UNIVERSITAT POLITÈCNICA DE VALÈNCIA
CAMPUS D'ALCOI

TRABAJO FIN DE GRADO

Control digital en el espacio de estados de un prototipo de motor de corriente continua con Arduino Due

Alumno: **Javier Ferrandiz Brotons**
Tutor: Adolfo Hilario Caballero

GRADO EN INGENIERÍA ELÉCTRICA

Convocatoria de defensa: **julio de 2016**

Resumen

Se diseñará un control digital por realimentación de la salida (observador + controlador) con acción integral sobre la velocidad de un motor de corriente continua. El modelo del motor se indentificará a partir de ensayos experimentales. El control se validará con LabVIEW. La implementación digital final se realizará con Arduino Due. Se analizará la aplicación del filtro de Kalman en el diseño del controlador por realimentación de la salida.

Quiero agradecer a Adolfo Hilario Caballero, profesor de Ingeniería de Control en la UPV Campus de Alcoy, por su ayuda a la hora de realizar este Trabajo Fin de Grado, prestándome material docente y los diagramas de bloques utilizados en este documento.

Índice general

Índice de figuras

Capítulo 1

Introducción

En el siguiente documento se va a explicar cómo realizar el control digital de la velocidad de un motor de corriente continua mediante el espacio de estados, utilizando la realimentación de la salida con acción integral. Además se analizará que ocurre al aplicar este método.

El periodo clásico de la teoría de control, caracterizado por el análisis del dominio de la frecuencia, se mantiene con fuerza actualmente, y ahora estamos en una fase "neo-clásica" – con el desarrollo de varias técnicas sofisticadas para sistemas multivariables. Pero concurrente con ello, está el periodo moderno, que empezó a finales de los años 50 y principios de los 60.

La teoría moderna de control fue introducida por los soviéticos con el lanzamiento del *Sputnik* en 1957. Este logro de la tecnología soviética centró la atención de los científicos e ingenieros en general, y a la comunidad del control automático en particular, sobretodo hacia la URSS. Por consenso global, Moscú fue la localización apropiada para el primer congreso de la Federación Internacional del Control Automático en 1960.

Los científicos e ingenieros de sistemas de control descubrieron un enfoque diferente a la teoría de control que el enfoque con el que estaban familiarizados. Las ecuaciones diferenciales fueron reemplazadas por funciones de transferencia para describir la dinámica de los procesos.

En pocos años de esfuerzo frenético, la teoría de control occidental había absorbido y dominado este nuevo enfoque del "espacio de estado" para el análisis y diseño de los sistemas de control, los cuales ahora se han convertido en la base de gran parte de la teoría de control moderna.

El concepto de espacio de estado ha ocasionado un impacto enorme en el pensamiento de estos científicos e ingenieros de control, que trabajan en la frontera de la tecnología. Estos conceptos han sido también utilizados con bastante éxito en un gran número de proyectos de alta tecnología–el proyecto del US Apollo es un buen ejemplo de ello. Sin embargo, la mayoría de los sistemas de control implementados en la actualidad son diseñados por métodos de una época anterior.

Muchos ingenieros de control colegiados en antiguos métodos han sentido que el enfoque del espacio de estado moderno es matemáticamente esotérico y más adecuado para la investigación avanzada que para el diseño de sistemas de control prácticos.

Gran parte del diseño práctico de la ingeniería está realizado con la ayuda de los computadores. Los sistemas de control no son una excepción. No sólo son equipos utilizados en línea, imple-

mentación en tiempo real de leyes de control de realimentación–en aplicaciones tan diversas como los pilotos automáticos de los aviones y controles de procesos químicos–pero son también utilizados extensamente para realizar los cálculos de diseño. En efecto, uno de las mayores ventajas del diseño del espacio de estado sobre el dominio de la frecuencia es que los primeros son más adecuados para la implementación por parte de los equipos digitales (Friedland 2012).

Las ventajas de la teoría moderna frente a la clásica son:

- Se puede representar de forma sencilla sistemas MIMO (*Multiple-Input, Multiple-Output*), sistemas con varias entradas y varias salidas.
- Teoría más adecuada para tratar no linealidades muy acusadas de la planta.
- El diseño se puede abordar mediante síntesis.

No obstante a las ventajas que presenta, como ya se ha dicho la mayoría de sistemas de control que se utilizan en la actualidad se basan en la teoría clásica. Esto se debe a la complejidad matemática que presenta la teoría moderna.

1.1 Espacio de estado

Los métodos de espacio de estado son la piedra angular de la teoría de control moderna. La principal característica del espacio de estado es la caracterización de los procesos de interés por ecuaciones diferenciales en vez de funciones de transferencia. Esto podría parecer como un regreso al periodo clásico donde las ecuaciones diferenciales también constituían los medios de representar el comportamiento de los procesos dinámicos. Pero en el periodo clásico los procesos eran simples suficiente para ser caracterizados por una única ecuación diferencial de orden pequeño. En el enfoque moderno el proceso está caracterizado por sistemas de orden n en el espacio de estados, que tendrán n ecuaciones lineales de primer orden. En principio no hay límite para el orden de los sistemas y en la práctica el único límite para el orden es que la disponibilidad del software informático sea capaz de realizar los cálculos necesarios de forma fiable.

El espacio de estados es el conjunto de todos los posibles estados de un sistema lineal, cada uno de los estados corresponde a un único punto en el espacio de estados.

Los sistemas lineales que aparecen tienen la siguiente forma, donde se puede destacar $u(t)$ que es la entrada del sistema, $y(t)$ que es la salida del sistema, $x_i(t)$ que son las variables de estado, y las constantes a_i, b_i, c_i y d que son parámetros que definen la representación en el espacio de estados.

$$\begin{aligned}
\dot{x}_1(t) &= a_{11}x_1(t) + a_{12}x_2(t) + \ldots + a_{1n}x_n(t) + b_1u(t) \\
\dot{x}_2(t) &= a_{21}x_1(t) + a_{22}x_2(t) + \ldots + a_{2n}x_n(t) + b_2u(t) \\
&\vdots \\
\dot{x}_n(t) &= a_{n1}x_1(t) + a_{n1}x_2(t) + \ldots + a_{nn}x_n(t) + b_nu(t)
\end{aligned} \tag{1.1}$$

$$y(t) = c_1x_1(t) + c_2x_2(t) + \ldots + c_nx_n(t) + du(t) \tag{1.2}$$

Considerando la ecuación (1.1) y (1.2) podemos obtener la representación matricial del espacio de estados:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{1.3}$$

Se puede ver que $x(t)$ es el vector de estado y $\dot{x}(t)$ es la derivada del vector de estado. A, B, C y D son matrices donde A es la matriz de estado, B la matriz de entrada, C la matriz de salida y D la matriz de transferencia directa. En sistemas donde sólo existe una entrada y una salida, D es un escalar. Además, cabe considerar que en la mayoría de los sistemas utilizados $D = 0$. En la figura 1.1 se puede ver el diagrama de bloques de la representación matricial de la ecuación (1.3).

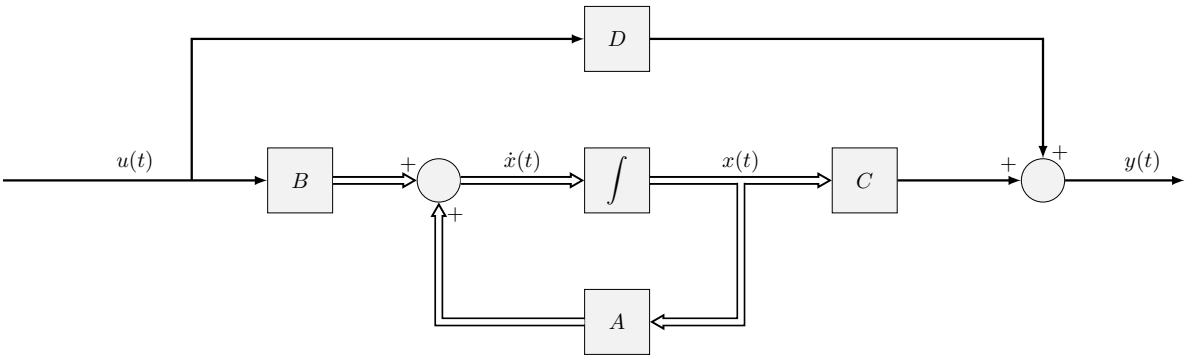


Figura 1.1: Diagrama de bloques de la representación matricial en el espacio de estados

Hay infinitos conjuntos de variables de estado, por lo que puede haber infinitos conjuntos de matrices para representar el mismo sistema.

Existen 3 métodos diferentes para el cálculo de las ecuaciones de estado de un sistema lineal. Estos son:

- Cuando se conocen las ecuaciones físicas del sistema.
- Cuando se conoce el diagrama de bloques detallado del sistema lineal.
- Cuando solo se dispone de la ecuación diferencial o la función de transferencia.

En esta memoria se analizará el tercer método ya que se ha hecho una simulación en *LabVIEW* y en *MATLAB* para la obtención de la función de transferencia correspondiente a nuestro motor, sección 3.1.

1.2 Rudolf E. Kalman

En la teoría de control moderna se puede hablar de muchos científicos que influyeron en el desarrollo de la misma. Uno de los que más contribuyeron en esta tarea fue Rudolf Kalman, conocido por el "Filtro de Kalman" y los conceptos de controlabilidad y observabilidad.

Rudolf Emil Kalman nació en Budapest en 1930, hijo de un ingeniero eléctrico decidió seguir los pasos de su padre. Durante la Segunda Guerra Mundial emigró a Estados Unidos, donde se doctoró en el M.I.T. en Ingeniería Eléctrica en 1954. Su interés por los sistemas de control fue aumentando y con el paso de los años hizo contribuciones importantes en el diseño de sistemas de control lineales de datos muestreados.

A partir de 1958, y hasta 1964, Kalman trabajó en el RIAS (Research Institute for Advanced Study), y fue allí donde hizo algunas de sus contribuciones más importantes para los sistemas de control moderno. Sus publicaciones durante estos años muestran la creatividad y su búsqueda por la unificación de la teoría de control. Entre ellas destacan los conceptos de observabilidad y controlabilidad, que ayudaron a poner bases teóricas sólidas de algunos de los aspectos estructurales más importantes de los sistemas de ingeniería. Estos aspectos son una forma de explicar por qué un método de diseñar compensadores para sistemas inestables cancelando polos inestables por ceros en el semiplano derecho está condenado a incluso fallar si la cancelación es perfecta.

El concepto de controlabilidad se define como un sistema completamente controlable si existe un control sin restricción $u(t)$ que puede llevar cualquier estado inicial $x(t_0)$ a cualquier otro estado deseado $x(t)$ en un tiempo finito $t_0 \leq t \leq t_1$. Por lo que respecta al concepto de observabilidad, un sistema es completamente observable si y sólo si existe un tiempo finito T de forma que el estado inicial $x(0)$ se pueda determinar a partir de la observación de la historia $y(t)$ dado el control $u(t)$.

El principal problema de la observabilidad es que tiene muchas aplicaciones importantes. Si un sistema es observable, no hay sistemas dinámicos internos, por tanto, podemos entender todo lo que está pasando a través de la observación de las entradas y salidas. El problema de la observabilidad es de un interés significativo debido a que ésta determinará si un conjunto de sensores es suficiente para controlar nuestro sistema.

Se considera el sistema de la ecuación (1.3) con $D = 0$ para simplificar el sistema:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned}$$

Por lo tanto, si queremos que un sistema sea flexible y nos permita colocar todos los polos arbitrariamente, el sistema ha de ser controlable y observable.

También fue durante su estancia en el RIAS cuando Kalman desarrolló lo que quizás sea su más conocida contribución, el llamado "Filtro de Kalman". Obtuvo resultados en tiempo discreto (datos muestreados) a finales de 1958 y principios de 1959. Combinó trabajos fundamentales en el filtrado de Wiener, Bode y otros con el enfoque moderno del espacio de estados. La solución del problema del tiempo discreto le permitió llegar al problema del tiempo continuo, y en 1960-1961 desarrolló la versión de tiempo continuo del "Filtro de Kalman".

El filtro de Kalman, y su posterior extensión a los problemas no-lineales, representa quizás la aplicación más importante de la teoría de control moderna. Podemos ver esta aplicación en la navegación y control de vehículos espaciales (Apollo), algoritmos para el seguimiento por radar, control de procesos, y sistemas socioeconómicos. Su popular aplicación es debida al hecho de que una computadora digital puede realizar la fase de diseño tan bien como la implementación de la misma.

1.3 Tiempo discreto

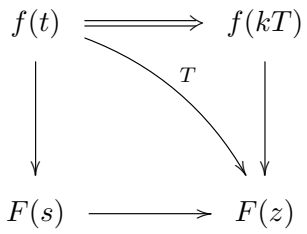
Un computador para obtener información de lo que ocurre en el exterior toma muestras periódicas cada T segundos (periodo de muestreo) de las señales analógicas que proporcionan los sensores y las convierte en valores que se pueden representar de forma digital.

Como el computador tiene que realizar transformaciones de analógico a digital, y viceversa, dispone de dos convertidores: ADC (*Analog to Digital Converter*) y DAC (*Digital to Analog Converter*).

La salida del convertidor ADC es una secuencia debido a que el computador trabaja en tiempo discreto, es decir, toma valores de la señal en saltos discretos del tiempo que corresponde con el tiempo de muestreo T. La entrada al convertidor es una señal en tiempo continuo llamada $y(t)$.

La CPU (*Central Processing Unit*) utiliza la secuencia que se obtiene a la salida del convertidor ADC $y(kT)$ y la señal de referencia $r(kT)$. Esto proporciona la señal $u(kT)$ que es una señal digital y el convertidor DAC la convierte otra vez en una señal analógica $u(t)$. Pero para poder trabajar en tiempo discreto es necesario que la CPU pueda pasar la señal en tiempo continuo a tiempo discreto.

Para ello, hay que conseguir que la transformada de Laplace, que convierte ecuaciones integro-diferenciales en ecuaciones algebraicas, pase a ser una transformada z, que convierte ecuaciones en diferencias en ecuaciones algebraicas simplificando el análisis en tiempo discreto. Se puede pasar de una a la otra directamente como podemos ver a continuación:



donde $f(t)$ es la función en tiempo continuo, $F(s)$ la transformada de Laplace de la función $f(t)$, $f(kT)$ es la función en tiempo continuo, $F(z)$ la transformada z de $f(kT)$, y como ya se ha dicho con anterioridad, T es el periodo de muestreo. El objetivo es obtener $F(z)$ ya que a partir de la transformada z se pueden calcular las ecuaciones en diferencias necesarias para realizar la implementación en el computador.

1.4 Algoritmo anti-windup

El efecto *windup* (*integral windup* en inglés) es un comportamiento no deseado del controlador que se produce cuando coinciden las siguientes circunstancias:

- El controlador incluye acción integral.
- La acción de control no puede superar unos valores límite (saturación).
- La señal de error se mantiene en valores elevados durante mucho tiempo.

Las dos primeras circunstancias se dan prácticamente siempre. La tercera es típica de sistemas muy lentos y en los que los cambios de referencia pueden ser muy grandes.

La acción de control generada por un controlador no puede superar unos determinados límites. Estos límites pueden ser restricciones del propio controlador o del actuador al que está conectado. En la figura 1.2 se representa gráficamente el diagrama de bloques del controlador con saturación sobre la señal $v(k)$.

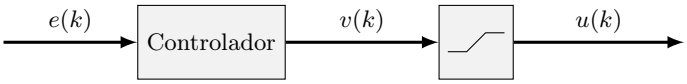


Figura 1.2: Diagrama de bloques del controlador con saturación de la acción de control

Observando la ecuación (1.4) se puede ver que la salida de control $u(k)$ será exactamente igual a la calculada por el controlador si $v(k)$ está entre el valor máximo y el mínimo. Si el valor calculado supera el valor máximo por encima, la salida se mantendrá en el máximo. Si supera por abajo el valor mínimo, se mantendrá en el mínimo.

$$u = \text{sat}(v) = \begin{cases} v, & \text{si } u_{min} \leq v \leq u_{max} \\ u_{max}, & \text{si } v > u_{max} \\ u_{min}, & \text{si } v < u_{min} \end{cases} \tag{1.4}$$

Es decir, el algoritmo del controlador da lugar a un valor cualquiera para la acción de control $v(k)$, el valor efectivo de la acción de control aplicada al sistema estará dentro de unos límites: $u(k) \in [u_{min}, u_{max}]$.

Existen dos algoritmos $anti - windup$ que nos ayudan a corregir este efecto:

- *Back calculation.*
- *Clamping.*

1.4.1 Algoritmo *Back calculation*

Cuando se utiliza el algoritmo *Back calculation* la acción integral en tiempo continuo se calcula de la siguiente forma:

$$u_i(t) = \int_0^t \left(K_i\, e(t) - K_b\, [v(t) - u(t)]\right) dt$$

La acción de control integral que quedaría aplicando el método de *Forward Euler* a la aproximación de la derivada es:

$$u_i(k+1) = u_i(k) + K_i\, T\, e(k) - K_b\, T\, [v(k) - u(k)]$$

Por lo que:

- Cuando la acción de control $u(k)$ no está saturada, es decir $u(k) = v(k)$, el integrador actúa normalmente:

$$u_i(k+1) = u_i(k) + K_i\, e(k)$$

- Cuando la acción de control se satura la señal de entrada del integrador es la resta de una señal proporcional al error y otra señal proporcional a la diferencia entre la acción de control no saturada $v(k)$ y la acción de control saturada $u(k)$.

1.4.2 Algoritmo *Clamping*

La estrategia *integrator clamping*, también nombrada *conditional integration* (Visioli 2006, Cap. 3), está basada en parar la integración (acumulación) del error cuando la acción de control está saturada:

$$u_i(t) = K_i \int_0^t e_i(t)dt; \qquad e_i(t) = \begin{cases} e(t), & \text{si } u(t) = v(t) \\ 0, & \text{si } u(t) \neq v(t) \end{cases}$$

Es decir, cuando la acción de control se satura, el integrador deja de acumular el error y la parte integral de la acción de control $u_i(t)$ mantiene su valor constante. Pero cuando la acción de control deja de estar saturada, el integrador retoma su trabajo de acumular el error para eliminar la desviación en régimen permanente.

Una versión mejorada de este algoritmo añade la condición $e(t) \cdot u(t) > 0$ a la condición de saturación de la acción de control:

$$u_i(t) = K_i \int_0^t e_i(t)dt; \qquad e_i(t) = \begin{cases} 0 & \text{si } u(t) \neq v(t) \text{ y } e(t) \cdot u(t) > 0 \\ e(t) & \text{en cualquier otro caso} \end{cases}$$

En esta versión del algoritmo, el integrador dejará de acumular el error si la acción de control $u(t)$ se satura y esta tiene el mismo signo que el error $e(t)$ es decir $e(t) \cdot u(t) > 0$. Esta nueva condición hace que el integrador ayude a sacar la acción de control de la saturación. Por ejemplo, si la acción de control está saturada en su máximo $u > 0$, el integrador dejará de acumular el error siempre que la salida controlada sea menor que la referencia $y < r \Rightarrow e > 0$; pero en el momento en que la salida controlada sobrepase la referencia $y > r \Rightarrow e < 0$, lo que interesa es que el integrador ayude a frenar el sistema sacando la acción de control de la saturación, que es lo que ocurre: $e(t) \cdot u(t) < 0$ y, por tanto, la acción integral acumulará errores negativos y ayudará a la acción de control a salir de la saturación.

1.5 Plan de trabajo

La resolución del problema planteado requiere seguir unos pasos que nos permitan llegar a una solución de forma rápida y clara. Para ello, el plan de trabajo siguiente enumera cada uno de los procedimientos que se han realizado en el Capítulo 3 para la obtención de la solución al problema:

1. Modelo matemático por identificación.
2. Representación en el espacio de estados.
3. Diseño del controlador.

   3.1. Observador.

   3.2. Observador con filtro de la medida.

   3.3. Filtro de Kalman.

4. Arduino Due.

5. Monitorización del Arduino.

   5.1. Controlador con observador.

   5.2. Controlador con observador más filtro de la medida.

   5.3. Controlador con filtro de Kalman.

Capítulo 2

Implementación digital del controlador

En los próximos apartados explicaremos teóricamente en qué consiste la realimentación de la salida y las tres formas diferentes de implementación que podremos encontrar en un sistema de control.

2.1 Realimentación de la salida

La realimentación de la salida es característica de los sistemas de control debido a que se realiza una operación inicial a la entrada del controlador donde la referencia $r(k)$ menos la salida $y(k)$ generan un error que el sistema corregirá con el paso del tiempo a partir de la acción de control integral $u_i(k)$ y el integrador discreto.

En muchos casos prácticos, sólo son medibles unas cuantas variables de estado de un sistema dado, mientras que las demás no lo son. Puede darse el caso que sólo las variables de salida son medibles. En caso de que esto ocurra, será necesario estimar las variables de estado que no se puedan medir directamente a partir de las variables de salida y las de control. Para ello será necesario el uso de un algoritmo que realice la estimación del estado basado en el conocimiento previo del modelo en tiempo discreto de la planta. En el diagrama de bloques de la figura 2.1 se muestra el esquema de la realimentación de la salida.

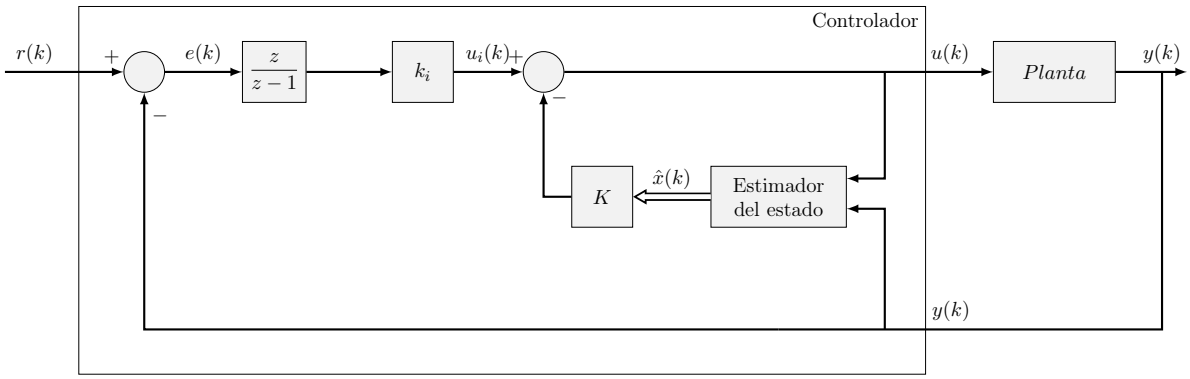


Figura 2.1: Diagrama de bloques del control digital por realimentación de la salida con acción integral

Se observa que el controlador engloba al estimador del estado, a la acción de control integral $u_i(k)$ y al vector K por la estimación de la variable de estado $\hat{x}(k)$.

En la sección 1.2 se ha comentado de que trata la controlabilidad y la observabilidad. No obstante, se ha hecho de forma muy general.

Como ya se ha dicho, el controlador está compuesto por el estimador del estado, por la acción de control integral $u_i(k)$ y por el vector K. Además, cabe destacar que el controlador presenta dos entradas, la referencia $r(k)$ y la salida del sistema $y(k)$, y una salida, la acción de control $u(k)$, ver figura 2.1.

Por lo tanto, la acción de control $u(k)$ por realimentación de la salida (control por realimentación del estado estimado) tiene la siguiente forma:

$$u(k) = -K\,\hat{x}(k) + u_i(k)$$

donde $u_i(k)$ es la acción de control integral, que es el producto de la constante k_i por la salida del integrador discreto que acumula el error $e(k) = r(k) - y(k)$. La ecuación en diferencias que corresponde con $u_i(k)$ se define como sigue:

$$U_i(z) = k_i \frac{z}{z-1} E(z) = \frac{k_i}{1 - z^{-1}} E(z)$$

$$\left[1 - z^{-1}\right] U_i(z) = k_i\, E(z)$$

$$u_i(k) = u_i(k-1) + k_i\, e(k)$$

Se puede apreciar claramente que cuando el valor del error es cero $e(k) = 0$, la acción de control integral se mantiene constante $u_i(k) = u_i(k-1)$. En cambio si hay error, por pequeño que sea, la acción de control integral irá variando su valor.

En la acción de control $u(k)$ la estimación del vector de estado $\hat{x}(k)$ se multiplica por el vector fila K, que corresponde con los parámetros de diseño del controlador.

A continuación se analizará qué ocurre en la realimentación de la salida con observador. Además, también se analizará la realimentación de la salida con observador y filtro de la medida, y la realimentación de la salida por filtro de Kalman.

2.2 Realimentación de la salida con observador

En el diagrama de bloques de la figura 2.1 se tiene un estimador del estado, por lo que éste, a continuación, se convertirá en un observador del estado, ya que es lo que se implementará en nuestro sistema.

El observador del estado utiliza la información del modelo del sistema expresado en el espacio de estado:

$$x(k+1) = A\,x(k) + B\,u(k) \tag{2.1}$$
$$y(k) = C\,x(k) \tag{2.2}$$

El algoritmo de estimación del vector de estado $\hat{x}(k)$ (basado en el observador de Luenberger) es:

$$\hat{x}(k+1) = A\,\hat{x}(k) + B\,u(k) + L\,[y(k) - C\,\hat{x}(k)]$$
$$\hat{y}(k) = C\,\hat{x}(k)$$

Debe considerarse que aunque el estado $x(k)$ no sea medible, la salida $y(k)$ si lo es, por lo que el desempeño del modelo dinámico puede mejorar si se utiliza la diferencia entre la salida medida $y(k)$ y la salida estimada $\hat{y}(k) = C\,\hat{x}(k)$ para monitorizar el estado $\hat{x}(k)$.

En la figura 2.2 se representa gráficamente el algoritmo de estimación del estado, donde se observan las dos entradas: la acción de control $u(k)$ y la salida realimentada $y(k)$, y la salida: el vector de estado estimado $\hat{x}(k)$. También aparece en el diagrama de bloques la salida estimada $\hat{y}(k)$, se puede apreciar claramente que la ganancia del observador (L) estará multiplicada por la diferencia de $y(k) - \hat{y}(k)$.

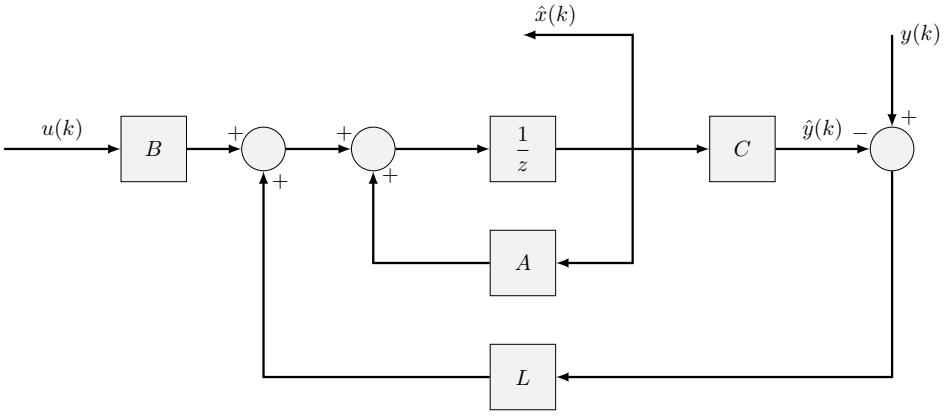


Figura 2.2: Diagrama de bloques del observador

Si el sistema es observable, la matriz L, matriz equivalente a la matriz de realimentación del observador K_e, puede ser escogida de modo que $[A - LC]$ sea asintóticamente estable y $\hat{x}(k)$ asintóticamente se aproxime a $x(k)$. El término $[y(k) - C\hat{x}(k)]$ proporciona un factor de corrección proporcional que asegura la estabilidad del observador incluso cuando el sistema es inestable y ayuda a reducir las diferencias entre el modelo dinámico y el modelo real.

Este diseño lo componen las matrices de la representación en el espacio de estados en tiempo discreto. Se puede ver que la matriz B es la matriz de entrada, la matriz C es la matriz de salida, la matriz A es la matriz de estado, y por último, como ya se ha dicho anteriormente, la matriz L es la matriz de realimentación del observador o ganancia del observador, y donde se encuentran los valores propios deseados.

2.3 Realimentación de la salida con observador y filtro de la medida

La presencia de perturbaciones es una de las razones principales para el uso de sistemas de control. Sin perturbaciones no habría necesidad de utilizar sistemas con realimentación. La naturaleza de las perturbaciones determina la calidad de regulación en un proceso de control, y además, transmiten información importante sobre las propiedades del sistema.

Hasta el momento se ha hablado sólo de la realimentación de la salida con observador y se ha considerado un sistema ideal en el que no hay ruido ni perturbaciones. Pero esto no es así. Todos los sistemas que encontramos se ven afectados tanto por ruido como por perturbaciones que impiden un correcto funcionamiento de nuestro sistema. Para corregir estos problemas una solución es filtrar la medida de la salida controlada.

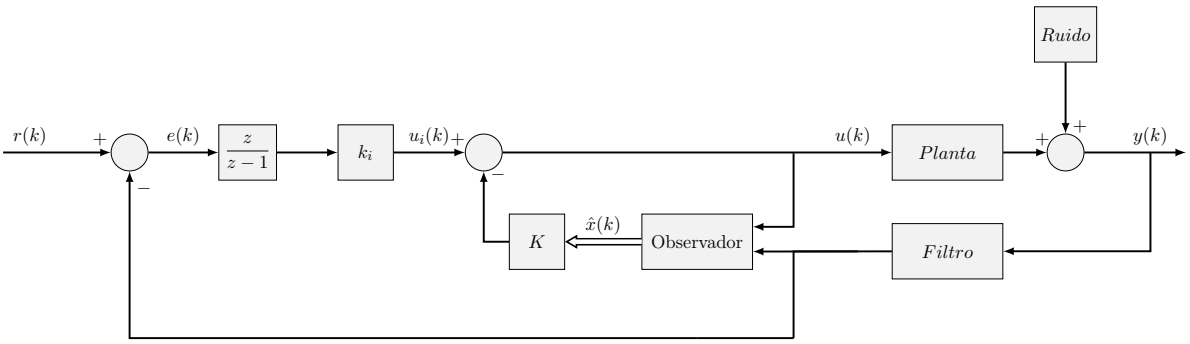


Figura 2.3: Diagrama de bloques real, con filtro y ruido

Como podemos ver en la figura 2.3 se incorpora en la realimentación un filtro de la medida que tomará el valor de la salida más el ruido generado. Este ruido es generado por el propio motor.

La forma de evitar que este ruido sea muy perjudicial para nuestro sistema es incorporar un filtro, de modo que reduzca las vibraciones que se produzcan en éste mediante un proceso matemático sobre la señal de entrada.

Los filtros son redes que procesan las señales de una manera dependiente de la frecuencia. El concepto básico de un filtro se puede explicar examinando la naturaleza de la frecuencia de la impedancia de los condensadores e inductores. Tienen muchas aplicaciones prácticas. Un simple filtro de paso bajo (integrador) a menudo se utiliza para estabilizar amplificadores por la atenuación de la ganancia a altas frecuencias donde el desplazamiento de la fase puede causar oscilaciones. Por otra parte, un simple filtro de paso alto puede ser usado para bloquear el offset de cc en los amplificadores de alta ganancia o circuitos de alimentación individual. Los filtros se pueden utilizar para separar señales, pasando las de interés, y atenuando las no deseadas.

Hay diferentes tipos de filtros de la medida. Se pueden destacar varios, como el filtro de Chebyshev, el filtro de Bessel y el filtro de Butterworth. Según Zverev y Blinchikoff (1976) cada uno de ellos presenta un mejor funcionamiento para determinados diseños. Para conocerlos un poco mejor, a continuación se puede ver una pequeña explicación de cada uno de los filtros que hemos nombrado.

El filtro de Bessel está optimizado para obtener una respuesta transitoria debido a una fase lineal (es decir, retardo constante) en la banda de paso. Esto significa que no habrá respuesta de frecuencia relativamente más pobre (menos discriminación de amplitud). Por lo tanto, se trata de un filtro con buena respuesta en el dominio del tiempo. Los polos del filtro de Bessel se pueden determinar localizando todos los polos en un círculo y separando la parte imaginaria.

Por lo que respecta al filtro de Chebyshev da una mejor discriminación de amplitud respecto a los otros dos, seguido del filtro de Butterworth y del de Bessel. Existen dos tipos diferentes de filtros de Chebyshev: tipo 1 (la onda sólo está permitida en la banda de paso) y tipo 2 (presentan la onda sólo en la banda de detención). Los que más se usan son los de tipo 1. A

medida que aumenta la ondulación, la atenuación se agudiza. La respuesta de Chebyshev es una óptima compensación entre esos dos parámetros.

Por último, el filtro de Butterworth es el que presenta mejor relación entre atenuación y respuesta de fase. No tiene onda en la banda de paso ni en la banda de detención, y debido a esto a veces se llama filtro de aplanamiento máximo. Logra su planeidad a pesar de una amplia región de transición de la banda de paso a la banda de detención, con características medias transitorias.

Al ser este último filtro mejor que los otros dos, ya que podríamos decir que se encuentra en medio, se ha escogido para nuestro sistema de modo que podamos atenuar el ruido del motor.

El filtro de Butterworth es un filtro digital de orden n, que puede presentar una frecuencia normalizada de corte ω_n que variará a nuestra elección, no obstante estas variaciones provocarán que la salida presente más o menos sobrepasamiento y un tiempo de establecimiento menor o mayor. Hay que tener en cuenta que la frecuencia normalizada de corte es la razón entre la frecuencia de corte ω_c en radianes por segundo y la frecuencia en H_z de la muestra, por tanto:

$$\omega_n = \frac{\omega_c}{f_s} = \frac{2\pi\, f_c}{\frac{1}{T_s}} = 2\pi\, T_s f_c$$

Introducir un filtro de la medida en nuestro sistema permite que la acción de control se suavice, y por lo tanto, que no aparezca ruido en ella. Sin embargo, dependiendo del orden del filtro, estaremos introduciendo más polos y ceros al sistema, con lo que las especificaciones de funcionamiento (tiempo de establecimiento y pico de sobrepasamiento) pueden variar.

2.4 Realimentación de la salida con filtro de Kalman

La ganancia de un observador se selecciona simplemente para mantener la estabilidad y para proporcionar una respuesta dinámica razonable. Este método de selección es a menudo adecuado. Hay dos casos, sin embargo, en los cuales es necesario el uso de un método más sistemático. El primer caso es que no haya una forma obvia de selección de la ganancia del observador. Esta situación generalmente surge cuando hay más términos en la matriz de la ganancia del observador de los que se necesitan para establecer los polos del observador. El segundo caso aparece cuando es importante que el estado estimado producido por el observador sea lo más preciso posible, y donde las propiedades del ruido en la medida se pueden determinar. Para estos casos es apropiado el uso de el filtro de Kalman, (Friedland 1995, p. 189).

En Åström y Wittenmark (2013), el filtro de Kalman se describe para sistemas discretos de la forma que sigue:

$$\begin{aligned} x(k+1) &= A\,x(k) + B\,u(k) + \omega(k) \\ y(k) &= C\,x(k) + \nu(k) \end{aligned}$$

donde ω y ν son el ruido en tiempo discreto del proceso y de la medida respectivamente. La función de covarianza de un proceso se define como:

$$R_1 = E\left[\omega(k)\,\omega(k)^T\right]; \quad R_2 = E\left[\nu(k)\,\nu(k)^T\right]; \quad R_{12} = E\left[\omega(k)\,\nu(k)^T\right];$$

R_1 y R_2 son variables independientes, es decir uno puede aumentar y no influiría en el otro, éste puede disminuir o aumentar arbitrariamente. Además, para simplificar, consideramos que $R_{12} = 0$. La estimación del próximo estado tiene la siguiente forma:

$$\hat{x}(k+1|k) = A\,\hat{x}(k|k-1) + B\,u(k) + L(k)\,[y(k) - C\,\hat{x}(k|k-1)] \tag{2.3}$$

por lo que la dinámica del error de estimación $e = x - \hat{x}$ pasa a ser:

$$e(k+1) = [A - L(k)\,C]\,e(k) + \omega(k) - L(k)\,\nu(k) \tag{2.4}$$

Como se ha comentado en la sección 2.2, la matriz L se utiliza para obtener los valores propios deseados. Pero el problema que encontramos aquí es diferente: las propiedades del ruido se tienen en cuenta y el criterio es minimizar la varianza del error de estimación, que se define por $P(k)$:

$$P(k) = E\left[(e(k) - E\,[e(k)])\,(e(k) - E\,[e(k)])^T\right] \tag{2.5}$$

El valor medio del error de estimación se obtiene desde la ecuación (2.4):

$$E\,[e(k+1)] = [A - L(k)\,C]\,E\,[e(k)]$$

Se puede observar que si $E\,[\hat{x}(0)] = E\,[x(0)]$, entonces el valor medio del error de estimación es cero para todos los tiempos independientes de $L(k)$. Teniendo en cuenta esto, la ecuación (2.5) que expresa la varianza del error de estimación puede escribirse como sigue:

$$P(k) = E\left[e(k)\,e(k)^T\right] \tag{2.6}$$

Siempre que $e(k)$, $\omega(k)$ y $\nu(k)$ sean independientes, la dinámica del error de estimación (ecuación 2.4) se puede utilizar en la expresión (2.6) para derivar en:

$$\begin{aligned} P(k+1) &= E\left[e(k+1)\,e(k+1)^T\right] = \\ &= [A - L(k)\,C]\,P(k)\,[A - L(k)\,C]^T + R_1 + P(k)\,R_2\,P(k)^T \end{aligned}$$

El algoritmo de minimización (Åström y Wittenmark 2013) da lugar al siguiente algoritmo para el cálculo de $L(k)$:

$$L(k) = A\,P(k)\,C^T\left[R_2 + C\,P(k)\,C^T\right]^{-1} \tag{2.7}$$

$$P(k+1) = A\,P(k)\,A^T + R_1 - L(k)\,C\,P(k)\,A^T \tag{2.8}$$

La reconstrucción definida por las ecuaciones (2.3), (2.7) y (2.8) se denomina *Filtro de Kalman*:

$$\begin{aligned} \hat{x}(k+1) &= A\,\hat{x}(k) + B\,u(k) + L(k)\,[y(k) - C\,\hat{x}(k)] \\ L(k) &= A\,P(k)\,C^T\left[R_2 + C\,P(k)\,C^T\right]^{-1} \\ P(k+1) &= A\,P(k)\,A^T + R_1 - L(k)\,C\,P(k)\,A^T \end{aligned}$$

Todo esto se resume en el siguiente teorema:

Teorema 3.4.1. *(Filtro de Kalman) Considerando el proceso aportado por esta sección (2.4). La reconstrucción de los estados usando el modelo de la expresión (2.3) es óptima en el sentido que la varianza del error de reconstrucción está minimizada si la matriz* $R_2 + C\,P(k)\,C^T$ *se define positiva y si la ganancia de la matriz se elige acorde con (2.7) y (2.8). Esta última ecuación nos proporciona la varianza del error de reconstrucción.*

En la figura 2.4 se puede ver un diagrama de bloques correspondiente a nuestro sistema con un filtro de Kalman.

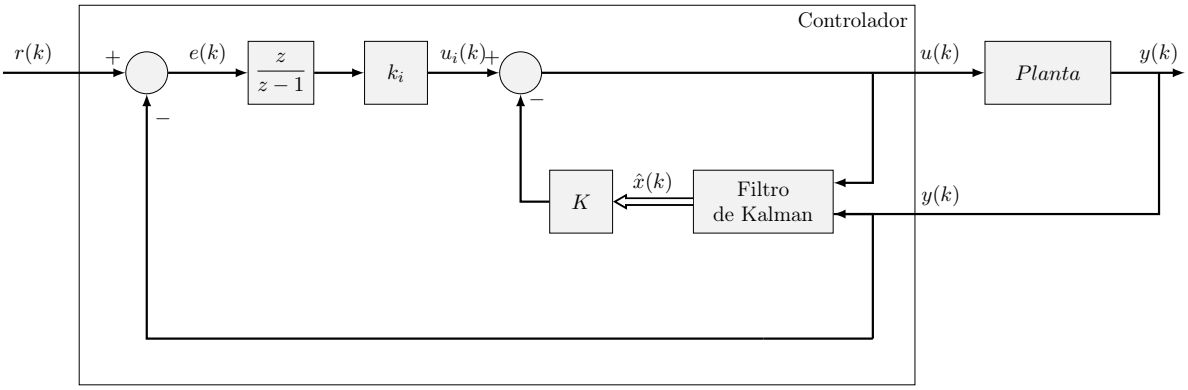


Figura 2.4: Diagrama de bloques con filtro de Kalman

2.5 Implementación en un computador

En esta sección se exponen los algoritmos de las siguientes estrategias de control digital por realimentación de la salida:

- Basado en observador del estado.
- Con observador del estado y filtro de la medida.
- Basado en la estimación del estado con un filtro de Kalman.

2.5.1 Realimentación de la salida con observador

La secuenciación de las operaciones a realizar por el computador para implementar un control por realimentación de la salida (basado en observador) viene definida por el siguiente algoritmo:

Algoritmo del Controlador

Datos: $K,\ K_e,\ k_i,\ r(k),\ y(k),\ u_i(k-1),\ \hat{x}(k)$;

Resultados: $e(k),\ u_i(k),\ u(k),\ \hat{x}(k+1)$;

Método:

1: $e(k) = r(k) - y(k)$;

2: $u_i(k) = u_i(k-1) + k_i\,e(k)$;

3: $u(k) = -K\,\hat{x}(k) + u_i(k);$

4: $\hat{x}(k+1) = A\,\hat{x}(k) + B\,u(k) + K_e\ (y(k) - C\,\hat{x}(k))\,;$

fControlador

Los parámetros que el algoritmo debe conocer son la constante k_i y los vectores K y K_e. También se suponen conocidos los valores de la señal de referencia $r(k)$ y la salida controlada $y(k)$ procedente del muestreo.

En la primera línea de programa se observa que el error $e(k)$ viene definido por la referencia $r(k)$ menos la realimentación de la salida $y(k)$. Por lo que el valor de la referencia en el instante actual de muestreo debe coincidir en escala y unidad con el valor de la salida.

En la segunda línea del algoritmo del controlador se calcula la acción de control integral $u_i(k)$ que depende de la acción de control integral en el anterior muestreo y del error $e(k)$ por la constante integral k_i. En caso de querer aplicar herramientas $anti-windup$ sería en esta línea de programa donde se tendría que aplicar. Si se aplicará el método *Back calculation* la línea de programa 2 quedaría como sigue:

$$u_i(k)\ =\ u_i(k-1)\ +\ k_i\,e(k)\ -\ k_b\,[v(k-1)\ -\ u(k-1)];$$

donde k_b es una variable que utiliza el $anti-windup$ *mode Back calculation*, $v(k-1)$ es la acción de control no saturada en el instante anterior, y $u(k-1)$ es la acción de control saturada también en el instante anterior. En caso de aplicar el método *Clamping* la línea de programa sería:

$$u_i(k)\ =\ u_i(k-1)\ +\ k_i\,e(k)\,[v(k-1) == u(k-1)];$$

La diferencia de utilizar el método *Clamping* es que en este caso se compara la acción de control no saturada $v(k-1)$ con la acción de control saturada $u(k-1)$.

El objetivo del algoritmo es calcular y ofrecer como resultado el valor de la acción de control $u(k)$ –línea de programa 3–. Pero como las variables se van actualizando para la siguiente ejecución, también es solución del algoritmo $\hat{x}(k+1)$, que se trata de la estimación del vector de estado en la próxima ejecución.

Cabe destacar que en la línea 4 de programa se realiza una diferencia interna en el propio observador entre la salida $y(k)$ y la estimación de la salida $\hat{y}(k)\ =\ C\,\hat{x}(k)$, que se multiplica por la constante K_e, que corresponde con la ganancia del observador de estado.

2.5.2 Realimentación de la salida con observador y filtro de la medida

La secuenciación correspondiente a la realimentación de la salida con observador y filtro de la medida es la que sigue:

Algoritmo del controlador con filtro de la medida

Datos: $K,\ K_e,\ k_i,\ r(k),\ y(k),\ y_f(k-1),\ y_f(k-2),\ u_i(k-1),\ \hat{x}(k)$;

Resultados: $y_f(k),\ e(k),\ u_i(k),\ u(k),\ \hat{x}(k+1)$;

Método:

1: $y_f(k) = -\frac{a_1}{a_0}\, y_f(k-1) - \frac{a_2}{a_0}\, y_f(k-2) + \frac{1}{a_0}\, y(k);$

2: $e(k) = r(k) - y_f(k);$

3: $u_i(k) = u_i(k-1) + k_i\, e(k);$

4: $u(k) = -K\, \hat{x}(k) + u_i(k);$

5: $\hat{x}(k+1) = A\, \hat{x}(k) + B\, u(k) + K_e\, [y_f(k) - C\, \hat{x}(k)]\,;$

fControlador con observador más filtro

Como se puede observar, este algoritmo es muy similar al algoritmo del controlador con observador. No obstante, la diferencia respecto al anterior es que se ha de tener en cuenta una nueva ecuación en diferencias correspondiente al filtro de la medida que se ha introducido en la realimentación para reducir y eliminar el ruido.

Por lo tanto el algoritmo del controlador con observador más filtro de la medida comienza con la línea de programa correspondiente al cálculo de $y_f(k)$, que es la salida filtrada que llegará al observador.

Se observa que para el cálculo de la salida filtrada $y_f(k)$ son necesarios los valores de la salida filtrada en las dos anteriores ejecuciones, $y_f(k-1)$ y $y_f(k-2)$, además, de la salida del sistema en el instante actual $y(k)$.

Hay que tener en cuenta los parámetros a_0, a_1 y a_2, que dependen de la frecuencia normalizada de corte del filtro $wnorm$ y del orden que tendrá el filtro de la medida nf, por lo tanto serán valores conocidos. Escoger una configuración distinta para el filtro de la medida, provocará un cambio en $y_f(k)$.

Por lo que respecta al resto de líneas de programa tendrán la misma función que se ha explicado en la subsección 2.5.1, no obstante, habrá que tener en cuenta que $y(k)$ en este algoritmo pasa a ser $y_f(k)$.

2.5.3 Realimentación de la salida con el filtro de Kalman

El algoritmo anterior nos permite obtener las operaciones que nuestro computador debe realizar para estimar las variables que utilizaremos en la siguiente secuenciación. No obstante, este algoritmo se centra en el observador y necesitamos otro algoritmo para el filtro de Kalman que se ha utilizado. Por lo tanto, el algoritmo que se ha de desarrollar corresponde al explicado en la sección 2.4 el cual se basa en Åström y Wittenmark (2013) "*Computer-controlled systems: theory and design*".

Algoritmo del filtro de Kalman

Datos: $r(k),\ y(k),\ K,\ k_i,\ R_1,\ R_2;$

Resultados: $e(k),\ u_i(k),\ u(k),\ L(k),\ P(k+1),\ \hat{x}(k+1);$

Método:

1: $e(k) = r(k) - y(k);$

2: $u_i(k) = u_i(k) + k_i\, e(k);$

3: $u(k) = -K\, \hat{x}(k) + u_i(k);$

4: $L(k) = A\,P(k)\,C^T\left[R_2 + C\,P(k)\,C^T\right]^{-1}$;
5: $P(k+1) = A\,P(k)\,A^T + R_1 - L(k)\,C\,P(k)\,A^T$;
6: $\hat{x}(k+1) = A\,\hat{x}(k) + B\,u(k) + L(k)\,[y(k) - C\,\hat{x}(k)]$;

fFiltro Kalman

Las tres primeras líneas del programa que se presentan son iguales que en el algoritmo del controlador con observador, ya que, al igual que cuando se utiliza observador, se ha de tener un error $e(k)$ (que es el resultado de la diferencia entre la referencia $r(k)$ y la salida del sistema $y(k)$) a partir del cual se obtiene la acción de control integral.

En la línea de programa 2 se observa que para el cálculo de la acción de control integral $u_i(k)$ se necesita esa misma variable. No obstante no corresponde a la misma $u_i(k)$, sino que se trata de la acción de control integral en el anterior instante de muestreo. Se nombran de la misma forma para no tener que realizar una actualización de variables posteriormente. Igual que se ha dicho en la subsección 2.5.1, si el $e(k) = 0$ la acción de control integral en este instante será igual al estado anterior.

La siguiente línea de programa (3) es la acción de control de nuestro sistema, es decir, la salida del controlador, y tiene en cuenta la acción de control integral en este instante de tiempo, además de la multiplicación del vector K (parámetros de diseño del controlador) y el vector del estado estimado en este instante también $\hat{x}(k)$. Al presentar el controlador una realimentación, estos dos últimos términos se restan a la acción de control integral.

Las líneas de programa 4, 5 son diferentes al controlador con observador, ya que aparecen conceptos que se necesitan al implementar un filtro de Kalman y que no son necesarios con un observador. Estos conceptos son: $L(k)$ y $P(k+1)$. $L(k)$ es una matriz de ponderación que irá cambiando de valor según pase el tiempo. Por lo que respecta a $P(k+1)$ se trata de la varianza del error de estimación en el siguiente instante de muestreo. También aparecen dos valores (R_1 y R_2) que corresponden a dos escalares independientes.

Cabe considerar que ambas líneas utilizan la matriz $P(k)$, varianza del error de estimación calculada en el anterior muestreo, y que en la línea 5 de programa, $P(k+1)$ utiliza la actual matriz de ponderación $L(k)$.

Por último, en la línea de programa 6 se define la ecuación del próximo estado estimado $\hat{x}(k+1)$. Para ello es necesario calcular en las líneas anteriores la acción de control $u(k)$ y la matriz $L(k)$. Se puede ver que se utiliza el actual estado estimado $\hat{x}(k)$ calculado en la anterior ejecución del programa, y la salida estimada $\hat{y}(k) = C\,\hat{x}(k)$.

2.6 Arduino due

Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo para facilitar el uso de la electrónica en proyectos multidisciplinares.

Arduino es una placa programable con entradas y salidas digitales y analógicas, el bajo coste la hacen ideal para iniciarse en automatización o realizar pequeños proyectos domésticos de electrónica y robótica. Esto quiere decir que disponemos de un pequeño "autómata", capaz de recibir información del entorno (sensores) y realizar acciones (actuadores, motores, ...), según un programa que introducimos con un ordenador, y que puede ejecutar de forma autónoma.

El modelo Arduino Due que utilizaremos para nuestro proyecto, es una placa electrónica basada en el *Atmel SAM3X8E ARM Cortex-M3 CPU*. Es la primera placa Arduino basada en un microcontrolador con núcleo ARM de 32 bits.

Algunas de las características que debemos tener en cuenta del Arduino Due son:

- 54 pines de entrada/salida donde 12 se pueden utilizar para salidas PWM.
- 12 entradas analógicas.
- 2 DAC (de digital a analógico).
- Botón de reinicio.
- Botón de borrado.

Advertencia: A diferencia de otras placas Arduino, esta placa funciona a 3,3 V. Por tanto, el voltaje máximo que los pines de E/S pueden tolerar es de 3,3 V. Proporcionar voltajes más altos, como 5 V a un pin de I/O podría dañar la placa.

La placa contiene todo lo necesario para dar soporte al microcontrolador. El Arduino Due es compatible con todos los escudos de Arduino que trabajen a 3,3 V y cumplan con 1, 0 Arduino *pinout*.

Para más información de las características de éste se puede ver en la siguiente página web: http://arduino.cc/en/Main/arduinoBoardDue

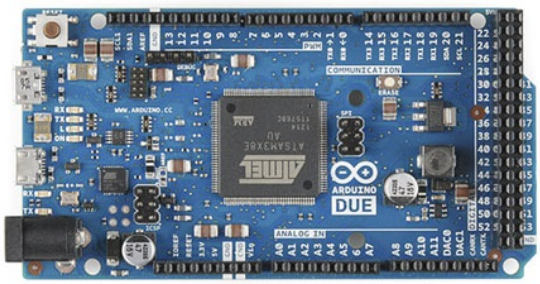


Figura 2.5: Arduino due

Al utilizar el Arduino como controlador de nuestro proceso de control nos encontramos delante de conceptos que debemos conocer para un buen uso del mismo.

⇒ Programa con Arduino:

Todos los programas con Arduino tienen que contener dos funciones de forma necesaria: `setup()` y `loop()`.

- `Setup()`: esta función tan solo se ejecuta una vez, después de cada encendido o con el reinicio de la placa de Arduino. En esta sección se suele encontrar la inicialización de variables, configuración de pines, inicialización de librerías, etc.

- `Loop()`: esta función se ejecutará continuamente en forma de bucle, contendrá el código que controlará nuestra placa de Arduino.

  Por tanto el código principal se quedará de la siguiente forma:

```
void setup(){
// Poner el código de configuración aquí, para ejecutar una vez:
}
void loop(){
// Poner el código principal aquí, para ejecutar varias veces:
}
```

⇒ Importación de la librería `DueTimer`:

Programaremos los temporizadores (timers) utilizando la librería `DueTimer`, para poder utilizarña debemos instalarla siguiendo los siguientes pasos:

1. Descargar la librería de: https://github.com/ivanseidel/DueTimer.
2. Descomprimir y modificar el nombre de la carpeta a "`DueTimer`".
3. Copiar la carpeta modificada en la carpeta de librerías de nuestro Arduino.
4. Importar el *.zip* desde el software de Arduino: Programa→Incluir Librería→Añadir librería .ZIP...

En el código a programar incluiremos la librería con la siguiente sentencia:

```
#include <DueTimer.h>
```

Esta librería contiene 9 *timers* para su utilización y programación, las funciones más comunes son:

- `getAvailable()`: para obtener el temporizador disponible.
- `attachInterrupt(void(*ISR)())`: para adjuntar una interrupción (función de devolución de llamada) al temporizador del objeto.
- `detachInterrupp()`: para separar la devolución de llamada actual del temporizador.
- `start(microsegundos=-1)`: para iniciar el temporizador con un parámetro, periodo opcional.
- `stop()`: para detener el temporizador.
- `long getFrecuency()`: para obtener la frecuencia del temporizador.
- `setPeriod(microsegundos long)`: para ajustar el periodo del temporizador (en microsegundos).
- `long getPeriod()`: para obtener el periodo del temporizador (en microsegundos).

Capítulo 3

Resultados obtenidos

En apartados anteriores se han explicado conceptos teóricos correspondientes al espacio de estados, a la realimentación de la salida y a aspectos básicos del tiempo discreto. Llegados a este punto, se analizarán qué cálculos se han realizado para el control del motor.

3.1 Modelo matemático por identificación

Las características de funcionamiento del motor son desconocidas, no tenemos datos sobre las componentes del rotor ni del estátor, por tanto es preciso la obtención de una función de transferencia que se aproxime al modelo de motor que tenemos. Para ello se ha generado una señal de entrada, la cual introduce escalones al motor de 1 V en un rango de 2-5 V como se puede ver en la figura 3.1. El criterio de elección de los escalones de referencia es aleatorio ya que el único impedimento que tenemos es la tensión aceptada por la placa de Arduino.

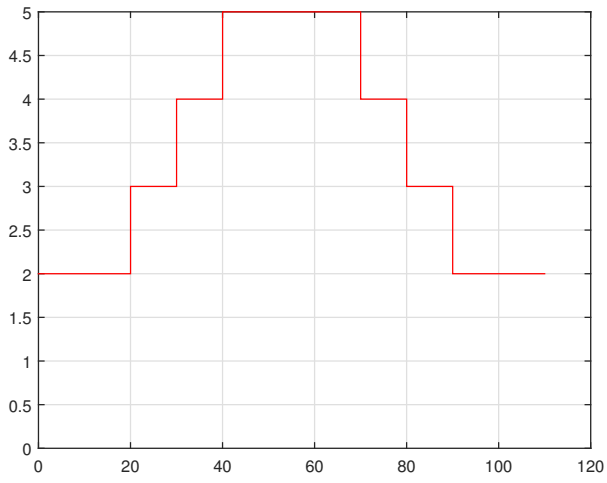


Figura 3.1: Escalones en la referencia

Al igual que la referencia es aleatoria, el tiempo de cada escalón también lo es. Los tiempos escogidos se deben a que se ha intentado escoger unos tiempos ante los que el sistema sea capaz de responder con rapidez (un tiempo de establecimiento pequeño) y sin picos de sobrepasamiento excesivos.

A continuación, se ha realizado un ensayo con *LabVIEW* donde se observa el comportamiento del motor delante de estos escalones. Se ha programado de tal modo que los datos obtenidos se guardarán en una carpeta (*dades*) y a través de *MATLAB* se realizará la simulación y obtendremos las gráficas correspondientes al comportamiento del motor delante de la referencia. Esto se puede comprobar en la figura 3.2.

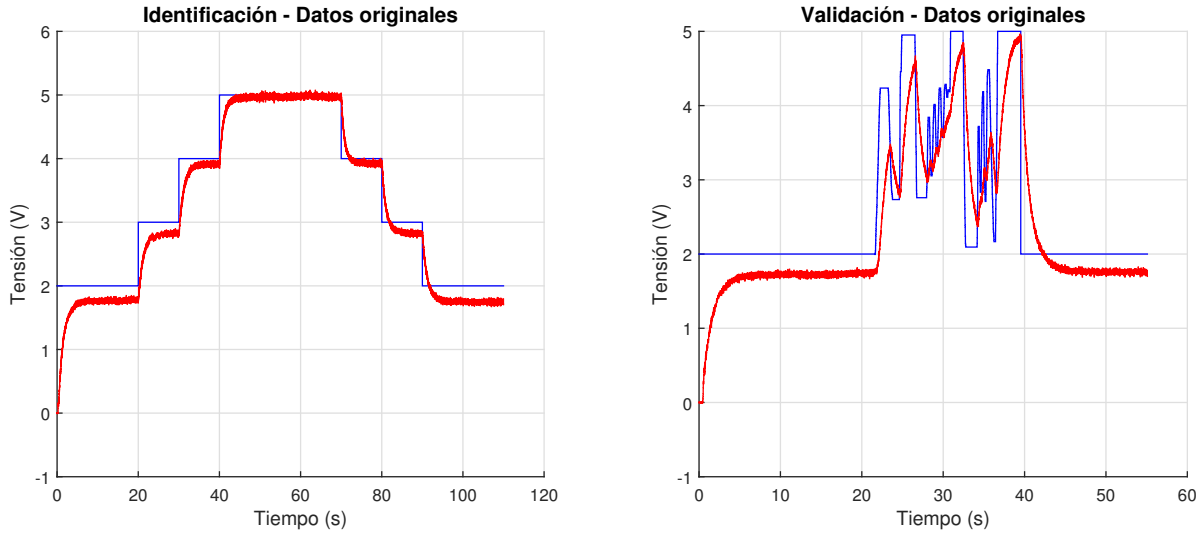


Figura 3.2: Resultado de la simulación en *LabVIEW*

Como se puede ver, la gráfica de la izquierda corresponde con la identificación de la respuesta del motor (línea roja). Esto quiere decir que el motor actúa de esa forma ante los escalones presentados en la figura 3.1 (línea azul).

Por lo que respecta a la gráfica de la derecha, se trata de comprobar que la identificación es buena, para ello se realizan cambios manuales aleatorios en la referencia sin pasarse de los valores mínimo (2 V) y máximo (5 V). La línea azul representaría los cambios que se han realizado manualmente y la línea roja la respuesta del motor ante estos cambios.

También se observa en la figura 3.2 que existe un offset, además de que el motor no empieza a trabajar directamente con los 2 V que nosotros le ponemos como valor mínimo. Por tanto, debemos recortar y eliminar el offset, es decir, cambiar la desviación de la respuesta y ajustarla a los escalones que se le han introducido. Para ello se ha realizado un script en *MATLAB* donde se ha resuelto este problema, ver figura 3.3.

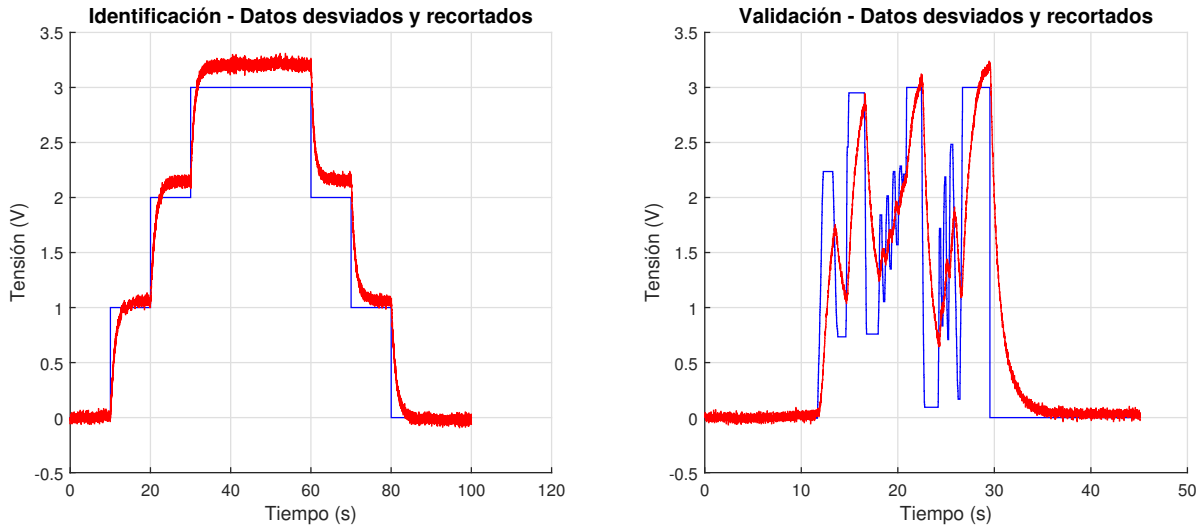


Figura 3.3: Datos desviados y recortados

En la figura 3.3 se puede destacar que el motor (línea roja) supera el valor máximo del escalón, pero esto no es un problema ya que la placa de Arduino Due que utilizamos para los ensayos tiene un rango de tensión de 0 a 3,3 V.

Siguiendo con la identificación del modelo del motor obtenemos la función de transferencia aproximada al motor DC a partir de los datos generados con *LabVIEW*. También se ha añadido un valor llamado T que corresponde al tiempo de muestreo, es decir, cada cuanto tiempo se tomará un valor.

Listado 1: Proceso de identificación de la F.d.T. del motor

```
%% Identificacio

dades_id = iddata(eixida,entrada,T); % DATOS PARA IDENTIFICAR

G = procest(dades_id,tipus_model,'Td',{'max',1}) % PROCESS ESTIMATOR

stop_time = T*(numel(eixida)-1);
t = 0 : T : stop_time;

[Ymodel,tmod] = lsim(G,entrada,t);
```

El listado 1 muestra el código de *MATLAB* que se ha utilizado para la obtención de la función de transferencia correspondiente al motor que se está utilizando de forma aproximada. En la expresión (3.1) se puede ver el resultado de la misma:

$$G(s) = \frac{49{,}159}{(s + 0{,}9404)(s + 48{,}97)} \tag{3.1}$$

En la figura 3.4 podemos observar la entrada medida, la salida medida y la salida del modelo que deseamos. También se ha añadido a la figura el ajuste del controlador, donde se observa que en la identificación tiene un valor del 97,5 %, ya que al ser una referencia establecida el controlador reacciona mucho mejor que con una referencia aleatoria. Por lo tanto, el ajuste de la validación será peor, y como se puede ver tiene un valor del 92,5 %.

El calculo del ajuste se realiza con la función `compare` como se puede ver en el siguiente código de *MATLAB* (se ha puesto el caso de la identificación, si se quiere calcular el de la validación se cambiaría "`dades_id`" por "`dades_valida`"):

```
[y,fit] = compare(dades_id,G);
```

Este ajuste se calcula en porcentaje, utilizando la desviación del error cuadrático medio normalizado ($NRMSE$), por tanto si el error cuadrático medio ($RMSE$) es:

$$RMSE = \sqrt{\sum (y - \hat{y})^2}$$

$$NRMSE = \frac{RMSE}{\mathrm{var}(y)} = \frac{\sum (y - \hat{y})^2}{\sum (y - \bar{y})^2}$$

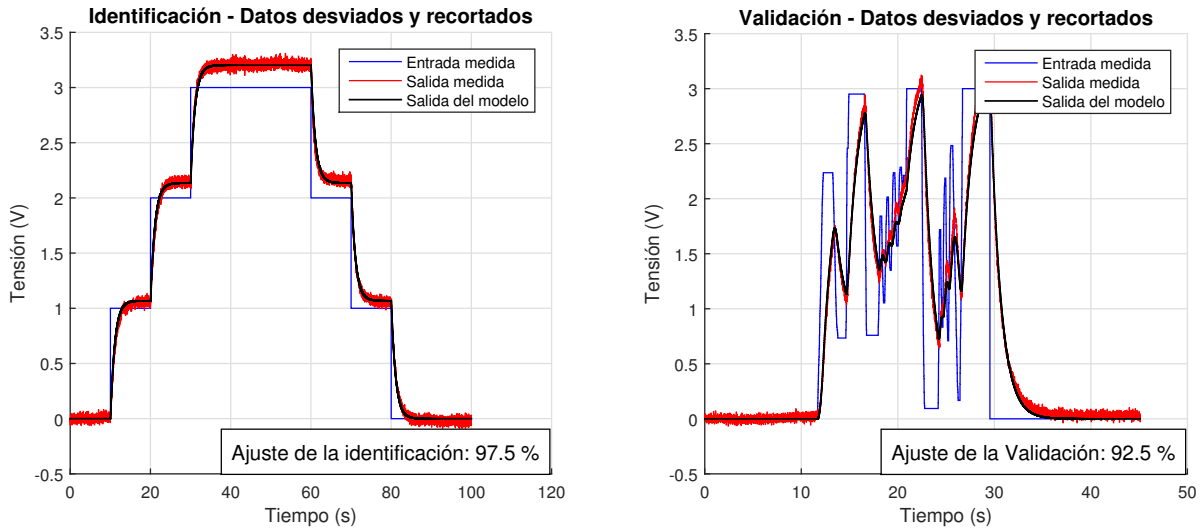


Figura 3.4: Comparación entre F.d.T. y respuesta real del motor

Por lo que el valor de "fit" será:

$$fit = (1 - NRMSE) \times 100 \quad (\%)$$

Como podemos observar, cuanto más grande sea el valor de "fit" mejor será el ajuste.

3.2 Representación en el espacio de estados

Una vez realizada la identificación del modelo matemático de nuestra planta debemos calcular las matrices correspondientes al espacio de estados, es decir, a partir de la función de transferencia $G(s)$ obtenida en el apartado anterior (ecuación 3.1). Para ello *MATLAB* nos permite utilizar la función *ss* (función para el cálculo del espacio de estados) como se puede ver en el siguiente código:

Listado 2: Representación en el espacio de estados

```
%% Funcion de transferencia de lazo abierto

L = zpk(G); % L es la f.d.t. de lazo abierto

[num,den] = tfdata(L);
num = cell2mat(num);
den = cell2mat(den);

%% Obtencion matrices en el espacio de estados

[A,B,C,D] = tf2ss(num,den)

model_motor = ss(A,B,C,D)
```

Con esto obtenemos las matrices A, B, C y D que corresponden a la matriz de estado (donde se encuentran los valores propios de nuestro sistema), la matriz de entrada, la matriz de salida y la matriz de transferencia directa, respectivamente. Los cálculos realizados en el listado 2 dan como resultado las siguiente matrices:

`model_motor` =

$$A = \begin{pmatrix} -49{,}91 & -46{,}05 \\ 1 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad C = \begin{pmatrix} 0 & 49{,}16 \end{pmatrix}, \quad D = 0$$

Para comprobar si se trata de un cálculo correcto del espacio de estados se pueden comparar ambas respuestas ante un escalón, la de la función de transferencia y la del espacio de estados del motor. Esto se puede realizar definiendo una constante t como una variable de muestreo para comparar las respuestas. Al representar un `step` (escalón) de la función de transferencia en un tiempo t y del espacio de estados se obtienen dos respuestas prácticamente similares. La comprobación de la exactitud de la representación en el espacio de estados es calculando el error tal como sigue:

$$e = y - yss$$

siendo y la representación de la f.d.t. y yss la representación en el espacio de estados. Una vez obtenido el error, podemos ver gráficamente cual es la media cuadrática del error (RMSE) y la diferencia entre ambas respuestas (figura 3.5).

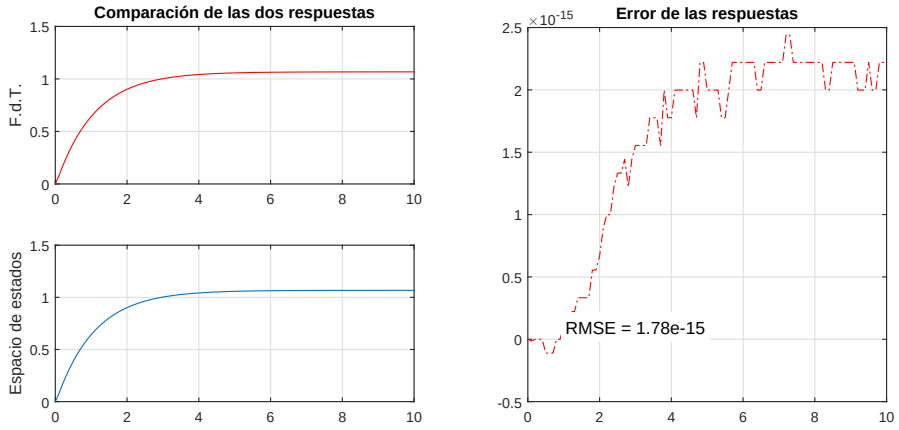


Figura 3.5: Diferencia de ambas respuestas y RMSE

Se observa que se trata de un error muy pequeño, de orden -15, por lo que se puede decir que se trata de una buena respuesta, tanto el espacio de estados como la función de transferencia obtenida por identificación.

Por otra parte, cabe considerar que trabajamos con un sistema discreto y hasta ahora hemos actuado considerando que nuestro sistema es continuo y que presenta una salida analógica. No obstante, los sistemas continuos se pueden representar matemáticamente con las ecuaciones en diferencias. Los sistemas discretos no utilizan la transformada de Laplace, en su caso se utiliza la transformada z. Esta representación resulta más cómoda a la hora de operar y, por tanto,

debemos pasar a modo discreto la anterior representación, con *MATLAB* el código a programar sería:

Listado 3: Representación en el espacio de estados en tiempo discreto

```
%% Espacio de estados discreto del motor

model_motor_z = c2d(model_motor,T,'zoh');
[Az,Bz,Cz,Dz] = ssdata(model_motor_z);

Gd = zpk(model_motor_z); % F.d.t. discreta
```

Al pasar a modo discreto como se ve en el listado 3 obtenemos unas nuevas matrices de estado, en este caso en tiempo discreto, por lo que pasan a tener los siguientes valores:

```
model_motor_z =
```

$$A_z = \begin{pmatrix} 0{,}6054 & -0{,}3623 \\ 0{,}0079 & 0{,}9980 \end{pmatrix}, \quad B_z = \begin{pmatrix} 0{,}00944 \\ 0{,}000051 \end{pmatrix}, \quad C_z = \begin{pmatrix} 0 & 49{,}16 \end{pmatrix}, \quad D_z = 0$$

Se observa que la matriz de salida C no sufre cambios al pasar de tiempo continuo a tiempo discreto. Además, la matriz D sigue siendo cero debido a que el diseño del diagrama de bloques no cambia.

La diferencia entre la transformada de Laplace y la transformada z viene dada por el tipo de ecuaciones que utiliza cada una. Con Laplace utilizamos ecuaciones diferenciales que trabajan en tiempo continuo, en cambio con la transformada z trabajamos con ecuaciones en diferencias. A continuación vamos a ver un ejemplo de ecuación en diferencias correspondiente a la función de transferencia discreta que hemos estimado de nuestro motor:

$$G_d(z) = \frac{0{,}076942\,(z + 0{,}2359)}{(z - 0{,}007469)(z - 0{,}9102)} = \frac{0{,}076942\,z + 0{,}0182}{z^2 - 0{,}9177\,z + 0{,}0068}$$

Si ordenamos términos, y sabiendo que $G_d(z)$ es el cociente entre la transformada z de la salida del sistema y la transformada z de la entrada:

$$G_d(z) = \frac{Y(z)}{U(z)} = \frac{0{,}0769\,z^{-1} + 0{,}0182\,z^{-2}}{1 - 0{,}9177\,z^{-1} + 0{,}0068\,z^{-2}} \tag{3.2}$$

A partir de la ecuación (3.2), multiplicando en el numerador y en el denominador por z^{-2}, despejamos $Y(z)$ y $U(z)$, y se aplica la transformada inversa de z de la forma que sigue:

$$\left[1 - 0{,}9177\,z^{-1} + 0{,}0068\,z^{-2}\right] Y(z) = \left[0{,}0769\,z^{-1} + 0{,}0182\,z^{-2}\right] U(z)$$

$$Y(z) - 0{,}9177\,z^{-1}\,Y(z) + 0{,}0068\,z^{-2}\,Y(z) = 0{,}0769\,z^{-1}\,U(z) + 0{,}0182\,z^{-2}\,U(z)$$

$$y(k) - 0{,}9177\,y(k-1) + 0{,}0068\,y(k-2) = 0{,}0769\,u(k-1) + 0{,}0182\,u(k-2)$$

Por último, despejando $y(k)$ obtenemos la ecuación en diferencias que modela el comportamiento de nuestro sistema:

$$y(k) = 0{,}9177\, y(k-1) - 0{,}0068\, y(k-2) + 0{,}0769\, u(k-1) + 0{,}0182\, u(k-2) \tag{3.3}$$

El objetivo de todo esto es conseguir, a partir de ecuaciones en diferencias, calcular el valor de la acción de control e implementarlo en el ordenador para poder obtener el valor de la salida en cada instante de tiempo.

3.3 Diseño del controlador

En esta sección afrontamos el problema del diseño del controlador que se utilizará en nuestro sistema. Se distinguirán tres tipos diferentes de controlador, todos ellos con estimación del estado:

- Observador.
- Observador más filtro de la medida.
- Filtro de Kalman

Los distintos controladores vendrán definidos por unas especificaciones de funcionamiento y unos polos que escogeremos de modo que podamos observar de qué forma presentan mejores respuestas. En el listado 4 se puede observar que pasos se han seguido para la obtención de los polos del sistema. Cabe destacar que las especificaciones de funcionamiento pueden ser escogidas según las prestaciones que queramos de nuestro sistema (para que sea más rápido disminuiríamos t_s pero podríamos tener problemas de sobrepasamiento, por ejemplo). También hay que decir que para obtener los polos en tiempo discreto se multiplicará por el periodo de muestreo T.

Listado 4: Polos de lazo cerrado

```
%% Especificaciones de funcionamiento

ts = 0.85;
Mp = 0.01;

%% Cálculo de polos

sigma = 4/ts;
wd = (-pi*sigma)/(log(Mp));

p1 = - sigma + wd*j; % A partir de este p1 modificaremos los polos
p2 = p1';
p_i = 10*real(p1);

P = [p1;p2]; % Vector de polos
Pci = [P;p_i]; % Añadimos el polo del integrador a P
Pz = exp(T*Pci); % Polos en el plano z
```

Una vez definido el cálculo de los polos del sistema, se precisa conocer si estamos ante un sistema controlable o no. Para ello, se debe estudiar la controlabilidad del sistema. Esto se puede realizar de forma sencilla con *MATLAB* según se puede ver en el listado 5.

Listado 5: Controlabilidad del sistema

```
%% Calculamos la controlabilidad del sistema

Co = crtb(model_motor_z);
n = length(Az); % Orden del sistema

if rank(Co) < n
fprintf('\nSistema no controlable\n');
break;
end

%% Añadimos el integrador a las matrices de estado del modelo discreto

AA = [Az, Bz; 0*Cz, 0];
BB = [0*Bz; 1];
```

Si el rango de la matriz C_o (matriz de controlabilidad) es menor que el orden del sistema n que estamos calculando, el sistema no será controlable, y por lo tanto, no podremos utilizar un estimador del estado que realice el control digital al sistema. En caso de que el rango de C_o sea mayor o igual que n el sistema será controlable y se podrá seguir con el cálculo del controlador.

Según Ogata 1996, p. 464 "*Sistemas de control tiempo discreto*", los parámetros de diseño del controlador son el vector fila K y la constante integral k_i. El procedimiento para el cálculo de ambos parámetros se determina a partir del siguiente código:

Listado 6: Parámetros de diseño del controlador

```
Kd = place(AA, BB, Pz); % Ubicación de polos

m = 1; % Número de entradas

KK = (Kd + [zeros(m,n), eye(m,m)]) * inv([Az-eye(n,n), Bz; Cz*Az, Cz*Bz]);

K = KK(1:m,1:n)
ki = KK(1:m,n+1:n+m)
```

Los valores de los parámetros de diseño del controlador para las especificaciones indicadas en el listado 4 son:

$$K = \begin{bmatrix} 5{,}6653 & 344{,}7841 \end{bmatrix};$$

$$k_i = 0{,}2504;$$

Con esto quedan definidos los parámetros de diseño del controlador del sistema. Sin embargo, como ya se ha dicho al principio de la sección, se utilizarán tres tipos de controlador: con observador, con observador más filtro de la medida y con filtro de Kalman.

3.3.1 Observador

En muchos casos prácticos, no se dispone de mediciones directas de todas las variables de estado, es decir, sólo son medibles unas cuantas variables de estado de un sistema dado, mientras que las demás no lo son. Puede darse el caso que sólo las variables de salida son medibles. En caso de que esto ocurra, será necesario estimar las variables de estado que no se puedan medir directamente a partir de las variables de salida y las de control.

Para ello es necesario el uso de un observador que estime estas variables, por lo que el listado 7 proporciona el valor del vector K_e que será la ganancia del observador que utilizaremos, y se trata de un valor interno del observador (en la figura 2.2 se puede ver el diagrama de bloques correspondiente al observador de Luenberger).

Listado 7: Diseño del observador

```
Pobs = exp(T*10*P);
Ke = place(Az', Cz', Pobs)'
```

Los polos del observador se han multiplicado por 10 ya que estos polos son más rápidos que los utilizados para calcular el controlador. La función `place` da lugar a un vector fila, pero la ganancia del observador K_e debe ser un vector columna por lo que se realiza la traspuesta del resultado. Por tanto, el valor de K_e será:

$$K_e = \begin{bmatrix} 0{,}0935 \\ 0{,}0085 \end{bmatrix};$$

El diagrama de bloques que corresponde al controlador con observador es el siguiente:

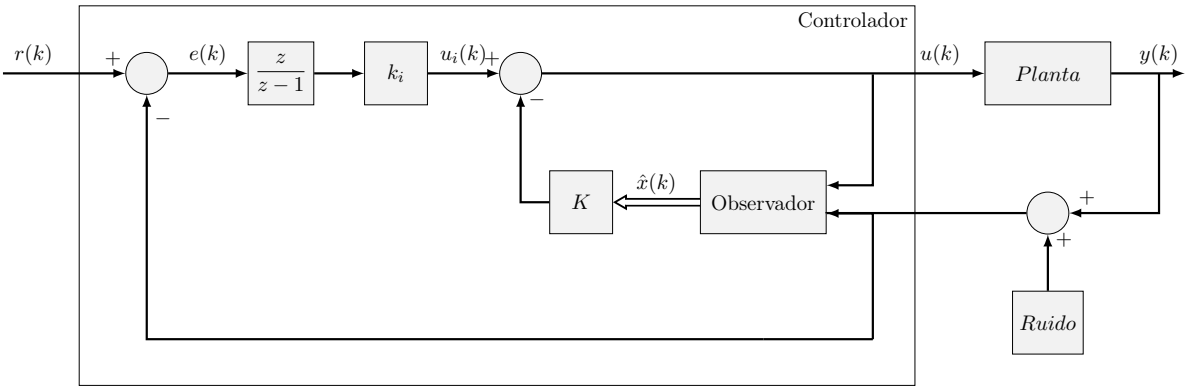


Figura 3.6: Diagrama de bloques con observador

donde K y k_i tienen los valores anteriormente calculados y la *Planta* del sistema es la función de transferencia G.

También se observa que tenemos un ruido en la realimentación, por lo que la señal de control tendrá muchas oscilaciones y no será una respuesta clara. Este problema se soluciona añadiendo un filtro de la medida como se verá en la subsección 3.3.2.

Para que las simulaciones tengan una respuesta parecida a los ensayos que realizaremos más adelante, se ha calculado el valor máximo y mínimo que tendrá el ruido, y se ha considerado en todas las simulaciones que se realizarán:

```
max_ruido = 0.1012;
min_ruido = -0.0968;
```

Estos dos valores serán utilizados en el bloque *Ruido* que se puede ver en los diagramas de bloques tanto del controlador con observador, como del controlador con observador más filtro de la medida, como del controlador con filtro de Kalman.

A continuación se puede ver una figura donde se ha simulado la respuesta que presenta el modelo del motor ante una entrada escalonada en la referencia.

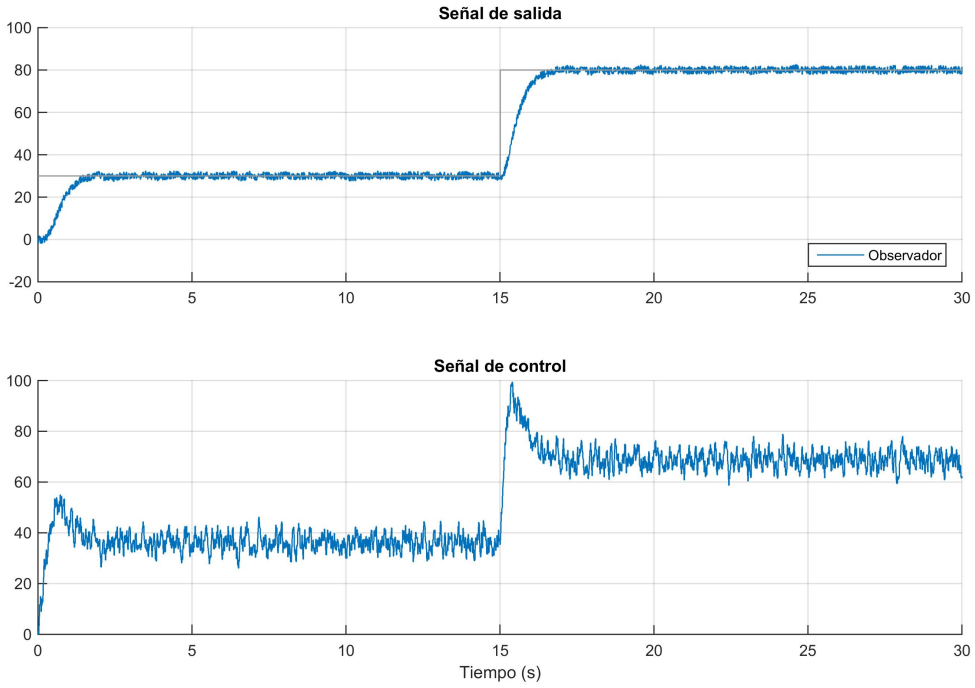


Figura 3.7: Simulación del diagrama de bloques con observador

La simulación de la figura 3.7 corresponde al diagrama de bloques de la figura 3.6 donde se ha introducido una referencia escalonada en porcentaje. La señal de salida cumple con las especificaciones de funcionamiento que se le han asignado ($t_s = 1{,}75;\ M_p = 0{,}01;$).

La acción de control que se observa presenta muchas oscilaciones debido a la cantidad de ruido producido por el propio motor, por lo que no es una buena solución.

Para poder corregir estas perturbaciones en el sistema se necesita la implementación de filtros de la medida, o bien de un filtro de Kalman, como veremos en las siguientes subsecciones.

3.3.2 Observador con filtro de la medida

Todos los sistemas que encontramos poseen ruido y perturbaciones que impiden un correcto funcionamiento de nuestro sistema. Para corregir estos problemas hay que añadir filtros que reduzcan las oscilaciones de la señal de control.

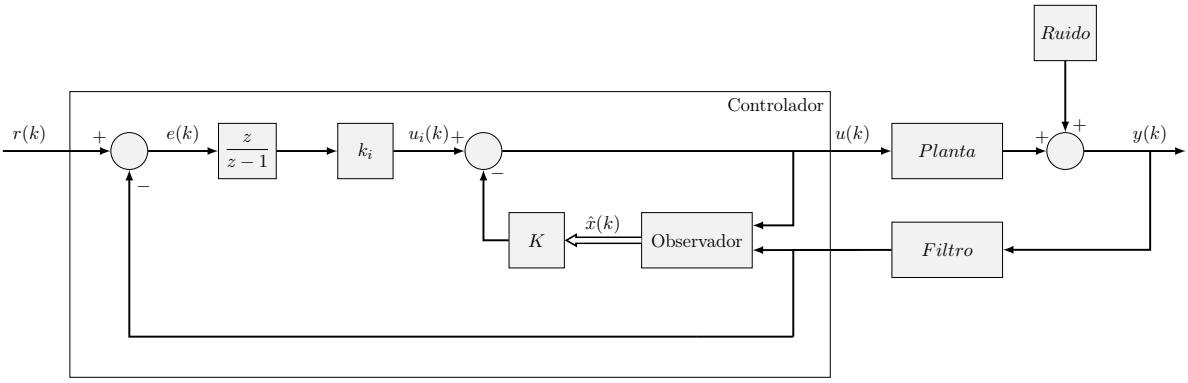


Figura 3.8: Diagrama de bloques real, con filtro y ruido

Como podemos ver en la figura 3.8 se incorpora en la realimentación un filtro de la medida que tomará el valor de la salida más el ruido generado.

La forma de evitar que el ruido sea muy perjudicial para las simulaciones es incorporar un filtro a nuestro sistema, de modo que reduzca las vibraciones que se produzcan en éste.

No obstante, debemos tener en cuenta que al utilizar un filtro de la medida estamos introduciendo dos polos al sistema y dos ceros en el lazo de realimentación. Esto puede provocar que aparezcan picos de sobrepasamiento y oscilaciones en nuestra respuesta. A pesar de ello, la respuesta de la acción de control es mejor que la conseguida al utilizar solamente el observador.

El filtro de la medida que se va a implementar en el sistema es un filtro de Butterworth. En la sección 2.3 hemos diferenciado varios tipos de filtros (Butterworth, Bessel y Chebyshev), y se ha decidido utilizar el de Butterworth ya que se encuentra en un punto intermedio como ya se ha explicado. Las características que presenta este filtro de la medida vienen definidas por el orden del filtro (`nf`) y por la frecuencia normalizada de corte (`wnorm`) como podemos ver en el siguiente código, donde la orden principal es `butter`, que genera una función de transferencia correspondiente al filtro de la medida:

Listado 8: Filtro de la medida

```
nf = 2;          % Orden del filtro
wnorm = 0.045;   % Frecuencia normalizada de corte del filtro

[num,den] = butter(nf,wnorm);

Gfz = tf(num,den,T); % Siendo T el periodo de muestreo
```

La frecuencia normalizada de corte puede variar a nuestra elección, no obstante estas variaciones provocarán que la salida presente más o menos sobrepasamiento y un tiempo de establecimiento menor o mayor. De hecho, si el valor de w_{norm} es muy pequeño aparecen oscilaciones. Teniendo

en cuenta las especificaciones de funcionamiento y el valor de w_{norm} el sistema se puede volver inestable.

Para las condiciones escogidas, la función de transferencia discreta del filtro de la medida corresponde a:

$$G_{fz}(z) = \frac{0{,}00454 + 0{,}00907\, z^{-1} + 0{,}00454\, z^{-2}}{1 - 1{,}801\, z^{-1} + 0{,}8188\, z^{-2}}$$

Esta función de transferencia da una ecuación en diferencias que se utilizará posteriormente en la implementación de la placa de Arduino. Esta ecuación es:

$$y(k) = 1{,}8006\, y(k-1) - 0{,}8188\, y(k-2) + 0{,}00454\, y_m(k) + 0{,}00907\, y_m(k-1) + 0{,}00454\, y_m(k-2)$$

En la figura 3.9 se ha realizado una comparación entre el controlador con observador y el controlador con observador más filtro de la medida.

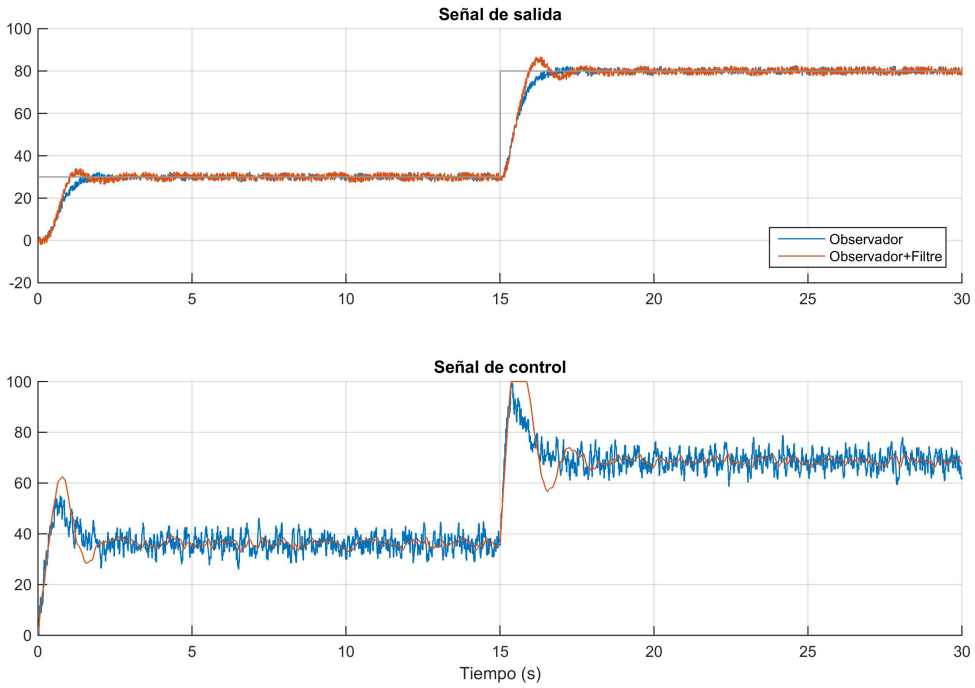


Figura 3.9: Simulación del controlador con observador y del controlador con observador más filtro de la medida

Las condiciones que se han utilizado para realizar las simulaciones son:

$$\text{Observador} = \begin{cases} t_s = 1{,}75 \\ M_p = 0{,}01 \end{cases}; \qquad \begin{matrix}\text{Observador} \\ \text{más filtro}\end{matrix} = \begin{cases} t_s = 1{,}75 \\ M_p = 0{,}01 \\ w_{norm} = 0{,}045 \\ n_f = 2 \end{cases};$$

Si se analiza detenidamente la señal de control de la figura, se puede ver claramente que la línea roja (acción de control del controlador con observador más filtro de la medida) consigue eliminar el error de forma notable, y por tanto, la respuesta es bastante aceptable, dentro de las condiciones marcadas. En cambio la línea azul (misma respuesta que en la figura 3.7) presenta muchas oscilaciones debido al ruido generado por el motor. Cabe destacar que la señal de la salida del controlador con observador más filtro de la medida presenta oscilaciones debido a que el valor de la frecuencia normalizada de corte es muy pequeño, lo que, además, provoca que se introduzcan dos polos al sistema que lo vuelven más inestable.

Existe una clara diferencia entre realizar una simulación con filtro a realizarla sin filtro como hemos podido observar. A pesar de ello, la acción de control sigue sin ser demasiado buena ya que aún aparecen oscilaciones que afectan a la señal de control.

3.3.3 Filtro de Kalman

En la subsección 3.3.2 se ha visto que la acción de control que presenta el controlador con observador más filtro de la medida no es tan mala. Sin embargo, para llegar a esa solución es necesaria la implementación de dos elementos, es decir, se tienen que realizar los cálculos para el observador y también para el filtro de la medida como ya hemos visto anteriormente. Una forma de evitar que esto ocurra es utilizar el filtro de Kalman. Por su parte, éste necesita dos valores que no son utilizados con el observador o observador más filtro de la medida. Estos parámetros son R_1 y R_2 que corresponden a términos independientes de la función de covarianza. Los valores escogidos para estos parámetros son: $R_1 = 0{,}00$ y $R_2 = 1{,}50$.

Para la simulación del controlador con filtro de Kalman se hará una comparación entre éste y el observador más filtro de la medida.

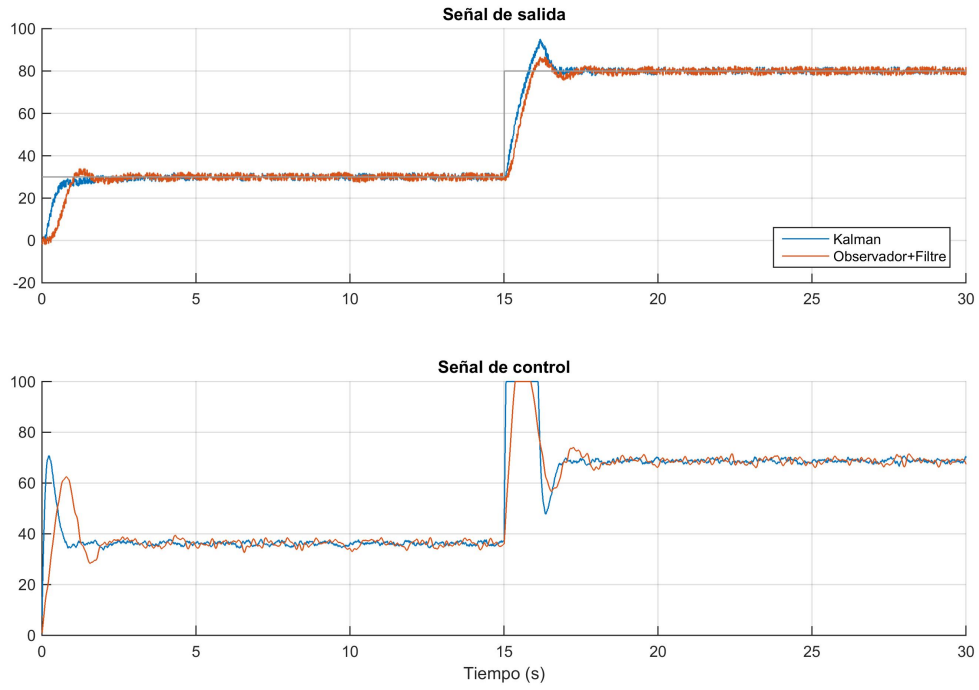


Figura 3.10: Simulación del controlador con filtro de Kalman y del controlador con observador más filtro

Para la simulación de la figura 3.10 se ha considerado la misma configuración que la utilizada en el diseño con observador más filtro, en cambio para el filtro de Kalman hemos reducido el tiempo de establecimiento ya que es un controlador al que se le puede exigir más ($t_s = 0{,}85$).

Se puede observar que, sin ser del todo buena la acción de control que obtenemos con el filtro de Kalman, es mucho mejor que la que se obtiene con el controlador con observador más filtro de la medida. Las oscilaciones que aparecen en la señal de control del observador más filtro no aparecen con el filtro de Kalman, además de que la acción de control es más suave.

Se han probado varias configuraciones para el filtro de la medida, no obstante, ninguno de ellos es capaz de filtrar la señal de salida de modo que obtengamos una señal de control como la que se obtiene con el filtro de Kalman.

Ambas simulaciones están hechas sin activar herramientas que eviten el efecto *windup* por lo que al exigirle más al filtro de Kalman (recordar que hemos reducido el t_s en casi un segundo) la señal de control se satura, de ahí la aparición del pico de sobrepasamiento.

De modo que, para evitar que esto ocurra, a continuación se va a realizar una simulación donde se vea la diferencia entre el filtro de Kalman con y sin $anti - windup$.

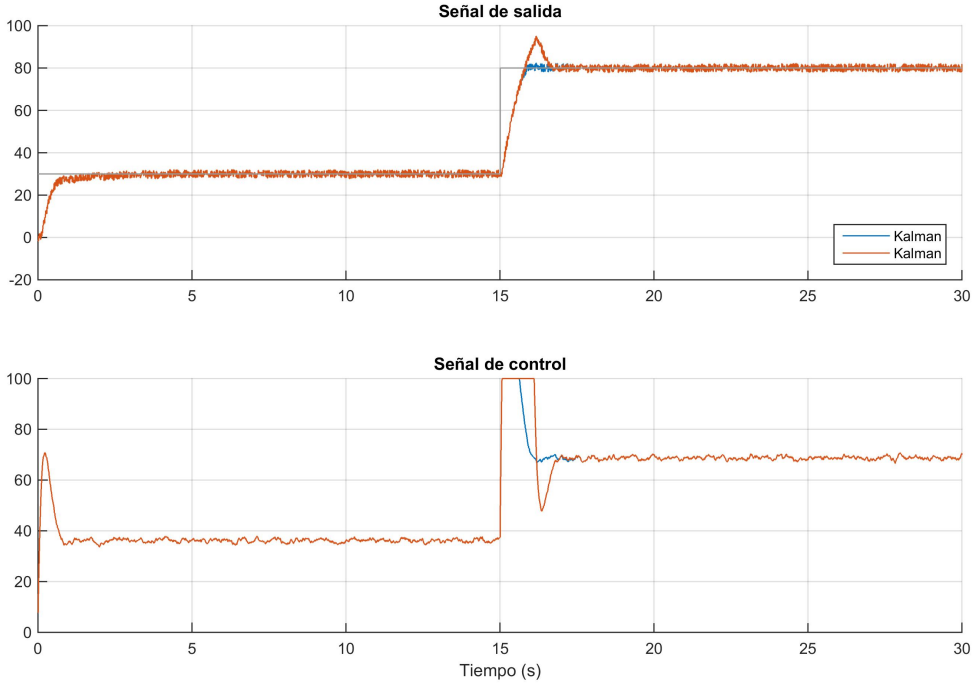


Figura 3.11: Simulación del filtro de Kalman, aplicando *Back calculation* y sin *anti-windup mode*

En la figura 3.11 se puede ver que a pesar de ser exigentes con el sistema pidiéndole un $t_s = 0{,}85$ s es capaz de obtener una acción de control muy buena, sin apenas ruido, y en la que no aparece saturación ya que el método *Back calculation* actúa sobre el sistema (línea azul).

3.4 Arduino Due

Se ha de programar el Arduino de modo que actúe como un controlador. Sustituiremos nuestro controlador por la placa de Arduino para que sea éste el que realice el algoritmo de controlar el motor, ver figura 3.12.

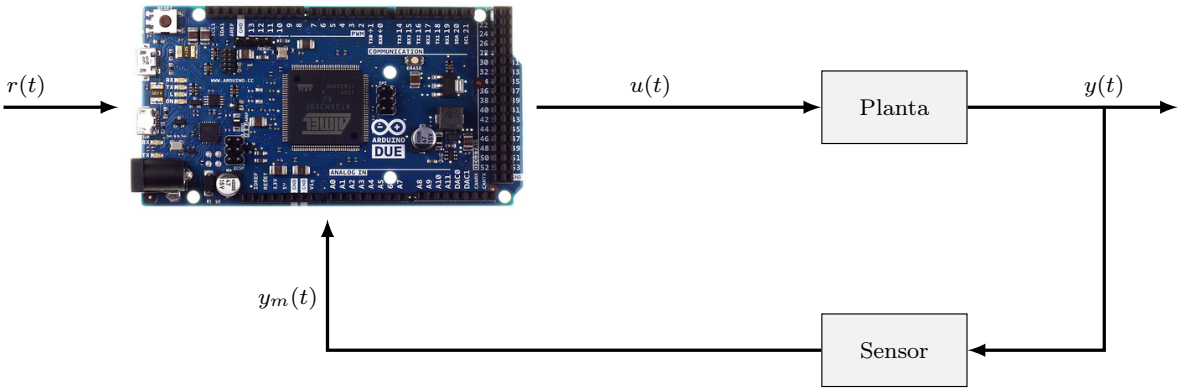


Figura 3.12: Sustitución del controlador por la placa de Arduino

A la hora de realizar el programa para el Arduino hay que tener en cuenta los siguientes factores:

- **Las entradas y las salidas del motor:** debemos tener muy claro que entradas y salidas tenemos que utilizar para hacer el control de la velocidad de nuestro motor. En nuestro caso, necesitaremos 2:

$$\text{ENT\_TACOMETRO} = \text{A0}$$
$$\text{ENT\_REFERENCIA} = \text{A1}$$

- **Cambios de escala:** se ha de tener en cuenta las zonas muertas del motor, como también el cambio de giro de éste, para que Arduino sea capaz de representar los datos leídos desde el motor y al contrario.
- **Como actuar sobre el sistema:** debemos tener claro como se pasarán los datos de nuestro controlador calculado al motor. Para ello se ha realizado este pequeño código:
- **Periodo de tiempo a ejecutar el controlador:** se quiere implementar nuestra función del controlador cada x milisegundos, de modo que el programa principal interrumpa el proceso para recalcular la función del controlador, de esta manera el programa principal podría realizar otras funciones.
- **Cambio de referencia:** para poder compara el control ideal (dado por *MATLAB* y *Simulink*) con el real (dado por el motor), debemos programar la misma referencia interna a Arduino que la establecida con *MATLAB*.

3.5 Monitorización del Arduino

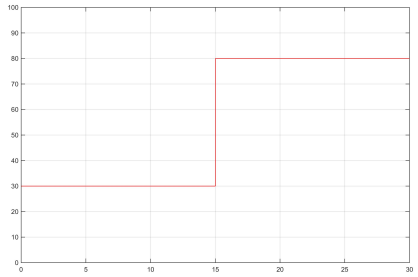


Figura 3.13: Escalones de la referencia para la monitorización

Para la monitorización del Arduino se ha utilizado *LabVIEW* introduciendo una secuencia de escalones aleatorios en la entrada de referencia de modo que se pueda ver la respuesta del controlador a implementar.

Cada una de las respuestas vendrán dadas en porcentaje, tanto la referencia, como las salidas de la medida (Arduino) y del modelo (Simulación), y la acción de control. Esto quiere decir que el 100 % serán el máximo que resiste la placa de Arduino Due ($3{,}3V$).

Los escalones que se han utilizado para la monitorización se pueden ver en la figura 3.13.

A partir de estos escalones de referencia se van a realizar varios ensayos en los que veremos la respuesta de diferentes controladores, controladores de los que ya se ha hablado anteriormente.

3.5.1 Controlador con observador

El primer ensayo que se va a realizar es del controlador con observador. Para poder ver de forma más clara la solución del ensayo se ha hecho una monitorización de 30 segundos mediante *LabVIEW* donde se han recogido los datos obtenidos en un documento .csv y que posteriormente se leerá en *MATLAB*, donde se hará una comparación con el modelo del motor.

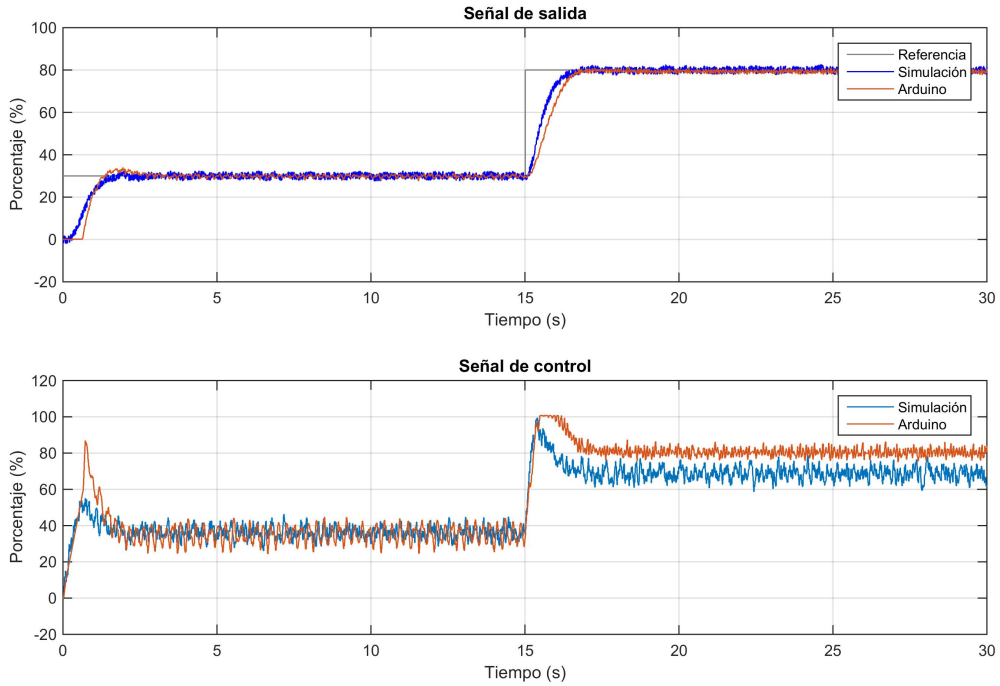


Figura 3.14: Monitorización del controlador con observador

En la figura 3.14 se distinguen dos gráficas. En la primera se observa la representación de la referencia (color gris) que corresponde con los escalones que se utilizan para la simulación, la representación de la salida del modelo calculado anteriormente en la sección 3.1 (simulación) y por último la representación de la señal de salida de la medida (Arduino). En la segunda se puede ver la acción de control del modelo del motor (color azul) y la acción de control de la medida tomada (color rojo).

El principal problema que surge en este ensayo es la cantidad de ruido que hay en la acción de control, es decir, el ruido que produce el propio motor no es filtrado por ningún elemento del sistema. Esto provoca una señal de control muy mala, por lo que será necesario la implementación de algún tipo de filtro para poder eliminar el ruido.

Se aprecia que tanto la simulación como el Arduino cumplen con las especificaciones de funcionamiento ($t_s = 1{,}75$ y $M_p = 0{,}1$), pero como ya se ha dicho, la señal de control presenta mucho ruido. También cabe destacar la discrepancia que hay entre ambas señales, esto se debe a la llamada "*dead zone*" (en la simulación se ha considerado este factor para que ambas respuestas fuesen iguales).

3.5.2 Controlador con observador más filtro de la medida

Como ya se ha visto en la subsección 3.3.2 el controlador con observador más filtro de la medida tiene una acción de control mejor que el controlador con observador. A continuación se puede ver el ensayo realizado donde se ve la monitorización del controlador a partir de la placa de Arduino comparada con la respuesta que se obtiene a partir del modelo del modelo que se ha obtenido con anterioridad.

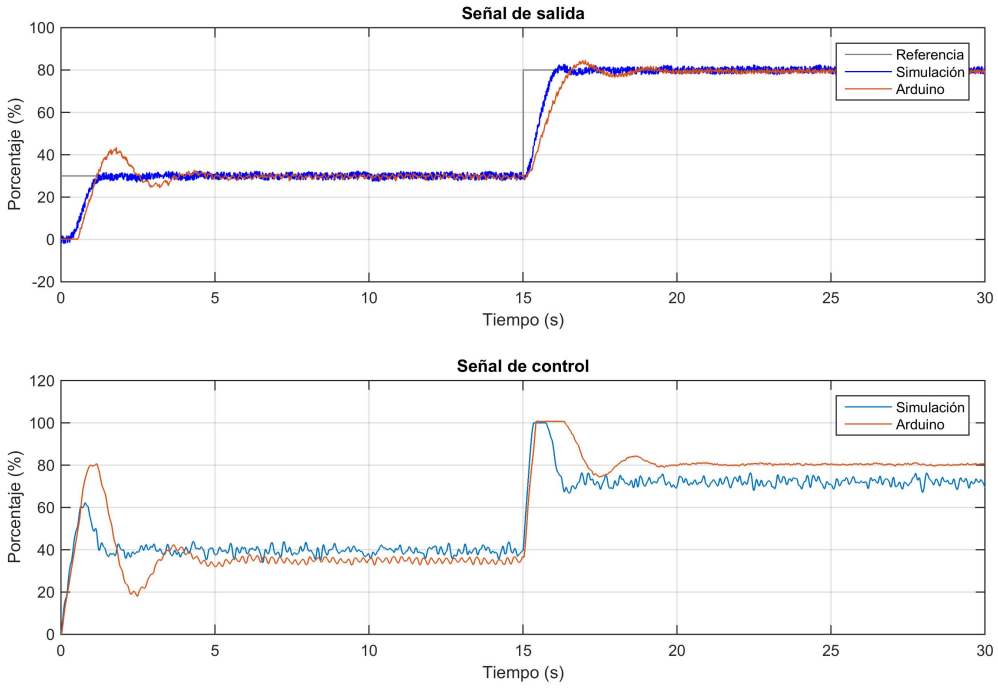


Figura 3.15: Monitorización del controlador con observador más filtro de la medida

En la figura 3.15 podemos ver la monitorización con Arduino del controlador con observador más filtro de la medida y la simulación del mismo con *MATLAB*. Se puede observar que la acción de control de ambas respuestas presenta mucho menos ruido que la monitorización de la figura 3.14. Con esto se destaca el trabajo que realiza el filtro de Butterworth, eliminando parte del ruido introducido. A pesar de ello, el ruido sigue estando muy presente en la acción de control y sigue presentando oscilaciones, por lo que no es una respuesta del todo aceptable. Esto implica que tendremos que buscar una solución para poder filtrar de forma más efectiva el ruido generado por el motor.

Cabe destacar que el observador con filtro de la medida introduce 2 polos en el sistema de lazo cerrado y por tanto, es más difícil de controlar. Además, vuelve a existir una discrepancia entre las dos señales de control teniendo en cuenta que la configuración es la misma ($t_s = 1{,}75$, $M_p = 0{,}01$, $nf = 2$ y $w_{norm} = 0{,}075$).

3.5.3 Controlador con filtro de Kalman

Después de ver qué ocurre al introducir un filtro de la medida en nuestro controlador con observador, se va a realizar un ensayo con un filtro de Kalman, ya que es la finalidad del control digital que se está realizando.

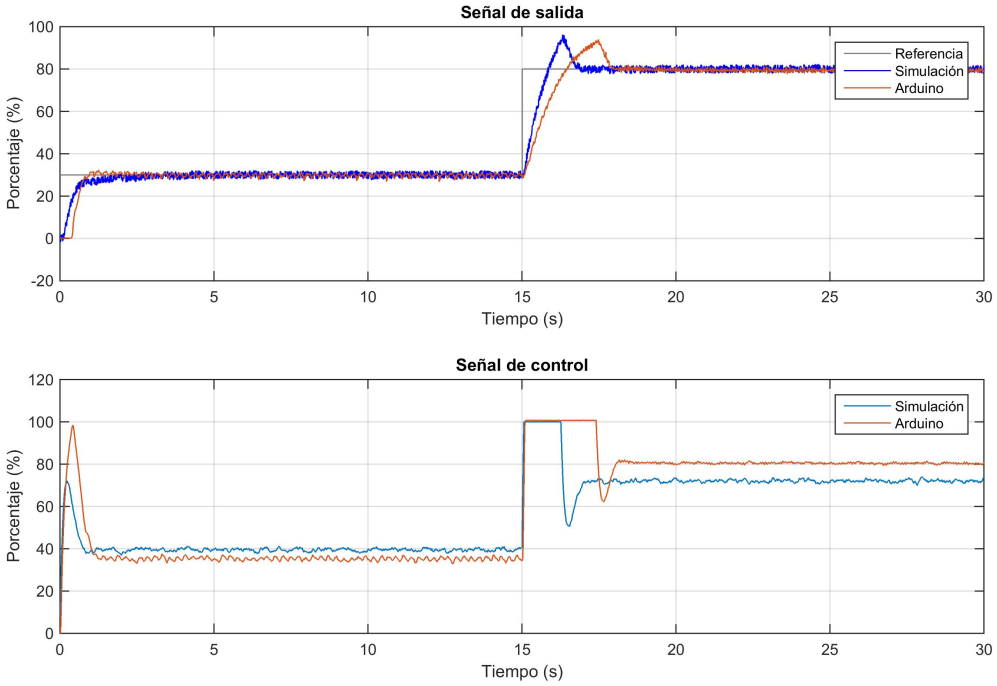


Figura 3.16: Monitorización del controlador con filtro de Kalman

En este ensayo (figura 3.16) se puede ver la monitorización del controlador con filtro de Kalman con Arduino, y su simulación con *MATLAB*.

Como se ha visto, el filtro de Kalman es capaz de trabajar con tiempos de establecimiento bajos, por lo tanto, se ha considerado un $t_s = 0{,}85$ s para esta monitorización.

Existe una clara diferencia entre ambas acciones de control, sobretodo por la banda muerta que aparece en la monitorización con Arduino. Esto provoca que los tiempos de establecimiento sean diferentes.

Cabe destacar que ambas acciones de control presentan poco ruido debido a que el filtro de Kalman está realizando su trabajo, es decir, a parte de realizar la estimación del estado correspondiente filtra el ruido que produce nuestro motor y evita que hallan oscilaciones en nuestra acción de control.

Las condiciones del filtro de Kalman se definen mediante los parámetros R_1 y R_2, varianza del ruido del proceso y varianza del ruido de la medida, respectivamente. Cambiar estos valores puede afectar al filtrado del ruido, ya sea para tener mayor o menor filtrado en la acción de control.

Otro factor a destacar es la saturación de las acciones de control. El último escalón produce que éstas se saturen, por lo que en la señal de salida ($t = 15s$) el tiempo de establecimiento aumenta considerablemente, y además aparece un pico de sobrepasamiento que se debe corregir. Para evitar que esto ocurra y sea perjudicial para el control digital, se utilizan métodos $anti-windup$, ver sección 1.4.

Tener saturación en la acción de control hace que no tengamos una buena solución del problema, ya que el objetivo es que obtengamos una respuesta sin ruido, sin picos de sobrepasamiento y rápida. El ruido se ha conseguido filtrar al utilizar el propio filtro de Kalman. Sin embargo, para que la respuesta no presente picos y sea rápida se tiene que aplicar algún método $anti-windup$, como se ha visto en la subsección 3.3.3. Para ello, en Arduino se ha escrito el siguiente código en el que se escogerá que método se va a utilizar:

Listado 9: Algoritmo en C para elegir el método *anti-windup*

```
// Calculamos la acción integral

switch (awm){
    case 0:
        ui_k = ui_k + ki*e_k; // Sin anti-windup
        break;
    case 1:
        ui_k = ui_k + ki*e_k - kb*(v_k - u_k); // Back calculation
        break;
    case 2:
        ui_k = ui_k + ki*e_k*((u_k==v_k)||(e_k*u_k<0)); // Clamping
        break;
}
```

donde `kb` es una variable perteneciente al método *Back calculation* y `awm` es el método $anti-windup$ que se escogerá con anterioridad en el programa Arduino: $0 \rightarrow$ *Sin anti-windup*, $1 \rightarrow$ *Back calculation*, $2 \rightarrow$ *Clamping*.

En el ensayo que se ha realizado para ver como actúa el $anti-windup$ se ha escogido `awm = 1`, por lo tanto se habrá elegido el método *Back calculation*. De este modo, la representación gráfica del ensayo mediante Arduino y de la simulación del modelo del motor quedaría como sigue:

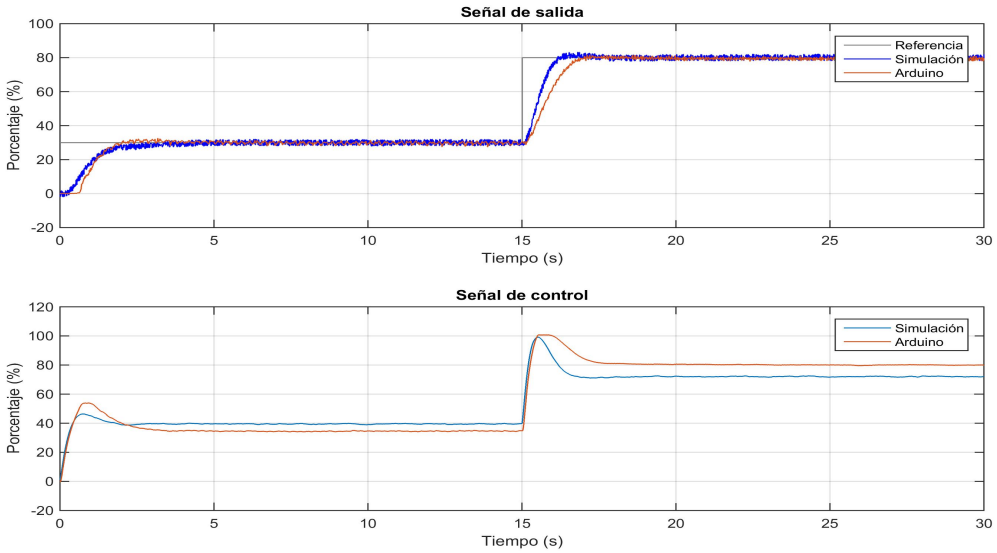


Figura 3.17: Monitorización del controlador con filtro de Kalman aplicando $anti - windup$

Las condiciones con las que se ha realizado el anterior ensayo del filtro de Kalman con $anti - windup$ son:

$$\text{Especificaciones de funcionamiento} = \begin{cases} \texttt{ts = 1.75;} \\ \texttt{Mp = 0.01;} \\ \texttt{R1 = 0.00;} \\ \texttt{R2 = 1.50;} \end{cases}$$

El ensayo se ha realizado utilizando un valor de la constante correspondiente al método *Back calculation* (K_b) igual a 0,75. Si observamos la figura 3.17 y la figura 3.16 en la que no se utiliza $anti - windup$ *mode* se puede apreciar que la acción de control de la simulación no satura casi nada en la figura 3.17. Esto se debe a que está actuando el $anti - windup$. Además, el filtro de Kalman permite que la acción de control sea prácticamente limpia, sin apenas oscilaciones, por lo que habremos conseguido eliminar el ruido. Cabe destacar también que tenemos una discrepancia entre las dos señales de control debido a la banda muerta que se genera en el sistema, aún así la discrepancia es mínima.

Se puede apreciar que el tener un tiempo de establecimiento mayor también ayuda a reducir que aparezca el efecto *windup*. No obstante, nosotros queremos pedirle más al sistema para ver que capacidad de reacción tiene el filtro de Kalman, de modo que vamos a intentar ser más agresivos con el sistema, es decir, vamos a forzar al sistema para que nos de una respuesta más rápida para ver si es capaz de soportarlo. Para ello se han escogido las mismas condiciones que se han utilizado para la figura 3.16, ya que para esas condiciones aparecía *windup*, y aplicaremos el método *Back calculation* para corregirlo:

$$\text{Especificaciones de funcionamiento} = \begin{cases} \texttt{ts = 0.85;} \\ \texttt{Mp = 0.01;} \\ \texttt{Kb = 2*ki} \end{cases}$$

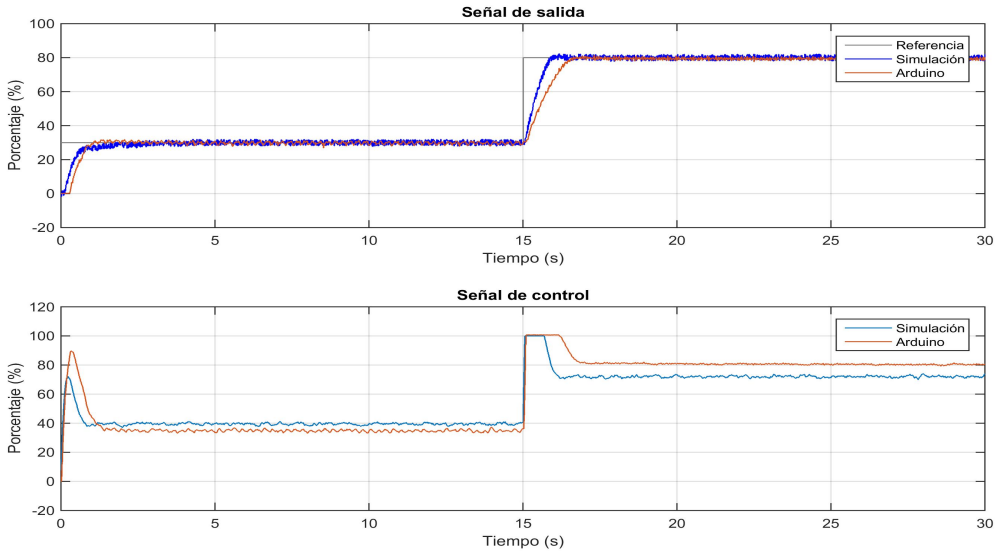


Figura 3.18: Monitorización del controlador con filtro de Kalman más rápido con *awm Back Calculation*

En la figura 3.18 se puede ver la representación gráfica del filtro de Kalman teniendo en cuenta las anteriores especificaciones de funcionamiento.

El método $anti-windup$ que se ha utilizado es *Back calculation* (siendo $kb = 2\,ki$) ya que a la hora de trabajar ante la saturación de la acción de control realiza operaciones menos agresivas. Este método no bloquea la integración, sino que la frena y la suaviza para evitar que aparezca saturación.

Se puede ver, a diferencia de la anterior representación (figura 3.17), que el pico de sobrepasamiento aumenta un poco más aun así el sistema se mantiene con una saturación pequeña y cumpliendo con el tiempo de establecimiento. Por otra parte, ambas acciones de control siguen siendo muy limpias, por lo tanto se puede decir que estamos ante un sistema rápido y sin ruido en la acción de control.

Se han realizado varios ensayos con distintos valores, tanto para el tiempo de establecimiento t_s como para el pico de sobrepasamiento M_p, y finalmente se ha decidido optar por los valores utilizados, y nombrados, con anterioridad. No obstante, se podría realizar un cálculo de polos óptimo para nuestro sistema, el cual colocaría los polos en el punto donde mejor trabaje el control del sistema.

Capítulo 4

Conclusiones

En este capítulo se pueden ver las conclusiones que se obtienen tras realizar las simulaciones y los ensayos en tiempo real sobre el prototipo real de cada controlador analizado.

Como se ha podido ver el controlador con observador presenta una respuesta del sistema muy mala debido a que el ruido no es filtrado por nada, es decir, el ruido se realimenta y la acción de control no es capaz de reducir el error que se genera. En cambio, el controlador con observador es capaz de cumplir con las especificaciones de funcionamiento, aunque para ello, la señal de control esté llena de perturbaciones como se ha demostrado tanto en la simulación correspondiente como en el ensayo de tiempo real, ver subsección 2.5.1.

Por lo que respecta al controlador con observador más filtro de la medida, cabe destacar que presenta una buena respuesta en la señal de control, cosa que no se puede decir si el filtro de la medida no estuviera implementado en nuestro sistema. A pesar de ello, la solución obtenida sigue presentando oscilaciones, además de la aparición de saturación que genera picos de sobrepasamiento y más lentitud al sistema. Aún aplicando las herramientas $anti-windup$ pertinentes, sigue sin tener una respuesta del todo limpia, y no consigue eliminar del todo el ruido generado por el motor.

Para conseguir una buena acción de control es necesario reducir mucho el valor de la frecuencia normalizada de corte (lo que puede traer problemas de inestabilidad al sistema), o bien aumentar el tiempo de establecimiento para que el sistema no sufra tanto. Aumentar la frecuencia normalizada de corte implicaría mayores perturbaciones en la acción de control, por lo que también sería un inconveniente.

Por otra parte, el filtro de Kalman depende de los parámetros de diseño del controlador, además de los valores de la varianza (R_1 y R_2). Con las especificaciones de funcionamiento escogidas y los parámetros de diseño definidos para el controlador se ha obtenido una acción de control bastante buena, tanto en el ensayo de tiempo real mediante $LabVIEW$ como en la simulación realizada con $MATLAB$, ver figura 3.16. A pesar de que el filtrado que realiza el filtro de Kalman en el ensayo mediante $LabVIEW$ en la señal de control es muy bueno, aparece el problema de la saturación, de ahí que se hayan añadido herramientas $anti-windup$ de modo que se pueda reducir el pico de sobrepasamiento que aparece y el tiempo de establecimiento se cumpla. Además, si la configuración del mismo tiene en cuenta los métodos $anti-windup$ que eviten que llegue a saturar el sistema, se tendría una respuesta filtrada, rápida y sin prácticamente saturación, ver figura 3.17.

Por último, cabe decir que una de la mejores configuraciones por las que se puede optar para tener un sistema capaz de soportar grandes cambios en la referencia y oscilaciones internas como el ruido o perturbaciones es el filtro de Kalman, ya que no es necesaria la implementación de un observador y de un filtro de la medida que filtre dichos problemas (el propio filtro de Kalman realiza ambas funciones). Además, la acción de control que se obtiene al realizar ensayos con el controlador con filtro de Kalman es muy limpia, cosa que no se consigue con el resto de controladores.

Capítulo 5

Anexos

A continuación se verá que pasos se han seguido de los diferentes programas utilizados (*MATLAB*, LabVIEW y Arduino) para obtener las conclusiones pertinentes al Trabajo Fin de Grado que se ha tratado.

Hay dos carpetas a partir de las cuales se ha ido ejecutando el trabajo:

- **Motor_DC_Javi_Ferrandiz**

  Dentro de esta carpeta podemos encontrar archivos de *MATLAB* y de LabVIEW. Además de otras carpetas donde se han guardado datos, figuras y funciones.

  - dades
  - figures
  - matlab

- **Arduino**

En la primer se encuentran los archivos, a partir de los que se han ido realizando el trabajo. Los pasos siguen el siguiente proceso:

1. `pas_01_entrades_identificacio_motor.m`

   Se prepara un escalón en la referencia para identificar el modelo del motor.

2. `pas_02_assaig_llac_obert.vi`

   Monitorizamos el escalón obtenido en el paso anterior.

3. `pas_03_identifica_valida_motor.m`

   A partir de los datos obtenido en el paso 2 que se encuentran en la carpeta *dades* obtenemos la f.d.t. del motor.

4. `pas_04_disseny.m`

   Se realizan los cálculos del controlador y observador, y se lanzan las simulaciones.

5. `pas_05_representa_assaig_controlat.m`

   Se obtienen los datos para realizar los ensayos con el motor.

6. `Monitoritzar Motor dc - SP.vi`

   Llegados a este paso ha sido necesaria la utilización de la carpeta Arduino de la forma que sigue:

   a) `controlador_Observador.ino`

   b) `controlador_Observador_filtre.ino`

   c) `controlador_Kalman.ino`

   Cada uno de estos archivos se utiliza para la monitorización de cada proceso, se obtienen datos que se guardan en la carpeta *dades*.

7. `pas_07_representa_assaig_controlat_observador.m`

   A partir de los datos generados por el paso 6_a se representa el controlador con observador.

8. `pas_07_representa_assaig_controlat_filtre_mesura.m`

   A partir de los datos generados por el paso 6_b se representa el controlador con observador más filtro de la medida.

9. `pas_07_representa_assaig_controlat_Kalman.m`

   A partir de los datos generados por el paso 6_c se representa el controlador con filtro de Kalman.

Bibliografía

Åström, Karl J y Björn Wittenmark (2013). *Computer-controlled systems: theory and design*. Courier Corporation (vid. págs. 13, 14, 17).

Friedland, Bernard (1995). *Advanced control system design*. Prentice-Hall, Inc. (vid. pág. 13).

— (2012). *Control system design: an introduction to state-space methods*. Courier Corporation (vid. pág. 2).

Ogata, K. (1996). *Sistemas de control tiempo discreto*. Segunda Edición. Prentice-Hall Internacional (vid. pág. 28).

Visioli, Antonio (2006). *Practical PID control*. Springer (vid. pág. 7).

Zverev, AI y HJ Blinchikoff (1976). *Filtering in the Time and Frequency Domain* (vid. pág. 12).