, del orden de 9 a 10 volts. Nuevamente, se menciona que esto fue así, debido a las limitaciones físicas de los circuitos que componen al controlador digital.

8.3.4. Controlador PD

Para este controlador se usaron los valores de los parámetros de control encontrados en el capítulo 7; de tal modo que en la interfaz, se introdujeron los siguientes datos:

- Periodo de muestreo introducido en la interfaz: 10 [milisegundos].
- Los parámetros introducidos fueron los que se calcularon en el capítulo 7, es decir, $K_C = 5.338$ y $T_d = 0.0117$.

La salida teórica de la planta con este controlador se muestra en la figura 7.8 del capítulo 7, mientras que la salida de las pruebas realizadas se muestra en la figura 8.5.

En la figura 8.5 se puede observar que la prueba de seguimiento hecha para este controlador es aceptable. La salida del sistema sigue a la entrada de referencia, incluso para cambios abruptos con grandes amplitudes. El tiempo de asentamiento es de aproximadamente 2 segundos. Mientras que el porcentaje de sobrepaso es menor al pronosticado.

Con respecto del error en estado estable, se puede observar que éste no es constante como debería ser. Sin embargo, la variación cae dentro de un intervalo aceptable.

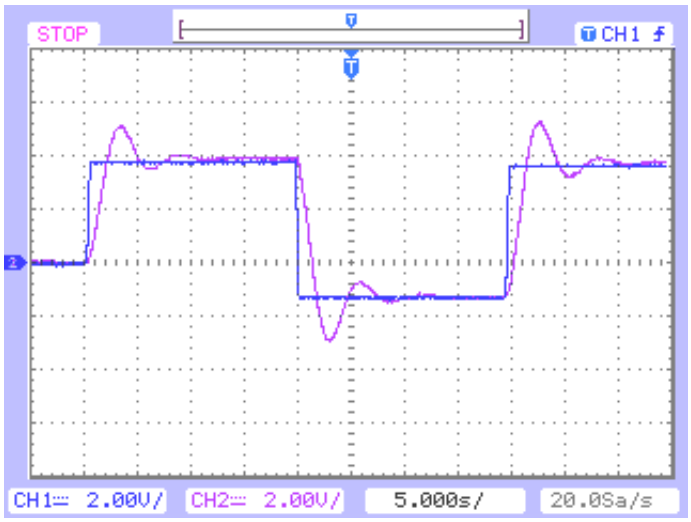


Figura 8.6: Resultados de la prueba realizada con el controlador Proporcional Integral Derivativo

8.3.5. Controlador PID

Este controlador fue probado con los siguientes valores:

- Periodo de muestreo introducido en la interfaz: 10 [milisegundos].
- Los parámetros introducidos fueron los que se calcularon en el capítulo 7, es decir, $K_C = 1$, $T_i = 0.4840$ y $T_d = 0.3345$.

En la figura 7.9 del capítulo 7 se observa la salida esperada cuando el controlador usado es el PID. Se observa que el error en estado estacionario es 0, mientras que el tiempo de levantamiento es muy pequeño. El porcentaje de sobrepaso no rebasa el 20 % y el tiempo de asentamiento es de 5 segundos (con un criterio del 5 %).

El resultado de las pruebas de seguimiento para el controlador en cuestión se muestra en la figura 8.6. En dicha figura, se puede observar que el tiempo de asentamiento es un poco mayor al pedido. Además, el número de oscilaciones es mayor al pronosticado. Sin embargo, la respuesta del controlador es aceptable debido a que la salida de la planta es similar al deseado, es decir, la salida de la planta sigue a la entrada de referencia.

De las pruebas hechas a todos los controladores, se puede concluir que el controlador digital construido hace un trabajo aceptable. Los posibles errores de magnitud, tanto del porcentaje de sobrepaso como del tiempo de asentamiento, caen dentro límites tolerables.

Algunos controladores, como el *ON-OFF*, el P y el PI, son prácticamente iguales a los resultados teóricos. Por tanto, el dispositivo digital construido puede servir para observar el comportamiento de un controlador previamente diseñado, y puede ser usado para apoyo didáctico.

Capítulo 9

CONCLUSIONES

El objetivo de este proyecto de tesis, fue construir un dispositivo de carácter didáctico, en donde los usuarios (estudiantes de la materia Fundamentos de Control de la Facultad de Ingeniería de la UNAM) puedan probar los diseños básicos de controladores, por medio de una interfaz simple, amigable y portable.

Debido al carácter didáctico de este controlador digital, el usuario tiene la oportunidad de usar solamente plantas de tipo electrónico y bajo las limitaciones físicas de las que se ha hablado en capítulos anteriores.

La facilidad de uso de este dispositivo, permite que un estudiante pueda dominar su uso, en muy poco tiempo.

Con respecto a la precisión, tal como se vio en el capítulo precedente, los resultados logrados en las pruebas con el controlador digital construido son aceptables; es decir, los resultados obtenidos son muy aproximados a los obtenidos por medio de una simulación por software. Las diferencias son debidas, principalmente, a ruido electrónico y errores de redondeo en el momento del diseño, sin embargo, estos errores son tolerables.

Es preciso recalcar que existen muchas versiones de controladores comerciales como el que aquí se ha presentado, incluso, con características que lo superan. Sin embargo, se concluye que esta versión de controlador podría mejorarse si se le agregan otras características, por ejemplo, el que en el PC se logren ver los resultados, es decir una gráfica de la respuesta en el dominio del tiempo y de la frecuencia sin necesidad de usar un osciloscopio. Sin embargo, eso queda fuera de los alcances del objetivo de este proyecto.

Por otro lado, aun cuando solamente se mostró el diseño de un controlador básico cuando se conoce la función de transferencia de la planta, se puede usar este mismo controlador para plantas cuyo modelo matemático no se conoce, como sucede la mayoría de las veces en la industria. Para hacer esto, el controlador PID puede ser diseñado por medio de las reglas de Ziegler-Nichols, tal como se mencionó en el capítulo 7.

Apéndice A

LISTADO DE PROGRAMAS

A.1. Programa en el microcontrolador

```
/*
 * Programa que controla una planta utilizando un controlador
 * ON-OFF, P, PI, PD o PID. El usuario escoge el tipo de
 * controlador a utilizar y el periodo de muestreo;  el micro-
 * controlador se encarga de tomar estos parámetros
 * y de realizar la tarea indicada.
 */

// ---------------------------------------
// Declaración de los archivos de cabecera
// ---------------------------------------
#include <stdio.h>
#include <iogp32.h>
#include <float.h>
#include <math.h>
#include <ctype.h>

// ----------------------------------
// Declaración de Variables Globales
// ----------------------------------
float b_0, b_1, b_2, a_1, a_2;
float EP, EANT, EANT2, MP, MANT, MANT2;
float NVTOV, Kc, Ventana;
char BYSAL, Amplitud;
unsigned char Contador,BL;
char CpIndicador,bit;
```

```
// -------------------------
// Declaración de funciones.
// -------------------------

/*
 * Esta función lee el valor analógico del canal 0 y lo convierte a
 * su equivalente digital. El valor equivalente digital se devuelve
 * como una variable unsigned char
 */
unsigned char lee_adc(unsigned char x)
{
     ADSCR = x;
     while(1)
     {
          if((ADSCR & 0x80)==0x80) // Checamos si COCO = 1
               return(ADR);
     }
}



/*
 * Las siguientes funciones ejecutan el diagrama de flujo que
 * resuelve los distintos controladores.
 */

// Controlador PID - Ecuación en diferencias 2do. orden.
void rcad_pid(void)
{
     BL = lee_adc(0x20);
     EP = 0.07843*BL - 10.0;
     MP = b_0*EP + b_1*EANT + b_2*EANT2 + a_1*MANT + a_2*MANT2;
     if(MP > 10)
          MP = 10;
     if(MP < -10)
          MP = -10;
     BYSAL = 12.75*(MP + 10);
     PTA = BYSAL;          // En el puerto A se coloca BYSAL
     MANT2 = MANT;
     MANT = MP;
     EANT2 = EANT;
     EANT = EP;
}
```

```
// Controladores PI y PD - Ecuación en diferencias 1er. orden.
void rcad_pi_pd(void)
{
     BL = lee_adc(0x20);
     EP = 0.07843*BL - 10.0;
     MP = b_0*EP + b_1*EANT + a_1*MANT;
     if(MP > 10)
          MP = 10;
     if(MP < -10)
          MP = -10;
     BYSAL = 12.75*(MP + 10);
     PTA = BYSAL;          // En el puerto A se coloca BYSAL
     MANT = MP;
     EANT = EP;
}

// Controlador P.
void rcad_p(void)
{
     BL = lee_adc(0x20);
     EP = 0.07843*BL - 10.0;
     MP = Kc*EP;
     BYSAL = 12.75*(MP - 10);
     PTA = BYSAL;
}

// Controlador ON-OFF
void rcad_onoff(void)
{
     BL = lee_adc(0x20);
     EP = 0.07843*BL - 10.0;
     if(EP >= (Ventana/2))
     {
          PTA = 0x00;
          asm("bclr #5, 0x03");
          Bandera = 0;
     }
     else
     {
          if(EP <= -(Ventana/2))
          {
               PTA = Amplitud;
               asm("bset #5, 0x03");
               Bandera = 1;
          }
          else // Entonces EP está dentro de la ventana
```

```
          {
               if(Bandera == 0)
               {
                    PTA = 0x00;
                    asm("bclr #5, 0x03");
               }
               else
               {
                    PTA = Amplitud;
                    asm("bset #5, 0x03");
               }
          }
     }
}
// Subrutina de servicio de interrupción por OverFlow del timer.
// El #pragma interrupt_handler le indica al compilador que manovf
// es una función de servicio de interrupción
#pragma interrupt_handler manovf
void manovf(void)
{
     asm("lda 0x20");
     asm("bclr #7,0x20");
     Contador++;
     if(Contador == NVTOV)     // Si ha transcurrido Per.Muestreo
     {
          Contador = 0;
          switch(CpIndicador)
          {
               case 1: // Ejecuta control ON-OFF
                    rcad_onoff();
                    break;
               case 2: // Ejecuta control P
                    rcad_p();
                    break;
               case 3: // Ejecuta control PI
                    rcad_pi_pd();
                    break;
               case 4: // Ejecuta control PD
                    rcad_pi_pd();
                    break;
               case 5: // Ejecuta control PID
                    rcad_pid();
                    break;
          }
     }
}
```

```
// Declaración del vector de usuario asociado al OverFlow
// del timer.
#pragma abs_address:0xFBF2
void (* const _int1[])(void) = {manovf};
#pragma end_abs_address

// Dirección del vector de usuario de RESET.
#pragma abs_address:0xFBFE
int xx = 0x82E3;
#pragma end_abs_address



/*************************************************************
 *
                     PROGRAMA PRINCIPAL.
 *
 *************************************************************/

void main(void)
{
     float Ti, Td, T, N, Tms;
     char *p, Indicador;

     Contador = 0;



     // ===================================
     // Configuración del Microcontrolador.
     // ===================================
     CONFIG1 = 0x01;        //Deshabilitación del watchdog
     SCC1 = 0x40;           // Configuración puerto serial...
     SCC2 = 0x0c;
     SCC3 = 0x80;
     SCS1 = 0x10;
     SCS2 = 0x00;
     SCBR = 0x30;
     CONFIG2=0x00;
     DDRD = 0xFF;           // PUERTO D  ES SALIDA.
     DDRA = 0xFF;           // PUERTO A  ES SALIDA.
     ADCLK = 0x60;          // Reloj externo, div / 8
     ADSCR = 0x20;          // Modo contínuo, canal 0, sin interr.
     asm("bset #7, 0x05");
```

```
while(1)
{
     Indicador = getchar();
     switch(Indicador)
     {
          case 1:
               // ================================
               // Configuración controlador ON-OFF
               // ================================

               // Deshabilitamos interrupción por timer1
               T1SC=0x00;
               printf("\n\rTimer1 deshabilitado...");
               printf("\n\rCONTROLADOR ON-OFF M");
               // Leemos el periodo en milisegundos.
               p=(char *)&Tms;
               *p = getchar();
               *(p+1) = getchar();
               *(p+2) = getchar();
               *(p+3) = getchar();
               printf("\n\rTms Recibido: %f",Tms);
               // Leemos la ventana:
               p=(char *)&Ventana;
               *p = getchar();
               *(p+1) = getchar();
               *(p+2) = getchar();
               *(p+3) = getchar();
               printf("\n\rVentana Recibida: %f",Ventana);
               // Leemos la Amplitud:
               p=(char *)&Amplitud;
               *p = getchar();
               printf("\n\rAmplitud Recibida: %X",Amplitud);
               // Dividimos T / tiempo base(10 ms)
               NVTOV=0.1*Tms;
               printf("\n\rNVTOV: %f ", NVTOV);

               break;

          case 2:
               // ============================
               // Configuración controlador P
               // ============================

               // Deshabilitamos int timer1
               T1SC=0x00;
               printf("\n\rTimer1 deshabilitado...");
```

```
        printf("\n\rCONTROLADOR P");
        // Inicialización de variables.
        EP = 0;       EANT = 0;       EANT2 = 0;
        MP = 0;       MANT = 0;       MANT2 = 0;
        // Leemos el periodo en milisegundos.
        p=(char *)&Tms;
        *p = getchar();
        *(p+1) = getchar();
        *(p+2) = getchar();
        *(p+3) = getchar();
        printf("\n\rTms Recibido: %f",Tms);
        // Leemos el parámetro Kc
        p=(char *)&Kc;
        *p = getchar();
        *(p+1) = getchar();
        *(p+2) = getchar();
        *(p+3) = getchar();
        printf("\n\rKc Recibido: %f",Kc);
        // Dividimos T / tiempo base(10 ms)
        NVTOV = 0.1*Tms;
        printf("\n\rNVTOV: %f ", NVTOV);

        break;


    case 3:
        // ============================
        // Configuración controlador PI
        // ============================

        // Deshabilitamos interrupción timerl
        T1SC=0x00;
        printf("\n\rTimer1 deshabilitado...");
        printf("\n\rCONTROLADOR PI ");
        // Inicialización de variables.
        EP = 0;       EANT = 0;       EANT2 = 0;
        MP = 0;       MANT = 0;       MANT2 = 0;
        // Leemos el periodo en milisegundos.
        p=(char *)&Tms;
        *p = getchar();
        *(p+1) = getchar();
        *(p+2) = getchar();
        *(p+3) = getchar();
        printf("\n\rTms Recibido: %f",Tms);
```

```
            // Leemos el parámetro Kc
            p=(char *)&Kc;
            *p = getchar();
            *(p+1) = getchar();
            *(p+2) = getchar();
            *(p+3) = getchar();
            printf("\n\rKc Recibido: %f",Kc);
            // Leemos el parámetro Ti
            p=(char *)&Ti;
            *p = getchar();
            *(p+1) = getchar();
            *(p+2) = getchar();
            *(p+3) = getchar();
            printf("\n\rTi Recibido: %f",Ti);
            // Convertimos T a segundos.
            T = 0.001*Tms;
            // Dividimos T / tiempo base(10 ms)
            NVTOV = 0.1*Tms;

            // Calculando los valores de los coeficientes.
            // de la ecuación en diferencias.
            // Usando método de Tustin.
            // Observe que está normalizado (a_0 = 1)
            a_1 = 1;
            b_0 = Kc*(T + 2*Ti)/(2*Ti);
            b_1 = Kc*(T - 2*Ti)/(2*Ti);

            // Mostramos los valores de los coeficientes
               calculados.
            printf("\n\r\n\r*** Valores Calculados ***");
            printf("\n\ra_1: %f ", a_1);
            printf("\n\rb_0: %f ", b_0);
            printf("\n\rb_1: %f ", b_1);
            printf("\n\rNVTOV: %f ", NVTOV);

            break;


        case 4:
            // ============================
            // Configuración controlador PD
            // ============================

            // Deshabilitamos interrupción por timerl
            T1SC=0x00;
            printf("\n\rTimer1 deshabilitado...");
```

```
printf("\n\rCONTROLADOR PD");
// Inicialización de variables.
EP = 0;          EANT = 0;          EANT2 = 0;
MP = 0;          MANT = 0;          MANT2 = 0;
// Leemos el periodo en milisegundos.
p=(char *)&Tms;
*p = getchar();
*(p+1) = getchar();
*(p+2) = getchar();
*(p+3) = getchar();
printf("\n\rTms Recibido: %f",Tms);


// Leemos el parámetro Kc
p=(char *)&Kc;
*p = getchar();
*(p+1) = getchar();
*(p+2) = getchar();
*(p+3) = getchar();
printf("\n\rKc Recibido: %f",Kc);

// Leemos el parámetro Td
p=(char *)&Td;
*p = getchar();
*(p+1) = getchar();
*(p+2) = getchar();
*(p+3) = getchar();
printf("\n\rTd Recibido: %f",Td);

// Convertimos T a segundos.
T = 0.001*Tms;
// Dividimos T / tiempo base(10 ms)
NVTOV = 0.1*Tms;

// Calculamos los valores de los coeficientes de
// la ecuación en diferencias
// usando discretización por Tustin.
// Observe que está normalizado (a_0 = 1)
a_1 = -1;
b_0 = Kc*(T + 2*Td)/T;
b_1 = Kc*(T - 2*Td)/T;
```

```
            // Mostramos los valores de los coeficientes
               calculados.
            printf("\n\r\n\r*** Valores Calculados ***");
            printf("\n\ra_1: %f ", a_1);
            printf("\n\rb_0: %f ", b_0);
            printf("\n\rb_1: %f ", b_1);
            printf("\n\rNVTOV: %f ", NVTOV);

            break;

        case 5:
            // ============================
            // Configuración controlador PID
            // ============================

            // Deshabilitamos interrupción por timerl
            T1SC=0x00;
            printf("\n\rTimer1 deshabilitado...");
            printf("\n\rCONTROLADOR PID M");
            // Inicialización de variables.
            EP = 0;          EANT = 0;       EANT2 = 0;
            MP = 0;      MANT = 0;       MANT2 = 0;
            // Leemos el periodo en milisegundos
            p=(char *)&Tms;
            *p = getchar();
            *(p+1) = getchar();
            *(p+2) = getchar();
            *(p+3) = getchar();
            printf("\n\rTms Recibido: %f",Tms);
            // Leemos el parámetro Kc
            p=(char *)&Kc;
            *p = getchar();
            *(p+1) = getchar();
            *(p+2) = getchar();
            *(p+3) = getchar();
            printf("\n\rKc Recibido: %f",Kc);
            // Leemos el parámetro Td
            p=(char *)&Td;
            *p = getchar();
            *(p+1) = getchar();
            *(p+2) = getchar();
            *(p+3) = getchar();
            printf("\n\rTd Recibido: %f",Td);
            // Leemos el parámetro Ti
            p=(char *)&Ti;
            *p = getchar();
```

```
                    *(p+1) = getchar();
                    *(p+2) = getchar();
                    *(p+3) = getchar();
                    printf("\n\rTi Recibido: %f",Ti);
                    // Convertimos T a segundos.
                    T = 0.001*Tms;
                    // Dividimos T / tiempo base(10 ms)
                    NVTOV = 0.1*Tms;
                    // Calculando los valores de los coeficientes.
                    // Estos coeficientes se calculan utilizando
                    // discretización por Tustin.
                    // Observe que está normalizado (a_0 = 1)
                    a_1 = 0;
                    a_2 = 1;
                    b_0 = Kc*(4*Ti*Td + 2*Ti*T + T*T)/(2*T*Ti);
                    b_1 = Kc*(T*T - 4*Ti*Td)/(Ti*T);
                    b_2 = Kc*(4*Ti*Td - 2*Ti*T + T*T)/(2*Ti*T);

                    // Mostramos los valores de los coeficientes
                       calculados

                    printf("\n\r\n\r*** Valores Calculados ***");
                    printf("\n\ra_1: %f ", a_1);
                    printf("\n\ra_2: %f ", a_2);
                    printf("\n\rb_0: %f ", b_0);
                    printf("\n\rb_1: %f ", b_1);
                    printf("\n\rb_2: %f ", b_2);
                    printf("\n\rNVTOV: %f ", NVTOV);

                    break;

               default: continue;
          } // del switch

          // Guardamos el indicador del tipo de control.
          CpIndicador =  Indicador;
          // Habilitamos el timer.
          T1SC = 0x40;
          T1MODH = 0x4E;          // T1MODH:T1MODL = 20000
          T1MODL = 0x20;          // Interrupción cada 10 ms
          asm("cli");
          printf("\n\rTimer1 HABILITADO...");
     } // del while

} // del main...
```

A.2. Interfaz de Java - Activación del dispositivo

Consiste en obtener el dispositivo seleccionado, configurarlo y abrirlo para la comunicación. El dispositivo es configurado para una comunicación 8N1 (8 bits de datos, sin bit de paridad y un bit de paro) con 9600 baudios.

```
StringselectedDevice = (String)DeviceCombo.getSelectedItem();
EnumerationportList = CommPortIdentifier.getPortIdentifiers();
if(serialPort!=null)
{
 serialPort.close();
}
CommPortIdentifier portId;
while(portList.hasMoreElements())
{
 portId = (CommPortIdentifier) portList.nextElement();
 if (portId.getName().equals(selectedDevice))
 {
  try
  {
   serialPort = (SerialPort) portId.open("Simple Write",2000);
  }
  catch(PortInUseException e)
  {
   continue;
  }
  try
  {
   serialOutput = serialPort.getOutputStream();
   serialInput = serialPort.getInputStream();
   lectura.start();
  }
  catch(IOException e)
  {
  }
  try
  {
   serialPort.setSerialPortParams(9600,SerialPort.DATABITS_8,
       SerialPort.STOPBITS_1,SerialPort.PARITY_NONE);
  }
  catch(UnsupportedCommOperationException e)
  {
  }
  try
  {
   serialPort.notifyOnOutputEmpty(true);
```

```
  }
  catch(Exception e)
  {
   System.out.println(e.toString());
   System.exit(-1);
  }
 }
}
```

A.3. Interfaz de Java - Controlador *ON-OFF*

En esta parte del programa de Java se validan los datos introducidos cuando el controlador elegido es el *ON-OFF*. Cuando los datos son válidos se envían al puerto serie de la PC.

```
float periodoOnOff = Float.valueOf(periodoOnOffTField.getText()).
   floatValue();
float supLim = Float.valueOf(supLimTField.getText()).floatValue();
int infLim = Math.round((Float.valueOf(infLimTField.getText())+10)
   *255/20);
if(periodoOnOff <= 0)
 getAvisoLabel().setText("El periodo de muestreo debe ser mayor a 0"
    );
else if (supLim <= 0 || supLim > 10)
{
 getAvisoLabel().setText("La ventana debe de encontrarse en el rango
     de 0 a 10 y ser mayor a 0");
}
else if (infLim < 0 || infLim > 255)
{
 getAvisoLabel().setText("La amplitud debe tener un valor entre -10
    y 10");
}
else
 getAvisoLabel().setText("Microcontrolador programado");

getInvalidInputDialog().pack();
getInvalidInputDialog().setLocationRelativeTo(null);
getInvalidInputDialog().setVisible(true);
int intPeriodo = Float.floatToRawIntBits(periodoOnOff);
int intInfLim = infLim;
int intSupLim = Float.floatToRawIntBits(supLim);
byte[] bytePeriodo = new byte[4];
byte[] byteInfLim = new byte[1];
byte[] byteSupLim = new byte[4];
```

```
bytePeriodo[3] = (byte) (intPeriodo & 0xff);
bytePeriodo[2] = (byte) (intPeriodo >> 8 & 0xff);
bytePeriodo[1] = (byte) (intPeriodo >> 16 & 0xff);
bytePeriodo[0] = (byte) (intPeriodo >> 24 & 0xff);
byteInfLim[0] = (byte) (intInfLim & 0xff);
byteSupLim[3] = (byte) (intSupLim & 0xff);
byteSupLim[2] = (byte) (intSupLim >> 8 & 0xff);
byteSupLim[1] = (byte) (intSupLim >> 16 & 0xff);
byteSupLim[0] = (byte) (intSupLim >> 24 & 0xff);
try
{
 serialOutput.write(1);
 Thread.sleep(1000);
 serialOutput.write(bytePeriodo);
 Thread.sleep(1000);
 serialOutput.write(byteSupLim);
 Thread.sleep(1000);
 serialOutput.write(byteInfLim);
 Thread.sleep(1000);
}
catch(IOException ioe)
{
 ioe.printStackTrace();
}
catch(Exception otras)
{
 otras.printStrackTrace();
}
```

A.4. Interfaz de Java - Controladores P, PI, PD y PID

En esta parte del programa de Java se validan los datos introducidos cuando el controlador elegido es el P, PI, PD o PID. Cuando los datos son válidos se envían al puerto serie de la computadora.

```
float periodo = Float.valueOf(periodoPIDTField.getText());
float kp = Float.valueOf(cteKpTField.getText());
float ki = Float.valueOf(cteKiTField.getText());
float kd = Float.valueOf(cteKdTField.getText());
int intPeriodo = Float.floatToRawIntBits(periodo);
int intKp = Float.floatToRawIntBits(kp);
int intKi = Float.floatToRawIntBits(ki);
int intKd = Float.floatToRawIntBits(kd);
byte[] bytePeriodo = new byte[4];
byte[] byteKp = new byte[4];
```

```
byte[] byteKi = new byte[4];
byte[] byteKd = new byte[4];

if (periodo <= 0) {
 getAvisoLabel().setText("El periodo de muestreo debe ser
    mayor a 0");
 getInvalidInputDialog().setVisible(true);
 return;
} else if (grupoTipoControl.getSelection() == null) {
 getAvisoLabel().setText("Debe seleccionar un tipo de
    control");
 getInvalidInputDialog().setVisible(true);
 return;
}

bytePeriodo[3] = (byte) (intPeriodo & 0xff);
bytePeriodo[2] = (byte) (intPeriodo >> 8 & 0xff);
bytePeriodo[1] = (byte) (intPeriodo >> 16 & 0xff);
bytePeriodo[0] = (byte) (intPeriodo >> 24 & 0xff);

byteKp[3] = (byte) (intKp & 0xff);
byteKp[2] = (byte) (intKp >> 8 & 0xff);
byteKp[1] = (byte) (intKp >> 16 & 0xff);
byteKp[0] = (byte) (intKp >> 24 & 0xff);

byteKd[3] = (byte) (intKd & 0xff);
byteKd[2] = (byte) (intKd >> 8 & 0xff);
byteKd[1] = (byte) (intKd >> 16 & 0xff);
byteKd[0] = (byte) (intKd >> 24 & 0xff);

byteKi[3] = (byte) (intKi & 0xff);
byteKi[2] = (byte) (intKi >> 8 & 0xff);
byteKi[1] = (byte) (intKi >> 16 & 0xff);
byteKi[0] = (byte) (intKi >> 24 & 0xff);

System.out.println(Integer.toHexString(bytePeriodo[0])+" "
  +Integer.toHexString(bytePeriodo[1])+" "
  +Integer.toHexString(bytePeriodo[2])+" "
  +Integer.toHexString(bytePeriodo[3]));

try {


 getAvisoLabel().setText("Microcontrolador programado");
 // System.out.println("Seleccion: "+grupoTipoControl.
    getSelection());
```

```
if (grupoTipoControl.getSelection() == ctrlPRadioButton.
   getModel()) {
 System.out.println("Control P");
 serialOutput.write(2);
 Thread.sleep(1000);
 serialOutput.write(bytePeriodo);
 Thread.sleep(1000);
 serialOutput.write(byteKp);
 Thread.sleep(1000);
} else if (grupoTipoControl.getSelection() ==
   ctrlPDRadioButton
  .getModel()) {
 System.out.println("Control PD");
 serialOutput.write(4);
 Thread.sleep(1000);
 serialOutput.write(bytePeriodo);
 Thread.sleep(1000);
 serialOutput.write(byteKp);
 Thread.sleep(1000);
 serialOutput.write(byteKd);
 Thread.sleep(1000);
} else if (grupoTipoControl.getSelection() ==
   ctrlPIRadioButton
  .getModel()) {
 System.out.println("Control PI "+bytePeriodo.length);
 serialOutput.write(3);
 Thread.sleep(1000);
 serialOutput.write(bytePeriodo);
 Thread.sleep(1000);
 serialOutput.write(byteKp);
 Thread.sleep(1000);
 serialOutput.write(byteKi);
 Thread.sleep(1000);
} else if (grupoTipoControl.getSelection() ==
   ctrlPIDRadioButton
  .getModel()) {
 System.out.println("Control PID");
 serialOutput.write(5);
 Thread.sleep(1000);
 serialOutput.write(bytePeriodo);
 Thread.sleep(1000);
 serialOutput.write(byteKp); // Kc
 Thread.sleep(1000);
 serialOutput.write(byteKd); // Kd
 Thread.sleep(1000);
 serialOutput.write(byteKi); // Ki
```

```
          Thread.sleep(1000);
         }
        } catch (IOException ioe) {

        } catch (Exception otras) {

        }
```

A.5. Programa en matlab para sintonización de controladores PID por medio del método de Ziegler-Nichols

```
% Programa en matlab para sintonizar un controlador PID por el
% método de Ziegler-Nichols. Primer método (curva de reacción).
%
% Primero se encuentra la respuesta a escalón unitario de la planta.
% Debe ser tipo "S". A partir de ahí, se encuentra L y T
%
% Para el controlador P
% KP = T/L,  Ti = oo , Td = 0
%
% Para el controlador PI
% KP = 0.9(T/L), Ti = L/0.3
%
% Para el controlador PID
% KP = 1.2(T/L), Ti = 2L, Td = 0.5L
%
%
% La FT del controlador es
%
%              (s + 1/L)^2
% gc = 0.6 T --------------
%                  s

% FT de la planta:
gp = zpk([],[-2.67 -1.14], 3.04)

% Encontrando la respuesta a escalón unitario de la planta
% en lazo abierto.
[y,t] = step(gp);

% Encontrando el máximo de la primera derivada de 'y', el cuál será
% su punto de inflexión, dado que 'y' tiene forma de S.
% Dy nos indica el valor del máximo (en otras palabras la pendiente
% de la curva 'y' en el punto de inflexión).
```

```
% t_c indica el índice del vector 't' en donde sucede el punto de
% inflexión. En otras palabras, el valor de t cuando sucede el punto
% de inflexión: la 't' crítica.

[Dy, t_c] = max(diff(y)./diff(t));

% Encontrando el valor de 't' cuando la pendiente en t_c corta al
% eje de las 't':
% Se hace por medio de la ecuación de la recta punto pendiente: La
% pendiente es Dy y el punto es (t(t_c), y(t_c))
% y - y(t_c) =  Dy (t - t(t_c))
%
% para este caso particular, necesitamos encontrar el valor de t=t0
% cuando y=0. Entonces, al despejar t de la ec. anterior:

t0 = t(t_c) - y(t_c)/Dy


% Encontrando el valor de 't' cuando la pendiente en t_c corta a la
% recta y=1:
% Se hace por medio de la ecuación de la recta punto pendiente: La
% pendiente es Dy y el punto es (t(t_c), y(t_c))
% y - y(t_c) =  Dy (t - t(t_c))
%
% para este caso particular, necesitamos encontrar el valor de t=t1
   cuando
% y=1, Entonces, al despejar t de la ec. anterior:

t1 = t(t_c) + (1 - y(t_c))/Dy

% Una vez que tenemos los valores de t0 y t1, se calculan los
% valores de L y T.

T = t1 - t0
L = t0

% Los valores de los parámetros del controlador PID se calculan de
% acuerdo a la tabla mostrada en el capítulo 7.
%
% la FT del controlador gc pid es:

gc = zpk([-1/L -1/L], [0], [0.6*T])
```

```
% El sistema en lazo cerrado es
siss= feedback(gp*gc,1)

% La gráfica de la respuesta a escalón unitario es:
step(siss,10)
```

Apéndice B

GUÍA DE USO DEL DISPOSITIVO

B.1. Instalación de la interfaz

Para poder hacer uso del controlador, es necesario contar con una PC/laptop con las siguientes características mínimas:

- Microprocesador Pentium®.
- Sistema operativo Windows® 98 en adelante.
- Memoria RAM de 256[MB].
- El espacio en disco duro depende de si ya se tiene instalada la Máquina Virtual de Java. En cuyo caso, solamente se instalará el programa interfaz. Si no, será necesario el espacio suficiente para instalar la Máquina Virtual de Java.
- Puerto serial disponible. En su defecto un puerto USB libre y un cable convertidor de USB a serial.

Para aquellas PC's/Laptops que no cuenten con un puerto serial disponible con conexión DB9 macho, se puede usar un puerto USB con una interfaz de puerto serial. Evidentemente, antes de usar el puerto USB, es necesario instalar un programa controlador, para que el sistema operativo pueda reconocer el puerto serial y la transmisión de datos se haga correctamente. El programa controlador, debe venir junto con el cable convertidor USB a serial.

Es importante mencionar aquí, que debido a la característica portable de Java, el controlador digital construido puede ser usado tanto en ambientes Windows® como en otros sistemas operativos, por ejemplo GNU/Linux. En este proyecto de tesis solamente se ha probado el programa interfaz en el sistema operativo Windows® XP.

En las siguientes subsecciones, se mencionan los pasos que se tienen que seguir para instalar correctamente la máquina virtual de java — en caso de que no se tenga instalada — y el manual de usuario del controlador digital construido.

B.1.1. Instalación del programa interfaz

Para usar el programa interfaz, es necesario contar con un programa conocido como *Máquina Virtual de Java*, por lo que el primer paso para usar el controlador digital construido es tener esa aplicación corriendo en la PC o Laptop que se usará.

Antes de poder hacer uso de la aplicación es necesario seguir los siguientes pasos:

1. Instalación de la máquina virtual de Java.
2. Instalación de las librerías de *jigloo* y la aplicación.

Instalación de la Máquina Virtual de Java (JRE)

La aplicación está escrita en Java, y para su ejecución es necesario instalar el ambiente de ejecución de Java en su versión 1.5 o superior. Éste puede obtenerse desde el portal de Sun® *http://java.sun.com/*.

Se deben seguir las instrucciones de instalación presentadas en el sitio, y notar donde se instala.

Instalación de la aplicación

La aplicación debe ser copiada, junto con sus librerías extras en un directorio conocido. Por ejemplo:

```
C:\Aplicaciones\PCD\
```

Conexión

La computadora y el controlador digital se conectan mediante un cable serial o mediante un cable serial y un convertidor serial-USB para el caso de las PC's que no cuenten con un puerto serial.

B.1.2. Guía de uso de la aplicación

La aplicación se ejecuta mediante la siguiente línea de comando, que debe ser ajustada para el sistema operativo en cuestión y para la localización de las diversas librerías de clases de Java:

Para el caso en que el sistema operativo sea Windows (de la versión 98 en adelante) se puede usar la siguiente línea:

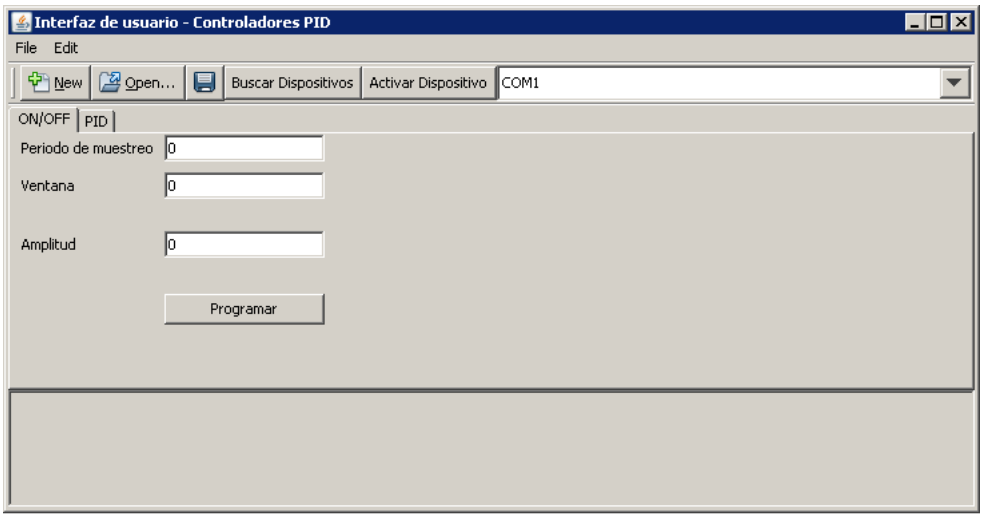


Figura B.1: Ventana de Inicio de la aplicación.

```
java -cp C:\Aplicaciones\PCD\jigloo_test.jar:
C:\Aplicaciones\PCD\appFramework-1.0.jar:
C:\Aplicaciones\PCD\forms-1.1.0.jar:
C:\Aplicaciones\PCD\jnlp.jar:
C:\Aplicaciones\PCD\looks-2.1.4.jar:
C:\Aplicaciones\PCD\RXTXcomm.jar exaple.swing.prueba.TesisGUI2
```

Esto puede colocarse en una sola línea en un archivo ejecutable de procesamiento por lotes y añadirse al escritorio o al menú del sistema para accesarlo de manera directa.

Al arrancar la aplicación se presenta una ventana, como el de la figura B.1.

Esta ventana cuenta con varias opciones. Lo primero que tiene que hacer el usuario es activar el puerto serial a usar. La forma de hacerlo se presenta en la siguiente subsección.

B.1.3. Selección del puerto de comunicación

Se procede a presionar el botón etiquetado *Buscar Dispositivos,* el cual rellena la lista que inicialmente aparece vacía junto al botón etiquetado *Activar Dispositivo.* Entonces se puede seleccionar el dispositivo apropiado y se presiona el botón *Activar Dispositivo.* Lo anterior inicializa la comunicación con el controlador digital, para que éste pueda ser programado.

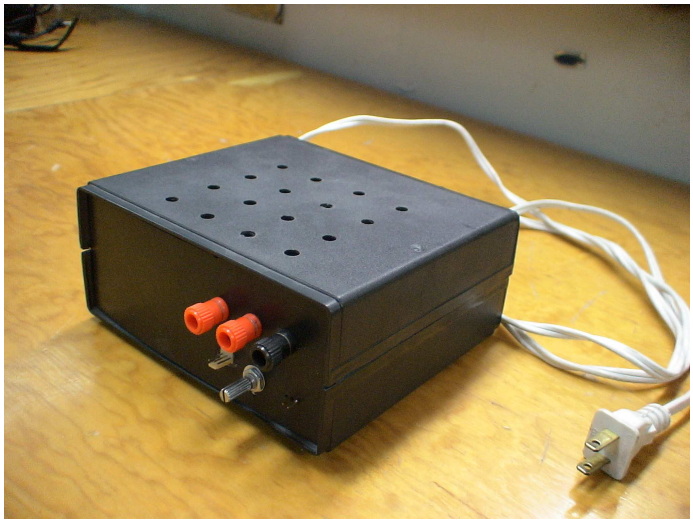


Figura B.2: El controlador digital construido.

Una vez que se tiene activado el puerto y decidido el controlador a simular, se puede programar el dispositivo.

B.1.4. Selección del tipo de controlador y programación del controlador

Para programar el controlador para ejecutar un tipo de control, se selecciona primero la pestaña para el tipo de control deseado, *ON-OFF* o PID. Si se selecciona *ON-OFF*, entonces se procede a rellenar los campos con los valores con que va a ser programado el controlador, y se presiona el botón programar, que inicia la comunicación con el controlador digital y envía los datos.

Si se va a programar un control P, PI, PD o PID, se selecciona la pestaña PID y se selecciona el tipo de control deseado. Entonces se rellenan los campos con los valores de constantes deseados y se selecciona uno de los cuatro controladores listados. Después se procede a presionar el botón *Programar* para que el controlador digital comience a simular el control del tipo seleccionado.

La ejecución del control inicia en el instante en el que el dispositivo se termina de programar.

B.2. Vistas del dispositivo construido

En este último apartado se presentan algunas fotografías tomadas al dispositivo construido. Eso se observa en las figuras B.2, B.3, B.4 y B.5, en donde se ven las conexiones hechas con la planta y con una PC/Laptop.

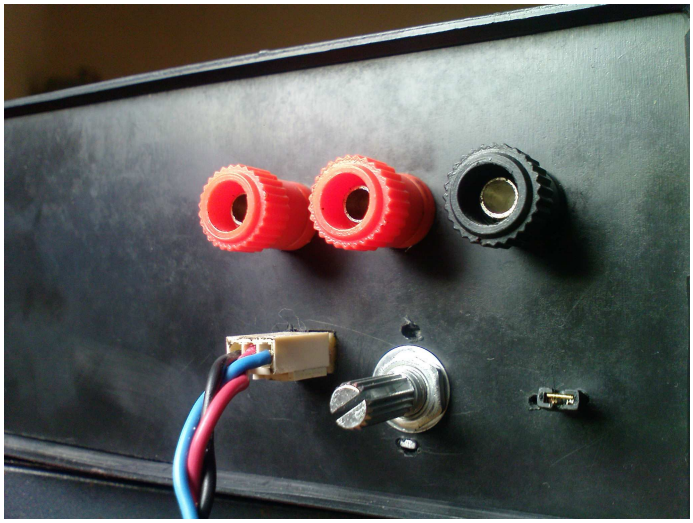


Figura B.3: Vista frontal del dispositivo construido.

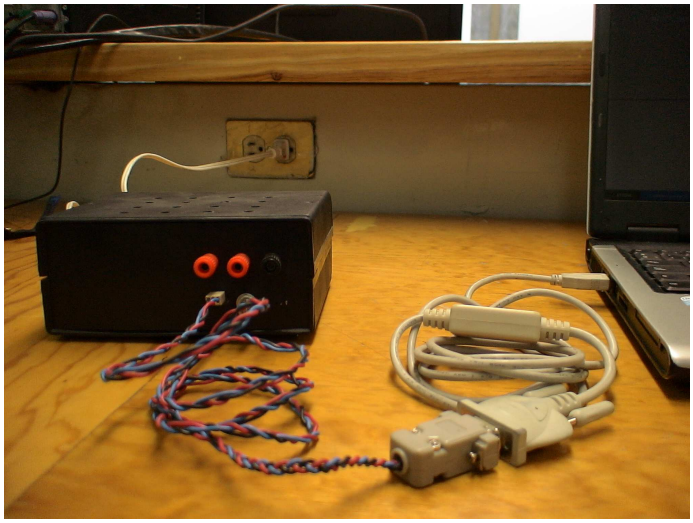


Figura B.4: Conexión con una PC/Laptop.

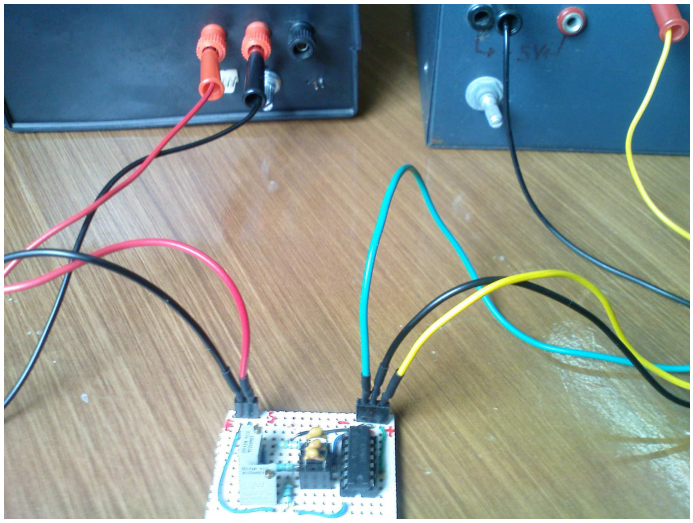


Figura B.5: Conexión con la planta de prueba y una fuente de alimentación.

Apéndice C

GLOSARIO

En este apéndice se muestra un glosario con las definiciones de varios términos utilizados en la teoría de control y en el Análisis de Sistemas y señales que fueron usados en la presente tesis.

Ancho de Banda : La frecuencia a la cual la magnitud de la respuesta en frecuencia es -3 [dB] por debajo de la magnitud en la frecuencia cero.

Cero : Aquellos valores de de la variable de la Transformada de Laplace, s, que hacen que la función de transferencia se vuelva cero.

Error : Es la diferencia entre la entrada de referencia y la salida de un sistema.

Error de Cuantización : Para sistemas lineales, al error asociado con la digitalización de señales, como resultado de diferencias finitas entre los niveles de cuantización.

Estabilidad : A la característica de un sistema definido por una respuesta natural que cae a cero conforme el tiempo tiende a infinito.

Función de Transferencia : Es la transformada de Laplace de la relación de la entrada y la salida de un sistema.

Ganancia : La relación entre la salida y la entrada. Generalmente se usa para describir la amplificación en estado estable de la magnitud de entradas senoidales, incluyendo DC.

Planta o Proceso : Subsistema cuya salida está siendo controlada por el sistema.

Polo : Son valores de s que hacen que la función de transferencia se vuelva infinita.

Polos dominantes : Aquellos polos que predominantemente generan la respuesta transitoria.

Realimentación : Un camino a través del cual una señal fluye de regreso a una señal previa en el camino de trayectoria directa con objeto de ser sumado o restado.

Relación de amortiguamiento : La relación entre la frecuencia de decaimiento exponencial y la frecuencia natural.

Respuesta de Entrada Cero : Es la parte de la respuesta total de un sistema que depende únicamente de sus condiciones iniciales y no de entrada alguna.

Respuesta de Estado Cero : La parte de la respuesta total de un sistema que depende únicamente de de la entrada y no de las condiciones iniciales.

Respuesta de Estado Estable : Para sistemas lineales, la parte de la respuesta total debida únicamente a la entrada. Tiene típicamente la misma forma que la entrada y sus derivadas.

Respuesta Transitoria : Es la parte de la respuesta total de un sistema, debido al sistema mismo y a la forma en que ésta adquiere y disipa la energía. En sistemas estables, es la parte de la respuesta que se grafica antes de la respuesta en estado estable.

Sistema de Lazo Abierto : Un sistema que no monitorea su salida ni corrige las posibles alteraciones a la misma.

Sistema de Lazo Cerrado : Un sistema que monitorea su salida y corrige las posibles alteraciones. Se caracteriza por un camino de realimentación desde su salida.

Subsistema : Es un sistema que pertenece a un sistema más grande. Generalmente se representa por un bloque en un diagrama de bloques.

Tiempo de Asentamiento : Es el tiempo necesario para que la respuesta a una entrada escalón alcance y permanezca dentro de $\pm 3\,\%$ del valor de la respuesta en estado estable. El porcentaje puede variar dependiendo del criterio empleado.

Tipo de Sistema : Es el número de integradores puros en la función de transferencia de un sistema.

BIBLIOGRAFÍA

[1] Gregory B.A. *Instrumentación eléctrica y sistemas de medida.* Gustavo Gili, S.A., Barcelona, 1984.

[2] John J. D'Azzo and Constantine H. Houpis. *Sistemas Realimentados de Control (Análisis y Síntesis).* Paraninfo S.A., Madrid, tercera edition, 1980.

[3] George Ellis. *Control Systems Design Guide. A practical Guide.* Elsevier, Academic Press, California S.F. USA, third edition, 2004.

[4] Texas Instruments. *JFET-input operational amplifiers. TL084.*

[5] Benjamin C. Kuo. *Sistemas de Control Automático.* Pearson, Prentice Hall, séptima edition, 1996.

[6] William Levine S. *Control System Fundamentals.* CRC, Press LLC, 2000.

[7] Apostol Tom M. *Calculus, One-Variable Calculus with an Introduction to Linear Algebra.* John Wiley & Sons, Inc., second edition, 1964.

[8] Mariano Mataix Lurda and Miguel Mataix Hidalgo. *Diccionario Inglés-Español de Electrónica, Informática y Energía Nuclear.* Díaz de Santos, Madrid, España.

[9] Motorola.com/Semiconductors. *68HC908GP32/H Technical Data, Rev 6, 8/2002.*

[10] Norman S. Nise. *Control Systems Engineering.* John Wiley & Sons, Inc., fifth edition, 2008.

[11] Katsuhiko Ogata. *Dinámica de Sistemas.* Prentice-Hall Hispanoamericana, S. A., primera edition, 1987.

[12] Katsuhiko Ogata. *Ingeniería de Control Moderna.* Prentice-Hall Hispanoamericana, S. A., segunda edition, 1993.

[13] National Semiconductor. *DAC0800/DAC0802, 8-Bit Digital-to-Analog Converters.*

[14] National Semiconductor. *LM78XX Series Voltages Regulators.*

[15] National Semiconductor. *LM79XX Series 3-Terminal Negative Regulators*.

[16] Robert C. Weyrick. *Introducción al Control Automático*. Gustavo Gili, S.A., Barcelona, 1977.