UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

Facultad de Ingeniería

"IMPLEMENTACIÓN DE UN CONTROL PID EMBEBIDO PARA MOTORES DE CORRIENTE DIRECTA"

Tesis

Que para obtener el título de:

Ingeniero en Mecatrónica

Presenta:

René Míchel Martínez Trujillo

Asesor:

Dr. Luis Omar Moreno Ahedo

Implementación de un control PID embebido para motores de corriente directa

René Míchel Martínez Trujillo

17 de Mayo del 2016

Índice general

Índice de figuras

Índice de tablas

Algoritmos

0.1. Resumen

El siguiente documento presenta el desarrollo e implementación del controlador de movimiento LM629 como un sistema de control PID embebido para motores de corriente directa, que con sus características distintivas y por ser un circuito integrado dedicado lo hacen una excelente opción para ser utilizado en distintas aplicaciones donde el movimiento de un motor sea critico para el proceso y/o sistema, agregando que es un controlador de precio accesible.

El documento esta conformado por cinco capítulos donde en el primer capítulo se

da la introducción al tema y se establecen los objetivos, así como la justificación y la problemática a resolver. En el capítulo dos se analizan conceptos básicos de sistemas de control, motores eléctricos, acciones de control, sintonización y las problemáticas de implementación del PID. En el capitulo tres se establece el desarrollo e implementación del sistema, donde se analiza detalladamente el controlador de movimiento LM629 y los componentes utilizados en el mismo. En el capítulo cuatro se abordan las pruebas realizadas y los resultados obtenidos de las mismas. Y por último en el capítulo cinco se muestran las conclusiones del desarrollo de este documento, así como los trabajos a futuros a a partir de esta aportación.

0.2. Abstract

The following document presents the development and implementation of the motion controller LM629 as an embedded PID control system for direct current motors, wich for its distinctive features and being a dedicated integrated circuit is an excellent choice to use in a variety of applications where a motor movement is critical to the process or system, including that it's a driver of an affordable price.

This document comprises five chapters. In the first chapter introduction to the theme and its objectives are given, as well as the justification and the problems that are set to solve. Chapter two discusses basic concepts of electrical control systems, motors, control, tuning and the problems of the implementation of PID actions. Chapter three establishes the development and implementation of the system, which analyzes in detail the LM629 motion controller and the rest of the components used for this application . The tests carried out and the results there of are discussed in chapter four. And finally in chapter five are the conclusions obtained during the development of this document, as well as a description of the future work that is expected as result of this contribution.

Parte I

Antecedentes

Capítulo 1

Introducción

En este capítulo se hace la descripción de los objetivos generales y específicos, la problemática a resolver, la aportación tecnológica y la justificación donde se exponen los motivos del desarrollo e implementación de este trabajo.

1.1. Objetivo general y específicos

1.1.1. Objetivo general

Diseñar e implementar un sistema digital de control de posición Proporcional-Integral-Derivativo (PID) embebido en motores de corriente directa.

1.1.2. Objetivos específicos

- Diseñar, seleccionar e implementar etapa de potencia.
- Diseñar una interfaz de comunicación entre el PID embebido y el microcontrolador.
- Implementar rutinas de programación para el PID embebido.
- Integrar etapa de potencia y de control.
- Diseñar y manufacturar una placa de circuito impreso para el PID embebido y para la etapa de potencia.

1.2. Aportación tecnológica

Este trabajo consta del desarrollo de un sistema de control PID digital de bajo costo y programación sencilla para motores de corriente directa, el cual puede implementarse en varias aplicaciones como son: máquinas herramienta CNC, robótica, interfaces típicas de PWM para motores, entre otras. Actualmente los sistemas de control utilizados en dichas aplicaciones suelen tener un alto costo debido a los elementos utilizados para obtener una mayor precisión y exactitud en el movimiento de los motores, es por ello que este trabajo representa una alternativa, ofreciendo versatilidad, precisión y un buen rendimiento para dichas aplicaciones.

1.3. Justificación

Un controlador PID tiene varias funciones importantes: proporciona retroalimentación; la cual tiene la capacidad de eliminar errores de estado estacionario a través de la acción integral; puede anticiparse a través de la acción derivada. Los controladores PID son suficientes para muchos problemas de control, sobre todo cuando la dinámica del proceso es sencilla y los requisitos de rendimiento son modestos. Se encuentran en gran cantidad en los procesos de control de muchas industrias, siendo más del 95 % lazos de control PID [3], aunque la mayoría de lazos son realmente controladores PI. Muchas características de implementación de los controladores PID no han salido a luz debido a que se consideran secretos comerciales, lo que hace complicado el poder desarrollar un controlador PID, por lo que el desarrollo de este tema es importante y actual. Además hoy en día prácticamente todos los controladores PID están basados en circuitos integrados especialmente en microprocesadores, lo cual otorga la oportunidad para proporcionar características adicionales como la sintonización automática, la programación de ganancia, y la adaptación continua, además de la eliminación de fuentes de ruido y de los efectos de retardo provocados por los elementos involucrados en el sistema de control. Todo esto en conjunto da la ventaja de poder crear un sistema de control versátil, es decir, un sistema flexible capaz de utilizarse en diversas aplicaciones.

También debe de tomarse en cuenta que el proceso de derivación es una operación matemática no suave como la integración. En general el diseño de controles derivativos

tanto analógicos como digitales implementados puede ser un proceso delicado y requiere un tratamiento especial de la señal.

1.4. Problemática

En los últimos años la implementación de controladores PID en la industria ha ido creciendo debido al buen rendimiento y seguridad que ofrecen en los sistemas. En general, este control puede llevarse a cabo por medio de dispositivos analógicos o de software, físicos o digitales, aunque estos primeros se han dejado de usar debido a la gran cantidad de dispositivos activos que se utilizan y el espacio considerable que pueden tener, salvo los circuitos de interfaz para acondicionar las señales del proceso, estos aun se continúan utilizando. Hoy en día muchos sistemas trabajan mediante microcontroladores, microprocesadores y circuitos integrados para el procesamiento de señales digitales esto ha provocado que surjan en el mercado un sin fin de controladores PID embebidos comerciales, los cuales ofrecen distintas características según el sistema o la aplicación que se esta realizando o se desea realizar. Muchos de estos controladores difieren en las estructuras de la ley de control (forma estándar, serie, paralela), en la parametrización, en la limitación de la ganancia de alta frecuencia (filtrado), y en como se introduce el valor de referencia. Para poder ser capaces de sintonizar el controlador debe conocerse su estructura y la parametrización del algoritmo de control, pero desafortunadamente esta información no suele estar disponible en los manuales de los controladores, y por ello se presentan problemas de mal funcionamiento. Por estas razones el desarrollo de este trabajo ofrece ser una gran alternativa para aplicar un sistema de control PID digital en alguna aplicación que se desee realizar sin tener que comprar un controlador PID de alto costo, ofrece versatilidad y simplicidad en su implementación, ademas de la descripción a detalle del controlador. Cabe destacar que solo es aplicable en motores de corriente directa así que su limitación esta dada a aplicaciones y/o sistemas donde se vea involucrado el uso de estos actuadores y cuyo control sea critico en el proceso.

1.5. Metodología

La metodología utilizada para el desarrollo e implementación del sistema de control PID embebido para motores de corriente directa esta planteada en la investigación

tecnológica. En la fase 1 se planteó el problema a resolver, se definieron los objetivos, se investigó información sobre otros tipos de controladores PID embebidos en la actualidad, ademas de los conceptos básicos necesarios para comprender los sistemas de control y por último se determinaron las variables de la investigación. En la fase 2 se llevó a cabo la investigación a fondo del controlador LM629, conociendo a detalle cada una de las características que posee, su implementación, su modo de programación, y la forma en la que se comunica por medio de su bus de datos, también se investigó las características individuales de cada uno de los demás elementos a utilizar en cada una de las partes del sistema de control ,en este caso el microcontrolador Arduino Mega ADK, el puente H LMD18201T, el encoder del motor y el motor mismo, para conocer a detalle sus propiedades y sus limitaciones, para tener un amplio panorama y ver posibles sustitutos de esos componentes en dado caso que no cumplieran con las especificaciones requeridas por el sistema. Se realizó el primer diseño del circuito de control y de la etapa de potencia, y se buscó un IDE para para programar la codificación de la interfaz de comunicación de datos y comandos en el microcontrolador, una vez teniendo toda esta información se pasó a la fase 3. En esta fase se implementó el diseño del circuito de control y de la etapa de potencia en un protoboard para observar y verificar su funcionamiento en general, tanto con la interfaz de comunicación entre el hyperterminal-microcontrolador-LM629, como el control de movimiento del motor. Además se realizaron y analizaron los cálculos pertinentes para programar la trayectoria trapezoidal y comprobar que el giro del motor cumplía los parámetros establecidos en dicho calculo. Esta trayectoria se programó en modo posición y se hicieron distintas pruebas para verificar si su paro era en la posición correspondiente, si la aceleración era la adecuada y su alcanzaba la velocidad máxima establecida. En la última fase se compiló toda la información recabada durante las pruebas, y se hicieron anotaciones sobre recomendaciones para una mejor implementación del control y del armado del circuito, ya que se presentaron problemas a lo largo del desarrollo del tema.

Capítulo 2

Fundamentos Teóricos

Este capítulo abarca los conceptos básicos de control, la representación y los tipos de sistemas de control existentes, el concepto y modelado de un motor eléctrico, en este caso especial los motores de corriente directa, así como las acciones de control y los distintos esquemas que se manejan en los controladores PID. El lector más avezado en el tema puede remitirse a la sección 2.6 para la descripción del problema.

2.1. Sistemas de control

Para poder definir un sistema de control, primeramente debe analizarse por separado cada palabra y comprender el concepto de ellas.

Hoy en día la palabra sistema posee una cantidad variable de conceptos, siendo alguno de ellos los siguientes:

1. Un sistema es un conjunto de cosas que relacionadas entre sí ordenadamente contribuyen a determinado objeto.[1]

2. Un sistema es una combinación de componentes que actúan juntos y realizan un objetivo determinado [2].

Cabe mencionar que un sistema no esta asociadamente limitado a fenómenos dinámicos (físicos, biológicos, etc.), sino que, también puede aplicarse a fenómenos abstractos,

[1]Real Academia Española. http://dle.rae.es/?id=Y2AFX5s

como los que hay en la economía.

Por otro lado, la palabra control es comúnmente asociada a conceptos como regular, dirigir o administrar. Entonces uniendo estas dos palabras podemos determinar que un sistema de control es una interconexión o una serie de elementos en conjunto que conforman una configuración del sistema, que mediante la regulación, administración, y dirección de variables de control, nos proporciona una respuesta deseada del mismo sistema.

Los sistemas de control se nos presentan a diario en nuestra vida cotidiana, ya que hay muchos objetivos que deben cumplirse, por ejemplo, en casa, cuando se requiere controlar la temperatura de la refrigeración para tener un ambiente cómodo. En el transporte, para controlar un automóvil de un lugar a otro de forma seguro. En la industria es en donde mas se presentan los sistemas de control, ya sea en lineas de ensamblaje, control de calidad, control de máquinas herramientas, sistemas de armas, control por computadora, por mencionar algunos.

Un sistema de control se puede decir que es ideal si cumple con los siguientes objetivos:

1. Ser estable y robusto a perturbaciones y errores en el modelo.
2. Ser eficiente en un rango establecido.
3. Ser sencillo de implementar y operar.

2.1.1. Representación de un sistema de control

Generalmente un sistema de control se representa en diagrama de bloques (ver Figura 2.1), el cual es una representación gráfica y simplificada de la interacción entre la entrada y la salida de un sistema físico, que facilita la comprensión funcional entre los distintos componentes del sistema de control.

Un sistema de control suele estar conformado por los siguientes elementos básicos:

1. Objetivos de control o entradas. Se pueden conocer también como variables manipuladas, u. Son estímulos o excitaciones aplicadas a un sistema de control, usualmente desde una fuente externa de energía, para producir una respuesta especifica del sistema de control [15].

2. Componentes del sistema de control. Se pueden conocer también como elementos del sistema. Estos son el controlador, la planta o proceso, amplificadores, transductores, etc.

3. Resultados o salidas. Se pueden conocer también como variables controladas, y. Son las respuestas reales que se obtienen de un sistema de control. Pueden ser o no igual a la respuesta implícita especificada por la entrada [15].

Figura 2.1: Componentes básicos de un sistema de control.

Usualmente el rectángulo que representa el bloque tiene la descripción o el nombre del elemento, o símbolo de la operación matemática que se realizará sobre la señal de entrada para obtener la señal de salida. Las flechas indican la dirección del flujo de la señal.

Las operaciones de suma y resta se representan con un pequeño círculo, llamado punto de suma, y llevan el signo apropiado más o menos, respecto a las flechas que entran al círculo. Su salida es la operación algebraica de las señales de entrada.

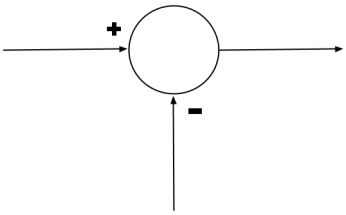


Figura 2.2: Representación de un punto de suma.

También existe el punto de toma, el cual hace que una señal o variable sea una entrada a más de un bloque punto de suma, permitiendo que la señal pueda tomar distintos trayectos sin sufrir cambios.

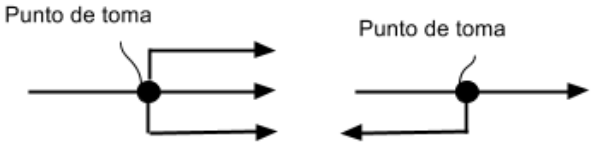


Figura 2.3: Representación de un punto de toma.

2.1.2. Tipos de sistemas de control

Los sistemas de control están divididos por múltiples criterios, unos de ellos son los siguientes:

1. Estrategia de control

    a) Sistemas de control en lazo abierto

    b) Sistemas de control en lazo cerrado

2. Método de diseño

    a) Sistemas de control continuos en el tiempo (analógicos)

Pero antes de profundizar en los sistemas de control, se deben conocer ciertos conceptos básicos.

Planta. Es cualquier objeto físico (sistema, subsistema) que se va a controlar, por ejemplo, un sistema de refrigeración o una lavadora [2].

Proceso. Es cualquier operación que se va a controlar, por ejemplo, un proceso químico o un proceso biológico [2].

Variable controlada o salida controlada. Es la variable de salida de la planta en la cual su valor o condición se mide y controla [15].

Señal de control o variable manipulada. Es la variable donde su valor o condición es modificada por los elementos anticipativos de control (controladores, compensadores y/o amplificadores) para afectar al valor de la variable controlada [15].

Controlador. Elemento de control que determina el error y determina que tipo de acción tomar [15].

Entrada de referencia o señal de referencia. Es un señal adicional aplicada al sistema de control con retroalimentación, generalmente en el primer punto de suma. Suele representar el comportamiento deseado (ideal) de la salida de la planta [15].

Señal actuante (error). Es la entrada de referencia más o menos la señal de retroalimentación. La acción de control se genera por esta señal en un sistema de control en lazo cerrado [15].

Perturbaciones. Es una señal que afecta negativamente al valor de la salida de un sistema. Pueden ser producidas internamente o externamente [2].

Señal de retroalimentación. Es una función de la salida controlada, sumada con la entrada de referencia para obtener el error o señal actuante. Esta señal solo esta presente en sistemas de control en lazo cerrado [15].

2.1.2.1. Sistemas de control en lazo abierto

Un sistema de control en lazo abierto es aquel donde no se toma en cuenta el valor de la variable controlada (salida) y sólo se utiliza la información de la entrada de referencia para efectuar una acción de control. Un ejemplo, en algunos países los semáforos que controlan el tráfico de vehículos, operan con una base de tiempo, mas nunca mide la señal de salida que en este caso sería el estado del tráfico.

El esquema de este sistema de control se presenta en el siguiente diagrama de bloques.

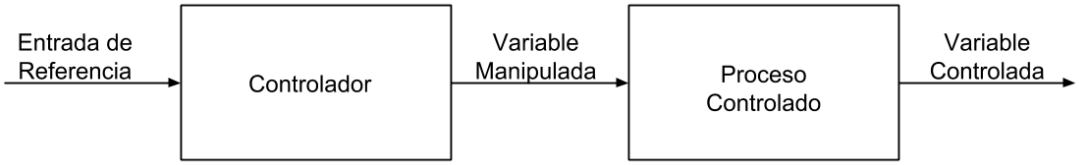


Figura 2.4: Diagrama de bloques de un sistema de control en lazo abierto.

En todos los sistemas de control en lazo abierto, la salida no se compara con la entrada de referencia, así que la precisión del sistema dependerá en gran medida de la calibración que se realice. Un sistema de control en lazo abierto que este expuesto a perturbaciones no trabajará de la manera deseada. Por eso es necesario antes de poner en practica de un sistema de control en lazo abierto, conocer la relación entre la entrada y la salida para tener una exactitud deseada del sistema (calibración), y ver si no hay perturbaciones externas ni internas.

2.1.2.2. Sistemas de control en lazo cerrado

Un sistema de control en lazo cerrado es aquel donde se compara constantemente la variable controlada con la señal de referencia, y utilizando la diferencia de estas dos señales se produce una acción de control que trata de reducir el error existente. Un ejemplo de este sistema de control puede ser un equipo de aire acondicionado, el cual por medio de un sensor registra constantemente la temperatura ambiente y la compara con la temperatura deseada, si no es la misma temperatura este comparador da la señal para que se efectué el control.

Este sistema de control se representan en el siguiente diagrama de bloques, donde se puede apreciar la interconexión de cada uno de los elementos que conforman dicho sistema en lazo cerrado.

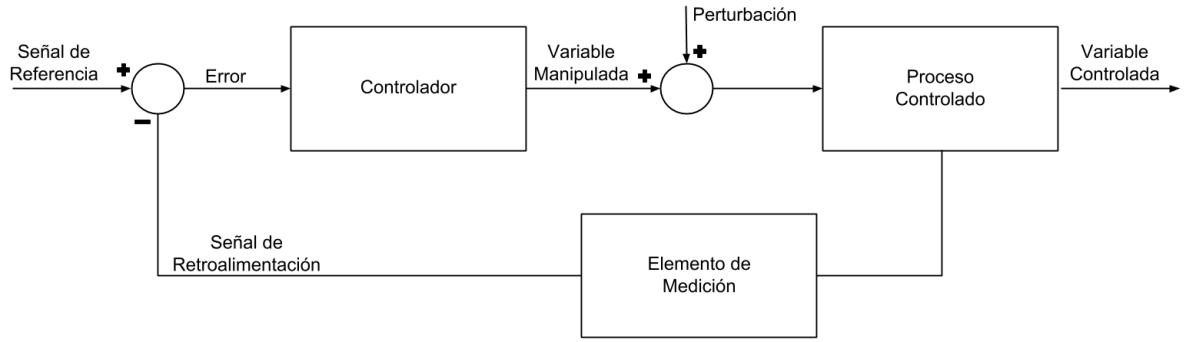


Figura 2.5: Diagrama de bloques de un sistema de control en lazo cerrado.

El uso de sistemas de control en lazo cerrado nos trae algunas ventajas, una de ellas, es que la retroalimentación vuelve la salida del sistema más estable a las perturbaciones externas que pudiesen existir, así como a las perturbaciones internas en los parámetros del sistema, existe un aumento considerable en la exactitud de reproducir fielmente la entrada, aumenta el ancho de banda y principalmente nos ayuda a reducir el error del sistema.

2.1.2.3. Sistemas de control continuos en el tiempo (analógicos)

Un sistema de control continuo en el tiempo es aquel que contiene o procesa unicamente variables o señales continuas y en función de un tiempo continuo (analógico). Estos sistemas de control son representados mediante ecuaciones diferenciales. En particular el sistema que analizaremos es un motor de corriente directa que se describe con ecuaciones diferenciales ordinarias.

2.2. Motores eléctricos

Un motor es una máquina eléctrica rotatoria que transforma en energía mecánica la energía eléctrica por medio de la generación de campos magnéticos entre sus bobinas

por el paso de corriente alterna o corriente directa, según sea el caso. En este caso se hablará unicamente de los motores de corriente directa.

2.2.1. Motor de corriente directa

Un motor de corriente directa es prácticamente un transductor que convierte la energía eléctrica en energía mecánica, como se mencionó anteriormente. El par o el torque del eje del motor es generalmente proporcional al flujo en el campo y a la corriente en la armadura. Este tipo de motor se utiliza en una gran gama de aplicaciones debido a la facilidad y flexibilidad con la que se puede controlar su velocidad, ya sea variando la corriente o el flujo del campo, variando el voltaje del inducido, y controlando la resistencia del rotor. Una característica importante de este motor es el par-velocidad, la cual puede variar suministrándole distintos valores de voltaje pudiendo así adaptar el motor para casi cualquier utilidad que se desee. Los motores de corriente directa pueden dar más de cinco veces el par nominal (siempre y cuando lo permita la fuente de alimentación), al contrario de los motores de corriente alterna, que tienden a detenerse al tener una carga mayor a la que pueden soportar. Otra característica que poseen es que pueden realizar su trabajo en sentido contrario sin tener que conmutar la fuente de alimentación. A continuación se explicarán los siguientes tipos de motores de corriente directa, el motor de corriente directa de imán permanente y el servomotor, los cuales se han vuelto muy populares en el uso de servosistemas.

2.2.1.1. Motor de corriente directa de imán permanente

Este tipo de motor como su nombre lo indica es un motor eléctrico que funciona en base a imanes permanentes en sus polos. Este motor en ciertas aplicaciones presenta varias ventajas respecto al motor de corriente directa en derivación o de campo devanado. Una de estas ventajas es que no requiere de un circuito externo para alimentarlo de energía eléctrica, además no tiene las mismas pérdidas por tener cobre en el circuito de campo por lo que es más eficiente y tiene un rápido enfriamiento. Debido a que no necesita devanado de campo, este tipo de motor puede ser de un tamaño relativamente pequeño. Este tipo de motor tiene su flujo del campo fijo, por lo que no es posible variar su velocidad controlando la corriente, solamente puede controlarse por voltaje y por la resistencia del rotor [16].

2.2.1.2. Motor de corriente directa sin escobillas

Este tipo de motor posee una armadura fija o estacionaria y una estructura rotatoria del campo, colocados de forma opuesta a como normalmente están colocados en un motor convencional de corriente directa. Tiene imanes permanentes que suministran el flujo magnético para el campo. Debido a su construcción disipa más rápido el calor y reduce la inercia del rotor [16].

2.2.1.3. Servomotor de corriente directa

Un servomotor es un motor de alto rendimiento el cual generalmente se usa en máquinas herramientas controladas numéricamente, en impresoras, entre otras aplicaciones donde se necesita un arranque y un paro con rapidez y exactitud. Este tipo de motor suelen tener armaduras de baja inercia por lo que responden con rapidez a los cambios en el voltaje de excitación, también son de peso ligero y poseen una inductancia muy baja de la armadura, por lo que puede dar una baja constante eléctrica de tiempo que beneficia aún más la respuesta del motor a las señales de comando[16].

2.2.2. Modelo matemático de un motor de corriente directa

El modelo matemático de un sistema dinámico, se interpreta mediante ecuaciones que describen el comportamiento real considerando todos los factores y variables físicas que intervienen y pueden afectarlo. Para obtener el modelo matemático de un motor de corriente directa de imán permanente, debemos plantear las ecuaciones físicas del sistema, una ecuación mecánica y una ecuación eléctrica. Estas ecuaciones están relacionadas entre si y se fundamentan en las leyes de Euler y Kirchhoff, respectivamente.

Para realizar un análisis más extenso del modelo matemático del motor de corriente eléctrica a continuación se muestra un esquema del circuito equivalente del motor de corriente directa donde se pueden apreciar todos sus componentes eléctricos como mecánicos.

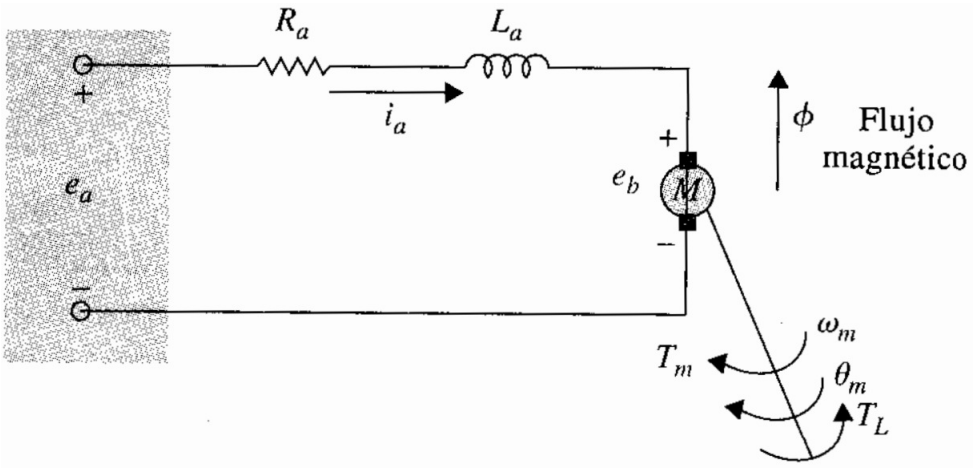


Figura 2.6: Esquema del circuito electro-mecánico de un motor de corriente directa [1].

En la imagen se puede apreciar que la armadura esta representada como un circuito con una resistencia R_a conectada en serie con una inductancia L_a, y a una fuente de voltaje e_b que representa la fuerza electromotriz (FEM) en la armadura cuando el rotor gira. Las variables representadas en el esquema se definen de la siguiente manera:

$i_a(t)$ = corriente de armadura	$T_m(t)$ = par del motor
R_a = resistencia de armadura	$\theta_m(t)$ = desplazamiento del rotor
L_a = inductancia de la armadura	$T_L(t)$ = par de carga
$e_a(t)$ = voltaje aplicado	K_i = constante del par
$e_b(t)$ = fuerza contraelectromotriz	K_b = constante de la fuerza contraelectromotriz
ϕ = flujo magnético en el entre hierro	J_m = momento de inercia del rotor
$\omega_m(t)$ = velocidad angular del rotor	B_m = coeficiente de fricción viscosa (entre el rotor y estator)

Tabla 2.1: Variables del circuito electro-mecánico de un motor de corriente directa [1].

Nota: Los valores de K_i y K_b representan la constante mecánica y la constante eléctrica respectivamente, esto considerando un flujo magnético constante.

Observando el esquema anterior (figura 2.6), se puede apreciar que el control del motor de corriente directa se lleva a acabo por medio de las terminales de la armadura en la forma del voltaje aplicado $e_a(t)$. Para que el análisis sea lineal debe suponerse que

el par desarrollado por el motor es proporcional al flujo magnético en el entre hierro y a la corriente de la armadura. Por lo tanto:

$$T_m(t) = K_m(t)\phi i_a(t) \tag{2.1}$$

Como el flujo magnético en el entre hierro es constante, la ecuación (2.1) se escribe como:

$$T_m(t) = K_i i_a(t) \tag{2.2}$$

en donde K_i es la constante del par en N-m/A, lb-pie/A, u oz-plg/A.

Al aplicar el voltaje de entrada de control $e_a(t)$, las ecuaciones de causa y efecto para el esquema electro-mecánico del motor de la figura 2.6 son:

$$\frac{di_a(t)}{dt} = \frac{1}{L_a}e_a(t) - \frac{R_a}{L_a}i_a(t) - \frac{1}{L_a}e_b(t) \tag{2.3}$$

$$T_m(t) = K_i i_a(t) \tag{2.4}$$

$$e_b(t) = K_b\frac{d\theta_m(t)}{dt} = K_b\omega_m(t) \tag{2.5}$$

$$\frac{d^2\theta_m(t)}{dt^2} = \frac{1}{J_m}T_m(t) - \frac{1}{J_m}T_L(t) - \frac{B_m}{J_m}\frac{d\theta_m(t)}{dt} \tag{2.6}$$

en donde $T_L(t)$ representa el par de carga debido a la fricción tal como la fricción de Coulomb.

La ecuación (2.3) considera que di_a/dt es el efecto inmediato de aplicar el voltaje $e_a(t)$, entonces en la ecuación (2.4), $i_a(t)$ produce el par $T_m(t)$, la ecuación (2.5) define la fuerza contraelectromotriz, y finalmente, en la ecuación (2.6), el par $T_m(t)$ produce la velocidad angular $\omega_m(t)$ y el desplazamiento $\theta_m(t)$.

En base a las ecuaciones anteriores y después de un proceso algebraico y de la aplicación de la transformada de Laplace, se obtiene la función de transferencia del

motor de corriente directa, representada de la siguiente forma:

$$\frac{\Theta_m(s)}{E_a(s)} = \frac{K_i}{L_a J_m s^3 + (R_a J_m + B_m L_a)s^2 + (K_b K_i + R_a B_m)s} \tag{2.7}$$

en donde $T_L(t)$ se igualó a cero.

La figura 2.7 muestra el sistema de un motor de corriente directa representado en diagrama de bloques, el cual nos ayuda a tener una mejor comprensión de la relación de las funciones de transferencia de cada bloque del sistema. Observando la ecuación (2.7) se puede definir que la función de transferencia $\Theta_m(s)/E_a(s)$ indica que el motor de corriente directa es esencialmente un dispositivo integrador entre estas dos variables, esto es, su salida se incrementará linealmente con el tiempo.

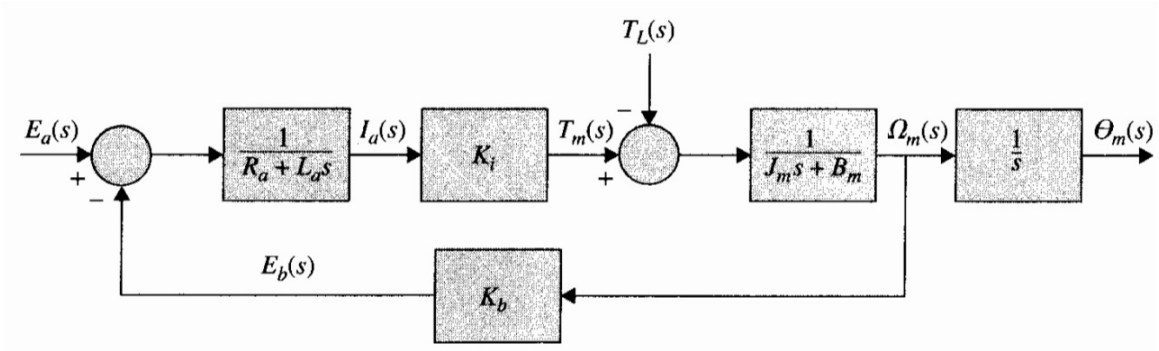


Figura 2.7: Diagrama de bloques de un sistema de motor de corriente directa [1].

A pesar de que el motor de corriente directa es por sí mismo un sistema en lazo abierto, en el diagrama de bloques de la figura 2.7 se puede apreciar que el motor tiene "implícitamente" un lazo retroalimentado provocado por la fuerza contraelectromotriz, esta fuerza físicamente representa la retroalimentación de la señal que es proporcional al negativo de la velocidad del motor. En la ecuación (2.7), la fuerza constante de la fuerza contraelectromotriz K_b representa un término que se añade a la resistencia R_a y al coeficiente de fricción viscosa B_m. Por tanto, la fuerza contraelectromotriz es equivalente a una "fricción eléctrica" que tiene a mejorar la estabilidad del motor, y en general, la estabilidad del sistema.

2.3. Acciones de control PID

Un controlador PID es un mecanismo de control en lazo cerrado, muy usado en la industria para el control de sistemas. Este controlador PID realiza la acción de comprar el valor real de la salida de un sistema con la entrada de referencia (el valor deseado), determina el error existente y produce una señal de control la cual reduce este error a cero o a un valor muy pequeño cercano a este, para ajustar la entrada del sistema. Las formas en las que puede el controlador PID producir la señal de control se les conoce como acciones de control. En la familia de los controladores PID, existen tres acciones de control: proporcional (P), integral (I) y derivativa (D), y se derivan los controladores P,PI,PD y PID, los cuales se explican a continuación.

2.3.1. Acción de control proporcional

Un controlador con acción proporcional, la relación entre la salida del controlador $u(t)$ y la señal de error $e(t)$ es:

$$u(t) = K_p e(t) \tag{2.8}$$

o bien, en cantidades transformadas por el método de Laplace:

$$\frac{U(s)}{E(s)} = K_p \tag{2.9}$$

donde K_p se considera la ganancia proporcional.

En esencia un controlador proporcional es un amplificador con una ganancia ajustable, ya que su salida es un múltiplo del porcentaje del cambio en la medición (error), es decir, es proporcional. Este controlador puede controlar cualquier planta estable, pero su desempeño es muy limitado y tiene error en estado estacionario (off-set). En la figura 2.8 se muestra un diagrama de bloques del controlador.

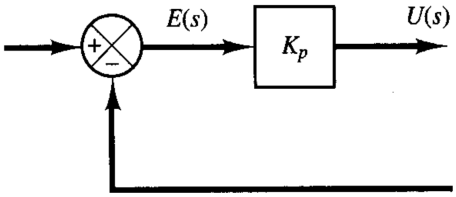


Figura 2.8: Diagrama de bloques de un controlador proporcional [2].

2.3.2. Acción de control integral

Un controlador con acción integral, el valor de la salida del controlador $u(t)$ se cambia a una razón proporcional a la señal de error $e(t)$. Es decir,

$$\frac{du(t)}{dt} = K_i e(t) \tag{2.10}$$

o bien

$$u(t) = K_i \int_0^t e(t)dt \tag{2.11}$$

en donde K_i es una constante ajustable. La función de transferencia del controlador integral es

$$\frac{U(s)}{E(s)} = \frac{K_i}{s} \tag{2.12}$$

Este controlador da una salida proporcional al error acumulado, lo que implica un control lento. Si se duplica el valor de $e(t)$, el valor de $u(t)$ varía dos veces mas rápido. La señal de control $u(t)$ tiene un valor diferente de cero cuando el valor del error $e(t)$ es cero. Por lo tanto esta acción de control dada una referencia constante, o perturbaciones, elimina el error en estado estacionario, provocado por el control proporcional. En ocasiones, esta acción de control es llamada control de reajuste (reset). En la figura 2.9 se muestra un diagrama de bloques del controlador.

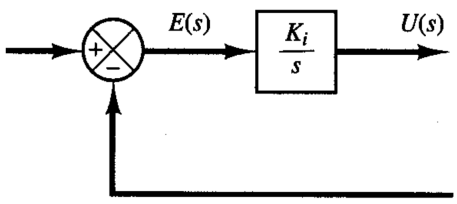


Figura 2.9: Diagrama de bloques de un controlador integral [2].

2.3.3. Acción de control proporcional-integral

Un controlador proporcional-integral (PI) se define mediante

$$u(t) = K_p e(t) + \frac{K_p}{T_i} \int_0^t e(t)dt \tag{2.13}$$

o la función de transferencia del controlador es

$$\frac{U(s)}{E(s)} = K_p \left(1 + \frac{1}{T_i s}\right) \tag{2.14}$$

en donde $K_p$es la ganancia proporcional y T_i se denomina tiempo integral. K_p y T_i son ajustables. El tiempo integral ajusta la acción de control integral, mientras que un cambio en el valor de K_p afecta las partes integral y proporcional de la acción de control. Con una acción de control proporcional, es necesario que exista error alguno para tener una acción de control distinta de cero. Con la acción de control integral, un error pequeño positivo siempre dará una acción de control creciente, y un error negativo una acción de control decreciente. Esta aseveración demuestra que el error en estado estacionario siempre será cero. En la figura 2.10 se muestra un diagrama de bloques del controlador.

Esta acción de control es muy utilizada en la industria, ya que es muy adecuado para procesos donde la dinámica es esencialmente de primer orden.

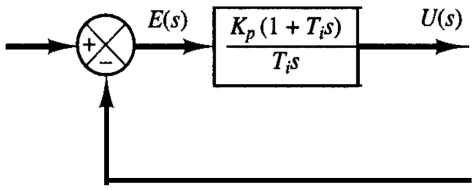


Figura 2.10: Diagrama de bloques de un controlador PI [2].

2.3.4. Acción de control proporcional-derivativa

Un controlador proporcional derivativa (PD) se define mediante

$$u(t) = K_p e(t) + K_p T_d \frac{de(t)}{dt} \tag{2.15}$$

y la función de transferencia es

$$\frac{U(s)}{E(s)} = K_p(1 + T_d s) \tag{2.16}$$

en donde K_p es la ganancia proporcional y T_d es una constante denominada tiempo derivativo. K_p y T_d son ajustables. La acción de control derivativa, también conocida como control de velocidad, ocurre donde la magnitud de la salida del controlador es proporcional a la velocidad de cambio de la señal de error. El tiempo derivativo T_d es el intervalo de tiempo durante el cual la acción de la velocidad hace avanzar el efecto de la acción de control proporcional. La acción de control derivativa no afecta en forma directa al error en estado estacionario, añade amortiguamiento al sistema, y, por tanto, permite un valor más grande que la ganancia K_p, lo cual provoca una mejora en la precisión en estado estable. Esta acción como es predictiva hace mas rápida la acción de control, pero tiene la desventaja de que amplifica las señales de ruido y puede provocar saturación en el actuador. Esta acción nunca debe utilizarse por sí sola, ya que solo es eficaz durante períodos transitorios. En la figura 2.11 se muestra un diagrama de bloques del controlador.

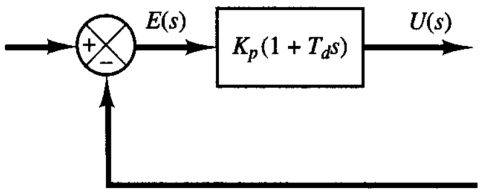


Figura 2.11: Diagrama de bloques de un controlador PD [2].

2.3.5. Acción de control proporcional-integral-derivativa

Un controlador proporcional-integral-derivativo (PID), como su nombre lo indica, es la combinación de las tres acciones de control existentes, la acción de control proporcional, la acción de control integral y la acción de control derivativa. Esta acción combinada tiene las ventajas de cada una de las tres acciones de control. La ecuación de un controlador con esta acción combinada se obtiene mediante

$$u(t) = K_p e(t) + \frac{K_p}{T_i} \int_0^t e(t)dt + K_p T_d \frac{de(t)}{dt} \tag{2.17}$$

o la función de transferencia es

$$\frac{U(s)}{E(s)} = K_p \left(1 + \frac{1}{T_i s} + T_d s\right) \tag{2.18}$$

en donde K_p es la ganancia proporcional, T_i es el tiempo integral y T_d es el tiempo derivativo. En la figura 2.12 se muestra un diagrama de bloques del controlador.

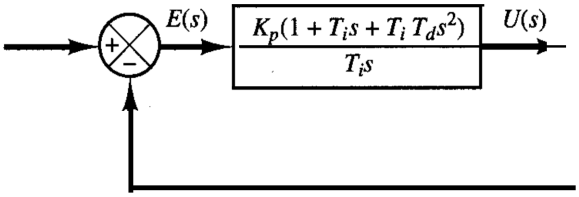


Figura 2.12: Diagrama de bloques de un controlador PID [2].

2.4. Sintonización

Hoy en día mas de la mitad de los controladores industriales que se usan utilizan esquemas de control PID o PID modificado. Hay controladores PID analógicos, prin-

cipalmente, hidráulicos, neumáticos, electrónicos o combinados. Y en la actualidad, muchos de estos están transformándose en formas digitales mediante el uso de microprocesadores. Casi todos los tipos de controladores PID se ajustan en el sitio donde se van a implementar, y existen diversos métodos diferentes de reglas de sintonización, que permiten llevar a cabo una sintonización delicada y fina de los controladores PID en el sitio. De igual manera, se han desarrollado métodos automáticos de sintonización y algunos de los controladores PID tienen la capacidad de sintonización en línea.

Cuando se puede obtener el modelo matemático de la planta, se pueden aplicar técnicas de diseño para determinar los parámetros del controlador que cumpla con las especificaciones en estado transitorio, y en estado estable del sistema en lazo cerrado. Pero cuando la planta es muy complicada y por ende no se puede obtener tan fácilmente el modelo matemático, y mucho menos hacer un enfoque analítico para el diseño del controlador PID, se debe recurrir a enfoques experimentales para la sintonización de los controladores PID. Entonces al proceso de ajustar los parámetros (ganancias) que cumplan con los requisitos de desempeño del controlador, se conoce como sintonización. A continuación se presentan algunas reglas de sintonización convenientes cuando no se conoce el modelo matemático.

2.4.1. Regla de Ziegler-Nichols

Existen dos métodos denominados reglas de sintonización de Ziegler-Nichols [2]. En los dos se busca obtener un 25 % de sobrepaso máximo en la respuesta escalón.

Primer método, método basado en la curva reacción. En este método, la respuesta de la planta a una entrada escalón unitario se obtiene de forma experimental. Si la planta no posee integrados ni polos dominantes complejos conjugados, la curva de respuesta escalón unitario puede tener forma de "S", como se puede apreciar en la figura 2.13. Si no se obtiene una curva con forma de "S" en la respuesta, este método no es pertinente. Estas curvas de respuesta escalón se generan experimentalmente o a partir de una simulación dinámica de la planta.

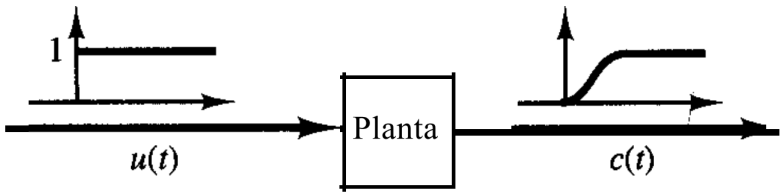


Figura 2.13: Respuesta escalón unitario de una planta [2].

La curva con forma de "S" se caracteriza por dos parámetros: el tiempo de retardo L y la constante de tiempo T. Estos dos parámetros se determinan dibujando una recta tangente en el punto de inflexión de la curva con forma de "S" y determinando las intersecciones de esta tangente con el eje del tiempo y la linea $c(t) = K$, como se aprecia en la figura 2.14. La función se aproxima mediante un sistema de primer orden con un retardo de transporte del modo siguiente:

$$\frac{C(s)}{U(s)} = \frac{Ke^{-Ls}}{Ts+1} \tag{2.19}$$

Esta regla sugiere establecer los valores de K_p,T_i y $T_d$de acuerdo con la fórmula de la tabla 2.2.

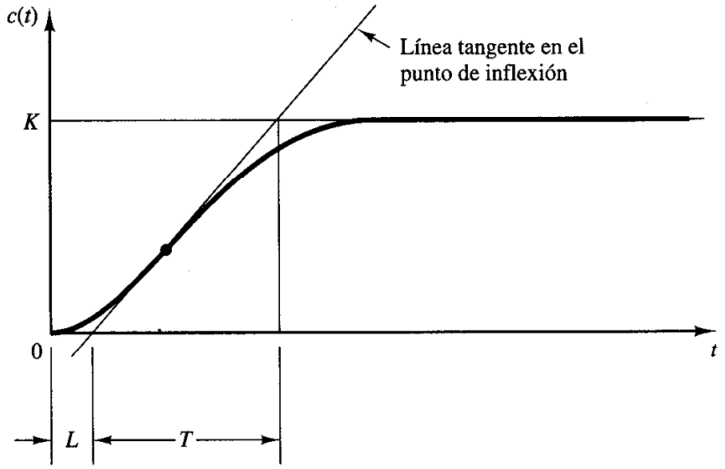


Figura 2.14: Curva de respuesta con forma de "S" [2].

Tipo de controlador	K_p	T_i	T_d
P	$\frac{T}{L}$	∞	0
PI	$0.9\frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{T}{L}$	$2L$	$0.5L$

Tabla 2.2: Regla de sintonización de Ziegler-Nichols basada en la respuesta escalón de la planta [2].

Sintonizando el controlador PID por este método se obtiene:

$$G_c(s) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right) \tag{2.20}$$

$$= 1.2\frac{T}{L}\left(1 + \frac{1}{2Ls} + 0.5Ls\right)$$

$$= 0.6T\frac{\left(s + \frac{1}{L}\right)^2}{s}$$

Por tanto, el controlador PID tiene un polo en el origen y un cero doble en $s = -1/L$.

Segundo método, método de oscilación. En este método primero se establece $T_i = \infty$ y $T_d = 0$. Utilizando solamente la acción de control proporcional, se incrementa K_p de 0 aun valor critico K_{cr} en donde la salida exhiba primero oscilaciones sostenidas. Este método no es aplicable si en la salida no hay oscilaciones sostenidas para cualquier valor que K_p. La ganancia crítica K_{cr} y el periodo P_{cr} correspondiente se determina experimentalmente (ver figura 2.15). Para establecer los valores de los parámetros K_p, T_i y T_d se sugiere utilizar la formula presente en la tabla 2.3.

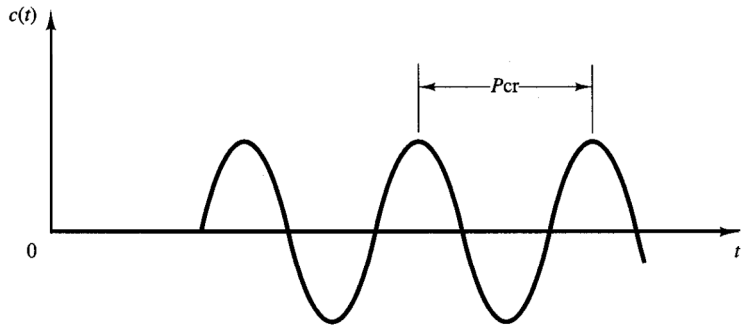


Figura 2.15: Oscilación sostenida con un periodo P_{cr} [2].

Tipo de controlador	K_p	T_i	T_d
P	$0.5K_{cr}$	∞	0
PI	$0.45K_{cr}$	$\frac{1}{12}P_{cr}$	0
PID	$0.6K_{cr}$	$0.5P_{cr}$	$0.125P_{cr}$

Tabla 2.3: Regla de sintonización de Ziegler-Nichols basada en la ganancia crítica K_{cr} y en el periodo crítico P_{cr} [2].

Sintonizar un controlador mediante este método produce lo siguiente

$$G,(s) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right) \tag{2.21}$$

$$= 0.6K_{cr}\left(1 + \frac{1}{0.5P_{cr}s} + 0.125P_{cr}s\right)$$

$$= 0.075K_{cr}P_{cr}\frac{\left(s + \frac{4}{P_{cr}}\right)^2}{s}$$

Por lo tanto, el controlador PID tiene un polo en el origen y cero doble en $s = -4/P_{cr}$.

2.5. Topologías del PID (esquemas de implementación)

El esquema PID dado por la ecuación (2.17) en la sección anterior, muy pocas veces es utilizado en la practica ya que se puede obtener un mejor rendimiento utilizando un esquema PID modificado. También sabemos que la ecuación (2.17) puede ser representada como una función de transferencia dada por la ecuación (2.18). Existe una versión diferente a estas ecuaciones y es muy común su uso en controladores comerciales, este controlador es descrito por la siguiente ecuación

$$G'(s) = K'\left(1 + \frac{1}{T'_i s}\right)(1 + T'_d s) \tag{2.22}$$

Existen dos esquemas para controladores PID, el esquema dado por la ecuación (2.18) conocido como estándar o no interactivo (ver figura 2.16), y el esquema dado por la ecuación (2.22) conocido como serie o interactivo (ver figura 2.17). La razón para esta

nomenclatura es que en el controlador descrito por la ecuación (2.18) la constante de tiempo integral T_i no influye en la parte derivativa, y la constante de tiempo derivativo T_d no influye en la parte integral. Por lo tanto estas partes no interactúan. En el controlador serie, la constante de tiempo derivativo T'_d influye en la parte integral. Por lo tanto la parte esta interactuando.

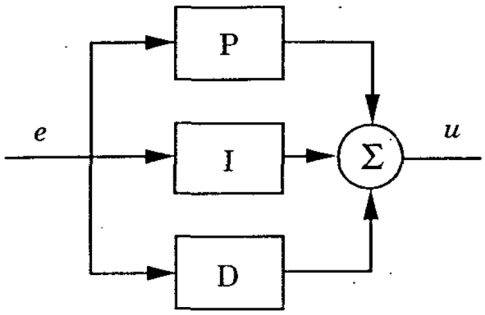


Figura 2.16: Esquema PID estándar o no interactivo [3].

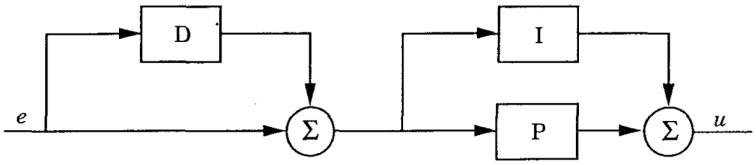


Figura 2.17: Esquema PID serie o interactivo [3].

El controlador serie (2.22) siempre puede ser representado como un controlador estándar (2.18), cuyos coeficientes están dados por

$$K = K' \frac{T'_i + T'_d}{T'_i} \tag{2.23}$$

$$T_i = T'_i + T'_d \tag{2.24}$$

$$T_d = \frac{T'_i T'_d}{T'_i + T'_d} \tag{2.25}$$

Un controlador serie de la forma (2.22) que corresponde a un controlador estándar (2.18) se puede encontrar solo si

$$T_i \geq 4T_d$$

entonces,

$$K' = \frac{K}{2}\left(1 + \sqrt{1 - 4T_d/T_i}\right) \tag{2.26}$$

$$T_i' = \frac{T_i}{2}\left(1 + \sqrt{1 - 4T_d/T_i}\right) \tag{2.27}$$

$$T_d' = \frac{T_i}{2}\left(1 - \sqrt{1 - 4T_d/T_i}\right) \tag{2.28}$$

El controlador serie es más fácil de sintonizar manualmente que un controlador estándar, esta es una de las razones de porque se prefiere este controlador modificado. Ademas históricamente se ha demostrado que desde los controladores neumáticos la construcción de estos en forma serie ha sido más sencilla. Cuando la tecnología de manufactura de estos controladores cambió de neumática a electrónica analógica y, finalmente a digital, se mantuvo la forma serie. Por lo tanto, la forma serie es la más común en los controladores de un solo lazo.

Es importante tener en mente que los diferentes controladores puede tener distintas estructuras. Esto es que si un controlador de un cierto lazo de control es reemplazado por otro tipo de controlador, los parámetros del controlador tienen que ser cambiados. También se debe tener en cuenta que que la forma serie y la forma estándar son diferentes cuando se utilizan las partes I y D del controlador. Si sólo se utiliza un controlador como P, PI o PD, las dos formas son equivalentes. Sin embargo existe otra representación para el controlador PID y esta dada por

$$G''(s) = \kappa + \frac{\kappa_i}{s} + \kappa_d s \tag{2.29}$$

Los parámetros están relacionados con los parámetros de la forma estándar a través de

$$\kappa = K \tag{2.30}$$

$$\kappa_i = \frac{K}{T_i} \tag{2.31}$$

$$\kappa_d = KT_d \tag{2.32}$$

La representación (2.29) es equivalente a la forma estándar, pero los valores de los parámetros son muy diferentes. Esto puede causar grandes dificultades para cualquier persona que no esta consciente de las diferencias, sobretodo si el parámetro $1/\kappa_i$ es llamado tiempo integral y κ_d tiempo derivativo. La forma dada por la ecuación (2.29) es a menudo muy útil en cálculos analíticos porque el parámetro aparece linealmente. La representación también tiene la ventaja de que es posible obtener la acción de control proporcional, integral o derivativa por valores finitos de los parámetros.

Resumiendo la información se tiene que existen tres tipos de formas de controladores PID

- La forma estándar o no interactuante dada por la ecuación (2.18).
- La forma serie o interactuante dada por la ecuación (2.22).
- La forma paralela dada por la ecuación (2.29).

La forma estándar es llamada en ocasiones esquema ISA o esquema ideal. Las acciones de control proporcional, integral y derivativa no interactúan en el dominio del tiempo. Este esquema admite ceros complejos, que es útil en sistemas de control con polos oscilatorios.

La forma serie es también llamada la forma clásica. Esta representación se obtiene naturalmente cuando un controlador es implementado como un dispositivo analógico basado en un sistema de compensación de fuerza neumática. La forma serie tiene una atractiva interpretación en el dominio de la frecuencia porque los ceros corresponden a los valores inversos del tiempo derivativo y del tiempo integral. Todos los ceros de el controlador son reales. No se puede obtener la acción integral o proporcional con valores finitos de los parámetros del controlador. La mayoría de los controladores utilizan esta forma.

La forma paralela es la formas más general, porque la acción proporcional o integral se pueden obtener con parámetros finitos. El controlador puede tener ceros complejos. De esta manera esta forma es la mas flexible. Sin embargo, también es la forma donde los parámetros tienen poca interpretación física.

2.6. Problemáticas de implementación

El efecto de derivación de una señal en términos de simulación e implementación en un controlador PID puede generar una cierta diversidad de problemas que a su vez ocasionan que el sistema sea inestable y no responda correctamente. Es pertinente saber que el proceso de derivación de la señal llevada a cabo por los controladores PID y los sistemas de simulación es poco conocida y no se sabe a detalle que técnica de derivación es utilizada ni como es implementada, tanto en simulación como en la práctica a nivel semiconductor.

2.6.1. Ejemplo de simulación: problemática

Para demostrar los problemas que nos puede ocasionar la acción derivativa de un controlador PID se llevará a cabo un experimento en Simulink de Matlab en donde se compara la acción derivativa en un sistema de control en lazo abierto con un sistema de control en lazo cerrado, con el fin de demostrar que cerrar el lazo en un sistema de control ayuda al control PID a evitar errores en la derivación. A continuación se muestra el esquema en diagramas de bloques de un controlador PID en lazo abierto con una entrada escalón unitario (ver figura 2.18). En este experimento se cambiara el valor de kd para ver el efecto que este parámetro tiene sobre la señal de salida.

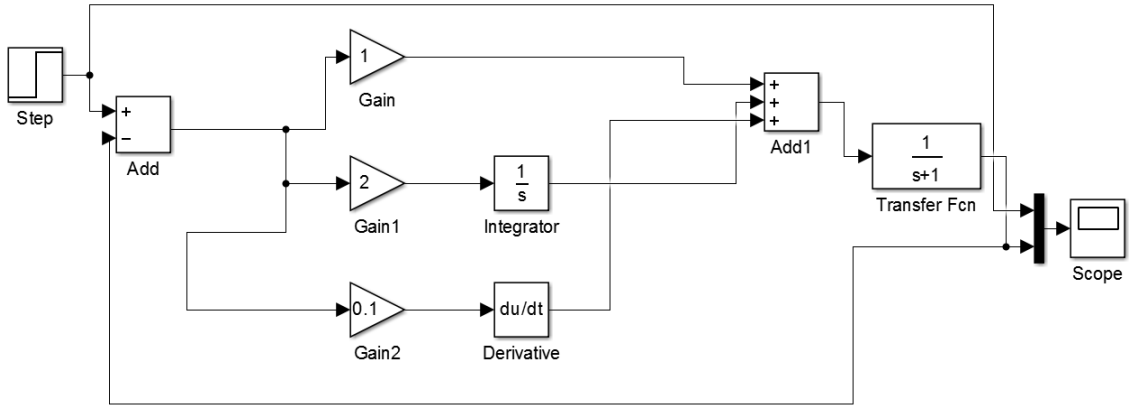


Figura 2.18: Esquema en diagramas de bloques del controlador PID en Simulink Matlab .

Como primer paso a la acción de control derivativa del controlador PID se le dio un valor pequeño de 0.1, y se obtuvo la siguiente señal de salida.

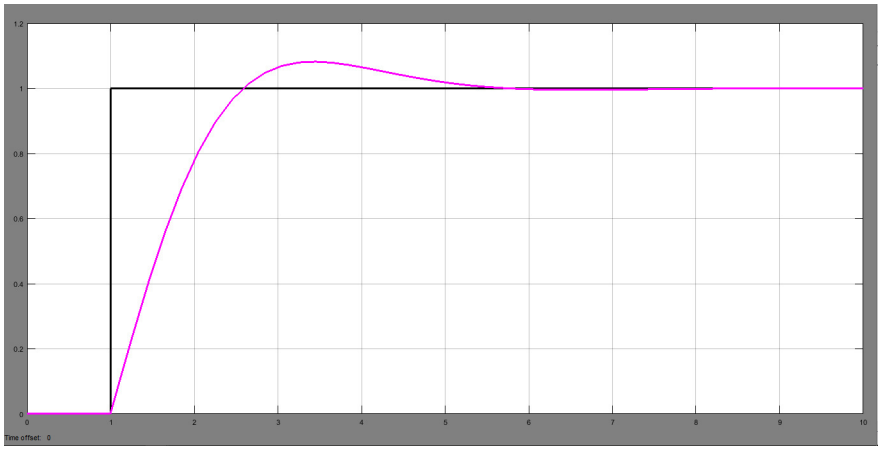


Figura 2.19: Señal de salida con un valor derivativo de 0.1.

Como se puede observar tenemos una señal de salida con poco sobrepaso debido al valor pequeño que se le dio a la acción derivativa. En el siguiente paso se aumentó el valor de 0.1 a 1.3, y se obtuvo la siguiente gráfica.

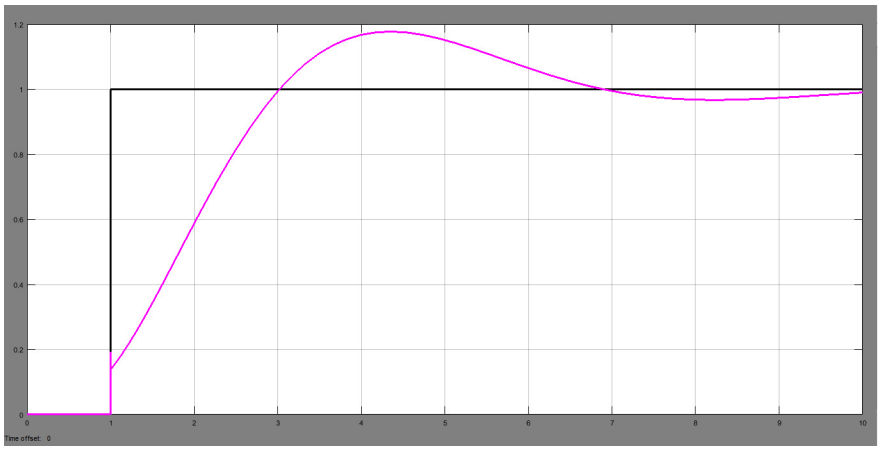


Figura 2.20: Señal de salida con un valor derivativo de 1.3.

En esta gráfica se observa que la velocidad de respuesta disminuyó pero aumento el sobrepaso algo que no es muy conveniente si queremos estabilidad en nuestro sistema. Por último a la acción derivativa se le dio un valor de 2 para ver el comportamiento de la gráfica.

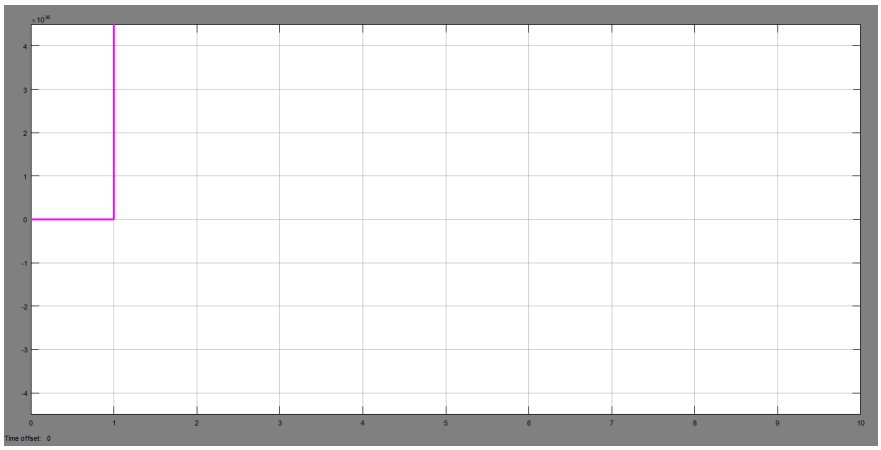


Figura 2.21: Señal de salida con un valor derivativo de 2 .

En esta gráfica se puede observar que la respuesta de la señal de salida prácticamente no existe debido a que el valor derivativo es relativamente grande. Se sabe que la acción derivativa anticipa el comportamiento futuro del error y la respuesta de la componente derivativa es proporcional a la tasa de cambio del error. Es por esto que la acción derivativa previene el sobrepaso y elimina las oscilaciones. En las gráficas se

observa que cuando la ganancia derivativa es muy pequeña existe poco sobrepaso, esto se debe a que la respuesta derivativa es altamente sensible al ruido que pueda existir en la señal de la variable del proceso. Si se tiene una señal de retroalimentación muy ruidosa, la acción derivativa puede provocar inestabilidad en el sistema de control. Por ende si colocamos un valor muy grande y en nuestra señal existe mucho ruido tendremos un sobrepaso muy alto inclusive infinito ya que la tangente en algún punto de la señal de ruido no esta definida.

Para eliminar este problema que ocurre con la acción derivativa, es necesario cerrar el lazo, obteniendo una nueva función de transferencia, la cual se muestra en el siguiente diagrama de bloques de Simulink.

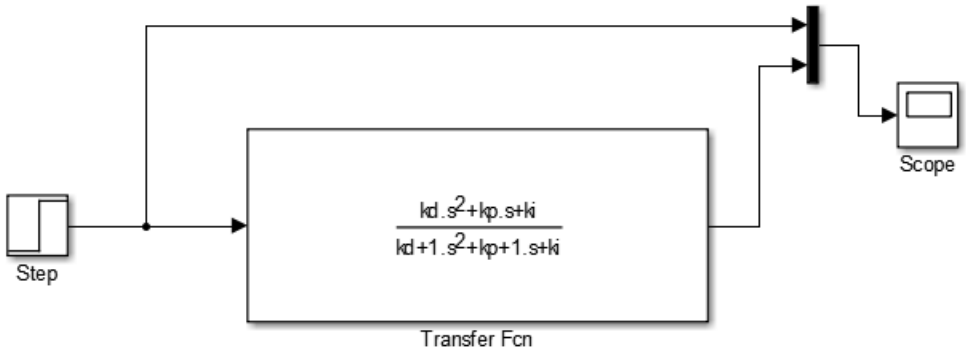


Figura 2.22: Esquema en diagramas de bloques del controlador PID en lazo cerrado.

Para efectos de comprobación de utilizaron los siguientes valores: $K_p = 1$, $K_i = 1$ y $K_d = 2$. Y se obtuvo la siguiente gráfica.

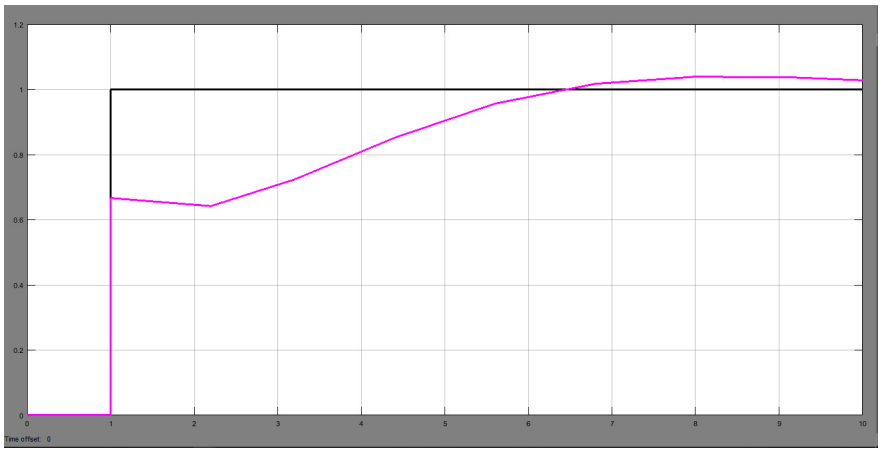


Figura 2.23: Señal de salida con un valor derivativo de 2 en lazo cerrado.

Como se puede observar en comparación con la figura 2.21 cerrar el lazo ayuda a la acción derivativa, ya que siempre la señal de error estará definida y por ende podrá ser derivable, también se puede observar que al cerrar el lazo la señal se vuelve un poco lenta y existe mucho menos sobrepaso, esto puede ayudar al sistema a tener una mejor estabilidad.

2.6.2. Operaciones numéricas de derivación e integración

Para la derivación e integración de una señal existen diversas técnicas matemáticas que nos dan una aproximación numérica cuando es difícil o imposible encontrar una formula matemática para la representación de la señal. A continuación se mencionan algunas de estas técnicas y/o teoremas utilizados en la derivación e integración de señales, capaces de otorgar buenos resultados en la implementación de un controlador PID.

2.6.2.1. Derivación

La derivación numérica es una técnica de análisis numérico para calcular una aproximación a la derivada de una función en un punto utilizando los valores y propiedades de la misma. Para iniciar con la derivación, se tiene que la definición de la derivada de una función $f(x)$ está dada por

$$f'(x_0) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{2.33}$$

siempre y cuando el límite exista. Esto conduce a una fórmula útil para aproximar la derivada en x. El teorema de Taylor dice que si f es dos veces continuamente diferenciable, entonces

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2} f''(c) \tag{2.34}$$

donde c se encuentra entre x y $x+y$. La ecuación 2.34 implica la siguiente fórmula, conocida como **diferencia hacia adelante de dos puntos**.

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{h}{2} f''(c) \tag{2.35}$$

donde c esta entre x y $x+h$. A estas formulas también se les llama aproximación por la derecha de dos puntos y dado que el exponente de h es 1, se llaman también aproximación de primer orden o de orden h.

Existe una técnica más avanzada en la cual es posible desarrollar una fórmula de segundo orden, de acuerdo con el teorema de Taylor, si f es tres veces continuamente diferenciable (se omite demostración). Se obtiene como resultado la siguiente formula de segundo orden [4]

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{h^2}{6} f'''(c) \tag{2.36}$$

donde $x - h < c < x + h$. El término de error es $-\frac{h^2}{6} f'''(c)$. Esta formula es conocida como **diferencia centrada en tres puntos**.

Para obtener más fórmulas de aproximación para derivadas con un orden superior pueden obtenerse de la misma forma, utilizando la expansión de Taylor. Así obtenemos la fórmula de la **diferencia centrada de tres puntos para la segunda derivada**

$$f''(x) = \frac{f(x-h) - 2f(x) + f(x+h)}{h^2} - \frac{h^2}{12} f^{(iv)}(c) \tag{2.37}$$

para alguna c entre $x - h$ y $x + h$. El término de error es $-\frac{h^2}{12} f^{(iv)}(c)$.

2.6.2.2. Integración

Las técnicas de integración numérica se usan cuando f(x) es difícil o imposible de integrar analíticamente, o cuando f(x) esta dada como un conjunto de valores tabulados. Existen varias técnicas, algunas de ellas son las siguientes.

Método del trapecio. Para encontrar el área bajo la curva este método divide una sección con forma trapezoidal, con un valor espaciado de x o un intervalo $[x_0, x_1]$, con base que descansa en el eje x, cerrando así precisamente el área del trapecio que se forma (ver figura 2.24).

La formula del método del trapecio esta dada por

$$\int_{x_0}^{x_1} f(x)dx = \frac{h}{2}(y_0 + y_1) - \frac{h^3}{12}f''(c) \tag{2.38}$$

donde $h = x_1 - x_0$ y c está ente x_0y x_1. El término de error es $-\frac{h^3}{12}f''(c)$.

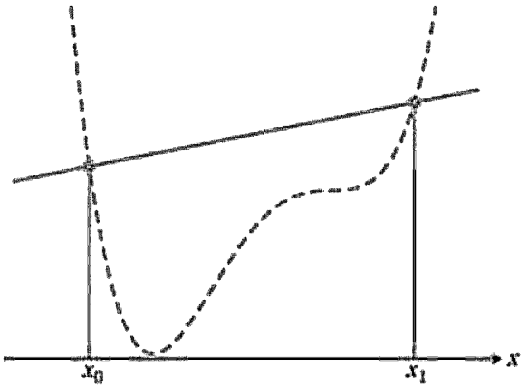


Figura 2.24: Método del trapecio [4].

Si se desea mejorar la exactitud de este método es recomendable dividir el intervalo $[x_0, x_1]$en un conjunto de segmentos y aplicar el método a cada uno de estos segmentos. Después sumar las áreas de los segmentos individuales y así obtener la integral sobre dicho intervalo.

Método de Simpson. Este método utiliza la misma filosofía del método del trapecio, es decir, tener un intervalo dividido en subintervalos o secciones las cuales definirán las áreas bajo la curva de $f(x)$, pero en vez de utilizar un polinomio de primer grado para la aproximación de los subintervalos se utilizan polinomios de segundo grado de la forma $ax_2 + bx + c$ formando un pequeño arco de parábola (ver figura 2.25).

La formula del método de Simpson esta dada por

$$\int_{x_0}^{x_2} f(x)dx = \frac{h}{3}(y_0 + 4y_1 + y_2) - \frac{h^5}{90}f^{(iv)}(c)$$

donde $h = x_2 - x_1 = x_1 - x_0$ y c está entre x_0 y x_2. El término de error es $-\frac{h^5}{90}f^{(iv)}(c)$.

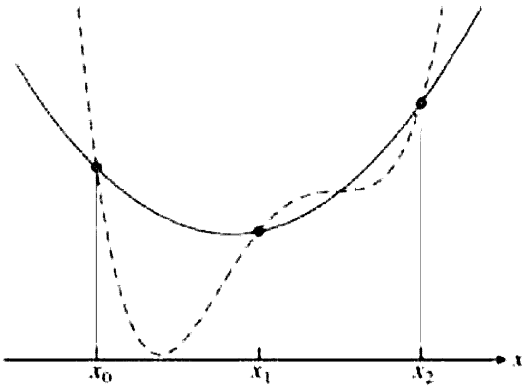


Figura 2.25: Método de Simpson [4].

Estos dos métodos son los mas sencillos de realizar y por ende son muy lentos cuando se quiere obtener cierto nivel de precisión. Existen otros métodos que son mas eficaces y tienen una mayor precisión pero a su vez su desarrollo no es simple y se torna muy complejo, a continuación se enlistan estos métodos:

- Cuadratura Gaussiana
- Integración de Romberg
- Integración de Monte Carlo
- Integración por valor promedio

2.6.3. Implementación básica digital y analógica

Un controlador PID puede implementarse de manera digital y analógica; los controladores PID digitales están implementados con microcontroladores, DSP, FPGA, entre otros, estos a su vez necesitan de conversores analógicos-digital (ADC) y digital-analógicos (DAC) y los controladores PID analógicos están implementados con amplificadores operacionales, resistores y capacitores que en conjunto forman una estructura de filtro que modifica la respuesta en frecuencia del sistema. A continuación se enlistan algunas ventajas y desventajas de los controladores PID tanto analógicos como digitales.

Ventajas	**Desventajas**
Amplio ancho de banda	Desgaste de componentes con el tiempo
Alta resolución	Problemas con la temperatura
Diseño fácil	Sólo diseños simples

Tabla 2.4: Ventajas y desventajas de un controlador PID analógico.

Ventajas	**Desventajas**
Programable	Diseño difícil
Precisión en el comportamiento	Uso de procesador de altos recursos
Algoritmos de diferenciación e integración avanzados	Generación de problemas numéricos
Se puede optimizar	

Tabla 2.5: Ventajas y desventajas de un controlador PID digital.

Los controladores PID digitales pueden implementarse de dos formas:

- Software
  - Computadora
  - Microcontrolador
  - Microprocesador
- Hardware
  - VSLI

- FPGA
- CPLD
- CMOS

Por hardware estos suelen estar diseñados por medio de software de diseño de circuitos integrados, donde en tamaños muy minúsculos suelen representar una gama compleja de circuitos lógicos como el de la figura 2.26.

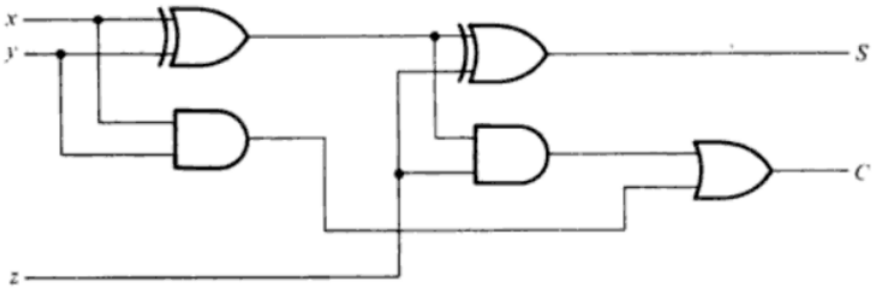


Figura 2.26: Circuito lógico sumador completo [5].

Los controladores PID analógicos son representados mediante amplificadores operacionales mediante sus formas de operación las cuales son modo inversor, modo integrador y modo derivador, que respectivamente representan la acción de control proporcional, integral y derivativa. A continuación se muestran el esquemático de un amplificador derivador y un amplificador integrador, no se profundiza más en este tipo de controladores debido a que no es el objetivo de estudio.

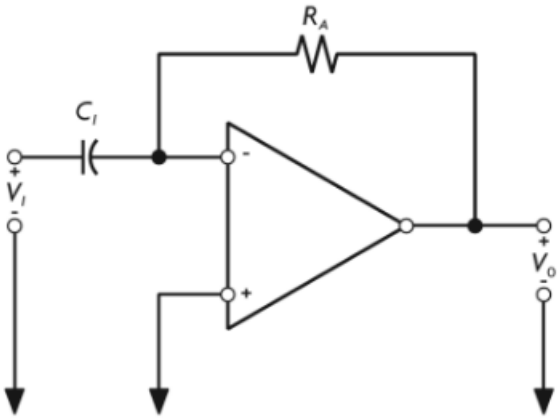


Figura 2.27: Amplificador derivador [6].

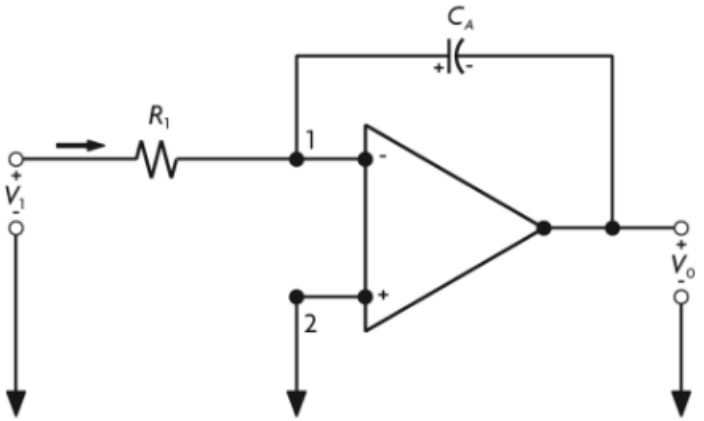


Figura 2.28: Amplificador integrador [6].

2.6.4. Implementación practica a nivel Semiconductor

Hoy en día la mayoría de los controladores PID son implementados digitalmente en circuitos integrados utilizando tecnologías como CMOS o la tecnología VLSI, ambos compatibles con entradas y salidas TTL. Dentro de un circuito integrado se construyen amplificadores operacionales, transistores, resistores ajustables, capacitores, conversores ADC, entre otros componentes, que combinados forman cada una de las partes del controlador PID.

Utilizando la tecnología VLSI y/o CMOS se ha logrado reducir el tamaño físico de los controladores PID, reducir el consumo de energía y por consecuencia la reducción del costo de estos circuitos integrados. A ciencia cierta no se conoce el diseño ni los componentes que se utilizan debido al secreto comercial, donde cada compañía dedicada a la fabricación de controladores PID tienen sus propias especificaciones y diseños. Lo que si se tiene en cuenta es que el cambio de controladores PID a digitales con la ayuda de los microprocesadores han logrado obtener una alta precisión a altas frecuencias, se puede implementar fácilmente a sistemas no lineales y existe un mayor rendimiento en sistemas donde se manejan bastante sensores y controladores.

A continuación se enlistan algunos de los controladores PID a nivel semiconductor existentes hoy en día.

- LM629 de Texas Intruments.
- MAXPOS y EPOS4 de Maxon motor.
- PIC16F684, PIC14C42, PIC18F2331 de Microchip.
- EZ4AXIS de All Motion.

En particular utilizaremos el LM629 por ser un controlador especialmente dedicado al control de movimientos de motores de corriente directa, en el capítulo 3 se profundizará mas a detalle sobre este controlador.

Parte II

Desarrollo y resultados

Capítulo 3

Implementación del sistema

En este capítulo se hace la descripción del diseño y la implementación cada uno de los componentes que conforman al sistema de control, mencionando los criterios de selección y las características propias que presentan.

El sistema de control que se implementó se puede observar en la figura 3.1, y este contendrá los siguientes elementos:

- Computadora, en ella se obtiene y visualiza la transmisión de datos y respuestas que nos otorgue el LM629, por medio del microcontrolador ATmega2560 y la comunicación serial USB que posee el Arduino Mega ADK.
- Arduino Mega ADK, utilizando su microcontrolador ATmega2560 y la interfaz de comunicación serial que posee, se llevará acabo el control del LM629 y además se mostrará la respuesta y transmisión de datos de éste en la computadora.
- LM629, circuito integrado especializado para el control de motores de corriente directa.
- AM26LS32, circuito integrado para el acondicionamiento de las señales del encoder.
- Motor A-max 16, en este caso es nuestro actuador y objetivo de control.
- Encoder MR tipo M, utilizado para retroalimentar el sistema de control por medio de las señales en cuadratura.

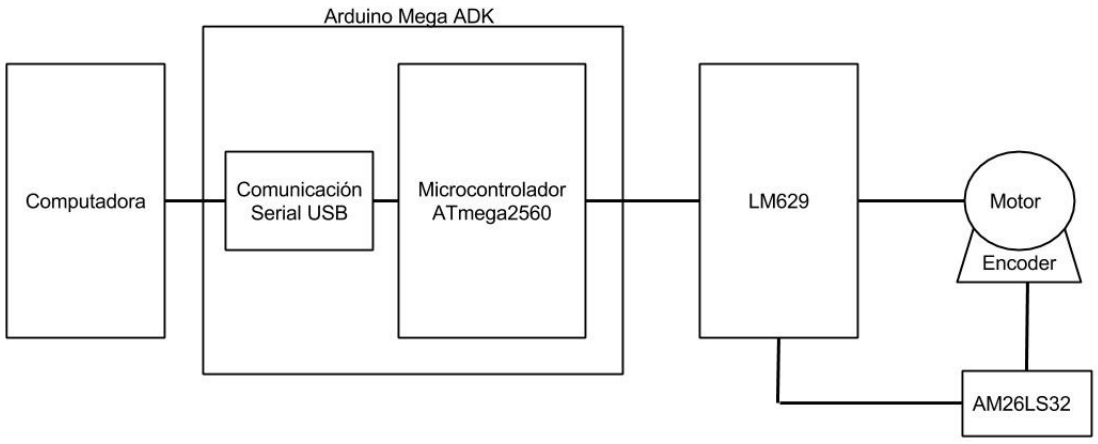


Figura 3.1: Esquema del sistema de control.

A pesar de que el LM629 ya tiene bastante tiempo en el mercado, aproximadamente desde los años 90, es un circuito integrado que aún se mantiene con validez debido a su relativa facilidad de implementación. El uso de este circuito integrado tiene varios beneficios, uno de ellos es que es de muy bajo costo en comparación a otros controladores de movimiento, tiene mejor precisión y exactitud, puede ejecutar tareas en tiempo real a una gran velocidad para obtener un alto rendimiento en el control digital, posee un filtro PID reprogramable en cualquier momento que se desee, y puede implementarse en varias aplicaciones, todas estas características lo hacen viable para el sistema de control que se requiere implementar.

3.1. Controlador de movimientos de precisión LM629

El LM629 es un circuito integrado diseñado para el control de movimientos de precisión para motores de corriente directa y de servomotores de corriente directa sin escobillas, además de otros tipos de servomecanismos que poseen como retroalimentación de posición una señal incremental en cuadratura.

Este controlador puede ejecutar tareas en tiempo real otorgándole un alto rendimiento en el control digital de movimiento. La interfaz de control de este dispositivo es llevado a cabo por comandos de programación de alto nivel, otorgándole al LM629 practicidad e idealidad en diversas aplicaciones.

La utilización del LM629 nos ayuda en la reducción de los componentes necesarios en el diseño en un servosistema, necesitando solamente un motor de corriente directa con encoder incremental y como etapa de potencia un puente H con cambio de giro y regulación PWM. El LM629 posee una señal de salida PWM en magnitud y signo con una resolución de 8 bits, facilitando el uso de un puente H.

Existen dos versiones de este controlador el LM629-6 y el LM629-8, este sufijo indica la velocidad de operación en la que pueden ser utilizados, ya sea a 6 MHz u 8 MHz respectivamente.

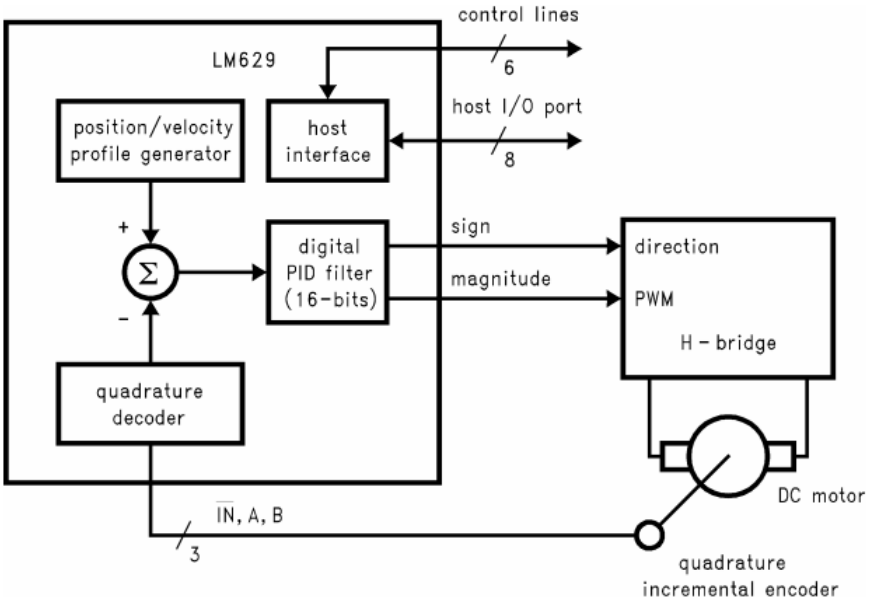


Figura 3.2: Diagrama de bloques típico de un sistema con LM629 [7].

3.1.1. Características funcionales del LM629

El LM629 presenta algunas de las siguientes características:

- 3 registros de 32 bit para la posición, velocidad y aceleración.
- Filtro PID digital programable con coeficiente de 16 bits.

- Intervalo de muestreo derivativo programable.
- Señal de salida PWM con magnitud y signo de 8 bits.
- Generador interno de perfil de velocidad trapezoidal.
- La velocidad, la posición y los parámetros del filtro pueden ser cambiados durante el movimiento.
- Funcionamiento en modo de posición o modo velocidad.
- Interrupciones programables en tiempo real.
- Interfaz paralela con servidor asíncrona de 8 bits.
- Interfaz con encoder incremental en cuadratura y entrada de pulso indexado.
- Empaquetado de 28 pines o 24 pines tipo DIP.

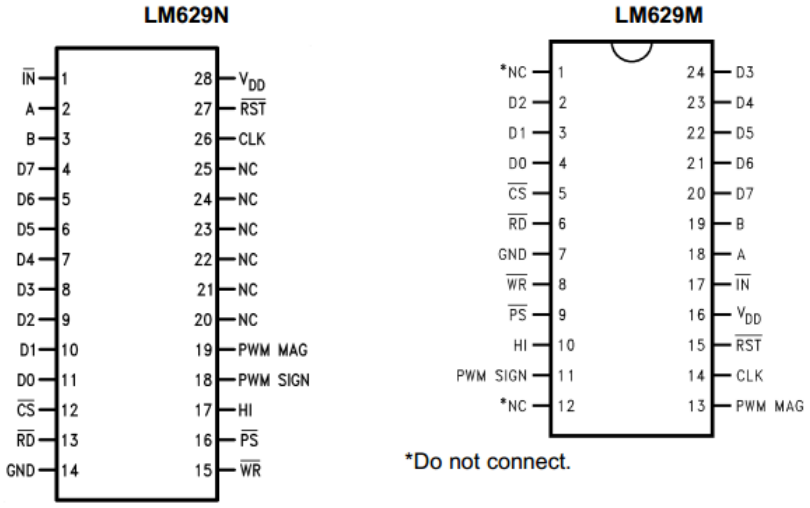


Figura 3.3: Distribución de pines del controlador LM629 [8].

3.1.2. Descripción de los pines del LM629

El LM629 utilizado posee 28 pines de los cuales solo 22 pines son utilizables (ver figura 3.3), a continuación se explica cada uno de ellos:

Pin 1, Entrada del índice ($\overline{\text{IN}}$): Este pin recibe el pulso del índice del encoder. Debe estar en estado lógico alto (uno lógico) si no es usado. La posición del índice es leída cuando los pines 1,2 y 3 están en estado lógico bajo (cero lógico).

Pin 2 y 3, Entradas para la señal del encoder (A, B): Estos pines reciben las dos fases en cuadratura provenientes del encoder incremental. Cuando el motor está en movimiento en dirección positiva, la señal en el pin 2 adelanta a la señal en el pin 3 90 grados.

Pines del 4 al 11, Puerto de entrada y salida (D0 a D7): Estos pines conforman el puerto de datos bidireccional el cual se conecta con la computadora. Es utilizado para escribir comandos y datos al LM629, y además para leer el byte de estatus y datos desde el LM629.

Pin 12, Entrada para selección de chip ($\overline{\text{CS}}$): Este pin es utilizado para activar las operaciones de escritura y lectura del LM629.

Pin 13, Entrada para lectura ($\overline{\text{RD}}$): Este pin es utilizado para activar la lectura de datos y del estatus.

Pin 14, Tierra (GND): Este pin debe ir conectado a la tierra de la fuente de alimentación.

Pin 15, Entrada para escritura ($\overline{\text{WR}}$): Este pin es utilizado para activar la escritura de datos y comandos.

Pin 16, Entrada para selección de puerto ($\overline{\text{PS}}$): Este pin es utilizado para seleccionar el puerto en modo de comando o en modo de datos. Para utilizar el puerto para comandos debe estar el pin en cero lógico, y para datos en uno lógico. Los siguientes modos de operación son controlados por el pin 16:

1. Cuando el pin 16 está en "0" lógico los comandos son escritos por el puerto de comandos.

2. Cuando el pin 16 está en “0” lógico el byte de estatus es leído por el puerto de comandos.
3. Cuando el pin 16 está en “1” lógico los datos son escritos y leídos por el puerto de datos.

Pin 17, Salida de interrupción (HI): Este pin emite un uno lógico como señal de alerta a la computadora cuando una condición de interrupción ha ocurrido.

Pin 18 y 19, Salida de signo y magnitud del Modulador de ancho de pulso (PWM Sign, PWM Mag): Estos pines son utilizados para la salida del signo y la magnitud del Modulador de ancho de pulso (PWM) respectivamente.

Pines del 20 al 25: Estos pines no son utilizados en el LM629. No deben conectarse.

Pin 26, Entrada de reloj (CLK): Este pin recibe el reloj del sistema, el cual es proporcionado externamente.

Pin 27, Entrada para Reiniciar ($\overline{\mathbf{RST}}$): El LM629 es reiniciado a sus condiciones iniciales cuando está en “0" lógico y se produce un disparo de flanco positivo. El pulso debe durar al menos 8 ciclos de reloj. Al reiniciar sucede lo siguiente:

1. Los coeficientes de los filtros y los parámetros de trayectoria son puestos a cero.
2. Establece el umbral de error de la posición al valor máximo (7FFF hex), y ejecuta efectivamente el comando LPEI.
3. Las interrupciones SBPA/SBPR son enmascaradas (deshabilitadas).
4. Las cinco interrupciones restantes son desenmascaradas (habilitadas).
5. Establece la posición actual a cero, o posición de origen (Home).
6. Establece el intervalo de muestreo derivativo a $2048/\mathrm{f}_{CLK}$ o 256 µs para un reloj de 8 MHz.

Pin 28, Entrada para fuente de voltaje: Este pin es utilizado para alimentar con +5V el LM629.

3.1.3. Teoría de operación

3.1.3.1. Introducción

El diagrama de bloques típico del sistema (ver figura 3.2) ilustra la construcción de un servo sistema utilizando el LM629. La comunicación entre el LM629 y el microcontrolador o computadora se da por medio del puerto de entradas y salidas, para así poder facilitar la programación del perfil de velocidad trapezoidal y los filtros de compensación digital. Una salida PWM nos da la señal para poder amplificarla y aplicarla al motor mediante un puente H. Un encoder incremental nos da la retroalimentación para cerrar el lazo de posición del servo. El generador de perfil de velocidad trapezoidal calcula la trayectoria requerida para cada modo de operación, ya sea posición o velocidad. En funcionamiento, el LM629 resta la posición actual (posición en retroalimentación) de la posición deseada (generador del perfil de posición), y el error de posición resultante es procesado por el filtro digital y acciona al motor a la posición deseada. La Tabla 3.1 proporciona un breve resumen de las especificaciones ofrecidas por el LM629.

Rango de posición	-1,073,741,824 a 1,073,741,823 cuentas o pulsos
Rango de velocidad	0 a $1{,}073{,}741{,}823/2^{16}$ cuentas/muestra
Rango de aceleración	0 a $1{,}073{,}741{,}823/2^{16}$ cuentas/muestra/muestra
Salida del motor	Señal PWM con signo y magnitud de 8 bits de resolución
Modos de operación	Posición y velocidad
Dispositivo de retroalimentación	Encoder incremental (con señal en cuadratura y soporte de pulso indexado)
Algoritmo de control	Control Proporcional Integral Derivativo (PID) (con límites de integración programable)
Intervalos de muestreo	Termino Derivativo: Programable desde $2048/f_{CLK}$ a $(2048*256)/f_{CLK}$ en pasos de $2048/f_{CLK}$
	Términos Proporcional e Integral: $2048/f_{CLK}$

Tabla 3.1: Resumen de especificaciones del sistema.

3.1.3.2. Interfaz de retroalimentación de la posición

La interfaz entre el LM629 y el motor es por vía de un encoder incremental. Las tres entradas nos proporcionan: entradas para dos señales en cuadratura, y una entrada para el pulso del índice. Las señales en cuadratura son utilizadas para realizar un seguimiento de la posición absoluta del motor. Cada vez que ocurre una transición lógica en una de las entradas de la señal en cuadratura, el registro interno de la posición del LM629 aumenta o decrementa según se presente el caso. Esto proporciona cuatro veces la resolución sobre el número de líneas proporcionado por el encoder. Ver figura 3.4. Cada una de las entradas de señal del encoder están sincronizadas con el reloj de LM629.

La salida opcional del pulso del índice que proporcionan algunos encoders asumen el estado lógico bajo (cero lógico) cada revolución. Si el LM629 es programado por el usuario, registrará la posición absoluta del motor en un registro dedicado (index register) cuando las tres entradas del encoder se encuentren en estado bajo.

Si el encoder no proporciona un pulso de índice, la entrada para el índice del LM629 puede ser utilizada también para guardar la posición inicial o posición origen del motor. En este caso, por lo general, el motor cerrará un interruptor el cual está puesto para provocar un estado lógico bajo en la entrada del índice (Pin 1), y así guardar la posición del motor en el registro dedicado. Si la entrada para el pulso del índice es colocada permanentemente a tierra puede causar un mal funcionamiento en el LM629.

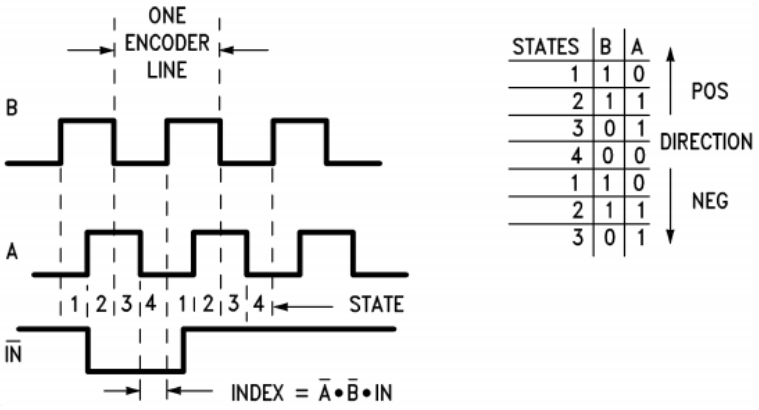


Figura 3.4: Señales en cuadratura de un encoder [8].

Cada uno de los cuatro estados de la señal en cuadratura de posición son decodificados por el LM629 aumentando cuatro veces la resolución sobre el número de líneas o pulsos del codificador (encoder).

3.1.3.3. Generación del perfil de velocidad (Trayectoria)

El generador de perfil de velocidad trapezoidal calcula la posición del motor contra el tiempo. Cuando opera en modo posición, el microcontrolador o la computadora especifican la aceleración, la velocidad máxima y la posición final. El LM629 utiliza esta información para poder modificar el movimiento de tal manera que el motor va acelerando según lo especificado hasta alcanzar la velocidad máxima o hasta que la desaceleración comience para llegar a la posición final especificada. El valor de la desaceleración es igual al valor de la aceleración. En cualquier momento durante el movimiento del motor la velocidad y/o la posición pueden ser modificadas, y el motor acelerar o desacelerar según sea el caso. La figura 3.5 muestra dos perfiles de velocidad trapezoidales. La figura 3.5a muestra un simple perfil trapezoidal, mientras que la figura 3.5b muestra un ejemplo de como se observa una trayectoria cuando la velocidad y la posición son cambiados en distintos tiempos durante el movimiento.

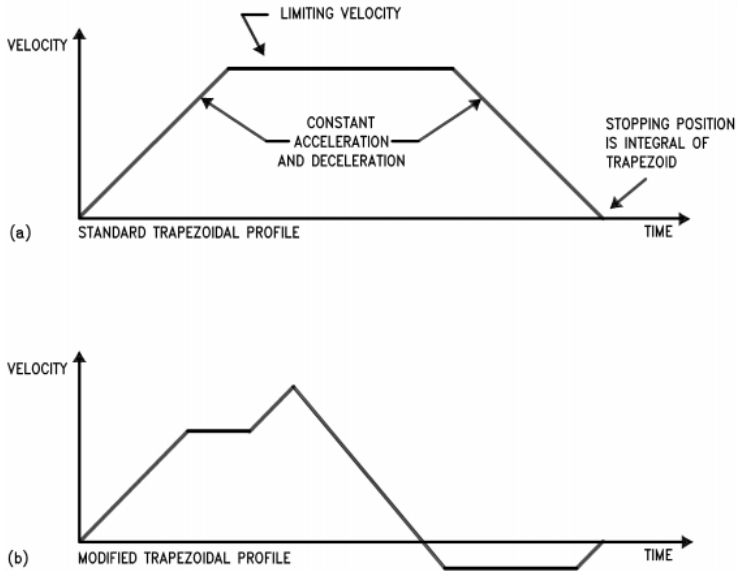


Figura 3.5: Perfiles de velocidad [8].

Cuando el LM629 está operando en modo velocidad, el motor acelera hasta la velocidad especificada con la tasa de aceleración especificada y mantiene dicha velocidad hasta que se ejecuta un comando de paro. La velocidad se mantiene avanzando a la posición deseada a un valor constante. Si se presentan perturbaciones en el movimiento mientras se esté operando en modo velocidad, la velocidad media a largo plazo se mantiene constante. Si el motor es incapaz de mantener la velocidad especificada (que podría ser causado por un bloqueo en el rotor, por ejemplo), la posición deseada seguirá aumentándose dando como resultado un error de posición muy grande. Si esta condición se ignora, y la fuerza que detiene al motor es liberada, el motor podría alcanzar una velocidad muy alta con el fin de alcanzar y posicionarse en la posición deseada. Esta condición se detecta fácilmente con los comandos LPEI y LPES, que se explican más adelante.

Todos los parámetros de trayectoria son valores de 32 bits. La posición es una cantidad con signo. La velocidad y la aceleración son de 16 bits, y solo números enteros fraccionables en 16 bits. La parte entera del número indica cuantas cuentas o cuantos pulsos por intervalo de muestro del motor cruzará. Y la parte fraccionaria (decimal) del número designa una cuenta o un pulso adicional por intervalo de muestreo. Aunque la resolución del LM629 se limite a cuentas enteras, las cuentas fraccionarias (decimales) proporcionan una mayor resolución de la velocidad media, lo mismo ocurre con la aceleración. En cada intervalo de muestreo el valor de la aceleración comandada es agregada a la velocidad deseada actual para así generar una nueva velocidad deseada (o hasta que haya alcanzado la velocidad deseada).

A fin de permitir velocidades muy bajas, es necesario tener velocidades con pulsos fraccionarios por muestra. La necesidad de los pulsos fraccionarios en la velocidad se puede ilustrar mediante el siguiente ejemplo: usando un codificador de 500 lineas (2000 pulsos) y un reloj de 8 MHz, dando un intervalo de muestreo de 256µs. Si la menor resolución es de 1 pulso por muestra, la velocidad mínima sería de 2 revoluciones por segundo o 120 rpm. (1/2000 revoluciones/pulsos $\times$ 1/256 µs pulsos/segundo). Muchas aplicaciones requieren velocidades y pasos en velocidad inferior a esta cantidad. Esto es proporcionado por los pulsos fraccionarios de aceleración y velocidad.

Para poder determinar los parámetros de la trayectoria para un movimiento deseado los cálculos se pueden realizar de la siguiente manera. Por ejemplo, se tiene un encoder de 500 pulsos por revolución, se desea que el motor tenga una aceleración de una revolución por segundo al cuadrado hasta que obtenga una velocidad de 600 revoluciones por minuto, y después desacelere hasta parar exactamente después de 100 revoluciones, la trayectoria se calcularía de la siguiente manera:

Para calcular posición:
Sea:

P=posicion deseada (unidades=Cuentas del encoder)

R=líneas del encoder*4 (resolucion del sistema)

entonces:

R=500*4=2000

calculando:

P=2000*número deseado de revoluciones

P=2000*100 revoluciones=200,000 cuentas (valor que se cargará a la trayectoria)

$P_{(código)}$=00030D40 (código hexadecimal que se enviará al LM629)

Para calcular velocidad:
Sea:

V=velocidad (unidades=cuentas/muestra)

T=tiempo de muestreo (segundos)=341us (con un reloj de 6 MHz)

C=factor de conversión=1 minuto/60 segundos

entonces:

V=R*T*C*revoluciones por minuto deseadas

calculando::

V=2000*341E-6*1/60*600 rpm

V=6.82cuentas/muestra

$V_{(escala)}$=6.82*65,536=446,955.52

$V_{(redondeo)}$=446,956 (valor que se cargará a la trayectoria)

$V_{(codigo)}$=0006D1EC (código hexadecimal que se enviará al LM629)

Para calcular aceleración:

Sea:

A=aceleración (unidades=cuentas/muestra/muestra)

entonces:

A=R*T^2*aceleración deseada (rev/seg/seg)

calculando:

A=2000*341E-6*1 rev/seg/seg

A=2.33E-4 cuentas/muestra/muestra

$A_{(escala)}$=2.33E-4*65,536=15.24

$A_{(redondeo)}$=15 (valor que se cargará a la trayectoria)

$A_{(código)}$=0000000F (código hexadecimal que se enviará al LM629)

Los valores de la posición, velocidad y la aceleración deben ser convertidos a código binario para poder ser cargados al LM629. Los valores de velocidad y aceleración deben multiplicarse por $65,536$ para ajustar el formato entero/decimal requerido de los datos de entrada, es decir la resolución de 2^{16}.

3.1.3.4. Filtro de compensación PID

A continuación por cuestiones de nomenclatura que se utiliza en la hoja de datos del controlador LM629, al controlador PID se le llamará filtro PID, esto para evitar confusiones.

El LM629 utiliza un filtro digital Proporcional Derivativo Integral (PID) para compensar el bucle de control. El motor se mantiene en la posición deseada aplicando una fuerza que compensa al motor siendo esta proporcional al error de posición, más la integral del error, más el derivativo del error. La siguiente ecuación del tiempo discreto ilustra el control realizado por el LM629:

$$u(n) = K_p * e(n) + K_i \sum + K_d[e(n') - e(n' - 1)] \tag{3.1}$$

Donde: $u(n)$ es la señal de salida del control de motor en una muestra de tiempo n, $e(n)$ es el error de posición en una muestra de tiempo, n' indica el muestreo a la frecuencia de muestreo derivativo, y K_p, K_i, y K_d son los parámetros de filtro en un

tiempo discreto cargados por el usuario.

El primer término, el término proporcional, proporciona una fuerza de compensación proporcional al error de posición, tal como lo hace un resorte, obedece la ley de Hooke. El segundo término, el término de integración, proporciona una fuerza de compensación que crece con el tiempo, y asegura que el error de posición estática sea cero. Si hay una carga de par constante, aun así el motor es capaz de lograr un error de posición cero.

El tercer término, el término derivativo, proporciona una fuerza de compensación proporcional a la velocidad de cambio del error de posición. Actúa igual que un amortiguamiento en un sistema de masa-resorte (como el amortiguador de un automóvil). El intervalo de muestreo relacionado con el termino derivativo puede ser seleccionado por el usuario; esto permite al LM629 ser capaz de controlar varios rangos de cargas de inercia, proporcionando una mejor aproximación del derivativo constante. En general, los intervalos de muestreo largos son útiles para operaciones de baja velocidad.

En operación, el algoritmo del filtro recibe un error de señal de 16 bits desde el punto de suma del lazo. La señal de error es saturada a 16 bits para asegurar un comportamiento predecible. Además de ser multiplicada por el coeficiente de filtro K_p, la señal de error es agregada a una acumulación de errores previos (para formar la señal integral) y, a una velocidad determinada por el intervalo de muestreo derivativo seleccionado, los errores previos se restan de ella (para formal la señal derivativa). Todas las multiplicaciones de filtros son operaciones de 16 bits; sólo se utilizan los 16 bits inferiores del producto.

La señal integral se mantiene a 24 bits, pero solo los 16 bits superiores son utilizados. Esta técnica de escalado resulta muy utilizable en los valores del coeficiente K_i por ser menos sensible. Los 16 bits están desplazados hacia la derecha 8 posiciones y están multiplicados por el coeficiente del filtro K_i para formar el término que contribuye a la salida del control del motor. La magnitud absoluta de este producto es comparada con el coeficiente *il* (limite de integración), y la magnitud menor apropiadamente entonces medida contribuye a la señal de control del motor.

La señal derivativa es multiplicada por el coeficiente K_d cada intervalo de muestreo derivativo. Este producto contribuye para salida de control del motor cada intervalo de muestreo, independientemente del intervalo de muestreo derivativo seleccionado por el usuario.

Los términos K_p, limite K_i, y K_d se suman para formar una cantidad de 16 bits. Dependiendo del modo de salida (tamaño de la palabra), o bien los 8 bits o 12 bits superiores se convierten en la señal de salida de control del motor.

3.1.3.5. Operaciones de escritura y lectura en el controlador LM629

El microcontrolador escribe los comandos en el LM629 por medio del puerto de entradas/salidas (I/O) cuando la entrada de selección de puerto ($\overline{\text{PS}}$) está en cero lógico. El código del comando deseado se aplica a la línea del puerto paralelo y la entrada de escritura ($\overline{\text{WR}}$) se pone en uno lógico. El byte del comando se queda enclavado en el LM629 mientras la entrada de escritura ($\overline{\text{WR}}$) se mantiene en flanco positivo. Cuando se escribe un comando es necesario primero leer el "Status Byte" y revisar el estado de la bandera llamada "Busy Bit" (Bit 0). Si la bandera "Busy Bit" está en lógica alta, el comando aún no puede ser escrito porque otro comando está en proceso. La bandera "Busy Bit" nunca dura más de 100 µs en estado alto, generalmente pasa al estado bajo entre 15 µs y 25 µs.

El microcontrolador lee el "Status Byte" del LM629 cuando está en uno lógico la entrada de escritura ($\overline{\text{RD}}$) y cuando la entrada de selección de puerto ($\overline{\text{PS}}$) está en cero lógico. Esta información permanece válida hasta que la entrada de escritura ($\overline{\text{RD}}$) pasa a cero lógico.

Escribir y leer datos en el LM629 (lo opuesto a escribir comandos y leer el "Status Byte") se puede realizar con la entrada de selección de puerto ($\overline{\text{PS}}$) en estado lógico alto. Estas escrituras y lecturas son siempre un número entero y palabras de dos bytes, con el primer byte de cada palabra siendo el más significativo. Cuando se transfieren palabras de datos, es necesario primero leer el "Status Byte" y revisar el estado de la bandera "Busy Bit". Si la bandera "Busy Bit" está en estado lógico bajo, el usuario puede enviar el par de bytes de la palabra secuencialmente pero debe ser revisada la

bandera de nuevo si se enviara otro par de bytes (esto solo cuando se envían múltiples palabras). Si se intenta enviar un dato o un comando cuando la bandera "Busy Bit" está en estado lógico alto, este dato o comando será ignorado.

3.1.3.6. Salidas al motor (Motor Outputs)

El LM629 proporciona una señal de salida de modulación por ancho de pulsos con signo y magnitud de 8 bits, la cual puede ser utilizada directamente por un amplificador puente H. La figura 3.6 muestra el formato de una señal de salida de modulación por ancho de pulsos (PWM).

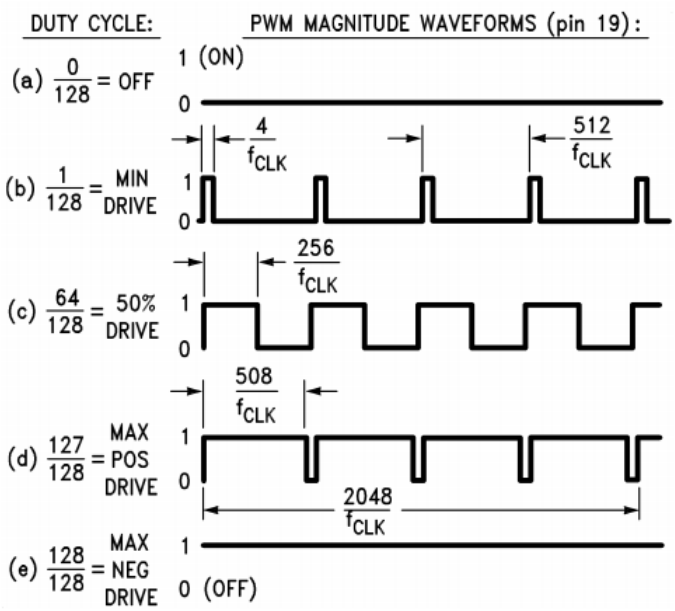


Figura 3.6: Formato de la señal de modulación de ancho de pulso [8].

3.1.3.7. Comandos de control establecidos para el usuario

El LM629 posee comandos establecidos para poder tener un control más sencillo sobre este. Algunos de estos comandos no necesitan datos adicionales para poder ejecutarse, por ejemplo el comando STT (STarT motion) no requiere datos adicionales, pero el comando LFIL (Load FILter) si requiere datos (intervalo de muestreo derivativo y los parámetros del filtro).

Los comandos están catalogados por función: inicialización, interrupciones de control, filtros de control, trayectoria de control y reporte de datos. Los comandos están listados en la Tabla 3.2 y son descritos en los párrafos siguientes.

Comando	**Categoría**	**Hex**	**Bytes de datos**	**Nota**
RESET	Inicialización	00	0	1
DFH	Inicialización	02	0	1
SIP	Interrupción	03	0	1
LPEI	Interrupción	1B	2	1
LPES	Interrupción	1A	2	1
SBPA	Interrupción	20	4	1
SBPR	Interrupción	21	4	1
MSKI	Interrupción	1C	2	1
RSTI	Interrupción	1D	2	1
LFIL	Filtro	1E	2 a 10	1
UDF	Filtro	04	0	1
LTRJ	Trayectoria	1F	2 a 14	1
STT	Trayectoria	01	0	3
RDSTAT	Reporte	–	1	1, 4
RDSIGS	Reporte	0C	2	1
RDIP	Reporte	09	4	1
RDDP	Reporte	08	4	1
RDRP	Reporte	0A	4	1
RDDV	Reporte	07	4	1
RDRV	Reporte	0B	2	1
RDSUM	Reporte	0D	2	1

Tabla 3.2: Comandos de control del LM629 establecidos para el usuario.

Nota:

1. Los comandos pueden ser ejecutados durante el movimiento.
2. Los comandos no pueden ser ejecutados durante el movimiento.
3. El comando puede ser ejecutado durante el movimiento si la aceleración no ha sido cambiada.
4. El comando no necesita ningún código debido a que el puerto de comando de estado del byte de lectura es totalmente compatible con el hardware.

3.1.3.7.1. Comandos de inicialización

Los siguientes comandos del LM629 son los primeros que se utilizan para poder iniciar el sistema para su uso.

Comando RESET: Reset LM629

Este comando establece con un valor de cero los coeficientes del filtro, los parámetros de trayectoria y la salida del motor.

Comando DHF: Define Home

Este comando almacena la posición actual como la posición cero o la posición de referencia.

3.1.3.7.2. Comandos de control de interrupciones

Los siguientes comandos están asociados con condiciones las cuales pueden ser utilizadas para generar una interrupción en el microcontrolador. Para que cualquier interrupción pueda interrumpir realmente al microcontrolador a través del pin 17, no deben estar enmascaradas por el comando MSKI, dejándolos en estado lógico alto.

Para conocer el estado de las interrupciones basta con revisar el "Status Byte".

Comando SIP: Set Index Position

Este comando establece la posición absoluta una vez que aparece el siguiente pulso del índice y es guardada en el registro dedicado (index register), y el bit 3 del "Status Byte" se pondrá en estado lógico alto. La posición se registra cuando ambas entradas de fase del encoder y el pulso indexado están en nivelo lógico bajo.

Comando LPEI: Load Position Error For Interrupt

Este comando guarda en un registro el valor limite que debe alcanzar el erro de posición para que se genere una interrupción por parte del LM629 al microcontrolador.

Comando LPES: Load Position Error For Stopping

Este comando es esencialmente similar al LPEI, pero adicionalmente tiene la capacidad de detener el motor automáticamente.

Comando SBPA: Set Breakpoint Absolut

Este comando permite al usuario establecer un punto de control en términos de posición absoluta.

Comando SBPR: Set Breakpoint Relative

Este comando es similar al SBPA pero este establece en términos de posición relativa.

Comando MSKI: Mask Interrupts

Este comando permite al usuario seleccionar cuales de las condiciones potenciales de interrupción están activadas y cuales no. En la Tabla 3.3 se muestran las interrupciones con su bit correspondiente que pueden ser activadas.

Posición del bit	**Función**
Bits 15 al 7	No utilizados
Bit 6	Interrupción para el punto de control (SBPA, SBPR)
Bit 5	Interrupción para error de posición (LPEI, LPES)
Bit 4	Interrupción de sobreflujo
Bit 3	Interrupción del pulso del índice (SIP)
Bit 2	Interrupción de trayectoria completa
Bit 1	Interrupción de error de comando
Bit 0	No utilizado

Tabla 3.3: Asignaciones de bits para enmascarado y reinicio de las interrupciones.

Comando RSTI: Reset Interrupts

Este comando es utilizado para reiniciar o restablecer a cero los bits de las interrupciones de la Tabla 3.3 que se hayan activado. Puede restablecer una interrupción o todas en un solo comando RSTI.

3.1.3.7.3. Comandos del filtro de control

Los siguientes comandos son utilizados para establecer el intervalo de muestro del termino derivativo, para ajustar los parámetros de los filtros para ajustar el sistema y para controlar el tiempo de estos cambios.

Comando LFIL: Load Filter Parameters

Este comando es utilizado para enviar los parámetros del filtro al LM629 para compensar el lazo de control son: K_p, K_i, K_d e il (limite de integración). También es utilizado para cargar el intervalo de muestro del termino derivativo. En la Tabla 3.4 se muestran los bits utilizados y su correspondiente función.

Posición del bit	**Función**
Bit 15	Intervalo de muestreo derivativo 7
Bit 14	Intervalo de muestreo derivativo 6
Bit 13	Intervalo de muestreo derivativo 5
Bit 12	Intervalo de muestreo derivativo 4
Bit 11	Intervalo de muestreo derivativo 3
Bit 10	Intervalo de muestreo derivativo 2
Bit 9	Intervalo de muestreo derivativo 1
Bit 8	Intervalo de muestreo derivativo 0
Bit 7	No utilizado
Bit 6	No utilizado
Bit 5	No utilizado
Bit 4	No utilizado
Bit 3	Término Kp
Bit 2	Término Ki
Bit 1	Término Kd
Bit 0	Término il

Tabla 3.4: Asignación de bit del control del filtro.

Los bits del 8 al 15 seleccionan el intervalo de muestreo del término derivativo. Ver Tabla 3.5. El usuario debe guardar y restaurar estos bits a nivel local durante las escrituras sucesivas de la palabra de control del filtro. Los bits del 4 al 7 del control del filtro no son utilizados.

Los bits del 0 al 3 informan al LM629 que alguno o todos (o ninguno) de los parámetros del filtro serán escritos.

Los bytes de datos especificados por e inmediatamente después de la palabra del control del filtro están escritos en pares para comprender palabras de 16 bits. El orden

para enviar las palabras de datos hacia el LM629 corresponde de manera descendente a como se muestran en la tabla, es decir, primero de carga el valor Kp, después Kd, Ki y al último *il*.

								Derivativo Seleccionado
15	14	13	12	11	10	9	8	**Intervalo de muestreo**
0	0	0	0	0	0	0	0	256 us
0	0	0	0	0	0	0	1	512 us
0	0	0	0	0	0	1	0	768 us
0	0	0	0	0	0	1	1	1024 us
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1	1	1	1	1	1	1	1	65,536 us

Tabla 3.5: Selección de código para el intervalo de muestro del término derivativo.

Comando UDF: Update Filter

Este comando es utilizado para actualizar los parámetros del filtro que han sido programados por medio del comando LFIL.

3.1.3.7.4. Comandos del control de trayectoria

Los siguientes comandos son utilizados para establecer los parámetros de control de la trayectoria (posición, velocidad y aceleración), el modo de operación (posición o velocidad), la dirección (solo en el modo velocidad), el tipo de movimiento (deseado o con paro manual), y el tiempo de los cambios del sistema.

Comando LTRJ: Load Trajectory Parameters

Este comando es utilizado para escribir en el LM629 los parámetros de la trayectoria. La siguiente tabla muestran los bits utilizados para el control de los parámetros de la trayectoria.

Posición Bit	Función
Bit 15	No utilizado
Bit 14	No utilizado
Bit 13	No utilizado
Bit 12	Dirección hacia adelante (solo en modo velocidad)
Bit 11	Modo velocidad
Bit 10	Paro lento (desaceleración programada)
Bit 9	Paro brusco (máxima desaceleración)
Bit 8	Apagar motor
Bit 7	No utilizado
Bit 6	No utilizado
Bit 5	Aceleración se cargará
Bit 4	Dato de aceleración es relativo
Bit 3	Velocidad se cargará
Bit 2	Dato de velocidad es relativo
Bit 1	Posición de cargará
Bit 0	Dato de posición es relativo

Tabla 3.6: Asignación de bit para el control de trayectoria.

El bit 12 determina la dirección del motor cuando esta en modo velocidad. Un uno lógico indica una dirección hacia adelante. Este bit no funciona cuando se esta en modo posición.

El bit 11 determina si el LM629 debe operar en modo velocidad (uno lógico) o en modo posición (cero lógico).

Los bits del 8 al 10 son utilizados para seleccionar un método de paro manual del motor. Estos bits solo pueden ser utilizados cuando el LM629 esta operando en modo velocidad, ya que en modo posición el motor siempre para de una forma suave al término de la trayectoria.

Los bits del 0 al 5 informan al LM629 si alguno o todos los parámetros que controlan la trayectoria serán escritos, y si los datos deben interpretarse como absolutos o relativos. Estos bits son activados con un uno lógico. Cualquier parámetro puede ser cambiado si el motor se encuentra en movimiento, sin embargo, si la aceleración es cambiada entonces el siguiente comando STT no debe ser ejecutado hasta que el LM629

haya completado el movimiento actual o haya sido parado manualmente.

Los bytes de datos especificados por e inmediatamente después de la palabra de control de trayectoria están escritos en pares que comprenden palabras de 16 bits. Cada parámetro requiere de dos palabras de datos de 16 bits. El orden para enviar las palabras de datos hacia el LM629 corresponde de manera descendente a como se muestran en la tabla, es decir, primero se carga la aceleración, después la velocidad y por último la posición.

El valor de la aceleración y la velocidad son de 32 bits, sólo positivos, y con un rango desde 0 (00000000 hex) a [2^{30}]-1 (3FFFFFFF hex). El valor de la aceleración no debe ser mayor al de la velocidad. El valor de la posición tiene signo y es un número entero de 32 bits, con un rango desde -[2^{30}] (C0000000 hex) a [2^{30}]-1 (3FFFFFFF hex).

Comando STT: Start Motion Control

Este comando es utilizado para ejecutar la trayectoria deseada, con las especificaciones que han sido programadas por medio del comando LTRJ.

3.1.3.7.5. Comandos de reporte de datos

Los siguientes comandos son utilizados para obtener datos desde varios registros en el LM629, los cuales son el estatus, la posición y la velocidad.

Comando RDSTAT: Read Status Byte

Este realmente no es un comando, pero esta listado porque su uso es muy frecuente en la comunicación del control con el microcontrolador. No posee código de identificación, es soportado directamente por el hardware y puede ser ejecutado en cualquier tiempo. La lectura del estado de un byte se selecciona mediante la colocación de CS, PS y RD a cero lógico. En la siguiente tabla se observa la asignación de cata byte del "Status Byte".

Posición del Bit	**Función**
Bit 7	Motor apagado
Bit 6	Punto de control alcanzado [Interrupción]
Bit 5	Error de posición excesiva [Interrupción]
Bit 4	Se ha producido un "wraparound" [Interrupción]
Bit 3	Pulso indexado observado [Interrupción]
Bit 2	Trayectoria completa [Interrupción]
Bit 1	Error de comando [Interrupción]
Bit 0	Bit Ocupado ("Busy Bit")

Tabla 3.7: Asignación de bits del "Status Byte".

Bit 7, bandera que indica si el motor esta apagado, esta en uno lógico cuando la salida de accionamiento del motor esta apagada.

Bit 6, bandera cuando el punto de control es alcanzado, esta en uno lógico cuando el punto de control ha sido superado.

Bit 5, bandera de interrupción cuando el error de posición es excesivo, esta en uno lógico cuando esta condición ha ocurrido.

Bit 4, bandera de interrupción cuando ha ocurrido un "wraparound", esta en uno lógico cuando esta condición ha ocurrido.

Bit 3, bandera de interrupción cuando un pulso indexado se ha adquirido, esta en uno lógico cuando un pulso indexado ha ocurrido (si el comando SIP ha sido ejecutado).

Bit 2, bandera de interrupción cuando se ha completado la trayectoria, esta en uno lógico cuando la trayectoria programada por el comando LTRJ e iniciada por el comando STT ha sido completada.

Bit 1, bandera de interrupción cuando hay un error de comando, esta en uno lógico cuando el usuario ha intentado leer datos cuando una escritura era lo apropiado (o viceversa).

Bit 0, bandera de "Busy Bit", esta en uno lógico cuando aún esta procesando datos o comandos, solo cuando esta en cero lógico los comandos y/o datos no serán ignorados.

Comando RDSIGS: Read Signals Register

Este comando es utilizado para leer señales internas registradas. El primer byte leído es el mas significativo. El byte menos significativo (con excepción del bit 0) duplica el byte de estado.

Posición del bit	**Función**
Bit 15	Interrupción al microcontrolador
Bit 14	Aceleración cargada (pero no actualizada)
Bit 13	UDF ejecutado (pero los filtro no han sido actualizados)
Bit 12	Dirección hacia adelante
Bit 11	Modo velocidad
Bit 10	Trayectoria completa (On Target)
Bit 9	Apagar Error de posición excesivo
Bit 8	Salida de 8 bits
Bit 7	Motor apagado
Bit 6	Punto de control alcanzado [Interrupción]
Bit 5	Error de posición excesivo [Interrupción]
Bit 4	Ha ocurrido un "wraparound" [Interrupción]
Bit 3	Pulso del índice adquirido [Interrupción]
Bit 2	Trayectoria completa [Interrupción]
Bit 1	Error de comando [Interrupción]
Bit 0	Adquirir siguiente índice (SIP Ejecutado)

Tabla 3.8: Asignación de bit de señales registradas.

Bit 15, bandera de interrupción al microcontrolador, cuando está en uno lógico indica que la salidas de las interrupciones (Pin 17) esta en uno lógico, por alguna de las seis condiciones de interrupciones que posee el LM629.

Bit 14, bandera de la aceleración cargada, está en uno lógico cuando el dato de la aceleración ha sido escrito en el LM629.

Bit 13, bandera de la ejecución del comando UDF, está en uno lógico cuando el comando UDF es ejecutado.

Bit 12, bandera de dirección hacia adelante solo cuando el LM629 opera en modo velocidad, está en uno lógico cuando el movimiento deseado esta hacia adelante.

Bit 11, bandera de modo velocidad, está en uno lógico cuando el usuario por medio del comando LTRJ selecciona el modo velocidad.

Bit 10, bandera de trayectoria completa, está en uno lógico cuando el generador de trayectoria ha completado sus funciones para el último comando STT establecido.

Bit 9, bandera de error de posición excesiva apagada, está en uno lógico cuando el comando LPES es ejecutado.

Bit 8, bandera de salida de 8 bits, esta en uno lógico cuando el LM629 ha sido reiniciado o cuando el comando PORT8 (solo en LM628) es ejecutado.

Bits del 0 al 7 son los utilizados por el "Status Byte" (ver Tabla 3.7), con excepción del bit 0.

Bit 0, bandera de pulso indexado adquirido, está en uno lógico cuando el comando SIP es ejecutado, y permanece asi hasta que otro pulso indexado ocurre.

Comando RDIP: Read Index Position

Este comando lee la posición guardada en el registro "index register".

Comando RDDP: Read Desires Position

Este comando lee la posición deseada de salida (temporal actual) instantánea del generador de perfil. Este es el "punto referencia" de entrada al comparador para el lazo de la posición.

Comando RDRP: Read Real Position

Este comando lee la posición actual del motor. Este comando retroalimenta la entrada del lazo en el punto de suma.

Comando RDDV: Read Desired Velocity

Este comando lee la parte entera y decimal de la velocidad deseada (temporal actual) del generador de perfil.

Comando RDRV: Read Real Velocity

Este comando lee la parte entera de la velocidad real instantánea actual del motor.

Comando RDSUM: Read Integration-Term Summation Value

Este comando lee el valor que se ha acumulado del término de integración. La capacidad de leer este valor puede ser útil en la iniciación o en el ajuste adaptativo del sistema.

3.1.4. Conexión del controlador LM629

La interfaz de conexión del LM629 con el microcontrolador (Host) es por medio de un puerto de datos o comandos (según sea el caso) de 8 bits que es controlado por cuatro líneas. Estas líneas son la entrada de selección de chip ($\overline{\text{CS}}$), la entrada de lectura ($\overline{\text{RD}}$), la entrada de escritura ($\overline{\text{WR}}$) y la entrada de selección de puerto ($\overline{\text{PS}}$), ver figura 3.7. Ademas posee opcionalmente una linea de interrupción (HI), desde el LM629 hacia el microcontrolador que se puede utilizar.

El LM629 también posee una interfaz de conexión con el encoder por medio de dos entradas (A y B) para el decodificador de cuadratura TTL del LM629. Opcionalmente también posee una entrada para el pulso del índice (IN), si esta entrada no se utilizará debe de estar en nivel alto.

Como salida hacia el motor, el LM629 posee una señal de modulación de ancho de pulso con signo y magnitud de 8 bits de resolución. Esta señal puede ser fácilmente utilizada para el control de la salida de un puente H.

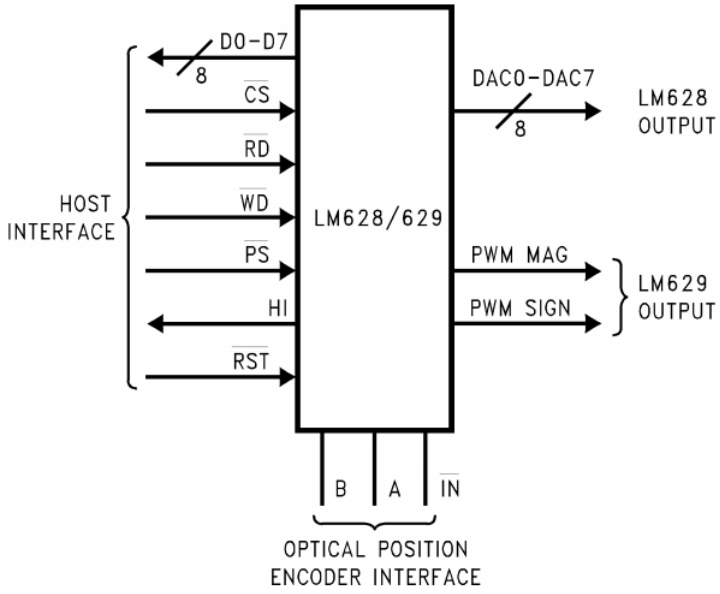


Figura 3.7: Conexión típica del LM629 [9].

3.1.5. Programación del controlador LM629

Para facilitar el proceso de programación del LM629 se utilizan distintos módulos funcionales, donde cada módulo ejecuta una tarea específica. A continuación se explican a detalle los módulos primordiales para realizar el control de un motor.

3.1.5.1. Módulo para verificar el "Busy-Bit"

Este primer módulo se requiere para una correcta programación del LM629, ya que es necesario verificar en que estado esta para poder enviar datos o comandos. El "Busy-Bit" es el bit cero del "Status byte" (ver figura 3.10), este bit se pone en uno lógico después de escribir un comando, o después de leer o escribir el segundo byte de una palabra de datos. Mientras esté en uno lógico este bit, el LM629 ignorará cualquier comando o transferencia de datos.

El módulo de verificación del "Busy-Bit" monitorea el "Status byte" y espera hasta que el bit este en cero lógico y se pueda restablecer la comunicación entre el microcontrolador y el LM629. Debe insertarse después de escribir un comando, o después de leer

o escribir el segundo byte de una palabra de datos. La figura 3.8 representa el modulo de control del "Busy-Bit.

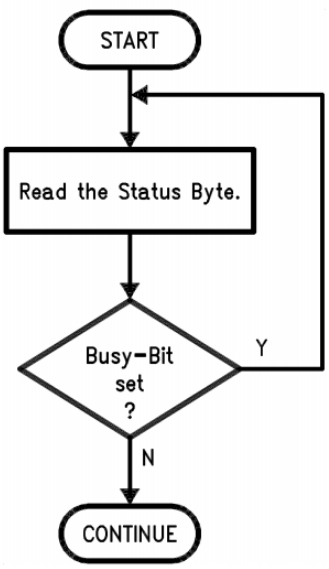


Figura 3.8: Modulo de verificación del "Busy-Bit" [7].

3.1.5.2. Módulo de inicialización

Este módulo contiene un comando de reinicio y otro de inicialización para interrupciones, datos y comandos.

Bloque de reinicio de hardware

Inmediatamente después de encender el LM629, debe ejecutarse el reinicio de hardware. El reinicio del hardware es inicializado al poner en cero lógico el pin 27 ($\overline{\text{RST}}$) por al menos ocho ciclos de reloj del LM629. La rutina de reinicio inicia después de que el pin 27 ($\overline{\text{RST}}$) ha retornado a uno lógico. Durante el tiempo de ejecución del reinicio, $1.5ms$ máximo, el LM629 ignorará cualquier comando o transferencia de datos.

El reinicio de hardware obliga al LM629 a tener el estado a continuación descrito:

- El coeficiente de muestreo derivativo es puesto a uno, y los coeficientes de los filtros son puestos a cero.

- Todos los parámetros de trayectoria son puestos a cero.
- La posición actual absoluta del eje es puesta a cero (Home).
- Los puntos de control son enmascarados (deshabilitados), y las interrupciones restantes son desenmascaradas (habilitadas).
- Se ajusta el umbral del error de posición a su valor máximo, 7FFF hexadecimal.

La siguiente figura muestra un bloque de reinicio de hardware que incluye una prueba de funcionalidad del LM629. Esta prueba deber ser completada inmediatamente después de todos los reinicio de hardware.

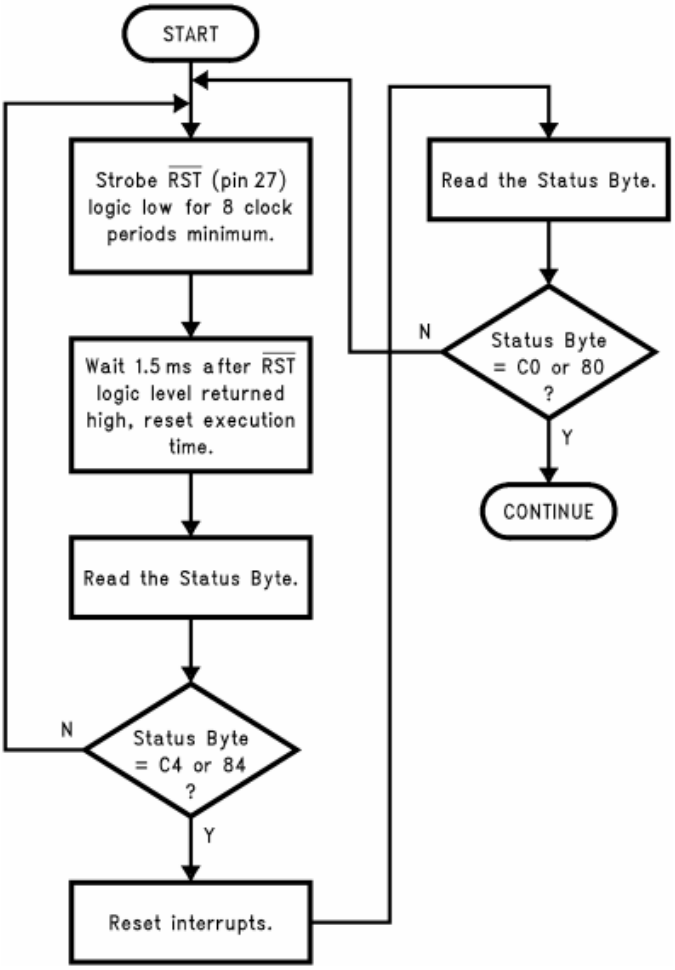


Figura 3.9: Bloque de reinicio de hardware [7].

Bloque de reinicio y enmascarado de interrupciones

Una secuencia del comando RSTI nos ayuda a reiniciar los bits de las banderas de las interrupciones, desde el bit 1 hasta el bit 6 del "Status Byte" (ver figura 3.10). El comando RSTI también reinicia la salida de interrupciones (pin 17) hacia el microcontrolador, y los 15 bits de las señales internas registradas (ver Tabla 3.8).

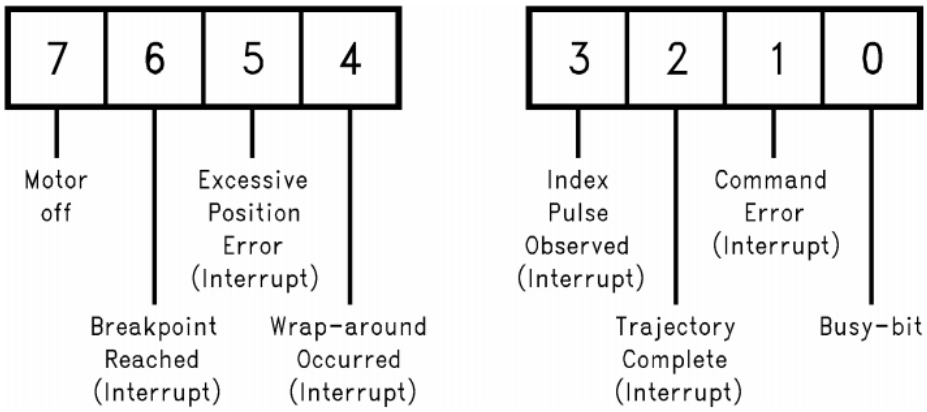


Figura 3.10: Asignación de los bits del "Status Byte" [7].

Inmediatamente después de ejecutar el comando RSTI, debe enviarse una palabra de datos, donde el primer byte no es utilizado. El segundo byte utilizando ceros lógicos del bit 1 al bit 6 reinicia las interrupciones correspondientes (ver figura 3.11). Cabe mencionar que se puede utilizar cualquier combinación de bits con un solo comando RSTI. Esta característica permite que las interrupciones puedan ser reiniciadas de acuerdo a la prioridad programada por el usuario.

Una vez ejecutado el comando RSTI, puede ejecutarse el comando MSKI para enmascarar o desenmascarar los bits de interrupciones que se deseen tener disponibles con el microcontrolador. Igual que con el comando RSTI, debe enviarse una palabra de datos, donde el primer byte no es utilizado y el segundo byte utilizando ceros lógicos del bit 1 al bit 6 desenmascara las interrupciones correspondientes.

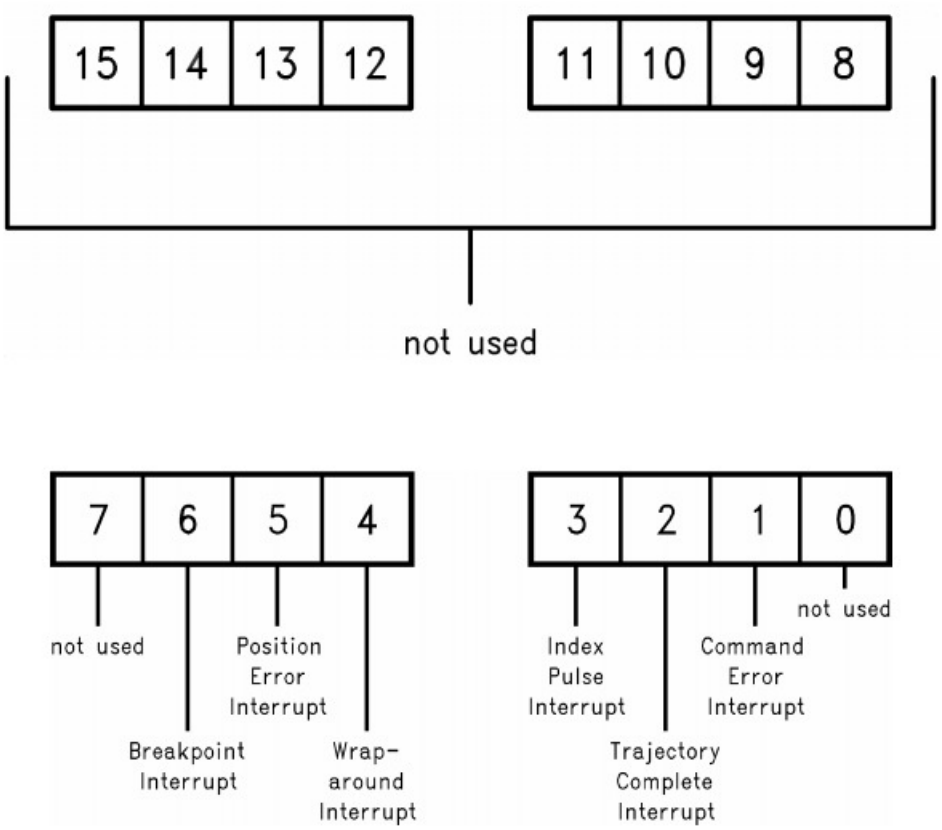


Figura 3.11: Asignación de los bits de interrupciones para RSTI o MSKI [7].

3.1.5.3. Módulo de programación de filtros

Este módulo contiene dos comandos para poder cargar y actualizar los parámetros (coeficientes) de los filtros para el control PID.

Cargar los parámetros del filtro

Para cargar los parámetros del filtro se utiliza el comando LFIL. el cual carga una palabra de control del filtro y un numero variable de palabras de datos.

Al ejecutar el comando LFIL inicializa los coeficientes del filtro, inmediatamente después se envían dos bytes de datos, donde el primer byte programa el coeficiente del intervalo de muestreo derivativo (ds) y el segundo byte, indica con unos lógicos en las posiciones de bits correspondientes, cuales de los cuatros filtros serán cargados al

LM629 (ver Tabla 3.5 y figura 3.12). Cualquier combinación de los cuatro coeficientes pueden ser cargados con un solo comando LFIL.

Inmediatamente después de la palabra de control del filtro, los coeficientes del filtro son escritos. Cada coeficiente es escrito en un par de bytes de datos, es decir, una palabra de datos. Debido a que se puede cargar cualquier combinación de los cuatro coeficientes con un solo comando LFIL, el numero de palabras de datos siguientes a la palabra de control pueden variar en un rango de cero a cuatro.

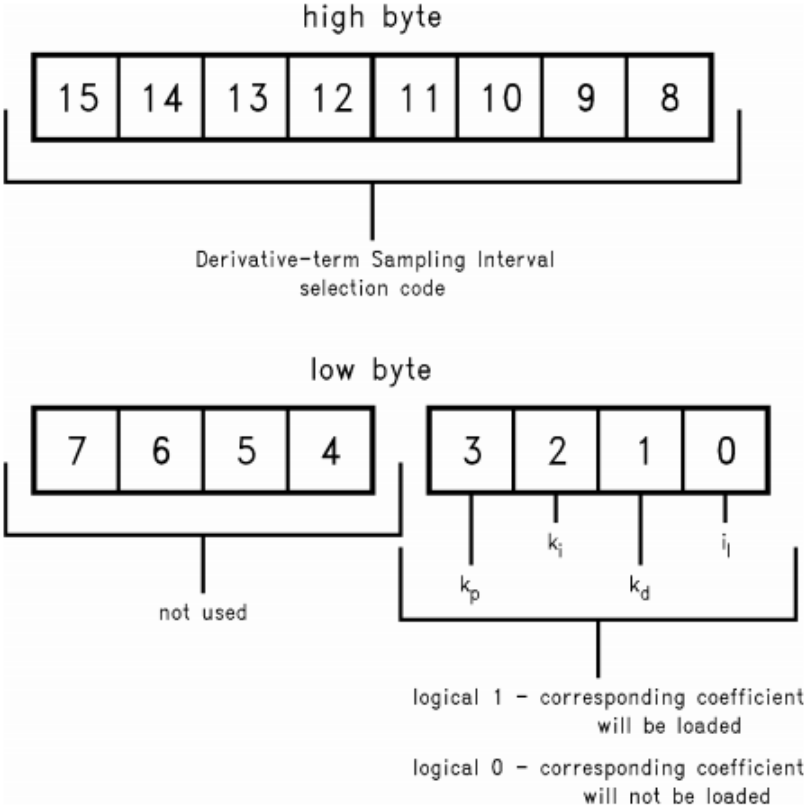


Figura 3.12: Asignación de bits del control del filtro [7].

Actualizar el filtro

El comando para actualizar el filtro (UDF), transfiere los nuevos coeficientes desde los buffers de entrada hasta los registros de trabajo. Los nuevos coeficientes del filtro no afectan a la característica de transferencia del filtro, hasta que el comando UDF se

ejecuta.

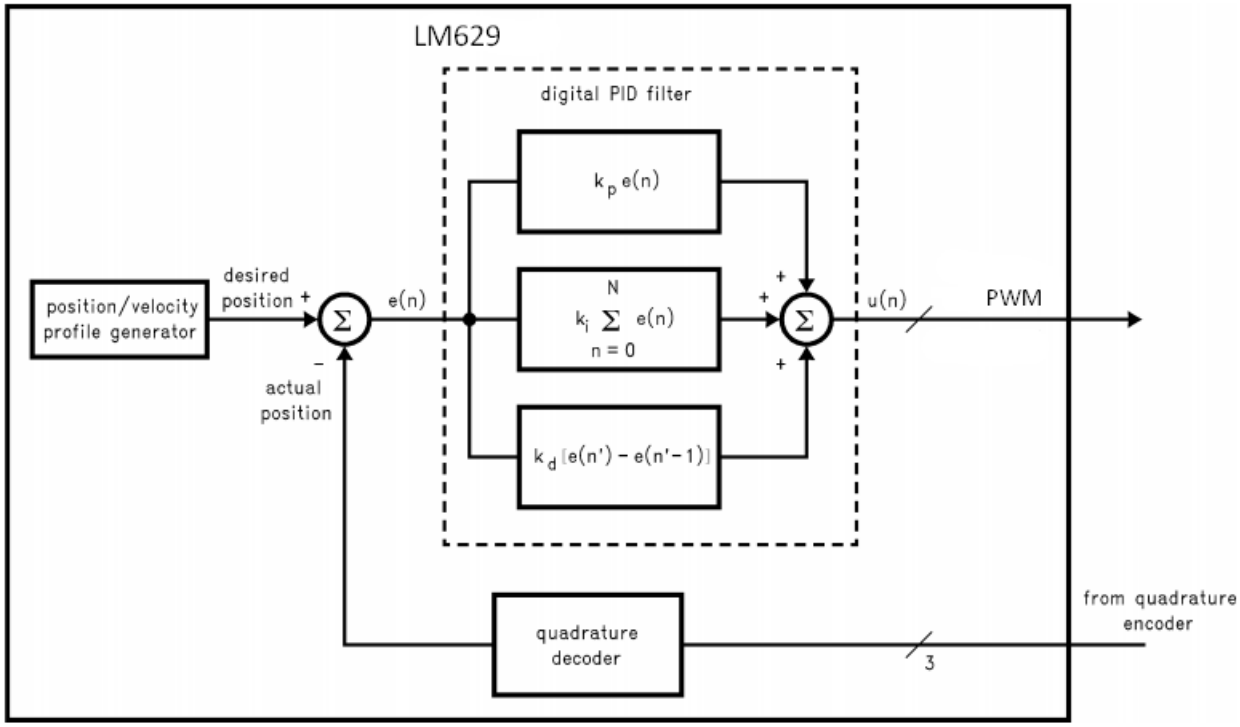


Figura 3.13: Diagrama de bloques del control PID [7].

3.1.5.4. Módulo de programación de trayectoria

Este módulo contiene dos comandos para cargar y actualizar los parámetros de la trayectoria.

Cargar parámetros de trayectoria

Para cargar los parámetros de la trayectoria se utiliza el comando LTRJ. el cual carga una palabra de control de trayectoria y un número variable de palabras de datos.

Al ejecutar el comando LTRJ se inicializan los parámetros de la trayectoria, inmediatamente después se envían dos bytes de datos, donde el primer byte programa con unos lógicos en las posiciones de bits correspondientes el modo de trayectoria (velocidad o posición), modo de dirección (modo velocidad) y el modo de paro (modo velocidad). El segundo byte, indica con unos lógicos en las posiciones de bits correspondientes,

cuales de los tres parámetros serán cargados al LM629, ademas de indicar si los parámetros son absolutos o relativos (ver Figura 3.14). Cualquier combinación de los tres parámetros pueden ser cargados con un solo comando LTRJ.

Inmediatamente después de la palabra de control de la trayectoria, los parámetros de la trayectoria son escritos. Cada parámetro es escrito en un par de bytes de datos, es decir, una palabra de datos. Debido a que se puede cargar cualquier combinación de los tres parámetros con un solo comando LTRJ, el numero de palabras de datos siguientes a la palabra de control pueden variar en un rango de cero a seis.

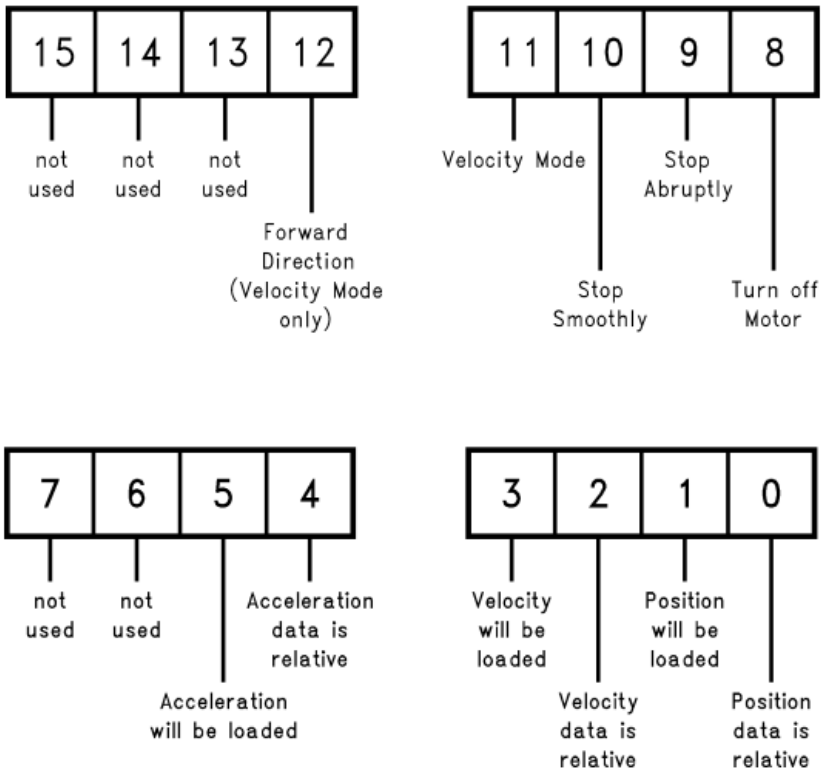


Figura 3.14: Asignación de bits del control de la trayectoria [7].

Iniciar el control de movimiento

El comando para iniciar el control de movimiento (STT), transfiere los nuevos parámetros de la trayectoria desde el buffer de entrada a los registros de trabajo y comienza con la ejecución de la nueva trayectoria. Hasta que se ejecute el comando STT, los

nuevos parámetros de la trayectoria no afectarán al movimiento del motor.

3.1.5.5. Módulo de paro

Este módulo muestra la programación que se debe realizar para detener el movimiento del motor.

Cuando el LM629 opera en modo posición, el paro del motor es lento y ocurre automáticamente una vez terminada la trayectoria especificada. (no necesita de un módulo de paro). Sólo se puede utilizar un modulo de paro prematura bajo condiciones excepcionales.

Cuando el LM629 opera en modo velocidad, el paro siempre es llevado a cabo por un modulo de paro.

Los bits del 8 al 10 de la palabra de control de la trayectoria seleccionan el modo de paro. Ver Figura 3.14.

Estableciendo el bit 8 en la palabra de control de la trayectoria se selecciona el modo motor apagado como método de paro deseado. Este modo de paro detiene el movimiento del motor estableciendo la señal de motor en cero.

Estableciendo el bit 9 en la palabra de control de la trayectoria se selecciona el modo de paro instantáneo como método de paro deseado. Este modo detiene el motor (con la máxima desaceleración) estableciendo la posición deseada igual a la posición actual.

Estableciendo el bit 10 en la palabra de control de la trayectoria se selecciona el modo de paro lento como método de paro deseado. Este modo detiene el motor utilizando la desaceleración programada por el usuario.

3.2. Arduino Mega ADK

Debido a la aplicación del sistema y de sus características propias, es necesario utilizar un microcontrolador que proporcione un procesamiento eficaz de datos y de control,

entre otras utilidades como un número amplio de puertos para la entrada y salida de datos, así como una interfaz de comunicación serial.

Para el sistema implementado, existe una amplia gama de microcontroladores (Microchip, Motorola, Intel, etc.) que ofrecen las características necesarias anteriormente mencionadas. Se seleccionó el Arduino Mega ADK ya que a diferencia de los demás, este posee un microcontrolador capaz de ofrecer un número amplio de puertos de datos así como la integración de una interfaz de comunicación serial sin utilizar circuitos integrados extras, ademas de su sencilla utilización y su amplia capacidad de ser programado en lenguaje C desde otro entorno de programación.

El Arduino Mega ADK posee las siguientes características:

- Microcontrolador ATmega2560
- Microcontrolador ATmega8U2 (programado como convertidor USB a serie)
- Dispone de 54 Entradas/Salidas digitales (14 salidas PWM)
- 16 entradas analógicas
- 4 UARTS (puertos seriales)
- Velocidad de reloj de trabajo de 16 MHz
- Conexión USB
- Conexión ICSP
- Pulsador para reiniciar

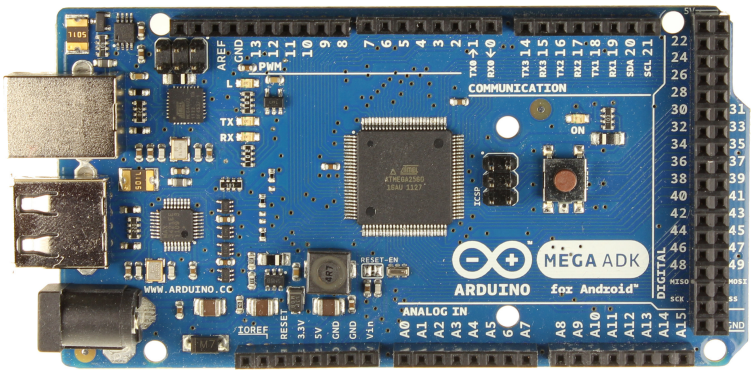


Figura 3.15: Arduino Mega ADK [10].

3.2.1. Alimentación

El Arduino puede alimentarse por USB o por una fuente externa. Si se llegase alimentar de ambas maneras el origen de la alimentación es seleccionado automáticamente.

El Arduino puede trabajar con un voltaje externo de entre 5.5V y 16V. Si el voltaje suministrado es menor a 7V, el pin de 5V puede proporcionar menos de 5V y el Arduino puede trabajar incorrectamente. Si se aplican mas de 12V, se puede sobrecalentar el regulador de voltaje y dañar el Arduino, por lo que lo recomendable es utilizar un voltaje en el rango de entre 7V y 12V.

Los pines de alimentación son los siguientes:

- VIN: entrada para fuente de alimentación externa.
- 5V: salida para obtener 5V del Arduino. Utiliza el voltaje suministrado por VIN o por la conexión USB.
- 3V: salida para obtener 3V del Arduino. Se obtiene este voltaje por medio del regulador de voltaje.
- GND: pin que va a tierra.

3.2.2. Memoria

El microcontrolador ATmega2560 tiene 256 KB de memoria flash para guardar codigo, 8 KB de memoria SRAM y 4 KB de memoria EEPROM.

3.2.3. Entradas y salidas

Cada uno de los 54 pines digitales del Arduino Mega pueden utilizarse como entradas o como salidas y operan a 5 volts. Cada pin puede proporcionar o recibir una corriente máxima de 40mA y tienen una resistencia interna pull-up de 20-50KΩ. También posee 16 entradas analógicas, y pueden proporcionar una resolución de 10 bits (1024 valores). Algunos de estos pines tienen funciones especiales, descritas a continuación:

- **Serie: 0 (RX) y 1 (TX); Serie 1: 19 (RX) y 18 (TX); Serie 2: 17 (RX) y 16 (TX); Serie 3: 15 (RX) y 14 (TX)**: son utilizados pata la transmisión (TX) y recepción (RX) de datos a través del puerto serie TTL. Los pines Serie: 0 (RX) y 1 (TX) están conectados al chip FTDI USB-to-TTL.
- **Interrupciones Externas: 2 (interrupción 0), 3 (interrupción 1), 18 (interrupción 5), 19 (interrupción 4), 20 (interrupción 3), y 21 (interrupción 2)**: son utilizados para configurar interrupciones en un valor de cero lógico o en uno lógico.
- **PWM, de 0 a 13**: proporcionan una salida PWM de 8 bits de resolución (valores de 0 a 255).
- **SPI: 50 (SS), 51 (MOSI), 52 (MISO), 53 (SCK)**: proporcionan comunicación SPI.
- **USB Host: MAX3421E**: comunica el Arduino Mega con el bus SPI.
- **LED, 13**: LED integrado en el Arduino Mega conectado al pin 13, cuando este pin tiene un uno lógico (5V) el LED se enciende y cuando tiene un cero lógico (0V) el LED se apaga.
- **I2C: 20 (SDA) y 21 (SCL)**: soporte para el protocolo de comunicaciones I2C (TWI).

- **AREF**: voltaje de referencia para las entradas analógicas.
- **Reset**: suministra un valor de cero lógico para reiniciar el microcontrolador.

3.2.4. Comunicaciones

Una de las ventajas que ofrece el Arduino Mega ADK es que facilita en muchos aspectos la comunicación entre el microcontrolador y la computadora, ya que nos proporciona cuatros puertos de comunicación de tipo serie UART TTL.

El ATmega8U2 integrado en el Arduino Mega realiza esta comunicación serie a través del puerto USB y los controladores proporcionan un puerto serial virtual en la computadora.

El ATmega2560 también soporta la comunicación I2C (TWI) y SPI.

3.2.5. Programación

El Arduino Mega se puede programar utilizando el software de Arduino. Pero también se pueden utilizar otros IDE como lo son Atmel Studio 7, con el cual se puede hacer una programación totalmente nativa en lenguaje C o en lenguaje ensamblador, aprovechando varios beneficios, algunos de ellos como la selección del reloj de trabajo, la utilización de Timers, la reducción del tiempo de ejecución entre lineas del código, entre algunos otros beneficios.

3.3. Motor Maxon

El motor utilizado en el sistema es de la marca Maxon A-max 16 (Figura 3.16), el cual es de corriente directa y que fue seleccionado debido a que es un motor donde el coeficiente de fricción es casi nulo, tiene ambivalencia de velocidades y por que se puede implementar un control mas sencillo integrando con el un encoder, el cual es detallado en la siguiente sección. Además es un motor que no exige una demanda muy alta de corriente y funciona hasta con 12V como máximo.

Figura 3.16: Motor Maxon A-max 16 [11].

El motor Maxon A-max 16 tiene los siguientes valores a un voltaje nominal:

Voltaje Nominal (V)	12V
Velocidad sin carga (rpm)	10400
Corriente sin carga (mA)	9.69
Velocidad nominal (rpm)	4900
Torque nominal (torque máximo continuo) (mNm)	2.16
Corriente nominal (corriente continua máxima) (A)	0.21
Torque con carga máxima (mNm)	4.13
Corriente con carga máxima (A)	0.386

Tabla 3.9: Valores del motor Maxon A-max 16 a voltaje nominal [14].

Y tiene las siguientes características utilizadas en la ecuación de control:

Resistencia en terminales (Ω)	31.1
Inductancia en terminales (mH)	1.13
Torque Constante (mNm/A)	10.7
Velocidad Constante (rpm/V)	893
Gradiente de torque/velocidad (rpm/mNm)	2600
Constante de tiempo mecánico (ms)	23.3
Inercia del rotor (gcm^2)	0.857

Tabla 3.10: Características del motor Maxon A-max 16 [14].

3.4. Encoder óptico de cuadratura MR Tipo M

Un encoder es un dispositivo utilizado en sistemas de control de lazo cerrado como componente de retroalimentación, monitoreando constantemente la posición de un eje

giratorio y generando señales digitales (ver Figura 3.17) en respuesta al movimiento de rotación de dicho eje.

Un encoder se utiliza especialmente para medir la velocidad o la posición con extrema precisión de un sistema o alguna aplicación, por mencionar algunas, un sistema robótico, un plotter, una máquina de ensamblaje, o una máquina de manufactura, como una fresadora CNC.

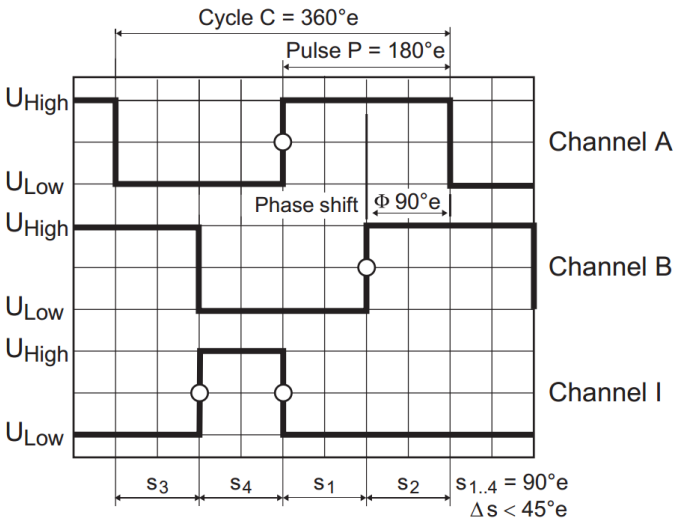


Figura 3.17: Señales en cuadratura del encoder [12].

El encoder utilizado en el sistema es de tipo incremental de la marca Maxon el cual tiene las siguientes características:

Cuentas por revolución	512
Numero de canales	3
Frecuencia máxima de operación (KHz)	320
Velocidad máxima (rpm)	37,500
Voltaje de alimentación (Vcc)	5V ± 5 %
Señal de salida	TTL
Cambio de fase	90že ± 45že
Ancho del pulso indexado	90že ± 45že
Rango de temperatura	-25žC a 85žC
Corriente de salida por canal	5 mA Máx.

Tabla 3.11: Características del encoder [12].

El encoder fue seleccionado por tener 512 cuentas o pulsos por revolución ya que lo hace muy viable para poder obtener un buen control de posición en el sistema, ademas de la velocidad máxima de operación que puede soportar. Al utilizar un motor con encoder integrado como lo es en este caso, es importante verificar la configuración de pines ya que puede dañarse al hacer una mala conexión, en este caso se presenta esta configuración (ver Figura 3.18), donde los pines 1 y 4 son exclusivos del motor y van conectados a la salida que otorga el puente H LMD18201T.

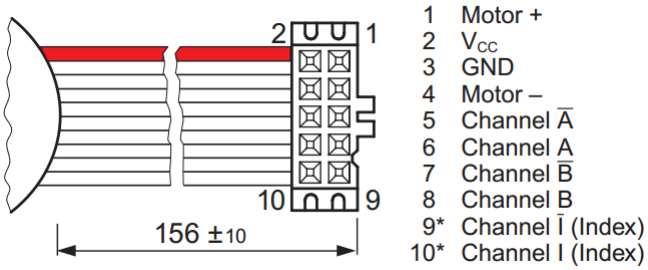


Figura 3.18: Configuración de pines del encoder [12].

También es importante tener en cuenta que no sólo es suficiente con conectar las salidas de las señales en cuadratura (A y B) y el índice del pulso ($\overline{IN}$) del encoder directamente a las entradas (Pines 1,2 y 3) del LM629, deben ser conectadas primeramente a un receptor de línea diferencial cuádruple, en este caso un AM26LS32AC, el cual cuenta con una alta impedancia de entrada e histéresis que ayuda a incrementar la inmunidad al ruido, obteniendo señales limpias y con su respectivo voltaje debido a una sensibilidad de entrada de ±200mV.

3.5. Puente H LMD18201T

Para seleccionar la etapa de potencia debe tomarse en cuenta algunas características del motor que se está utilizando como lo es el voltaje que maneja y la corriente máxima a la cual éste puede llegar a utilizar en funcionamiento, también debe considerarse que la etapa de potencia seleccionada debe ser capaz de interpretar la señal PWM que le otorga el LM629.

En este caso el motor Maxon A-max 16 no utiliza mucha corriente ya que su corriente máxima alcanza los 386 mA y trabaja con 12V como valor límite, por ello se seleccionó el puente H LMD18201T de la marca National Semiconductor, el cual puede llegar a manejar corrientes de 3A, soporta corrientes pico de salida de hasta 6A, permite entrada de señales PWM con signo; además es diseñado para aplicaciones de control de movimiento, haciéndolo ideal para utilizarlo en motores de corriente directa, como se requiere en este sistema.

Algunas características del puente H LMD18201T son las siguientes:

- Entrega una corriente de salida continua de hasta 3A.
- Opera con un voltaje de alimentación de hasta 55V.
- Entradas compatibles TTL y CMOS.
- Protección a corto-circuitos.
- Bandera de advertencia térmica a 145žC.
- Apagado térmico (salidas apagadas) a 170žC.

También tiene más aplicaciones, en la cuales destacan las siguientes:

- Soporte para motores de corriente directa y de paso.
- Servomecanismos de posición y velocidad.
- Automatización de robots industriales.
- Máquinas de control numérico.
- Impresoras y plotters.

A continuación se muestra el diagrama de bloques funcional del puente H LMD18201T (Figura 3.19), utilizado para accionar el motor.

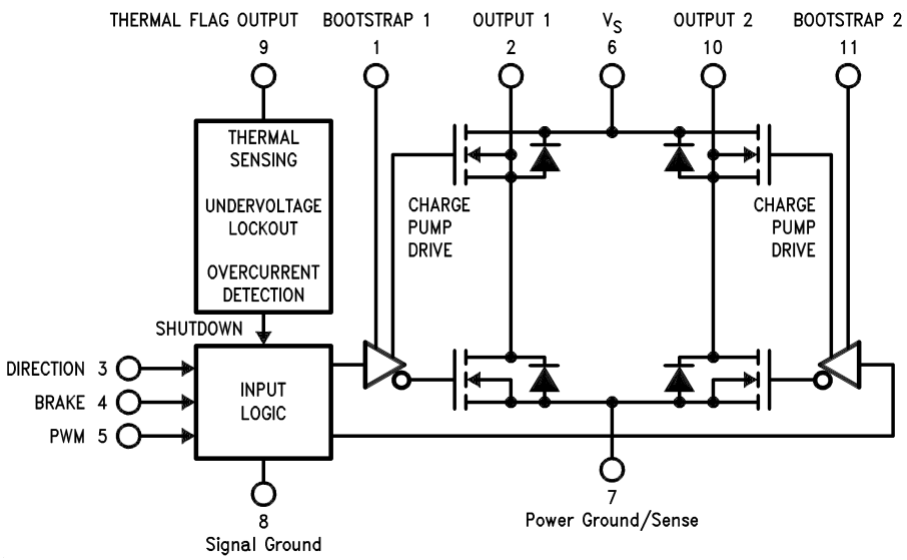


Figura 3.19: Diagramas de bloques del LMD18201T [13].

3.5.1. Descripción de pines

En la Figura 3.20 se puede apreciar la distribución de los pines del puente H LMD18201T. La función de cada pin se explica a continuación:

Pin 1, Entrada Bootstrap 1: Este pin es utilizado para colocar entre los pines 1 y 2 un capacitor de arranque (10 nF).

Pin 2, Salida 1: Salida 1 del puente H.

Pin 3, Entrada de dirección: Este pin de entrada controla la dirección del flujo de la corriente entre la salida 1 y la salida 2 (pines 2 y 10), y por consecuencia, direccionar la rotación del motor.

Pin 4, Entrada de paro: Este pin es usado para frenar el motor cortocircuitando efectivamente sus terminales.

Pin 5, Entrada para PWM: Este pin recibe la señal PWM del LM629.

Pin 6, Voltaje de alimentación: Este pin recibe el voltaje de alimentación, que puede ir desde los 12V a los 55V.

Pin 7, Conexión a tierra: Este pin recibe la tierra de la fuente de alimentación.

Pin 8, Señal de tierra: Este pin recibe la tierra del circuito integrado utilizado para la conmutación del control del PWM.

Pin 9, Salida de bandera de advertencia térmica: Este pin otorga la señal de salida de la bandera de advertencia térmica.

Pin 10, Salida 2: Salida 2 del puente H.

Pin 11, Entrada Bootstrap 2: Este pin es utlizado para colocar entre los pines 10 y 11 un capacitor de arranque (10 nF).

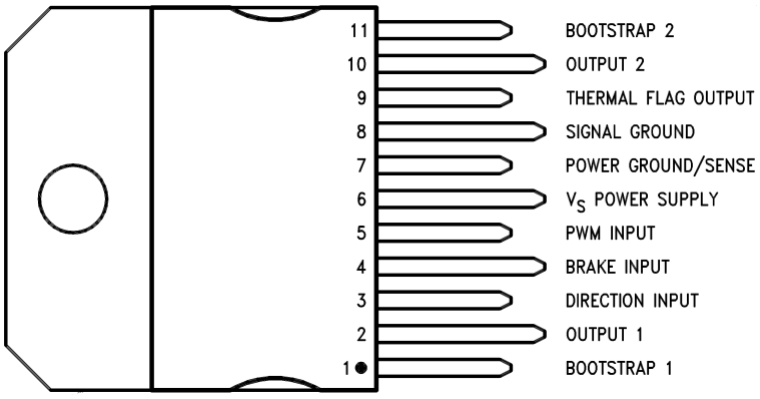


Figura 3.20: Distribución de pines del LMD18201T [13].

3.5.2. Modos de operación

El puente H tiene distintos modos de operación los cuales se describen en la siguiente tabla, en la cual se muestra las combinaciones que se pueden permitir entre las entradas (PWM, dirección, freno).

PWM	Dirección	Freno	Salida Activa
H	H	L	Source 1, Sink 2
H	L	L	Sink 1, Source 2
L	X	L	Source 1, source 2
H	H	H	Source 1, Source 2
H	L	H	Sink 1, Sink 2
L	X	H	None

Tabla 3.12: Tabla de verdad del LMD18201T.

En la Figura 3.21 se puede observar como el puente H realiza el control de velocidad por PWM que recibe del LM629, y que esta una vez captada el puente H otorgará al motor la velocidad deseada a travez de los pines 2 y 10.

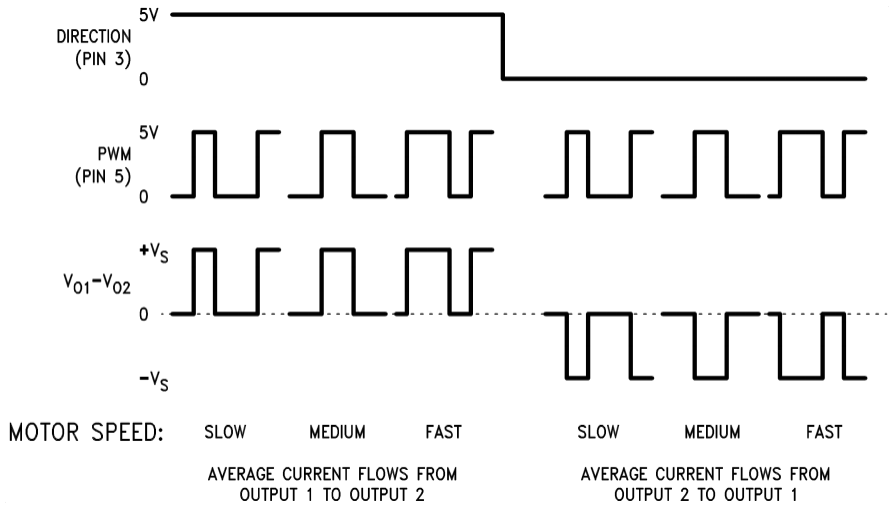


Figura 3.21: Control de dirección y magnitud del PWM [13].

La señal PWM de salida del LM629 esta en formato signo/magnitud como se puede ver en la Figura 3.20. El puente H LMD18201T recibe la señal del LM629, después de procesarla, entrega como salida una señal PWM con un valor de potencia acorde a las necesidades de la velocidad impuesta en la señal PWM del LM629.

3.6. Diagrama de flujo y código implementado

3.6.1. Diagrama de flujo

Para efectuar el control del LM629 por medio de un microcontrolador se utilizó el diagrama de flujos proporcionado por el reporte de aplicaciones AN-706 LM628/629 de Texas Instruments, el cual se muestra a continuación.

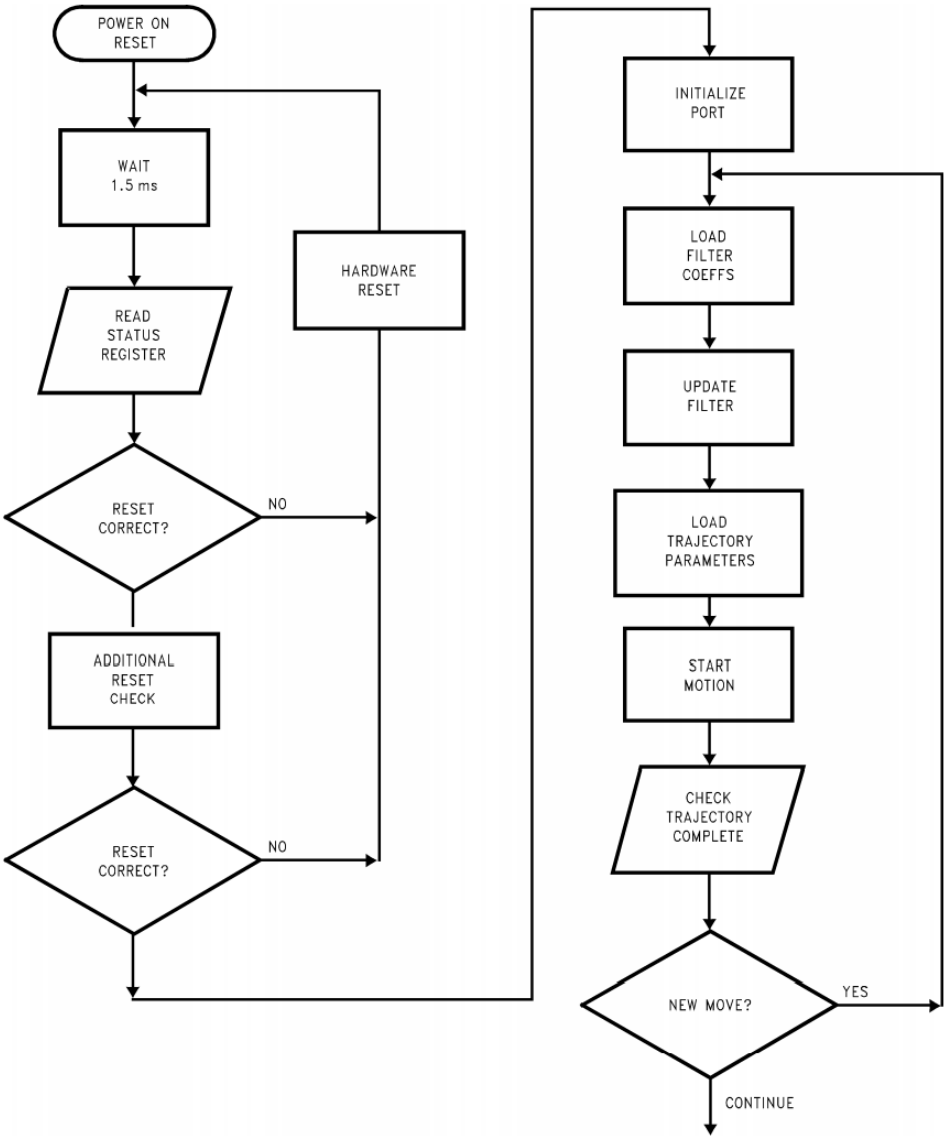


Figura 3.22: Diagrama de flujo básico para el LM629 [9].

Tomando en cuenta que el circuito integrado es el LM629 debe omitirse el paso de inicializar puerto ya que es propio del LM628, además se puede omitir el paso de verificar trayectoria completa ya que al utilizarse el LM629 en modo posición éste verifica automáticamente si ya se cumplió.

3.6.2. Código

El código fue realizado en lenguaje C para microcontroladores AVR, utilizando la IDE Atmel Studio 7. A continuación se muestra el código separado por módulo.

Algoritmo 3.1 Módulo de librerías y variables.

```
#define F_CPU 16000000UL

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

#define USART_BAUDRATE 9600
#define BAUD_PRESCALLER (((F_CPU / (USART_BAUDRATE * 16UL))) - 1)

//********** Timer **********//
#define TMR1 0 //for 8MHz: TMR1 = 0, for 6MHz: TMR1 = 0.33328
```

Algoritmo 3.2 Módulo de comandos para el LM629.

```
//********** LM629 Command Set **********//
#define RESET 0x00 //reset lm629
#define RSTI 0x1D //reset interrupts
#define DFH 0x02 //define home
#define SIP 0x03 //set index position
#define LPEI 0x1B //interrupt on error
#define LPES 0x1A //stop on error
#define SBPA 0x20 //set break point absolute
#define SBPR 0x21 //set break point relative
#define MSKI 0x1C //mask interrupts
#define LFIL 0x1E //load filter parameters
#define UDF 0x04 //update filter
#define LTRJ 0x1F //load trajectory
#define STT 0x01 //start motion
#define RDSIGS 0x0C //read signals register
#define RDIP 0x09 //read index position
#define RDDP 0x08 //read desired position
#define RDRP 0x0A //read real position
#define RDDV 0x07 //read desired velocity
#define RDRV 0x0B //read real velocity
#define RDSUM 0x0D //read integration sum
```

Algoritmo 3.3 Módulo de comunicación serial.

```
//********** Comunicacion Serial **********//
char String[]="LM629 Control";

void USART_init(void){
UBRR0H = (uint8_t)(BAUD_PRESCALLER>>8);
UBRR0L = (uint8_t)(BAUD_PRESCALLER);
//UCSR0B = (1<<RXEN0)|(1<<TXEN0);
//UCSR0C = (3<<UCSZ00);
/* Enable receiver and transmitter */
UCSR0B = (1<<RXEN1)|(1<<TXEN1)|(0<<UCSZ12)|(1<<UDRIE1) |(1<<RXCIE1) ;
/* */
UCSR0C = (1<<UCSZ10)|(1<<UCSZ11) |(1<<UPM11)|(0<<UPM10)|(0<<USBS1) ;
}

unsigned char USART_receive(void){
while(!(UCSR0A & (1<<RXC0)));
return UDR0;
}

void USART_send( unsigned char data){
while(!(UCSR0A & (1<<UDRE0)));
UDR0 = data;
}

void USART_putstring(char* StringPtr){
while(*StringPtr != 0x00){
USART_send(*StringPtr);
StringPtr++;}
}
```

Algoritmo 3.4 Módulo de variables de control.

```
//Control
#define Control_PortC() DDRC=0xEF; PORTC=0xFF; // PORTC Output... PORTC Set

#define RSTL PORTC&=~(1<<PC5); // RST Low -> Enabled
#define RSTH PORTC|=(1<<PC5); // RST High -> Disabled

//CS=L PS=L RD=L WR=H (Read Status)
#define RD_StatusByte() PORTC&=~(1<<PC0);
PORTC&=~(1<<PC3);
PORTC&=~(1<<PC1);
PORTC|=(1<<PC2);

//RD
#define RDL PORTC&=~(1<<PC1); // RD Low -> Enabled
#define RDH() PORTC|=(1<<PC1); _delay_us(30); // RD High -> Disabled

//CS=L PS=H RD=L WR=H (Read Data)
#define RD_Data() PORTC&=~(1<<PC0);
PORTC|=(1<<PC3);
PORTC&=~(1<<PC1);
PORTC|=(1<<PC2);}

//CS=L PS=L WR=L RD=H (Write Command)
#define WR_Command() PORTC&=~(1<<PC0);
PORTC&=~(1<<PC3);
PORTC&=~(1<<PC2);
PORTC|=(1<<PC1);

//CS=L PS=H WR=L RD=H (Write Data)
#define WR_Data() PORTC&=~(1<<PC0);PORTC|=(1<<PC3); PORTC&=~(1<<PC2);
PORTC|=(1<<PC1);

//WR
#define WRL PORTC&=~(1<<PC2); // WR Low -> Enabled
#define WRH() _delay_us(30); PORTC|=(1<<PC2); // WR High -> Disabled

//Databus Input/Output
#define DatabusIn() DDRA=0x00; PORTA=0x00; //PORTA Input... PORTA Set
#define DatabusOut() DDRA=0xFF; PORTA=0x00; //PORTA Output... PORTA Set
```

Algoritmo 3.5 Módulo de lectura del Status Byte.

```
unsigned char Status_Byte() // Lee Status Byte
{
unsigned char readbyte;
Control_PortC();
RD_StatusByte();
DatabusIn();
readbyte=PINA;
DatabusOut();
RDH();
return readbyte;
}
```

Algoritmo 3.6 Módulo de revisión del Busy Bit.

```
void Check_BusyBit()
{while(Status_Byte() & 0x01);
}
```

Algoritmo 3.7 Módulo del reinicio por hardware.

```
void Reset_LM629()
{
Control_PortC();
RSTL;
_delay_ms(2);
RSTH;
_delay_ms(2);
}

void Initialization_LM629()
{
unsigned char stb, stb2;
Reset_LM629();
stb=Status_Byte();
USART_send(stb);
if(stb==0xC4 || stb==0x84)
{
MSKI_LM629(0x0000); //
RSTI_LM629(0x0000); //
_delay_ms(2);
stb2=Status_Byte();
USART_send(stb2);
if(stb2==0xC0 || stb2==0x80)
{
Check_BusyBit();
_delay_ms(500);
}
else
{Initialization_LM629();}
}
else
{Initialization_LM629();}
}
```

Algoritmo 3.8 Módulo de enmascaramiento de los bits del Status Byte.

```
void MSKI_LM629(unsigned int mask) //
{
Control_PortC();
DatabusOut();
PORTA=MSKI;
WR_Command();
WRH();
Check_BusyBit();
PORTA=(unsigned char)((mask&0xFF00)>>8);
WR_Data();
WRH();
PORTA=(unsigned char)(mask&0x00FF);
WR_Data();
WRH();
Check_BusyBit();
}
```

Algoritmo 3.9 Módulo de reinicio de los bits del Status Byte.

```
void RSTI_LM629(unsigned int rst) //
{
Control_PortC();
DatabusOut();
PORTA=RSTI;
WR_Command();
WRH();
Check_BusyBit();
PORTA=(unsigned char)((rst&0xFF00)>>8);
WR_Data();
WRH();
PORTA=(unsigned char)(rst&0x00FF);
WR_Data();
WRH();
Check_BusyBit();
}
```

Algoritmo 3.10 Módulo de carga de los coeficientes del filtro PID.

```
void LFIL_LM629(unsigned int cmd, unsigned int kp, unsigned int ki, unsigned int kd, unsigned char il) //Load Filter
Parameters
{
Control_PortC();
DatabusOut();
PORTA=LFIL;
WR_Command();
WRH();
Check_BusyBit();
//Command for Derivative Sampling Interval & Select Load Filters
USART_send(cmd);
PORTA=(unsigned char)((cmd&0xFF00)>>8);
WR_Data();
WRH();
PORTA=(unsigned char)(cmd&0x00FF);
WR_Data();
WRH();
Check_BusyBit();
//Kp
PORTA=(unsigned char)((kp&0xFF00)>>8);
WR_Data();
WRH();
PORTA=(unsigned char)(kp&0x00FF);
WR_Data();
WRH();
USART_send(kp);
Check_BusyBit();
//Ki
PORTA=(unsigned char)((ki&0xFF00)>>8);
WR_Data();
WRH();
PORTA=(unsigned char)(ki&0x00FF);
WR_Data();
WRH();
Check_BusyBit();
//Kd
PORTA=(unsigned char)((kd&0xFF00)>>8);
WR_Data();
WRH();
PORTA=(unsigned char)(kd&0x00FF);
WR_Data();
WRH();
Check_BusyBit();
//il
PORTA=(unsigned char)((il&0xFF00)>>8);
WR_Data();
WRH();
PORTA=(unsigned char)(il&0x00FF);
WR_Data();
WRH();
Check_BusyBit();
UDF_LM629();
}

void UDF_LM629() //Update Filter
{
Control_PortC();
DatabusOut();
PORTA=UDF;
WR_Command();
WRH();
Check_BusyBit();
}
```

Algoritmo 3.11 Módulo de carga de los parámetros de la trayectoria.

```
void LTRJ_LM629(unsigned int cmd, unsigned int long acc, unsigned long vel, signed long pos) //Load Trajectory
Parameters
{
Control_PortC();
DatabusOut();
PORTA=LTRJ;
WR_Command();
WRH();
Check_BusyBit();
//Trajectory mode(velocity or position), velocity mode direction, and stopping mode
PORTA=(unsigned char)((cmd&0xFF00)>>8);
WR_Data();
WRH();
PORTA=(unsigned char)(cmd&0x00FF);
WR_Data();
WRH();
Check_BusyBit();
if(cmd&0x0020) //Acceleration will be loaded
{
PORTA=(unsigned char)((acc&0xFF000000)>>24);
WR_Data();
WRH();
PORTA=(unsigned char)((acc&0x00FF0000)>>16);
WR_Data();
WRH();
Check_BusyBit();
PORTA=(unsigned char)((acc&0x0000FF00)>>8);
WR_Data();
WRH();
PORTA=(unsigned char)(acc&0x000000FF);
WR_Data();
WRH();
Check_BusyBit();
}
if(cmd&0x0008) //Velocity will be loaded
{
PORTA=(unsigned char)((vel&0xFF000000)>>24);
WR_Data();
WRH();
PORTA=(unsigned char)((vel&0x00FF0000)>>16);
WR_Data();
WRH();
Check_BusyBit();
PORTA=(unsigned char)((vel&0x0000FF00)>>8);
WR_Data();
WRH();
PORTA=(unsigned char)(vel&0x000000FF);
WR_Data();
WRH();
Check_BusyBit(); //
}
```

Algoritmo 3.12 Continuación Módulo de carga de los parámetros de la trayectoria.

```
if(cmd&0x0002) //Position will be loaded
{
PORTA=(unsigned char)((pos&0xFF000000)>>24);
WR_Data();
WRH();
PORTA=(unsigned char)((pos&0x00FF0000)>>16);
WR_Data();
WRH();
Check_BusyBit();
PORTA=(unsigned char)((pos&0x0000FF00)>>8);
WR_Data();
WRH();
PORTA=(unsigned char)(pos&0x000000FF);
WR_Data();
WRH();
Check_BusyBit();
}
}
```

Algoritmo 3.13 Módulo de inicio de movimiento.

```
void STT_LM629() //Start Motion
{
Control_PortC();
DatabusOut();
PORTA=STT;
WR_Command();
WRH();
Check_BusyBit();
}
```

Algoritmo 3.14 Módulo principal del programa.

```
int main(void){

unsigned char inicio;

DDRB=0x20;
//TIMER1 (16 bits) in mode TOP
TCCR1B |= (1 << CS10); //selecting prescaler 0b001 (Tclk/1)
TCCR1B &= ~((1<<CS12) | (1<<CS11)); // turn off CS12 and CS11 bits

TCCR1A |= ((1<<WGM11) | (1<<WGM10)); //Configure timer 1 for TOP mode (with TOP =
OCR1A)
TCCR1B |= ((1<<WGM13) | (1<<WGM12));

TCCR1A |= (1 << COM1A0); // Enable timer 1 Compare Output channel A in toggle mode
TCCR1A &= ~(1 << COM1A1);
TCNT1 = 0;

OCR1A = TMR1;

USART_init(); //Call the USART initialization code

//while(1){ //Infinite loop

//inicio=USART_receive();

//if(inicio=='i')
//{
Initialization_LM629();
LFIL_LM629(0x0008,0x0032,0x0000,0x0000,0x0000); //
LTRJ_LM629(0x002A,0x00000002,0x000174D2,0x0003E800);
STT_LM629();
//_delay_ms(10000);
//}
//}
return 0;
}
```

3.7. Esquemático PCB

El prototipo del sistema fue armado en un inicio en una placa de pruebas (protoboard), primeramente para conocer sus conexiones y poder realizar un diagrama electrónico con todos los componentes involucrados, poder realizar pruebas y además verificar su funcionalidad en conjunto con la etapa de potencia. Una vez que se obtuvo

el diagrama de conexiones final se procedió a diseñarlo en un programa especializado para hacer placas de circuito impreso (PCB's), con el fin de ahorrarnos espacio y utilizar en menor medida cables para conectar todos los componentes.

En este caso se utilizó el programa PCB Wizard 3.50, el cual es un programa con una interfaz sencilla y muy amigable para el usuario, tiene una amplia cantidad de componentes en formato DIP y en formato SMD, y especialmente tiene la función de poder obtener archivos Gerber, los cuales nos sirven para implementarlos en un programa CAM y poder maquinar nuestra placa en una fresadora CNC y así evitar el proceso tradicional de impresión por calor.

La placa del sistema fue maquinada en una fresadora CNC, para poder obtener un mejor acabado y evitar problemas como cortocircuitos, tierras incomunicadas, vías incompletas, entre otros problemas que se pueden presentar en las placas. Además se tiene un estilo mas profesional y una mejor presentación. A continuación se muestra el diseño realizado en PCB Wizard y la placa obtenida después del maquinado.

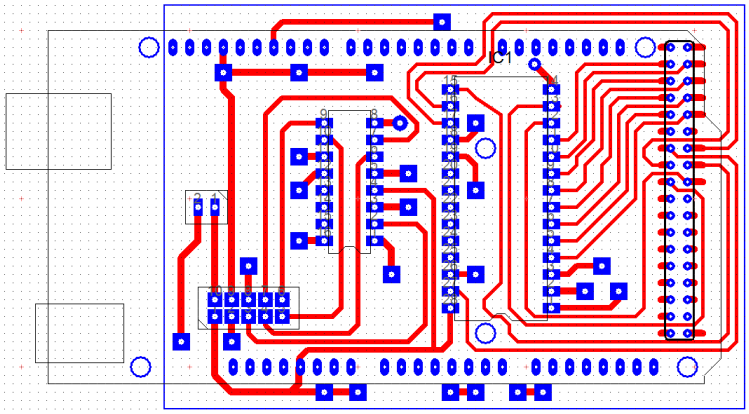


Figura 3.23: Diseño en PCB Wizard.

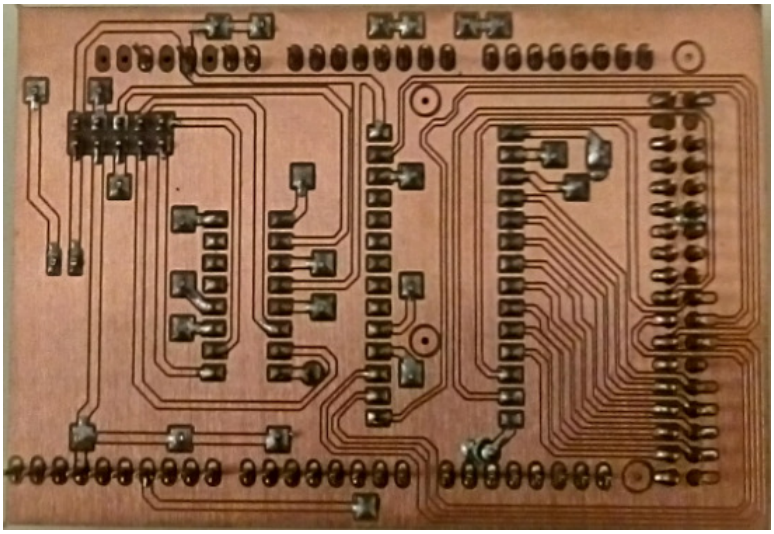


Figura 3.24: PCB maquinada.

Capítulo 4

Pruebas y resultados

En este capítulo se presentan las pruebas aplicadas al controlador de movimiento LM629 en conjunto con los elementos utilizados para crear el sistema de control en lazo cerrado.

Para la aplicación de estas pruebas fue necesario el uso del módulo VirtualBench de National Intruments que posee varias herramientas que facilitan el desarrollo del proyecto. Las funciones utilizadas fue la de un generador de funciones, para generar una señal de reloj de prueba de 8MHZ que activará el funcionamiento del LM629, esta señal de reloj después se obtuvo del microcontrolador. Se utilizó el módulo como fuente de alimentación de 5V para el LM629, el microcontrolador Arduino Mega ADK y como fuente de alimentación de 12V para el motor Maxon A-max 16. Se hizo uso de la herramienta como Osciloscopio para revisar las señales cuadradas que nos otorgaba el encoder y verificar que la señal de reloj de por parte del microcontrolador Arduino Mega ADK fuese cuadrada y de 8MHz.

4.1. Inicializando el LM629

Para comprobar la comunicación del LM629-micorcontrolador-cpu fue necesario el uso del programa *Terminal v1.93b*, el cual por medio de comunicación Serial recibe la información que el LM629 envía después de recibir algún dato o comando por parte del microcontrolador Arduino. Para ello se alimentó al LM629 y al microcontrolador con 5V, y se utilizó la señal de reloj del microcontrolador de 8MHz en el LM629 (ver figura

4.1), esta señal de reloj es utilizada para establecer los tiempos de espera entre el recibo y envío de datos y comandos, que debe de ser de al menos de 8 ciclos de reloj.

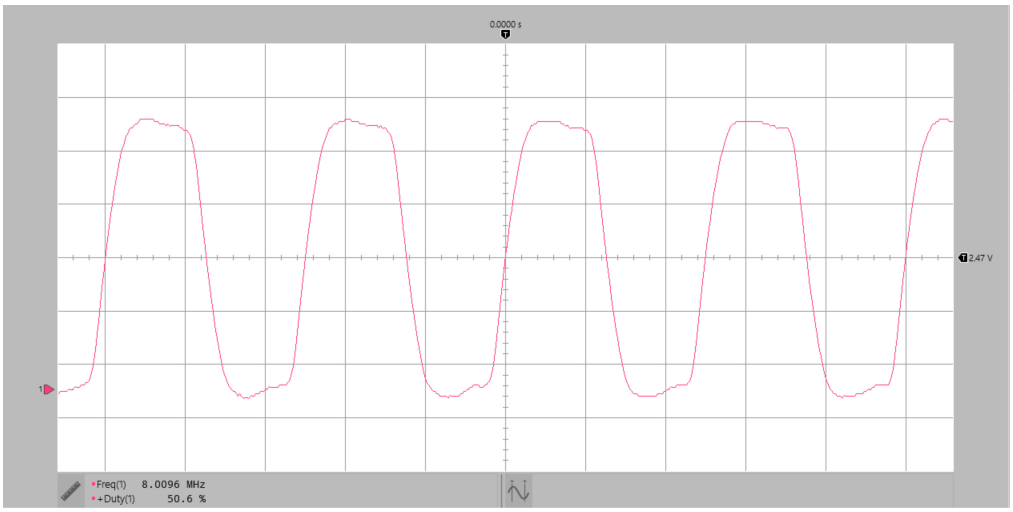


Figura 4.1: Señal de reloj de 8MHz .

La programación utilizada en este paso es la descrita en la sección 3.1.5.2, y en la figuras 4.2 y 4.3, se observa que los datos enviados por el LM629 y leídos por medio del "Status byte" indican que el controlador se ha inicializado correctamente.

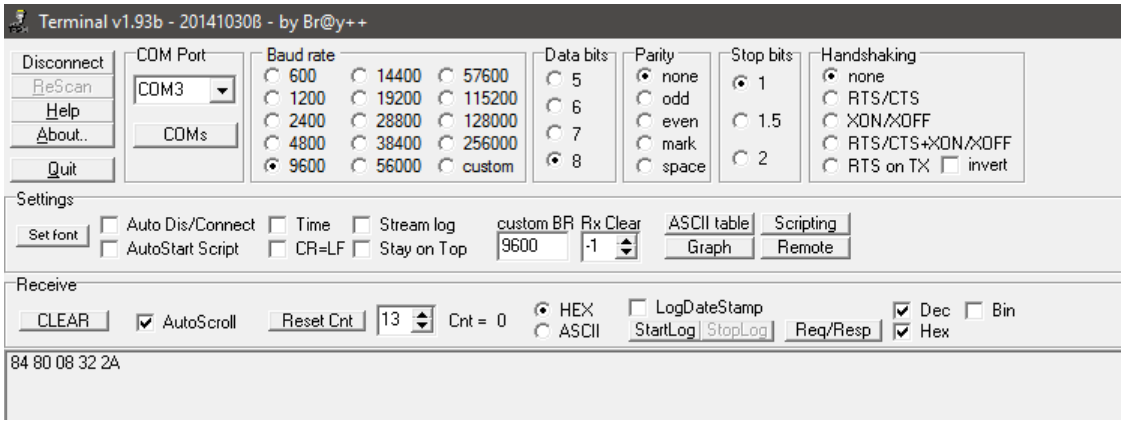


Figura 4.2: Respuesta de inicialización del LM629 .

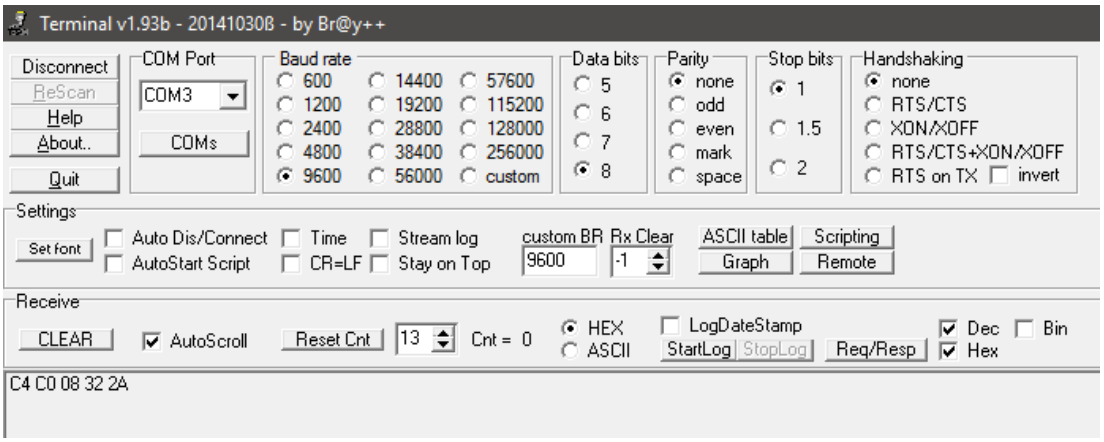


Figura 4.3: Respuesta de inicialización del LM629 (variante) .

En ambas imágenes la inicialización del LM629 es correcta, solo que en ocasiones el LM629 no enviará los mismos valores hexadecimales, esto se puede observar en los dos primeros pares de números hexadecimales. En la figura 4.2 se muestra la pareja de datos [84 80] y en la figura 4.3 [$C4\,C0$], puede ocurrir que la combinación sea [$84\,C0$] ó [$C4\,80$] de igual forma el controlador seguirá inicializando correctamente, indicando que se ha reiniciado todos los parámetros y esta listo para recibir los valores de la trayectoria y del filtro PID, tal como se muestra en la figura 3.9.

4.2. Carga de trayectoria y parámetros del filtro PID

Una vez inicializado el controlador LM629, se prosiguió a cargar los parámetros del filtro PID y de la trayectoria.

Los parámetros de la trayectoria se determinaron para un trayectoria de control de posición absoluta, que cumpliera los siguientes requisitos:

- Movimiento de 2 revoluciones
- Aceleración y desaceleración de 1 rev/seg^2
- Velocidad máxima de 2 rev/seg

Como se esta utilizando un reloj de 8MHZ nuestro tiempo de muestreo es de 256μs., y en base a la tabla 3.5 se cargará al LM629 el valor 0000 0000 hexadecimal para establecer este tiempo de muestreo.

Para determinar los parámetros de aceleración, velocidad y posición se utilizaron los cálculos mostrados como ejemplo en la sección 3.1.3.3, obteniendo los siguientes valores.

Posición:

$$P = 4,096\, pulsos$$

$$P = 0000\,1000\, Hex$$

Velocidad:

$$V = 68,719\, \frac{pulsos}{muestra}$$

$$V = 0001\,0C6F\, Hex$$

Aceleración:

$$A = 9\, \frac{pulsos}{muestra^2}$$

$$A = 0000\,0009\, Hex$$

En la figura 4.2 se indica en el quinto par el numero hexadecimal $2A$ indicando que los valores de aceleración, velocidad y posición fueron cargados al LM629 en modo absoluto esto en base a la figura 3.14.

En los parámetros del filtro PID solo se utilizó la acción de control proporcional (P) debido a la falta de herramientas con las que se pudiera analizar mejor el comportamiento del motor al iniciar el programa. Este parámetro fue determinado experimentalmente y por medio de observación se seleccionó el valor, tomando en cuenta el comportamiento

del motor al girar. Los valores probados fueron los siguientes:

- En la primer prueba se utilizó el valor de 1 (0001 Hex). Se observó que el motor da una vuelta y un tercio de la otra, no logrando completar las dos vueltas establecidas.
- En la segunda prueba se utilizó un valor de 5 (0005 Hex). Se observó que termina a menos de 90° en completar las dos vueltas.
- En la tercer prueba se utilizó un valor de 10 (000A Hex). Con este valor casi se obtiene el objetivo de dos vueltas, sin embargo cada vez que gira va terminando aproximadamente unos 10° antes del objetivo.
- En la cuarta prueba se utilizó un valor de 25 (0019 Hex). Con este valor fue posible acercarse mas al objetivo, pero cada vez terminaba aproximadamente un poco antes del objetivo.
- En la quinta prueba se utilizó un valor de 50 (0032 Hex). Para ver si este valor era fiable se realizaron cinco intentos, el motor logró posicionarse exactamente en las cinco ocasiones en la posición de origen al dar las dos vueltas.

El valor utilizado de 50 en la acción de control proporcional (P) puede ser comprobado en la figura 4.2 donde en el cuarto par de números se observa el valor hexadecimal 32, ademas en el tercer par se observa el valor 08 hexadecimal, indicando que solamente la acción de control proporcional fue cargado al LM629, esto en base a la figura 3.12.

Capítulo 5

Conclusiones

En este capitulo se presentan las aportaciones y conclusiones del desarrollo e implementación del trabajo realizado. Así como un listado de trabajos a futuro que se pueden realizar a partir de las aportaciones de este trabajo.

5.1. Conclusiones y aportaciones

En este trabajo se logró implementar con éxito el controlador de movimiento LM629 como sistema de control digital de posición PID para motores de corriente directa , creando una opción de bajo costo con características similares a los controladores existentes en el mercado, con la capacidad de ser adaptado a cualquier sistema o maquinaria, donde se vea involucrado el control critico de los motores de corriente directa, haciéndola muy versátil.

Aunque solamente se pudo implementar la acción de control proporcional debido a la falta de un tacómetro para verificar la velocidad y graficarla y poder realizar un análisis de sobrepaso al momento del arranque del motor, los resultados en las pruebas que se realizaron con el sistema fueron satisfactorias ya que se pudo lograr el objetivo de cargar una trayectoria y el motor cumpliera con dichos parámetros.

También se logró implementar una etapa de potencia utilizando el LMD18201T, a pesar de que el motor Maxon A-max 16 no necesita mucha corriente, esta etapa de potencia utilizada puede ser capaz de soportar motores de hasta 3A. A la etapa de po-

tencia se le desarrolló una placa de circuito impreso al igual que al controlador LM629.

Con el fin de que este trabajo pueda ser utilizado en otros proyectos, se agregó el código utilizado debido a que casi no existe información detallada sobre que debe considerarse a la hora de hacer la programación. Un problema muy común en la programación del LM629 es que debe considerarse el tiempo de espera que dura en procesar los datos y comandos enviados y/o recibidos, ya que posee un tiempo de sincronización de al menos 30μs, el no considerarlo generó que la información enviada al LM629 no fuera interpretada correctamente, haciendo que nunca se inicializara el controlador.

5.2. Trabajos futuros

Con la aportación que ha dejado este trabajo se han establecido a futuro el desarrollo de los siguientes temas:

- Desarrollo de una interfaz en Labview.
- Sintonización PID del controlador LM629
- Desarrollo de aplicaciones robóticas, como un robot de un grado de libertad tipo planar o manipulador para analizar el desempeño del PID.
- Probar robustez ante perturbaciones.
- Decodificación de la posición mediante un decodificador de cuadratura dedicado para analizar el desempeño.

Apéndice A

Cálculos

En esta sección se muestran los cálculos realizados para los valores mostrados en la sección 4.2.

A.1. Cálculos de trayectoria para control de movimiento de posición absoluta

Realizar el movimiento de 2 revoluciones desde la posición de inicio tomando en cuenta las siguientes especificaciones:

- $Aceleración\, y\, desaceleración = 1\frac{rev}{seg^2}$
- $Velocidad\, máxima = 2\frac{rev}{seg}$
- $Señal\, de\, reloj = 8MHz$

Estableciendo el tiempo de muestro

$$T_s = 2048\left(\frac{1}{CLK}\right)$$

$$T_s = 2048\left(\frac{1}{8MHz}\right)$$

$$T_s = 256x10^{-6}\frac{seg}{muestra}$$

$$T_s = 256\mu s$$

Calculando pulsos del encoder

$$R = líneas\,del\,encoder * 4(resolucion\,del\,sistema)$$

$$R = 512 * 4$$

$$R = 2,048\,\frac{pulsos}{rev}$$

Calculando Posición

$$P = R * revoluciones\,deseadas$$

$$P = 2,048 * 2$$

$$P = 4,096\,pulsos$$

$$P = 0000\,1000\,Hex$$

Calculando Velocidad

$$V = R * T_s * velocidad\,máxima$$

$$V = 2048\frac{pulsos}{rev} * 256x10^{-6}\frac{seg}{muestra} * 2\frac{rev}{seg}$$

$$V = 1.0486\frac{pulsos}{muestra}$$

Escalando el valor a la resolución

$$V = 1.0486 * 65536 = 68,719.47$$

Redondeando

$$V = 68,719$$

$$V = 0001\,0C6F\,Hex$$

Calculando Aceleración

$$A = R * T_s^2 * aceleración\,deseada$$

$$A = 2048\frac{pulsos}{rev} * \left(256x10^{-6}\frac{seg}{muestra}\right)^2 * 1\frac{rev}{seg^2}$$

$$A = 134.2177x10^{-6}\frac{pulsos}{muestra^2}$$

Escalando el valor a la resolución

$$A = 134.2177x10^{-6} * 65536 = 8.7960$$

Redondeando

$$A = 9$$

$$A = 0000\,0009\,Hex$$

Apéndice B

Hojas de datos

En esta sección se encuentras recopiladas las hojas de datos de los componentes utilizados, mostrando unicamente la primer hoja, como ayuda visual para su búsqueda.

B.1. Controlador de movimiento LM629

LM628/LM629 Precision Motion Controller

Check for Samples: LM628, LM629

FEATURES

- **32-bit Position, Velocity, And Acceleration Registers**
- **Programmable Digital PID Filter with 16-bit Coefficients**
- **Programmable Derivative Sampling Interval**
- **8- or 12-bit DAC Output Data (LM628)**
- **8-bit Sign-magnitude PWM Output Data (LM629)**
- **Internal Trapezoidal Velocity Profile Generator**
- **Velocity, Target Position, and Filter Parameters may be Changed During Motion**
- **Position and Velocity Modes of Operation**
- **Real-time Programmable Host Interrupts**
- **8-bit Parallel Asynchronous Host Interface**
- **Quadrature Incremental Encoder Interface with Index Pulse Input**
- **Available in a 28-pin Dual In-line Package or a SOIC-24 Package (LM629 Only)**

DESCRIPTION

The LM628/LM629 are dedicated motion-control processors designed for use with a variety of DC and brushless DC servo motors, and other servomechanisms which provide a quadrature incremental position feedback signal. The parts perform the intensive, real-time computational tasks required for high performance digital motion control. The host control software interface is facilitated by a high-level command set. The LM628 has an 8-bit output which can drive either an 8-bit or a 12-bit DAC. The components required to build a servo system are reduced to the DC motor/actuator, an incremental encoder, a DAC, a power amplifier, and the LM628. An LM629-based system is similar, except that it provides an 8-bit PWM output for directly driving H-switches. The parts are fabricated in NMOS and packaged in a 28-pin dual in-line package or a SOIC-24 package (LM629 only). Both 6 MHz and 8 MHz maximum frequency versions are available with the suffixes -6 and -8, respectively, used to designate the versions. They incorporate an SDA core processor and cells designed by SDA.

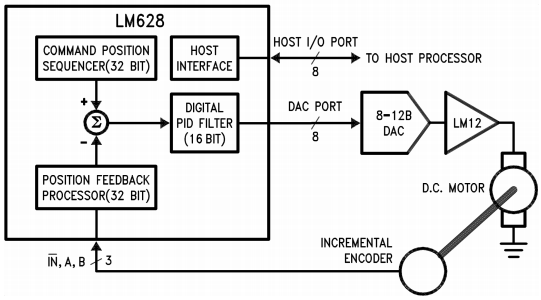


Figure 1. Block Diagram

B.1.1. Guía de programación LM629

User's Guide
SNVA025D–January 1999–Revised May 2013

AN-693 LM628 Programming Guide

This application note discusses programming of the LM628/LM629 motion control processors.

Contents

List of Figures

B.1.2. Reporte de Aplicación

Application Report
SNOA184D–October 1993–Revised May 2013

AN-706 LM628/629

ABSTRACT

This application report is intended to explain and complement the information in the data sheet and also address the common user questions. While no initial familiarity with the LM628/629 is assumed, it will be useful to have the LM628/629 data sheet close by to consult for detailed descriptions of the user command set, timing diagrams, bit assignments, pin assignments, and so on.

Contents

B.2. Puente H LMD18201T

LMD18201 3A, 55V H-Bridge

Check for Samples: LMD18201

FEATURES

- **Delivers up to 3A Continuous Output**
- **Operates at Supply Voltages up to 55V**
- **Low $R_{DS(ON)}$ Typically 0.33Ω per Switch at 3A**
- **TTL and CMOS Compatible Inputs**
- **No "Shoot-Through" Current**
- **Thermal Warning Flag Output at 145°C**
- **Thermal Shutdown (Outputs Off) at 170°C**
- **Internal Clamp Diodes**
- **Shorted Load Protection**
- **Internal Charge Pump with External Bootstrap Capability**

APPLICATIONS

- **DC and Stepper Motor Drives**
- **Position and Velocity Servomechanisms**
- **Factory Automation Robots**
- **Numerically Controlled Machinery**
- **Computer Printers and Plotters**

DESCRIPTION

The LMD18201 is a 3A H-Bridge designed for motion control applications. The device is built using a multi-technology process which combines bipolar and CMOS control circuitry with DMOS power devices on the same monolithic structure. The H-Bridge configuration is ideal for driving DC and stepper motors. The LMD18201 accommodates peak output currents up to 6A. Current sensing can be achieved via a small sense resistor connected in series with the power ground lead. For current sensing without disturbing the path of current to the load, the LMD18200 is recommended.

Functional Diagram

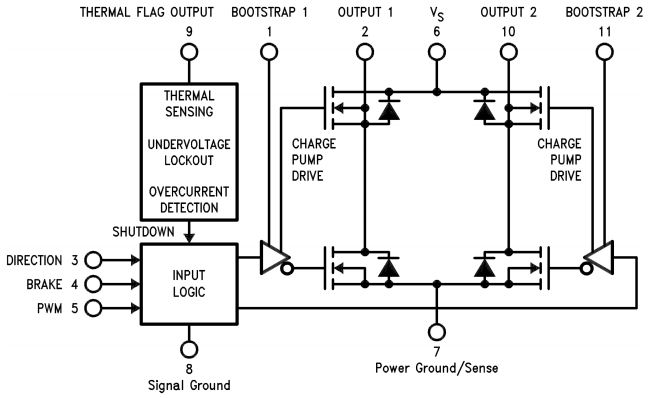


Figure 1. Functional Block Diagram of LMD18201

B.3. Receptor de línea diferencial cuádruple AM26LS33

TEXAS INSTRUMENTS
www.ti.com

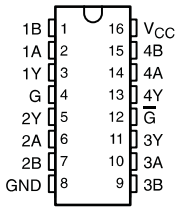
AM26LS32AC, AM26LS32AI, AM26LS33AC
AM26LS32AM, AM26LS33AM
QUADRUPLE DIFFERENTIAL LINE RECEIVERS

SLLS115E–OCTOBER 1980–REVISED OCTOBER 2007

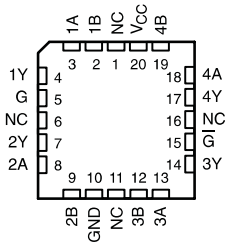
FEATURES

- **AM26LS32A Devices Meet or Exceed the Requirements of ANSI TIA/EIA-422-B, TIA/EIA-423-B, and ITU Recommendations V.10 and V.11**
- **AM26LS32A Devices Have ±7-V Common-Mode Range With ±200-mV Sensitivity**
- **AM26LS33A Devices Have ±15-V Common-Mode Range With ±500-mV Sensitivity**
- **Input Hysteresis . . . 50 mV Typical**
- **Operate From a Single 5-V Supply**
- **Low-Power Schottky Circuitry**
- **3-State Outputs**
- **Complementary Output-Enable Inputs**
- **Input Impedance . . . 12 kΩ Minimum**
- **Designed to Be Interchangeable With Advanced Micro Devices AM26LS32™ and AM26LS33™**

AM26LS32AC . . . D, N, NS, OR PW PACKAGE
AM26LS32AI, AM26LS33AC . . . D, OR N PACKAGE
AM26LS32AM, AM26LS33AM . . . J PACKAGE
(TOP VIEW)



AM26LS32AM, AM26LS33AM . . . FK PACKAGE
(TOP VIEW)



NC – No internal connection

DESCRIPTION

The AM26LS32A and AM26LS33A devices are quadruple differential line receivers for balanced and unbalanced digital data transmission. The enable function is common to all four receivers and offers a choice of active-high or active-low input. The 3-state outputs permit connection directly to a bus-organized system. Fail-safe design ensures that, if the inputs are open, the outputs always are high.

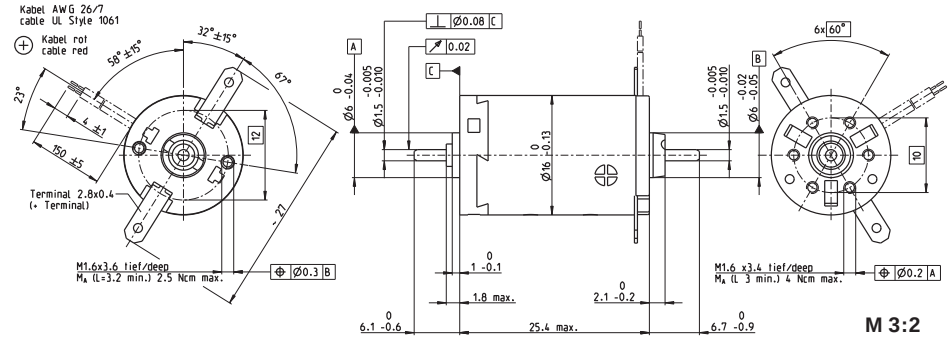
Compared to the AM26LS32 and the AM26LS33, the AM26LS32A and AM26LS33A incorporate an additional stage of amplification to improve sensitivity. The input impedance has been increased, resulting in less loading of the bus line. The additional stage has increased propagation delay; however, this does not affect interchangeability in most applications.

The AM26LS32AC and AM26LS33AC are characterized for operation from 0°C to 70°C. The AM26LS32AI is characterized for operation from –40°C to 85°C. The AM26LS32AM and AM26LS33AM are characterized for operation over the full military temperature range of –55°C to 125°C.

B.4. Motor Maxon A-max 16

maxon A-max

A-max 16 ∅16 mm, Precious Metal Brushes CLL, 1.2 Watt



Stock program
Standard program
Special program (on request)

	Part Numbers									
with terminals	110051	110052	**110053**	110054	**110055**	110056	110057	110058	110059	110060
with cables	139823	352825	352826	352827	352828	352829	352830	352831	352832	352833
Motor Data										
Values at nominal voltage										
1 Nominal voltage	V	1.2	2.4	6	7.2	9	12	15	18	18	30
2 No load speed	rpm	8560	9730	10000	9740	9120	10400	11600	11600	10300	11300
3 No load current	mA	73.9	44.1	18.3	14.7	10.8	9.69	8.99	7.49	6.34	4.33
4 Nominal speed	rpm	7170	6310	4540	4200	3530	4900	6090	6050	4580	5500
5 Nominal torque (max. continuous torque)	mNm	0.694	1.29	2.18	2.17	2.16	2.16	2.13	2.12	2.09	2.04
6 Nominal current (max. continuous current)	A	0.6	0.6	0.407	0.327	0.244	0.21	0.185	0.153	0.134	0.0862
7 Stall torque	mNm	3.83	3.61	4.03	3.86	3.57	4.13	4.54	4.48	3.84	4.04
8 Stall current	A	2.93	1.58	0.723	0.561	0.39	0.386	0.378	0.311	0.236	0.164
9 Max. efficiency	%	71	70	71	71	70	71	72	72	71	71
Characteristics										
10 Terminal resistance	Ω	0.41	1.52	8.3	12.8	23.1	31.1	39.7	57.9	76.2	183
11 Terminal inductance	mH	0.017	0.0519	0.306	0.467	0.831	1.13	1.42	2.05	2.61	6.01
12 Torque constant	mNm/A	1.31	2.29	5.57	6.88	9.17	10.7	12	14.4	16.3	24.7
13 Speed constant	rpm/V	7290	4170	1720	1390	1040	893	795	663	587	387
14 Speed / torque gradient	rpm/mNm	2280	2770	2560	2590	2620	2600	2630	2670	2750	2880
15 Mechanical time constant	ms	25.3	23.7	23.2	23.3	23.3	23.3	23.4	23.3	23.4	23.8
16 Rotor inertia	gcm²	1.06	0.818	0.866	0.857	0.847	0.857	0.85	0.836	0.814	0.791

Specifications

Thermal data

17	Thermal resistance housing-ambient	29.8 K/W
18	Thermal resistance winding-housing	5.5 K/W
19	Thermal time constant winding	3.55 s
20	Thermal time constant motor	165 s
21	Ambient temperature	-30…+65°C
22	Max. winding temperature	+85°C

Mechanical data (sleeve bearings)

23	Max. speed	11000 rpm
24	Axial play	0.05 - 0.15 mm
25	Radial play	0.012 mm
26	Max. axial load (dynamic)	0.8 N
27	Max. force for press fits (static)	35 N
	(static, shaft supported)	280 N
28	Max. radial load, 5 mm from flange	1.4 N

Mechanical data (ball bearings)

23	Max. speed	11000 rpm
24	Axial play	0.05 - 0.15 mm
25	Radial play	0.025 mm
26	Max. axial load (dynamic)	2.2 N
27	Max. force for press fits (static)	30 N
	(static, shaft supported)	280 N
28	Max. radial load, 5 mm from flange	**7.8 N**

Other specifications

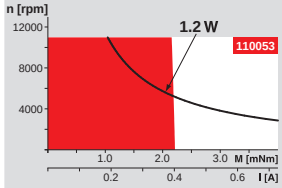
29	Number of pole pairs	1
30	Number of commutator segments	7
31	Weight of motor	22 g

CLL = Capacitor Long Life

Values listed in the table are nominal.
Explanation of the figures on page 107.

Option
Ball bearings in place of sleeve bearings
Without CLL

Operating Range



Comments

Continuous operation
In observation of above listed thermal resistance (lines 17 and 18) the maximum permissible winding temperature will be reached during continuous operation at 25°C ambient.
= Thermal limit.

Short term operation
The motor may be briefly overloaded (recurring).

Assigned power rating

maxon Modular System

Overview on page 20–25

Spur Gearhead
∅16 mm
0.01 - 0.1 Nm
Page 281–284

Planetary Gearhead
∅16 mm
0.1 - 0.6 Nm
Page 285/286

Spindle Drive
∅16 mm
Page 329–331

Recommended Electronics:

Notes	Page 22
ESCON Module 24/2	378
ESCON 36/2 DC	378
EPOS2 24/2	386
EPOS2 Module 36/2	386
EPOS3 70/10 EtherCAT	393
MAXPOS 50/5	396

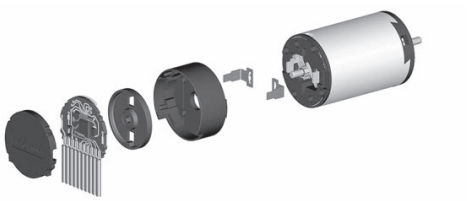
Encoder MR
32 CPT,
2 / 3 channels
Page 351

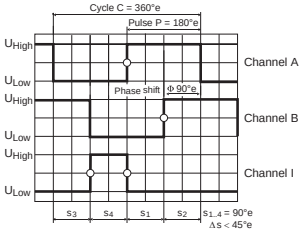
Encoder MR
128 / 256 / 512 CPT,
2 / 3 channels
Page 353

Encoder MEnc
∅13 mm
16 CPT, 2 channels
Page 371

B.5. Encoder MR Tipo M

Encoder MR Type M, 128–512 CPT, 2/3 Channels, with Line Driver





Direction of rotation cw (definition cw p. 106)

Stock program
Standard program
Special program (on request)

Part Numbers	228179	228177	228181	228182	201937	201940
Type						
Counts per turn	128	128	256	256	512	512
Number of channels	2	3	2	3	2	3
Max. operating frequency (kHz)	80	80	160	160	320	320
Max. speed (rpm)	37500	37500	37500	37500	37500	37500

maxon Modular System

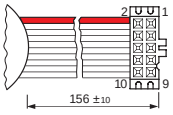
+ Motor	Page	+ Gearhead	Page	+ Brake	Page	Overall length [mm] / ● see Gearhead					
RE 16, 2 W	130					28.0	28.0	28.0	28.0	28.0	28.0
RE 16, 2 W	130	GP 16, 0.1 - 0.6 Nm	285/286			●	●	●	●	●	●
RE 16, 2 W	130	GP 16 S	329/330			●	●	●	●	●	●
RE 16, 3.2 W	132					45.4	45.4	45.4	45.4	45.4	45.4
RE 16, 3.2 W	132	GP 16, 0.1 - 0.6 Nm	285/286			●	●	●	●	●	●
RE 16, 3.2 W	132	GP 16 S	329/330			●	●	●	●	●	●
RE 16, 4.5 W	134					48.4	48.4	48.4	48.4	48.4	48.4
RE 16, 4.5 W	134	GP 16, 0.1 - 0.6 Nm	285/286			●	●	●	●	●	●
RE 16, 4.5 W	134	GP 16 S	329/330			●	●	●	●	●	●
A-max 16	150/152					30.4	30.4	30.4	30.4	30.4	30.4
A-max 16	150/152	GS 16, 0.01 - 0.1 Nm	281-284			●	●	●	●	●	●
A-max 16	150/152	GP 16, 0.1 - 0.6 Nm	285/286			●	●	●	●	●	●
A-max 16	150/152	GP 16 S	329/330			●	●	●	●	●	●
A-max 19, 1.5 W	154					34.0	34.0	34.0	34.0	34.0	34.0
A-max 19, 1.5 W	154	GP 19, 0.1 - 0.3 Nm	288			●	●	●	●	●	●
A-max 19, 1.5 W	154	GP 22, 0.5 - 2.0 Nm	293/295			●	●	●	●	●	●
A-max 19, 1.5 W	154	GS 24, 0.1 Nm	300			●	●	●	●	●	●
A-max 19, 1.5 W	154	GP 22 S	332/333			●	●	●	●	●	●
A-max 19, 2.5 W	156					35.8	35.8	35.8	35.8	35.8	35.8
A-max 19, 2.5 W	156	GP 19, 0.1 - 0.3 Nm	288			●	●	●	●	●	●
A-max 19, 2.5 W	156	GS 20 0.06 - 0.25 Nm	290			●	●	●	●	●	●
A-max 19, 2.5 W	156	GP 22, 0.5 - 2.0 Nm	293/295			●	●	●	●	●	●
A-max 19, 2.5 W	156	GS 24, 0.1 Nm	300			●	●	●	●	●	●
A-max 19, 2.5 W	156	GP 22 S	332/333			●	●	●	●	●	●
A-max 22	158/160					36.9	36.9	36.9	36.9	36.9	36.9
A-max 22	158/160	GP 22, 0.1 - 0.6 Nm	291/292			●	●	●	●	●	●
A-max 22	158/160	GP 22, 0.5 - 2.0 Nm	293/295			●	●	●	●	●	●
A-max 22	158/160	GS 24, 0.1 Nm	300			●	●	●	●	●	●
A-max 22	158/160	GP 22 S	332/333			●	●	●	●	●	●

Technical Data

Supply voltage V_{CC}	5 V ± 5%
Output signal	TTL compatible
Phase shift Φ	90°e ± 45°e
Index pulse width	90°e ± 45°e
Operating temperature range	-25…+85 °C
Moment of inertia of code wheel	≤ 0.09 gcm²
Output current per channel	max. 5 mA

The index signal I is synchronized with channel A or B.

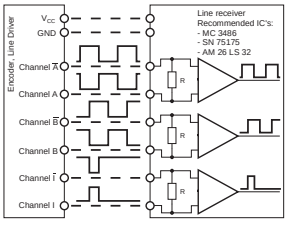
Pin Allocation



1	Motor +
2	V_{CC}
3	GND
4	Motor –
5	Channel $\overline{A}$
6	Channel A
7	Channel $\overline{B}$
8	Channel B
9*	Channel $\overline{I}$ (Index)
10*	Channel I (Index)

DIN Connector 41651/
EN 60603-13
flat band cable AWG 28
*version with 3 channels

Connection example



Opt. terminal resistance R > 1 kΩ

Bibliografía

[1] B. C. Kuo, *Sistemas de control automático*. Pearson Educación, 1996.

[2] K. Ogata, *Ingeniería de control moderna*. Pearson Educación, 2003.

[3] K. J. Astrom, *PID controllers: theory, design and tuning*. Research Triangle Park, 1995.

[4] T. Sauer, *Análisis Numérico*, segunda edición ed. Pearson, 2013.

[5] M. M. Mano, *Lógica Digital y diseño de computadores*. Pearson Educación, 1982.

[6] J. M. C. Leonel Germán Corona Ramírez, Griselda Stephany Abarca Jiménez, *Sensores y actuadores: Aplicaciones para Arduino*, J. E. Callejas, Ed. Grupo Editorial Patria, 2014.

[7] *AN-693 LM628 Programming Guide*, Texas Instruments, Dallas, Texas, March 2013.

[8] *LM628/LM629 Precision Motion Controller*, Texas Instruments, Dallas, Texas, March 2013.

[9] *AN-706 LM628/629 Aplication Report*, Texas Instruments, Dallas, Texas, March 2013.

[10] *Arduino MEGA ADK*, Arduino LLC Std., 2016. [Online]. Available: https://www.arduino.cc/en/Main/ArduinoBoardMegaADK

[11] *Motor Maxon A-max 16*, Maxon Motor Std., 2016. [Online]. Available: http://www.maxonmotor.com/maxon/view/content/products

[12] *Encoder MR Type M, 128Ũ512 CPT, 2/3 Channels, with Line Driver*, Maxon Motor, Sachseln, Switzerland, April 2015.

[13] *LMD18201 3A, 55V H-Bridge*, Texas Instruments, LMD18201, April 2013.

[14] *Maxon A-max 16 Diameter 16 mm, Precious Metal Brushes CLL, 1.2 Watt*, Maxon Motor, Sachseln, Switzerland, May 2015.

[15] J. J. Distefano, A. R. Stubberud, I. J. Williams, and R. G. Cruz, *Retroalimentación y sistemas de control.* McGraw-Hill, 1995, vol. 2.

[16] S. J. Chapman and E. Rozo Castillo, *Máquinas eléctricas.*, 2000.

[17] R. L. Burden, J. D. Faires, and O. Palmas, *Análisis numérico.* Thomson Learning México, 2002, no. QA297. B87 1985.

[18] D. Carter, "Perturbation techniques in mathematics, physics and engineering (richard bellman)," 1965.

[19] J. P. Centeno, "Compensacion de zona muerta y variacion de carga en el control de velocidad de motores de corriente directa," Master's thesis, Instituto Politécnico Nacional, 2008.

[20] R. C. Dorf, R. H. Bishop, S. D. Canto, R. D. Canto, and S. Dormido, *Sistemas de control moderno.* Pearson Prentice Hall, 2005.

[21] R. A. Española, *DLE: Sistema*, Real Academia Española Std. [Online]. Available: http://dle.rae.es/?id=Y2AFX5s

[22] W. Mark and H. Seth, *Vidyasagar. Robot Modeling and Control [M].* JOHN WILEY & SONS, INC, 2005.

[23] K. Ogata, *Sistemas de control en tiempo discreto.* Pearson educación, 1996.

[24] J. A. G. Robles, *Análisis Numérico*, P. E. R. Vázquez, Ed. Mc Graw Hill, 2010.

[25] C. A. Smith, A. B. Corripio, and S. D. M. Basurto, *Control automático de procesos: teoría y práctica.* Limusa, 1991.