UNIVERSIDAD MAYOR DE SAN ANDRES

FACULTAD TECNICA

CARRERA: ELECTRONICA Y TELECOMUNICACIONES

EXAMEN DE GRADO

NIVEL LICENCIATURA

TRABAJO DE APLICACIÓN

"SISTEMA DE CONTROL DE MOTORES EN TIEMPO REAL MEDIANTE MATLAB"

Postulante: Beatriz Ramírez Sainz

La Paz- Bolivia

2011

DEDICATORIA

El presente trabajo de titulación está dedicado a mi familia, que con su esfuerzo y cariño ha logrado entregarme el apoyo necesario para cumplir mis sueños y metas.

A mi madre Teresa Sainz por su amor, apoyo incondicional y dedicación.

A mi padre Luís Ramírez (Q.E.P.D.) por el amor y valores que me inculcó.

A mi hermano Marcelo Ramírez Sainz quién a pesar de la distancia sé que está conmigo.

A mis amigas y amigos por los momentos bonitos y las palabras de apoyo en todo momento.

AGRADECIMIENTOS

Dar las gracias sobre todas las cosas a Dios todo poderoso que está con migo en todo momento y lugar.

Agradecer a mi madre Teresa Sainz por su esfuerzo, comprensión y cariño, guiándome por el camino del bien y apoyándome profesionalmente.

Gracias al cuerpo docente por sus enseñanzas, siendo fundamental para mi desarrollo como profesional.

INDICE

Pág.

RESUMEN…… 1

1. PLANTEAMIENTO DEL PROBLEMA…… 2

2.JUSTIFICACION DEL TRABAJO…… 2

3. OBJETIVOS…… 2

3.1 OBJETIVO GENERAL…… 2

3.2 OBJETIVOS ESPECIFICOS…… 2

4. FUNDAMENTACION TEORICA…… 3

4.1 Introducción a MATLAB…… 3

4.2 Real Time Windows Target …… 4

4.2.1 Aplicación Real-Time …… 4

4.2.2 Análisis y adquisición de señales…… 5

4.2.3 Ajuste de parámetros …… 6

4.2.4 Entorno del software …… 6

4.2.4.1 Simulación en tiempo no real…… 6

4.2.4.2 Ejecución en tiempo real …… 7

4.2.5 Simulink en modo externo …… 8

4.2.6. Buffers y transferencia de datos …… 8

4.3 Elementos físicos de la aplicación en Tiempo Real …… 9

4.3.1 PC…… 10

4.3 1.1 Placa de adquisición de datos …… 10

4.3.1.1.1 Sensores y actuadores…… 11

4.3.1.1.2 Hardware de adquisición de datos …….. 11

4.3.1.1.3 Hardware de acondicionamiento de la señal …… 12

4.3.1.1.4 La computadora ……..…… 12

4.3.1.1.5 Software ……..…… 12

4.3.2 Driver del Motor CC …… 12

4.3.2.1 Microcontrolador…..…… 13

4.3.2.2 GAL(GateArrayLogic) ……..…… 13

4.3.2. Puente en H …….……. 13

4.3.2.4 Cuadrantes de trabajo de un motor …… 14

4.4 Motor de corriente continua……..…… 15

4.4.1Principio de Funcionamiento…… ……. 17

4.4.2 Fuerza contra electromotriz inducida en un motor …… 18

4.4.3 Número de escobillas ……. 19

4.4.4. Sentido de giro…….. 19

4.4.5. Tipos de motores D.C ……..… 20

4.4.5.1.*MOTOR SHUNT*…….. 21

4.4.5.2. *MOTOR SERIE* ……..…….. 21

4.4.5.3 *MOTOR COMPUND(Compuesto)…..……...* 21

4.4.5.4 MOTOR SHUNT ESTABILIZADO…..…….. 22

4.5. Velocidad, Torque y Potencia (HP)…….. 23

4.5.1. Ecuación general del motor ……..…… 24

4.6. Módulo de Control…… 27

4.7. Elementos de Software……. 28

4.7.1 Herramienta GUIDE…….. 28

5 MEMORIA DE CALCULO…….. 30

5.1Modelado de la planta ……..…….. 30

5.1.1 Aliasing……..…….. 32

5.1.1.1 Aliasing en Fenómenos periódicos…….. 32

5.1.2. Muestreo de una Señal sinusoidal…… 33

5.1.3. Funciones de transferencia discretas…..…….. 35

5.2 Diseño de controladores…….. 36

5.2.1 Proporcional integral derivativo……..…… 37

5.2.1.1.Funcionamiento…….…… 38

5.2.1.2 Proporcional…… 40

5.2.1.3 Integral……..…….. 41

5.2.1.4.Derivativo……..…….…… 42

5.2.2 Controlador P…….……..…… 44

5.2.3. Controlador PI…….. …… 47

5.3. Funcionamiento del interfaz Gráfico……..…….…… 49

5.3.1 Elección del modelo de trabajo….. …… 50

5.3.1.1 Modelo de Lazo cerrado con Controlador y carga…… 51

5.3.1.2 Modelo en Lazo Abiero……. 52

6. CONCLUSIONES……. 44

7. RECOMENDACIONES…… 44

BIBLIOGRAFIA…… 45

ANEXO

RESUMEN.

Actualmente dentro de la electrónica industrial se está implementando programas para mayor facilidad como en este caso el de plantear la realización de un entorno gráfico que permita el control ajustable de la velocidad de una máquina de corriente continua en tiempo real.

Para el desarrollo de la aplicación en tiempo real, se trabaja con una herramienta de MATLAB llamada *GUIDE* (Graphical Use Interface Development Environment). Esta herramienta está pensada para desarrollar GUIs, fácil y rápidamente haciendo sencillo el diseño y presentación de los controles de la interfaz, reduciendo la labor en el momento de seleccionar, deshacer, arrastrar y centrar controles, así como la personalización de las propiedades de estos.

El proceso a seguir para el desarrollo de un programa mediante GUIDE es que una vez se tienen todos los controles en posición, se editan las funciones de llamada (Callback) de cada uno de ellos, escribiendo el código de MATLAB que se ejecutará cuando el control sea utilizado.

Cuando se crea una GUI se generan dos ficheros: Un archivo .FIG, que es el que contiene los elementos gráficos así como las propiedades de la interfaz. Un archivo .M que es el contiene el código con las correspondencias de los botones de control de la interfaz.

Se dimencionará dos tipos de controladores: el controlador P que introduce una ganancia a la entrada de la planta y un controlador PI que incrementa el tiempo de establecimiento y elimina el error en estado estacionario.

Para los elementos físicos de la aplicación puede estar controlado por el driver de corriente continua, los módulos, alimentación externa, realimentación, PC.

1. PLANTEAMIENTO DEL PROBLEMA.

Hoy en día se ha visto que en trabajos anteriores y sobre todo actuales, es desconocida la realización de programas con sistemas reales y en tiempo real mediante el programa de Matlab para cualquier máquina eléctrica que se pueda requerir.

Existe la necesidad de poder visualizar gráficamente parámetros y controladores de un motor.

La implementación de hardware no suele ser la adecuada el cual requiere una simulación.

2. JUSTIFICACION DEL TRABAJO.

Mediante el desarrollo de la aplicación en Tiempo Real se pretende dar una visión práctica del diseño de controladores, permitiendo entender y visualizar gráficos del controlador que se haya diseñado.

Poder realizar una simulación antes de una implementación, mediante el uso de "Simulink" por Matlab.

3. OBJETIVOS.

3.1. OBJETIVO GENERAL.

Realizar un entorno gráfico que permita el control ajustable de la velocidad de una máquina de corriente continua en tiempo real.

3.2. OBJETIVOS ESPECIFICOS

- Aplicar el programa MatLab para el control de motores.
- Realizar la simulación correspondiente mediante el Simulink.
- Dar una visión práctica del diseño de controladores, permitiendo entender y visualizar gráficos del controlador diseñados.

4. FUNDAMENTACION TEORICA.

4.1. Introducción a MATLAB

El nombre de MATLAB proviene de la contracción de los términos **MAT**rix **LAB**oratory y fue concebido para el fácil acceso a las librerías que son de gran importancia en el campo de la computación y el cálculo matricial.

MATLAB es un entorno de computación y desarrollo de aplicaciones totalmente integrado, orientado para el desarrollo de proyectos con elevados cálculos matemáticos y la visualización gráfica de estos. MATLAB integra análisis numérico, cálculo matricial, procesado de señal, todo ello en un entorno fácil para el usuario.

Tanto en el mundo universitario como en el industrial, MATLAB se ha convertido en una herramienta básica para la resolución de complejos problemas matemáticos en diferentes áreas como la computación, el cálculo numérico, prototipaje algorítmico, teoría de control automático, estadística, etc.

MATLAB consta de diferentes aplicaciones o *toolboxes* especializados orientados a ingenieros, científicos y todo tipo de profesionales técnicos. Entre ellos destacan: Sistemas de Control, Adquisición de Datos, Tiempo Real, Lógica Fuzzy, Procesamiento de imágenes, Redes Neuronales, Optimización, Procesamiento de Señal, etc.

MATLAB presenta una aplicación para hacer simulaciones en tiempo real, la *toolbox* Real Time Windows Target. Esta herramienta permite realizar aplicaciones de control y simulaciones en tiempo real para plantas físicas, como puede ser el caso que nos ocupa: un motor de corriente continua.

4.2. *Real Time Windows Target*

Real Time Windows Target es una herramienta de MATLAB que permite capturar y generar señales en tiempo real mediante diagramas de bloques generados con Simulink. Además, se pueden visualizar estas señales, cambiando y controlando parámetros, todo en tiempo real. Para hacerlo posible tiene que haber un elemento físico que interactúe entre Simulink y el elemento exterior que queremos controlar, recoger señales,... este elemento es la placa de adquisición de datos DAQ, que es la que permite operar con señales de entrada y/o salidas analógicas y digitales.

Un componente clave del Real Time Windows Target es un *kernel* en tiempo real que hace de interfaz con el sistema operativo Windows para asegurar que la aplicación en tiempo real se está ejecutando en el tiempo de muestreo seleccionado. El *kernel* asigna la prioridad más elevada de ejecución para la aplicación en tiempo real, y lo hace utilizando el reloj interno del ordenador como fuente principal de tiempo.

4.2.1. Aplicación Real-Time

Características:

Código compilado: es el resultado de compilar el código fuente. En nuestro caso el código es el modelo.

Relación con el modelo de Simulink: el ejecutable contiene una relación binaria de todos los componentes del modelo, conexiones entre bloques, dependencias de tiempo y variables.

Relación con el *kernel*: el modelo tiene que ser cargado y ejecutado directamente por el *kernel* del Real Time Windows Target. Esta

nunca podrá ser ejecutada sin el *kernel*.

Checksum: el modelo y el ejecutable de Simulink contienen un valor de Checksum. El *kernel* utiliza este valor para comparar el modelo y el ejecutable, si estos son coherentes permitirá realizar la ejecución. Cuando se haga un cambio en el modelo de Simulink el valor de Checksum no varía hasta que no se hace un "rebuild". Esto permite que cuando se cambian valores de los parámetros durante la ejecución, el mapeo de parámetros a memoria se haga de manera correcta.

4.2.2. Análisis y adquisición de señales

Se puede adquirir, visualizar y salvaguardar señales utilizando el bloque *Scope* de Simulink, trabajando en modo externo. Quiere decir que se puede observar el comportamiento del modelo en tiempo real y guardar los datos en el Workspace.

Existen dos modalidades de captura y visualización de las señales:

Signal Tracing: proceso mediante el cual se puede adquirir y visualizar señales durante la ejecución de una aplicación en tiempo real. Es decir, nos permitirá visualizar los datos mientras que los está capturando, sin tener que esperar a que acabe la simulación (sólo se permite mediante el bloque *Scope*, no permite exportarlos a medida que los captura).

***Signal Logging*:** proceso mediante el cual se puede adquirir y visualizar señales procedentes de la aplicación en tiempo real, una vez haya acabado la ejecución o bien se haya parado manualmente. Hasta que no se cumpla alguna de las dos condiciones no se permite visualizar, guardar y/o exportar las señales (es un problema cuando se trabaja con

una interfaz gráfica externa a Simulink).

4.2.3. Ajuste de parámetros

Cambiar parámetros y observar los cambios que se producen durante la ejecución en tiempo real es posible desde Simulink o bien a través de una interfaz gráfica externa.

Modo *External* en Simulink: trabajar en este modo nos permite cambiar valores de parámetros y transferirlos automáticamente mientras se ejecuta la aplicación en tiempo real.

4.2.4. Entorno del software

El entorno de software es un sitio para diseñar, construir, y testear en tiempo no real y en tiempo real.

4.2.4.1. Simulación en tiempo no real

Se crea un modelo con Simulink y se pone en modo “normal” para llevar a cabo una simulación en tiempo no real.

Al realizar una simulación en tiempo no real, Simulink utiliza un vector de tiempo para gestionar el modelo. Después de que las salidas sean computadas por un determinado valor de tiempo, Simulink inmediatamente repite esta operación para el siguiente valor del vector de tiempo y así sucesivamente. Este proceso finalizará en el momento en que se para la simulación o bien cuando se llegue al tiempo final.

Dado que este vector de tiempo no está asociado a ningún reloj, las salidas son calculadas en tiempo no real, y lo hacen tan rápido como el ordenador le permita.

4.2.4.2. Ejecución en tiempo real

Para ejecutar una aplicación en tiempo real, se crea un modelo en Simulink y se pone en modo *External*.

Para hacer una aplicación en tiempo real intervienen Real-Time Workshop, Real Time Windows Target y el compilador C/C++. La respuesta del compilador es un ejecutable que el *kernel* puede ejecutar en tiempo real. Esta aplicación utiliza como parámetros iniciales los que hay en el modelo en el momento de la generación del código.

Durante la ejecución en tiempo real, utilizando el tiempo de muestreo fijado por el usuario, el Real Time Windows Target utiliza las interrupciones para gestionar la aplicación en tiempo real. En cada nueva interrupción el ejecutable computa los valores de salida de los bloques del modelo.

Para generar código con Real-Time Workshop se utiliza el algoritmo *Fixed-Step Discrete*. Este algoritmo calcula el valor del tiempo para la siguiente muestra añadiendo un incremento de tiempo al actual. La precisión de la señal a representar y la longitud del tiempo de simulación resultante dependen del tiempo de muestreo escogido. Es decir, cuanto menor sea el tiempo de muestreo escogido mayor será la calidad de la señal muestreada, pero menor será el tiempo de simulación máximo. En caso que el usuario no seleccione un tiempo de muestreo, Simulink escoge uno por defecto dividiendo por 50 la diferencia del tiempo final e inicial. También hay que remarcar que este algoritmo no permite simular estados continuos.

4.2.5. Simulink en modo externo

El modo externo requiere una interfaz de comunicación para pasar parámetros externos a Simulink, y al acabar la recepción, el mismo protocolo de

comunicación tiene que ser utilizado para aceptar nuevos valores e insertarlos en la correspondiente posición de memoria para que sean utilizados por la aplicación en tiempo real.

En algunas tarjetas de Real-Time Workshop la interfaz de comunicación utiliza protocolo TCP/IP. En el caso de Real Time Windows Target, el ordenador servidor también sirve como ordenador objetivo. Entonces, sólo se necesita un driver virtual para intercambiar parámetros entre MATLAB, el espacio de memoria de Simulink, y la memoria que es accesible para la aplicación en tiempo real.

Hay que recordar que trabajando en este modo se puede obtener y visualizar señales con el bloque *Scope*, así como ajustar parámetros. Cuando se modifica algún parámetro de un bloque de entrada, la simulación queda pausada hasta que no se cierra la ventana de diálogo.

4.2.6. Buffers y transferencia de datos

Para cada intervalo de muestreo de la aplicación en tiempo real, Simulink almacena de manera continua datos en memoria hasta que el *buffer* de datos se llena. Una vez el *buffer* está lleno, Simulink suspende la captura de datos hasta que éstos no hayan sido transferidos a MATLAB a través del modo externo de Simulink. Cuando esto sucede, la aplicación en tiempo real continúa ejecutándose.

Los datos capturados dentro del *buffer* son continuos. Cuando el *buffer* ha sido transferido a Simulink, se visualiza inmediatamente en el bloque *Scope*. Existe la posibilidad de poder guardar estos datos directamente en un fichero MAT.

4.3. Elementos físicos de la aplicación en Tiempo Real

El conjunto del *hardware* que forma la aplicación en Tiempo Real se puede desglosar en seis bloques de elementos que interactúan entre sí.

PC
Entradas/Salidas
Driver del motor de corriente continua
Motor de corriente continua
Alimentación externa
Realimentación

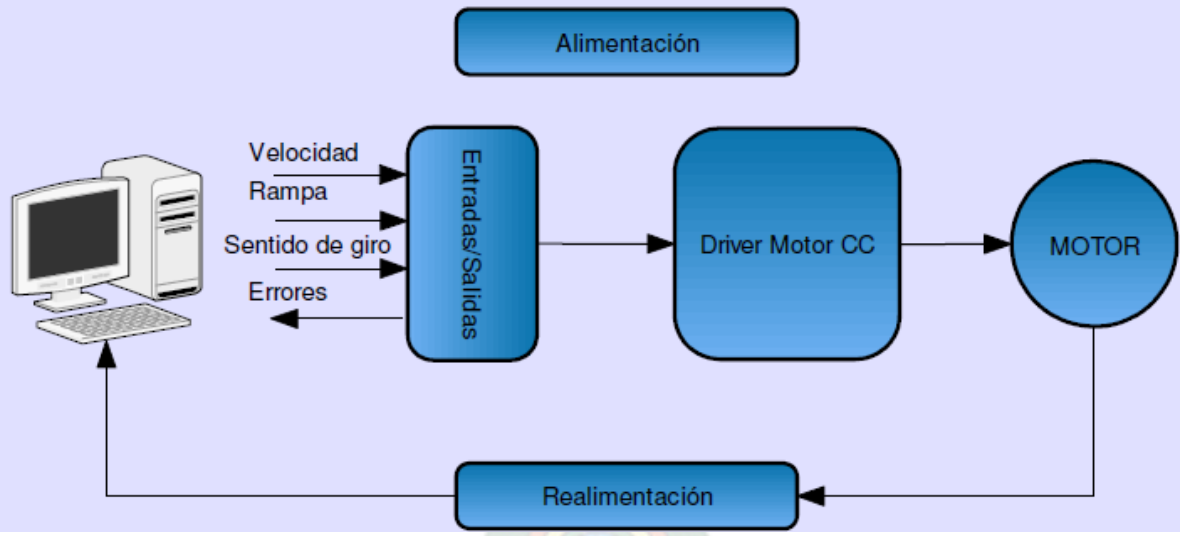


Ilustración 1: Representación en bloques de los componentes del controlador.
Fuente: Rafael Moragues, *Sistema de control en temps real*

4.3.1. PC

El ordenador personal es el encargado de capturar los datos que transmite el usuario al *driver* (velocidad, rampa, etc), y los datos obtenidos del motor (errores, datos de la realimentación, etc), para realizar los cálculos algorítmicos necesarios y así actualizar los valores del modelo en cada intervalo de tiempo. Es decir, la computadora y el modulo de entrada/salida, o también

placa de adquisición de datos, establecerán un intercambio de parámetros para cada instante de tiempo.

El PC aloja en su interior la tarjeta de adquisición de datos, los *drivers* de la misma, el programa MATLAB, la aplicación en tiempo real, y Simulink. Las características de la computadora utilizada y su sistema operativo son:

Procesador INTEL PENTIUM 4 a 1,80 GHz
512 MB de RAM
WINDOWS 2000 (Service Pack 4)
MATLAB R2006a v7.2.0.232
SIMULINK R2006a v6.4
VISUAL C/C++ v6.0

4.3.1.1 Placa de adquisición de datos

La tarjeta de adquisición de datos permite capturar y/o generar señales reales e interactuar con ellas desde la aplicación en tiempo real. Un sistema de adquisición de datos esta formado por un hardware y un software que permite a un sistema digital conectarse al mundo real. El sistema de adquisición de datos típico está formado por:

Sensores y actuadores
Hardware de adquisición de datos
Hardware de acondicionamiento de señal
Computadora o procesador
Programa

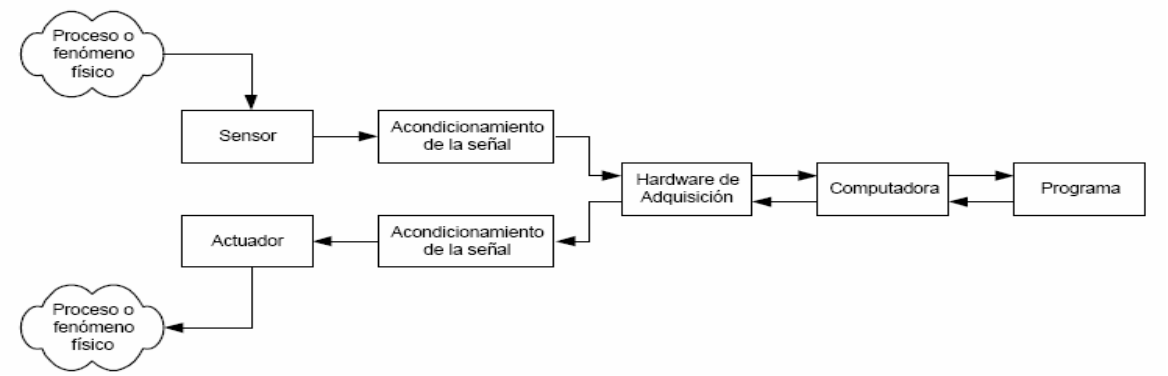


Ilustración 2: Esquema del proceso de adquisición de datos

Fuente: Rafael Moragues, *Sistema de control en temps real*

4.3.1.1.1 Sensores y actuadores

Los sensores y actuadores son aquellos que actúan como transductores, es decir estos elementos transforman una señal capturada de una naturaleza en otra señal de salida de otra naturaleza. En el caso que nos ocupa, el sensor seria el tacómetro que lee las vueltas que realiza el eje del motor y genera una señal eléctrica proporcional. El actuador es el que mediante una señal eléctrica hace que el motor gire a determinada velocidad.

4.3.1.1.2. Hardware de adquisición de datos

Es el corazón de cualquier sistema de adquisición de datos. Su función es convertir señales analógicas provenientes del mundo real a señales digitales, o bien convertir señales digitales a analógicas

4.3.1.1.3. Hardware de acondicionamiento de la señal

Normalmente las señales de los sensores son incompatibles con el hardware de adquisición de datos. Para lograr esta compatibilidad habrá que acondicionar la señal, es decir que si la señal es muy pequeña habrá que amplificarla, de lo

contrario atenuarla. También es muy común eliminar componentes frecuenciales de las señales.

4.3.1.1.4. La computadora

La computadora proporciona un procesador, un reloj, un bus para transferir datos y espacio de memoria o disco para almacenar datos.

4.3.1.1.5. Software

El software de adquisición de datos permite intercambiar información entre la computadora y el hardware. Por ejemplo, los programas típicos permiten configurar la tasa de muestreo de una tarjeta de adquisición y adquirir un número concreto de muestras.

4.3.2. Driver del Motor CC

El driver del motor de corriente continua está formado por toda aquella circuiteria electrónica que ha sido diseñada para gobernar la tensión de armadura del motor, entre otras funciones. Para poder realizar sus funciones consta de dos partes: la de control y la de potencia. En la de control tendremos el microcontrolador y la GAL (Gate Array Logic) que controlan al puente en H que se encarga de suministrar la potencia necesaria según mande el driver.

4.3.2.1. Microcontrolador

Para realizar la gestión de las consignas de velocidad y rampa se utilizará el PIC18F252. Este elemento dispone de seis entradas analógicas y dos salidas de pwm.

Sus funciones serán:

Generar una señal de salida PWM en función de la consigna de velocidad, que variará tan rápido como lo permita el valor de la rampa.

Mediante un bit externo se indicará al microcontrolador el sentido de giro del motor. En función del valor se activará una diagonal u otra del puente en H, lo que permite girar a derechas o a izquierdas. Es decir, se genera un bit para cada transistor del puente, siendo el valor cero como el corte y uno la saturación.

4.3.2.2. GAL(Gate Array Logic)

El dispositivo Lógico Programable GAL es el que confiere seguridad y fiabilidad a los elementos de potencia, protegiéndolos de la recirculación de sobrecorrientes, evita cruces en las ramas de los puentes y trasmite el PWM a la señal de activación de la parte alta de los transistores del puente.

El modelo de GAL es el Ispv22v10 y este tipo de dispositivos destacan por la gran rapidez - del orden de nanosegundos - , característica importante para la protección de sobrecorrientes.

4.3.2.3. Puente en H

La configuración del puente en H se basa en el funcionamiento de cuatro interruptores controlados, que gestionan en todo momento la tensión media y la recirculación de la corriente de armadura.

La excitación del puente en H se realiza mediante modulación por anchura de pulsos y existen dos estrategias:

- Mantener una diagonal bloqueada y actuar en los dos transistores opuestos - superior e inferior -. Para invertir el sentido de giro habrá que realizar la situación contraria.

- Mantener una diagonal bloqueada y actuar con el PWM en el transistor superior y dejando el inferior de la otra diagonal activado. Para invertir el sentido de giro se intercambian los papeles de las diagonales.

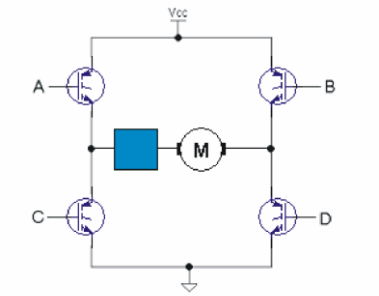


Ilustración 3: Estructura de puente en H típica mediante transistores IGBT
Fuente: Enciclopedia Wikipedia

4.3.2.4 Cuadrantes de trabajo de un motor

El motor de corriente continua puede trabajar en cuatro cuadrantes en función de la polaridad de la tensión y del sentido de la corriente de armadura. Modificando estos dos parámetros podremos variar velocidad o par en el eje del motor, respectivamente. Partiendo de las analogías par-intensidad y velocidad-tensión, se puede construir la siguiente tabla.

Cuadrante	Velocidad	Par
1º	+	+
2º	+	-
3º	-	-
4º	-	+

Tabla 1: Tabla cuadrantes Par-Velocidad

Fuente: Enciclopedia Wikipedia

Para entender el funcionamiento hay que pensar en un montacargas, que es el ejemplo más completo al respecto de los cuatro cuadrantes.

En el primer cuadrante velocidad y par son positivos, lo que se pretende es subir la carga a velocidad máxima.

El segundo cuadrante sería cuando se está llegando al destino y hay que frenar, entonces el par – o análogamente intensidad – se invierte para continuar subiendo pero frenando.
El tercero hace bajar la carga a velocidad máxima, es la situación inversa del primer cuadrante, velocidad y par negativos.

El cuarto cuadrante pasa al revés que en el segundo, se está llegando al destino y se disminuye la velocidad invirtiendo el sentido del par – el motor trabaja como generador. Este sistema se llama *freno o contramarcha*

Con una buena gestión del motor se pueden controlar todas las situaciones que pueda demandar una carga. Para ello es necesaria una topología en H.

4.4. Motor de corriente continúa

El **motor de corriente continua** es una máquina que convierte la energía eléctrica en mecánica, provocando un movimiento rotatorio. En la actualidad existen nuevas aplicaciones con motores eléctricos que no producen movimiento rotatorio, sino que con algunas modificaciones, ejercen tracción sobre un riel. Estos motores se conocen como motores lineales.

Las expresiones que definen el funcionamiento de un motor de corriente continua son relativamente sencillas. Si se considera que el flujo de excitación o también flujo de campo es constante se pueden tomar las siguientes consideraciones:

Par proporcional a la intensidad de inducido o de armadura.
Velocidad proporcional a la f.e.m interna.

Esto nos simplifica mucho el control ya que modificando la tensión de armadura podremos regular la velocidad. Obtener una tensión continua variable es relativamente sencillo, además si se alimenta el motor cc con una etapa de

potencia adecuada y se escoge una técnica de control adecuada, se consigue una muy buena respuesta dinámica (fiable y rápida)

Existen diversos tipos de motores cc: con escobillas, sin escobillas (brushless) y los motores pasos a paso. Nos centraremos en el de escobillas porqué es el caso que nos ocupa.

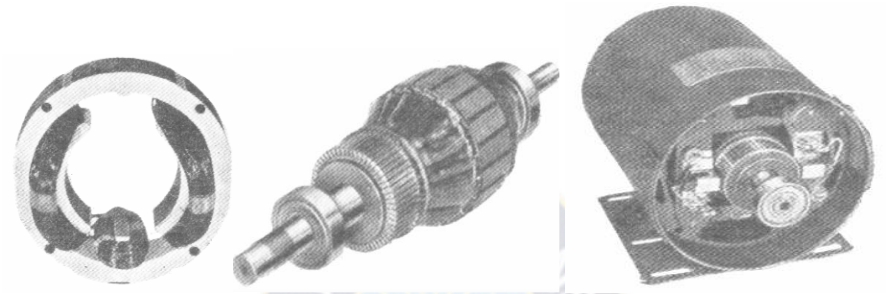


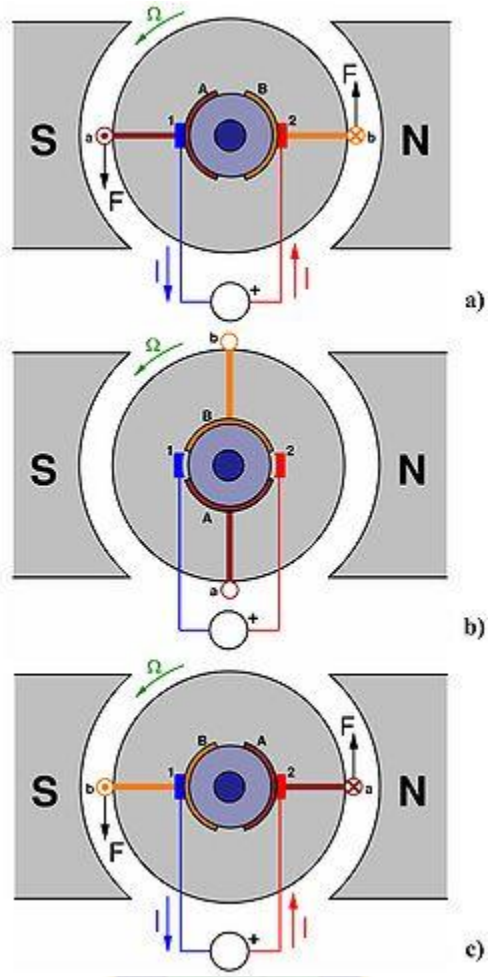
Ilustración 4: Estator, rotor y motor cc montado (se señala una escobilla), sucesivamente.
Fuente: Enciclopedia Wikipedia

La función del colector de delgas y las escobillas es alimentar correctamente las bobinas del devanado de armadura del rotor. El colector de delgas está unido al rotor y las escobillas al estator.

En ciertas potencias existen diferentes configuraciones para las conexiones entre el devanado de excitación y armadura: excitación independiente, paralelo o shunt, serie y la compuesta. La que se va utilizar es la independiente por tanto nos centraremos en solo en esta configuración.

- La principal característica del motor de corriente continua es la posibilidad de regular la velocidad desde vacío a plena carga.
- Su principal inconveniente, el mantenimiento, muy caro y laborioso.

4.4.1. Principio de funcionamiento



Ilustracion 5: Esquema del funcionamiento de un motor de c.c.
Fuente: Jimmy Wales- Wikipedia

Esquema del funcionamiento de un motor de c.c. elemental de dos polos con una sola bobina y dos delgas en el rotor. Se muestra el motor en tres posiciones del rotor desfasadas 90º entre sí.

1,2:Escobillas;
A,B:**Delgas**;
a, b: Lados de la bobina conectados respectivamente a las delgas A y B.

Según la Ley de Lorentz, cuando un conductor por el que pasa una corriente eléctrica se sumerge en un campo magnético, el conductor sufre una fuerza

perpendicular al plano formado por el campo magnético y la corriente, siguiendo la regla de la mano derecha, con módulo

$$F = B \cdot l \cdot I$$

- **F:** Fuerza en newtons
- **I:** Intensidad que recorre el conductor en amperios
- **l:** Longitud del conductor en metros
- **B:** Densidad de campo magnético o densidad de flujo teslas

El rotor tiene varios repartidos por la periferia. A medida que gira, la corriente se activa en el conductor apropiado.

Normalmente se aplica una corriente con sentido contrario en el extremo opuesto del rotor, para compensar la fuerza neta y aumentar el momento.

4.4.2. Fuerza contra electromotriz inducida en un motor

Es la tensión que se crea en los conductores de un motor como consecuencia del corte de las líneas de fuerza, es el efecto generador de pines.

La polaridad de la tensión en los generadores es inversa a la aplicada en bornes del motor.

Las fuertes puntas de corriente de un motor en el arranque son debidas a que con la máquina parada no hay fuerza contraelectromotriz y el bobinado se comporta como una resistencia pura del circuito.

4.4.3. Número de escobillas

Las escobillas deben poner en cortocircuito todas las bobinas situadas en la zona neutra. Si la máquina tiene dos polos, tenemos también dos zonas neutras. En consecuencia, el número total de escobillas ha de ser igual al número de polos de la máquina.

En cuanto a su posición, será coincidente con las líneas neutras de los polos.

4.4.4. Sentido de giro

El sentido de giro de un motor de corriente continua depende del sentido relativo de las corrientes circulantes por los devanados inductor e inducido.

La inversión del sentido de giro del motor de corriente continua se consigue invirtiendo el sentido del campo magnético o de la corriente del inducido.

Si se permuta la polaridad en ambos bobinados, el eje del motor gira en el mismo sentido.

Los cambios de polaridad de los bobinados, tanto en el inductor como en el inducido se realizarán en la caja de bornes de la máquina, y además el ciclo combinado producido por el rotor produce la fmm (fuerza magnetomotriz).

El sentido de giro lo podemos determinar con la regla de la mano derecha, la cual nos va a mostrar el sentido de la fuerza. La regla de la mano derecha es de la siguiente manera: el pulgar nos muestra hacia donde va la corriente, el dedo índice apunta en la dirección en la cual se dirige el flujo del campo magnético, y el dedo medio hacia donde va dirigida la fuerza resultante y por lo tanto el sentido de giro.

Los motores y los generadores de corriente continua están constituidos esencialmente por los mismos elementos, diferenciándose únicamente en la forma de utilización.

Por reversibilidad entre el motor y el generador se entiende que si se hace girar al rotor, se produce en el devanado inducido una fuerza electromotriz capaz de transformarse en energía en el circuito de carga.

En cambio, si se aplica una tensión continua al devanado inducido del generador a través del colector de delgas, el comportamiento de la máquina ahora es de motor, capaz de transformar la fuerza contraelectromotriz en energía mecánica.

En ambos casos el inducido está sometido a la acción del campo inductor principal.

4.4.5. Tipos de motores D.C

Los motores D.C se clasifican de acuerdo al tipo de bobinado del campo como motores Serie, Shunt, Shunt estabilizado, o Compuesto (Compound). Sin embargo algunos de ellos pueden ser auto excitados o de excitación separada o pueden tener campos de imán permanente.

Ellos muestran curvas muy diferentes de torque-velocidad y se conectan en diferentes configuraciones para diferentes aplicaciones.

Algunos motores D.C utilizan imán permanente como campo principal, especialmente los de potencia (HP) fraccionada (1/4,1/2,3/4) y baja potencia.

Los motores de imán permanente tienen la ventaja de no requerir una fuente de potencia para el campo, pero tienen la desventaja de ser susceptibles a la desmagnetización por cargas de choque eléctricas o mecánicas. Los campos de imán permanente no se pueden ajustar para entonar el motor para ajustarse a la aplicación, como pueden los de campo bobinado.

4.4.5.1. MOTOR SHUNT

En un motor shunt, el flujo es constante si la fuente de poder del campo es fija. Asuma que el voltaje de armadura Et es constante. A medida que la corriente de la carga disminuye desde plena carga a sin carga, la velocidad debe aumentar proporcionalmente de manera que la fuerza contra electromotriz Ec aumentará para mantener la ecuación en balance. A voltaje nominal y campo completo, la velocidad del motor shunt aumentará 5% a medida que la corriente de carga

disminuya de plena carga a sin carga. La reacción de armadura evita que el flujo de campo permanezca absolutamente constante con los cambios en la corriente de la carga. La reacción de armadura, por lo tanto causa un ligero debilitamiento del flujo a medida que la corriente aumenta. Esto tiende a aumentar la velocidad del motor. Esto se llama “inestabilidad” y el motor se dice que está inestable.

4.4.5.2. MOTOR SERIE

En un motor serie, el flujo del campo es una función de la corriente de la carga y de la curva de saturación del motor. A medida que la corriente de la carga disminuye desde plena carga, el flujo disminuye y la velocidad aumenta. La rata de incremento de velocidad es pequeña al principio pero aumenta a medida que la corriente se reduce. Para cada motor serie, hay una mínima carga segura determinada por la máxima velocidad de operación segura.

4.4.5.3. MOTOR COMPUESTO (COMPOUND)

Los motores compuestos tienen un campo serie sobre el tope del bobinado del campo shunt como se ve en la figura. Este campo serie, el cual consiste de pocas vueltas de un alambre grueso, es conectado en serie con la armadura y lleva la corriente de armadura.

El flujo del campo serie varia directamente a medida que la corriente de armadura varia, y es directamente proporcional a la carga. El campo serie se conecta de manera tal que su flujo se añade al flujo del campo principal shunt. Los motores compound se conectan normalmente de esta manera y se denominan como compound acumulativo.

Esto provee una característica de velocidad la cual no es tan “dura” o plana como la del motor shunt, no tan “suave” como un motor serie. Un motor compound tiene un limitado rango de debilitamiento de campo, la debilitación del campo puede resultar en exceder la máxima velocidad segura del motor sin carga. Los motores

D.C compound son algunas veces utilizados donde se requiera una respuesta estable de torque constante a través de un amplio rango de velocidad.

4.4.5.4. MOTOR SHUNT ESTABILIZADO

Para vencer la potencial inestabilidad de un motor recto shunt y reducir la “caída” de velocidad de un motor compound, un ligero devanado serie es arrollado sobre el devanado shunt. El flujo del devanado serie aumenta con la corriente de carga y produce un motor estable con una característica de caída de velocidad para todas las cargas.

El devanado serie es llamado un campo estabilizador o “stab” y el motor un motor shunt estabilizado. La regulación de velocidad de un motor shunt estabilizado es típicamente menor al 15%.

La mayoría de los motores Reliance Super RPM y RPM III son shunt estabilizados. Cuando el campo shunt del motor es debilitado para aumentar la velocidad a un nivel de operación mas alto, el flujo del devanado serie llega a ser un porcentaje mayor del flujo total, de manera que a medida que la corriente aumenta, la caída de velocidad es un porcentaje mayor que antes.

En aplicaciones donde la instabilidad resultante pudiera afectar seriamente el funcionamiento de la maquina (movida por el motor), el campo serie puede desconectarse. En aplicaciones donde los efectos de estabilidad nos son críticos, como en un frenado regenerativo, el campo serie puede utilizarse para mejorar el rendimiento que el provee.

Cuando el campo serie no se conecta, el fabricante del control debe asegurar que la máxima velocidad segura del motor no es excedida y debe reconocer la perdida de torque que resulta de la operación del motor shunt estabilizado sin el devanado serie.

4.5. Velocidad, Torque y Potencia (HP)

Las características velocidad - torque dan al motor D-C una versátil aplicación. El torque de régimen de un motor D.C es dado a una velocidad específica llamada **Velocidad Base**.

La velocidad base se define como las RPM de un motor D.C cuando opera a:

1.- Corriente de campo de régimen

2.- Voltaje de armadura de régimen

3.- Carga de régimen (Corriente de Armadura)

La velocidad base (RPM) se muestra en la placa del motor. Típicas velocidades base para motores D.C son: 850, 1150, 1750 y 2500 RPM. A velocidad base, un motor D.C entrega la velocidad, torque y HP de régimen (nominales). La tolerancia de la NEMA para la velocidad base es de 7½ % .

La combinación de velocidad y torque desarrolla los HP de régimen de acuerdo con la siguiente relación:

H.P = (1)

5250

Donde:

Torque (T): libra-pie

Velocidad (N): r.p.m

Esta fórmula establece los HP del motor a un torque y velocidad específicos. Los motores se acoplan a reductores, correas y poleas, y otros dispositivos modificadores de velocidad, para producir torque y/o velocidades mayores que las

de placa, pero esta combinación nunca debe exceder el valor de los HP de placa. Dicho de otra manera, pueden obtenerse torques mayores, pero solo a proporcionalmente velocidades menores, o se disponen de velocidades mayores (hasta la máxima velocidad de placa con debilitamiento del campo) si proporcionalmente se acepta un menor torque.

4.5.1. Ecuación general del motor

Con la excepción de los controladores que también regulan la corriente de campo, el voltaje de armadura Et es el único parámetro que el controlador puede directamente cambiar o regular. Los sistemas de control pueden clasificarse como reguladores de voltaje, velocidad, corriente (torque), tensión o posición. Todos estos sistemas utilizan un dispositivo de realimentación apropiado para permitir al controlador regular la función deseada.

La ecuación general del motor define el funcionamiento del motor bajo diferentes condiciones de voltaje y carga

$$Et = Ec + Ia.Ra \quad (2)$$

Donde:

Et: Voltaje en los terminales de la armadura

Ec: Fuerza contra electromotriz

Ia: corriente de la armadura

Ra: resistencia de la armadura

$$Ec = K..N \quad (3)$$

Siendo:

: el flujo magnético

N: velocidad (rpm)

El voltaje Ec se opone al voltaje aplicado a la armadura Et, y por esta razón es llamado fuerza contra electromotriz (FCM). Este voltaje es el resultado del corte del campo magnético al girar los conductores de la armadura, produciendo así un voltaje generado. Bajo condiciones normales de operación, este término (Ec) es mucho mayor que el termino Ia.Ra. La velocidad del motor es proporcional al voltaje aplicado en los terminales, el cual es el voltaje que se muestra en la placa del motor.

Típicamente, la resistencia de la armadura (Ra), está en el orden de 1 Ohm, o menos, y la corriente de armadura (Ia), es función de la carga mecánica del motor. Por ejemplo, considere un motor de 20 HP:

Voltaje en los terminales: Et = 240 V

Corriente de armadura: Ia = 71 Amp. A plena carga

Resistencia de armadura:Ra = 0.15 Ohm

Caída de voltaje en la resistencia de armadura:Ia.Ra = 10.65 V

a plena carga.

De la ecuación (2) se deduce:

$$Ec = Et - Ia.Ra$$

$$Ec = 240 - 10.65 = 229.35 \text{ VDC}$$

Los motores D.C generan torque a través de la interacción de los campos magnéticos. El campo magnético principal es desarrollado por los polos del motor. El campo magnético que interactúa con el campo magnético principal es producido por la armadura y su amplitud está determinada por la corriente de armadura.

El torque desarrollado en un motor D.C es función del radio de la armadura, el número de conductores y la fuerza ejercida sobre los conductores. La fuerza depende del flujo, la corriente y la longitud de los conductores de la armadura. Expresado matemáticamente como:

$$T = K..Ia \quad (4)$$

Donde el torque (T) es medido en libra-pié, K es una constante determinada por el número de polos en la máquina y el número de arreglos de los conductores en la armadura, es el flujo total por polo en el entrehierro, y Ia es la corriente de armadura en amperios.

Por lo tanto, el torque de un motor shunt con excitación constante varia directamente con la corriente de armadura.

Para un motor serie, el flujo varia con la corriente de la carga excepto por efectos de saturación así que el torque varia aproximadamente con el cuadrado de la corriente de la carga..

La característica de torque de un motor compound está entre el shunt y el serie.

Para observar el efecto del voltaje y la corriente sobre la velocidad y el torque, la tabla # 1 será útil. Se utilizará como ejemplo un motor de 5 HP con los siguientes datos de placa:

Et = 180 VDC

Ia = 24 Amp, a plena carga

Ra = 0.591Ohm

Velocidad Base: 1750 rpm

Asumiendo una carga del 75% de la nominal = 18 Amp.

Calculemos la relación Volts/RPM de la Ec.

Et = Ec + Ia.Ra Ec = Et - Ia.Ra = 180 VDC - (24 x 0.591)

Ec = 180 - 14.184 = 165.81 VDC

Volts./ RPM = 165.816/1750 = 0.094752 VDC/RPM

4.6. Módulo de Control

El módulo de control se tiene que utilizar conjuntamente con la unidad de motor y carga. El modulo es un dispositivo controlador que tiene integrado un variador de frecuencia para suministrar la energía eléctrica y poder realizar el control de máquinas trifásicas, así como mostrar los valores medidos de velocidad y par en la pantalla de 7 segmentos. Existe la posibilidad conectarse mediante una comunicación serie a través de un PC y registrar valores.

Las características técnicas:

Alimentación 230V, a frecuencia de 47 a 62 Hz, 2 kW
Control automático de velocidad (5000 a -5000 rpm)
Control automático de par (±9 N·m)
Los valores se pueden modificar mediante un potenciómetro externo
Simulación de diferentes cargas: $k \cdot n^2$, k/n, características arbitrarias Mi ~ ni, etc
Permite realizar un control externo inyectando una valor de continua de -10 V a 10 V
Visualización de cuadrantes de trabajo leds de siete segmentos de 25 mm, 4 dígitos para la velocidad y 3 para el par
Monitorización de la temperatura: Device Under Test (DUT) de la unidad de motor y carga
Medida directa de voltaje para el DUT
Comunicación serie para (RS232) para la transmisión de valores medidos y el

control remoto.

Comunicación serie para la conexión de equipos adicionales

Existen 6 tipos de error que puede dar el dispositivo:

Tipo de Error	Descripción
ERR 1	Generator operation
ERR 2	Temperature of the DUT
ERR 3	Overvoltage in the link circuit of the control unit
ERR 4	Temperature of the D & B Unit
ERR 5	Temperature of the control unit
ERR 6	DUT is switched off via the output DUT ENABLE by removing the coupling guard or the shaft end guard

4.7. Elementos de Software

4.7.1. Herramienta GUIDE

Para el desarrollo de la aplicación en tiempo real, se trabaja con una herramienta de MATLAB llamada *GUIDE* (Graphical Use Interface Development Environment). Esta herramienta esta pensada para desarrollar GUIs (Graphical User Interfaces) fácil y rápidamente haciendo sencillo el diseño y presentación de los controles de la interfaz, reduciendo la labor en el momento de seleccionar, deshacer, arrastrar y centrar controles, así como la personalización de las propiedades de estos.

El proceso a seguir para el desarrollo de un programa mediante GUIDE es que una vez se tienen todos los controles en posición, se editan las funciones de llamada (Callback) de cada uno de ellos, escribiendo el código de MATLAB que se ejecutará cuando el control sea utilizado. GUIDE está diseñado para hacer menos tedioso el proceso de desarrollo de la interfaz gráfica, para ello cuenta con un editor de propiedades (property editor) con el que se podrá modificar en cualquier momento los nombres, valores por defecto y las propiedades de los

elementos.

Para poder acceder a la herramienta GUIDE se puede de tres maneras:

Tecleando el comando >>*guide* desde el prompt de Matlab

A través del *Launch Pad*

Pinchando en File -> New ->GUI dentro de Matlab, como se muestra a continuación

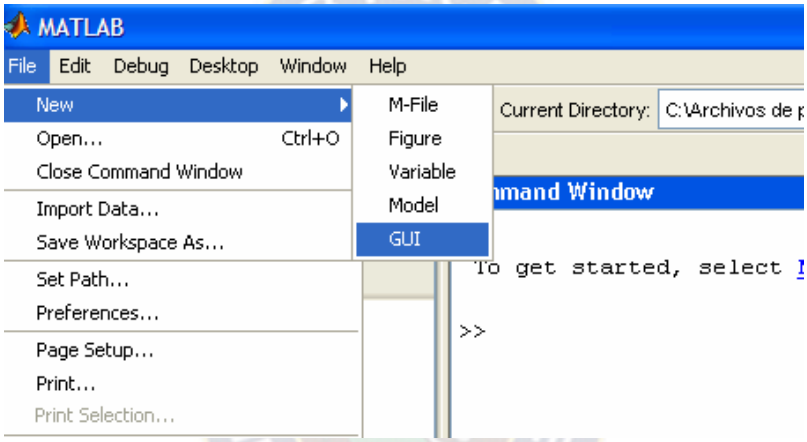


Ilustración 7: Inicio de la aplicación GUIDE mediante menú
Fuente: The MathWorks, *Real-Time Windows Target User's Guide.*

Una vez iniciemos la aplicación aparece la siguiente interfaz, y nos pedirá si queremos realizar una aplicación nueva o bien abrir una existente. En el caso que se cree una nueva aplicación se puede partir de aplicaciones prefabricadas, como por ejemplo: con controles, con una pantalla para gráficos y menú, o un cuadro de dialogo.

Cuando crea una GUI se generan dos ficheros:

Un archivo .FIG, que es el que contiene los elementos gráficos así como las propiedades de la interfaz.

Un archivo .M que es el contiene el código con las correspondencias de los botones de control de la interfaz. Cada vez que se introduzca un elemento gráfico en el .FIG se generará unas líneas de programa

automáticamente asociadas a ese tipo de control. Estas líneas de programas son vacías, es decir, es como un contenedor que necesita que ser llenado para llevar a cabo alguna acción durante la ejecución del programa. Este modo de trabajo es como el de LABWINDOWS, VISUAL BASIC, etc

5. MEMORIA DE CÁLCULO

5.1. Modelado de la planta

Para diseñar un controlador sobre un sistema real se podría realizar de diferentes formas: empiríricamente o analíticamente. Los métodos empíricos se pueden englobar en 3 grupos: el analítico, el de lazo abierto y el de lazo cerrado. El analítico se basa en calcular y simular las diferentes acciones, solas y conjuntamente e ir viendo el comportamiento de la respuesta hasta que sea satisfactoria. El de lazo abierto y el lazo cerrado son métodos en que la sintonía de los parámetros del controlador se realiza aplicando las variables obtenidas empíricamente sobre las formulas de unas tablas.

El método escogido es el analítico porque se ha pensado que era el más interesante, ya que permite utilizar los conocimientos de teoría de control y es más atractivo para el alumnado. Para la consecución de este método es necesario determinar la planta de todo el sistema que controla el motor de corriente continua. Para ello se hacen pruebas en lazo abierto del sistema según la Ilustración 11. Se utiliza la carga y el modulo de *Leybold* para obtener la lectura de velocidad.

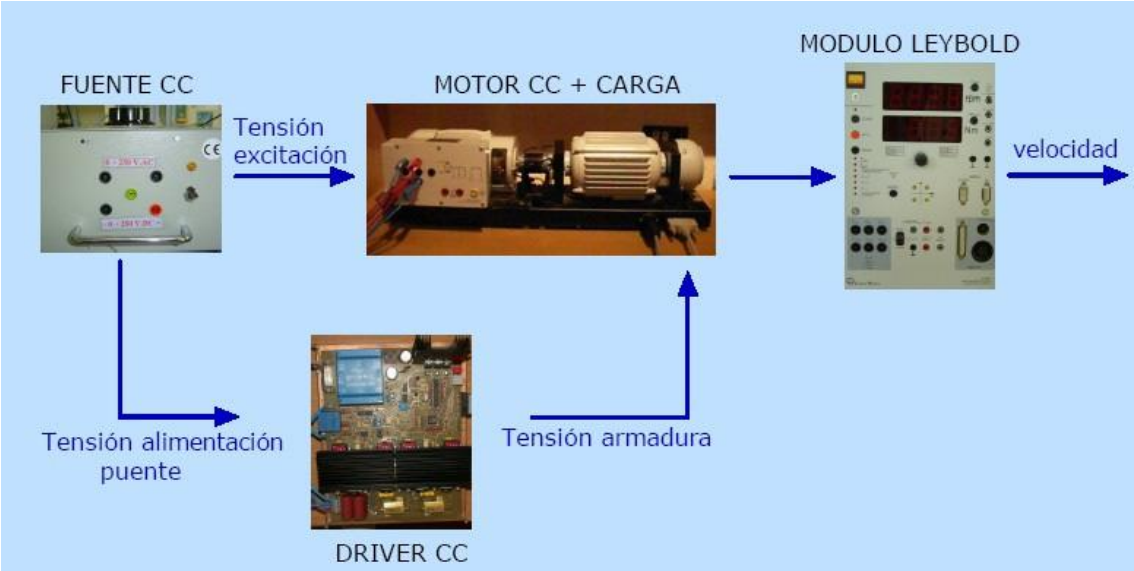


Ilustración 8: Diagrama de bloques del sistema

***Fuente**: Sistemas de control en tiempo discreto*, Katsuhiko Ogata, Ed. Pentice Hall, 2 Ed.

Para capturar los parámetros que definen la respuesta del sistema se utiliza un modelo en Simulink que permita introducir una consigna y visualizar los valores de la salida, aunque también se utilizado el osciloscopio para asegurar que la respuesta sea correcta. Los valores de salida permitirán reconstruir y estudiar los valores que definen la respuesta del motor de corriente contínua en lazo abierto. La frecuencia de muestreo escogida es superior a 10 veces la frecuencia del sistema, para evitar el fenómeno de *aliasing*, y así obtener una buena representación.

5.1.1. Aliasing

En estadística, procesamiento de señales, computación gráfica y disciplinas relacionadas, el **aliasing** es el efecto que causa que señales continuas distintas se tornen indistinguibles cuando se muestrean digitalmente. Cuando esto sucede, la señal original no puede ser reconstruida de forma unívoca a partir de la señal digital. Una imagen limitada en banda y muestreada por debajo de su frecuencia de Nyquist en las direcciones "x" e "y", resulta en una superposición de las

replicaciones periódicas del espectro G(fx, fy). Este fenómeno de superposición periódica sucesiva es lo que se conoce como aliasing o Efecto Nyquist.

El aliasing es un motivo de preocupación mayor en lo que concierne a la el muestreo incorrecto de señales analógicas puede provocar que señales de alta frecuencia presenten dicho aliasing con respecto a señales de baja frecuencia. El aliasing es también una preocupación en el área de la computación gráfica e infografía, donde puede dar origen a patrones de moiré (en las imágenes con muchos detalles finos) y también a bordes dentados. El aliasing nos puede traer problemaconversión analógica-digital de señales de audio y vídeo:s sobre todo en el campo de visión por computadores, ya que al procesar imágenes, si no es correcta la imagen obtenida con la realidad, podemos tener problemas con el hardware.

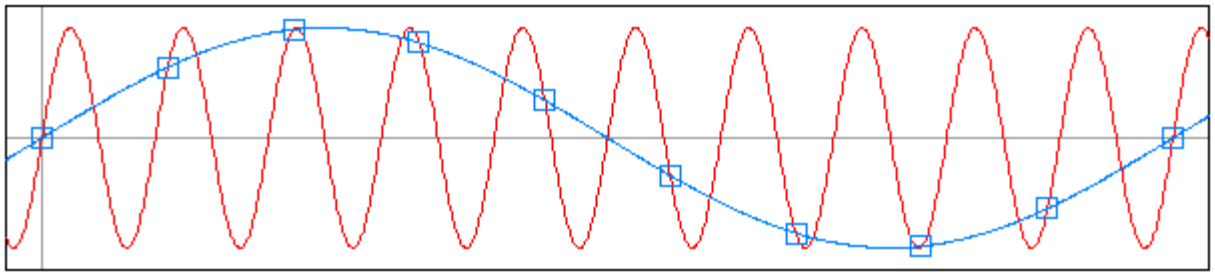
5.1.1.1. Aliasing en fenómenos periódicos

El Sol tiene un movimiento aparente de este a oeste en la bóveda celeste, con 24 horas entre cada amanecer. Si tomásemos una fotografía del cielo cada 23 horas, el sol parecería moverse de oeste a este, con 24·23=552 horas entre cada amanecer. El mismo fenómeno causa que las aspas de un ventilador parezcan a veces girar en el sentido inverso del que en realidad lo hacen, cuando se les filma o cuando son iluminadas por una fuente de luz parpadeante, tal como una lámpara estroboscópica, un tubo de rayos catódicos o una lámpara fluorescente, o simplemente, cuando el ventilador es iluminado por la parpadeante luz de la televisión.

5.1.2. Muestreo de una señal sinusoidal

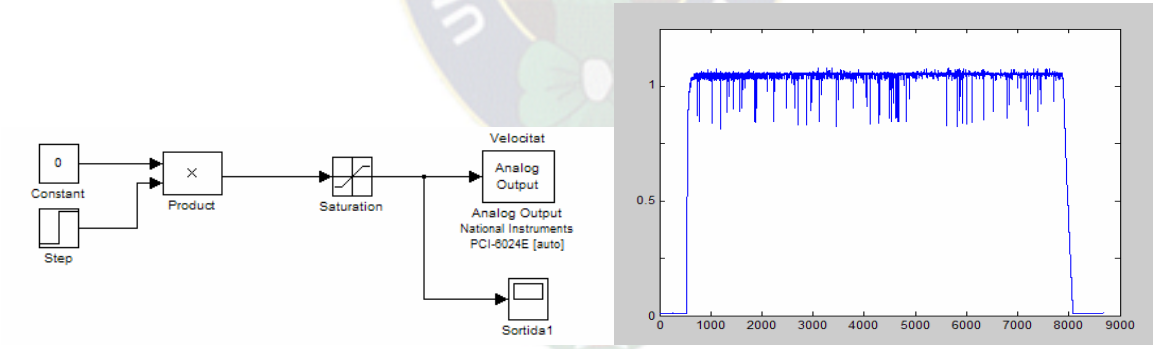
Cuando se obtienen muestras periódicas de una señal sinusoidal, puede ocurrir que se obtengan las mismas muestras que se obtendrían de una señal sinusoidal igualmente pero con frecuencia más baja. Específicamente, si una sinusoide de frecuencia f Hz es muestreada s veces por segundo, y $s \leq 2 \cdot f$, entonces las muestras resultantes también serán compatibles con una sinusoide de frecuencia

fm - f, donde fm es la frecuencia de muestreo. En la jerga inglesa de procesamiento de señales, cada una de las sinusoides se convierte en un "alias" para la otra.

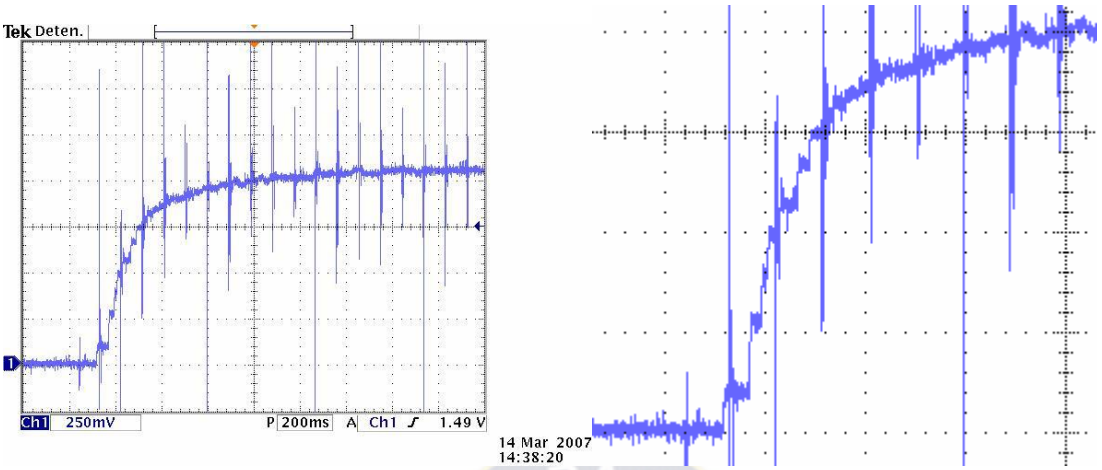


Por tanto, si se muestrea a la frecuencia *s* una señal analógica que contiene las dos frecuencias, la señal no podrá ser reconstruida con exactitud.

•Velocidad de muestreo recomendada: –2xfrecuencia mayor (medida de frecuencia) –10xfrecuencia mayor (detalle de la forma de onda)



a) b)



(c) (d)

Ilustración 9: Modelo creado para la obtención de la planta y la respuesta obtenida para el escalón unitario (figura a y b) con un tiempo de muestreo de 0,001s. Y la obtención de la planta mediante el osciloscopio (figura c y d).

***Fuente**: Sistemas de control en tiempo discreto*, Katsuhiko Ogata, Ed. Pentice Hall, 2 Ed.

La respuesta obtenida se aproxima a una función de primer orden, cuya expresión y respuesta característica quedan definidas en la ecuación (10) y la Ilustración 10.

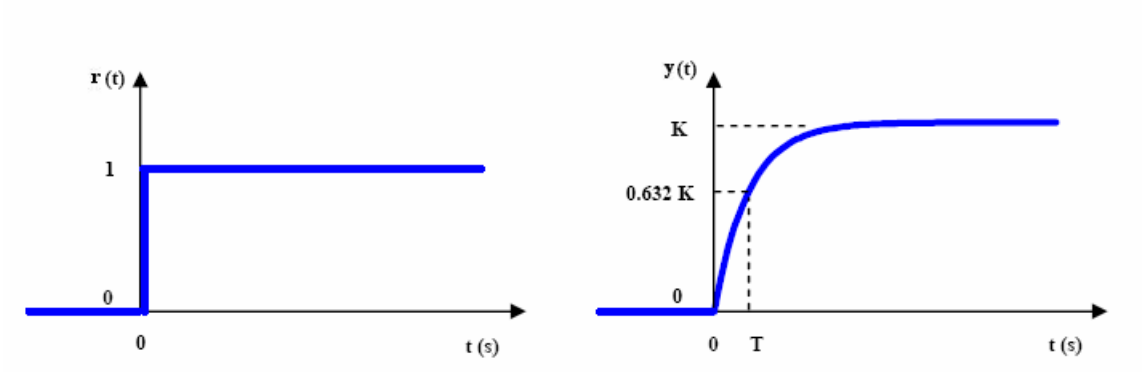


Ilustración 10: Respuesta de un sistema de primer orden a una entrada de tipo escalón unitaria.

***Fuente**: Sistemas de control en tiempo discreto*, Katsuhiko Ogata, Ed. Pentice Hall, 2 Ed.

A partir de la gráfica de la Ilustración 10 se obtiene la constante de tiempo que define el sistema de primer orden de la planta del motor de cc. La constante τ corresponde al 63,2 % del valor final de l repuesta y(t)

5.1.3. Funciones de transferencia discretas.

Dado que la aplicación de MATLAB/SIMULINK en tiempo real trabaja mejor para frecuencias de muestreo menores. Entonces, se puede afirmar que cuanto menor sea la frecuencia de muestreo:

- Mayor es el intervalo de tiempo que se podrá visualizar/ejecutar en la aplicación en tiempo real.
- Mayor es el número de bloques que se podrán insertar en Simulink sin errores de *buffer*.

Tiempo de muestreo (s)	**Función de Transferencia Discreta (Z)**
0,01	$G(z) = \frac{0{,}06196}{z - 0{,}944}$ **(22)**
0,001	$G(z) = \frac{0{,}00636}{z - 0{,}994}$ **(23)**

Tabla 3: Funciones de transferencia discreta en función del tiempo de muestreo
Fuente: Rafael Moragues, *Sistema de control en temps real a través de MATLAB/SIMULINK*, PFC

5.2. Diseño de controladores

Un sistema de control realimentado o también conocido como sistema de control en lazo cerrado es aquel sistema que mantiene una relación entre la salida y la entrada de referencia, comparándolas y usando la diferencia como medio de control.

La ventaja de un sistema de control en lazo cerrado es que el uso de la realimentación vuelve la respuesta del sistema relativamente insensible a las perturbaciones externas y a las variaciones internas en los parámetros del sistema. Aunque, en sistemas sin perturbaciones y donde se conozcan con anticipación las entradas es aconsejable utilizar un control en lazo abierto.

Algunos aspectos que habrá que tener en cuenta en el diseño del controlador es la frecuencia de muestreo a seleccionar en la aplicación en tiempo real, porque afectará a la planta del sistema, modificando la ganancia y la dinámica del sistema. Por lo tanto, el diseño de los controladores se hará para los tiempos de muestreo reflejados en la Tabla 3, mediante el método de diseño del lugar de las raíces. A continuación se muestra la estructura básica de un controlador PID discreto.

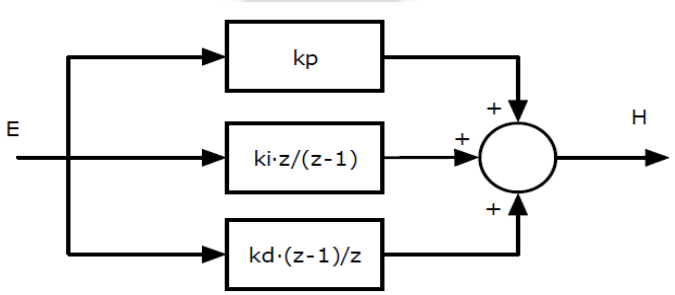


Ilustración 11: Estructura de controlador discreto con acciones proporcional, integral y derivativa
Fuente: Rafael Moragues, *Sistema de control en temps real a través de MATLAB/SIMULINK*, PFC

Desarrollando analíticamente la Ilustración 11, se obtiene la función de transferencia de un controlador con todas las **acciones (proporcional, integral y derivativa)**.

$$\frac{H(z)}{E(z)} = \frac{(kp + ki + kd)\cdot z^2 - (kp + 2\cdot kd)\cdot z + kd}{z^2 - z}$$

Acción proporcional e integral

$$\frac{H(z)}{E(z)} = \frac{(kp + ki)\cdot z - kp}{z - 1}$$

Acción proporcional

$$\frac{H(z)}{E(z)} = kp$$

5.2.1 Proporcional integral derivativo

Un PID (Proporcional Integral Derivativo) es un mecanismo de control por realimentación que calcula la desviación o error entre un valor medido y el valor que se quiere obtener, para aplicar una acción correctora que ajuste el proceso. El algoritmo de cálculo del control PID se da en tres parámetros distintos: el proporcional, el integral, y el derivativo. El valor Proporcional determina la reacción del error actual. El Integral genera una corrección proporcional a la integral del error, esto nos asegura que aplicando un esfuerzo de control suficiente, el error de seguimiento se reduce a cero. El Derivativo determina la reacción del tiempo en el que el error se produce. La suma de estas tres acciones es usada para ajustar al proceso vía un elemento de control como la posición de una válvula de control o la energía suministrada a un calentador, por ejemplo. Ajustando estas tres variables en el algoritmo de control del PID, el controlador puede proveer un control diseñado para lo que requiera el proceso a realizar. La respuesta del controlador puede ser descrita en términos de respuesta del control ante un error, el grado el cual el controlador llega al "set point", y el grado de oscilación del sistema. Nótese que el uso del PID para control no garantiza control óptimo del sistema o la estabilidad del mismo. Algunas aplicaciones pueden solo requerir de uno o dos modos de los que provee este sistema de control. Un controlador PID puede ser llamado también PI, PD, P o I en la ausencia de las acciones de control respectivas. Los controladores PI son particularmente comunes, ya que la acción derivativa es muy sensible al ruido, y la ausencia del proceso integral puede evitar que se alcance al valor deseado debido a la acción de control.

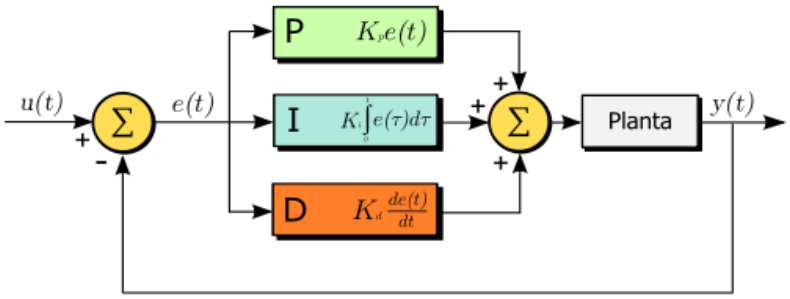


Ilustración 12: Diagrama en bloques de un control PID.
Fuente: Jimmy Wales- Wikipedia.

5.2.1.1. Funcionamiento

Para el correcto funcionamiento de un controlador PID que regule un proceso o sistema se necesita, al menos:

1. Un sensor, que determine el estado del sistema (termómetro, caudalímetro,manómetro, etc).
2. Un controlador, que genere la señal que gobierna al actuador.
3. Un actuador, que modifique al sistema de manera controlada (resistencia eléctrica, motor, válvula, bomba, etc).

El sensor proporciona una señal analógica o digital al controlador, la cual representa el *punto actual* en el que se encuentra el proceso o sistema. La señal puede representar ese valor en tensión eléctrica, intensidad de corriente eléctrica o frecuencia. En este último caso la señal es de corriente alterna, a diferencia de los dos anteriores, que son con corriente continua.

El controlador lee una señal externa que representa el valor que se desea alcanzar. Esta señal recibe el nombre de punto de consigna (o punto de referencia), la cual es de la misma naturaleza y tiene el mismo rango de valores que la señal que proporciona el sensor. Para hacer posible esta compatibilidad y que, a su vez, la señal pueda ser entendida por un humano, habrá que establecer algún tipo de interfaz(HMI-Human Machine Interface), son pantallas de gran valor visual y fácil manejo que se usan para hacer más intuitivo el control de un proceso.

El controlador resta la señal de punto actual a la señal de punto de consigna, obteniendo así la señal de error, que determina en cada instante la diferencia que hay entre el valor deseado (consigna) y el valor medido. La señal de error es utilizada por cada uno de los 3 componentes del controlador PID. Las 3 señales sumadas, componen la señal de salida que el controlador va a utilizar para gobernar al actuador. La señal resultante de la suma de estas tres se llama **variable manipulada** y no se aplica directamente sobre el actuador, sino que debe ser transformada para ser compatible con el actuador utilizado.

Las tres componentes de un controlador PID son: parte **P**roporcional, acción **I**ntegral y acción **D**erivativa. El peso de la influencia que cada una de estas partes tiene en la suma final, viene dado por la constante proporcional, el tiempo integral y el tiempo derivativo, respectivamente. Se pretenderá lograr que el bucle de control corrija eficazmente y en el mínimo tiempo posible los efectos de las perturbaciones.

5.2.1.2. Proporcional

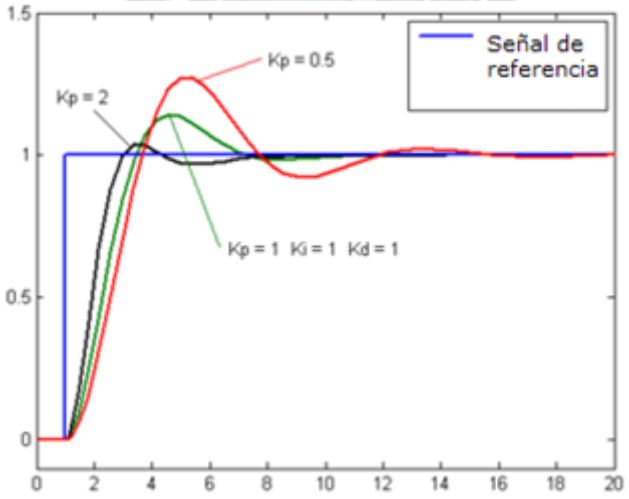


Ilustración 13: proporcional
Fuente: Jimmy Wales- Wikipedia.

La parte proporcional consiste en el producto entre la señal de error y la constante proporcional como para que hagan que el error en estado estacionario sea casi nulo, pero en la mayoría de los casos, estos valores solo serán óptimos en una

determinada porción del rango total de control, siendo distintos los valores óptimos para cada porción del rango. Sin embargo, existe también un valor límite en la constante proporcional a partir del cual, en algunos casos, el sistema alcanza valores superiores a los deseados. Este fenómeno se llama sobreoscilación y, por razones de seguridad, no debe sobrepasar el 30%, aunque es conveniente que la parte proporcional ni siquiera produzca sobreoscilación. Hay una relación lineal continua entre el valor de la variable controlada y la posición del elemento final de control (la válvula se mueve al mismo valor por unidad de desviación). La parte proporcional no considera el tiempo, por lo tanto, la mejor manera de solucionar el error permanente y hacer que el sistema contenga alguna componente que tenga en cuenta la variación respecto al tiempo, es incluyendo y configurando las acciones integral y derivativa.

La fórmula del proporcional esta dada por: $P_{\text{sal}} = K_p\, e(t)$

El error, la banda proporcional y la posición inicial del elemento final de control se expresan en tanto por uno. Nos indicará la posición que pasará a ocupar el elemento final de control

5.2.1.3. Integral

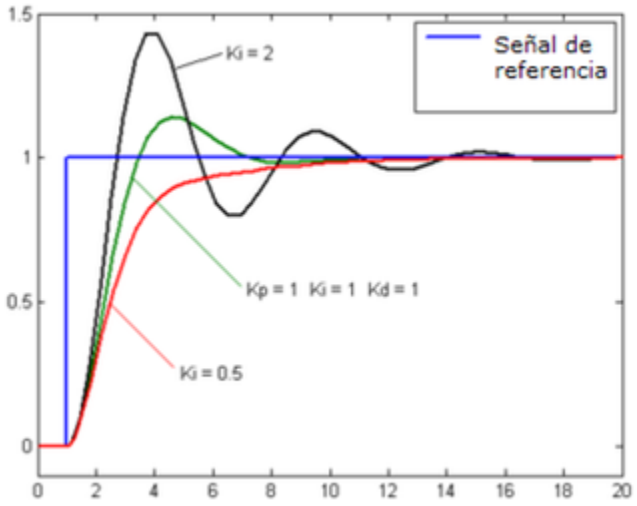


Ilustración 14: Integral
Fuente: Jimmy Wales- Wikipedia.

El modo de control Integral tiene como propósito disminuir y eliminar el error en estado estacionario, provocado por el modo proporcional. El control integral actúa cuando hay una desviación entre la variable y el punto de consigna, integrando esta desviación en el tiempo y sumándola a la acción proporcional. El *error* es integrado, lo cual tiene la función de promediarlo o sumarlo por un período determinado; Luego es multiplicado por una constante **I**. Posteriormente, la respuesta integral es adicionada al modo Proporcional para formar el control P + I con el propósito de obtener una respuesta estable del sistema sin error estacionario.

El modo integral presenta un desfasamiento en la respuesta de 90º que sumados a los 180º de la retroalimentación (negativa) acercan al proceso a tener un retraso de 270º, luego entonces solo será necesario que el tiempo muerto contribuya con 90º de retardo para provocar la oscilación del proceso. <<< la ganancia total del lazo de control debe ser menor a 1, y así inducir una atenuación en la salida del controlador para conducir el proceso a estabilidad del mismo. >>> Se caracteriza por el tiempo de acción integral en minutos por repetición. Es el tiempo en que delante una señal en escalón, el elemento final de control repite el mismo movimiento correspondiente a la acción proporcional.

El control integral se utiliza para obviar el inconveniente del offset (desviación permanente de la variable con respecto al punto de consigna) de la banda proporcional.

La formula del integral esta dada por: $$I_{\mathbf{sal}} = K_i \int_0^t e(\tau)\, d\tau$$

Ejemplo: Mover la válvula (elemento final de control) a una velocidad proporcional a la desviación respeto al punto de consigna (variable deseada).

5.2.1.4. Derivativo

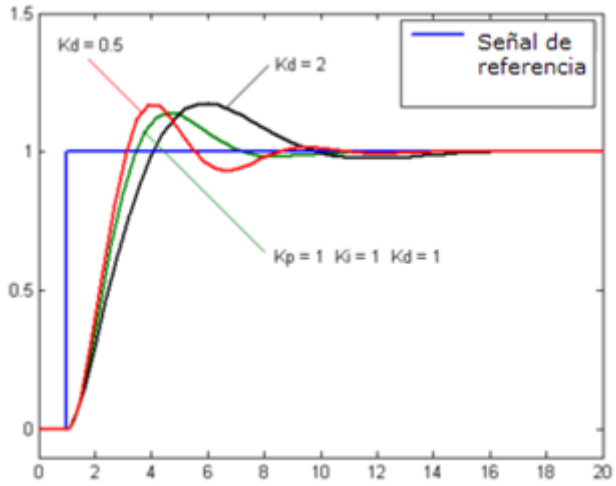


Ilustración 15: Derivativo
Fuente: Jimmy Wales- Wikipedia.

La acción derivativa se manifiesta cuando hay un cambio en el valor absoluto del error; (si el error es constante, solamente actúan los modos proporcional e integral).

El *error* es la desviación existente entre el punto de medida y el valor consigna, o "*Set Point*".

La función de la acción derivativa es mantener el error al mínimo corrigiéndolo proporcionalmente con la misma velocidad que se produce; de esta manera evita que el error se incremente.

Se deriva con respecto al tiempo y se multiplica por una constante **D** y luego se suma a las señales anteriores (P+I). Es importante adaptar la respuesta de control a los cambios en el sistema ya que una mayor derivativa corresponde a un cambio más rápido y el controlador puede responder acordemente.

La fórmula del derivativo esta dada por: $$D_{\text{sal}} = K_d \frac{de}{dt}$$

El control derivativo se caracteriza por el tiempo de acción derivada en minutos de anticipo. La acción derivada es adecuada cuando hay retraso entre el movimiento de la válvula de control y su repercusión a la variable controlada.

Cuando el tiempo de acción derivada es grande, hay inestabilidad en el proceso. Cuando el tiempo de acción derivada es pequeño la variable oscila demasiado con relación al punto de consigna. Suele ser poco utilizada debido a la sensibilidad al ruido que manifiesta y a las complicaciones que ello conlleva.

El tiempo óptimo de acción derivativa es el que retorna la variable al punto de consigna con las mínimas oscilaciones

Ejemplo: Corrige la posición de la válvula (elemento final de control) proporcionalmente a la velocidad de cambio de la variable controlada.

La acción derivada puede ayudar a disminuir el rebasamiento de la variable durante el arranque del proceso. Puede emplearse en sistemas con tiempo de retardo considerables, porque permite una repercusión rápida de la variable después de presentarse una perturbación en el proceso.

5.2.2. Controlador P

La estructura de un controlador P introduce una ganancia a la entrada de la planta. La estructura es la que se muestra en la Ilustración 28. Para sistemas de tipo cero, el efecto del controlador proporcional es el siguiente:

Disminuye el tiempo de subida
Aumenta el sobrepico
No provoca grandes cambios en el tiempo de establecimiento
Disminuye el error en régimen permanente, pero no lo elimina

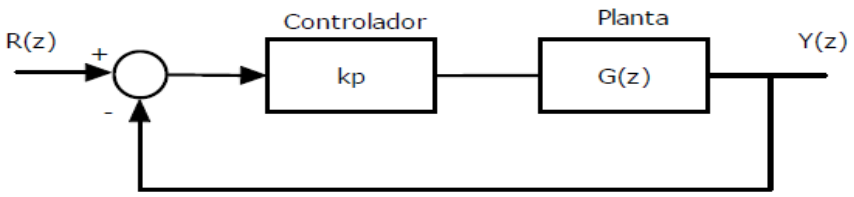


Ilustración 16: Diagrama de bloques del controlador P con la planta del sistema
Fuente: Rafael Moragues, *Sistema de control en temps real a través de MATLAB/SIMULINK*, PFC

Para realizar el ajuste de la constante proporcional hay que obtener la función de transferencia en lazo cerrado con la incógnita *kp*. Mediante la regla de *Mason* se cierra el lazo quedando la expresión siguiente.

$$\frac{Y(z)}{R(z)} = \frac{kp \cdot G(z)}{1 + kp \cdot G(z)}$$

La expresión del error:

$$E(z) = \frac{R(z)}{1 + kp \cdot G(z)}$$

Haciendo el desarrollo de la primera expresión y sustituyendo el valor del tiempo de muestreo, se aplica el Teorema del Valor Final:

$$y_{ss} = \lim_{z \to 1} \frac{(z-1)}{z} \cdot Y(z)$$

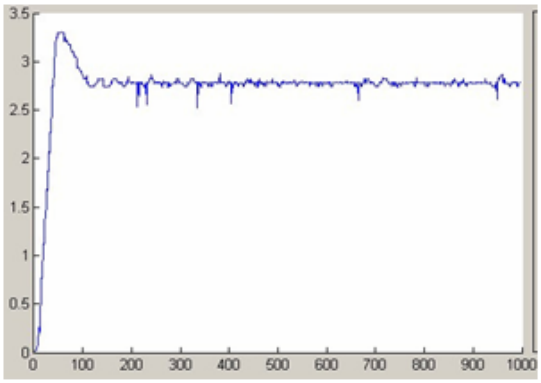
A continuación se muestra un resumen de los valores obtenidos para cada tiempo de muestreo. Se considera la entrada como un escalón unitario y el error en estado estacionario es del 1%, lo que quiere decir que el valor de la salida (y_{SS}) es 0,99. El valor de la constante proporcional se obtiene aislando la última expresión.

Tiempo de muestreo (s)	Valor Constante Proporcional kp
0,01	89,5
0,001	93,4

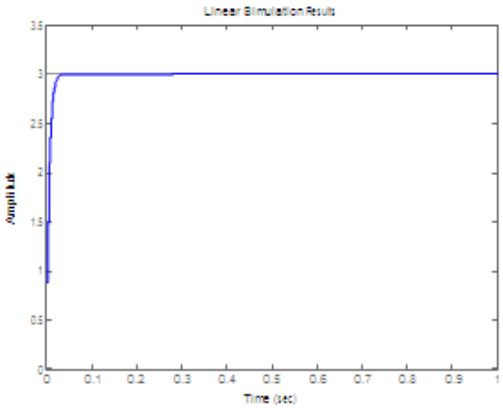
Tabla 4: Valores kp en función del tiempo de muestreo
Fuente: Rafael Moragues, *Sistema de control en temps real a través de MATLAB/SIMULINK*, PFC URV

En este caso, al ser un sistema de primer orden, mirar el lugar de las raíces no aporta información significativa debido a que no se introduce ningún polo o cero. Por tanto no habrá grandes cambios dentro de los valores de ganancia para los que el sistema es estable.

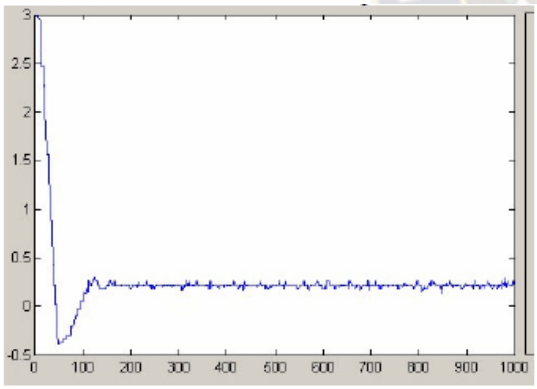
Para el tiempo de muestreo correspondiente a 0,001 segundos y con consigna de 3:



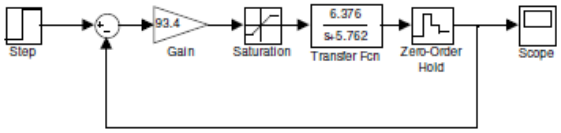
(a) Representación tensión de salida en función del número de muestras. Respuesta real controlador a Ts=0,001s kp=93,4



(b) Representación tensión de salida en función del tiempo. Respuesta simulación controlador a Ts=0,001s con kp=93,4

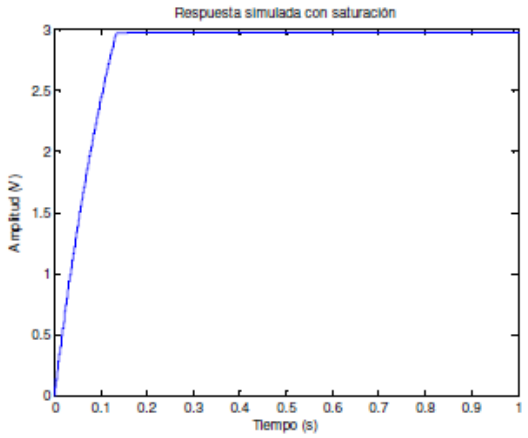


(c) Representación tensión de salida en función del número de muestras. Error real Controlador a Ts=0,001s

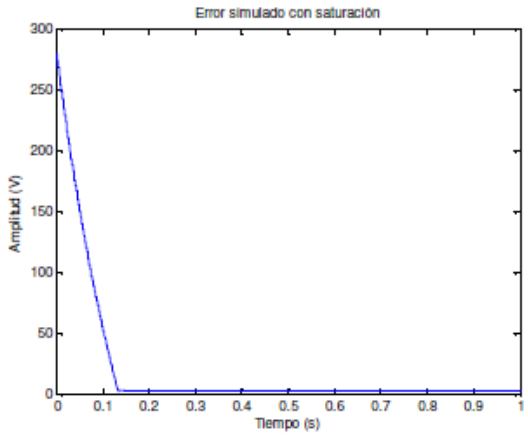


(d) Modelo con controlador proporcional y saturación a Ts=0,001s

Ilustración 17: Comparación entre controlador P real y simulado
Fuente: Propia



(a) Representación tensión de salida en función del tiempo. Respuesta simulación controlador a Ts=0,001s con kp=93,4



(b) Representación tensión de salida en función del tiempo. Error real a Ts=0,001s con kp=93,4

Ilustración 18: Respuestas obtenidas teniendo en cuenta la saturación
Fuente: Propia

5.1.3. Controlador PI

El controlador PI, al contener la parte integral, a diferencia del anterior incrementa el tiempo de establecimiento y elimina el error en estado estacionario. Con la acción combinada de la parte integral y proporcional habrá que reducir el sobrepico y el tiempo de establecimiento. En la siguiente ilustración se muestra el esquema de bloques típico de un sistema en lazo cerrado con un controlador PI.

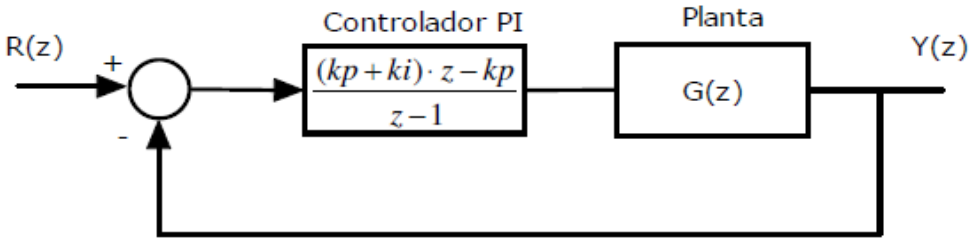
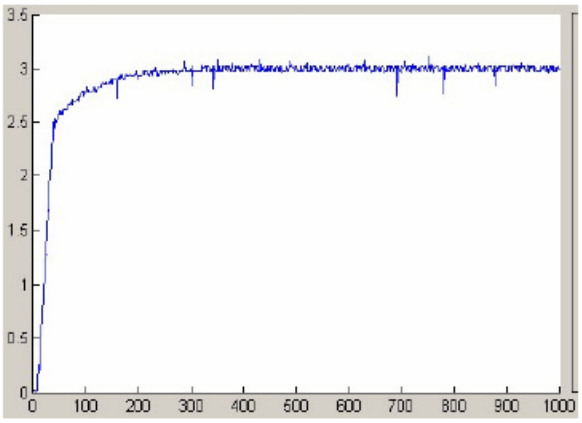


Ilustración 19: Diagrama de bloques del controlador PI con la planta del sistema
Fuente: Rafael Moragues, *Sistema de control en temps real a través de MATLAB/SIMULINK*, PFC

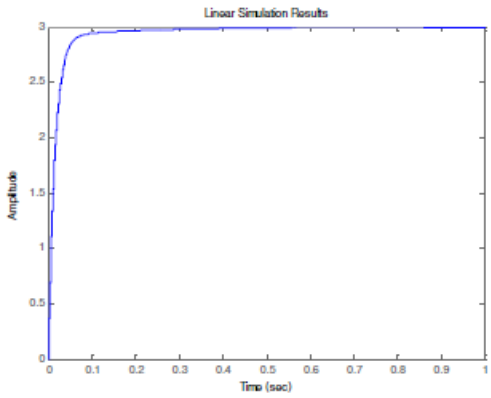
La expresión del controlador PI:

$$\frac{Y(z)}{R(z)} = kp + ki \cdot \frac{z}{z-1}$$

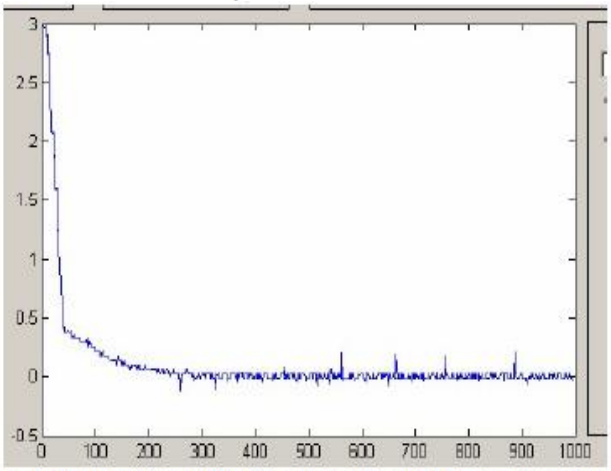
Las respuestas obtenidas en la simulación y en la aplicación Real Time para el tiempo de muestreo de 0,001 segundos e introduciendo una consigna escalón de 3 se representan en la siguiente Ilustración



(a) Representación tensión de salida en función del número de muestras. Respuesta real controlador a Ts=0,001s



(b) Representación tensión de salida en funció del tiempo. Respuesta simulación controlador Ts=0,001s



(c) Representación tensión de salida en función del número de muestras. Error real Controlador a Ts=0,001s

Ilustración 19: Comparación entre controlador PI real y simulado
Fuente: Propia

Las funciones de transferencia de los controladores se reflejan en la siguiente tabla.

Tiempo de muestreo (s)	Función de Transferencia Discreta (Z)
0,01	$D(z) = \frac{z - 0{,}996}{z - 1}$ (33)
0,001	$D(z) = \frac{z - 0{,}98}{z - 1}$ (34)

Tabla 5: Funciones de transferencia en función del tiempo de muestreo
Fuente: Propia

5.3. Funcionamiento del interfaz gráfico

A continuación se presenta el interfaz gráfico realizado con la herramienta de Matlab, GUIDE. El interfaz permitirá controlar la velocidad de un motor y su carga en lazo cerrado, pudiendo modificar parámetros como: velocidad, rampa, sentido de giro, valores de controlador, tiempo de simulación, tiempo de muestreo, etc. y una vez acabada la simulación se podrán visualizar los parámetros y grabarlos en un fichero.MAT. Otra característica interesante es poder decirle al programa si queremos trabajar en lazo abierto o cerrado, según convenga. Más adelante se explicará con detalle que opciones tiene el programa, como trabaja, y cual es el código que lo hace funcionar.

El programa según las partes más relevantes, que son las siguientes:

Elección del modelo de trabajo
Parámetros de entrada
Parámetros del controlador
Parámetros de ejecución
Visualización de datos
Guardar datos
Casilla de diálogo
Función de cierre de programa

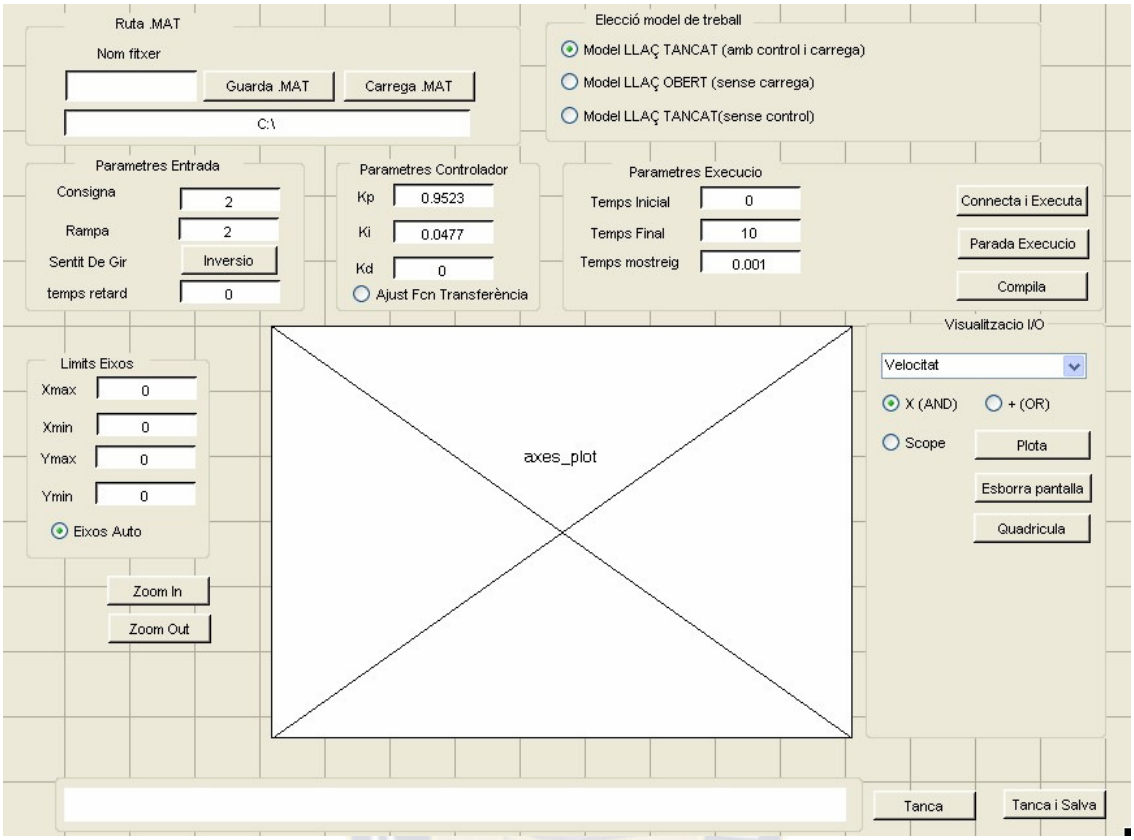


Ilustración 20: Interfaz gráfica en fase de diseño
Fuente: Propia

5.3.1. Elección del modelo de trabajo

A través de las casillas de selección se puede escoger el modelo de Simulink con el que se desea trabajar. Existen tres opciones:

Modelo en lazo cerrado con controlador discreto (en Z) y con carga
Modelo en lazo abierto
Modelo en lazo cerrado sin controlador y con carga

5.3.1.1 Modelo en Lazo cerrado con controlador y carga

Esta es la elección por defecto con la que arranca el programa. El modelo que se abre es el fichero *rtw.mdl* y tiene el aspecto de la Ilustración 19. Por defecto arranca con un controlador PI diseñado para las condiciones con las que inicia.

Aunque también es posible diseñar cualquier otro controlador que tenga las partes proporcional, integral y/o derivativa.

El patrón utilizado en este modelo es el mismo que se utiliza en los otros modelos, de esta manera las rutas y los nombres de las variables y bloques serán aprovechables en la parte de programación, lo que simplificará mucho el trabajo

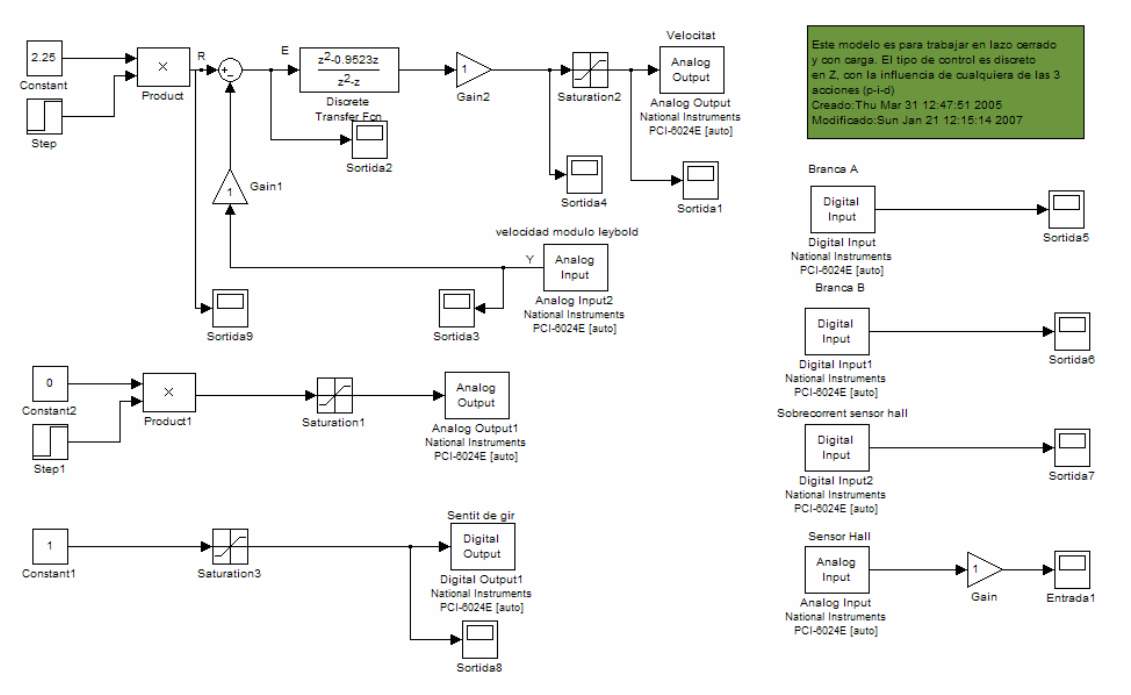


Ilustración 21: Aspecto del modelo de Simulink en lazo cerrado con controlador y con carga
Fuente: Propia

Observando la parte principal del control, podemos ver que mediante la variación de bloque *Constant* se podrá modificar el valor del escalón. El bolque *Sortida9* permitirá visualizar las diferencias entre el valor de entrada respecto al de salida. Mediante el bloque *Sortida2* se podrá visualizar el error del controlador. Se ha puesto un limitador en la salida para absorber cualquier variación imprevista superior a unos valores de 0 a 5 V. De esta manera se protege la entrada del driver CC que trabaja con rangos de tensiones TTL. Después del limitador se encuentra el bloque de salida que inyectará el valor de la velocidad al driver de cc en ese instante de tiempo. El valor *real* de la velocidad del conjunto motor+carga es capturado por el tacómetro de la carga y reinyectado en el

sumador para volver a calcular el valor velocidad en el siguiente instante de tiempo, haciendo que el error tienda a cero.
El valor de la constante *Constante2* es el valor de la rampa, que como hemos dicho anteriormente el driver cc trabaja con lógica TTL, los valores admisibles son de 0 a 5 V y para que se cumpla esta condición está el limitador. Cuanto más alto sea el valor de la constante más lenta será la aceleración y la frenada.

El valor de la constante *Constant1* es el valor para modificar el sentido de giro, que en este modelo la opción queda deshabilitada a través del GUI debido a conflictos con el módulo D&B de LEYBOLD.

A mano derecha de la ilustración 21 quedan las entradas:

Sobre intensidad en la rama A
Sobre intensidad en la rama B
Sobre intensidad en el sensor Hall
Medición procedente del sensor Hall

5.3.1.2. Modelo en Lazo abierto

Esta es la segunda elección posible y el fichero que abre es el *rtwol.mdl*, y en caso que existiera cualquier otro modelo abierto lo cerraría guardando los cambios. El modelo no tiene ningún control como es lógico, ya que es un lazo abierto. El patrón que sigue es como el explicado en el anterior apartado.

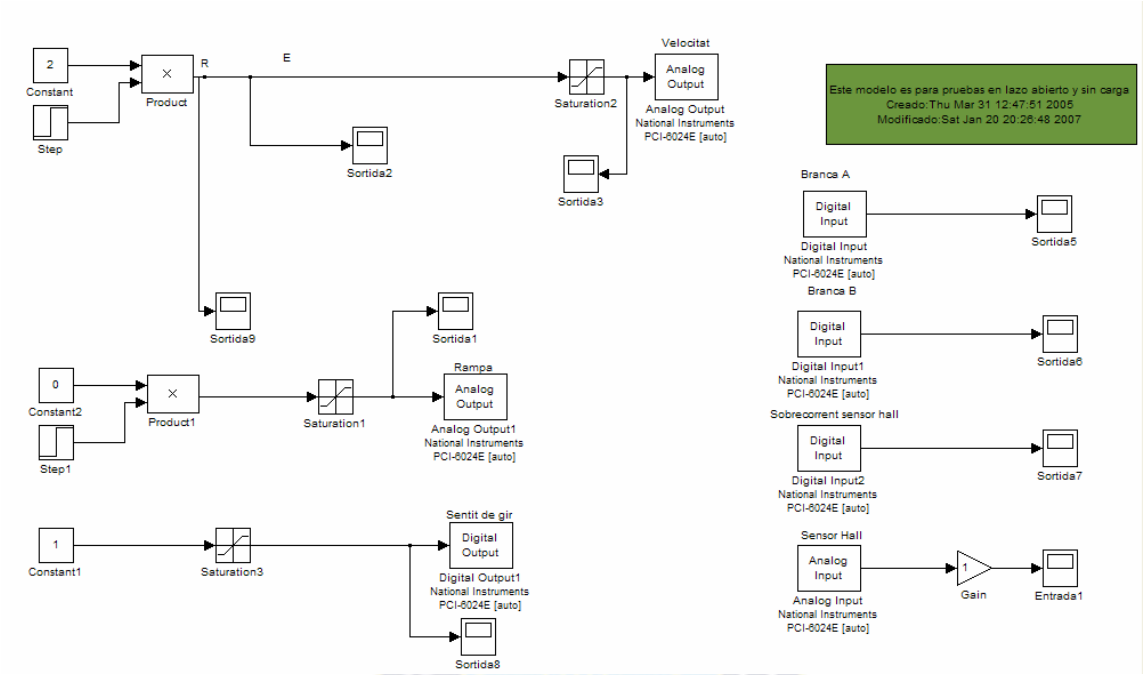


Ilustración 22: Aspecto del modelo en lazo abierto
Fuente: Propia

En esta opción es posible realizar cambios en el sentido de giro del motor, además de poder regular la rampa de aceleración y deceleración como se puede hacer en los otros modelos.

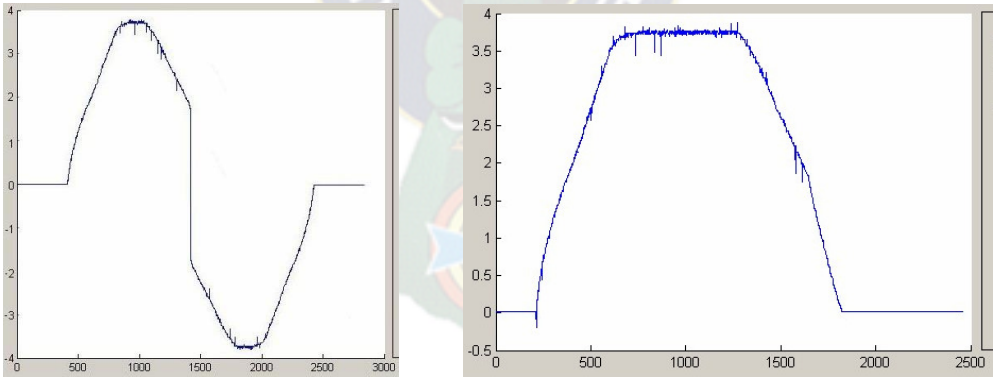


Ilustración 23: Inversión de giro y aceleración/deceleración del motor cc.
Fuente: Propia

La Ilustración 23 muestra el cambio en el sentido de giro del motor, se puede ver como primeramente como acelera a una rampa de valor 2 hasta conseguir el valor de la consigna que es 4, es a partir de este momento que se solicita el cambio de giro. Cuando se realiza el cambio de giro sería de esperar que fuera

decrementando hasta el valor de-4, pero por lo contrario el tramo de 2 a -2 lo realiza de manera vertiginosa. Cuando alcanza el valor de -4 lo obligamos a ir a consigna cero.

5.3.1.3. Modelo en lazo cerrado sin controlador y con carga.

Esta es la tercera opción posible, y abrirá el modelo *rtw_cl2.mdl* que es una réplica del *rtw.mdl* pero sin controlador. De esta manera se deja una puerta abierta a un posible control diferente al Z, como por ejemplo mediante la utilización de lógica borrosa.

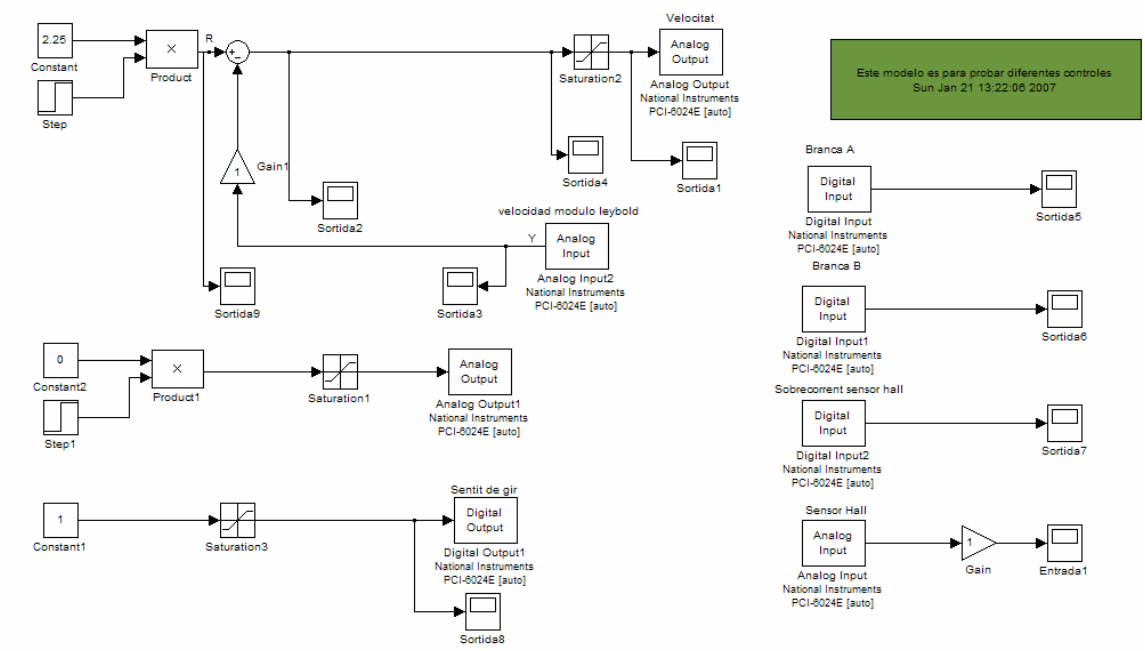


Ilustración 24: Aspecto del modelo en lazo cerrado sin controlador y con carga

Fuente: Propia

5.3.2. Diagrama de Flujo General

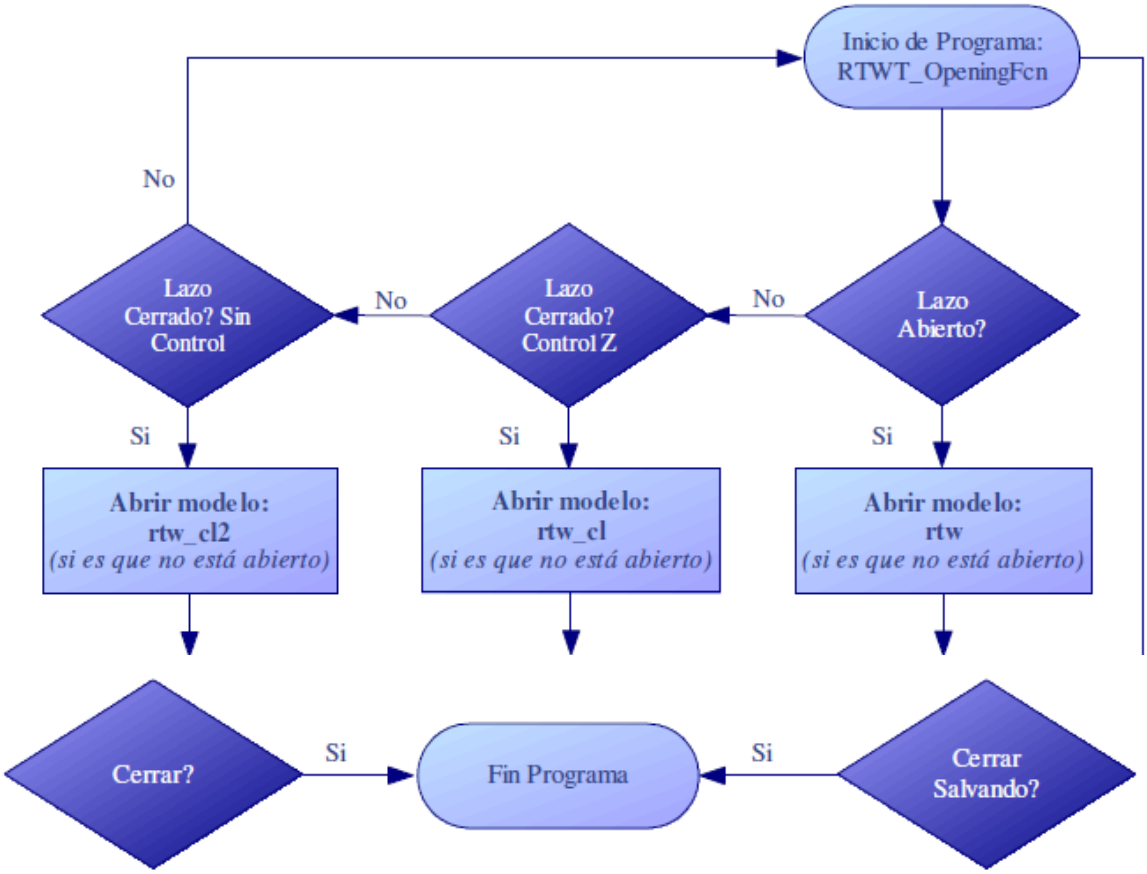


Ilustración 24: Diagrama de flujo general de la GUI
Fuente: Propia

6. CONCLUSIONES

Se puede decir que se han cumplido satisfactoriamente los propósitos planteados inicialmente, consiguiendo de esta manera sentar las bases para trabajos futuros en un entorno muy desconocido como es el trabajo con sistemas reales y en tiempo real mediante Matlab.

Se ha utilizado con éxito y ha quedado documentada la información referente a las herramientas GUIDE, Simulink, Real Time Windows Target, del programa de MatLab.

Todo ello se ha relacionado con la teoría de control para la obtención de un sistema realimentado para la regulación de la velocidad de un motor de corriente continua. A través de la aplicación ideada se podrá diseñar cualquier tipo de controlador soportado por Simulink (fuzzy, discreto o continuo), para posteriormente visualizar las gráficas reales y corroborarlas con las teóricas.

7. RECOMENDACIONES

Para trabajos futuros añadir un filtro en el modelo de Simulink para eliminar el ruido provocado por la conmutación de los IGBTs del puente en H.

Estar pendiente de nuevas actualizaciones de MatLab para la resolución del problema de la imposibilidad para capturar datos y/o realizar operaciones con las variables hasta que no haya acabado la ejecución en tiempo real de la aplicación.

BIBLIOGRAFIA.

1. Pol Toldrà Fernández, *Implementació d'un driver commutat a alta freqüència per un accionament regulat amb motor DC*, PFC URV,2004.

2. Rafael Moragues, *Sistema de control en temps real a través de MATLAB/SIMULINK*, PFC URV,
2004.

3. David Oliveras, *Desenvolupament d'una Plataforma Hardware/Software per a l'Experimentació d'Accionaments DC*, PFC URV, 2005.

4. Pedro Garcés, Apuntes de la asignatura de Ingeniería de Control II, URV 2004.

5. Sistemas de control en tiempo discreto, Katsuhiko Ogata, Ed. Pentice Hall, 2 edición. [6] Ayudas de Matlab dentro de Programa.

6. http://www.mathworks.com

7. The MathWorks, *Real-Time Windows Target User's Guide (Version 2).*

ANEXO
PROGRAMACION EN MATLAB

Funciónde Apertura

```
functionRTWT_OpeningFcn(hObject,eventdata,handles,varargin)
 %%%Secomienzaconelmodeloellazocerradoycontroladorpordefecto
 set(handles.rboto_cl,'Value',1);
 cl=1;ol=0;cl2=0;
 assignin('base','cl',cl)
 assignin('base','ol',ol)
 assignin('base','cl2',cl
 inici(handle
 actualitza(handle

 ChoosedefaultcommandlineoutputforRTWT
 handles.outputhObject;


 Updatehandlesstructure
 guidata(hObject,handles
 );
functioninici(handles)  cl=evalin('base','cl');  ol=evalin('base','ol');
cl2=evalin('base','cl2');
 %%%%%%%%%%%INICIOPARAM.GENERALES%%%%%%%%%%%%%
 %Secapturanlosvaloresdelosparametrosgenerales
 ruta=cd;
 set(handles.edit_ruta,'String',ruta);
 %Sehaceinvisiblelaeditboxdelnombredelavariableaguardarenel
 WS
 set(handles.edit_nomvar,'Visible','off');
 %Variableparaelgridparaqueactuecomounpushbuttonperosin
 %enclavarse
 cnt=0;
 assignin('base','cnt',cnt);
 %Parametrosparaplotarpordefecto
 set(handles.rboto_and,'Value',1);
 set(handles.rboto_or,'Value',0);
 set(handles.rboto_scope,'Value',0);
 set(handles.popupmenu1,'Value',1);
 pand=1;or=0;scope_stat=0;popval=1;
 assignin('base','pand',pand);
 assignin('base','or',or);
 assignin('base','scope',scope_stat);
 assignin('base','popval',popval);
 %%%%%%%%%%%FINPARAM.GENERALES%%%%%%%%%%%%%
ifcl==1

     ifisempty(find_system('Name','rtw'))open_system('rtw');end
     ifisempty(find_system('Name','rtwol'))else
 close_system('rtwol',1);end
     ifisempty(find_system('Name','rtw_cl2'))else

set(handles.edit_consigna,'String',get_param('rtw/Constant','Value'));
     %Rampa
     set(handles.edit_rampa,'String',get_param('rtw/Constant2','Value'));
     %Tiempoderetardoparalarampayelescalón
     set(handles.edit_td,'String',get_param('rtw/Step','time'));
```

```
    limitapunts(handles);
    controlador(handles);
end

ifol==1

    ifisempty(find_system('Name','rtwol'))open_system('rtwol');end
    ifisempty(find_system('Name','rtw'))elseclose_system('rtw',1);end
    ifisempty(find_system('Name','rtw_cl2'))else
close_system('rtw_cl2',1);end
    %Secogenlosvaloresdelmodeloyseponendentrodelaseditboxes
    %Consigna

set(handles.edit_consigna,'String',get_param('rtwol/Constant','Value'));
    %Rampa

set(handles.edit_rampa,'String',get_param('rtwol/Constant2','Value'));
    %Tiempoderetardoparalarampayelescalón
    set(handles.edit_td,'String',get_param('rtwol/Step','time'));
    %Sentidodegiro

set(handles.tboto_inversio,'Value',str2num(get_param('rtwol/Constant1','
Value')));
    limitapunts(handles);
end

ifcl2==1

    ifisempty(find_system('Name','rtw_cl2'))open_system('rtw_cl2');end
    ifisempty(find_system('Name','rtwol'))else
close_system('rtwol',1);end
    ifisempty(find_system('Name','rtw'))elseclose_system('rtw',1);end
    %Secogenlosvaloresdelmodeloyseponendentrodelaseditboxes
    %Consigna

set(handles.edit_consigna,'String',get_param('rtw_cl2/Constant','Value')
);
    %Rampa

set(handles.edit_rampa,'String',get_param('rtw_cl2/Constant2','Value'));
    %Tiempoderetardoparalarampayelescalón
    set(handles.edit_td,'String',get_param('rtw_cl2/Step','time'));
    limitapunts(handles);
end
functionactualitza(handles)
ol=evalin('base','ol');
cl=evalin('base','cl');
cl2=evalin('base','cl2');
ifcl==1
    ifisempty(find_system('Name','rtw'))open_system('rtw');
    end
ifisempty(find_system('Name','rtwol'))else
close_system('rtwol',1);end
    ifisempty(find_system('Name','rtw_cl2'))else
close_system('rtw_cl2',1);end
    %figure(RTWT);
```

```
    %Secogeelvalordelaeditboxiloponeelbloquedelmodelo
    set_param('rtw/Constant','Value',...
    get(handles.edit_consigna,'String'))
set_param('rtw/Constant2','Value',...
get(handles.edit_rampa,'String'))
%Tiempoderetardoparaelescalonylarampa
set_param('rtw/Step','Time',get(handles.edit_td,'String'));
set_param('rtw/Step1','Time',get(handles.edit_td,'String'));
%Secogeelvalordelaeditboxysecolocacomoparametrogeneral
%lasimulación

%PARAMETROSDELASIMULACION
set_param('rtw','FixedStep',get(handles.edit_ts,'String'));
set_param('rtw','Starttime',get(handles.edit_ti,'String'));
set_param('rtw','Stoptime',get(handles.edit_tf,'String'));
limitapunts(handles);
controlador(handles);
 end
 ifol==1
    ifisempty(find_system('Name','rtwol'))open_system('rtwol');end
    ifisempty(find_system('Name','rtw'))elseclose_system('rtw',1);end
    ifisempty(find_system('Name','rtw_cl2'))else
 close_system('rtw_cl2',1);end
    %figure(RTWT);

 set_param('rtwol/Constant1','Value',num2str(get(handles.tboto_inversio,'
 Value')));
    %Secogeelvalordelaeditboxiloponeelbloquedelmodelo

 set_param('rtwol/Constant','Value',get(handles.edit_consigna,'String'));

 set_param('rtwol/Constant2','Value',get(handles.edit_rampa,'String'));
    %Tiempoderetardoparaelescalonylarampaset_param('rtwol/Step','
    Time',get(handles.edit_td,'String'));
    set_param('rtwol/Step1','Time',get(handles.edit_td,'String'));

 PARAMETROSDELASIMULACION
 set_param('rtwol','FixedStep',get(handles.edit_ts,'String'));
 set_param('rtwol','Starttime',get(handles.edit_ti,'String'));
 set_param('rtwol','Stoptime',get(handles.edit_tf,'String'));
 limitapunts(handles),


 ifcl2==1

    ifisempty(find_system('Name','rtw_cl2'))open_system('rtw_cl2');end
    ifisempty(find_system('Name','rtwol'))else
 close_system('rtwol',1);end
    ifisempty(find_system('Name','rtw'))elseclose_system('rtw',1);end
    %Secogeelvalordelaeditboxiloponeelbloquedelmodelo
    set_param('rtw_cl2/Constant','Value',...
    get(handles.edit_consigna,'String'))
set_param('rtw_cl2/Constant2','Value',...
get(handles.edit_rampa,'String'))
%Tiempoderetardoparaelescalonylarampaset_param('rtw_cl2/Step','Time'
,get(handles.edit_td,'String'));
set_param('rtw_cl2/Step1','Time',get(handles.edit_td,'String'));
```

```
%Secogeelvalordelaeditboxysecolocacomoparametrogeneral
%lasimulación
%PARAMETROSDELASIMULACION
set_param('rtw_cl2','FixedStep',get(handles.edit_ts,'String'));
set_param('rtw_cl2','Starttime',get(handles.edit_ti,'String'));
set_param('rtw_cl2','Stoptime',get(handles.edit_tf,'String'));
limitapunts(handles);
end
```

Parámetrosdeentrada

```
 functionedit_consigna_Callback(hObject,eventdata,handles)
 actualitza(handles)
functionedit_rampa_Callback(hObject,eventdata,handles)
 actualitza(handles);
functiontboto_inversio_Callback(hObject,eventdata,handles)
 actualitza(handles);
functionedit_td_Callback(hObject,eventdata,handles)
 actualitza(handles);
```

Parámetrosdelcontrolador

```
 functionedit_kp_Callback(hObject,eventdata,handles)
controlador(handles);
functionrboto_ajustFcnTf_Callback(hObject,eventdata,handles)
     if(get(hObject,'Value'))
         set(handles.edit_kp,'Visible','off');
         set(handles.edit_ki,'Visible','off');
         set(handles.edit_kd,'Visible','off');
         open_system('rtw/DiscreteTransferFcn');
     else
set(handles.edit_kp,'Visible','on');
set(handles.edit_ki,'Visible','on');
set(handles.edit_kd,'Visible','on');
     end
functioncontrolador(handles)
kd=str2double(get(handles.edit_kd,'String'));
ki=str2double(get(handles.edit_ki,'String'));
kp=str2double(get(handles.edit_kp,'String'));
p=(kp+ki+kd);
q=-(kp+2*kd);
r=kd;
 u=1;
 v=-1;
 w=0;
num=[pqr];
 den=[uvw];
 numdz=strcat('[',num2str(num),']');
 dendz=strcat('[',num2str(den),']');
 set_param('rtw/DiscreteTransferFcn','Denominator',dendz);
 set_param('rtw/DiscreteTransferFcn','Numerator',numdz);
```

Parámetrosdeejecución

```
 functionedit_ti_Callback(hObject,eventdata,handles)
```

```
 actualitza(handles);
functionedit_tf_Callback(hObject,eventdata,handles)
 actualitza(handles);
 functionedit_ts_Callback(hObject,eventdata,ha
 ndles)
 actualitza(handles);
functionboto_connecta_Callback(hObject,eventdata,handles)
 ol=evalin('base','ol');
 cl=evalin('base','cl');
 cl2=evalin('base','cl2');

 ifcl==1
     set(handles.text_dialeg,'String',...
                  'Connectantiexecutant...');
     set_param('rtw','SimulationMode','external');
     set_param('rtw','SimulationCommand','connect');
     set_param('rtw','SimulationCommand','start');
 end

 ifol==1
     set(handles.text_dialeg,'String',...
                 'Connectantiexecutant...');
     set_param('rtwol','SimulationMode','external');
     set_param('rtwol','SimulationCommand','connect');
     set_param('rtwol','SimulationCommand','start');
 end
 ifcl2==1
     set(handles.text_dialeg,'String',...
                 'Connectantiexecutant...');
     set_param('rtw_cl2','SimulationMode','external');
     set_param('rtw_cl2','SimulationCommand','connect');
     set_param('rtw_cl2','SimulationCommand','start');
 end
 assignin('base','cl',cl);
 assignin('base','ol',ol);
 assignin('base','cl2',cl2);

 functionboto_stop_Callback(hObject,eventdata,handles)
 ol=evalin('base','ol');
 cl=evalin('base','cl');
 cl2=evalin('base','cl2');

     ifcl==1set_param('rtw','SimulationCommand','stop');
         set_param('rtw','SimulationCommand','disconnect');
         set(handles.text_dialeg,'String',...
                     'S´haparatl´Execució del´aplicació"Real
 Time"!');
     end

     ifol==1set_param('rtwol','SimulationCommand','stop');
         set_param('rtwol','SimulationCommand','disconnect');
         set(handles.text_dialeg,'String',...
                     'S´haparatl´Execució del´aplicació"Real
 Time"!');
     end
```

```
    ifcl2==1set_param('rtw_cl2','SimulationCommand','stop');
        set_param('rtw_cl2','SimulationCommand','disconnect');
        set(handles.text_dialeg,'String',...
                    'S´haparatl´Execucióde l´aplicació"Real
 Time"!');
    end

 assignin('base','cl',cl);
 assignin('base','ol',ol);
 assignin('base','cl2',cl2);
 functionboto_compila_Callback(hObject,eventdata,handles)
 ol=evalin('base','ol');
 cl=evalin('base','cl');
 cl2=evalin('base','cl2');

    ifcl==1
        set(handles.text_dialeg,'String',...
                    'Compilant...');
pause(0.5);
rtwbuild('rtw');
%figure(RTWT);
set(handles.text_dialeg,'String',...
            'Finalitzadatascadecompilar');
    end
    ifcl2==1
    set(handles.text_dialeg,'String',...
                    'Compilant...');
pause(0.5);
rtwbuild('rtw_cl2');
%figure(RTWT);
set(handles.text_dialeg,'String',...
            'Finalitzadatascadecompilar');
End
 assignin('base','cl',cl);
 assignin('base','ol',ol);
 assignin('base','cl2',cl2);
```

Limitacióndepuntosdeejecución

```
functionlimitapunts(handles)
ol=evalin('base','ol');
cl=evalin('base','cl');

 ti=str2num(get_param(gcs,'Starttime'));
 tf=str2num(get_param(gcs,'Stoptime'));
 At=tf-ti;
 ifcl==1&&ol==0
    Ts=str2num(get_param('rtw','FixedStep'));
 elseifcl==0&&ol==1
    Ts=str2num(get_param('rtwol','FixedStep'));
 end
 %puntosmáximosderepresentación
 pt=str2num(get_param(gcs,'Decimation'));
```

```
 %Calculodelnumerodepuntos
 div=pt/Ts;
 Npt=At*div;
 Nptmax=40000;
 if(Npt>Nptmax)
     At=(Nptmax*Ts)/pt;
     ti=0;
     tf=num2str(At);

     msge='S´haexceditelnúmeromáximdepunts...Recàlculambla
 freqüènciademostreigseleccionada:';
     msge2='tf=';
     msge3=strcat(msge,msge2,tf);
     set(handles.text_dialeg,'String',...
         msge3);
 end

 %Seponenlosvaloresenlascasillascorrespondientesset(hand
 les.edit_ti,'String',num2str(ti));
 set(handles.edit_tf,'String',num2str(tf));
 set(handles.edit_ts,'String',num2str(Ts));
%Seactualizanlosvaloresenelmodelo
ifcl==1&&ol==0
     set_param('rtw','Starttime',num2str(ti));
     set_param('rtw','Stoptime',num2str(tf));
     set_param('rtw','FixedStep',num2str(Ts));
 elseifcl==0&&ol==1
     set_param('rtwol','Starttime',num2str(ti));
     set_param('rtwol','Stoptime',num2str(tf));
     set_param('rtwol','FixedStep',num2str(Ts));
 end
```

Visualizacióndegráficos

```
functionrboto_and_Callback(hObject,eventdata,handles)
 pand=get(hObject,'Value');
 ifpand==1
     set(handles.rboto_or,'Value',0);
     set(handles.rboto_scope,'Value',0);
     scope_stat=0
     pand=1;
     or=0;
 else
set(handles.rboto_or,'Value',0);
set(handles.rboto_and,'Value',1);
set(handles.rboto_scope,'Value',0);
scope_stat=0;
or=0;
pand=1;
 end
 assignin('base','pand',pand);
 assignin('base','or',or);
 assignin('base','scope_stat',scope_stat);
```

OpcióndegráficoOR

```
 functionrboto_or_Callback(hObject,eventdata,handles)
 or=get(hObject,'Value');
 ifor==1
     set(handles.rboto_and,'Value',0);
     set(handles.rboto_scope,'Value',0);
     scope_stat=0;
     or=1;
     pand=0;
 else
set(handles.rboto_and,'Value',0);
set(handles.rboto_or,'Value',1);
set(handles.rboto_scope,'Value',0);
scope_stat=0;
pand=0;
or=1;
 end
 assignin('base','or',or);
 assignin('base','pand',pand);
 assignin('base','scope_stat',scope_stat);
```

OpcióndegráficoSCOPE

```
functionrboto_scope_Callback(hObject,eventdata,handles)
scope_stat=get(hObject,'Value');
assignin('base','scope_stat',scope_stat);
 ifscope_stat==1
     set(handles.rboto_and,'Value',0);
     set(handles.rboto_or,'Value',0);
     scope_stat=1;
     or=0;
     pand=0;
 else
set(handles.rboto_and,'Value',0);
set(handles.rboto_or,'Value',0);
set(handles.rboto_scope,'Value',1);
scope_stat=1;
pand=0;
 or=0;

 end
 assignin('base','or',or);
 assignin('base','pand',pand);
 assignin('base','scope_stat',scope_stat);
functionpopupmenu1_Callback(hObject,eventdata,handles)
popval=get(hObject,'Value');
assignin('base','popval',popval);
functionboto_esborrap_Callback(hObject,eventdata,handles)
 holdoff;
 newplot
 quadricula=evalin('base',quadricula);
   ifcnt==1
       gridon;
end
functionboto_grid_Callback(hObject,eventdata,handles)
```

```
 quadricula=get(hObject,'Value')
 cnt=evalin('base','cnt');
 ifquadricula==1cnt=~cntassignin
     ('base','cnt',cnt)
 end
 ifcnt==1
     holdon;
     gridon
 elseifcnt==0
     holdoff
     gridoff
 end
functionboto_zi_Callback(hObject,eventdata,handles)
%handles.axes_plot.zoomIn
 if1
xm=str2double(get(handles.edit_xmin,'String'));
xM=str2double(get(handles.edit_xmax,'String'));
ym=str2double(get(handles.edit_ymin,'String'));
yM=str2double(get(handles.edit_ymax,'String'));
axis([0.618*xm0.618*xM0.618*ym0.618*yM]); eixos_e=axis;
set(handles.edit_xmin,'String',num2str(eixos_e(1,1)));
set(handles.edit_xmax,'String',num2str(eixos_e(1,2)));
set(handles.edit_ymin,'String',num2str(eixos_e(1,3)));
set(handles.edit_ymax,'String',num2str(eixos_e(1,4)));
functionboto_zo_Callback(hObject,eventdata,handles)
functionboto_plot_Callback(hObject,eventdata,handles)
 pand=evalin('base','pand');
 or=evalin('base','or');
 popval=evalin('base','popval');
 scope_stat=evalin('base','scope_stat');
 cnt=evalin('base','cnt');
 ol=evalin('base','ol');
 cl=evalin('base','cl');
 cl2=evalin('base','cl2');
 ifor==1
ifpopval==1
     x=evalin('base','velocitat(:,2)');
     plot(x);
 elseifpopval==2
     x=evalin('base','error(:,2)');
     plot(x);
 elseifpopval==3
     x=evalin('base','shall(:,2)');
     plot(x);
 elseifpopval==4
     x=evalin('base','sia(:,2)');


     plot(x);
  elseifpopval==5
     x=evalin('base','sib(:,2)');
     plot(x);
  elseifpopval==6
     x=evalin('base','sishall(:,2)');
     plot(x);
  elseifpopval==7
```

```
    x=evalin('base','entrada(:,2)');
    plot(x);
 elseifpopval==8
    x=evalin('base','sortida(:,2)');
    plot(x,'y');
 elseifpopval==9
    x=evalin('base','sortida3(:,2)');
    plot(x,'g');
end
```

Parámetrosdeguardarycargardatos

```
functionedit_nomvar_Callback(hObject,eventdata,handles)
 a=get(hObject,'String');
 assignin('base','a',a);
functionedit_ruta_Callback(hObject,eventdata,handles)
 ruta=get(handles.edit_ruta,'String');
 cd(ruta)
functionboto_guarda_Callback(hObject,eventdata,handles)
 %Esmiraquelavaraexisteixi,queeslaqueportaelnomevalin('base'
 ,'exista');
 f=evalin('base','ans');

 if(f)a=evalin('base','a');
     assignin('base','a',a);
     set(handles.edit_nomvar,'Visible','on');
     set(handles.edit_nomvar,'String',a);
     set(handles.text_dialeg,'String',...
     'Mirisielnomdelavariableilarutasoncorrectesipremi
 "guarda"');
     a=get(handles.edit_nomvar,'String');
     assignin('base','a',a);
     if(~isempty(a))cola='.mat';
         assignin('base','cola',cola)
         g=evalin('base','strcat(a,cola)');
         set(handles.text_dialeg,'String',...
'NomFitxerExistent.VolSobreescriue?
      if(handles.boto_guarda)
'base','save(a)');
    set(handles.text_dialeg,'String',...
    'Fitxer.MATGuardat');
     end
end
functionboto_carga_Callback(hObject,eventdata,handles)
 evalin('base','exista');
 f=evalin('base','ans');

 if(f)a=evalin('base','a');
     assignin('base','a',a);
     set(handles.edit_nomvar,'Visible','on');
     set(handles.edit_nomvar,'String',a);
     set(handles.text_dialeg,'String',...
     'Mirisielnomdelavariableilarutasoncorrectesipremi
 "carrega"');
     a=get(handles.edit_nomvar,'String');
```

```
      assignin('base','a',a);

      if(~isempty(a))cola='.mat';
          assignin('base','cola',cola)
          g=evalin('base','strcat(a,cola)');
          if(exist(g,'file'))
          evalin('base','load(a)');
          set(handles.text_dialeg,'String',...
              'Fitxer.MATCarregat');
       else
set(handles.text_dialeg,'String',...
'Fitxer.MATNOExistent.Torna-hi');
      end
end
 else
set(handles.edit_nomvar,'Visible','on');
set(handles.text_dialeg,'String',...
    'Introdueixalacasellaelnomdelfitxer.MATacarregar');
a=get(handles.edit_nomvar,'String');
assignin('base','a',a);
%Estornaacomprovarqueshaguiintroduitunnom
      evalin('base','exista');
      f=evalin('base','ans');
      e=evalin('base','isempty(a)');
      if(f==1&&e==0)evalin('base','load(a)');
          set(handles.text_dialeg,'String',...
              'Fitxer.MATCarregat');
       else
set(handles.text_dialeg,'String',...
    'Introdueixalacasellaelnomdelfitxer.MATa
      carregar'
      end
end
functionboto_tanca_Callback(hObject,eventdata,handles)
 ifisempty(find_system('Name','rtw'))elseclose_system('rtw');end
 ifisempty(find_system('Name','rtwol'))elseclose_system('rtwol');end
       close
 functionboto_s_i_g_Callback(hObject,eventdata,handles)
 close_system('rtw',1);
 close_system('rtwol',1);
       close
```