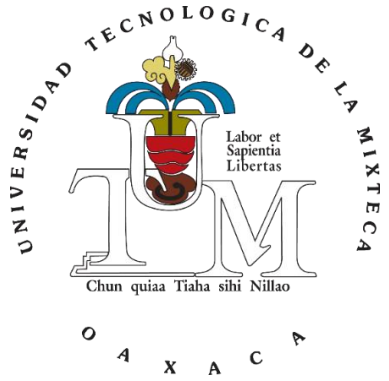


UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

EMULADOR DE UN ROBOT MANIPULADOR DE 2 GRADOS DE LIBERTAD CON BASE EN UN CONTROLADOR DIGITAL DE SEÑALES

TESIS

PARA OBTENER EL GRADO DE:

MAESTRO EN ROBÓTICA

PRESENTA:

ING. DAVID BEDOLLA MARTÍNEZ

DIRECTOR DE TESIS:

DR. FERMÍN HUGO RAMÍREZ LEYVA

HUAJUAPAN DE LEÓN, OAXACA, MÉXICO, FEBRERO DEL 2017

Tesis presentada el 2 de febrero de 2017 ante los siguientes sinodales:

Dr. José Aníbal Arias Aguilar
Dra. Esther Lugo González
Dr. Marco Antonio Contreras Ordaz
Dr. Jesús Linares Flores

Bajo la dirección de:

Dr. Fermín Hugo Ramírez Leyva

Dedicatoria

Con mucho cariño y todo mi amor, dedico este trabajo a las personas que me han brindado su apoyo incondicional.

A MI ESPOSA:

Elizabeth Cortes Suarez

A MIS PADRES:

Paula Martínez López e Ignacio Bedolla Arango

A MIS HERMANOS:

Camilo, Lucia, Vicente, Beatriz, Victorio y Abelina Bedolla

Martínez

La familia que tanto amo

David.

Agradecimientos

Quiero agradecer a todas las personas que de alguna forma contribuyeron a mi formación personal y profesional.

A mi director de tesis y gran amigo Fermín Hugo Ramírez Leyva, mil gracias por su apoyo, motivación, entusiasmo, confianza, consejos, tiempo, paciencia y por su valiosa amistad.

A mi madre por el tierno amor, comprensión y aliento que recibí cada día, por todo el esfuerzo y dedicación. Gracias por darme la vida.

A mi padre que ha sabido guiarme por buen camino.

A mi esposa por su apoyo constante e incondicional, Te amo.

A Camilo, Lucia, Vicente, Beatriz, Victorio y Abelina porque juntos crecimos, reímos y lloramos. Y compartimos momentos de inmensa felicidad. Los amo hermanos.

Al presbítero Ángel Esteban Hernández Sarabia por su apoyo moral durante toda mi vida.

A mi abuela Petra López por su apoyo, amor y cariño.

A mis abuelos Victorio Bedolla y Carmen Arango ejemplos de sacrificio y trabajo constante.

A mi profesor Santos Nieblas por su amistad y sus consejos.

A Deira Sosa por su incansable amistad y compañerismo. Eres una grandiosa y valiosa persona.

A mis sinodales por su tiempo en revisar este trabajo, sus observaciones fueron de gran ayuda.

A la Universidad Tecnológica de la Mixteca, por permitirme desarrollar el trabajo de tesis dentro de sus instalaciones.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por el apoyo económico brindado.

A todos ustedes: mil gracias.

Los quiere y los aprecia.

David

I. Índice

II. Índice de Figuras

III. Índice de Tablas

Resumen

La simulación HIL se usa para acelerar el proceso de diseño en los sistemas de control, a esto se le llama prototipado rápido de control. En el mercado existen plataformas de simulación HIL, pero los precios por unidad son elevados. Como alternativa, se implementó un sistema HIL para un robot manipulador de 2 grados de libertad, con base en un controlador digital de señales de la firma Texas Instruments, para la solución, en tiempo real, del modelo dinámico. En una computadora personal se ejecuta una interfaz gráfica, donde se muestran los movimientos del robot en un ambiente virtual. Se presenta un manual de usuario para utilizar el simulador. Se implementaron una serie de controladores de posición para validar el funcionamiento del mismo. Con este sistema se pueden realizar prácticas de control de robots sin necesidad de la plataforma real que es costosa y en ocasiones no es posible contar con ella.

Introducción

Debido a que algunos sistemas consumen un tiempo y costo significativo, para su diseño e implementación durante su desarrollo y construcción, en ocasiones se opta por realizar una simulación en tiempo real, en condiciones reales, integrando componentes, que emulen, de forma precisa, el comportamiento del sistema.

La técnica Hardware-in-the-loop (HIL) se usa en el desarrollo y prueba de sistemas embebidos complejos. En una simulación HIL, el sistema representado consiste de la parte simulada y de la parte real, es decir el hardware real interactúa con la simulación [1]. HIL Es una herramienta poderosa para reducir costos en el proceso de desarrollo, diseño y manufactura de sistemas ingenieriles.

Se usa para reducir el tiempo de diseño y validación de sistemas complejos. Este enfoque hace uso de hardware reprogramable que es capaz de interactuar con otros dispositivos y con señales del mundo real. Tiene gran flexibilidad porque permite usar componentes reales en un modelo matemático [2]. La diferencia de la simulación HIL, con la simulación numérica es el manejo de conexiones eléctricas y la solución en tiempo real del modelo. La simulación HIL obtiene respuestas mejor aproximadas a las que se obtienen en la aplicación real, debido a que las condiciones son parecidas porque existe ruido.

En este trabajo se implementó un sistema de simulación HIL para emular el comportamiento de un robot manipulador de 2 grados de libertad, dicha emulación sirve como paso intermedio entre la simulación pura y la implementación con la planta real. Este emulador permite probar nuevos algoritmos de control de forma rápida, al contar con una planta virtual que es flexible en cuanto a sus parámetros físicos. Por otra parte el paso a la implementación real no requiere de cambios en las conexiones eléctricas.

Objetivos de la tesis

Objetivo General

Diseñar y construir un simulador HIL con base en un DSC que emule la dinámica de un robot manipulador de 2 GDL y muestre el movimiento del mecanismo usando un ambiente virtual para tener una mejor representación.

Objetivos Específicos

1. Implementar el simulador HIL usando un controlador digital de señales DELFINO de la familia C2000 de Texas Instruments.
2. Obtener una respuesta en tiempo real del simulador HIL.
3. Presentar una Interfaz Gráfica donde se muestre el modelo tridimensional del robot manipulador de 2 GDL usando el software computacional LabVIEW.
4. Probar la viabilidad de un enfoque como HIL en un DSC para un robot de 2 GDL, al resolver en tiempo real, las ecuaciones diferenciales del sistema.
5. Elaborar un manual de usuario para el simulador HIL para un robot de 2 GDL.
6. Probar 2 controladores de posición y comparar los resultados con HIL y el simulador.

Planteamiento del problema

La enseñanza de las áreas de control, mecatrónica y robótica en las universidades se da a través de laboratorios, los cuales están equipados con plantas de control, como son: el carro péndulo, viga bola, levitador magnético y robots manipuladores. Sin embargo son costosas, y requieren mantenimiento continuo, además de que son de arquitectura cerrada o son obsoletas.

El área de control de robots manipuladores tiene una gran importancia, tanto en el área teórica como experimental. Un robot manipulador está compuesto de eslabones rígidos conectados por juntas y son diseñados para funcionar como un brazo humano pero con fuerza y capacidad de carga mejorada [3]. La mayoría de robots comerciales son de arquitectura cerrada, por lo que no es posible probar nuevos algoritmos de control, para ello es necesario contar con un robot de arquitectura abierta, los cuales son costosos y no hay facilidad de obtenerlos.

Una alternativa viable para contar con este tipo de plantas es el desarrollo de sistemas, mediante hardware con velocidad de procesamiento elevada. Donde se resuelve numéricamente la dinámica del robot, se generan las salidas y entradas del sistema de forma analógica, es decir hacer su implementación en HIL.

Con base en la investigación realizada de sistemas hechos con HIL, éstos se implementan con Matlab/Simulink y plataformas de hardware como son, xPC Target, dSPACE, Módulos PXI de National Instruments. Lo que tiene en común estas plataformas es que son muy rápidas para realizar operaciones matemáticas y poseen sistemas de adquisición de datos de alta velocidad.

Existen varias alternativas para realizar un simulador HIL, como son FPGAS, una PC con tarjeta de adquisición de datos de alta velocidad, microcontroladores y controladores digitales de señales (DSC). En este trabajo se implementó un simulador HIL, de un robot de 2 GDL, con base en un DSC de la firma Texas Instruments.

Para la implementación de un simulador HIL de un robot de 2 grados de libertad, en primer lugar se obtiene el modelo dinámico (dos ecuaciones diferenciales no lineales de segundo orden). Estas ecuaciones se cargan en el sistema de cómputo y las resuelve en tiempo real, de tal manera que en función de las entradas, a su salida se tiene las posiciones de cada uno de sus eslabones. Las entradas y salidas se manejan mediante señales de voltaje normalizados.

Para tener un realismo y representación del sistema, se desarrolló una interfaz gráfica, desde la cual se muestra los movimientos del robot en un ambiente virtual, con el fin de mostrar la forma en que se mueve todo el sistema. Finalmente en una computadora personal u otro DSC se van a implementar dos algoritmos de control de posición, y se van a comparar los resultados de simulación contra los que arroja el simulador. Finalmente se va a realizar una página WEB en donde cualquier interesado va a poder descargar la aplicación para validar los resultados.

Justificación

Para desarrollar y probar nuevas técnicas de control es necesario tener una planta en correcto funcionamiento, HIL es una herramienta viable para conseguir este objetivo. HIL proporciona gran flexibilidad debido a que solo es necesario cambiar las ecuaciones de la planta para obtener un comportamiento diferente. Con esto se ataca la problemática en la falta de repertorio de plantas en

los laboratorios de control en las instituciones educativas, facilitando la realización de prácticas en ambientes más parecidos a la realidad

La técnica de HIL tiene una alta fiabilidad de los resultados obtenidos en la simulación, debido a que se agrega hardware a la simulación acelerando el proceso, obteniendo datos que se apegan a los reales debido al ruido presente en la interfaz de entrada/salida. Al usar un simulador HIL se realizan pruebas sin comprometer hardware o personas. Las pruebas son repetibles y algunos escenarios que son difíciles de configurar en el mundo real se logran en un simulador HIL.

Estado del arte

En los sistemas modernos de control, la reducción del tiempo en el diseño de la planta, controladores y ambiente de trabajo, hacen de la técnica HIL una herramienta viable, y se usa cuando el desarrollo de las aplicaciones toma mucho tiempo, las pruebas son costosas o el modelado es complejo.

La literatura reporta una amplia gama de sistemas con base en sistemas HIL. La simulación HIL es un concepto usado especialmente dentro de la industria automotriz, se usa principalmente para prototipado rápido, pruebas y optimización de sistemas [1].

Haciendo una búsqueda de las diferentes áreas en las que se utiliza un sistema HIL se encontraron aplicaciones en: aeronáutica [4] y [5], industria automotriz [6], [7], [8], [9] y [10], misiones espaciales [11], [12] y [13], trenes [14] y [15], producción de energía renovable [16], [17], [2], [18], [19] y [20], mecanismos flexibles [21] y [22], control automático [23], [24], [25] y [26], Otras aplicaciones [27], [28], [29] y [30].

En el área de robótica se tienen diferentes enfoques de aplicación de HIL cada uno con resultados favorables. Los enfoques son Robot-Hardware-in-the-Loop (RHIL) donde se usa un robot real como parte de la simulación [31], Joint-in-the-Loop se usan motores reales para no modelar la dinámica de las juntas [32]. Controller-in-the-Loop donde se utiliza un sistema de control real como parte de la simulación [33].

Los sistemas robóticos son multidisciplinarios y los diseñadores frecuentemente emplean un sistema de síntesis y análisis donde dividen al sistema en subsistemas. El uso de hardware-in-the-loop proporciona al diseñador escenarios que son difíciles de modelar de esta forma el diseño de los robots manipuladores en concurrencia es efectivo refiriéndose a tiempo y a costo [31].

En aplicaciones de HIL en el área de robótica móvil, en [34] se desarrolla un sistema HIL para simular el comportamiento de robots distribuidos, se propone este simulador como paso intermedio para mejorar el proceso de la validación de algoritmos, el sistema consiste de 27 unidades embebidas con comunicación inalámbrica. En [35] se desarrolla un ambiente que permita a los robots y sensores ser integrados en un simulador HIL usando arquitectura de alto nivel. En [36] se usa un algoritmo de sincronización para reducir la diferencia en el comportamiento que presente un robot virtual y un robot real.

Las aplicaciones de HIL que se encontraron sobre manipuladores robóticos se describen a continuación:

En [37] se usa un simulador HIL para un robot paralelo con el fin de evaluar 3 diferentes técnicas de control antes de pasar a la implementación con el sistema real. En [38] se desarrolla un simulador HIL para un robot paralelo de 6 GDL manejado por cable llamado IPAnema 3.

En [32] se usa un simulador joint-in-the-loop para un sistema biomecánico multicuerpo para cadera, un robot de 6 ejes genera las cargas en una endoprótesis de cadera real, con el fin de diseñar los parámetros óptimos para cada paciente en particular.

En [39] se desarrolla un simulador HIL para un robot manipulador manejado eléctricamente el modelo dinámico se programa en una tarjeta FPGA y el controlador se programa usando el ambiente de Simulink Matlab.

En [31] se utiliza un enfoque de diseño concurrente mecatrónico para robots manipuladores usando un simulador HIL para un robot (RHIL), para diseñar de manera eficiente la dinámica y la cinemática de un robot manipulador. El robot ha sido un manipulador de 3 GDL donde cada grado se utiliza un motor real para emular la dinámica de las uniones, el simulador HIL usado, se observa en la Figura 1.

En [40] utiliza un simulador HIL para un robot manipulador de 2 GDL como herramienta educativa especialmente en el área de control, robótica y mecatrónica. Al igual que en [31] utiliza un motor real para emular la dinámica de las juntas y con otro motor se simula el par de carga. Esta plataforma es de arquitectura abierta, y se combina con un servidor que permite acceder a éste a través de Internet.

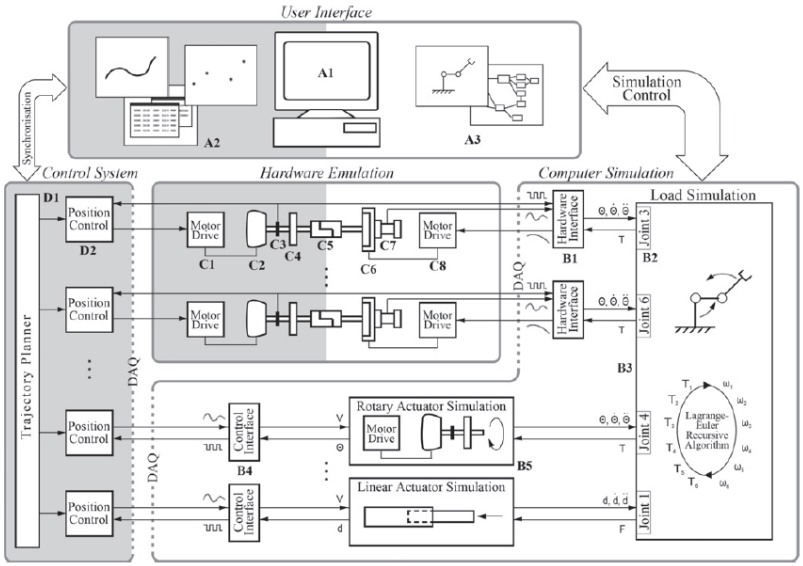


Figura 1 Simulador HIL para un robot

En [33] se desarrolla un simulador HIL para el robot Hyundai como prototipado rápido de sistemas de control reduciendo drásticamente el tiempo de diseño de los controladores. En [41] se usa un simulador HIL para herramientas de maquinado en sistemas de manufactura con el fin de optimizar sistemas mecatrónicos y el software que los controla.

En [42] se usa un control LQR digital para un péndulo doble invertido usando un simulador HIL a través de comunicación serial con un tiempo de muestreo de 0.2 segundos. En [43] a través de Joint-in-the-loop se emula la dinámica de una junta y se diseña un simulador HIL para un robot

industrial de 5 GDL. En [44] se aplica un control de impedancia en un manipulador robótico en un ambiente rígido usando HIL.

En [11], [45], [46], [47] y [48] se usan diferentes simuladores RHIL para emular la dinámica de contacto entre dos satélites para su acoplamiento en el espacio exterior mediante un robot manipulador industrial real. En [49] se usa un simulador HIL para emular el proceso de contacto de dos objetos en vuelo en el espacio exterior identificando los parámetros de rigidez y amortiguamiento.

Con respecto a aplicaciones de HIL con interfaz gráfica, se encontraron los siguientes trabajos. En [50] se evalúan 4 estructuras de control de la familia de controladores saturados para un robot manipulador de 2 GDL en un simulador e interfaz gráfica con base en una computadora personal. En [51] se desarrolla una interfaz gráfica para un robot de 3 GDL donde se resuelve el modelo dinámico. En [52] se desarrolló un simulador para un robot SCARA de 4 grados de libertad el modelo dinámico se obtuvo mediante el software SYMORO. Permite usar 4 tipos de trayectorias predefinidas usando dos tipos de control.

En [53] se valida el modelo dinámico de una plataforma robótica HOAP-3 con un simulador basado en OpenHRP3, se hace una comparativa entre las tareas ejecutadas en simulación y con el robot real. En [54] y [55] se desarrolla un laboratorio virtual remoto de tipo cliente/servidor permite evaluar diferentes estrategias de control con múltiples usuarios, el simulador resuelve el modelo dinámico de una plataforma.

En [56] se desarrolló un simulador de un robot manipulador de 6 GDL con una interfaz gráfica donde se calcula la cinemática inversa mediante la técnica de desacople cinemático. En [57] se diseña e implementa un simulador para entrenamiento en cirugía robótica donde se calcula la cinemática inversa y se muestra el movimiento en una interfaz gráfica. En [58] se desarrolló un simulador para la enseñanza de la robótica paralela se controla un robot tipo LEGO usando la cinemática inversa.

Estructura de la tesis

En el capítulo 1 se muestra el marco teórico de robots manipuladores y sistemas de simulación HIL.

En el capítulo 2 se presenta el diseño del hardware del simulador HIL que incluye los diagramas esquemáticos del sistema empotrado y del controlador con base en una tarjeta de adquisición de datos.

En el capítulo 3 se presenta el diseño del software donde se describen los módulos para la implementación del ERM2GDL y la interfaz gráfica.

En el capítulo 4 se presentan las pruebas y resultados obtenidos con el simulador HIL.

En el capítulo 5 se presentan las conclusiones del trabajo de investigación

Capítulo 1. Marco Teórico

Los robots manipuladores son populares en la industria, debido a la importancia que ocupan en este sector, como herramientas clave para la modernización de las empresas. Hoy en día, la automatización de procesos industriales es realizada a través de robots y esto trae como consecuencia competitividad, productividad, eficiencia y rentabilidad de las empresas [59].

Un robot manipulador es un sistema altamente complejo cuya descripción analítica requiere de ecuaciones diferenciales. La naturaleza no lineal, multivariable y acoplada de su comportamiento dinámico ofrece un amplio espectro en la formulación de problemas de control teóricos y prácticos. La utilidad del modelo dinámico de robots manipuladores es fundamental para propósitos de simulación, diseño y construcción del sistema mecánico, así como análisis y diseño de algoritmos de control [59].

En la Figura 1.1 se observa un manipulador Robótico, el cual consta de eslabones y articulaciones. Los eslabones sirven de soporte a las articulaciones que es en donde se genera el movimiento. La posición final del efector del robot está en función de la posición angular que adquiere cada articulación (coordenadas articulares). La posición final del robot **X** es un vector con tres componentes que se muestran en la ecuación (1.1).

$$\boldsymbol{X}^T = [x_1 \quad x_2 \quad x_3]^T \tag{1.1}$$

En forma general un robot se compone de *n* articulaciones, por lo cual el vector de posición $\boldsymbol{q}$ está dado por la ecuación (1.2), donde cada término corresponde a la coordenada de cada articulación.

$$\boldsymbol{q^T} = [q_1 \quad q_2 \quad \cdots \quad q_n]^T \tag{1.2}$$

Al ponerse en movimiento del robot se tiene *n* velocidades que corresponden a la derivada de la posición y se muestran en la ecuación (1.3), donde $\dot{\boldsymbol{q}}$ es el vector de velocidades.

$$\dot{\boldsymbol{q}}^{\boldsymbol{T}} = [\dot{q}_1 \quad \dot{q}_2 \quad \cdots \quad \dot{q}_n]^T \tag{1.3}$$

Normalmente en cada articulación se tiene un motor que genera el par mecánico para mover la estructura mecánica. Para simplificar al sistema no se toma en cuenta la dinámica de éste y únicamente se usa como entrada el vector de par $\boldsymbol{\tau}$ de cada articulación, que está dado en la ecuación (1.4), donde cada término del vector corresponde al par aplicado en cada articulación.

$$\boldsymbol{\tau^T} = [\tau_1 \quad \tau_2 \quad \cdots \quad \tau_n]^T \tag{1.4}$$

La posición $\boldsymbol{q}$, velocidad $\dot{\boldsymbol{q}}$ y el par $\boldsymbol{\tau}$ en cada articulación forma un vector de dimensión *n*. La posición final del efector es el vector $\boldsymbol{X}$ que es de dimensión 6, donde 3 elementos corresponden a la posición y 3 de orientación.

El modelado del robot se divide en dos partes que son el modelo cinemático y el dinámico. En el primero se observa cómo se mueve el mecanismo sin tomar en cuenta las fuerzas que intervienen para producir dicho movimiento. Con base en éste se determina el espacio de trabajo en la comunidad de robótica es ampliamente aceptado el uso de los parámetros Denavit-Hartenberg. En el modelado

dinámico se obtienen las ecuaciones diferenciales que gobiernan el funcionamiento del sistema y es el enfoque que se usa en este trabajo.

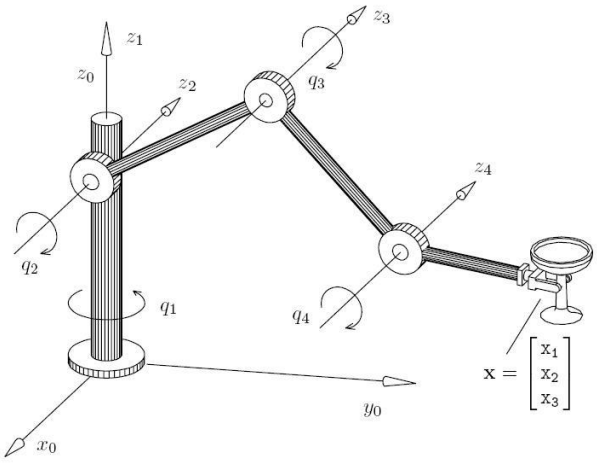


Figura 1.1 Robot manipulador

El modelado dinámico se obtiene a partir de las ecuaciones de Euler-Lagrange y el procedimiento para conseguirlo es el siguiente:

1. Obtener la cinemática directa

$$\boldsymbol{X} = f(q_1, q_2, \dots q_n) \tag{1.5}$$

2. Modelo de energía
- Calculo de la energía cinética *K*
- Calculo de la energía potencial *U*

3. Cálculo del Lagrangiano

$$L = K - U \tag{1.6}$$

4. Desarrollo de las ecuaciones de Euler-Lagrange

$$\frac{d}{dt}\left(\frac{\partial L(\boldsymbol{q}, \dot{\boldsymbol{q}})}{\partial \dot{\boldsymbol{q}}}\right) - \frac{\partial L(\boldsymbol{q}, \dot{\boldsymbol{q}})}{\partial \boldsymbol{q}} = \boldsymbol{\tau} \tag{1.7}$$

Esta metodología es la usada en la literatura especializada de robótica [59] y la ecuación (1.7) representa a todas las ecuaciones diferenciales del sistema. Si el robot consta de 2 articulaciones esto implica que se tienen 2 ecuaciones diferenciales.

Si se utiliza directamente la metodología de Euler-Lagrange, se obtienen ecuaciones que están acopladas, las cuales son complicadas de manipular. Un enfoque alternativo es usar una representación general del sistema como se muestra en la ecuación (1.8) la cual permite desacoplar directamente las ecuaciones diferenciales [59].

$$\tau = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{f_f}(\dot{\boldsymbol{q}}, \boldsymbol{f_e}) \tag{1.8}$$

Donde $\boldsymbol{M}(\boldsymbol{q}) \in R^{nxn}$ es la matriz de inercia, $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \boldsymbol{R}^{nxn}$ es la matriz de Coriolis y fuerza centrípeta, $\boldsymbol{g}(\boldsymbol{q}) \in \boldsymbol{R}^n$ el par gravitacional y $\boldsymbol{f}(\dot{\boldsymbol{q}}, \boldsymbol{f_e}) \in \boldsymbol{R}^n$ la fricción, $\boldsymbol{\tau} \epsilon R^n$ el par aplicado cada una de las articulaciones. Los términos $\ddot{\boldsymbol{q}} \in \boldsymbol{R}^n$, $\dot{\boldsymbol{q}} \in \boldsymbol{R}^n$ y $\boldsymbol{q} \in \boldsymbol{R}^n$ corresponden a la aceleración, velocidad y posición [60].

La matriz de inercia es la propiedad que tiene un objeto de oponerse al cambio de su estado inicial en cuanto a velocidad y posición. La matriz de Coriolis representa las fuerzas generadas por la velocidad de un cuerpo que gira con respecto a un sistema de coordenadas. El par gravitacional corresponde a la influencia de la gravedad en el movimiento del robot. El fenómeno de fricción tiene el efecto físico de oponerse al movimiento del robot, su característica principal es un fenómeno disipativo en velocidades diferentes a cero y con entradas acotadas dentro del primer y tercer cuadrante, lo que permite considerar los modelos tradicionales de fricción viscosa, de Coulomb y estática [59].

1.1 Ejemplo de robots manipuladores

Para dar un ejemplo de los modelos de robots manipuladores se va a partir del péndulo que es un robot de 1 Grado De Libertad (GDL), el cual se muestra en la Figura 1.2. Donde I_1 es la inercia del eslabón, m_1 la masa del eslabón, g la gravedad, l_1 la longitud del eslabón, l_{c1} la distancia entre el eje de giro y el centro de gravedad del eslabón, b el coeficiente de fricción viscosa, f_c el coeficiente de fricción de Coulomb, f_e el coeficiente de fricción estática y τ la entrada de control o Par.

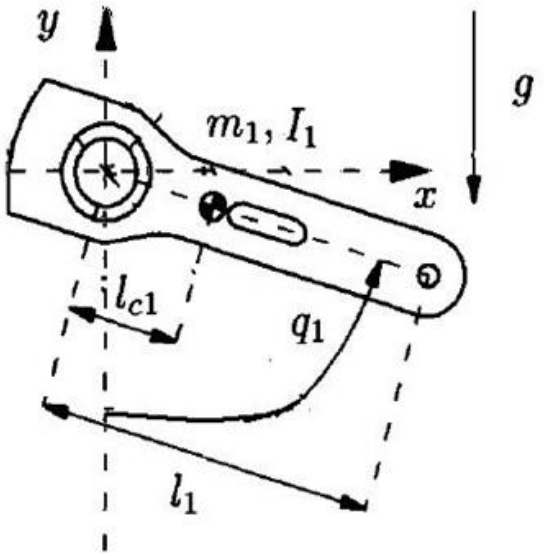


Figura 1.2 Robot manipulador de 1 GDL

Siguiendo la metodología de Euler-Lagrange se obtiene la ecuación diferencial del robot manipulador que se ve en la ecuación (1.9), el procedimiento de cómo se obtiene se explica con detalle en [59].

$$\tau = [m_1 l_{c1}^2 + I_1]\ddot{q}_1 + m_1 g l_{c1} sen(q_1) + b\dot{q}_1 + f_c signo(\dot{q}_1) + f_e[1 - |signo(\dot{q}_1)|] \qquad (1.9)$$

1.2 Robot manipulador de 2 GDL

Un robot manipulador de 2 GDL se muestra en la Figura 1.3, el cual está compuesto de 2 articulaciones, 2 eslabones y dos entradas de control. Los parámetros del sistema son: m_1 la masa del eslabón 1, l_1 la longitud del eslabón 1, I_1 la inercia del eslabón 1, l_{c1} el centro de masa del eslabón 1, q_1 la posición articular del eslabón 1, m_2 la masa del eslabón 2, l_2 la longitud del eslabón 2, I_2 la inercia del eslabón 2, l_{c2} el centro de masa del eslabón 2, q_2 la posición articular del eslabón 2.

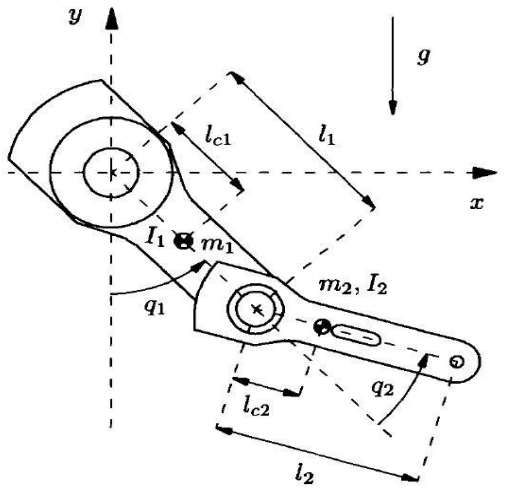


Figura 1.3 Robot manipulador de 2 GDL

El modelo dinámico se obtuvo de [59] y se muestra en la ecuación (1.10). Sus términos son: La matriz de Inercia $\boldsymbol{M}(\boldsymbol{q})$ y matriz de Coriolis $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$ son de 2x2, el vector de par gravitacional $\boldsymbol{g}(\boldsymbol{q})$, fricción $\boldsymbol{f_f}(\dot{\boldsymbol{q}})$ son vectores de 2x1.

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{f_f}(\dot{\boldsymbol{q}}) = \boldsymbol{\tau}$$

$$\mathbf{M}(\boldsymbol{q}) = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}$$

$$\boldsymbol{C}(\boldsymbol{q}) = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \qquad (1.10)$$

$$\boldsymbol{g}(\boldsymbol{q}) = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}$$

$$\boldsymbol{f_f}(\dot{\boldsymbol{q}}) = \begin{bmatrix} f_{f1} \\ f_{f2} \end{bmatrix}$$

Los parámetros del modelo son función de las coordenadas articulares, velocidades y posiciones articulares y los parámetros físicos del sistema están en las ecuaciones de la (1.11) a la (1.22), los cuales son función de los vectores de posición y velocidad articular. Donde la función $sat(\tau, f_e)$ que depende del par de entrada y del coeficiente de fricción estática, representa el signo y magnitud del coeficiente de fricción estática.

$$M_{11} = m_1 l_{c1}^2 + m_2[l_1^2 + l_{c2}^2 + 2l_1 l_{c2} cos(q_2)] + I_1 + I_2 \quad (1.11)$$

$$M_{12} = m_2[l_{c2}^2 + l_1 l_{c2} cos(q_2)] + I_2 \quad (1.12)$$

$$M_{21} = m_2[l_{c2}^2 + l_1 l_{c2} cos(q_2)] + I_2 \quad (1.13)$$

$$M_{22} = m_2 l_{c2}^2 + I_2 \quad (1.14)$$

$$\mathrm{C}_{11} = -m_2 l_1 l_{c2}\,\mathrm{sen}(q_2)\,\dot{q}_2 \quad (1.15)$$

$$\mathrm{C}_{12} = -m_2 l_1 l_{c2} sen(q_2)[\dot{q}_1 + \dot{q}_2] \quad (1.16)$$

$$\mathrm{C}_{22} = 0 \quad (1.17)$$

$$\mathrm{g}_1 = [m_1 l_{c1} + m_2 l_1] g sen(q_1) + m_2 l_{c2} g sen(q_1 + q_2) \quad (1.18)$$

$$\mathrm{g}_2 = m_2 l_{c2} g sen(q_1 + q_2) \quad (1.19)$$

$$f_{f1} = b_1 \dot{q}_1 + f_{c1} signo(\dot{q}_1) + [1 - |signo(\dot{q}_1)|] sat(\tau_1; f_{e1}) \quad (1.20)$$

$$f_{f2} = b_2 \dot{q}_2 + f_{c2} signo(\dot{q}_2) + [1 - |signo(\dot{q}_2)|] sat(\tau_2; f_{e2}) \quad (1.21)$$

$$sat(\tau, f_e) = \begin{cases} f_e & si\ \tau > f_e \\ \tau & si - f_e \leq \tau \leq f_e \\ -f_e & si\ \tau < -f_e \end{cases} \quad (1.22)$$

1.3 Robot manipulador de 3 GDL

Un robot antropomórfico de 3 GDL está formado por 3 articulaciones rotacionales, en la Figura 1.4 (a) y (b) se muestra su configuración. Los parámetros físicos que intervienen en su movimiento son: m_1, m_2 y m_3 las masas de cada eslabón, I_{x1}, I_{x2}, I_{x3}, I_{y1}, I_{y2}, I_{y3}, I_{z1}, I_{z2} y I_{z3} las inercias de cada eslabón, l_1, l_2 y l_3 las longitudes de cada eslabón, l_{c1}, l_{c2} y l_{c3} los centros de masa de cada eslabón, b_1, b_2 y b_3 los coeficientes de fricción viscosa en cada articulación, f_{c1}, f_{c2} y f_{c3} los coeficientes de fricción de Coulomb en cada articulación, f_{e1}, f_{e2} y f_{e3} los coeficientes de fricción estática en cada articulación.

(a) (b)

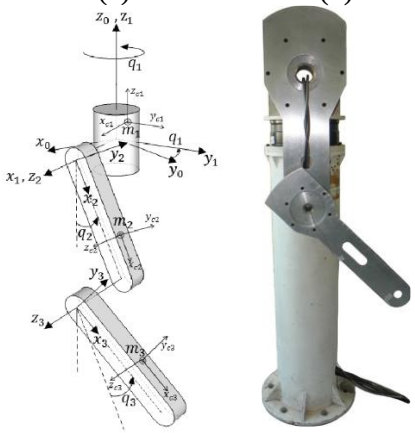


Figura 1.4 Robot manipulador de 3 GDL

En la Ecuación (1.23) se muestra la ecuación diferencial que rige el comportamiento de un robot manipulador de 3 GDL, los cuales son matrices de 3x3 y vectores de 3x1.

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{f}(\boldsymbol{q}) = \tau$$

$$\boldsymbol{M} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

$$\boldsymbol{C} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \qquad (1.23)$$

$$\boldsymbol{g} = \begin{bmatrix} g_1(q) \\ g_2(q) \\ g_3(q) \end{bmatrix}$$

$$\boldsymbol{f} = \begin{bmatrix} f_1(q) \\ f_2(q) \\ f_3(q) \end{bmatrix}$$

Cada uno de los términos del modelo del robot de 3 GDL se muestra en las ecuaciones (1.24) a (1.47), las cuales se obtuvieron de [59].

$$m_{11} = I_{y2}sen^2(q_2) + I_{y3}sen^2(q_2+q_3) + I_{z1} + I_{z2}cos^2(q_2) + I_{z3}cos^2(q_2+q_3) + m_2 l_{c2}^2 cos^2(q_2) + m_3[l_2 cos(q_2) + l_{c3}cos(q_2+q_3)]^2 \quad (1.24)$$

$$m_{12} = 0 \quad (1.25)$$

$$m_{13} = 0 \quad (1.26)$$

$$m_{21} = 0 \quad (1.27)$$

$$m_{22} = I_{x2} + I_{x3} + m_3 l_2^2 + m_2 l_{c3}^2 + m_3 l_{c3}^2 + 2m_3 l_2 l_{c3} cos(q_3) \quad (1.28)$$

$$m_{23} = I_{x3} + m_3 l_{c3}^2 + m_3 l_2 l_{c3} cos(q_3) \quad (1.29)$$

$$m_{31} = 0 \quad (1.30)$$

$$m_{32} = I_{x3} + m_3 l_{c3}^2 + m_3 l_2 l_{c3}^2 cos(q_3) \quad (1.31)$$

$$m_{33} = I_{x3} + m_3 l_{c3}^2 \quad (1.32)$$

$$c_{11} = \left[I_{y2} - I_{z2} - m_2 l_{c2}^2\right]cos(q_2)sen(q_2)\dot{q}_2 + \left[I_{y3} - I_{z3}\right]cos(q_2+q_3)sen(q_2+q_3)\dot{q}_2 - m_3[l_2 cos(q_2) + l_{c3}cos(q_2+q_3)](l_2 sen(q_2)\dot{q}_2 + l_{c3}sen(q_2+q_3)\dot{q}_2) + \left[I_{y3} - I_{z3}\right]cos(q_2+q_3)sen(q_2+q_3)\dot{q}_3 - m_3 l_{c3}sen(q_2+q_3)(l_2 sen(q_2)\dot{q}_3 + l_{c3}sen(q_2+q_3)\dot{q}_3) \quad (1.33)$$

$$c_{12} = \left[I_{y2} - I_{z2} - m_2 l_{c2}^2\right]cos(q_2)sen(q_2)\dot{q}_1 + \left[I_{y3} - I_{z3}\right]cos(q_2+q_3)sen(q_2+q_3)\dot{q}_1 - m_3[l_2 cos(q_2) + l_{c3}cos(q_2+q_3)]l_2 sen(q_2)\dot{q}_1 - m_3[l_2 cos(q_2) + l_{c3}cos(q_2+q_3)]l_{c2}sen(q_2+q_3)\dot{q}_1 \quad (1.34)$$

$$c_{13} = \left[I_{y3} - I_{z3}\right]cos(q_2+q_3)sen(q_2+q_3)\dot{q}_1 - m_3 l_{c2}sen(q_2+q_3)(l_{c1}cos(q_2)\dot{q}_1 + l_{c2}cos(q_2+q_3)\dot{q}_1) \quad (1.35)$$

$$c_{21} = \left[I_{z2} - I_{y2} + m_2 l_{c1}^2\right]cos(q_2)sen(q_2)\dot{q}_1 + \left[I_{z3} - I_{y3}\right]cos(q_2+q_3)sen(q_2+q_3)\dot{q}_1 + m_3[l_1 cos(q_2) + l_{c2}cos(q_2+q_3)](l_1 sen(q_2)\dot{q}_1 + l_{c2}sen(q_2+q_3)\dot{q}_1) \quad (1.36)$$

$$c_{22} = -l_1 m_3 l_{c2} sen(q_3)\dot{q}_3 \quad (1.37)$$

$$c_{23} = -l_1 m_3 l_{c2} sen(q_3)(\dot{q}_2 + \dot{q}_3) \quad (1.38)$$

$$c_{31} = \left[I_{z3} - I_{y3}\right]cos(q_2+q_3)sen(q_2+q_3)\dot{q}_1 + m_3 l_{c2}sen(q_2+q_3)[l_1 cos(q_2) + l_{c2}cos(q_2+q_3)]\dot{q}_1 \quad (1.39)$$

$$c_{32} = l_1 m_3 l_{c2} sen(q_3)\dot{q}_2 \quad (1.40)$$

$$c_{33} = 0 \quad (1.41)$$

$$g_1(q) = 0 \quad (1.42)$$

$$g_2(q) = l_{c2}m_2 sen(q_2) + m_3 l_2 sen(q_2) + m_3 l_{c3} sen(q_2+q_3) \quad (1.43)$$

$$g_3(q) = m_3 l_{c3} sen(q_2+q_3) \quad (1.44)$$

$$f_1(q) = b_1\dot{q}_1 + f_{c1}signo(\dot{q}_1) + f_{e1}[1 - |signo(\dot{q}_1)|] \quad (1.45)$$

$$f_2(q) = b_2\dot{q}_2 + f_{c2}signo(\dot{q}_2) + f_{e2}[1 - |signo(\dot{q}_2)|] \quad (1.46)$$

$$f_3(q) = b_3\dot{q}_3 + f_{c3}signo(\dot{q}_3) + f_{e3}[1 - |signo(\dot{q}_3)|] \quad (1.47)$$

En la Tabla 1.1 se indica el valor de cada uno de los parámetros físicos del Robot de Transmisión Directa (ROTRADI) que tienen en la Benemérita Universidad Autónoma de Puebla (BUAP). Este robot consta de 3 motores de transmisión directa en cada articulación. El espacio de trabajo es una esfera de 90 cm de radio y pesa más de 60 kg. Es de arquitectura abierta por lo que se puede experimentar con diferentes tipos de controladores. Su costo es de más de $200,000 pesos y no es fácil contar con uno en el laboratorio para experimentar en los cursos de robótica.

	Significado	**Notación**	**Valor**
Base	Masa del eslabón 1	m_1	26.9 kg
	Longitud del eslabón 1	l_1	0.45 m
	Inercia del eslabón 1	I_{z1}	1.266 N·m· seg^2/rad
	Inercia del eslabón 1	I_{y1}	0.089 N·m· seg^2/rad
	Inercia del eslabón 1	I_{x1}	0.03 N·m· seg^2/rad
	Centro de masa del eslabón 1	l_{c1}	0.091 m
	Coeficiente de fricción viscosa	b_1	2.288 N·m·seg/rad
	Coeficiente de fricción de Coulomb	f_{c1}	7.17 N·m
	Coeficiente de fricción estática	f_{e1}	8.8 N·m
	Par Máximo	τ_1	50 N·m
Hombro	Masa del eslabón 2	m_2	30 kg
	Longitud del eslabón 2	l_2	0.45 m
	Inercia del eslabón 2	I_{z2}	0.084 N·m· seg^2/rad
	Inercia del eslabón 2	I_{y2}	0.003 N·m· seg^2/rad
	Inercia del eslabón 2	I_{x2}	0.005 N·m· seg^2/rad
	Centro de masa del eslabón 2	l_{c2}	0.038 m
	Coeficiente de fricción viscosa	b_2	0.2 N·m·seg/rad
	Coeficiente de fricción de Coulomb	f_{c2}	1.9 N·m
	Coeficiente de fricción estática	f_{e2}	2.1 N·m
	Par Máximo	τ_2	50 N·m
Brazo	Masa del eslabón 3	m_3	3.88 kg
	Longitud del eslabón 3	l_3	0.45 m
	Inercia del eslabón 3	I_{z3}	0.056 N·m· seg^2/rad
	Inercia del eslabón 3	I_{y3}	0.0012 N·m· seg^2/rad
	Inercia del eslabón 3	I_{x3}	0.009 N·m· seg^2/rad
	Centro de masa del eslabón 3	l_{c3}	0.048 m
	Coeficiente de fricción viscosa	b_3	0.175 N·m.seg/rad
	Coeficiente de fricción de Coulomb	f_{c3}	1.743 N·m
	Coeficiente de fricción estática	f_{e3}	1.87 N·m
	Gravedad	g	9.81 m/seg^2
	Par Máximo	τ_2	13 N·m

Tabla 1.1 Parámetros del robot antropomórfico de 3 GDL

Una vez que se describió el modelo dinámico, en la siguiente sección se explica los métodos numéricos utilizados para la programación del simulador HIL del robot manipulador de 2 grados de libertad.

1.4 Métodos numéricos para la solución de ecuaciones diferenciales ordinarias.

Los métodos numéricos son algoritmos para resolver las ecuaciones diferenciales con una computadora. Existen varios tipos como son Euler y Runge–Kutta [61], que son de paso fijo o variable. Para aplicaciones de tiempo real se utiliza el paso fijo y el método más sencillo es el método de Euler.

Sea la ecuación diferencial de primer grado la cual es continua. Donde y representa la solución de la ecuación diferencial y depende de la variable t, definida en la ecuación (1.48)

$$y' = f(t, y) \tag{1.48}$$

Para un problema de valor inicial, se tiene que t_0 es un valor inicial de la variable t, y_0 es el valor de la función solución evaluada en el tiempo t_0:

$$\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0 \tag{1.49}$$

Las fórmulas que determinan la solución numérica aproximada se muestran en la ecuación (1.50) y (1.51) que representan el método de Euler, donde $h = t_n - t_{n-1}$ es el paso o periodo de muestreo y n es el número del punto aproximado actual, donde t_i es el tiempo donde se obtiene la solución numérica.

$$t_n = t_{n-1} + h \tag{1.50}$$
$$y_n = y_{n-1} + f(t_{n-1}, y_{n-1})h \tag{1.51}$$

El procedimiento para obtener la solución numérica inicia con un punto (t_0, y_0) que representa las condiciones iniciales del sistema. La solución numérica aproximada y_1 se obtiene de la siguiente forma:

$$y(t_1) \cong y_1 = y_0 + f(t_0, y_0)h \tag{1.52}$$

Con esta solución numérica, se repite el método para obtener otra solución numérica aproximada y_2 de la forma:

$$y(t_2) \cong y_2 = y_1 + f(t_1, y_1)h \tag{1.53}$$

Para ejemplificar este procedimiento se resolverá numéricamente el modelo dinámico del robot de un 1 GDL, el cual consta de una ecuación diferencial ordinaria de segundo grado.

$$\tau = [m_1 l_{c1}^2 + I_1]\ddot{q}_1 + m_1 g l_{c1} sen(q_1) + b\dot{q}_1 \tag{1.54}$$

Se despeja $\ddot{q}_1$, y se obtiene lo siguiente.

$$\ddot{q}_1 = f_1(q_1, \dot{q}_1) = \frac{\tau - m_1 g l_{c1} sen(q_1) - b\dot{q}_1}{m_1 l_{c1}^2 + I_1} \tag{1.55}$$

A partir de las condiciones iniciales para $t = 0$, $q_1(t_0)$ y $\dot{q}_1(t_0)$, se obtiene un valor numérico para $\ddot{q}_1$. Este resultado se integra dos veces para obtener valores de q_1 y $\dot{q}_1$ en $t_1 = t_0 + h$, donde h es el incremento de tiempo o periodo de muestreo. Se repite el proceso para $t_2 = t_1 + h$.

El método de integración de Euler es sencillo y se resume en

$$\dot{q}_1(t_i) = \dot{q}_1(t_{i-1}) + \ddot{q}_1(t_{i-1})h \tag{1.56}$$

$$q_1(t_i) = q_1(t_{i-1}) + \dot{q}_1(t_{i-1})h \tag{1.57}$$

De tal forma que programar la solución de ecuaciones diferenciales con la computadora es un proceso que se hace relativamente directo. Para aplicaciones de tiempo real se necesita que el tiempo en que se resuelve la ecuación numérica deba ser menor al tiempo de muestreo. Esto funciona tanto para ecuaciones diferenciales lineales como no lineales. Para resolver un sistema de ecuaciones de orden superior, se necesita reescribirlas en variables de estado, y dividirlas en ecuaciones diferenciales de primer grado y cada una se resuelve con el procedimiento de la ecuación (1.51).

Por otra parte se denomina método de Runge-Kutta de cuarto orden a las ecuaciones (1.58).

Este algoritmo es de uso extendido, y es reconocido como una valiosa herramienta de cálculo, por la buena aproximación que produce. Presenta un error y una estabilidad mayor en comparación con el método de Euler con el mismo tiempo de muestreo.

$$k_1 = hf(t_i, y_i)$$

$$k_2 = hf\left(t_i + \frac{h}{2}, y_i + \frac{1}{2}k_1\right)$$

$$k_3 = hf\left(t_i + \frac{h}{2}, y_i + \frac{1}{2}k_2\right) \tag{1.58}$$

$$k_4 = hf(t_i + h, y_i + k_3)$$

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Donde h es el paso de simulación, y_i es el estado actual (condiciones iniciales del sistema), $f(t, y)$ es la función que representa a la ecuación diferencial a resolver tomando un punto, dado por un tiempo t y un estado y.

En la siguiente sección se hablará sobre los tipos de simulación y las diferentes técnicas para simular sistemas.

1.5 Simulación HIL

La simulación es el método para realizar experimentos sobre modelos de sistemas. Durante la simulación los estados del modelo cambian. Dependiendo del dominio del modelado existen diferentes tipos de simulación [1].

- DESS: (Diferencial Equation System Specification, por sus siglas en inglés) – Tiempo continuo.
- DTSS: (Discrete Time System Specification, por sus siglas en inglés) – Tiempo discreto
- DEVS: (Discrete Event Systems Specification, por sus siglas en inglés) – Se ve como una versión más general de DTSS.

El prototipado rápido es una técnica de desarrollo de sistemas a partir de la variación de parámetros en una simulación, esto permite obtener un sistema óptimo a partir de los requerimientos de diseño. De modo que se evita la construcción de un sistema físico para cada variación de parámetros.

Hardware-in-the-loop es una técnica que se usa en el desarrollo y prueba de sistemas embebidos complejos. En una simulación HIL el sistema representado consiste de la parte simulada y de la parte real, es decir el hardware real interactúa con la simulación [1], en el área de control se refiere a la simulación de plantas mediante hardware.

- *Robot-Hardware-in-the-loop* es donde se usan sistemas robóticos complejos desde manipuladores hasta robots móviles aéreos, acuáticos y terrestres, la simulación interactúa con un sistema robótico.
- *Controller-in-the-loop* es donde el controlador real interactúa con un modelo matemático que se resuelve en un procesador.
- *Joint-in-the-loop* es donde una computadora calcula las cargas en las juntas de robots manipuladores y las emula en actuadores reales.

En este trabajo se usa Hardware-in-the-loop debido a que se tiene la necesidad de emular una planta para aplicaciones de prototipado rápido de control y de enseñanza.

1.6 Controlador digital de señales

El procesamiento digital de señales (DSP, por sus siglas en inglés) es la aplicación de operaciones matemáticas a señales presentadas digitalmente, las señales son representadas digitalmente como secuencias de muestras. Frecuentemente estas muestras se obtienen de señales físicas a través del uso de transductores y convertidores analógico digital (ADC, por sus siglas en inglés). Después del procesamiento matemático las señales digitales son convertidas a señales físicas usando convertidores digital analógico (DAC, por sus siglas en inglés) [62].

Se define un sistema DSP como cualquier sistema electrónico que hace uso de procesamiento digital de señales.

Una característica clave de los sistemas DSP es su periodo de muestreo, es decir la frecuencia con la que las muestras son procesadas o producidas. Combinados con la complejidad de los algoritmos, el periodo de muestreo determina la velocidad requerida de la implementación [62].

Los microcontroladores son microcomputadoras de un solo chip, diseñadas para la automatización y control de máquinas y procesos. Tienen una Unidad Central de Procesamiento

(CPU, por sus siglas en inglés), memoria, puertos de entrada y salida, contadores y relojes, convertidores analógico digital y digital analógica, puerto serie, interrupciones lógicas y muchas funcionalidades [63].

Un controlador digital de señales es una hibridación entre un DSP y un microcontrolador, lo que permite manejar interrupciones, realizar operaciones aritméticas y leer señales tanto digitales como analógicas.

En el mercado, actualmente existen numerosas opciones de tarjetas de desarrollo, entre las cuales destaca Arduino; que es la más popular y sus numerosos clónicos, como una opción alternativa se tiene la familia LaunchPad desarrollada por Texas Instruments (TI) [64].

Para su programación se tienen disponibles diversas herramientas de desarrollo que se adaptan a las necesidades y habilidades del usuario, esto permite elegir la opción que más convenga para desarrollar la aplicación de forma efectiva. Para la familia de productos de TI, Las opciones con la que se cuenta para su programación son las siguientes:

- Energía es un compilador open-source de un proyecto iniciado en el 2012 y está basado en Wirng, lo que lo hace tener características y entorno similares al ambiente de desarrollo integrado (IDE, por sus siglas en inglés) de Arduino.
- *IAR Embeded Workbench* es un IDE que está diseñado para desarrollar y depurar aplicaciones basadas en C/C++ para los micro-controladores de la familia MSP430. Es ideal para usuarios de nivel intermedio/ avanzado.
- *Code Composer Studio* es un IDE completo recomendado por Texas Instruments (TI) y soporta toda la familia de micro-controladores y procesadores de esta compañía [65].

Para la implementación de HIL se usa la tarjeta de desarrollo TI C2000 Delfino F28377S LaunchPad que se basa en el microcontrolador C2000 Delfino TMS320F28377S y que se muestra en la Figura 1.5. El cual es capaz de realizar 400 millones de instrucciones por segundo (MIPS, por sus siglas en inglés) de rendimiento total de sistema entre una CPU C28x de 200 MHz y un acelerador de la ley de control (CLA, por sus siglas en inglés) en tiempo real de 200 MHz. Es una plataforma de evaluación de bajo costo que ofrece a los diseñadores realizar aplicaciones de control digital de alto rendimiento [66].

Cuenta con 4 ADC de 12 bits con una velocidad de 3.5 millones de muestras por segundo, se multiplexa para manejar 16 entradas en modo simple o 8 en modo diferencial, comparadores, filtros sigma-delta, 3 DAC de 12 bits, Entradas de Encoder de cuadratura y 14 salidas modulador de ancho de pulso (PWM, por sus siglas en inglés), de las cuales 9 son de alta resolución, unidad de punto flotante (FPU, por sus siglas en inglés), unidad Trigonométrica (TMU, por sus siglas en inglés), unidad Compleja (VCU, por sus siglas en inglés), protocolos de comunicación entre los cuales esta *Universal Serial Bus* (USB, por sus siglas en inglés).

Trabaja entre -40°C y 125°C, esta tarjeta está calificada para aplicaciones automotrices basado en Q100 para trabajar en ambientes extremos. Estas características permiten obtener señales de voltaje a una velocidad de 1.6 millones de muestras por segundo para 3 entradas de analógicas, ejecutar instrucciones con 5 ciclos de reloj, acelerando el proceso por unidades especializadas en operaciones complejas, trigonométricas y de punto flotante, permite generar 3 salidas analógicas y 15 salidas digitales con PWM’s.

Para la implementación HIL del robot manipulador, se requiere un tiempo de muestreo de 2 milisegundos para converger, por tanto se tienen 2 millones de ciclos de reloj para resolver el modelo matemático, por lo que esta tarjeta es capaz de generar una respuesta en tiempo real.

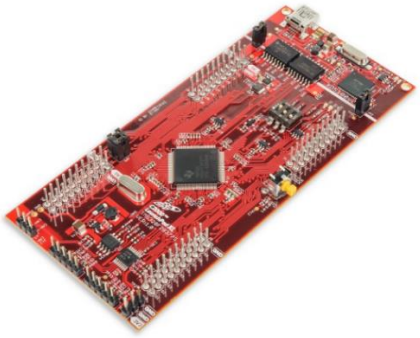


Figura 1.5 C2000 Delfino MCUs F28377S LaunchPad Kit de Desarrollo

1.7 Ambientes virtuales

La evolución de los simuladores se ha dado en función a los avances en las computadoras y en los lenguajes de programación; los simuladores actualmente son una herramienta poderosa, exacta y amigable en la interacción con el usuario. Es importante recalcar que un simulador es una herramienta que requiere del modelo dinámico del sistema y de una estructura de control que permita realizar correctamente la tarea asignada. En cambio, una animación son solamente fotogramas sucesivos que permiten brindar la sensación de movimiento [50].

Evidentemente el sistema que se pretende simular es modelado para tener una representación del mismo y de su evolución. Muchas de las aplicaciones de representación tridimensional y de realidad virtual disponen de opciones de modelado, y en muchas otras se emplean aplicaciones y herramientas específicas para ello [67].

Habitualmente, la enseñanza de una asignatura técnica necesita el uso de materiales de laboratorio caros, por lo que, en muchas ocasiones, éstos son insuficientes para todos los alumnos. Además, se requiere que asistan a los laboratorios a realizar prácticas, debiéndose acomodar a los horarios ofrecidos. Así, surge la necesidad de métodos alternativos, como los laboratorios virtuales o remotos. Los cuales juegan un papel importante en el aprendizaje y asimilación de los conceptos prácticos por parte de los estudiantes. Este tipo de medios permite acceder a recursos, que son limitados debido a su coste, desde cualquier localización sin necesidad de desplazamiento a los laboratorios en determinados horarios [68].

La simulación robótica se ha convertido en una herramienta indispensable para los ingenieros de diseño de productos y de procesos, así como para los investigadores que verifican y validan sus nuevos conceptos; más aún, la simulación robótica se ha convertido en el puente de comunicación entre los equipos de diseño y los consumidores [69].

Capítulo 2. Diseño del Hardware

En este capítulo se muestra el diseño del hardware utilizado en la implementación del simulador HIL para el robot manipulador de 2 GDL, el cual está constituido por el ERM2GDL y el controlador. El diagrama a bloques del simulador HIL se muestra en la Figura 2.1 donde se observan el controlador (con base en la tarjeta DAQ 6062E y una computadora personal), y el ERM2GDL (que consta de la etapa de acondicionamiento de señales, el sistema empotrado y la interfaz gráfica). El controlador se ejecuta en la computadora personal usando Matlab/Simulink, y éste se conecta eléctricamente al ERM2GDL.

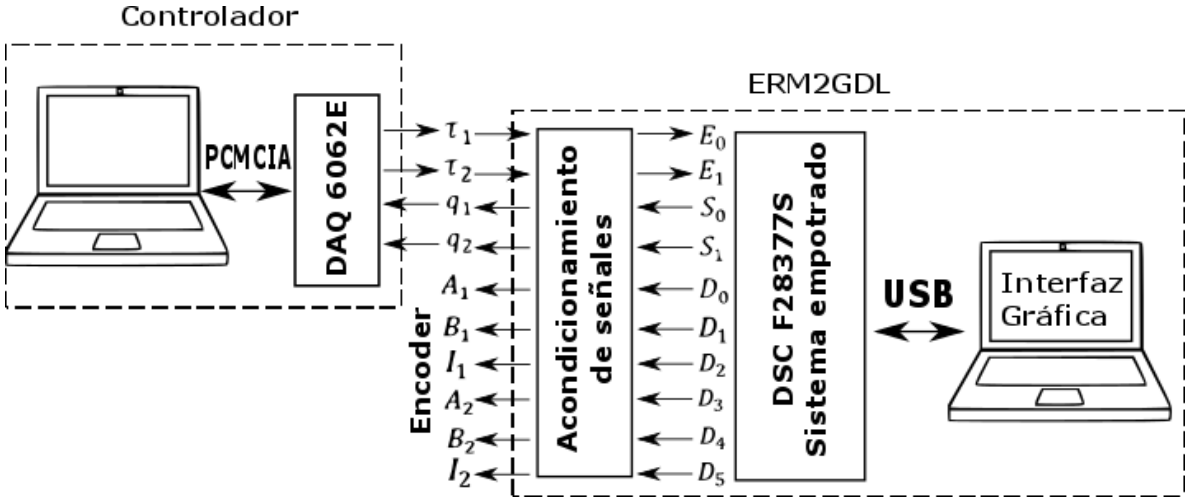


Figura 2.1 Diagrama del simulador HIL

El sistema empotrado ejecuta en tiempo real el modelo del robot en el DSC Delfino F28377S. Para incrementar el realismo se cuenta con una interfaz gráfica, en la que se observan los movimientos del robot y se muestra el historial de la posición, velocidad y el par. La comunicación entre el DSC y la computadora personal es por el puerto USB, usando un protocolo de comunicaciones tipo maestro-esclavo.

El acondicionamiento de señales tiene como entradas el par de cada una de las articulaciones (τ_1 y τ_2), las salidas son las posiciones (q_1 y q_2) y el enconder de cada articulación (A_1, B_1, I_1, A_2, B_2 y I_2). El controlador consta de una tarjeta de adquisición de datos y una PC, tiene como entradas las posiciones (q_1 y q_2) y genera las salidas del par (τ_1 y τ_2). Las entradas del sistema empotrado son los pares (E_o y E_1); sus salidas son las posiciones (S_0 y S_1) y los Encoders (D_0, D_1, D_2, D_3, D_4 y D_5).

Las señales de entrada y salida, entre el controlador y el acondicionamiento de señales, son voltajes en el rango de ±10V. Entre el sistema empotrado y el acondicionamiento de señales el rango es de 0 a 3.3V, el Encoder maneja voltajes digitales de 0 a 3.3V.

En las siguientes secciones se explica con más detalle cada uno de los módulos (la información se obtuvo de [70] y [71]).

2.1 Sistema empotrado

El sistema empotrado se basa en el kit C2000 Delfino MCUs F28377S LaunchPad Development Kit de la firma Texas Instruments (TI), cuya fotografía se muestra en la Figura 1.5. El DSC tiene las siguientes características:

- 2 procesadores a 200 MHz.
- Convertidores ADC y DAC.
- Salidas PWM.
- Unidad de punto flotante, unidad de matemática compleja y unidad trigonométrica.
- Protocolos de comunicación: CAN, SPI, SCI, I2C.
- Pines digitales de propósito general de entrada/salida.

El DSC F28377S cuenta con dos procesadores que trabajan a una frecuencia de 200 MHz, los cuales se denominan CLA y CPU principal. El CLA es un procesador independiente, ha sido diseñado para aplicaciones de control. El CPU principal se encarga del manejo de los recursos del sistema, es decir, salidas y entradas tanto analógicas como digitales, comunicación serial y comunicación con el procesador CLA, tiene un desempeño de 400 MIPS. Cuenta con 4 convertidores analógico digital, cada uno con una frecuencia máxima de muestreo de 3.5 MS/s con una resolución de 16 bits y se multiplexa para conseguir 14 entradas analógicas (Figura 2.2), por lo que si se muestrean todos los canales se obtiene una frecuencia máxima de 250 kS/s por canal (con una resolución de 12 bits). EL DSC cuenta con 4 salidas analógicas, que funcionan como convertidor digital analógico, su frecuencia de corte es de 10 kHz con una resolución de 12 bits. Tienen 14 módulos PWM, de los cuales 9 son de alta resolución. Los pines GPIO (Entradas/salidas de propósito general) se configuran como entrada o salida digital; en total se tienen 168 pines de tipo GPIO.

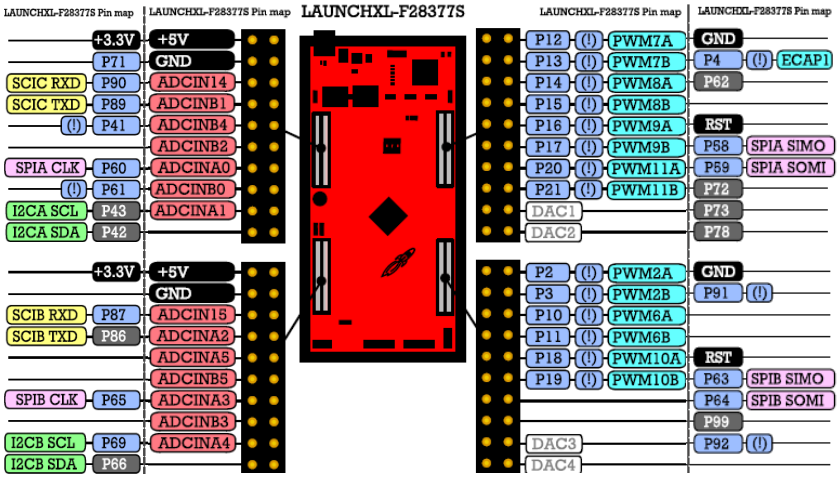


Figura 2.2 Recursos y pines del F28377S

En la Figura 2.3 se muestra el diagrama conceptual de los recursos que utiliza el sistema empotrado, los cuales son: Las salidas de PWM (Modulador de Ancho de Pulso), convertidores ADC y DAC, comunicación serial USB (SCI) y Salidas digitales. Las salidas indican la posición y velocidad angular, se representan por voltaje analógico, salida de Encoder y datos enviados por comunicación serial. En las siguientes subsecciones se explica cada una de ellas.

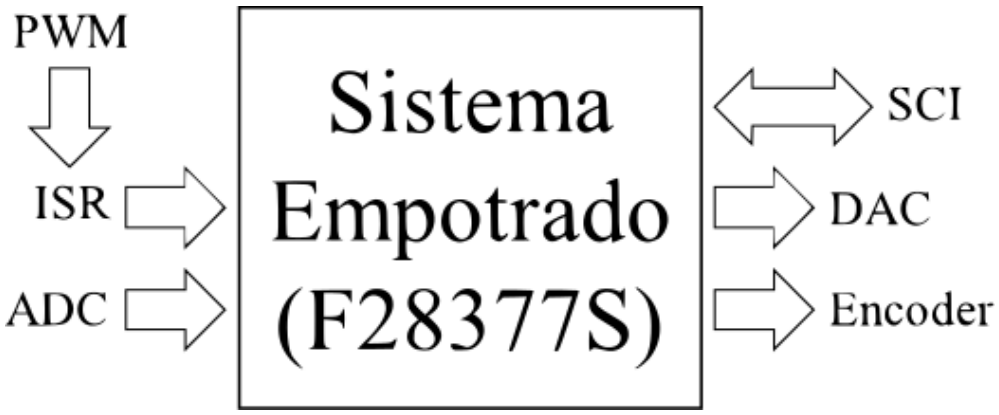


Figura 2.3 Diagrama del ERM2GDL

2.1.1 Unidad Central de Procesos C28X y Acelerador de la ley de control

El CPU C28X es un procesador de punto fijo de 32 bits (Figura 2.4), contiene una ALU de punto fijo con multiplicadores y sumadores de 16 y 32 bits, una ALU para enteros de 16 bits y una unidad de control para direccionar los valores a ser procesados. Cuenta con un bus de registros para leer y escribir simultáneamente resultados generados por unidades especializadas y dedicadas a cierto tipo de operaciones matemáticas, estas son: Unidad de punto flotante de 32 bits de precisión simple (FPU), Unidad Trigonométrica (TMU) y Unidad de matemática compleja (VCU).

Cada una de las unidades especializadas completa cálculos matemáticos en un número de ciclos de reloj menor al que tomaría por CPU. La lectura y escritura simultánea en el bus de registros por el CPU y las unidades especializadas toma sólo 1 ciclo de reloj. La unidad de punto flotante extiende la habilidad del CPU al proporcionarle la capacidad de hacer operaciones de este tipo de 32 bits. Tiene 8 registros para almacenar el resultado, los registros no se leen directamente, sólo el CPU y la FPU acceden a ellos y se pueden leer cuando termina una operación.

La unidad trigonométrica se usa para facilitar el cálculo de operaciones de punto flotante como el seno, coseno, tangente y operaciones relacionadas con el número π. Cuenta con 8 registros para guardar los resultados. Es importante tomar en cuenta que la entrada (en radianes) a la unidad trigonométrica está normalizada entre ± 1, donde el máximo valor es 2π, por lo que cuando se requiera obtener el $\sin(\pi)$, se tiene que normalizar π de la siguiente forma: $\frac{\pi}{2\pi} = 0.5$ por lo que el valor a insertar en la función es 0.5, es decir, $\sin(0.5)$ el resultado equivaldrá a $\sin(\pi)$ en radianes.

La unidad de matemática compleja da soporte al CPU para realizar cálculos como la Transformada Rápida de Fourier, así como protocolos de comunicación basados en algoritmos, los resultados se almacenan en sus registros. La unidad está especializada en decodificación Viterbi, el cual es un método para verificar la integridad de los datos cuando se tiene una gran cantidad de ellos, como son, paquetes de comunicación y secciones de código. También es capaz de realizar cálculos para la implementación de filtros digitales complejos.

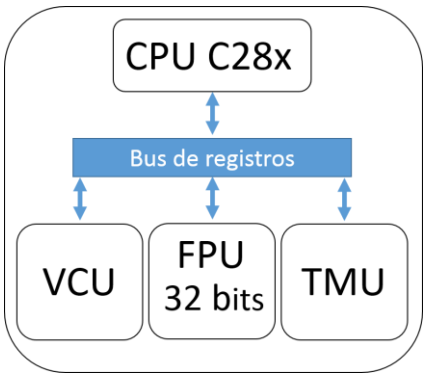


Figura 2.4 Unidad Central de Procesos

El acelerador de la ley de control (CLA) es un procesador de punto flotante de 32 bits, el cual es independiente del CPU C28X, lo que le permite tener una ejecución concurrente al F28377S. Sus principales características son:

- Frecuencia de trabajo de 200 MHz al igual que la del CPU principal.
- Arquitectura independiente con acceso de lectura y escritura al bus de registros del CPU.
- 4 registros de 32 bits para compartir los resultados con el CPU.
- 2 registros de 16 bits para compartir resultados con el CPU
- Precisión simple de punto flotante.
- Manejo de interrupciones producidas por periféricos mediante MPERINT1 a MPERINT8.
- Puede configurar periféricos.
- Capacidad de generar hasta 8 interrupciones, mediante el PIE.

El CLA puede configurar los periféricos, acceder a los resultados de los ADC's, modificar el ciclo de trabajo de las salidas PWM, etc. Genera hasta 8 interrupciones a partir de periféricos o por el CPU (CLA_INT1 a CLA_INT8). Activa una interrupción, que ejecutará el CPU, mediante los grupos de interrupción 11 y 12. Los resultados del CLA se guardan en los registros _MR0 a _MR3 y MAR0 a MAR1, que son leídos por el CPU en el bus de datos. Cabe mencionar que dichos registros no pueden ser accedidos directamente por el usuario.

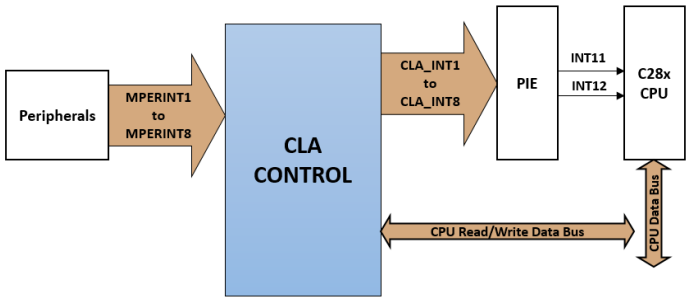


Figura 2.5 Acelerador de la ley de control

2.1.2 Módulo PWM

El F28377S cuenta con 14 módulos PWM, de los cuales 9 son de alta resolución. Cada módulo se configura con 32 registros de 16 bits. Dos canales PWM trabajan de forma independiente o sincronizada. Sus principales características son:

- Un contador de 16 bits.
- Frecuencia de operación de 12.5 MHz a 1.56 MHz.
- Control de fase programable.
- Manejo de tiempos muertos.

En la Figura 2.6 se observa el diagrama a bloques del módulo PWM, el cual consta de 11 submódulos funcionales de los cuales los más importantes son: base de tiempo (BT), contador comparador (CC), calificador de acción (CA), generador de tiempos muertos (TM), interruptor de PWM (IP), zona de fallas (ZF), activador de eventos (AE) y comparador digital (CD).

El submódulo de base de tiempo, determina la frecuencia de operación del módulo PWM. Se usa para especificar la frecuencia y modo de operación del contador. Gestiona la sincronización con otros módulos PWM y establece el periodo de la señal de salida PWM.

El submódulo contador comparador determina el ciclo de trabajo del PWM. El calificador de acción determina el valor digital en la salida PWM, dependiendo del ciclo de trabajo y el contador. El generador de tiempos muertos crea retrasos de tiempo en los cambios de estado, lo cual es útil en aplicaciones de control de motores con transistores.

El submódulo interruptor de PWM permite que una señal portadora de alta frecuencia module la forma de onda de la salida PWM. El de zona de fallas determina el valor digital en la salida PWM dependiendo del tipo de falla presente. El activador de eventos se encarga de generar interrupciones al CPU o iniciar una conversión con el ADC. El submódulo comparador digital genera eventos para los submódulos AE, ZF y BT a partir de señales externas.

El reloj TBCLK, se obtiene del prescalador que varía la frecuencia en el rango de $[1.56, 12.5]\ MHz$. La frecuencia de operación resultante será usada como reloj por el contador de 16 bits. El comparador usa el registro del periodo y el contador para asignar los valores digitales a la salida con el calificador de acción.

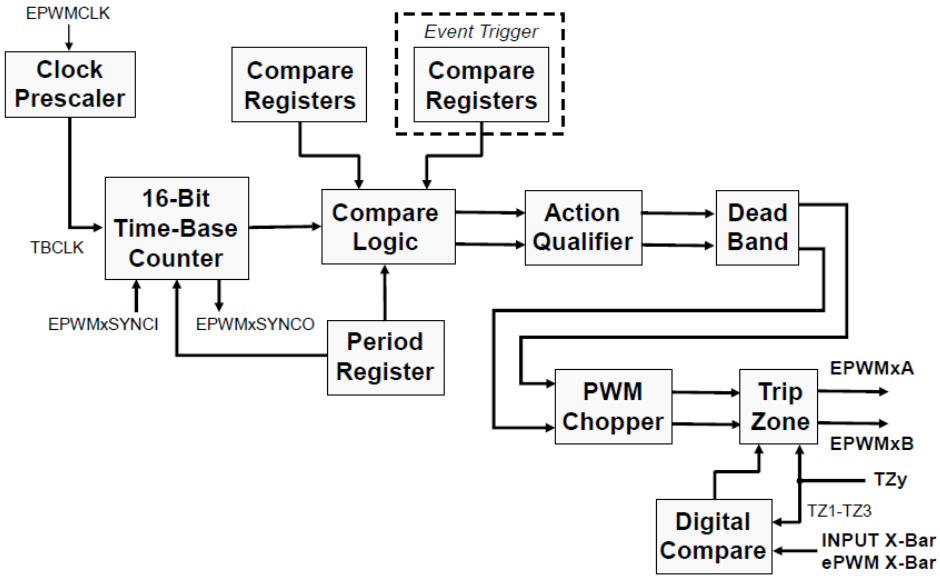


Figura 2.6 Diagrama de bloques del canal de PWM

2.1.2.1 Configuración del PWM

En la Tabla 2.1 se muestra el código en lenguaje C para configurar una salida de PWM, de tal forma que active una conversión con el ADC en los cambios de estado. Esta configuración es conveniente debido a que estos cambios se generan en periodos fijos de tiempo.

La configuración del módulo PWM usa 28 registros de los cuales los más importantes son: SOCAEN, SOCASEL, SOCAPRD, CMPA, TBRD y CTRMODE. En la Tabla 2.1 se explica el uso de cada uno de ellos y se muestra un ejemplo, donde se configura el periodo y ciclo de trabajo del PWM, y la activación de una conversión con el ADC por PWM. Los pasos para configurar la salida de PWM son:

- Apagar la salida PWM para efectuar las modificaciones necesarias y obtener el comportamiento deseado.
- Se elige el registro CMPA para la comparación con el contador y ejecutar los cambios de estado. Se establece en 1 el número de cambios de estado a esperar, para realizar una conversión. Se seleccionó $CMPA = 1$ como límite superior del contador del PWM, es decir cada vez que el contador sea igual o mayor que el valor del registro CMPA habrá un cambio de estado en la salida del PWM.
- Se estableció el periodo del PWM en 25000 ciclos, con base en los cálculos mostrados en el paso 6, con lo cual se tiene un tiempo de muestreo de 100 µs.
- Se hace una pausa antes de activar el PWM.

	Descripción	Código en C
1	Habilitar escritura de registros importantes.	EALLOW;
2	Inhabilitar inicio de conversión del ADC para configurar el PWM.	EPwm1Regs.ETSEL.bit.SOCAEN = 0;
3	Seleccionar CMPA como límite superior (el cual se define en el paso 5) para el contador del PWM.	EPwm1Regs.ETSEL.bit.SOCASEL = 4;
4	Establecer el número de cambios de estado requeridos para generar una lectura.	EPwm1Regs.ETPS.bit.SOCAPRD = 1;
5	Establecer valor de CMPA que es el límite superior del contador del PWM. El valor actual es de 1; esto indica que el contador se inicializará en cada cambio de estado del PWM.	EPwm1Regs.CMPA.bit.CMPA = 0x0001;
6	Establecer periodo del PWM con la variable ciclos. La variable ciclos se calcula a partir del periodo de muestreo h requerido: $ciclos = 2 * h * 12500000$ Por ejemplo si se requiere un periodo de muestreo de $1ms$ se requerirán de 25000 ciclos. $ciclos = 2 * (1x10^{-3}) * 12500000 = 25000$	EPwm1Regs.TBPRD = ciclos;
7	Congelar contador para inicializar, posteriormente las conversiones.	EPwm1Regs.TBCTL.bit.CTRMODE = 3;
8	Inhabilitar la escritura de registros importantes.	EDIS;

Tabla 2.1 Configuración del PWM para inicializar el ADC a un periodo de muestreo constante

2.1.3 Módulo ADC

Un ADC convierte una señal analógica en una representación digital, con este proceso un dispositivo toma lecturas de fenómenos físicos mediante transductores y realiza cálculos conforme a esto. El F28377S cuenta con 4 módulos ADC 1-4. Cada módulo tiene las siguientes características.

- 2 ADC independientes (A y B).
- Resolución de 12 o 16 bits.
- Voltaje de referencia interna o externa.
- Modo de trabajo simple (de 12bits) para mediciones referenciadas a tierra.
- Modo de trabajo diferencial (de 16 bits) usando dos canales para la conversión.
- Se tiene 16 canales en modo simple u 8 canales en modo diferencial.
- 16 registros de resultado de la conversión.
- Modo de trabajo ráfaga de 3.5 MS/s.
- Lectura de la conversión por software o interrupción.

En la Figura 2.7 se muestra el diagrama de bloques funcional del convertidor analógico digital, el cual consta de 5 bloques que son: Muestreo y retención, Bloque de control, Prescalador, Registros de control y el convertidor ADC.

El bloque de muestreo y retención es un circuito que toma el nivel de voltaje de una señal y lo retiene el tiempo necesario para que pueda ser digitalizado. El bloque de control se encarga de

habilitar o deshabilitar el ADC. El prescalador se encarga de reducir la frecuencia de trabajo del reloj del ADC. Los registros de control del ADC se encargan del inicio de una conversión por software, interrupción, una señal interna o externa.

Cada módulo ADC cuenta con 2 canales multiplexables a 16 entradas: ADCINA0 a ADCINA7 y ADCINB0 a ADCINB7. Dichas entradas son capturadas por el circuito de muestreo y retención, y son digitalizadas por el ADC de 12 bits. El resultado de la conversión se almacena en un buffer de 16 localidades, el cual se puede configurar a máxima velocidad con un solo canal o a mínima velocidad con 16 canales.

El reloj HSPCLK es prescalado del reloj SYSCLKOUT de 200 MHz hasta un mínimo de 14.3 MHz. Los inicios de conversión (SOC Start Of Conversion) se activan mediante software o interrupción de periféricos (GPIO o PWM), sirven como activador de una conversión.

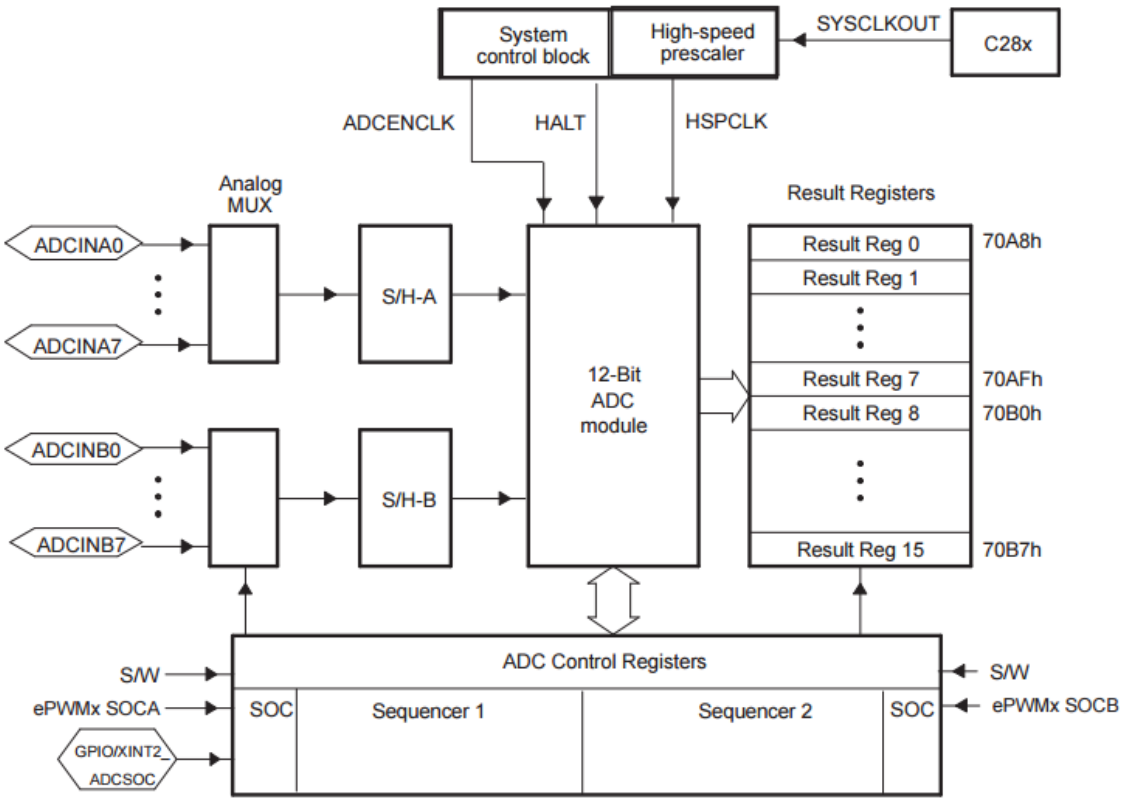


Figura 2.7 Diagrama del ADC

2.1.3.1 Configuración ADC

La activación de captura de datos con el ADC se hace por software o por Reloj principal, PWM u otro periférico. Para esta aplicación se realiza una conversión periódica mediante PWM la cual activa una interrupción. El ADC se utiliza en modo simple, es decir, se usa un canal con referencia a tierra para las conversiones.

El ADC en modo simple maneja una resolución de 12 bits, por lo cual tienen 4095 valores del rango de $0V - 3V$, esto equivale a $0.73mV/LSB$. Si el rango de operación se expande con el acondicionamiento analógico a un rango de $\pm 10V$ se tiene una resolución de $4.88mV/LSB$.

El seudocódigo para la configuración del ADC se muestra en la Tabla 2.2 y Tabla 2.3 donde el orden de las instrucciones es importante. El módulo ADC contiene 13 registros para su configuración, de los cuales los más importantes son: CHSEL, ACQPS, TRIGSEL, INT1SEL, INT1E, ADCINT1, PRESCALE, INTPULSEPOS y ADCPWDNZ.

El registro CHSEL es el selector de canales, toma valores entre [0,15] para definir el canal del módulo ADC a utilizar (ADCIN0 a ADCIN15). En este caso se utiliza el canal ADCIN0. Se selecciona PWM1 como el disparador. El registro INT1SEL indica la terminación de una conversión con 0.

	Descripción	**Código en C**
1	Habilitar escritura de registros importantes	EALLOW;
2	Se selecciona el canal ADCIN0 para la conversión.	AdcaRegs.ADCSOC0CTL.bit.CHSEL = 0;
3	Establecer tamaño de la ventana de adquisición mediante la fórmula. $$t = \frac{(14+1)}{200 \times 10^6} = 75 \times 10^{-9}$$ La ventana de adquisición se refiere al tiempo que el capacitor interno del ADC alcanza 0.5 LSB de la señal de entrada, para este caso se estableció en 14 para un tiempo de 75 ns.	AdcaRegs.ADCSOC0CTL.bit.ACQPS = 14;
4	Seleccionar PWM1 como disparador del ADC 1.	AdcaRegs.ADCSOC0CTL.bit.TRIGSEL = 5;
5	Establecer bandera para finalización de conversión.	AdcaRegs.ADCINTSEL1N2.bit.INT1SEL = 0;
6	Habilitar bandera para finalización de conversión.	AdcaRegs.ADCINTSEL1N2.bit.INT1E = 1;
7	Se habilita el ADC 1.	AdcaRegs.ADCINTFLGCLR.bit.ADCINT1 = 1
8	Inhabilitar escritura de registros importantes.	EDIS;

Tabla 2.2 Configuración del ADC

Con la configuración anterior se realiza una conversión con el ADC una vez que en el PWM se detecta un cambio de estado.

La frecuencia del reloj del ADC es de 25 MHz, lo cual se consigue dividiendo los 100 MHz del reloj de entrada entre 4. Se establece el modo de operación en modo simple con una resolución de 12 bits. Se configura la interrupción para que se genere al final de la conversión.

	Descripción	Código en C
1	Habilitar escritura de registros importantes.	EALLOW;
2	Establecer frecuencia del reloj del ADC con una división entre 4. Por tanto el registro PRESCALE de ADCCTL2 se establece en 6.	AdcaRegs.ADCCTL2.bit.PRESCALE = 6;
3	Establecer modo de operación y resolución del ADC. Modo simple a 12 bits.	**AdcSetMode**(ADC_ADCA, ADC_RESOLUTION_12BIT, ADC_SIGNALMODE_SINGLE);
4	El valor 1 de INTPULSEPOS indica que es al final de la conversión cuando se genera una interrupción.	AdcaRegs.ADCCTL1.bit.INTPULSEPOS = 1;
5	Encender el ADC.	AdcaRegs.ADCCTL1.bit.ADCPWDNZ = 1;
6	Esperar 1 milisegundo para que el ADC encienda.	DELAY_US(1000);
7	Inhabilitar escritura de registros importantes.	EDIS;

Tabla 2.3 Inicialización del ADC

2.1.4 Módulo DAC

Para generar una señal analógica a partir de un valor digital se usan DAC's. Una DAC se puede implementar usando una señal PWM y mediante filtros pasabajas o mediante redes resistivas. La tarjeta F28377S cuenta con 4 DAC con base en filtro pasabajas y generadores de PWM dedicados, que tienen frecuencia de operación fija. Sus principales características son:

- Resolución de 12 bits.
- Referencia de voltaje seleccionable (interna de 3.3V o externa hasta 5V).
- Buffer y resistencias tipo Pull-down en la salida.
- Habilidad de sincronizar la salida con una señal PWM o reloj SYSCLK.

En la Figura 2.8 se muestra el diagrama a bloques del módulo DAC, el cual consta de 4 bloques que son: bloque de actualización, bloque de referencia de voltaje máximo, DAC de 12 bits y buffer. A continuación se describe cada uno de ellos.

El bloque de actualización se encarga de renovar el valor digital a convertir a partir del uso de un PWM dedicado a la sincronización de eventos o una señal de reloj. Se tienen n señales de PWM para la actualización de la salida analógica, de las cuales se elige una señal PWM entre las disponibles ($n = 16$, para el F28377S), por otra parte se selecciona el reloj SYSCLK para obtener una actualización inmediata. Mediante Latches se captura el valor digital y con un multiplexor se selecciona la forma como será actualizada la salida analógica por PWM o Reloj SYSCLK. El valor actualizado es procesado por el DAC de 12 bits.

El bloque de referencia se encarga de definir el voltaje mínimo y máximo de la salida DAC. En este caso, VSSA y VDDA indican los límites de voltaje, mínimo y máximo en la salida analógica, respectivamente. El voltaje en VSSA por default es de 0V con respecto a tierra, cuando es diferente de 0V toma valores positivos menores al valor que tenga VDDA. El voltaje de VDDA es seleccionable entre VDAC (Referencia de voltaje externa hasta de 5V) y VHREFHI (Referencia de voltaje interna de 3.3V).

El bloque de buffer se usa para mantener el voltaje en la salida y limitarla conforme a los valores de VSSA y VDDA, cuando esta deshabilitado cuenta con una resistencia pull-down que permite tener un voltaje conocido en todo momento.

La configuración del DAC se hace a través de 13 registros, de los cuales los más importantes son: DACVALA (registro de sólo lectura del valor en la salida analógica), DACVALS (registro auxiliar de DACVALA editable por el usuario), DACVAL (candado para el registro DACVALS), LOADMODE (selecciona el modo de actualización de DACVALA), SYNCSEL (selecciona un PWM para sincronizar la salida analógica) y DACREFSEL (selecciona la referencia de voltaje a usar).

El registro DACVAL es un candado para controlar la escritura de DACVALS que se usa como variable auxiliar para actualizar a DACVALA, la cual representa el valor en la salida analógica. El registro LOADMODE determina, la forma en que DACVALA se actualiza con el valor de DACVALS, mediante reloj SYSCLK o una señal PWM. El registro SYNCSEL, selecciona entre 16 señales de PWM, la señal PWMSYNC que actualizará el valor del registro DACVALA. El registro DACREFSEL selecciona la referencia de voltaje interna o externa a usar por la DAC.

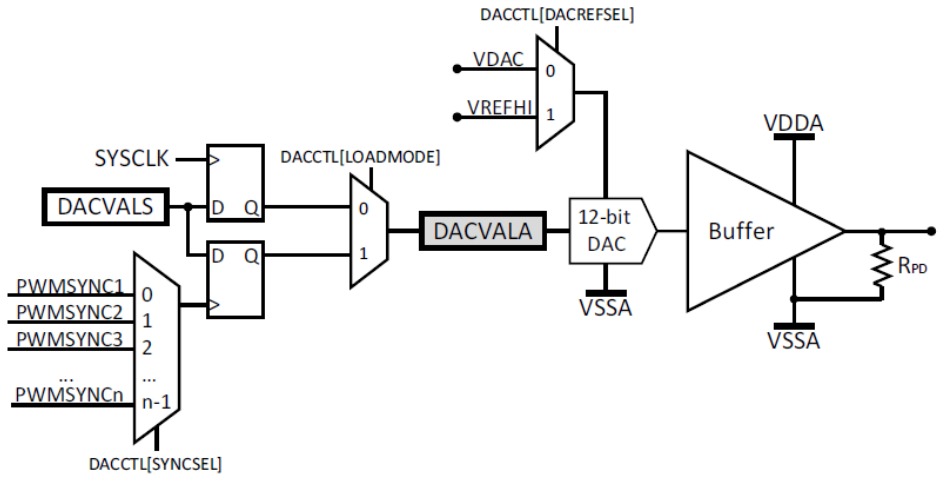


Figura 2.8 Diagrama de bloques del módulo DAC

En la Figura 2.9 se muestra el diagrama esquemático del filtro pasabajas del DAC, el cual extrae el voltaje promedio de la señal modulada en PWM. Como se ve el filtro es de tipo RC de primer grado (con R=1kΩ y C=0.1µF), lo que da una frecuencia de corte de 10 kHz.

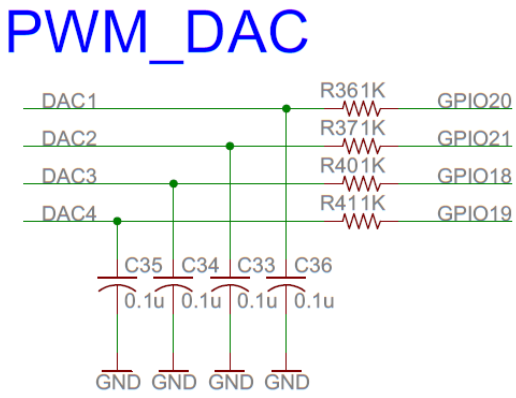


Figura 2.9 PWM y Filtro pasabajas

2.1.4.1 Configuración DAC

En la tabla 2.4 se muestra el código que realiza la configuración de la salida analógica. Se establece una referencia de voltaje de 3.3V, la velocidad de actualización del registro de salida analógica se sincroniza con la señal SYSCLK.

	Configuración DAC	Código en C
1	Habilitar escritura de registros importantes.	EALLOW;
2	Establecer la referencia de voltaje como 3.3 V, usando la referencia interna. Para usar una externa se asigna un valor de 0.	DacbRegs.DACCTL.bit.DACREFSEL=1;
3	El modo de actualización del registro DACVALA se realiza mediante el reloj SYSCLK. En caso de utilizar una señal de PWM se usa un valor de 1 para el registro LOADMODE.	DacbRegs.DACCTL.bit.LOADMODE=0;
4	Habilitar salida analógica. En caso de tener un valor de 0 en el registro DACOUTEN la salida analógica estará deshabilitada	DacbRegs.DACOUTEN.bit.DACOUTEN=1;
5	Inhabilitar escritura de registros importantes.	EDIS;

Tabla 2.4 Configuración DAC

2.1.5 Módulo SCI

El SCI es un protocolo de comunicación asíncrono y es frecuentemente usado de acuerdo al estándar RS232 de EIA (Electronics Industry Association) algunos de sus parámetros son: un “espacio” (0 lógico) está entre 3 y 25 V, una “marca” (1 lógico) estará entre -3 y -25 V, un circuito abierto que no excede 25 V con referencia a tierra y la región entre 3 y -3 es indefinida.

Para el F28377S, los niveles de voltaje son de 0V para el 0 lógico y 3.3V para el 1 lógico. El formato de la trama consta de las siguientes partes: Un bit de arranque, de 1 a 8 bits de datos, un bit opcional de paridad, 1 o 2 bits de parada y un bit extra para distinguir direcciones de los datos.

Este protocolo de comunicación es usado para recibir y enviar datos, de y hacia un dispositivo externo, que en esta aplicación es una computadora personal con una interfaz gráfica. El kit F28377S cuenta con 4 módulos SCI y cada uno tiene las siguientes características:

- 2 pines externos; transmisor y receptor, son usados como GPIO en caso de no usarse para comunicación serial.
- Habilidad de direccionar los pines por USB.
- Velocidad de transmisión programable a 64000 diferentes velocidades.
- 4 banderas de detección de error.
- Operación en modo Half o Full-duplex.
- Hardware de auto detección de velocidad.
- 2 pilas de datos con 16 niveles tipo FIFO.

En la Figura 2.10 se muestra el diagrama a bloques del protocolo de comunicación SCI del DSC los cuales son: El bloque de control de sistema, el prescalador y el módulo SCI.

El bloque de control de sistema habilita el módulo SCI. El prescalador cambia la frecuencia de trabajo del módulo, es decir, a partir del reloj principal obtiene una frecuencia máxima de 25 MHz y mínima de 381 Hz. El SCI cuenta con la capacidad de generar una interrupción cuando recibe o envía algún dato debido a que está conectado con la expansión periférica de interrupción. El CPU lee y escribe los datos enviados o recibidos y resetea al módulo SCI.

La configuración del módulo SCI se da con 13 registros de los cuales los más importantes son: SCITXBUF que es el buffer de datos a ser enviados. TXSHF toma los datos del buffer bit por bit y los asigna al SCITXD, que es el dato a enviar. Por otra parte el receptor tiene el registro RXSHF que toma los datos bit por bit y lo almacena en el buffer o SCIRXBUF, el cual es leído por el CPU. LSPCLK es un reloj que define la velocidad de transmisión. Los registros TXINT y RXINT generan una interrupción cuando se envía o recibe un dato, o por el número de palabras almacenadas en la pila FIFO.

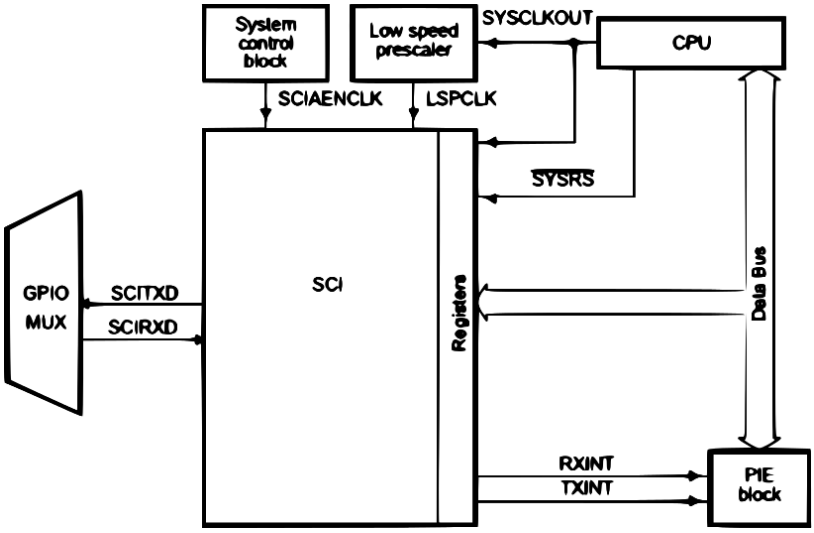


Figura 2.10 Diagrama de bloques del SCI

2.1.5.1 Configuración del SCI

En la Tabla 2.5 se muestra el código en C para inicializar y configurar el SCI donde el orden de las instrucciones es importante. En este caso se estableció una velocidad de 57600 baudios, sin paridad y 8 bits de datos, se inicializan las pilas tipos FIFO del transmisor y receptor de 16 niveles, se desactiva el modo de auto detección de velocidad, se deshabilita las interrupciones por transmisión

y recepción. Se configura el módulo de tal forma que cuando se resetea el sistema, se reinicia la comunicación.

	Configuración del SCI	Código en C
1	Inicializar FIFO del transmisor; se configuran los subregistros para asegurar el buen inicio y funcionamiento de la pila FIFO del transmisor. En la Tabla 2.6 se muestra a detalle el efecto que tiene el valor E040.	SciaRegs.SCIFFTX.all=0xE040;
2	Inicializar FIFO del receptor, la descripción del efecto del valor 2044 se muestra en la Tabla 2.7.	SciaRegs.SCIFFRX.all=0x2044;
3	Se deshabilita la auto detección de velocidad de transmisión y se define el retraso entre transferencias en este caso el retraso es de 0.	SciaRegs.SCIFFCT.all=0x0;
4	Inicializar registro del control de la comunicación. Estableciendo el número de bits de paro a utilizar en este caso se usa un bit de paro. Se establece en paridad impar.	SciaRegs.SCICCR.all =0x0007;
5	Configurar registro de control 1 con el valor 3: • Se deshabilita la interrupción cuando se presente un error en la transmisión y resetea las máquinas de estado. • Se deshabilita el modo dormir • Se habilita el transmisor y receptor.	SciaRegs.SCICTL1.all =0x0003;
6	Configurar registro de control 2. Con el valor 3 habilita la interrupción con la bandera TXRDY que indica cuando el buffer está listo para recibir una nueva trama.	SciaRegs.SCICTL2.all =0x0003;
7	Establecer velocidad de 57600 baudios. El valor de 53 hexadecimal se obtiene a partir de: $53 = \frac{LSPCLK}{57600 * 8} - 1$ Donde 57600 indica la velocidad deseada, y LSPCLK es el reloj que opera a 25 MHz en este caso.	SciaRegs.SCIHBAUD.all =0x0000; SciaRegs.SCILBAUD.all =53;
8	Configurar registro de control 1 con el valor 23 se rehabilita la comunicación SCI cuando se resetea el F28377S.	SciaRegs.SCICTL1.all =0x0023;
9	Configura los pines de salida 85 y 84 que están conectados al puerto USB.	**GPIO_SetupPinMux**(85, GPIO_MUX_CPU1, 5); **GPIO_SetupPinOptions**(85, GPIO_INPUT, GPIO_PUSHPULL); **GPIO_SetupPinMux**(84, GPIO_MUX_CPU1, 5); **GPIO_SetupPinOptions**(84, GPIO_OUTPUT, GPIO_ASYNC);

Tabla 2.5 Configuración del SCI

En el paso 1 se asigna el valor de los subregistros pertenecientes a SCIFFTX como lo muestra la Tabla 2.6. El registro SCIFFTX tiene un tamaño de 16 bits, el valor asignado E040 se traduce como 1110000001000000 en binario, los bits se asignan de acuerdo al lugar que ocupan en el registro, cabe mencionar que los bits de sólo lectura no pueden ser modificados.

Bit	Subregistro	Descripción

15	SCIRST=1	Reinicio del módulo SCI.
14	SCIFFENA=1	Mejoras de la pila FIFO habilitadas.
13	TXFIFORESET=1	Rehabilitar operación de transmisión de la pila FIFO.
12-8	TXFFST=0	Subregistro de sólo lectura e indica cuantas datos existen en la pila.
7	TXFFINT=0	Subregistro de sólo lectura e indica si alguna interrupción referente al envío de datos ha ocurrido, en este caso indica que no ha ocurrido una interrupción.
6	TXFFINTCLR=1	Bit de sólo lectura e indica el reseteo del subregistro TXFFINT
5	TXFFIENA=0	La interrupción basada en el subregistro TXFFIL esta deshabilitada con un valor de 0 y habilitada con un valor de 1.
4-0	TXFFIL=0	Indica el número de palabras permitidas en el FIFO de transmisión antes de generar una interrupción.

Tabla 2.6 Registro SCIFFTX

En el paso 2 se asigna el valor 2044 a los subregistros del registro SCIFFRX y la descripción se muestra en la Tabla 2.7.

Bit	Subregistro	Descripción
15	RXFFOVF=0	Es una bandera que indica cuando se desborda el FIFO pero genera una interrupción por sí misma. Un valor de 0 indica que no se ha desbordado la pila FIFO de recepción, la cual tiene un tamaño de 16 palabras.
14	RXFFOVRCLR=0	Cuando se requiere inicializar el bit RXFFOVF se asigna 1 al bit RXFFOVRCLR.
13	RXFIFORESET=1	Rehabilita la operación del FIFO del receptor
12-8	RXFFST=0	Subregistro de sólo lectura e indica el número de palabras presentes en el FIFO del receptor en este caso está vacío.
7	RXFFINT=0	Bit de sólo lectura e indica si alguna interrupción referente a la recepción de datos ha ocurrido, en este caso indica que no ha ocurrido una interrupción.
6	RXFFINTCLR=1	Bit para inicializar el bit RXFFINT en caso de asignarle un valor de 0 no tendrá efecto alguno.
5	RXFFIENA=0	La interrupción basada en el subregistro RXFFIL esta deshabilitada con un valor de 0 y habilitada con un valor de 1.
4-0	RXFFIL=4	Indica el número de palabras permitidas en el FIFO del receptor antes de generar una interrupción.

Tabla 2.7 Registro SCIFFRX

2.1.6 Módulo ISR

Las interrupciones son eventos asíncronos generados por señales externas o internas. Estos eventos provocan la pausa de la ejecución del programa actual y ejecutan una rutina de servicio, al terminar se continúa con la ejecución del programa pausado.

El F28377S cuenta con una expansión periférica de interrupción (PIE) que se muestra en la Figura 2.11. El PIE consta de 14 grupos de interrupción, de los cuales 2 están conectados directamente a los relojes principales CpuTimer1 y CpuTimer2, dichas interrupciones no son mascárables, es decir no se desactivan o bloquean por el usuario. Los 12 grupos de interrupción restantes individualmente manejan 16 interrupciones mascárables, lo que da un total de 192 interrupciones que son activadas mediante software o periféricos como son: PWM, ADC, SPI, SCI, I2C, CAN, unidad de punto flotante, entre otros; además del reloj principal CpuTimer0 y la estructura GPIO. El Delfino F28377S cuenta con 141 registros para generar las interrupciones.

Los registros IFR (Interrup Flag Register) e IER (Interrupt enable Register) se utilizan cuando las interrupciones se activan por software mientras el registro INTM maneja las interrupciones por

hardware. La expansión periférica de interrupción se diseñó con el fin de administrar las interrupciones y liberar al CPU de dicha tarea.

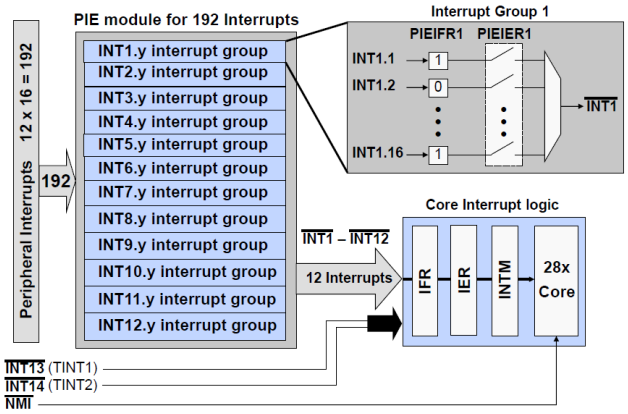


Figura 2.11 Diagrama de la expansión periférica de interrupción

En la Figura 2.12 se muestra que es posible generar una interrupción mascarable con el reloj principal CpuTimer0, periféricos, X-BAR o cuando se reactiva el F28377S después de una suspensión. Estas interrupciones son gestionadas por los grupos de interrupción 1 a 12. Las entradas X-BAR son pines del módulo GPIO para activar una interrupción mascarable de forma externa con un máximo de 5 pines para 5 subrutinas.

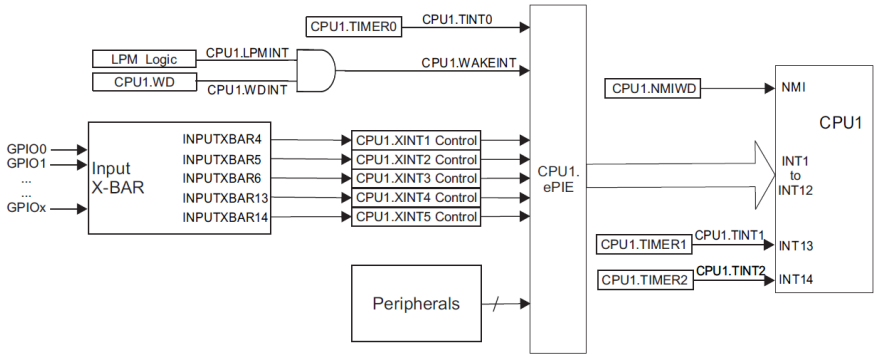


Figura 2.12 Diagrama del ISR

La prioridad en un grupo de interrupciones decrece entre más alto es el número del grupo de interrupción, es decir el grupo de interrupción 1 tiene la más alta prioridad mientras el grupo 14 tiene la prioridad más baja. Lo mismo sucede dentro de un grupo de interrupción el cual maneja 16 líneas, la línea 1 tiene mayor prioridad que la línea 16.

2.1.6.1 Configuración del ISR

En esta aplicación se configura el sistema para que se genere una interrupción en cada conversión del ADC, el cual es activado mediante PWM. En la Tabla 2.8 se muestra el código en C y lo que se requiere para lograrlo.

Las instrucciones en los pasos 1-4 y 6, son funciones predefinidas que se usan para inicializar los periféricos a su estado por default, esto asegura el buen funcionamiento de los módulos. Se

resetean los registros indicando que no existe una interrupción activa. Se seleccionó el grupo de interrupción 1 para resolver el modelo dinámico. Se definió el ADC como periférico que genera una interrupción cuando se completa una conversión. En este caso la función que resuelve el modelo dinámico se llama "adca1_isr".

	Configuración del ISR	Código en C
1	Resetear sistemas de control: PLL, WhatchDog y Relojes periféricos.	**InitSysCtrl**();
2	Resetear salidas y entradas de propósito general (GPIO).	**InitGpio**();
3	Limpieza de interrupciones e inhabilitación de interrupciones.	DINT;
4	Resetear registros del sistema de interrupciones.	**InitPieCtrl**();
5	Resetear banderas del sistema de interrupciones en este caso se asigna el valor de 0 a todos los bits e indica que actualmente no hay alguna interrupción en servicio.	IER = 0x0000; IFR = 0x0000;
6	Inicializar apuntadores de la rutina de servicio de interrupciones	**InitPieVectTable**();
7	Se configura el grupo de interrupciones 1 para generar una interrupción cuando se tenga una conversión del ADC 1 donde "adca1_isr" es el nombre de la función de subrutina a ejecutar.	EALLOW; PieVectTable.ADCA1_INT = &adca1_isr; EDIS;
8	Habilitar interrupciones tanto para software como hardware.	IER |= M_INT1 EINT; ERTM;
9	Habilitar el primer grupo de interrupciones.	PieCtrlRegs.PIEIER1.bit.INTx1 = 1;

Tabla 2.8 Configuración del ISR

2.1.7 Salida de Encoder

El Encoder es un transductor de posición que genera pulsos digitales en función del movimiento rotacional o lineal que se tenga, de tal manera que contando el número de pulsos generados se conoce el desplazamiento. En esta aplicación se emuló una salida de encoder usando las salidas digitales del DSC. Una señal de Encoder tiene las siguientes características:

- 2 señales, que normalmente se le llaman A y B para indicar dirección y posición.
- 1 señal llamada índex que genera un pulso cuando se cruza el origen del Encoder.
- Una resolución específica, en este caso 1000 pulsos por revolución (ppr).

En la Figura 2.13 se muestran los estados que sigue una salida de Encoder, existe un desfase de medio periodo o 90° entre las salidas A y B.

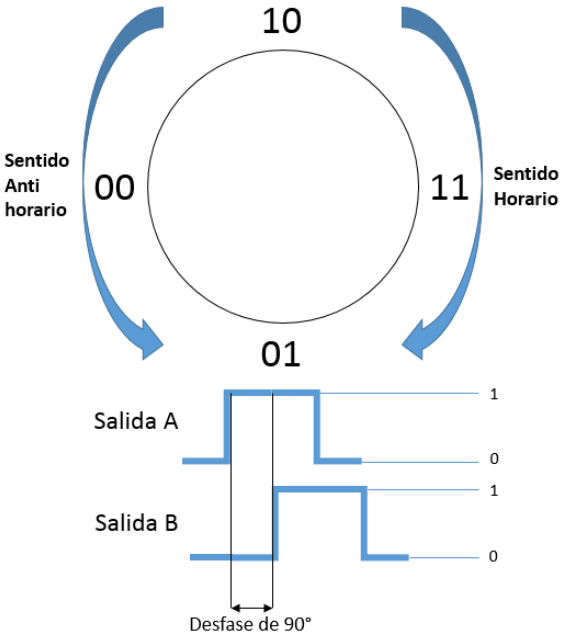


Figura 2.13 Diagrama de estados del Encoder

Se diseñó una función, que se ejecuta en el DSC, que emula un Encoder mediante la manipulación de 3 salidas digitales. En esta aplicación se requiere emular 2 Encoders, cada uno de los cuales tiene el Index y los canales A y B, por tanto se usaron 6 pines. Se estableció una resolución de 1000 pulsos por revolución.

En la Figura 2.14 se muestra el diagrama de flujo de la salida de Encoder. Se inicializan las variables a utilizar, las cuales son:

- $anterior$ es la posición de la articulación antes de resolver el modelo dinámico.
- $actual$ es la posición de la articulación después de resolver el modelo dinámico.
- $resolución$ se obtiene a partir de dividir 2π entre el número de pulsos por vuelta, en este caso se tiene $resolución = \frac{2\pi}{1000} = 2\pi \times 10^{-3}$ radianes.
- $acumulador$ es una variable auxiliar para almacenar cambios de posición menores a la $resolución$.

El algoritmo tiene el siguiente orden en su ejecución: se resuelve el modelo dinámico, se lee la posición actual y se calcula el error con respecto a la anterior. El error se suma a un acumulador, se verifica que sea mayor o igual a la resolución, en caso afirmativo, se calcula el número de pulsos a generar con la salida digital y se repite el proceso al resolver el modelo dinámico.

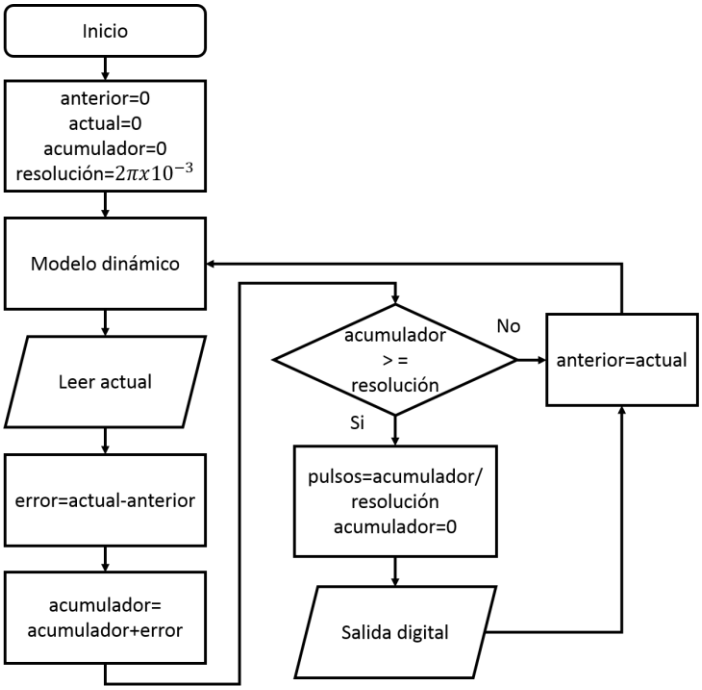


Figura 2.14 Diagrama a bloques del Encoder

El diagrama de flujo de la Figura 2.15 muestra el algoritmo que genera los pulsos a partir de los estados del encoder de la Figura 2.13. Las variables principales son:

- $pulsos$ es la variable de entrada, se refiere al número de pulsos a generar con la salida digital que se obtiene del diagrama de la Figura 2.14.
- $signo$ es la dirección de giro. Un +1 indica que el giro es en sentido contrario a las manecillas del reloj. Un -1 es un giro en sentido de las manecillas del reloj.
- n es un contador descendente que inicia con el valor de $pulsos$ y termina en 0.

El algoritmo se ejecuta de la siguiente forma, se obtiene la dirección de giro que se guarda en la variable *signo*. Se guarda el valor absoluto de *pulsos* en un contador descendente. Usando el signo del giro y con el contador descendente se generan pulsos hasta que tiene un valor de 0.

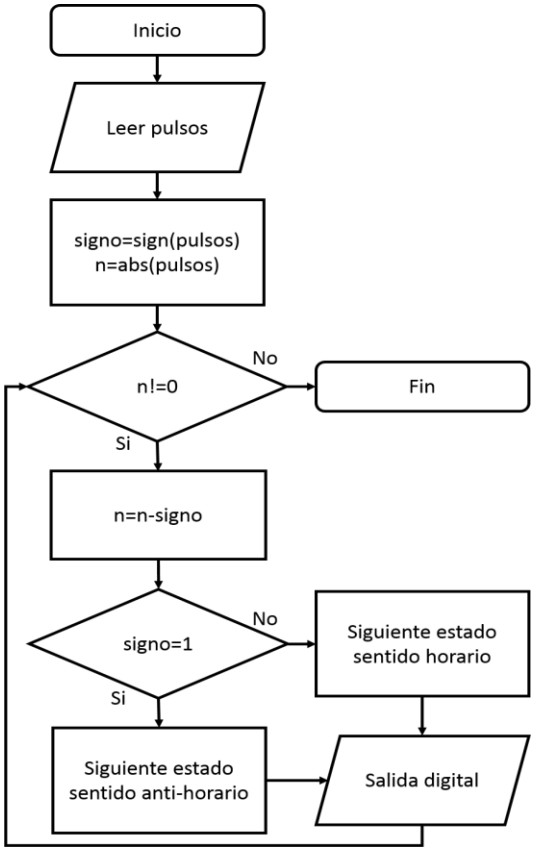


Figura 2.15 Salida digital

En el siguiente ejemplo se muestra el funcionamiento del algoritmo de la salida de encoder.

Supongamos que se define a $resolución = 2\pi \times 10^{-3} rad$, $acumulador = 0\, rad$, $anterior = 0\, rad$, posteriormente se resuelve el modelo dinámico del robot y se obtiene una nueva posición $actual = \pi \times 10^{-3}\, rad$, se calcula la diferencia entre las posiciones $actual$ y $anterior$, añadiéndola al acumulador. Por tanto $acumulador = acumulador + (actual - anterior) = 0 + (\pi \times 10^{-3}\, rad + 0) = \pi \times 10^{-3}\, rad$. Basándose en el algoritmo de la Figura 2.14 se puede observar que el valor de $acumulador$ es menor al valor de $resolución$.

Por tanto $anterior = actual = \pi \times 10^{-3}\, rad$. Se resuelve nuevamente el modelo dinámico obteniendo una nueva posición $actual = 4\pi \times 10^{-3} rad$, se suma la diferencia de posición al $acumulador = acumulador + (actual - anterior) = \pi \times 10^{-3} + (4\pi \times 10^{-3} - \pi \times 10^{-3}\, rad) = 4\pi \times 10^{-3}\, rad$. Se observa que $acumulador$ es mayor que $resolución$, por tanto $pulsos = \frac{acumulador}{resolución} = \frac{4\pi \times 10^{-3}}{2\pi \times 10^{-3}} = 2$. Por tanto el estado del encoder se moverá 2 veces en sentido anti-horario como se muestra en la Figura 2.13.

Con el fin de que la salida de Encoder sea compatible con la mayoría de tarjetas, se usa un convertidor de niveles lógicos bidireccional, en este caso se utilizó la tarjeta BOB-12009 de la firma SparkFun Electronics (SE). La cual corre los niveles lógicos de 3.3V a TTL. En la Figura 2.16 a) se muestra el circuito que se alimenta con ambos niveles de voltaje, es decir, con 3.3V en LV y 5V en HV referenciados a tierra, los pines TX_LV o TX_HV sirven como entradas/salidas dependiendo del

pin que sea alimentado. En este caso para convertir de 3.3V a 5V, se conecta 3.3V en el pin TX_LV y en el pin TX_HV se tienen 5V.

La tarjeta BOB-12009 se basa en transistores tipo MOSFET de montaje superficial, los cuales cambian los voltajes de saturación y corte lo que es útil para esta aplicación. En la Figura 2.16 b) se muestra una vista superior de la tarjeta, la cual consta de 4 líneas de conversión, sus entradas, salidas y pines de alimentación.

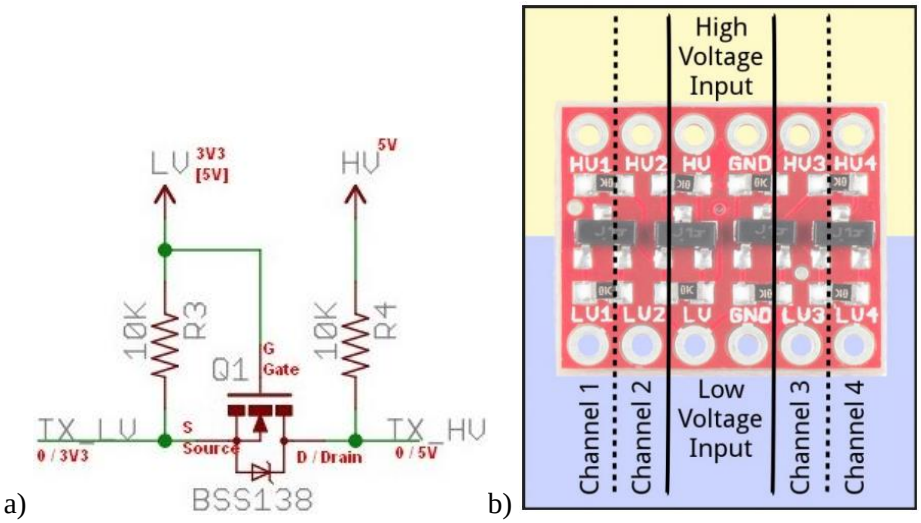


Figura 2.16 Convertidor de niveles lógico bidireccional BOB-12009

2.1.7.1 Configuración Encoder

En la Tabla 2.9 se ve el código en C para configurar las salidas de Encoder donde es importante el orden de la primer y última instrucción.

Se usaron 3 pines del módulo GPIO y se configuraron como salidas digitales, en este caso se usaron los pines 12 y 13 como canal A y B para indicar cambio de posición y dirección, el pin 14 se usa como Índex.

	Configuración del Encoder	Código en C
1	Habilitar escritura de registros importantes.	EALLOW;
2	Al configurar el canal A El pin 12 es seleccionado como GPIO asignando al registro GPAMUX1 un valor de 0 y se configura como pin de salida digital asignando al registro GPADIR un valor de 1. Se configura igual al pin 13 y 14 en el paso 3 y 4.	GpioCtrlRegs.GPAMUX1.bit.GPIO12 = 0; GpioCtrlRegs.GPADIR.bit.GPIO12 = 1;
3	Configuración del canal B	GpioCtrlRegs.GPAMUX1.bit.GPIO13 = 0; GpioCtrlRegs.GPADIR.bit.GPIO13 = 1;
4	Configuración del Index	GpioCtrlRegs.GPAMUX1.bit.GPIO14 = 0; GpioCtrlRegs.GPADIR.bit.GPIO14 = 1;
5	Inhabilitar escritura de registros importantes.	EDIS;

Tabla 2.9 Configuración del Encoder

2.2 Acondicionamiento de señales

Para adecuar los niveles de voltaje entre el sistema empotrado y el controlador se requiere de un módulo de acondicionamiento de señales, el cual mueve los niveles de voltaje entre ambos bloques. El circuito que realiza esta función, para señales analógicas, se le conoce como CAS (Circuito Acondicionador de Señales) y se realiza con amplificadores operacionales. Para las señales analógicas se requiere de dos CAS, uno de subida que incrementa el voltaje y el de bajada que lo reduce.

2.2.1 CAS de subida

El CAS de subida se muestra en la Figura 2.17, se usa para obtener una señal de voltaje en la salida de $v_{out1} = \pm 10V$ a partir de una señal $v_{in1} = [0,3]V$. Esto con la finalidad de que el ERM2GDL sea compatible con la mayoría de tarjetas de adquisición de datos sin pérdida de resolución. La entrada V_{offset} es una señal de voltaje fija de 1.5V.

En la entrada de la señal v_{in1} se tiene un seguidor de voltaje para eliminar los efectos de impedancias que se puedan presentar. La Ecuación 2.1 es la función de transferencia del CAS de subida, donde v_{in1} es el voltaje de entrada, V_{offset} es un voltaje fijo y v_{out1} es la señal de salida.

$$v_{out1} = \frac{20}{3}\left(v_{in1} - V_{offset}\right) \tag{2.1}$$

2.2.2 CAS de bajada

El acondicionamiento analógico de bajada se muestra en la Figura 2.17, se usa para obtener a la salida un señal de voltaje de $v_{out2} = [0,3]V$ con una señal en la entrada $v_{in2} = \pm 10V$. La entrada es la señal v_{in2}, tiene un seguidor de voltaje para eliminar los efectos de impedancias que se puedan presentar.

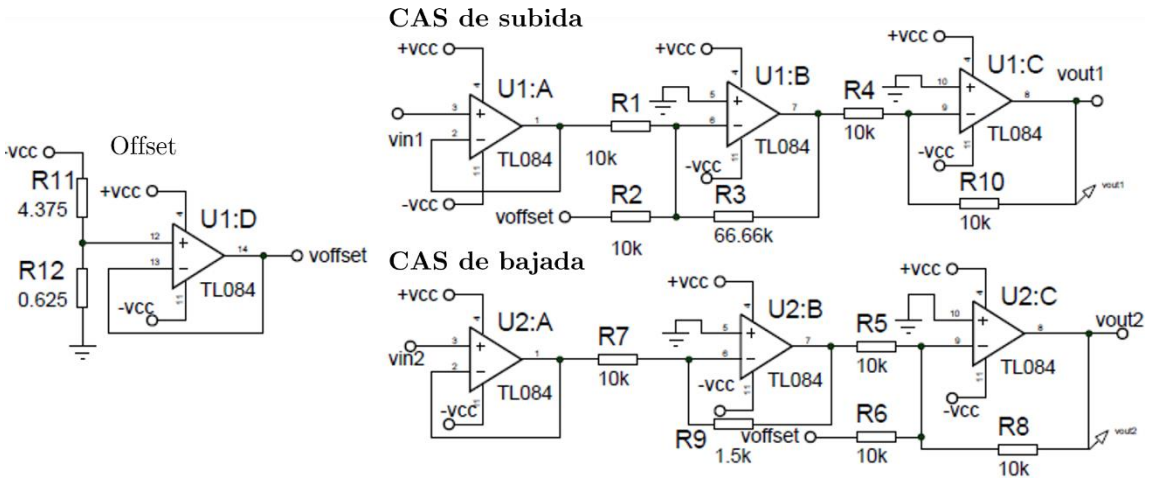


Figura 2.17 CAS de subida y bajada

La Ecuación 2.2 es la función de transferencia del CAS de bajada, donde v_{in2} es el voltaje de entrada, V_{offset} denota un voltaje fijo y v_{out2} es la señal de salida. El voltaje de offset se consigue mediante un divisor de voltaje de tal forma que se obtenga -1.5V.

$$v_{out2} = \frac{3v_{in2}}{20} + V_{offset} \tag{2.2}$$

2.3 Controlador con base en DAQ 6062E y PC

El controlador del robot de 2 GDL se hizo con una tarjeta de adquisición de datos y una computadora personal. Para facilitar la implementación de los algoritmos de control se utilizó Matlab/Simulink. La tarjeta usada es la DAQ 6062E de la firma National Instruments [72], la que se muestra en la Figura 2.18. Esta tarjeta se conecta a un bus PCMCIA que fue ampliamente usado en computadoras portátiles, el cual ya está descontinuado. Sus principales características son:

- Entrada analógica
  - 16 canales en modo simple
  - 8 canales en modo diferencial
  - Canales seleccionables por software
  - Tipo de conversión por aproximación sucesiva
  - Resolución de 12 bits
  - Máxima frecuencia de muestreo: 500 kS/s
  - Rango de $\pm 10V$ hasta $\pm 50mV$
- Salida analógica
  - 2 salidas
  - Resolución de 12 bits
  - Frecuencia de generación de 850 kS/s
  - Referencia externa máxima de 11V
  - Impedancia de salida de 0.1 Ω
  - Acoplamiento DC
  - Salida seleccionable por software
- Salidas digitales

- 8 pines I/O con voltaje configurable máximo de 5V (TTL)
- Reloj a 20 MHz
- Resistencias pull-up

Figura 2.18 DAQ 6062E

La DAQ 6062E se conecta al puerto PCMCIA de la computadora y se interconecta a un bloque de conexiones RC68-68 con un cable plano con conector tipo D de 68 pines. El bloque de conexiones contiene tornillos para facilitar la sujeción de los cables que contienen las señales eléctricas (Figura 2.19). Una vez que se conecta se instalan los controladores y desde Matlab/Simulink se pueden utilizar los recursos de éste.

Figura 2.19 Diagrama a bloques del controlador

Capítulo 3. Diseño del Software

En este capítulo se presenta el diseño del software que controla el funcionamiento del ERM2GDL, el cual consta de dos partes, la Interfaz Gráfica (IG) basada en LabVIEW y el software embebido en el DSC F28377S. En las siguientes secciones, se explica la forma en que fueron desarrollados e implementados.

En la Figura 3.1 se muestra el diagrama conceptual del diseño del software, que constituye a todo el sistema, el software embebido y la IG se interconectan por medio de una conexión serial por USB. La IG realiza peticiones de datos al DSC de la posición, velocidad y par del robot emulado. La IG usa estos datos para mostrar el movimiento del robot y generar gráficas con el historial de su comportamiento.

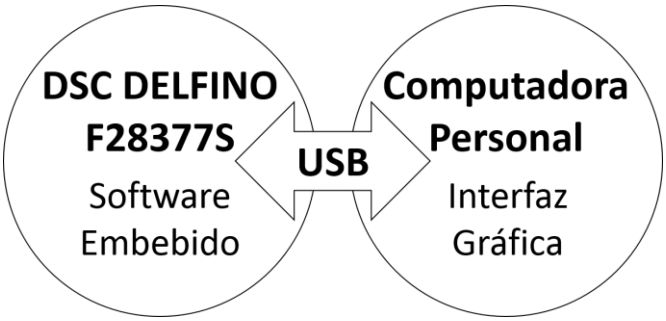


Figura 3.1 Diagrama conceptual del Software

El diseño de ambas partes, se hizo de forma paralela debido a que están estrechamente relacionadas. Para el desarrollo de esta fase, se utilizó el modelo incremental (Figura 3.2), para la mejora continua de todo el software, basada en los pasos: *definición y análisis de requerimientos, diseño, codificación y pruebas del software,* que establece [73]. La fase inicial es recabar los requerimientos iniciales, los que son analizados y a partir de esto se divide en módulos, se codifican y se hacen pruebas. En caso de que no cumpla con los requerimientos, se repite el proceso.

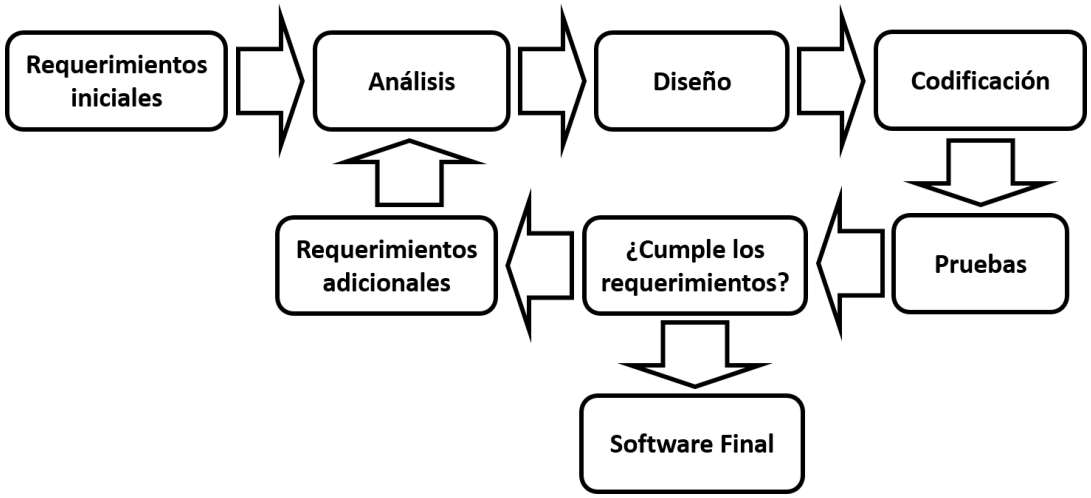


Figura 3.2 Pasos del modelo incremental

3.1 Modelo incremental aplicado al ERM2GDL

La aplicación del modelo incremental en el diseño del ERM2GDL se explica en las siguientes subsecciones. El software embebido en el DSC, tiene la función de obtener y procesar las entradas al sistema, resolver numéricamente las ecuaciones diferenciales del modelo dinámico del robot manipulador, generar las salidas correspondientes en tiempo real e interactuar con la IG.

3.2 Requerimiento general del ERM2GDL

Emular el comportamiento de un robot manipulador de 2 GDL en tiempo real con base en un controlador digital de señales, mostrar su movimiento en un ambiente virtual y establecer comunicación por puerto USB, lo que permite cambiar los parámetros del sistema en tiempo de ejecución y contar con el historial de comportamiento del mismo.

3.2.1 Requerimientos específicos

Los requerimientos específicos son las acciones que el software diseñado debe cumplir, los cuales se dividen en 2 grupos.

Requerimientos del software embebido en el DSC.

- Leer los voltajes del ADC y escalarlos para su manipulación.
- Resolver el modelo dinámico en tiempo real.
- Generar salidas analógicas y digitales con la posición de las articulaciones por medio de un DAC y un encoder.
- Mediante un protocolo de comunicaciones, enviar los datos de posición, velocidad y par del robot, por puerto serie.
- Recibir comandos para configurar parámetros físicos del modelo dinámico y cambiar el modo de funcionamiento del DSC.

Requerimientos de la interfaz gráfica.

- Protocolo de comunicación serial para pedir información del robot y cambiar los parámetros físicos del modelo.
- Interfaz con botones para enviar comandos por puerto serie al DSC, el cual modifica los parámetros del modelo.
- Mostrar los datos recibidos en graficas de posición, velocidad y par de entrada.
- Mostrar el movimiento del Robot Manipulador de 2 GDL en un ambiente virtual.

3.3 Análisis de requerimientos

En la parte de análisis se establecieron los elementos del funcionamiento del sistema, tomando en cuenta los requerimientos específicos. Se analizaron conceptualmente los módulos de cada parte del software para establecer los recursos necesarios y la solución que tendrían.

3.3.1 Análisis conceptual

El análisis conceptual proporciona las pautas para el cumplimiento de los requerimientos del software al establecer una serie de módulos, que como conjunto dan solución al requerimiento general.

Para cubrir los requerimientos del Software embebido, éste se divide en 5 módulos:

- **Módulo de comunicaciones.** Se encarga de establecer el protocolo de comunicación serial, enviar datos del estado del robot y recibir comandos de modificación de parámetros físicos.
- **Módulo de obtención y procesamiento de entradas analógicas.** Este módulo utiliza los ADC's para convertir el voltaje en las entradas en datos binarios y los escala para ser incorporados en la solución del modelo dinámico.
- **Módulo de solución del modelo dinámico.** Mediante un método numérico, le da solución al modelo dinámico del robot, tomando como entradas el par y el estado actual del robot (posición y velocidad angular de las articulaciones) obteniendo un nuevo estado del robot.
- **Módulo de salida de Encoder.** Genera pulsos proporcionales al desplazamiento que tenga cada articulación.
- **Módulo de salida analógica.** Genera voltajes analógicos proporcionales a la posición angular de cada articulación.

El software de la interfaz gráfica se divide en 4 módulos con los que se cumple con los requerimientos de diseño.

- **Módulo de comunicaciones.** Establece el protocolo de comunicación serial, genera comandos para modificar parámetros del modelo dinámico y recibe los datos posición, velocidad y par.
- **Módulo de procesamiento.** Procesa los datos para mostrarlos en gráficas y en un ambiente virtual en la IG.
- **Interfaz de usuario.** Se encarga de mostrar las gráficas de par de entrada, posición y velocidad angular. Proporciona al usuario un control sobre los parámetros físicos del robot de 2 GDL y del funcionamiento del DSC.
- **Ambiente virtual.** Muestra los movimientos del robot a partir de la posición angular recibida.

3.4 Diseño de los módulos

El diseño consistió en la estructuración de los módulos en diagramas a bloques que sirvieron para la codificación y para realizar las pruebas. Los diagramas a bloques se presentan en las siguientes secciones. La subsección 3.4.1 describe los módulos del software embebido en el DSC y la subsección 3.4.2 de la interfaz gráfica.

3.4.1 Software Embebido

El software embebido contiene las funciones necesarias para configurar los periféricos; procesar las entradas y realizar los cálculos requeridos para la solución del modelo dinámico del robot, además, genera las salidas correspondientes. El lenguaje de programación usado es C y se utilizó el ambiente de programación Code Composer Studio de Texas Instruments. En la Figura 3.3 se muestra el diagrama a bloques en que se divide el software, donde:

- $\boldsymbol{V_{in}}$ son los voltajes de entrada al ADC.
- $\boldsymbol{B_{in}}$ son los datos leídos con el ADC y representa el par de entrada en cada articulación.
- $\boldsymbol{\tau_A}$ los valores escalados a partir de $\boldsymbol{B_{in}}$ e indica el par leído de las entradas analógicas en $[Nm]$.
- $\boldsymbol{\tau_T}$ representa el par total que es la suma del par de perturbación y del par en el modo de trabajo.
- $\boldsymbol{\ddot{q}, \dot{q}\ y\ q}$ son la aceleración, velocidad y posición angular, de cada articulación en $[rad^2/seg]$, $[rad/seg]$ y $[rad]$.
- $\boldsymbol{B_{out}}$ es la posición angular de cada articulación.
- $\boldsymbol{V_{out}}$ representa el voltaje de salida generado por los DAC, que es la posición angular de cada articulación.
- Se tiene un módulo de comunicación que recibe por puerto serie una petición y envía la información de la posición, velocidad y del par de entrada.

La entrada ADC convierte el voltaje que representa al par en un valor digital que se usa en la solución del modelo dinámico. Como salida se tiene la aceleración, la cual es integrada por el método numérico, obteniendo posición y velocidad. Dicha posición se usa para generar la salida analógica y para el Encoder. El par, la posición y velocidad se envían por puerto serie a la IG.

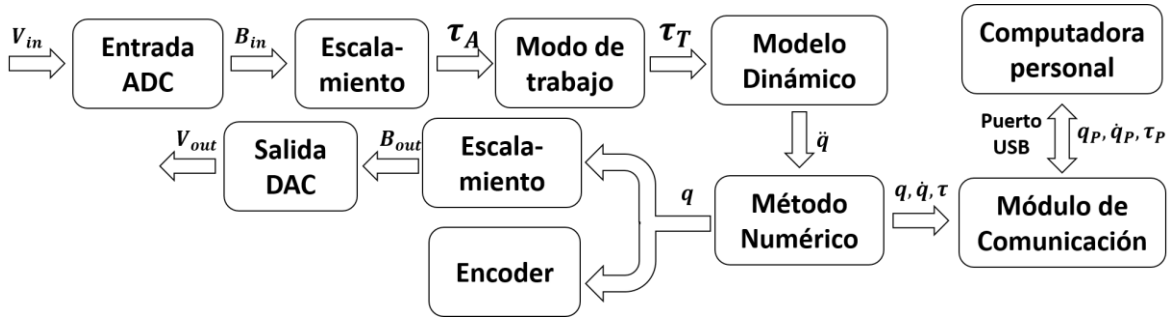


Figura 3.3 Diagrama a bloques del software embebido en el DSC

3.4.1.1 Entrada ADC y escalamiento

El sistema embebido requiere de un par de entrada para cada articulación, dicha información se consigue a partir de una señal de voltaje externa al DSC. Por tanto, para obtener el valor del par, se usa un ADC que convierte el voltaje de entrada a un dato digital de tipo binario, este valor binario a su vez, es escalado para representar el par aplicado en $[Nm]$.

Los voltajes en las entradas del ERM2GDL deben estar en el rango de $[0, 3]V$ lo que produce un valor binario entre $[0, 4095]$. Este dato se escala en función del par máximo aplicable en cada articulación. Para escalar el valor obtenido con el ADC, se usa la Ecuación (3.1) de la cual resulta un valor representado en $[Nm]$, donde $\boldsymbol{B_{in}}$ es el valor a escalar y $\boldsymbol{\tau_M}$ es el par máximo que puede aplicar el motor en la articulación y $\boldsymbol{\tau_A}$ es el par leído en $[Nm]$.

$$\boldsymbol{\tau_A} = \frac{(\mathbf{B_{in}} - 2048)}{2048}\boldsymbol{\tau_M} \tag{3.1}$$

Para esta aplicación en particular se tomó el par máximo en la primer y segunda articulación de acuerdo a la Tabla 1.1 como: $\tau_{M1} = 50\ Nm$ y $\tau_{M2} = 15\ Nm$.

Para el escalamiento se toma $\boldsymbol{B_{in}}$ y $\boldsymbol{\tau_M}$ como entradas, lo cual produce $\boldsymbol{\tau_A}$ como salida. El par resultante está en el rango de $\boldsymbol{\tau_A} = \pm\boldsymbol{\tau_M}$. Por ejemplo, un valor $\boldsymbol{B_{in}} = 0$ equivale a $\boldsymbol{\tau_A} = -\boldsymbol{\tau_M}$. Un valor $\boldsymbol{B_{in}} = 2048$ equivale a $\boldsymbol{\tau_A} = 0$ y un valor $\boldsymbol{B_{in}} = 4095$ equivale a $\boldsymbol{\tau_A} \cong \boldsymbol{\tau_M}$.

El diagrama de flujo de la Figura 3.4 es el que implementa la Ecuación (3.1). Al inicio se lee el dato del ADC (Bin), se escala para obtener el par de cada articulación (TaoA). En este caso el par máximo (TaoM) es de 50 y 15 Nm.

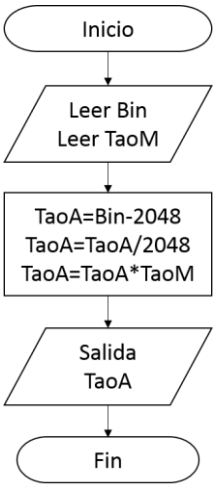


Figura 3.4 Escalamiento

3.4.1.2 Modos de trabajo

Para tener diferentes formas de operar el robot se definió el módulo de modos de trabajo, el cual permite trabajar con un controlador interno, con entrada cero o controlado desde las entradas analógicas. Estos tres modos de trabajo se describen en la Tabla 3.1. Este modo es configurado por los datos que se reciben por el puerto serie.

Modo de trabajo	Descripción
Par cero	El valor de $\boldsymbol{\tau_A}$ (par de entrada) es cero independientemente del voltaje en la entrada de los ADC's, sirve para realizar pruebas en lazo abierto.
Par de PID interno	El valor de $\boldsymbol{\tau_A}$ se calcula usando un controlador PID de posición a partir de las referencias $ref1$ y $ref2$. Sirve para obtener posiciones deseadas y después cambiar al modo **Par de entradas analógicas** para aplicar algún controlador externo.
Par de entradas analógicas	El valor de $\boldsymbol{\tau_A}$ se obtiene usando los ADC's y se escala.

Tabla 3.1 Modos de operación del ERM2GDL

La salida de este módulo es $\boldsymbol{\tau_T}$, que es la suma de un par generado y el de perturbación $\boldsymbol{\tau_A}$. En la Figura 3.5 se muestra el diagrama de flujo que realiza este módulo. Su salida es el par de cada articulación el cual se le llama *TaoT*, las variables de entrada son:

- *modo* representa el modo de trabajo, toma valores entre 1 y 3.
- *error* es la diferencia aritmética entre la posición del robot y la referencia de posición deseada.
- *Pert* es un par de perturbación que se modifica desde la interfaz gráfica.

Para ejemplificar su funcionamiento se plantean los siguientes casos.

- **Par cero**: Se tiene *modo=1* y *Pert=10* por lo que *TaoA=0*, se tiene como salida *TaoT=TaoA+Pert=0+10=10*.
- **Par de PID interno**: Se tiene *modo=2* y *Pert=1* por lo que *TaoA* toma el valor que produce el controlador PID el cual está en función del *error*, por simplicidad se toma *TaoA=PID(error)=3*. Se tiene como salida *TaoT=TaoA+Pert=3+1=4*.
- **Par de entradas analógicas**: Se tiene *modo=3* y *Pert=4* por lo que *TaoA* se captura con el ADC y se escala. Por simplicidad *TaoA=5*, teniendo como salida *TaoT=TaoA+Pert=5+4=9*.

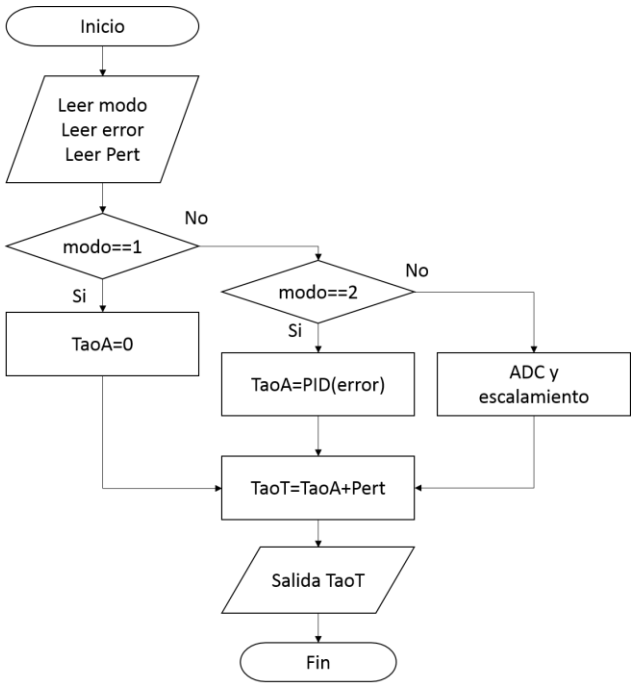


Figura 3.5 Modos de trabajo

3.4.1.3 Solución del modelo dinámico

El modelo dinámico del robot de 2 grados de libertad se resuelve al discretizar el sistema de ecuaciones, usando un método numérico que aproxime la solución. El proceso de discretización incluye resolver el problema de las dinámicas acopladas que se presentan. En esta aplicación se usó el método de Runge-Kutta de orden 4 y el de Euler.

Las entradas de las ecuaciones del modelo incluyen el estado actual del robot $\dot{q}_1(t_{i-1})$, $\dot{q}_2(t_{i-1})$, $q_1(t_{i-1})$, $q_2(t_{i-1})$ y el par de entrada (τ_1 y τ_2). Las salidas son las aceleraciones de cada articulación $\ddot{q}_1(t_i)$ y $\ddot{q}_2(t_i)$, las cuales se usan como entradas para el método numérico que calcula un nuevo estado $\dot{q}_1(t_i)$, $\dot{q}_2(t_i)$, $q_1(t_i)$, $q_2(t_i)$.

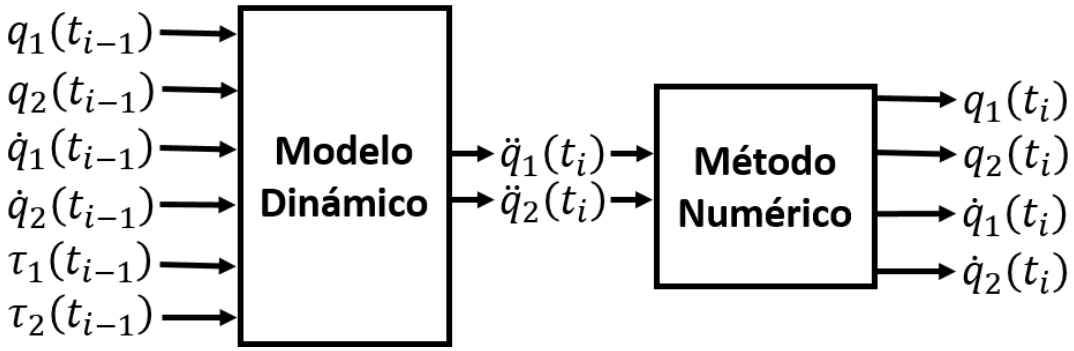


Figura 3.6 Solución del modelo dinámico

3.4.1.3.1 Discretización del modelo dinámico

A partir de las Ecuaciones (1.10) a (1.22) se tiene:

$$M_{11}\ddot{q}_1 + M_{12}\ddot{q}_2 + C_{11}\dot{q}_1 + C_{12}\dot{q}_2 + g_1 + f_{f1} = \tau_1 \tag{3.2}$$

$$M_{21}\ddot{q}_1 + M_{22}\ddot{q}_2 + C_{21}\dot{q}_1 + C_{22}\dot{q}_2 + g_2 + f_{f2} = \tau_2 \tag{3.3}$$

Las Ecuaciones (3.2) y (3.3) representan un sistema de ecuaciones diferenciales con las dinámicas acopladas. Para resolver el modelo es necesario desacoplar tales dinámicas. En las Ecuaciones (3.4) y (3.5) se muestra el modelo dinámico desacoplado.

$$\ddot{q}_1 = \frac{M_{22}V_1 + M_{12}V_2}{detM} \tag{3.4}$$

$$\ddot{q}_2 = \frac{-M_{21}V_1 + M_{11}V_2}{detM} \tag{3.5}$$

Donde

$$V_1 = \tau_1 - C_{11}\dot{q}_1 - C_{12}\dot{q}_2 - g_1 - f_{f1} \tag{3.6}$$

$$V_2 = \tau_2 - C_{21}\dot{q}_1 - C_{22}\dot{q}_2 - g_2 - f_{f2} \tag{3.7}$$

$$detM = M_{11}M_{22} - M_{21}M_{12} \tag{3.8}$$

Una vez desacopladas las dinámicas se procede a discretizar al sistema de ecuaciones (3.4) y (3.5).

$$\ddot{q}_1(t_i) = \frac{M_{22}(t_{i-1})V_1(t_{i-1}) + M_{12}(t_{i-1})V_2(t_{i-1})}{detM(t_i)} \tag{3.9}$$

$$\ddot{q}_2(t_i) = \frac{-M_{21}(t_{i-1})V_1(t_{i-1}) + M_{11}(t_{i-1})V_2(t_{i-1})}{detM(t_{i-1})} \tag{3.10}$$

Donde t_{i-1} indica el tiempo de muestreo actual y t_i el siguiente tiempo de muestreo. Por tanto $q_1(t_{i-1})$ denota el valor numérico que adquiere q_1 en el tiempo de muestreo $i-1$ actual y $\ddot{q}_1(t_i)$ denota el valor numérico que toma $\ddot{q}_1$.en el tiempo de muestreo i para la siguiente iteración.

3.4.1.3.2 Método numérico

El método numérico toma los resultados obtenidos en las Ecuaciones (3.9) y (3.10) que son: $\ddot{q}_1$ y $\ddot{q}_2$, y realiza 2 integraciones para obtener el valor actual $\dot{q}_1$, $\dot{q}_2$, q_1 y q_2. Estos valores se usarán como entradas para el modelo dinámico en la siguiente iteración con los nuevos valores de par leídos. Además serán usados por los módulos de las salidas analógicas, Encoder y comunicación serial.

El método numérico es seleccionable desde la IG en esta aplicación. Se utilizaron:

- Runge-Kutta de orden 4 por su buena aproximación, el cual se muestra en la ecuación (1.58), este método requiere de 4 soluciones en diferentes puntos de las ecuaciones diferenciales. Experimentalmente se encontró que en el DSC requiere de 30 µs para resolverlo.
- El método de Euler, que se muestra en la Ecuación (1.51), y con el que se obtuvo un tiempo de muestreo mínimo de 10 µs.

3.4.1.4 Salidas DAC y escalamiento

Los valores de q_1 y q_2 son usados por el módulo de salidas analógicas del sistema empotrado y las convierte a voltajes proporcionales a la posición del robot de 2 GDL para que pueda ser leída por un controlador externo. En esta aplicación las salidas generan voltajes que representan el valor de las posiciones angulares q_1 y q_2. Para generar un voltaje a partir de un dato se usan los DAC. La posición angular que va desde $[-\pi, \pi]$ se escala para obtener un dato entre $[0,\ 4095]$.

La Ecuación (3.11) escala el valor de las posiciones angulares a valores binarios, donde $\boldsymbol{q}$ es la posición angular en radianes y $\boldsymbol{B_{out}}$ es el dato que se asignará al registro que controla la salida analógica que se explicó en la sección 2.1.4.

$$\boldsymbol{B_{out}} = \frac{4095(\boldsymbol{q} + \pi)}{2\pi} \tag{3.11}$$

Por ejemplo un valor de $\boldsymbol{q} = -\pi$ resulta en $\boldsymbol{B_{out}} = 0$. Un valor de $\boldsymbol{q} = 0$ produce $\boldsymbol{B_{out}} = 2047$ y un valor de $\boldsymbol{q} = \pi$ un valor de $\boldsymbol{B_{out}} = 4095$.

Siguiendo el diagrama de flujo de la Figura 3.7, se define *q=3.14* (el valor de 3.14 se toma por simplicidad) como entrada que representa la posición angular en radianes y *Bout* como salida que representa el dato que se convertirá a voltaje. Se realizan las siguientes instrucciones *Bout=q+3.14=6.28*, *Bout=Bout*4095=25716.6* y *Bout=Bout/6.28=25716.6/6.28=4095*. Teniendo como salida *Bout=4095*.

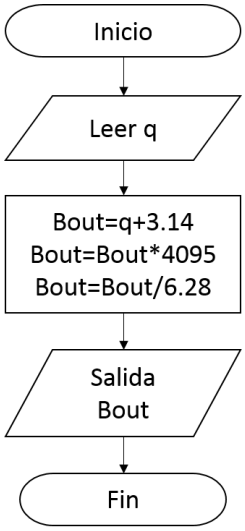


Figura 3.7 Escalamiento

3.4.1.5 Módulo de comunicación serial

El módulo de comunicación serial se encarga implementar el protocolo de comunicaciones que envía y recibe datos por la interfaz USB, lo que permite el intercambio de datos con otros dispositivos. En este caso, se estableció conexión con una computadora personal. Las datos enviados y recibidos por puerto serie del DSC deben tener un formato específico para su correcto procesamiento, dicho formato se explica en las siguientes secciones.

El diagrama de flujo del módulo de comunicación serial se muestra en la Figura 3.8, donde se observa su funcionamiento general. El cual recibe como entrada una cadena *P* y se verifica si es un salto de línea. En caso afirmativo, el módulo envía una cadena con la información de la posición, velocidad y par. En caso contrario, se procesa la entrada para modificar el parámetro físico de acuerdo a la Tabla 3.2. Cuando hay un error en la trama se termina la comunicación.

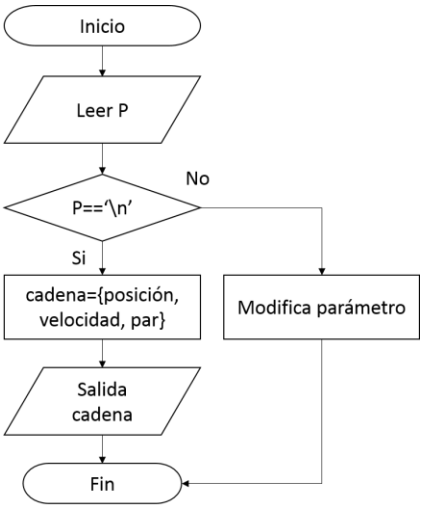


Figura 3.8 Módulo de comunicación

3.4.1.5.1 Recepción de datos

La interfaz gráfica envía comandos al DSC por puerto serie, los cuales son cadenas de caracteres. Sus funciones son: solicitar datos y modificar parámetros del ERM2GDL.

Para modificar los parámetros, el módulo de comunicaciones del sistema empotrado reconoce el siguiente formato para la cadena: *una letra del alfabeto* (entre a-q), *el valor numérico a actualizar* y *un salto de línea* ("\n"), sin espacios. Por ejemplo al enviar la cadena "a0.3\n", se le dice al DSC que modifique el parámetro físico l_1 (longitud del eslabón 1) con un valor de 0.3 metros. En la Tabla 3.2 se muestra la relación entre la letra del alfabeto y el comando a ejecutar.

Para la solicitud de datos debe recibirse un *salto de línea*, el sistema empotrado enviará por puerto serie la posición, velocidad y par de ambas articulaciones.

Letra (minúscula)	**Comando**	**Rango de Valores**
a	Modifica parámetro físico l_1	Real positivo en metros
b	Modifica parámetro físico l_{c1}	Real positivo en metros
c	Modifica parámetro físico l_{c2}	Real positivo en metros
d	Modifica parámetro físico m_1	Real positivo en kilogramos
e	Modifica parámetro físico m_2	Real positivo en kilogramos
f	Modifica parámetro físico I_1	Real positivo en $N \cdot m \cdot seg^2/rad$
g	Modifica parámetro físico I_2	Real positivo en $N \cdot m \cdot seg^2/rad$
h	Modifica parámetro físico b_1	Real positivo en $N \cdot m \cdot seg/rad$
i	Modifica parámetro físico b_2	Real positivo en $N \cdot m \cdot seg/rad$
j	Modifica parámetro físico f_{c1}	Real positivo en $N \cdot m$
k	Modifica parámetro físico f_{c2}	Real positivo en $N \cdot m$
l	Selecciona método numérico a usar	0 para Euler y 1 para Runge-Kutta
m	Modifica modo de trabajo del ERM2GDL	1 par cero, 2 par de PID y 3 par de entradas analógicas
n	Modifica referencia del controlador PID interno $ref1$	Real en rad
o	Modifica referencia del controlador PID interno $ref2$	Real en rad
p	Modifica perturbación $per1$	Real en $N \cdot m$
q	Modifica perturbación $per2$	Real en $N \cdot m$
Salto de línea	Petición de datos	

Tabla 3.2 Relación de comandos

3.4.1.5.2 Envío de datos

La salida de datos por puerto serie hacia la interfaz gráfica, sucede una vez que se recibe una petición de datos. El ERM2GDL envía la información del último estado del robot de 2 GDL (posición y velocidad de ambas articulaciones) y el par de entrada en el momento que se hizo la solicitud.

Antes de enviar los datos por puerto serie, los valores de las velocidades y posiciones son convertidos a $[grados/seg]$, $[grados]$ y $[N \cdot m]$. Los datos son enviados en el siguiente orden q_1, q_2, $\dot{q}_1$, $\dot{q}_2$, τ_1 y τ_2.

El formato de envío es: el valor a enviar más un ";". Por ejemplo se envían los datos de la siguiente forma "179.99;40.50;30.83;60.72;40.51;10.40;" sin espacios. La interfaz gráfica detecta a ";" como la terminación de un dato, por tanto habrá recibido 6 datos, note el orden en el que se envían los datos.

3.4.2 Interfaz Gráfica

Para tener una mejor representación del funcionamiento del ERM2GDL se desarrolló una interfaz gráfica usando LabVIEW 2013 ®. Dicha interfaz se compone de 2 bloques conceptuales que se muestran en la Figura 3.9. El bloque de procesamiento envía y recibe datos por puerto serie con el sistema empotrado. El bloque de presentación, se encarga de mostrar la información del estado del robot por medio de gráficas y el movimiento del mismo en un ambiente virtual.

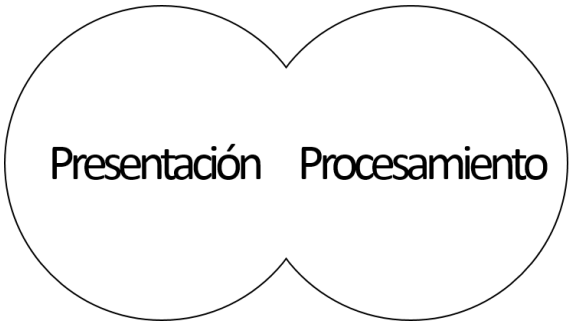


Figura 3.9 Esquema conceptual de la interfaz gráfica

En esta sección se explican los módulos del software (Figura 3.10) que intervienen en el funcionamiento de la interfaz, donde cada uno se describe a detalle en la Sección 3.4.2.1. Los cuales fueron diseñados a partir de los requerimientos específicos.

Las acciones que realiza la IG inician con la ejecución de **configuración de puerto serie** y el **lector de modelos CAD** que toma el gráfico del robot, ambos módulos se ejecutan una sola vez. El **generador de comandos** elabora la cadena de comandos que se envían cuando se requiere modificar un parámetro. El módulo de **lectura y escritura** envía y recibe las tramas de y hacia el sistema empotrado. El módulo de **extracción de datos** se encarga de recibir y procesar los datos numéricos. El **generador de escenas** muestra el movimiento en el ambiente virtual en función de la información recibida y de los modelos CAD leídos. Se repite el proceso con el **generador de comandos** para realizar una nueva petición de datos o para cambiar parámetros.

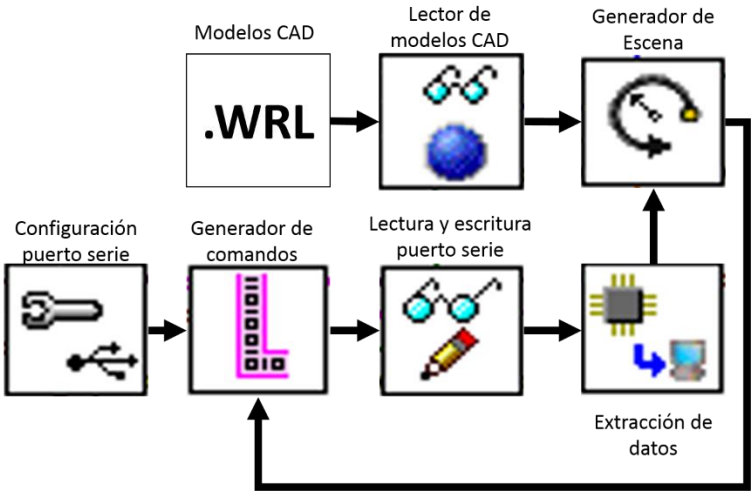


Figura 3.10 Diagrama de bloques de la interfaz gráfica

3.4.2.1 Módulos de procesamiento

Los módulos de procesamiento manipulan las entradas y generan salidas, en este caso las entradas son los datos recibidos por puerto serie y los controles de la interfaz. Las salidas son los datos enviados por puerto serie y los datos de las gráficas de posición, velocidad y par.

3.4.2.1.1 Comunicación serial

Para la comunicación por puerto serie, se usó el software NI-VISA que es una implementación de la firma National Instruments del estándar VISA (Virtual Instruments Software Architecture), el cual es una librería desarrollada por varios fabricantes de equipos que proporciona un estándar en cuanto a software para las operaciones de escritura y lectura en instrumentación [74]. Con VISA se puede establecer comunicaciones a través de GPIB, puerto serie, PXI, VXI o Ethernet. En esta aplicación se utiliza el puerto serie.

En la Figura 3.11 se muestra el módulo para la configuración del puerto serie, el cual se ejecuta una vez. Como resultado, se establece comunicación por el puerto COM especificado, pudiendo enviar o recibir datos. Sus salidas consisten de una bandera de error llamada **error out** y **VISA Refnum out** que contiene la información necesaria del puerto COM. Sus entradas son las siguientes:

- **Termination Character** (Carácter de terminación), es un carácter que se usa como bandera de terminación en el proceso de lectura y escritura, en esta aplicación se definió como un salto de línea.
- **VISA Refnum in** (Número de referencia de entrada VISA), en este caso es la información del puerto COM a usar.
- **End Read on Termination Character** (Finalizar lectura con el carácter de terminación), se usa para activar la terminación de una lectura cuando, en la cadena de caracteres a enviar, se presente el **Termination Character** seleccionado.
- **End Write on Termination Character** (Finalizar escritura con el carácter de terminación), se usa para la terminación de una escritura cuando, en la cadena de caracteres a enviar, se presente el **Termination Character** seleccionado.

- **Serial Settings** (Parámetros del puerto serie), se usa para especificar la velocidad de transmisión, el tamaño de los datos, la paridad, el bit de paro y el control de flujo. En esta aplicación se seleccionó 57600 baudios, 8 bits, sin paridad, 1 bit de paro y sin control de flujo.
- **XON/XOFF Characters** (Caracteres XON/XOFF), se usan cuando se activa el control de flujo y sirve para iniciar/detener la transmisión, el protocolo serial los define como caracteres con el valor de XOFF=19 y XON=17.

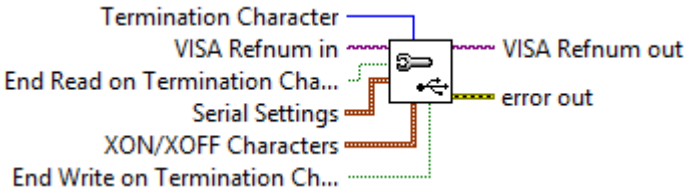


Figura 3.11 Configuración del puerto serie

Una vez configurada la comunicación por puerto serie, se usa el módulo de la Figura 3.12 el cual permite la lectura y escritura de datos por puerto serie, usando bloques funcionales. Como salida se tiene una bandera de error llamada **error out**, **VISA Refnum out** que contiene la información del puerto COM y **read buffer** que es la cadena leída y almacenada en el buffer de entrada. Las entradas son:

- **Read** es una bandera que indica el inicio del proceso de lectura del buffer de entrada.
- **VISA Refnum in** o Número de referencia de entrada VISA en este caso es el numero generado por el módulo de configuración de puerto serie.
- **Write** es una bandera que indica el inicio del proceso de escritura en el buffer de salida.
- **read buffer** es una cadena de caracteres e indica el dato leído desde el buffer de entrada.
- **Command** es una cadena de caracteres e indica el dato a enviar al buffer de salida.
- **error in** o error de entrada, es un cluster en el que lleva la información de errores ocurridos en pasos anteriores.

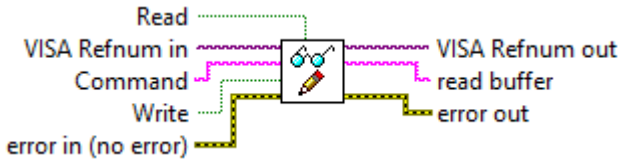


Figura 3.12 Lectura y escritura por puerto serie

3.4.2.1.2 Manejo de datos

Los datos recibidos o enviados por puerto serie son cadenas de caracteres. Los datos son procesados por el módulo que se muestra en la Figura 3.13. Como salida se obtiene un clúster **Cluster out**, el cual está formado por la posición, velocidad y par de ambas articulaciones. Como entrada se tiene una cadena de caracteres llamada **String**.

Figura 3.13 Extracción de datos

En la Tabla 3.3 se muestra un ejemplo de la cadena de caracteres (sin espacios) recibida del DSC, el módulo de extracción de datos toma los valores en el orden mostrado, donde el carácter de ";" indica la terminación de un dato.

Orden	q_1	q_2	$\dot{q}_1$	$\dot{q}_2$	τ_1	τ_2
Cadena	0.1;	0.423;	0.75;	0.23;	12.282;	15.3;

Tabla 3.3 Orden de los datos

Los comandos a enviar se generan en el módulo de la Figura 3.14, donde a partir de los botones que se presionan en la interfaz se obtienen los clústeres **input cluster** e **input cluster 1**, que se usan en para formar la cadena de caracteres de salida **String** que se enviará al DSC por puerto serie.

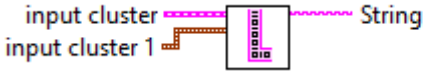


Figura 3.14 Generador de comandos

En la Tabla 3.4 se muestra un ejemplo de cadena de carácter que se envía hacia el DSC. En la sección 3.4.1.5 se muestra el significado de la cadena.

Cadena	b0.3\n

Tabla 3.4 Cadena de caracteres de entrada

3.4.2.1.3 Manejo de ambiente virtual

El módulo que se muestra en la Figura 3.15 se usa para leer los dibujos CAD y procesarlos para ser mostrados en la interfaz usando un ambiente virtual, este módulo se ejecuta una vez, como resultado se genera una escena inicial, es decir una vista en 3D del robot manipulador. Las entradas son:

- **SceneGraphDisplay in** contiene el nombre del directorio raíz donde se encuentran los archivos importantes para el correcto funcionamiento de la IG.
- **name or relative path** contiene el nombre de la carpeta donde se encuentran los modelos CAD.
- **name or relative path 2** contiene el nombre del modelo CAD del primer eslabón.
- **name or relative path 3** contiene el nombre del modelo CAD del segundo eslabón.

Como salida se tiene una escena con el nombre de **Scene Object out**, un número llamado **Light Number** que indica el tipo de iluminación usada en la escena y una bandera de error llamada **error out** que indica si hubo algún error a leer los modelos CAD.

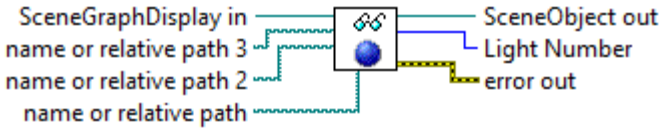


Figura 3.15 Lectura del dibujo CAD en formato WRL

En la Figura 3.16 se muestra el módulo que se usa para generar una nueva escena a partir de la escena inicial producida por el módulo de la Figura 3.15. Como salida se tiene la escena **Scene Object out** y una bandera que indica si hubo algún error llamada **error out**. Las entradas son:

- **Light State** indica el tipo de iluminación usada en la escena anterior.
- **Light Number** indica el tipo de iluminación a usar en la nueva escena.
- **angle** indica el valor de la posición angular de la articulación 1.
- **angle 2** indica el valor de la posición angular de la articulación 2.
- **Scene Object in** es una escena que se modifica de acuerdo a **angle** y **angle 2**.
- **error in** es una bandera que indica si hubo algún error en los pasos anteriores.

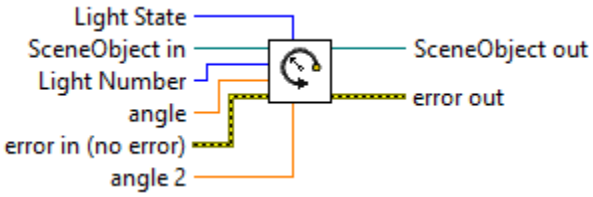


Figura 3.16 Generador de escenas

3.4.2.2 Módulo de presentación

El módulo de presentación se encarga de mostrar los datos procesados, por lo que la interfaz de usuario consiste de un ambiente virtual donde se muestra los movimientos del Robot de 2 GDL, gráficas de posición, velocidad, par de entrada y control de configuración de comunicación por puerto serie. La interfaz se divide 5 pestañas las cuales son:

- **Interfaz Gráfica** es la pestaña por defecto y se muestra en la Figura 3.17.
- **Posición** en esta pestaña se muestran las gráficas de posición angular de ambas articulaciones.
- **Velocidad** se muestran las gráficas de velocidad angular de ambas articulaciones.
- **Par de entrada** se muestran las gráficas del par aplicado a ambas articulaciones.
- **Comunicación serial** se muestran los controles para modificar los parámetros de comunicación.

En la pestaña por defecto se detiene la interfaz gráfica con el **botón de paro** y se modifican los parámetros del robot con el **Panel de parámetros físicos**. Se muestra el **Ambiente virtual**, diferentes **Pestañas**, un **Led indicador** para mostrar la ejecución de la interfaz. Para mayor detalle en el uso de la interfaz revisar el Apéndice A.

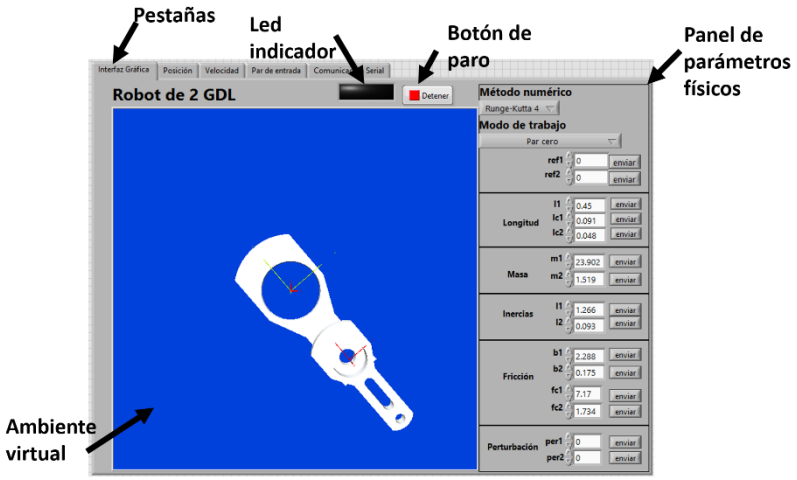


Figura 3.17 Interfaz gráfica (Pestaña por defecto)

En las pestañas de **Posición**, **Velocidad** y **Par de entrada** se tienen los mismos elementos en su presentación que son: **Gráficas**, **Indicadores** y **Unidad de medida**

La pestaña de **Posición** se muestra en la Figura 3.18 donde se ven las **Gráficas** de la posición angular de ambas articulaciones. Los datos se representan en grados desde -180° a 180°. La **Unidad de medida** es en grados. Contiene **Indicadores** para representar el último dato graficado.

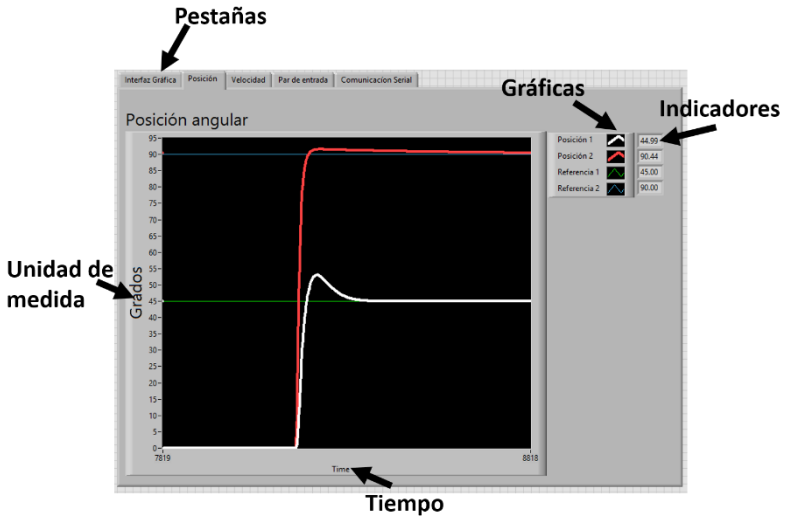


Figura 3.18 Posición angular

La pestaña de **Comunicación serial** se muestra en la Figura 3.19, donde se ven los controles para modificar los parámetros necesarios, en la configuración del puerto serie. En esta aplicación, se asignó el **puerto COM** número 7 al F28377S. El sistema empotrado se configuró para una velocidad de 57600 baudios, de 8 bits de datos, sin paridad y con 1 bit de paro. El **Carácter de terminación** de recepción o envío se estableció en un salto de línea por lo que la escritura y lectura terminan cuando se presenta un salto de línea. El **Buffer** almacena los datos recibidos, que se muestran en un indicador.

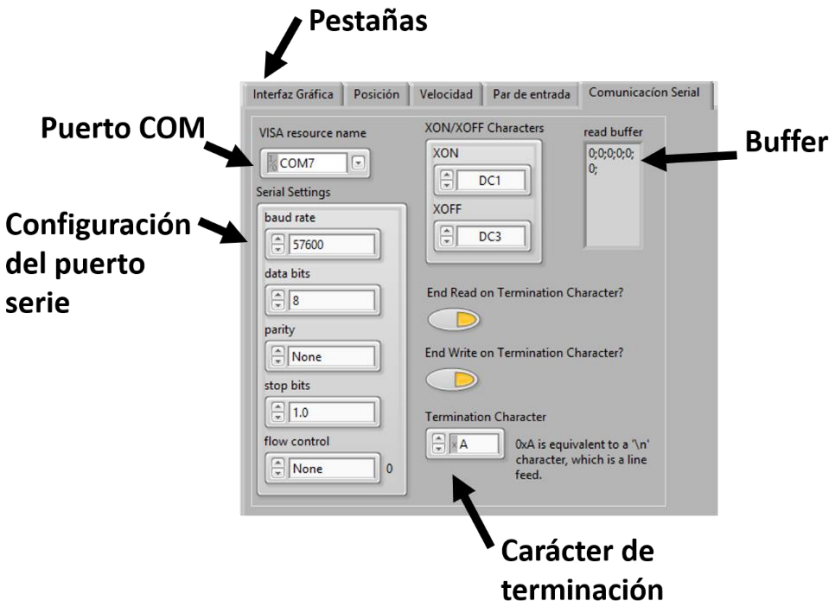


Figura 3.19 Configuración del puerto serie

Capítulo 4. Pruebas y resultados

En este capítulo se muestran las pruebas realizadas al DSC para verificar su velocidad de respuesta en las entradas y salidas, así como los tiempos de ejecución de las operaciones aritméticas que son la base para resolver el modelo dinámico del robot en tiempo real. También se muestra la respuesta del ERM2GDL en lazo abierto y cerrado con los siguientes controladores PID, PD con compensación de gravedad, retroalimentación por todos los estados y modos deslizantes.

4.1 Pruebas con el DSC F28377S

Para conocer las capacidades del DSC F28377S en la solución del modelo dinámico. Es necesario saber con certeza, el tiempo que le lleva ejecutar operaciones aritméticas con datos enteros y flotantes. Esto con el fin de que se establezca a priorí el número de operaciones que puede realizar, entre los diferentes tiempos de muestreo en aplicaciones de tiempo real.

Las pruebas que se hicieron al respecto son: escritura digital, escritura y lectura analógica, caracterización de las entradas y salidas analógicas, el número de ciclos de reloj le lleva ejecutar las operaciones aritméticas. A continuación se explica cómo se hizo cada una de ellas y los resultados obtenidos.

Para la medición de los tiempos de respuesta se usa Code Composer Studio (CCS), el cual tiene un modo de operación remota, donde se puede ejecutar instrucción por instrucción, además de que se puede contabilizar el número de ciclos de reloj que tarda en ejecutar cada instrucción.

4.1.1 Prueba escritura digital

El objetivo de esta prueba es conocer la frecuencia máxima con la que cambia el estado de las salidas digitales en el DSC. Para tener una medición del tiempo se hizo un programa que cambia de estado la salida digital en un ciclo infinito. En la salida digital se conecta un osciloscopio (las mediciones se hacen con respecto a tierra) y con él se mide el tiempo de respuesta.

Experimentalmente se encontró, que un cambio de estado tarda 5 ciclos de reloj. En el osciloscopio se registró una frecuencia máxima de 1.0869 MHz la cual oscilaba alrededor de 1 MHz. El resultado no coincidió con lo esperado debido a que 5 ciclos de reloj, a 200 MHz, equivalen a 40 MHz. La diferencia está en el tiempo que tarda en ejecutar la parte que controla el ciclo.

4.1.2 Prueba de lectura y escritura analógica

En esta prueba se obtiene el tiempo que toma el proceso de digitalizar una entrada analógica y enviarla hacia el convertidor digital analógico, sin mayor procesamiento. El objetivo es cuantificar el tiempo que tarda el DSC en hacer esta operación, que es básica en aplicaciones de tiempo real ya que determina la frecuencia máxima de trabajo. Se hicieron dos pruebas; una usando una entrada y una salida y la otra usando dos entradas y dos salidas. La medición de los ciclos se hizo en el CCS contando los ciclos de reloj.

La prueba se basa en el fenómeno de aliasing [62], que se produce cuando la frecuencia de muestreo es menor a la mitad de la frecuencia de una señal periódica. Esto produce una señal capturada con frecuencia menor a la original.

Se usa un generador de funciones para crear una señal periódica. Dicha señal se captura usando el ADC en modo simple activado por interrupción como se explicó en la sección 2.1.3. El DAC genera el voltaje de la señal leída y su frecuencia se obtiene usando un osciloscopio.

El experimento consiste en incrementar gradualmente la frecuencia de una señal periódica de entrada mientras se verifica que la señal de salida tenga la misma frecuencia. En caso contrario se ha producido el fenómeno de aliasing, en el cual las frecuencias de la señal de entrada y de salida son diferentes.

4.1.2.1 Una entrada analógica y una salida analógica

Para realizar esta prueba se usó la entrada ADC 1 y la salida DAC 1, como se muestra en la Figura 4.1, donde V_{in} es una señal sinusoidal con voltaje máximo de 3V y mínimo de 0V. El valor binario leído con el ADC es B, el cual se asigna al DAC para convertirlo en una señal de voltaje V_{out} con el mismo rango de voltaje.

En esta prueba las instrucciones se ejecutaron en 15 ciclos de reloj medidos con CCS. A una frecuencia de 200 MHz, da una frecuencia de operación teórica de 13 MHz. Experimentalmente se encontró una frecuencia máxima de operación de 0.9615 MHz. La diferencia se da por la parte que controla las interrupciones.

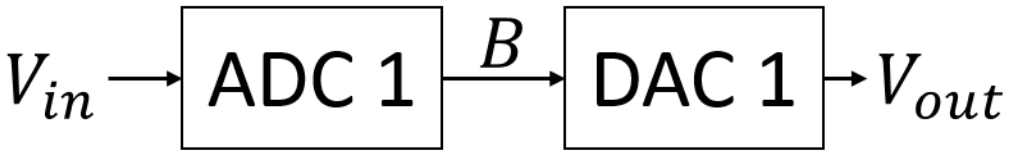


Figura 4.1 Una entrada y una salida

4.1.2.2 Dos entradas analógicas y dos salidas analógicas

Para realizar esta prueba se usaron dos entradas ADC 1 y ADC 2, y dos salidas DAC 1 y el DAC 2, como se muestra en la Figura 4.2, donde V_{in1} y V_{in2} son señales de tipo sinusoidal con voltaje máximo de 3V y mínimo de 0V. El valor binario leído con el ADC 1 y 2 son B_1 y B_2, los cuales se asignan a los DAC 1 y 2 para convertirlos en señales de voltaje V_{out1} y V_{out2} con el mismo rango de voltaje.

En esta prueba el DSC ejecutó el proceso en 30 ciclos de reloj a 200 MHz, lo que teóricamente da una frecuencia de operación de 6 MHz. Experimentalmente se encontró una frecuencia máxima de operación de 0.8849 MHz. La diferencia se da por la parte que controla las interrupciones.

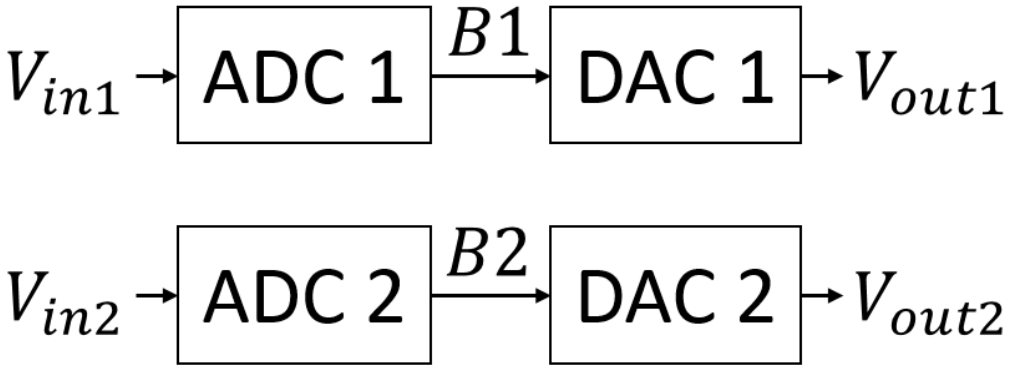


Figura 4.2 Dos entradas y dos salidas

4.1.3 Caracterización de las entradas y salidas analógicas

Para tener una correcta interpretación de los voltajes de entrada y de salida se necesita caracterizar los voltajes de entrada y el código que genera, así como a partir del código, que voltaje se tiene a la salida. Idealmente la relación que existe es lineal y se basa en la resolución de los convertidores, sin embargo en la práctica es necesario caracterizarlo para tener la función de trasferencia de las entradas y las salidas. La ecuación que modela el comportamiento es lineal por lo cual se requiere obtener la pendiente y la ordenada en función de las mediciones haciendo un ajuste por mínimos cuadrados.

4.1.3.1 Caracterización entradas analógicas

Para esta prueba se conecta a cada una de las entradas del ADC una fuente de voltaje, con un multímetro se mide el voltaje que se tiene a la entrada del convertidor y desde el CCS se obtiene el valor binario que se tiene para ese voltaje. Se hicieron mediciones en el rango de 0 a 3V. Posteriormente mediante aproximación por mínimos cuadrados se obtuvieron dos ecuaciones que relacionan el valor digital obtenido con el voltaje real en la entrada. La Ecuación (4.1) muestra el comportamiento de la entrada analógica 1 y la Ecuación (4.2) el comportamiento de la entrada analógica 2. Donde B indica el valor binario obtenido del ADC y V_{in} el voltaje presente en la entrada analógica.

$$B_1 = 1364.2208V_{in1} - 4.235 \tag{4.1}$$

$$B_2 = 1361.8758V_{in2} - 8.843 \tag{4.2}$$

Para ejemplificar, se toma en cuenta un voltaje de $V_{in} = 1.5V$ en ambas entradas, para la entrada analógica 1, de la Ecuación (4.1) tenemos $B_1 = 1364.2208(1.5) - 4.235 = 2042.0962$, experimentalmente se registró un valor de 2043 y para la entrada analógica 2, de la Ecuación (4.2) tenemos $B_2 = 1361.8758(1.5) - 8.843 = 2033.9707$, experimentalmente se registró un valor de 2035.

4.1.3.2 Caracterización salidas analógicas

Para la caracterización de las salidas analógicas se varía el dato asignado al convertidor y se mide el voltaje que genera. Se hicieron 17 mediciones en el rango de 0 a 4095 y con un multímetro se mide el voltaje que se tiene a la salida, el cual varía en el rango de 0 a 3V, con mínimos cuadrados se obtiene los parámetros de la función de transferencia. La Ecuación (4.3) modela el comportamiento de la salida analógica 1 y la Ecuación (4.4) de la salida analógica 2. Donde V_{out} indica el voltaje en volts en la salida y B el valor digital que procesará el DAC.

$$V_{out1} = \frac{1}{1363.36} B_1 - \frac{1}{186.969} \tag{4.3}$$

$$V_{out2} = \frac{1}{1364.95} B_2 + \frac{1}{103.33} \tag{4.4}$$

Para ejemplificar, en el caso de la salida analógica 1, se supone un valor de $B_1 = 2052$. De la Ecuación (4.3) se tiene $V_{out1} = \frac{1}{1363.36}(2052) - \frac{1}{186.969} = 1.49975V$, experimentalmente se registró un valor de $1.5V$ y para la entrada analógica 2, se supone un valor de $B_2 = 2035$. De la Ecuación (4.4) se tiene $V_{out2} = \frac{1}{1364.95}(2035) + \frac{1}{103.33} = 1.50057V$, experimentalmente se registró un valor de $1.5V$.

4.1.4 Tiempo de ejecución de las operaciones aritméticas

El sistema a emular utiliza operaciones de punto flotante para resolver en tiempo real el modelo dinámico del robot de 2 grados de libertad. El tiempo que le lleva realizarlo depende del tiempo de ejecución de las sumas, restas, multiplicaciones y divisiones. Por lo cual se hicieron pruebas de cuánto tiempo le lleva a cada una de estas operaciones. La forma más rápida de ejecutar es mediante las unidades especializadas en operaciones de punto flotante y funciones trigonométricas. Se hizo un programa en el que se ejecutaban instrucciones de 8 tipos y se contabilizaba el número de ciclos de reloj que tomaba cada una.

En la Tabla 4.1 muestra los resultados obtenidos, donde una operación sucesiva indica que el resultado depende de un valor obtenido en una operación anterior. Una operación simple indica que el resultado no depende de un valor anterior. Las operaciones sucesivas requieren de más ciclos de reloj. La instrucción seno se ejecuta en 8 ciclos de reloj usando la unidad trigonométrica y en 32 ciclos de reloj usando la librería math.h.

Operación	Ciclos de reloj
División sucesiva	12
Multiplicación sucesiva	9
Suma sucesiva	9
División simple	5
Multiplicación simple	5
Suma simple	5
Seno unidad trigonométrica	8
Seno librería math.h	32

Tabla 4.1 Ciclos de reloj para una operación matemática

Con un tiempo de muestreo de $100\mu s$ a una frecuencia de 200 MHz, se tienen 20,000 ciclos de reloj. Que pueden ejecutar 4000 divisiones de punto flotante.

4.2 Comparativa en lazo abierto Simulink-ERM2GDL

Para verificar el correcto funcionamiento de la ejecución del modelo en el DSC, se compara la respuesta entre lo que emula el sistema ERM2GDL y una simulación estandarizada en Matlab Simulink. Para esto, se elaboró en Simulink el modelo del robot que se muestra en la Figura 4.3. Donde el bloque llamado "MATLAB Function" calcula la aceleración en ambas articulaciones a partir del par de entrada, posición y velocidad angular del robot. Dicha aceleración es integrada para obtener la posición y velocidad angular para el siguiente paso de simulación.

Los parámetros del robot son los que se muestran en la Tabla 1.1, los cuales se usan en el modelo del sistema ERM2GDL y en el modelo de Simulink.

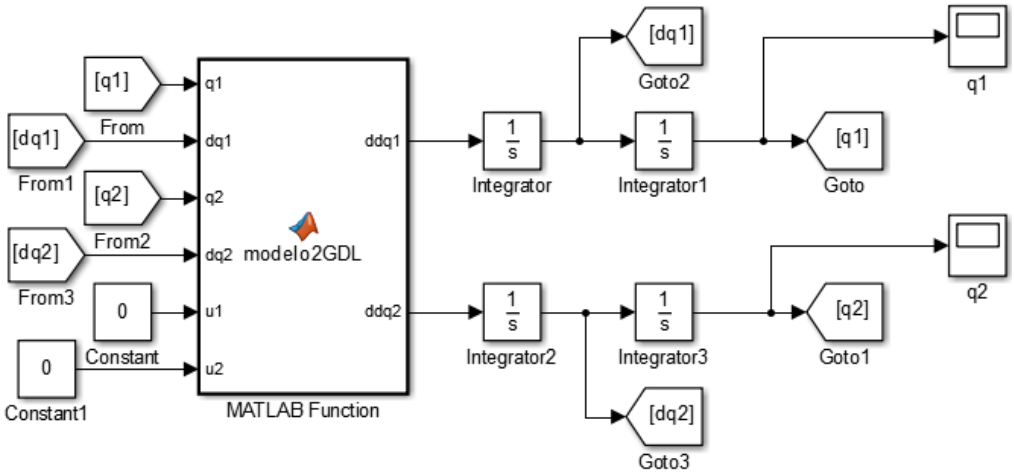


Figura 4.3 Diagrama de bloques en Simulink del modelo del Robot de 2 GDL

Las pruebas se hacen en lazo abierto, es decir sin entrada de control ya que el modelo de Simulink no responde en tiempo real, que es el caso del ERM2GDL, para que se tenga movimiento se inicia con condiciones iniciales en ambas articulaciones. Se realizaron 4 pruebas diferentes que son:

- Modelo sin fricción.
- Modelo con fricción viscosa.
- Modelo con fricción de Coulomb.
- Modelo con fricción de Coulomb y con fricción viscosa.

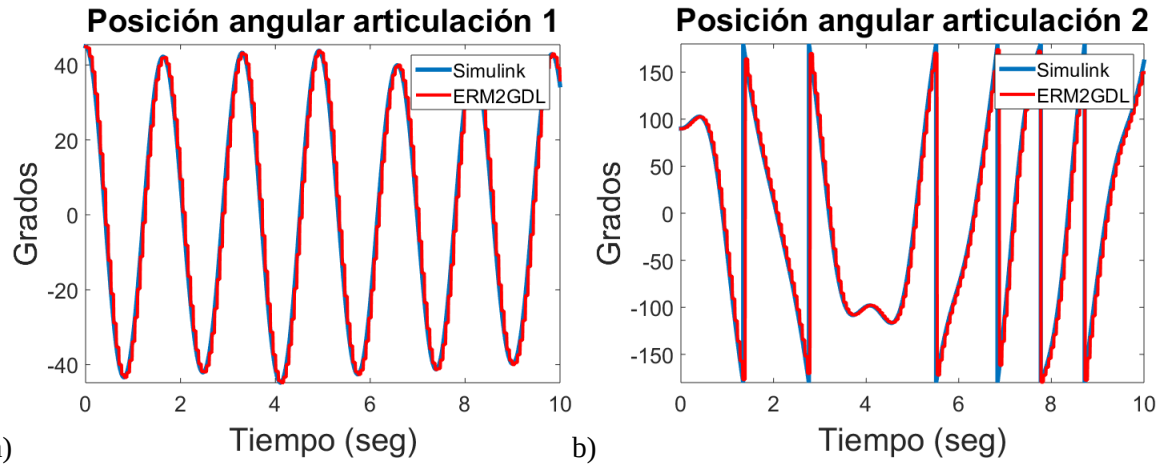
Las condiciones de simulación del ERM2GDL y de Simulink son: Tiempo de simulación de 10 segundos con periodo de muestreo de 100 µs; condiciones iniciales de cada articulación $q_1 = 45°$ y $q_2 = 90°$, velocidad inicial cero y par de entrada cero.

Los datos en Simulink se exportan a Matlab y los datos del ERM2GDL se guardan usando la interfaz gráfica, como lo muestra la sección 5.2.1 del Apéndice A. Se graficó la respuesta y el error

de posición angular en la primer y segunda articulación, comparando los resultados obtenidos de Simulink y el ERM2GDL.

4.2.1 Modelo sin fricción

En esta prueba se verifica la respuesta del sistema sin fricción de Coulomb ni viscosa. En la Figura 4.4 a) y b) se muestra la posición angular obtenida con Simulink y el ERM2GDL para ambas articulaciones. Esta información se usa para calcular el error. Además se observa que se produce un movimiento periódico y perpetuo debido a que no existe disipación de energía.



a) b)

Figura 4.4 Respuesta de la posición angular

El error se obtiene a partir del valor absoluto de la diferencia aritmética entre ambas respuestas. En la Figura 4.5 se muestra el error de la posición angular entre Simulink y ERM2GDL para la primera y segunda articulación. Donde el error máximo, mínimo y promedio se muestra en la Tabla 4.2. Los picos de error que se observan en la gráfica se presentan cuando la velocidad angular en las articulaciones es elevada, dichos picos de error son relativamente pequeños debido a que no superan los 2.2 grados de amplitud. Cabe mencionar que el error es mayor en la segunda articulación debido a que tiene menor masa inercial.

Error	Máximo [grados]	Mínimo [Grados]	Promedio [grados]
Articulación 1	0.3334	0.0037	0.0446
Articulación 2	2.1513	0.0022	0.3763

Tabla 4.2 Error máximo, mínimo y promedio

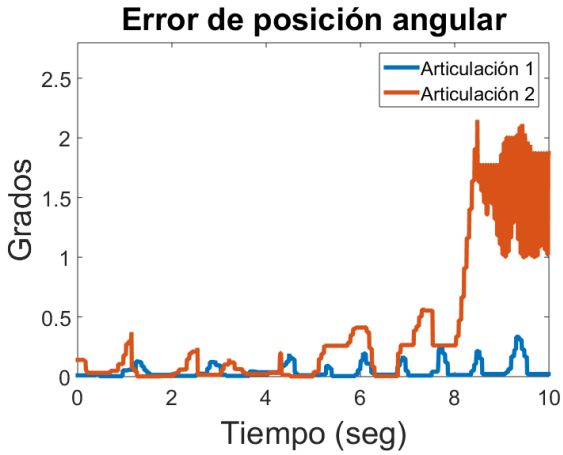


Figura 4.5 Error de posición angular

4.2.2 Modelo con fricción viscosa

Esta prueba compara la respuesta del modelo del robot únicamente con fricción viscosa. En la Figura 4.6 a) y b) se muestra la posición angular con Simulink y el ERM2GDL para ambas articulaciones. Esta información se usa para calcular el error. Ambas articulaciones oscilan hasta establecerse en 0 grados debido a la disipación de la energía por la fricción viscosa.

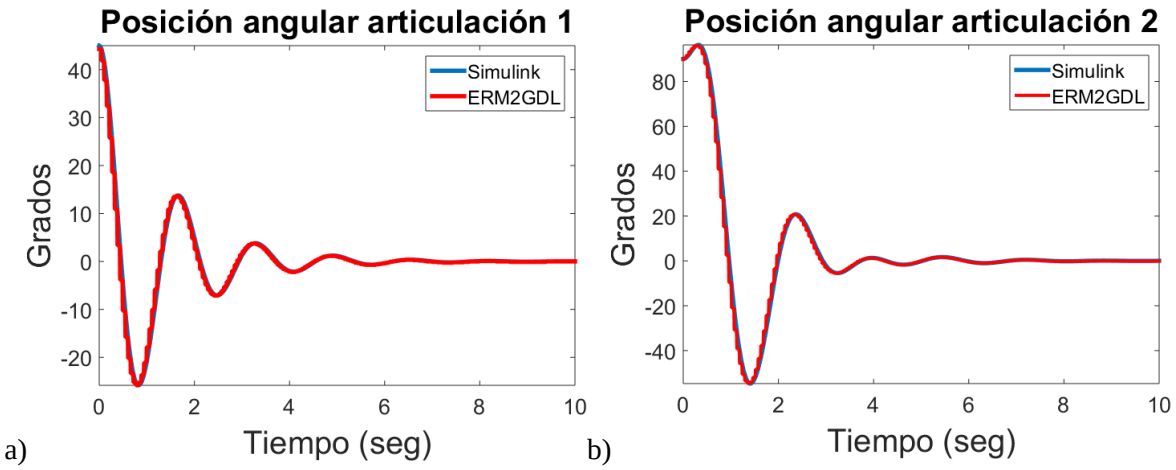


Figura 4.6 Respuesta de la posición angular

En la Figura 4.7 se muestra el error de la posición angular entre Simulink y ERM2GDL para la primera y segunda articulación. Donde el error máximo, mínimo y promedio se muestra en la Tabla 4.3. Los picos de error que se muestran en la gráfica se presentan cuando la velocidad angular en las articulaciones es elevada. En este caso el pico de error es mayor en la segunda articulación debido a su menor masa inercial. Los picos de error son pequeños menores a 0.5 grados.

Error	Máximo [grados]	Mínimo [Grados]	Promedio [grados]
Articulación 1	0.2471	3.73e-5	0.0177
Articulación 2	0.4718	8.5e-5	0.0308

Tabla 4.3 Error máximo, mínimo y promedio

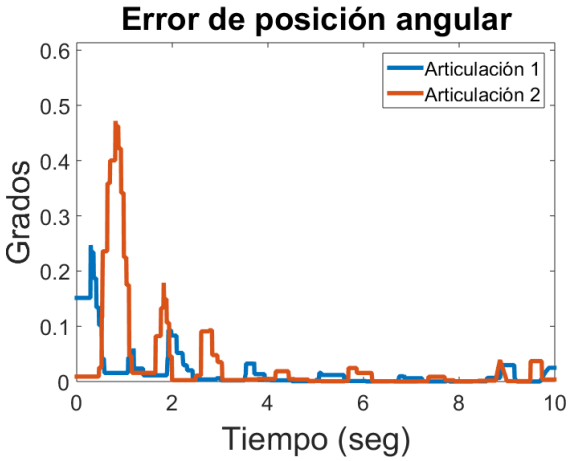


Figura 4.7 Error de posición angular

4.2.3 Modelo con fricción de Coulomb

Esta prueba compara la respuesta del modelo del robot únicamente con fricción de Coulomb. En la Figura 4.8 a) y b) se muestra la posición angular con Simulink y el ERM2GDL para ambas articulaciones. Esta información se usa para calcular el error. En la articulación 2 se observa un desplazamiento máximo de 0.3 grados debido a que la fuerza de fricción de Coulomb es mayor al par gravitatorio en dicha articulación.

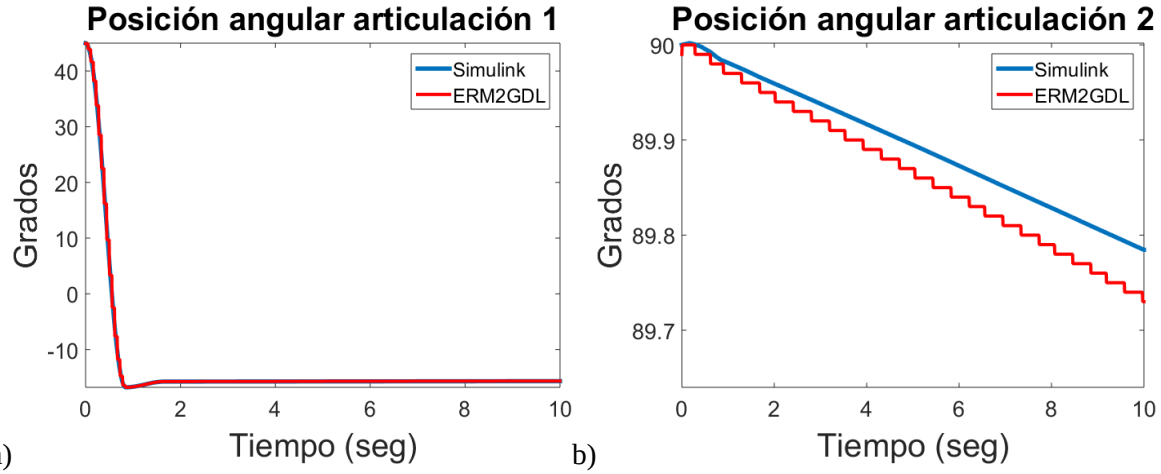


Figura 4.8 Respuesta de la posición angular

En la Figura 4.9 se muestra el error de la posición angular entre Simulink y ERM2GDL para la primera y segunda articulación. Donde el error máximo, mínimo y promedio se muestra en la Tabla 4.4. En este caso se observa una gráfica de error escalonado el cual se produce por la diferencia en la precisión numérica que manejan ambos sistemas. Los picos de error se presentan cuando la velocidad

angular en las articulaciones es elevada. En esta prueba se tuvo un mayor error en la primera articulación debido a que la fricción de Coulomb evita que la segunda articulación tome valores de velocidad altos. Los errores de posición angular son relativamente pequeños debido a que no superan los 0.3 grados.

Error	Máximo [grados]	Mínimo [Grados]	Promedio [grados]
Articulación 1	0.2454	4.2e-4	0.0205
Articulación 2	0.0453	8e-5	0.0224

Tabla 4.4 Error máximo, mínimo y promedio

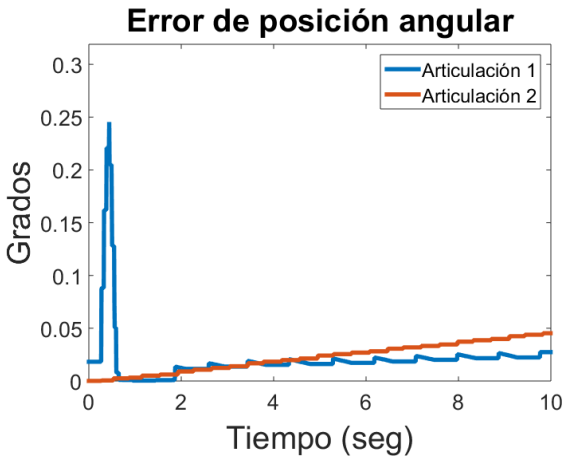


Figura 4.9 Error de posición angular

4.2.4 Modelo con fricción de Coulomb y fricción viscosa

Esta prueba compara la respuesta del modelo del robot con fricción de Coulomb y viscosa. En la Figura 4.10 a) y b) se muestra la posición angular con Simulink y el ERM2GDL para ambas articulaciones. Esta información se usa para calcular el error. La segunda articulación se desplaza 0.3 grados debido a que el par de fricción de Coulomb es mayor al par gravitatorio en dicha articulación.

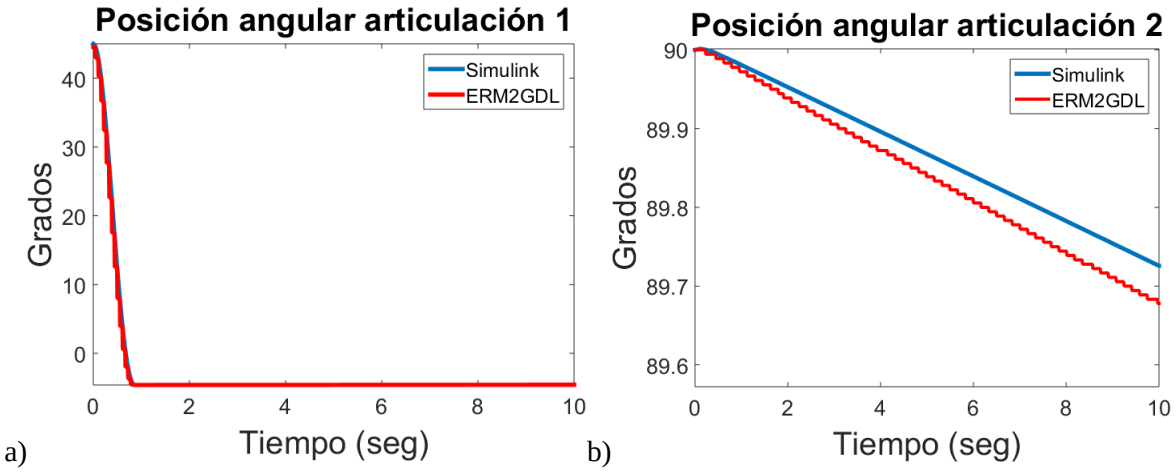


Figura 4.10 Respuesta de la posición angular

En la Figura 4.11 se muestra el error de la posición angular entre Simulink y ERM2GDL para la primera y segunda articulación. Donde el error máximo, mínimo y promedio se muestra en la Tabla 4.5. Los picos de error que se muestran en la gráfica se presentan cuando la velocidad angular en las articulaciones es elevada. Los errores de posición angular son pequeños debido a que no superan los 0.045 grados.

Error	Máximo [grados]	Mínimo [Grados]	Promedio [grados]
Articulación 1	0.0432	1.32e-6	0.002
Articulación 2	0.0434	0	0.023

Tabla 4.5 Error máximo, mínimo y promedio

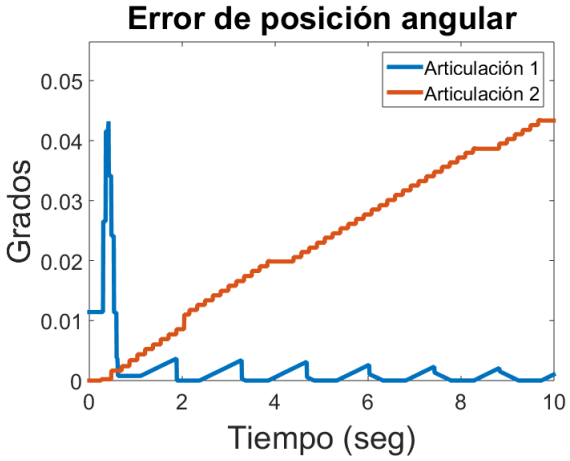


Figura 4.11 Error de posición angular

4.3 Simulador HIL del robot manipulador de 2 GDL

Un simulador HIL consta de una planta y un controlador, donde al menos uno es emulado, esto con el fin de facilitar la configuración de parámetros y acelerar el proceso de diseño de sistemas de control. En este caso el ERM2GDL emula a la planta y el diseño se centra en el controlador, a esto se llama prototipado rápido de control.

La diferencia entre un simulador HIL y una simulación numérica pura, es que el simulador HIL maneja las conexiones eléctricas como en la aplicación real. Es decir, el controlador se conecta eléctricamente a la planta emulada y una vez diseñado el controlador, el paso a la implementación real, no requiere de cambios adicionales en cuanto a conexiones eléctricas.

4.3.1 Simulador HIL

El ERM2GDL se conectó eléctricamente a un controlador con base en una computadora personal y una tarjeta de adquisición de datos. Con lo cual se conforma un sistema de simulación HIL, el cual se muestra en la Figura 4.12 donde se observan sus componentes. Para comprobar el funcionamiento del simulador HIL se implementó una serie de controladores de posición programados en Simulink. Dichos controladores son PID, PD con compensación de gravedad, retroalimentación de todos los estados y modos deslizantes.

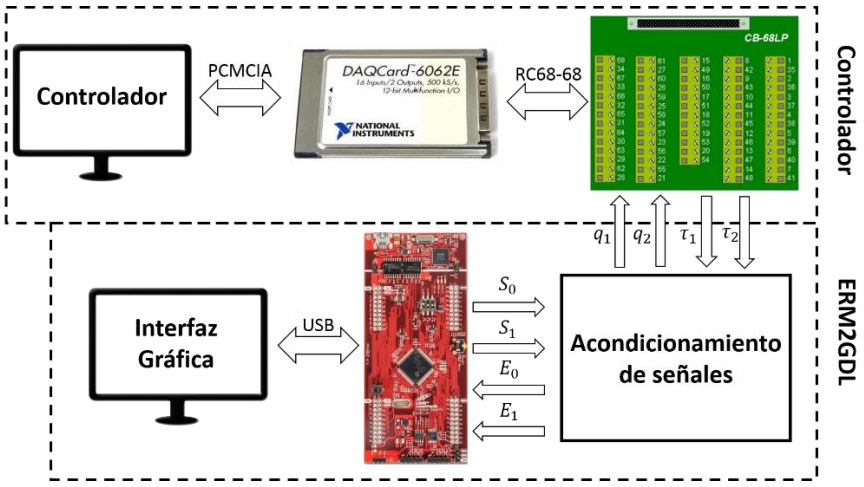


Figura 4.12 Diagrama de bloques del simulador HIL

En las siguientes subsecciones se explica a detalle las conexiones del controlador y del ERM2GDL.

4.3.2 Conexiones del controlador

El controlador está formado por una computadora que ejecuta en tiempo real la ley de control. Las entradas y salidas hacia la planta se generan con la tarjeta de adquisición de datos DAQ 6062E, la cual se conecta a la computadora personal en el puerto PCMCIA. Las señales se conectan al block de conexiones CB-68LP y este a su vez a la tarjeta con el cable RC68-68.

Las conexiones eléctricas del controlador son:

- $\boldsymbol{q_1}$ es una señal de entrada hacia el controlador y representa la posición angular de la articulación 1, con un voltaje en el rango de $\pm 10V$.
- $\boldsymbol{q_2}$ es una señal de entrada hacia el controlador y representa la posición angular de la articulación 2, con un voltaje en el rango de $\pm 10V$.
- $\boldsymbol{\tau_1}$ es una señal de salida del controlador y representa el par de control para la articulación 1, con un voltaje en el rango de $\pm 10V$.
- $\boldsymbol{\tau_2}$ es una señal de salida del controlador y representa el par de control para la articulación 2, con un voltaje en el rango de $\pm 10V$.

En la Figura 4.13 se muestra el diagrama de conexiones del block CB-68LP. Donde ACH0 y ACH1 son las entradas analógicas del controlador $\boldsymbol{q_1}$ y $\boldsymbol{q_2}$, los pines usados son 68 y 33 respectivamente, ambas entradas son referenciadas a tierra (pin 67). Las salidas analógicas $\boldsymbol{\tau_1}$ y $\boldsymbol{\tau_2}$, del controlador son DAC0OUT y DAC1OUT con los pines 22 y 21 respectivamente, referenciadas a tierra (pin 55).

68 ACH0	**61 ACH12**	**15 DGND**	**08 +5V**	01 DIO6P2
34 ACH8	**27 AIGND**	***49 DIO2P0***	42 DIO3P1	**35 DGND**
67 AIGND	**60 ACH5**	***16 DIO6P0***	**9 DGND**	2 DIO4P2
33 ACH1	**26 ACH13**	**50 DGND**	43 DIO2P1	**36 DGND**
66 ACH9	**59 AIGND**	***17 DIO1P0***	10 DIO1P1	3 DIO1P2
32 AIGND	**25 ACH6**	***51 DIO5P0***	**44 DGND**	37 DIO0P2
65 ACH2	**58 ACH14**	**18 DGND**	11 DIO0P1	**4 DGND**
31 ACH10	**24 AIGND**	***52 DIO0P0***	45 DIO2P2	38 DIO7P1
64 AIGND	**57 ACH7**	***19 DIO4P0***	**12 DGND**	5 DIO6P1
30 ACH3	**23 ACH15**	**53 DGND**	46 DIO3P2	39 DIO7P2
63 ACH11	**56 AIGND**	**20 RESERVED**	**13 DGND**	6 DIO5P1
29 AIGND	**22 DAC0OUT**	**54 AOGND**	***47 DIO3P0***	40 DIO5P2
62 AISENSE	**55 AOGND**	NA NA	**14 +5V**	**7 DGND**
28 ACH4	**21 DAC1OUT**	NA NA	***48 DIO7P0***	41 DIO4P1

Figura 4.13 Bloque de conexiones CB-68LP

Cabe mencionar que el puerto PCMCIA está descontinuado, por lo que es difícil encontrar una computadora con el mismo. Esto no presenta un problema para usar el ERM2GDL debido a que es compatible con cualquier tipo de controlador digital o analógico.

4.3.3 Conexiones del ERM2GDL

Como se mencionó anteriormente, el ERM2GDL que emula la planta, consta de 3 elementos que son el Circuito Acondicionador de Señales (CAS), sistema empotrado e Interfaz gráfica. En la Figura 4.14 se muestran sus elementos, donde el CAS se conecta al sistema empotrado mediante cables que contienen señales analógicas y digitales, y este a su vez, se conecta a la computadora mediante puerto USB para enviar y recibir datos de la interfaz gráfica. En las siguientes secciones se muestran las conexiones eléctricas del CAS a detalle.

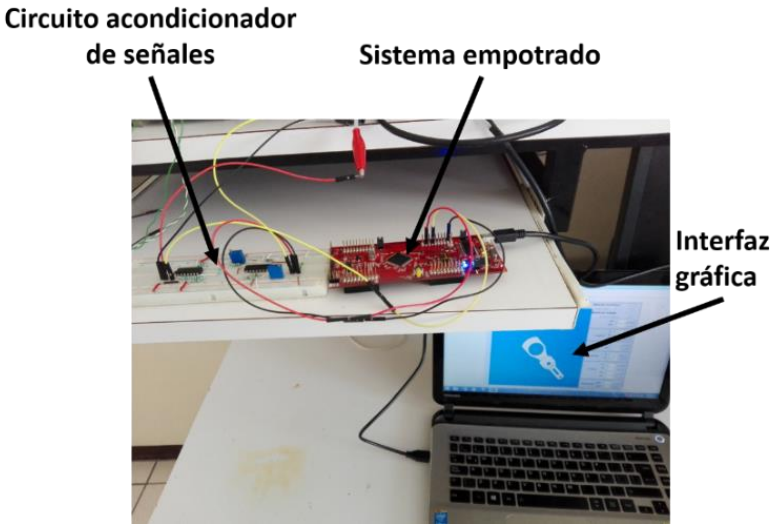


Figura 4.14 Conexiones del ERM2GDL

El CAS escala el voltaje para que el DSC pueda leerlas sin sufrir daños. El cual consta de 2 elementos que son el acondicionamiento analógico y el convertidor de niveles lógicos. En esta sección se explican sus conexiones para que el DSC sea compatible con otras tarjetas de adquisición de datos.

En la Figura 4.15 se observan de forma general las salidas y entradas analógicas, salidas digitales del Encoder, botón de RESET y entrada mini-USB para la comunicación serial del sistema empotrado.

Figura 4.15 Entradas y salidas

4.3.3.1 Circuito acondicionador de señales (CAS)

Para conectar eléctricamente el CAS al sistema empotrado, en la Figura 4.16 se muestran los pines usados del DSC para esta aplicación, donde se ven las entradas/salidas analógicas y digitales. Los pines 24 y 29 con la etiqueta de $\boldsymbol{S_1}$ *y* $\boldsymbol{S_2}$ indican las salidas analógicas, los pines 27 y 28 con la etiqueta $\boldsymbol{E_1}$ *y* $\boldsymbol{E_2}$ indican las entradas analógicas. Los pines 40, 39 y 38 con las etiquetas de $\boldsymbol{D_0}, \boldsymbol{D_1}$ *y* $\boldsymbol{D_2}$ indican la salida digital de Encoder para la articulación 1 (A, B e Índex), los pines 37, 36 y 35 con las etiquetas de $\boldsymbol{D_3}, \boldsymbol{D_4}$ *y* $\boldsymbol{D_5}$ indican la salida digital de Encoder para la articulación 2 (A, B e Índex).

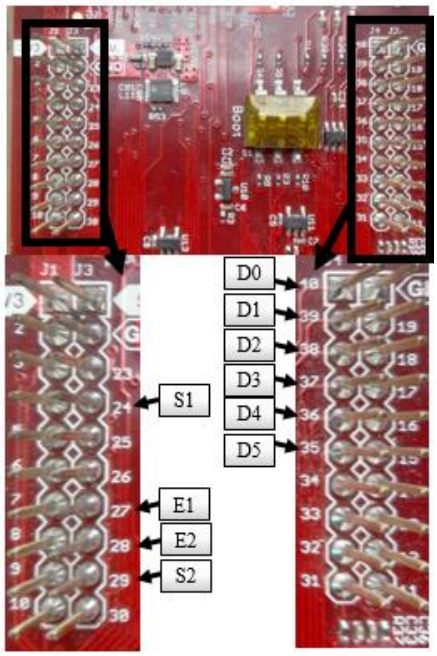


Figura 4.16 Entradas/salidas analógicas y digitales

Para las conexiones analógicas se usan las entradas y salidas siguientes:

- $\boldsymbol{E_1}$ es una señal de entrada al sistema empotrado y representa el par para la articulación 1, con un voltaje en el rango de *0* a *3V*.
- $\boldsymbol{E_2}$ es una señal de entrada al sistema empotrado y representa el par para la articulación 2, con un voltaje en el rango de *0* a *3V*.
- $\boldsymbol{S_1}$ es una señal de salida del sistema empotrado y representa la posición angular de la articulación 1, con un voltaje en el rango de *0* a *3V*.
- $\boldsymbol{S_2}$ es una señal de salida del sistema empotrado y representa la posición angular de la articulación 2, con un voltaje en el rango de *0* a *3V*.

4.3.3.1.1 Acondicionamiento analógico

El acondicionamiento analógico sirve para emparejar el rango de voltaje del ERM2GDL ($0 - 3V$) al del controlador ($\pm 10V$). Se usa un CAS de subida, para ampliar el rango de voltaje y un CAS de bajada, para disminuir el rango de voltaje. Las entradas/salidas (Figura 4.17) y rangos de voltaje del CAS analógico son:

- **El CAS de bajada** tiene como entrada un voltaje en el rango de $\pm 10V$ y en la salida tiene un rango de voltaje de $0 - 3V$.
- **El CAS de subida** tiene una entrada de voltaje en el rango de $0 - 3V$ y una salida con voltaje en el rango de $\pm 10V$.

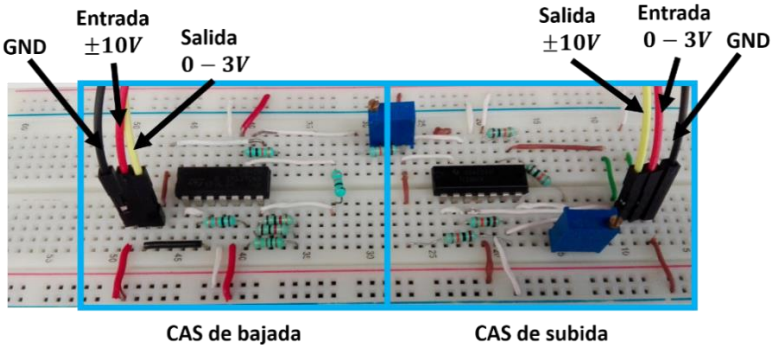


Figura 4.17 Circuito acondicionador de señales analógico de bajada y subida

Las conexiones entre el sistema empotrado, acondicionamiento analógico y controlador, se basan en la Figura 4.13, Figura 4.16 y Figura 4.17, donde se muestran entradas/salidas del acondicionamiento analógico, DSC y el controlador. En los siguientes puntos se explican las conexiones.

- La salida del CAS de bajada ($0 - 3V$) se conecta a la entrada analógica del sistema empotrado ($\boldsymbol{E_1}$ y $\boldsymbol{E_2}$).
- La entrada del CAS de bajada ($\pm 10V$) se conecta a la salida del controlador ($\boldsymbol{\tau_1}$ y $\boldsymbol{\tau_2}$).
- La entrada del CAS de subida ($0 - 3V$) se conecta a la salida analógica del sistema empotrado ($\boldsymbol{S_1}$ y $\boldsymbol{S_2}$).
- La salida del CAS de subida ($\pm 10V$) se conecta a la entrada del controlador ($\boldsymbol{q_1}$ y $\boldsymbol{q_2}$).

Las entradas y salidas del sistema empotrado se observan en la Figura 4.18.

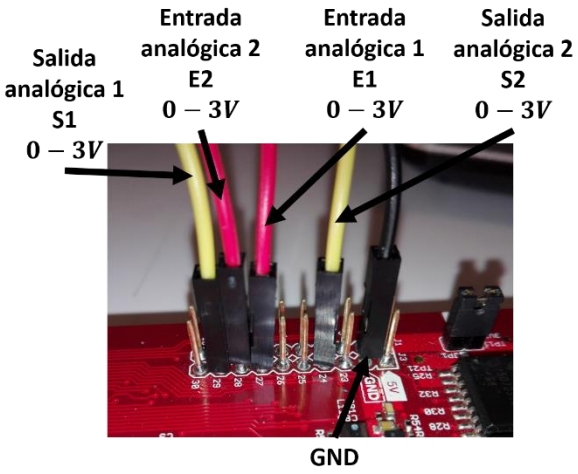


Figura 4.18 Entradas/Salidas analógicas

4.3.3.1.2 Convertidor de niveles lógicos

El convertidor de niveles lógicos se usa para elevar o disminuir el voltaje máximo de la salida digital. En esta aplicación se usa para elevar el voltaje de $3V$ del DSC a $5V$ para que sea compatible con otras tarjetas. En esta sección se explican las conexiones referentes al convertidor de niveles lógicos para elevar el voltaje de la salida de Encoder.

Las conexiones referentes a la salida de Encoder se basan en la Figura 4.16 y Figura 4.19. Los pines de la salida del Encoder se muestran en la Figura 4.16 donde el rango de voltaje es de $0 - 3V$, el cual se puede elevar al rango de $0 - 5V$ usando el convertidor de niveles lógicos bidireccional BOB-12009 que se muestra en la Figura 4.19. El cual se alimenta con ambos niveles de voltaje, es decir, con 3V en LV y 5V en HV referenciados a tierra.

Los pines LVx o HVx (donde x va de 1-4) sirven como entradas/salidas, dependiendo del pin que sea alimentado. Es decir, si se requiere que LVx sea la entrada y HVx la salida, se debe alimentar LVx y viceversa. Por ejemplo, para convertir de 3.3V a 5V, se conecta una de las salidas digitales del Encoder de 3.3V al pin LV1 y en el pin HV1 se tienen 5V.

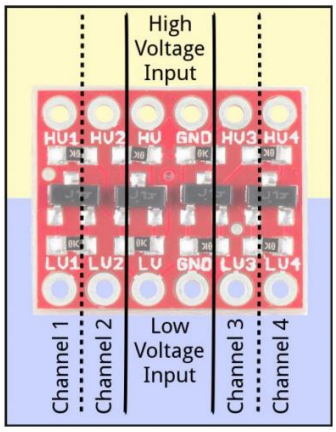


Figura 4.19 Convertidor de niveles lógicos BOB-12009

Las conexiones de la salida de encoder se muestran en la Figura 4.20.

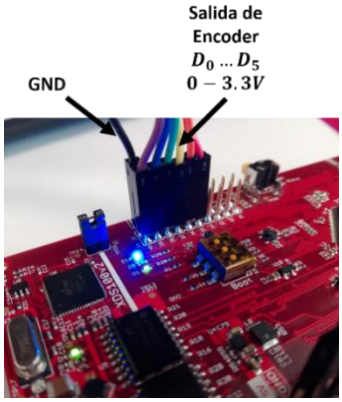


Figura 4.20 Salida de Encoder

4.4 Controladores de posición del robot de 2 GDL

El diseño del controlador consiste en modificar las características de sus elementos de tal manera que la respuesta de la configuración en lazo cerrado satisfaga los requisitos de funcionamiento [75].

En esta sección se implementaron 4 diferentes tipos de sistema de control de ganancia ajustable para regular la posición angular del robot de 2 GDL a una referencia deseada en ambas articulaciones. Dichos sistemas de control son: control por retroalimentación de todos los estados, PID y PD con compensación de gravedad. Los cuales se sintonizaron usando algoritmos genéticos multiobjetivo para disminuir el tiempo de establecimiento y el máximo sobreimpulso. También se diseñó un controlador por modos deslizantes de forma analítica.

Los parámetros del robot usados son los que se muestran en la Tabla 1.1, sin fricción de coulomb. En las secciones 4.4.1 a 4.4.4 se muestran las estructuras de control implementadas. Los resultados se presentan en las secciones 4.4.5 y 4.4.6.

4.4.1 Control PID

Esta técnica de control, requiere el valor del error de posición entre la referencia deseada y la posición del robot, para generar el par de control usando sus ganancias. El controlador PID está constituido de las acciones de control Proporcional, Integral y Derivativa. El diagrama a bloques de dicho controlador se muestra en la Figura 4.21.

Las variables K_p, K_i y K_d, representan las ganancias proporcional, integral y derivativa. Como salida se obtiene τ que representa el par de control, que disminuye el error y se define como la sumatoria de las ganancias multiplicadas por el error de posición.

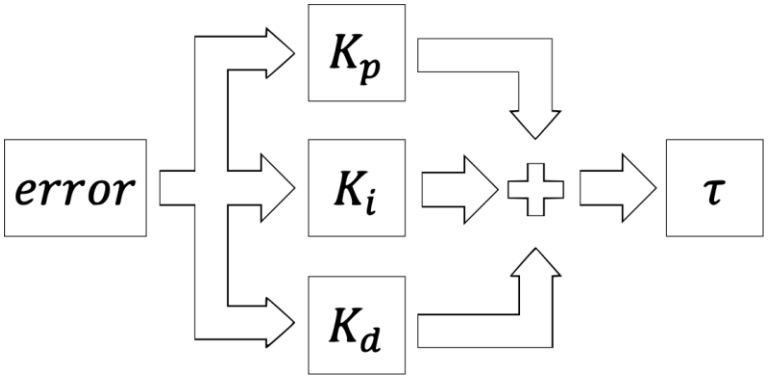


Figura 4.21 Controlador PID

4.4.2 Control PD con compensación de gravedad

El control PD con compensación de gravedad suprime las no linealidades que se presentan en el modelo dinámico del robot producidos por el par gravitatorio. Para ello se debe contar con una ecuación que describa el par gravitacional para eliminarlo, en esta aplicación es el término $\boldsymbol{g}(\boldsymbol{q})$ de la Ecuación 1.10.

El diagrama a bloques del controlador se muestra en la Figura 4.22, donde el error denota la diferencia entre la referencia deseada y la posición actual de una articulación. Las variables K_p y K_d representan las ganancias proporcional y derivativa. El par gravitacional compensa los efectos de la gravedad. La salida es τ que representa el par de control para disminuir el error.

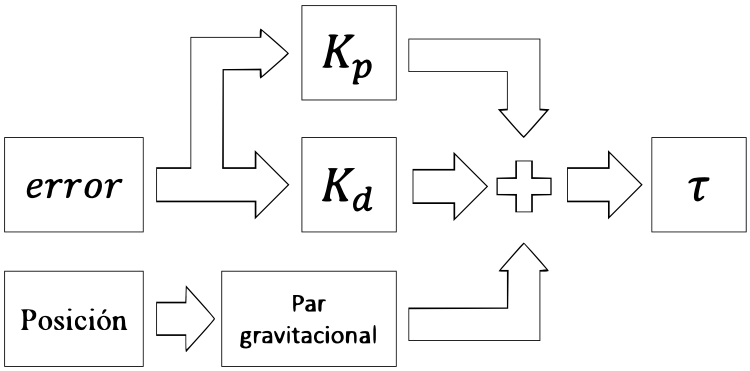


Figura 4.22 Controlador PD con compensación de gravedad

4.4.3 Control por retroalimentación de todos los estados

Esta técnica de control requiere del error de los estados del robot para calcular el par de control. En total son 4 estados que representan la posición y velocidad de cada eslabón. El diagrama a bloques del controlador se muestra en la Figura 4.23, donde $\hat{x}_i$ indica el error entre el valor actual y el valor deseado del estado i. Las variables K_{x1}, K_{x2}, K_{x3} y K_{x4}, representan las ganancias del primer hasta el cuarto estado. τ representa el par de salida para disminuir el error.

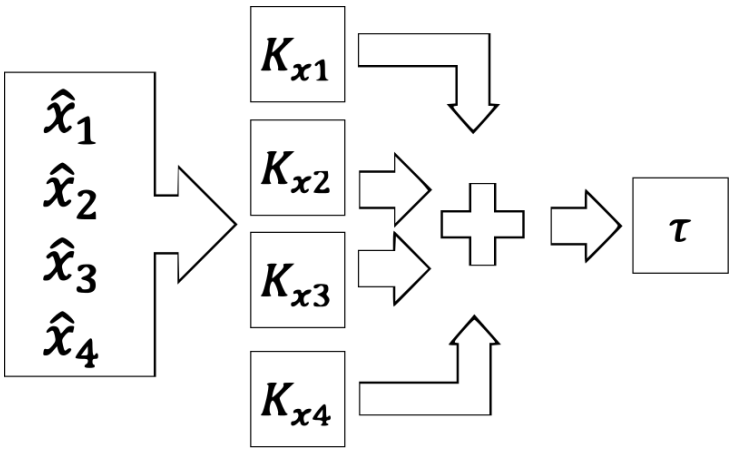


Figura 4.23 Controlador por retroalimentación de todos los estados

4.4.4 Control por modos deslizantes

Esta técnica de control requiere del conocimiento del modelo dinámico de la planta, de la posición y velocidad angular. Para esto se diseñaron dos controladores por modos deslizantes de forma analítica, donde la dinámica de los controladores se muestra en las ecuaciones 4.5 y 4.6. Donde Vaux1 y Vaux2 son señales de control auxiliares para cada articulación.

$$u_1 = M_{11}V_{aux1} + M_{12}V_{aux2} + C_{11}\dot{q}_1 + C_{12}\dot{q}_2 + g_1 \tag{4.5}$$

$$u_2 = M_{21}V_{aux1} + M_{22}V_{aux2} + C_{21}\dot{q}_1 + C_{22}\dot{q}_2 + g_2 \tag{4.6}$$

Donde $M_{ij},\ g_i\ y\ C_{ij}$ se toman de la representación matricial del sistema en las ecuaciones (1.10), y el control auxiliar son función de la superficie de deslizamiento s1 y s2 las cuales deben ser de grado relativo 1, así como de un voltaje de control equivalente Veq1 y Veq2 y las ganancias λ_1 y λ_2.

$$V_{aux1} = -\lambda_1 |V_{eq1}| sign(s_1) \tag{4.7}$$

$$V_{aux2} = -\lambda_2 |V_{eq2}| sign(s_2) \tag{4.8}$$

Donde $\lambda_1 = 10$ y $\lambda_2 = 20$.

Los voltajes equivalentes son función del error de posición de la articulación 1 y 2 y se acotan al valor máximo que se espera tener.

$$V_{eq1} = 2{c_1}^2 e_1 + 20 \tag{4.9}$$

$$V_{eq2} = 2{c_2}^2 e_2 + 20 \tag{4.10}$$

Donde $c_1 = 6$ y $c_2 = 6$.

La superficie de deslizamiento es:

$$s_1 = c_1 e_1 + \dot{q}_1 \tag{4.11}$$

$$s_2 = c_2 e_2 + \dot{q}_2 \tag{4.12}$$

Donde el error de posición angular es:

$$e_1 = q_1 - \bar{q}_1 \tag{4.13}$$

$$e_2 = q_2 - \bar{q}_2 \tag{4.14}$$

Las referencias deseadas son $\bar{q}_1$ y $\bar{q}_2$.

4.4.5 Sintonización de controladores de posición usando algoritmos genéticos multiobjetivo

Se van a probar 3 técnicas de control sintonizadas con algoritmos genéticos (AG). Éstas son retroalimentación por todos los estados, PD con compensación de gravedad y el PID. Existen muchas formas de ajustar las ganancias de dichos controladores, sin embargo una alternativa son los AG los cuales tienen las siguientes ventajas.

- No requieren conocimiento específico del problema para resolverlo.
- Operan de forma simultánea con varias soluciones, tomando información de varios puntos del espacio de búsqueda como guía.
- Resultan menos afectados por los máximos locales.
- Tienen habilidad para manipular muchos parámetros simultáneamente y alcanzar varios objetivos a la vez.

Un AG es un algoritmo matemático que transforma un conjunto de objetos matemáticos individuales, con respecto al tiempo, usando operaciones modeladas de acuerdo al principio Darwiniano de reproducción y supervivencia del más apto, y tras haberse presentado de forma natural una serie de operaciones genéticas de entre las que destaca la recombinación sexual. Cada uno de estos objetos matemáticos suele ser una cadena de caracteres (letras o números) de longitud fija que se ajusta al modelo de las cadenas de cromosomas, y se les asocia con una cierta función matemática que refleja su aptitud [76].

Los AG se usan como herramientas de búsqueda y optimización. En este caso se requiere encontrar una configuración de ganancias, que disminuyan el máximo sobreimpulso y el tiempo de establecimiento y el control de la posición de las articulaciones. Se usó un AG multiobjetivo por los siguientes puntos:

- Sintonizar 2 controladores en forma paralela, debido a que la acción de uno repercute en el comportamiento del otro. En caso de sintonizar cada controlador por separado y ejecutarlos al mismo tiempo se produce un mal funcionamiento de los controladores.
- Para optimizar 4 parámetros (Multiobjetivo), es decir, se disminuye el tiempo de establecimiento y el máximo sobreimpulso para ambas articulaciones simultáneamente.
- La búsqueda de ganancias se da en un número mínimo de iteraciones y de forma automatizada.
- La desventaja es el tiempo de búsqueda, debido a que se simula el sistema para cada configuración de ganancias.

Con una simulación numérica pura se sintonizó una serie de controladores de posición, con base en AG multiobjetivo. El objetivo de control es un corto tiempo de establecimiento con un criterio del 2% manteniendo al mínimo el máximo sobreimpulso (0% en el mejor de los casos) en la respuesta del sistema en ambas articulaciones del robot de 2 GDL.

La parte más importante de un AG es su función objetivo, con la cual se evalúa cada individuo para después compararlos y encontrar al más apto. Por lo que definimos a un individuo como una tupla, formada de un conjunto de números reales positivos que representan las ganancias de 2 controladores. El tamaño de la tupla depende del controlador a aplicar. Para sintonizar dos controladores PID se requieren de 6 ganancias y para dos controladores con base en retroalimentación de todos los estados se requieren de 8 ganancias.

En la Figura 4.24 se observa el diagrama de bloques de la función objetivo que se usa en el AG para determinar el desempeño de un individuo, dicha función simula el modelo no lineal del robot manipulador de 2 GDL en lazo cerrado con el controlador propuesto por un solo individuo. Al finalizar la simulación del individuo, se usa la posición angular en todo el tiempo de simulación para obtener el tiempo de establecimiento y máximo sobreimpulso para ambas articulaciones. Estas características definen el desempeño del individuo.

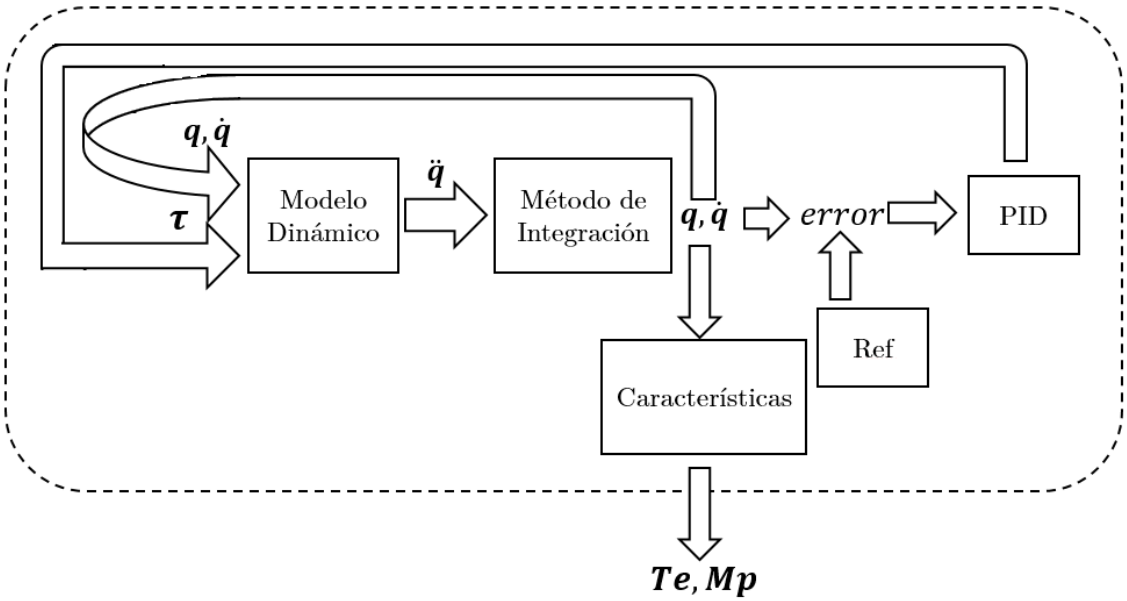


Figura 4.24 Diagrama de bloques de la función fitness

En la Figura 4.25 se muestra el diagrama de bloques del AG, el cual inicializa aleatoriamente *n* individuos. Cada individuo es evaluado con la función objetivo para obtener su desempeño. Se seleccionan los mejores individuos para la siguiente generación y se verifica que los criterios de paro han sido cumplidos, en caso contrario, se eliminan los peores individuos. Con los individuos de mejor desempeño se realiza el cruce y mutación para obtener nuevos individuos que formarán parte de la nueva población de *n* individuos para la siguiente generación. Se repite el proceso hasta cumplir con los criterios de paro.

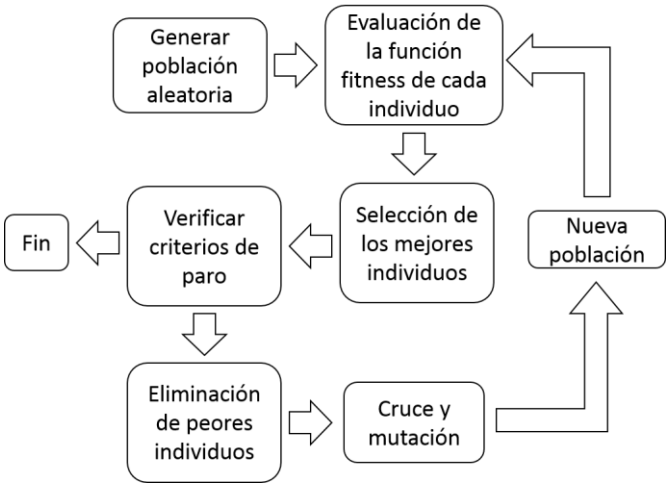


Figura 4.25 Diagrama del algoritmo genético

Para las secciones 4.4.5.1 a 4.4.5.3 se usó el mismo AG con las siguientes condiciones para la sintonización de las ganancias.

- El tipo de población es double
- El tamaño de la población es de 10
- El tipo de selección es con torneos con un tamaño de 2
- La probabilidad de cruce es de 0.8
- La probabilidad de mutación es de 0.001
- El número de generaciones es de 100

A continuación se presentan las ganancias obtenidas en la sintonización, las cuales se usaron en los controladores basados en la DAQ 6062E aplicados al simulador HIL. Las gráficas de posición son obtenidas de las salidas del ERM2GDL y se realiza una comparativa con los resultados obtenidos con simulación numérica pura.

4.4.5.1 Controlador PID sintonizado usando algoritmos genéticos multiobjetivo

Los rangos de ganancia para la búsqueda usando el algoritmo genético son los siguientes:

- El rango de búsqueda para la ganancia proporcional es de [0 300]
- El rango para la ganancia integral es de [0 600]
- El rango para la ganancia derivativa es de [0 50]

Las ganancias encontradas para el controlador de la primer articulación son $K_{p1} = 97.97$, $K_{i1} = 133.05$ y $K_{d1} = 31.2$. Las ganancias para el controlador de la segunda articulación son $K_{p2} = 241.83$, $K_{i2} = 31.87$ y $K_{d2} = 31.87$. Las referencias usadas en la búsqueda son $\bar{q}_1 = 180°$ y $\bar{q}_2 = 180°$. De acuerdo a las pruebas hechas con el controlador en lazo cerrado, se puede decir, que es estable de forma global, por lo que no es necesario resintonizar las ganancias para diferentes referencias.

Se implementó el controlador PID con las ganancias encontradas para la simulación HIL y simulación pura. La comparativa se muestra en la Figura 4.26 a), donde la referencias deseadas son $\bar{q}_1 = 90°$ y $\bar{q}_2 = 0°$. En la Tabla 4.6 se muestran los tiempos de establecimiento (T_e) y máximo sobreimpulso (M_p) para ambas articulaciones usando simulación HIL y simulación pura. Se observa que la respuesta en simulación HIL de la articulación 2 presenta una perturbación debida al ruido. En la simulación numérica se tiene un sobreimpulso y tiempo de establecimiento mayor para la primera articulación en comparación con lo obtenido en HIL. El tiempo de establecimiento máximo no supera los 3.5 segundos y el máximo sobreimpulso es de 16%.

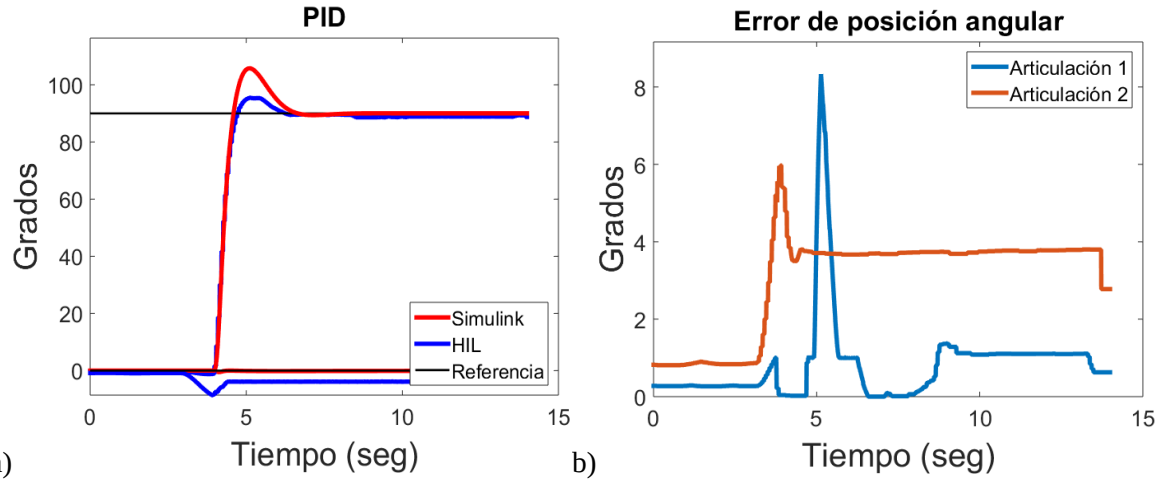
PID	Articulación 1		Articulación 2	
Característica	$T_e[seg]$	$M_p[\%]$	$T_e[seg]$	$M_p[\%]$
Simulación HIL	2.13	5	0	0
Simulación Pura	3.5	16	0	0

Tabla 4.6 Comparativa de tiempo de establecimiento y máximo sobreimpulso

Se obtuvo el error entre la respuesta con simulación HIL y simulación pura, el cual se muestra en la Figura 4.26 b) para la primera y segunda articulación. En la Tabla 4.7 se muestra el error máximo y mínimo para ambas articulaciones. Donde los picos de error se presentan cuando la velocidad en la articulación crece. El error se mantiene en su máximo valor durante un tiempo pequeño, se puede decir que el error es relativamente bajo.

Error	Máximo [grados]	Mínimo [Grados]	Promedio [grados]
Articulación 1	8.33	0.0015	3.04
Articulación 2	5.97	0.811	0.83

Tabla 4.7 Error máximo y mínimo



a) b)

Figura 4.26 Comparativa de las respuestas del sistema

4.4.5.2 Controlador PD con compensación de gravedad sintonizado usando algoritmos genéticos multiobjetivo

Los rangos de ganancia para la búsqueda con el algoritmo genético son los siguientes:

- El rango de búsqueda para la ganancia proporcional es de [0 300]
- El rango para la ganancia derivativa es de [0 300]

Las ganancias encontradas para el controlador de la primera articulación son $K_{p1} = 199.04$ y $K_{d1} = 50.76$. Las ganancias encontradas para el controlador de la segunda articulación son $K_{p2} = 192.19$ y $K_{d2} = 141.767$. Las referencias usadas en la búsqueda son $\bar{q}_1 = 180°$ y $\bar{q}_2 = 180°$. De acuerdo a las pruebas hechas con el controlador en lazo cerrado se puede decir que, es estable de forma global, por lo que no es necesario resintonizar las ganancias para diferentes referencias.

La respuesta del sistema con simulación HIL y la simulación pura se muestran en la Figura 4.27 a), donde las referencias deseadas son $\bar{q}_1 = 45°$ y $\bar{q}_2 = 90°$. En la Tabla 4.6 se muestran los tiempos de establecimiento (T_e) y máximo sobreimpulso (M_p) de ambas articulaciones con simulación HIL y simulación pura. Se observa una diferencia considerable entre las gráficas, esto se debe a la parte derivativa del controlador que genera un par muy elevado producido por el ruido presente en la interfaz analógica. El tiempo de establecimiento máximo no supera los 4 segundos en la articulación 2 y el máximo sobreimpulso es de 22%.

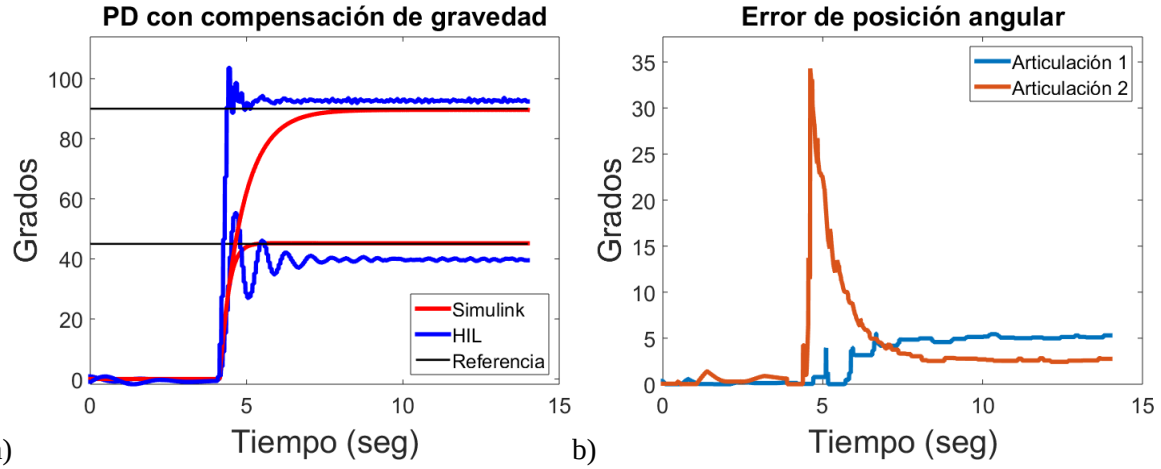
PD con compensación de gravedad	Articulación 1	Articulación 2		
Característica	$T_e[seg]$	$M_p[\%]$	$T_e[seg]$	$M_p[\%]$
Simulación HIL	3.38	22	1.48	15
Simulación Pura	0.99	0	4	0

Tabla 4.8 Comparativa de tiempo de establecimiento y máximo sobreimpulso

Se obtuvo el error entre la respuesta con simulación HIL y simulación pura, el cual se muestra en la Figura 4.27 b) para la primer y segunda articulación. En la Tabla 4.9 se muestra el error máximo, mínimo y promedio para ambas articulaciones. El error promedio muestra que el error de posición angular es pequeño en la mayoría del tiempo.

Error	Máximo [grados]	Mínimo [Grados]	Promedio [grados]
Articulación 1	5.5	0	2.85
Articulación 2	34.29	0.0066	3.65

Tabla 4.9 Error máximo y mínimo



a) b)

Figura 4.27 Comparativa de las respuestas del sistema

4.4.5.3 Controlador por retroalimentación de todos los estados sintonizado usando algoritmos genéticos multiobjetivo

Los rangos de ganancias para la búsqueda con el algoritmo genético son los siguientes:

- El rango de búsqueda para la ganancia del primer estado es de [0 600]
- El rango de búsqueda para la ganancia del segundo estado es de [0 600]
- El rango de búsqueda para la ganancia del tercer estado es de [0 600]
- El rango de búsqueda para la ganancia del cuarto estado es de [0 600]

Las ganancias encontradas para el controlador de la primera articulación son $K_{x1} = 40.65$ y $K_{x2} = 9.07$. Las ganancias encontradas para el controlador de la segunda articulación son $K_{x3} = 55.52$ y $K_{x4} = 3.78$. Las referencias usadas en la búsqueda son $\bar{q}_1 = 180°$ y $\bar{q}_2 = 180°$. De acuerdo a las pruebas hechas con el controlador en lazo cerrado, se puede decir, que las ganancias funcionan para referencias cercanas a las que fueron sintonizadas. Para alcanzar otras referencias es necesario resintonizar el controlador.

La respuesta del sistema con simulación HIL y la simulación pura se muestran en la Figura 4.28 a), donde las referencias deseadas son $\bar{q}_1 = 180°$ y $\bar{q}_2 = 180°$. En la Tabla 4.10 se muestran los tiempos de establecimiento (T_e) y máximo sobreimpulso (M_p) de ambas articulaciones usando la simulación HIL y simulación pura. Se observa que el sobreimpulso de la simulación pura es ligeramente mayor a la simulación HIL. El tiempo de establecimiento máximo para ambas gráficas no supera 1.5 segundos.

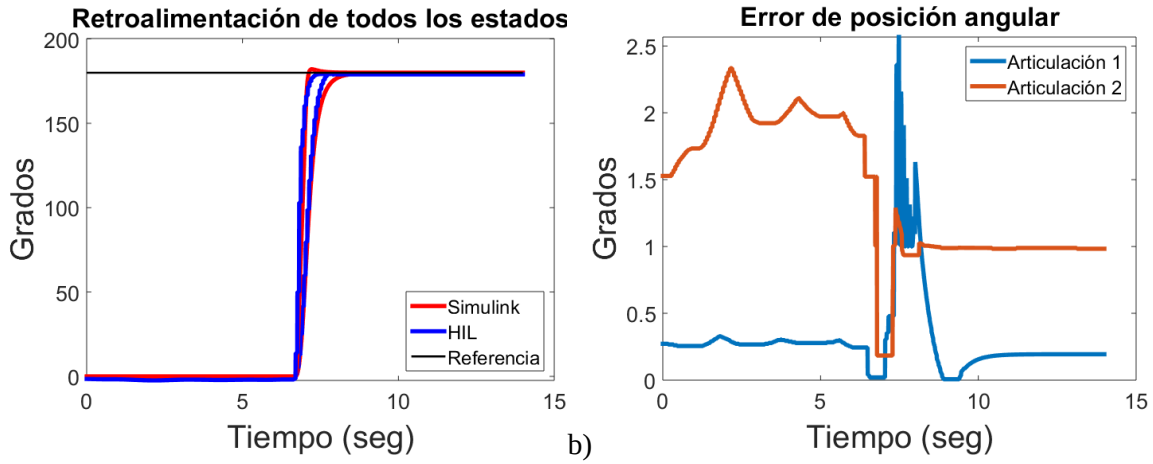
Retroalimentación de todos los estados	Articulación 1		Articulación 2	
Característica	$T_e[seg]$	$M_p[\%]$	$T_e[seg]$	$M_p[\%]$
Simulación HIL	0.73	0	1	0
Simulación Pura	1.41	0	0.44	1

Tabla 4.10 Comparativa de tiempo de establecimiento y máximo sobreimpulso

Se obtuvo el error entre la respuesta con simulación HIL y simulación pura, el cual se muestra en la Figura 4.28 b) para la primer y segunda articulación. En la Tabla 4.11 se muestra el error máximo y mínimo para ambas articulaciones. Se observa un error pequeño debido a que las respuestas son parecidas aun cuando se inició de 0° hasta 180°.

Error	Máximo [grados]	Mínimo [Grados]	Promedio [grados]
Articulación 1	2.7891	0.0069	0.29
Articulación 2	2.333	0.1840	1.4

Tabla 4.11 Error máximo y mínimo



a) b)

Figura 4.28 Comparativa de las respuestas del sistema

4.4.6 Controlador por modos deslizantes

La respuesta del sistema con simulación HIL y la simulación pura se muestran en la Figura 4.29 a). En la Tabla 4.12 se muestran los tiempos de establecimiento (T_e) y máximo sobreimpulso (M_p) de ambas articulaciones usando el simulador HIL y simulación pura. Las respuestas son relativamente parecidas pero en el simulador HIL se tiene un error de posición y se presenta ruido, los cuales no se presenta en la simulación numérica. El tiempo de establecimiento máximo no supera 1 segundo.

Modos deslizantes	Articulación 1		Articulación 2	
Característica	$T_e[seg]$	$M_p[\%]$	$T_e[seg]$	$M_p[\%]$
Simulación HIL	0.87	0	1	0
Simulación Pura	0.85	0	0.973	0

Tabla 4.12 Comparativa de tiempo de establecimiento y máximo sobreimpulso

Se obtuvo el error entre la respuesta con simulación HIL y simulación pura, el cual se muestra en la Figura 4.29 b) para la primer y segunda articulación. En la Tabla 4.11 se muestra el error máximo y mínimo para ambas articulaciones. Se observa un error pequeño que se debe a la robustez de este controlador.

Error	Máximo [grados]	Mínimo [Grados]	Promedio [grados]
Articulación 1	2.0718	7.42e-7	1.04
Articulación 2	2.4411	0.7611	1.57

Tabla 4.13 Error máximo y mínimo

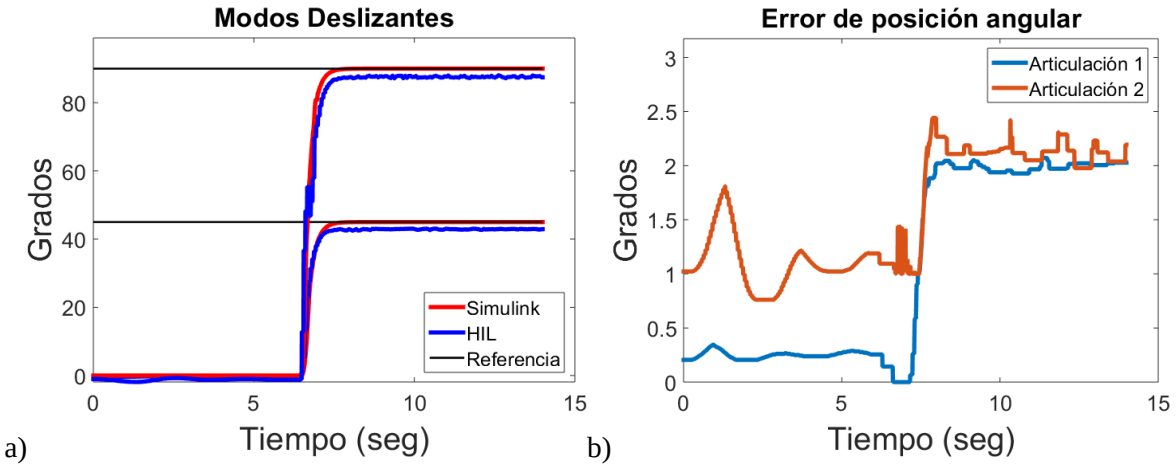


Figura 4.29 Error de la posición angular

4.4.7 Comparativa entre los controladores

La comparativa en el desempeño de los controladores usando el error en estado estacionario se muestra en la Tabla 4.14, donde se observa que el control por retroalimentación de todos los estados tiene un error menor pero como se mencionó, este desempeño se obtiene con la referencia a la que se sintonizó. Por otro lado el control por modos deslizantes demuestra un mejor desempeño al operar con cualquier referencia.

Error en estado estacionario	Articulación 1 [grados]	Articulación 2 [grados]
Retroalimentación de todos los estados	0.2	1
Modos deslizantes	1.98	2.02
PID	1.11	3.87
PD con compensación de gravedad	2.32	5.02

Tabla 4.14 Comparativa del error en estado estacionario

El error en estado estacionario se presentó en simulación HIL, esto tiene que ver con la aproximación en el comportamiento de los convertidores ADC y DAC usados en el sistema empotrado y el controlador con base en la DAQ, dado que no se comportan de forma ideal.

Capítulo 5. Conclusiones y trabajo a futuro

En este trabajo se desarrolló un emulador de un robot manipulador de 2 GDL que opera de acuerdo a los objetivos y metas planteadas en la propuesta de tesis. El cual es flexible, portátil, de fácil uso y de bajo costo.

El ERM2GDL opera en tiempo real con un periodo de muestreo de $100\,\mu s$, es decir, se obtuvo un periodo de muestreo 20 veces más pequeño, en comparación con lo planteado en la meta establecida, de un periodo de muestreo mínimo de $2\,ms$, para que la solución del modelo dinámico converja.

Este tiempo se obtuvo debido al número de ciclos de reloj generados en $100\,\mu s$ con una frecuencia de operación de 200 MHz, lo cual es una limitante. El tiempo se reducirá, al implementar el sistema en una tarjeta de mayor frecuencia y con recursos mejorados u optimizados. Que aparecerán en el mercado en algunos años, con precios accesibles.

En el sistema se tienen 2 entradas eléctricas analógicas, con las cuales se obtiene el par de entrada, para resolver el modelo dinámico del robot manipulador, dicho par de entrada puede ser generado por una tarjeta de adquisición de datos, que funcione como controlador. Para generar las salidas de forma eléctrica, se tienen 2 salidas de Encoder de cuadratura de tipo digital que indican el desplazamiento angular de ambas articulaciones y dos salidas analógicas que indican la posición angular absoluta.

Por tanto el control se puede realizar al leer las salidas analógicas o el Encoder. Esto permite un acercamiento a la aplicación real, debido a que hay que enfrentar los problemas que se presentan en las conexiones eléctricas, como son los efectos de impedancias en los sistemas electrónicos y el ruido que siempre está presente en cualquier tipo de sistema.

Por otra parte, se desarrolló una interfaz gráfica que permite al usuario modificar los parámetros físicos del robot, observar y registrar los efectos con gráficas de par, posición y velocidad angular. Dicha interfaz cuenta con un ambiente virtual en donde se muestra los movimientos del robot.

El ambiente virtual, sirve como apoyo en el proceso de estudio de la respuesta dinámica, del robot manipulador, porque muestra de forma visual, en tiempo real, los efectos físicos que producen ciertas entradas de par o la implementación de un sistema de control conectado de forma eléctrica.

La interfaz gráfica usa el protocolo de comunicación serial SCI, lo que implica que puede trabajar con cualquier tarjeta que maneje este protocolo. Esto permite, usar otro tipo de tarjetas para emular el robot manipulador. Desde la interfaz, se pueden configurar los parámetros de comunicación lo que le da mayor flexibilidad.

Se elaboró un manual de usuario, donde se explican las conexiones importantes del emulador, los pasos de instalación de la interfaz gráfica en una computadora personal, la programación del DSC y la forma de hacer uso del ERM2GDL. Esto con el fin, de que cualquier persona pueda reproducir el sistema diseñado, usarlo en prácticas de control y robótica para aplicar sistemas de control, sin la necesidad de tener la planta real en el laboratorio.

Además, de tener la facilidad de replicarlo, para contar con varios sistemas de este tipo, que es útil cuando se tiene una cantidad considerable de practicantes. Por otra parte el costo de este sistema es bajo, por lo que, la implementación de este tipo de sistemas para muchos usuarios es viable.

Se implementaron de manera satisfactoria, algunos controladores donde el ruido se presentó como en las aplicaciones reales. Como era de esperarse, los resultados obtenidos en la simulación numérica con Simulink, difieren de los obtenidos con simulación HIL. En gran parte se debe al ruido presente en la interfaz analógica.

Estas no linealidades normalmente desconocidas y difíciles de modelar, dan un mayor acercamiento a la implementación real. El error entre ambas respuestas, es relativamente pequeño, por lo que se puede decir que el ERM2GDL opera de forma adecuada.

La sintonización de algunos controladores usando algoritmos genéticos multiobjetivo, demuestra que dichos métodos de computación flexible, que se comportan como algoritmos de búsqueda y optimización, son una alternativa viable, para encontrar una serie de ganancias, que reduzcan el tiempo de establecimiento y sobreimpulso máximo en la respuesta del sistema.

Partiendo de un espacio de búsqueda grande, con un número mínimo de iteraciones y sin información previa de la planta, se obtiene un resultado, que cumple los requerimientos de diseño del controlador de forma razonable. Esto implica, un coste computacional considerablemente menor, al que se produce al mapear la mayoría del espacio de búsqueda y la reducción del tiempo de diseño en comparación, con la sintonización manual, que normalmente se realiza heurísticamente.

En este caso, se sintonizó un controlador, para un sistema de tipo MIMO altamente no lineal, lo cual es un problema complejo de resolver por métodos analíticos, donde normalmente, se debe tener conocimiento parcial o total del modelo dinámico de la planta, la planta debe estar linealizada alrededor de un punto de operación y en algunos casos, se debe conocer de forma exacta, los valores de los parámetros físicos.

Como trabajo a futuro, se plantea la implementación de un emulador de un robot de 3 a 6 grados de libertad u otra planta como: el carro péndulo, sistema bola viga o levitación magnética, los cuales son sistemas útiles para la implementación y pruebas de nuevos sistemas de control o para el practicante que requiera modelar y controlar dichas plantas.

Referencias

[1] C. Kohler, Enhancing Embedded Systems Simulation. A chip-Hardware-in-the-Loop Simulation Framework., Ute Wrasmann: Springer Fachmedien Wiesbaden GmbH, ISBN 978-3-8348-1475-3, 2011.

[2] G. Bracco, E. Giorcelli, G. Mattiazzo, V. Orlando y M. Raffero, «Hardware-In-the-Loop test rig for the ISWEC wave energy system,» *Mechatronics*, vol. 25, pp. 11-17, ISSN 0957-4158, February 2015.

[3] S. Manzoor, R. U. Islam, A. Khalid, A. Samad y J. Iqbal, «An open-source multi-DOF articulated robotic educational platform for autonomous object manipulation,» *Robotics and Computer-Integrated Manufacturing*, vol. 30, nº 3, pp. 351-362, ISSN 0736-5845, June 2014.

[4] M. Montazeri-Gh, M. Nasiri y S. Jafari, «Real-time multi-rate HIL simulation platform for evaluation of a jet engine fuel controller,» *Simulation Modelling Practice and Theory*, vol. 19, nº 3, pp. 996-1006, ISSN 1569-190X, March 2011.

[5] M. J. Mansur, «Rapid Prototyping and Evaluation of Control System Designs for Manned and Unmanned Applications,» *American Helicopter Society Annual Forum*, p. 1571, 2000.

[6] A. Palladino, G. Fiengo y D. Lanzo, «A portable hardware-in-the-loop (HIL) device for automotive diagnostic control systems,» *ISA Transactions*, vol. 51, nº 1, pp. 229-236, ISSN 0019-0578, January 2012.

[7] H. He, R. Xiong, K. Zhao y Z. Liu, «Energy management strategy research on a hybrid power system by hardware-in-loop experiments,» *Applied Energy*, vol. 112, pp. 1311-1317, ISSN 0306-2619, December 2013.

[8] Y. Kim, A. Salvi, J. B. Siegel, Z. S. Filipi, A. G. Stefanopoulou y T. Ersal, «Hardware-in-the-loop validation of a power management strategy for hybrid powertrains,» *Control Engineering Practice*, vol. 29, pp. 277-286, ISSN 0967-0661, August 2014.

[9] R. Moore, K. Hauer, G. Randolf y M. Virji, «Fuel cell hardware-in-loop,» *Journal of Power Sources*, vol. 162, nº 1, pp. 302-308, ISSN 0378-7753, 8 November 2006.

[10] V. Tavoosi, R. Kazemi y S. Hosseini, «Vehicle Handling Improvement with Steer-by-Wire System Using Hardware in the Loop Method,» *Journal of Applied Research and Technology*, vol. 12, nº 4, pp. 769-781, ISSN 1665-6423, August 2014.

[11] O. Ma, A. Flores-Abad y T. Boge, «Use of industrial robots for hardware-in-the-loop simulation of satellite rendezvous and docking,» *Acta Astronautica*, vol. 81, nº 1, pp. pp. 335-347, ISSN 0094-5765, December 2012.

[12] K. Saulnier, D. Pérez, R. Huang, D. Gallardo, G. Tilton y R. Bevilacqua, «A six-degree-of-freedom hardware-in-the-loop simulator for small spacecraft,» *Acta Astronautica,* vol. 105, nº 2, p. December 2014, 444-462, ISSN 0094-5765.

[13] H. Benninghoff, F. Rems y T. Boge, «Development and hardware-in-the-loop test of a guidance, navigation and control system for on-orbit servicing,» *Acta Astronautica,* vol. 102, pp. 67-80, ISSN 0094-5765, September–October 2014.

[14] B. Allotta, R. Conti, E. Meli, L. Pugi y A. Ridolfi, «Development of a HIL railway roller rig model for the traction and braking testing activities under degraded adhesion conditions,» *International Journal of Non-Linear Mechanics*, vol. 57, December 2013.

[15] R. Conti, E. Meli, A. Ridolfi y A. Rindi, «An innovative hardware in the loop architecture for the analysis of railway braking under degraded adhesion conditions through roller-rigs, Mechatronics, Volume 24, Issue 2, March 2014, pp. 139-150, ISSN 0957-4158.,» *Mechatronics,* vol. 24, nº 2, pp. 139-150, ISSN 0957-4158, March 2014.

[16] Y. Zhou, J. Lin, Y. Song, Y. Cai y H. Liu, «A power hardware-in-loop based testing bed for auxiliary active power control of wind power plants,» *Electric Power Systems Research,* vol. 124, pp. 10-17, ISSN 0378-7796, July 2015.

[17] M. Shahbazi, P. Poure, S. Saadate y M. R. Zolghadri, «Five-leg converter topology for wind energy conversion system with doubly fed induction generator,» *Renewable Energy,* vol. 36, nº 11, pp. 3187-3194, ISSN 0960-1481, November 2011.

[18] J. She y J. Jiang, «Potential improvement of CANDU NPP safety margins by shortening the response time of shutdown systems using FPGA based implementation,» *Nuclear Engineering and Design,* vol. 244, pp. 43-51, ISSN 0029-5493, March 2012.

[19] J.-H. Jung, «Power hardware-in-the-loop simulation (PHILS) of photovoltaic power generation using real-time simulation techniques and power interfaces, Journal of Power Sources,» *Journal of Power Sources,* vol. 285, pp. 137-145, ISSN 0378-7753, 1 July 2015.

[20] H. Giberti y D. Ferrari, «A novel hardware-in-the-loop device for floating offshore wind turbines and sailing boats,» *Mechanism and Machine Theory,* vol. 85, pp. 82-105, ISSN 0094-114X, March 2015.

[21] P. Boscariol, A. Gasparetto y V. Zanotto, «Design and experimental validation of a hardware-in-the-loop simulator for mechanisms with link flexibility,» *Proceedings of the ASME 2010, 10th Biennial Conference on Engineering Systems Design and Analysis, ESDA2010*, July 12-14, 2010, Istanbul, Turkey.

[22] K. J. Kaliński y M. A. Galewski, «Vibration surveillance supported by Hardware-In-the-Loop Simulation in milling flexible workpieces,» *Mechatronics,* vol. 24, nº 8, pp. 1071-1082, ISSN 0957-4158, December 2014.

[23] S. Wahyudi, M. J.E. y A. Albagul, «Development of a Microcontroller-Based Control System with a Hardware-in-the-Loop (HIL) Method for Control Education using MATLAB/Simulink/xPC Target,» *Int. J. Engng,* vol. 21, nº 5, pp. 846-854, 2005.

[24] A. M. El-Nagar y M. El-Bardini, «Interval type-2 fuzzy neural network controller for a multivariable anesthesia system based on a hardware-in-the-loop simulation,» , *Artificial Intelligence in Medicine,* vol. 61, nº 1, pp. 1-10, ISSN 0933-3657, May 2014.

[25] A. M. El-Nagar y M. El-Bardini, «Practical Implementation for the interval type-2 fuzzy PID controller using a low cost microcontroller,» *Ain Shams Engineering Journal,* vol. 5, nº 2, pp. 475-487, ISSN 2090-4479, June 2014.

[26] O. I. Caldas, «Sistema de Control de una Planta Embebida en FPGA Empleando hardware-in-the-loop,» pp. pp. 51-59, ISSN 0012-7353, April 2013.

[27] C. Wang, Z. Jiao, S. Wu y Y. Shang, «An experimental study of the dual-loop control of electro-hydraulic load simulator (EHLS),» *Chinese Journal of Aeronautics,* vol. 26, nº 6, pp. 1586-1595, ISSN 1000-9361, December 2013.

[28] L. Pugi, E. Galardi, C. Carcasci, A. Rindi y N. Lucchesi, «Preliminary design and validation of a Real Time model for hardware in the loop testing of bypass valve actuation system,» *Energy Conversion and Management,* vol. 92, pp. 366-384, ISSN 0196-8904, 1 March 2015.

[29] O. Crăciun, A. Florescu, I. Munteanu, A. I. Bratcu, S. Bacha y D. Radu, «Hardware-in-the-loop simulation applied to protection devices testing,» *International Journal of Electrical Power & Energy Systems,* vol. 54, pp. 55-64, ISSN 0142-0615, January 2014.

[30] G. Randolf y R. M. Moore, «Test system design for Hardware-in-Loop evaluation of PEM fuel cells and auxiliaries,» *Journal of Power Sources,* vol. 158, nº 1, pp. 392-396, ISSN 0378-7753, 14 July 2006.

[31] R. Chhabra y M. R. Emami, «A holistic concurrent design approach to robotics using hardware-in-the-loop simulation,» *Mechatronics,* vol. 23, nº 3, pp. 335-345, ISSN 0957-4158, April 2013.

[32] M. Kähler, R. Rachholz, S. Herrmann y C. Woernle, «Development of a Biomechanical Multibody Model for the Hardware-in-the-loop Simulation of Total Hip Endoprostheses,» de *1st Joint International Conference on Multibody System Dynamics*, Lappeenranta, Finland.

[33] S. Seung-Woo, K. Eui-Jin, L. Chan-Ho y H. Jong-Sung, «Development of a Novel HyRoHILS System and Its Application to Parameter Identification of an Indutrial Robot,» de *Robotics*, July 2010.

[34] R. Lal y R. Fitch, «A Hardware-in-the-Loop Simulator for Distributed Robotics,» *Australasian Conference on Robotics and Automation (ACRA),* December 2-4, 2009, Sydney, Australia.

[35] A. Vasconcelos, L. F. S. Costa, J. C. Viera y S. Junior, «Control and Monitoring of Mobile Robots through Hardware-in-the-loop simulation,» pp. 1-2, October 2014.

[36] K. Seong-Hoon y P. Hong Seong, «Design of a Robot-in-the-loop Simulation Based on OPRoS,» *Journal of Institute of Control,* February 2013.

[37] J. Rios, R. Yoldi, A. Plaza y X. Iriarte, «Real-Time Hardware-in-the-loop Simulation of a Exaglide Type Parallel Manipulator on a Real Machine Controler,» December 2012.

[38] P. Tempel, P. Miermeister y A. Pott, «Kinematics and Dynamics Modeling for Real-Time Simulations of the Cable-Driven Parallel Robot IPAnema 3,» October 2015.

[39] L. Zouari, M. B. Ayed y A. Mariem, «A hardware in the loop simulation for electrically driven robot manipulator,» *Canadian Conference on Electrical and Computer Engineering,* pp. 1495-1501, June 2015.

[40] H. Temeltas, M. Gokasan y S. Bogosyan, «Hardware in the Loop Robot Simulators for On-site and Remote Education in Robotics,» *Int. J. Engng Ed.,* vol. 22, nº 4, pp. 815-828, 2006.

[41] G. Stöpler, T. Menzel y S. Douglas, «Hardware-in-the-loop simulation of machine tools and manufacturing systems,» *Computing and Control Engineering,* vol. 1, nº 16, pp. 10-15, February 2015.

[42] G. D. B. Salazar, D. A. Hurtado y O. R. Sandoval, «Hardware-in-the-loop simulation and digital control of double inverted pendulum,» December 2015.

[43] M. Adrian y E. Reza, «Dynamic Load Emulation in Hardware-in-the-loop Simulation of Robot Manipulators,» *IEEE Transactions on Industrial Electronics,* vol. 7, nº 58, pp. 2980-2987, July 2011.

[44] H. Philippe, B. Pascal, B. Julien y B. Michel, «Hardware-in-the-loop simulation of an impedance controlled robot using a direct-drive test bench,» may 2008.

[45] W. F. Xu, B. Liang, Y. S. Xu, C. Li y W. Y. Qiang, «A ground experiment system of free-floating space robot for capturing space target,» *J. Intell. Robot. Syst,* vol. 2, nº 48, pp. 187-208, 2007.

[46] H. Sawada, S. Szuki y M. Oda, «A contact dynamics analysis of spacecraft capturing and rigidizing,» de *Proceedings of the 17th Worksho on JAXA astrodynamics and Flight Mechanics,* 2007.

[47] T. Boge y O. Ma, «Using advanced industrial robotics for spacecraft rendezvous and docking simulation,» de *IEEE International Conference on Robotics and Automation,* Shanghai China, 2011.

[48] M. Zebenay, T. Boge, R. Krenn y D. Choukroun, «Analytical and experimental stability investigation of a hardware-in-the-loop satellite docking simulator,» *J. Aerosp. Eng,* vol. 4, nº 229, pp. 666-681, 2015.

[49] C. Qi, X. Zhao, F. Gao, A. Ren y Q. Sun, «Contact stiffness and damping identification for hardware-in-the-loop contact simulator with measuremente delay compensation,» *Acta astronautica,* nº 123, pp. 171-180, March 2016.

[50] R. J. Guillermo, M. J. Tuxpan, P. Sánchez-Sánchez, F. Reyes-Cortés, L. C. Gómez, A. Michua-Camarillo, B. Calderón-Flores, J. Cid-Monjaraz, J. A. Mancilla-Morales y G. Morales-Timal, «Evaluación de 4 estructuras de Control Mediante un simulador Basado en un Robot de 2 Grados de Libertad,» *Congreso Nacional de Mecatrónica,* Noviembre 2009.

[51] P. Bravo, F. Reyes, S. Vergara y E. Ríos, «Diseño de un Simulador en 3D para Robots Manipuladores de 3 Grados de Libertad,» de *8 Congreso Nacional de Mecatrónica*, Veracruz, Veracruz, Noviembre 2009.

[52] D. Checa, D. Luna y V. Mosquera, «Simulador de un Robot SCARA de 4 Grados de Liberad Basado en Realidad Virtual,» de *Cuarto Congreso colombiano de Computación*, Bucaramanga, 2009.

[53] T. Ramos, «Simulación de la plataforma robótica HOAP-3 en el simulador OPENHRP3,» 2009.

[54] A. Loyarte, «Plataforma de Simulación de un Sistema Real de Control de Niveles Basada en Software Libre,» de *Congreso Nacional de Ingeniería Informática y sistemas de Información* , Córdoba, Argentina, November 2013.

[55] A. Loyarte, M. Julia-Vega y J. Ruben, «Laboratorio Virtual Remoto para Educación en Control: Emulación de una Planta Real de Control de Niveles».

[56] S. Jose T, R. Lui E, M.-K. Carmen y A. Granados, «Simulador de un Manipulador de 6 Grados de Libertad,» pp. 27-30, May 2015.

[57] C. Zelayeta, M. Sabalza, G. Gentiletti y Y. J. Garbino, «Diseño e Implementación de un Simulador para Entrenamiento en Cirugia Robótica,» de *XIX Congreso Argentino de Bioingenieria y VIII Jornada de Ingeniería Clínica*, San Miguel Tucumán, ISBN: 978-987-23-950-7-0, September 2013.

[58] F. J. Martínez, R. González, F. Rodriguez y J. L. Guzmán, «Laboratorio Virtual y Remoto para la Enseñanza de Robótica Paralela,» de *CEA Jornadas de Enseñanza de la Ingenieria de Sistemas y Automática*, 2010.

[59] F. R. Cortés, Robótica, Control de Robots Manipuladores, México D.F.: Alfaomega Grupo Editor, febrero 2013.

[60] F. H. R. Leyva, L. A. P. Gaspar y F. S. Espinosa, «Simulacion de un Robot de Dos Grados de Libertad Usando "Hardware-in-the-loop" con Base en Instrumentacion Programable,» *Pistas educativas.*

[61] L. WILEY, Métodos numéricos, México D.F.: Grupo Noriega Editores, 2013.

[62] P. Lapsley, DSP Processor Fundamentals: Architectures and Features, Berkeley, California: Wiley-IEEE Press, 1 edition (February 7, 1997).

[63] A. V. Deshmukh, Microcontrollers: Theory and Applications, Delhi: TATA MC Graw-Hill, ISBN 0-07-058595-4, 2005.

[64] F. REYES CORTÉS y J. CID MONJARAZ, ARDUINO - Aplicado en Robótica, Mecatrónica e Ingenierías, Alfaomega, 2015.

[65] Texas Instruments, «Code composer Studio,» Texas Instruments, [En línea]. Available: http://www.ti.com/tool/ccstudio. [Último acceso: 21 Octubre 2015].

[66] Texas Instruments, «Texas Instruments Homepage,» [En línea]. Available: www.ti.com. [Último acceso: 2015 Octubre 2015].

[67] F. S. Adán y E. J. Macías, «Ingeniería Gráfica en simulación de sistemas y procesos,» *Libro de actas de las XXII Jornadas de Automatica de la CEA-IFAC*, 2002.

[68] F. T. Medina, F. A. C. Herías, S. T. P. Méndez, F. G. O. Zamora, J. P. Baeza y G. Vázquez, «Laboratorio virtual remoto para la enseñanza de robótica,» 2002.

[69] E. Y. L. Gu, A Journey from Robot to Digital Human, Mathematical Principles and Applications with MATLAB Programming, New York: Springer-Verlag Berlin Heidelberg., 2013.

[70] «Texas Instruments,» [En línea]. Available: www.ti.com/lit/pdf/SPRUHX5. [Último acceso: 24 09 2016].

[71] «Texas Instruments,» [En línea]. Available: www.ti.com/lit/pdf/spru430. [Último acceso: 24 09 2016].

[72] «National Instruments,» [En línea]. Available: www.ni.com/pdf/manuals/370724c.pdf. [Último acceso: 27 09 2016].

[73] R. Pressman, Ingeniería del software: un enfoque práctico., España: 5° Edición, McGraw Hill Interamericana de España, 2001.

[74] J. R. Lajara y J. P. Sebastia, LABVIEW: Entorno Gráfico de Simulación, México: ALFAOMEGA: MARCOMBO, 2011.

[75] R. Hernández, Introducción a los sistemas de control, Naucalpan de Júarez, Estado de México: Pearson Educación, ISBN: 978-607-442-842-1, 2010.

[76] J. Holland, «Adaptation in Natural and Artificial Systems,» *University of Michigan Press*, p. 211, 1975.

Acrónimos

ADC (Analog to Digital Converter)

CLA (Control Law Accelerator)

DAC (Digital to Analog Converter)

DSC (Digital Signal Controller)

EPWM (Enhanced Pulse Width Modulation)

FPGA (Field Programmable Gate Array)

GDL (Grados de Libertad)

GPIO (General Purpose Input Output)

HIL (Hardware-in-the-loop)

VISA (Virtual Instruments Software Architecture)

HRPWM (High Resolution Pulse Width Modulation)

ISR (Interrupt Service Routine)

PWM (Pulse Width Modulation)

RHIL (Robot-Hardware-in-the-loop)

ROTRADI (Robot de Transmisión Directa)

SCI (Serial Communication Interface)

IFR (Interrupt Flag Register)

IER (Interrupt Enable Register)

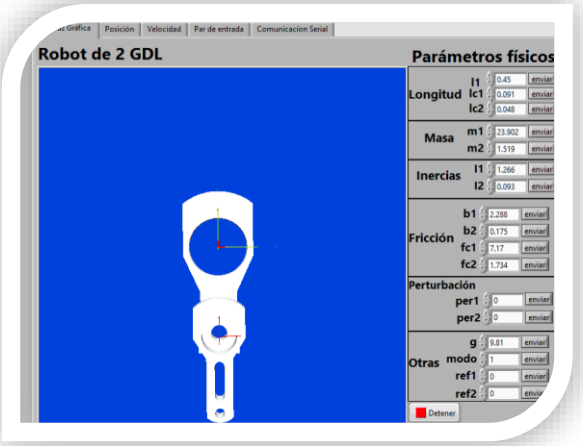
PIE (Peripheral Interrupt Expansion)

ERM2GDL (Emulador de un Robot Manipulador de 2 Grados de Libertad)

DAQ (Data Acquisition)

Apéndice A. Manual de usuario

MANUAL DE USUARIO



Emulador de un robot manipulador de 2 grados de libertad

UTM

Septiembre 2016

1. Introducción

El Emulador de un Robot Manipulador de 2 Grados de Libertad (ERM2GDL) es una herramienta didáctica que puede ser usada en las materias de Dinámica de Sistemas, Control Automático y Robótica, con esta herramienta el alumno de ingeniería puede tener una mejor comprensión del modelado, comportamiento y control del Robot Manipulador de 2 grados de libertad (GDL), el cual es un sistema no lineal de múltiples entradas y múltiples salidas (MIMO).

El sistema consta de un controlador digital de señales (DSC) Delfino F28377S de la firma Texas Instruments (TI), el cual resuelve en tiempo real el modelo dinámico del robot de 2 grados de libertad. Por su puerto serie envía hacia una computadora personal la información de la posición, velocidad y par que se le aplica al modelo del robot. En ésta se ejecuta una interfaz gráfica (IG) en la que se tiene un modelo virtual del robot, en la que muestra la posición en la que se encuentra, además de que se puede configurar los parámetros del mismo. Para manejar al ERM2GDL el usuario no requiere conocer la arquitectura del DSC.

Los componentes principales del ERM2GDL (Figura A.1) son:

- Interfaz Gráfica de Usuario instalada en una computadora personal.
- Acondicionamiento de señales, que contiene el circuito acondicionador de señales y el convertidor de niveles lógico bidireccional BOB-12009.
- Kit Delfino F28377S donde sus componentes se muestran en la Figura A.2 los cuales son:
    - Tarjeta LAUNCHXL-F28377S
    - Cable Mini-USB, de 0.5 metros.

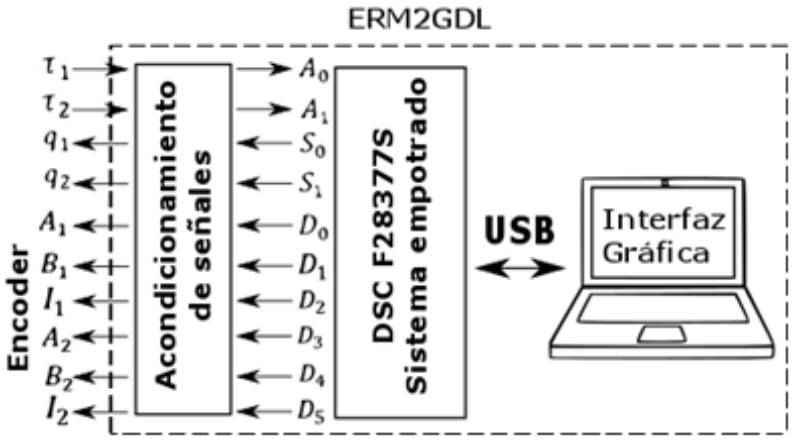


Figura A.1 ERM2GDL

Figura A.2 Kit de desarrollo Delfino F28377S

2. Programación del DSC

Para que el DSC se comporte eléctricamente igual al robot es necesario programarlo. Para esto se requiere tener instalado el software computacional Code Composer Studio (CCS) 6.1.0 o superior.

Como primer paso, se ejecuta CCS para importar el proyecto que contiene el código fuente del sistema empotrado. En la ventana principal de CCS, en el menú de “File” se selecciona “Import” (Figura A.3).

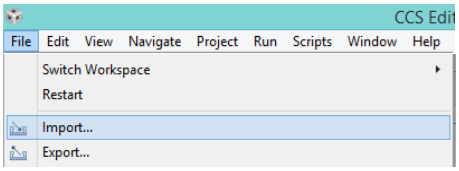


Figura A.3 Importar proyecto

Aparece una nueva ventana que se muestra en la Figura A.4 donde se elige el tipo de archivo. Debido a que el proyecto se encuentra en formato comprimido, se selecciona “Archive File” y se presiona “Next”.

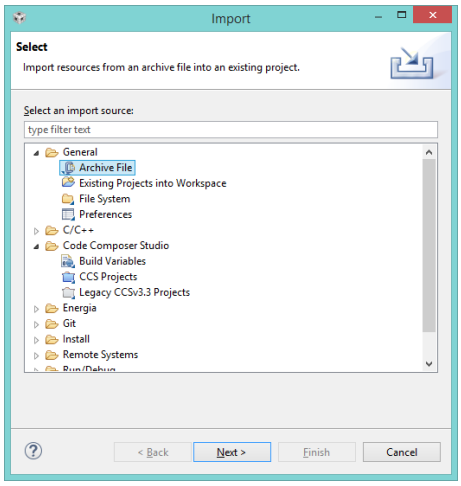


Figura A.4 Selección tipo de archivo

En la siguiente ventana (Figura A.5) se presiona el botón de “Browse” para buscar el archivo a importar el cual tiene el nombre de “ERM2GDL.zip”. Una vez seleccionado el archivo se presiona “Finish” para importar el proyecto.

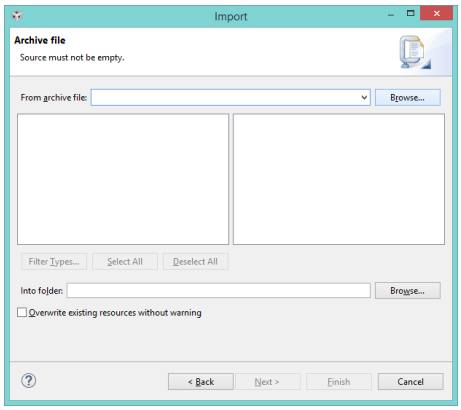


Figura A.5 Búsqueda del archivo

Una vez que se importó el proyecto se observa la carpeta “ERM2GDL” que se muestra en la Figura A.6. La cual debe seleccionarse.

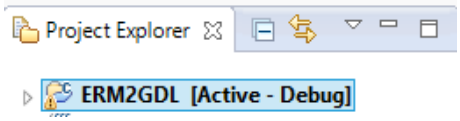


Figura A.6 Proyecto importado

Una vez elegido el proyecto ERM2GDL, se presiona el botón para compilar y posteriormente se presiona el botón para programar el DSC como lo muestra la Figura A.7.

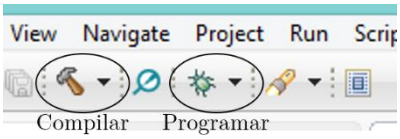


Figura A.7 Compilar y programar

Al compilar el código se observa que existen advertencias como lo muestra la Figura A.8, estas advertencias no son de importancia, por lo que es seguro programar el DSC.

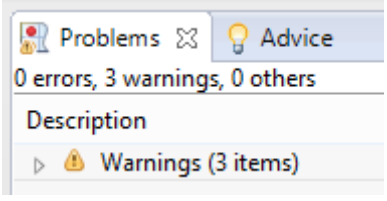


Figura A.8 Advertencias

Durante el proceso de programación es importante no desconectar la tarjeta ni presionar el botón de Reset. Esto puede provocar que la tarjeta quede inservible. Una vez que se completa la programación aparecen los botones que se muestran en la Figura A.9. Se debe presionar el botón de detener para que CCS regrese a su ventana inicial. En este punto el DSC puede ser desconectado y reconectado sin pérdida de programación. Pudiendo usarlo con o sin la interfaz gráfica.

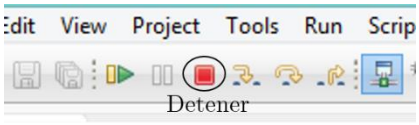


Figura A.9 DSC programado

3. Instalación del software

Para que el sistema funcione con la interfaz se requiere instalar dos paquetes de software en una computadora personal. Los cuales son la interfaz gráfica y el controlador del DSC. Los requerimientos mínimos son:

- Procesador Pentium 4/M.
- 128 MB de RAM.
- Espacio en disco de 220 MB.
- Resolución de pantalla 1024x768 pixeles.
- Windows XP o superior.
- Arquitectura de 64 bits.

El software mencionado sólo puede instalarse en sistemas de 64 bits. El orden de instalación debe ser el siguiente para evitar problemas.

1. **Controlador del DSC F28377S** es necesario para establecer comunicación por USB entre el DSC y la Interfaz Gráfica.
2. **Instalador de la Interfaz gráfica** para visualizar el ambiente virtual y modificar parámetros físicos, dicho instalador contiene el software necesario para agregar el paquete NI-VISA al equipo, dicho paquete se usa para establecer comunicación con diferentes tipos de tarjeta. Para este caso, usa la comunicación serial con el DSC.

3.1 Instalación de los controladores del Delfino F28377S

Los controladores del DSC Delfino F28377S sirven para que la computadora detecte a la tarjeta como un puerto de comunicaciones. Antes de la instalación de la interfaz, se deben instalar los controladores del DSC, en la carpeta de instalación se ejecuta el archivo DPInst64.exe en modo administrador, como se muestra en la Figura A.10

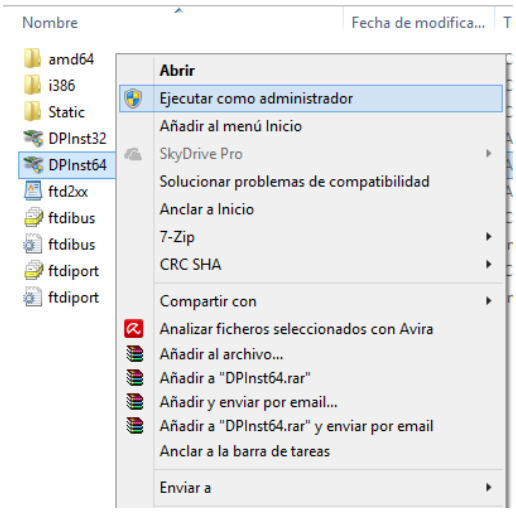


Figura A.10 Instalador de los controladores

Una vez que se ejecuta el archivo DPInst64.exe, se muestra la ventana de la Figura A.11, donde se ve el asistente de instalación de controladores de dispositivos, presione el botón siguiente para iniciar la instalación.

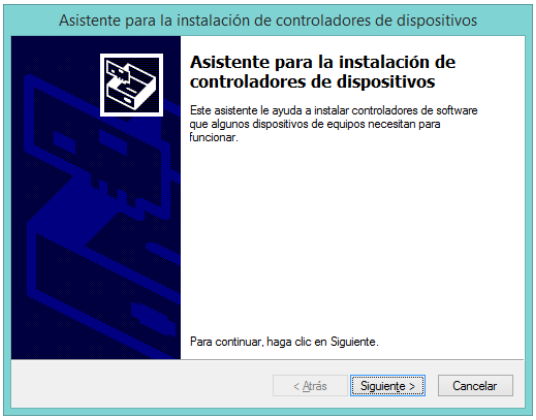


Figura A.11 Instalación de controladores

Cuando la instalación está completa se muestra la ventana de la Figura A.12, donde los controladores están instalados, para cerrar la ventana presione el botón de finalizar.

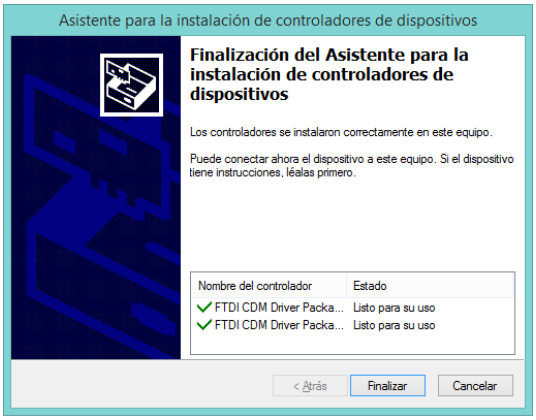


Figura A.12 Controladores instalados

3.2 Instalación de la interfaz gráfica

La interfaz gráfica fue diseñada para ser ejecutada en una computadora personal o laptop. Se utilizó el lenguaje de programación gráfica LabVIEW. Se generó un instalador para que el programa se pueda configurar adecuadamente. En esta sección, se explican los pasos a seguir para la instalación de la interfaz, tome en cuenta que los controladores del DSC deben estar previamente instalados en el equipo.

Para instalar la interfaz, en la carpeta que contiene los archivos de instalación, se busca el archivo “setup.exe” como se muestra en la Figura A.13, y se **ejecuta**.

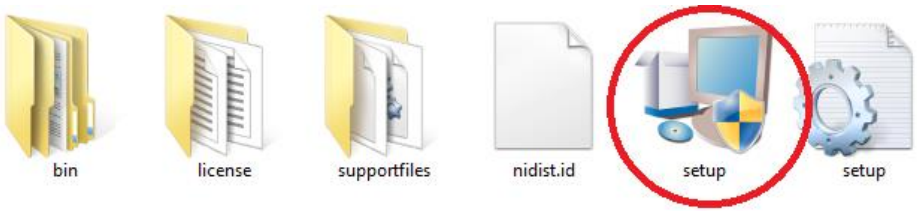


Figura A.13 Instalador de la interfaz gráfica

Una vez que se ejecuta el instalador, se abre una ventana (Figura A.14) donde se puede seleccionar el directorio para la instalación de los paquetes, se recomienda dejarlo por **default** y presionar **siguiente** para continuar. En esta ventana se muestra los paquetes a instalar, es decir, controladores de VISA y la interfaz gráfica.

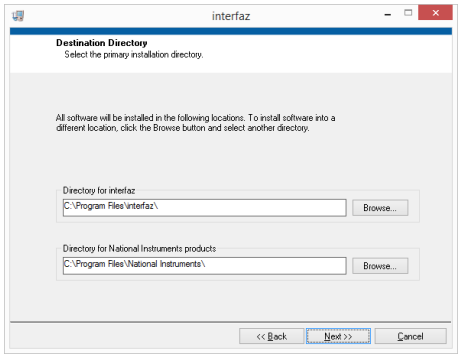


Figura A.14 Selección de directorios

Se muestra el contrato de licencia, una vez que lo lea, haga click a "**I accept the above 2 License Agreements(s)**" y a **siguiente** (Figura A.15) para continuar con la instalación.

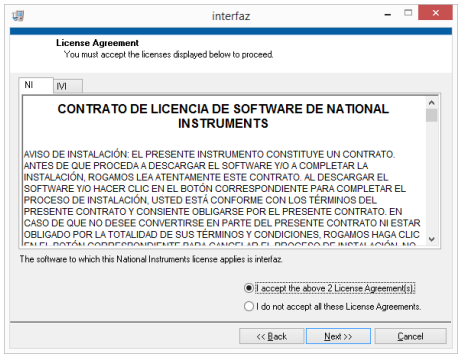


Figura A.15 Contrato de licencia de software

Para prevenir problemas en la instalación de hardware nuevo, es conveniente deshabilitar el **inicio rápido de Windows**, esta operación es **altamente recomendable**. Presione el botón de **siguiente** (Figura A.16) para deshabilitar el inicio rápido y continuar.

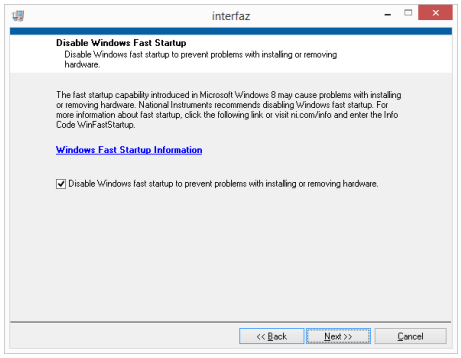


Figura A.16 Deshabilitar inicio rápido de Windows

Aparecerá la ventana de la Figura A.17, donde se muestran los paquetes a instalar, presione **siguiente** para iniciar la instalación.

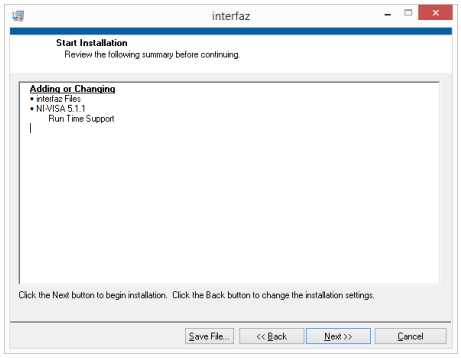


Figura A.17 Iniciar instalación y revisión de los paquetes a instalar

La Figura A.18 muestra la instalación en proceso, que puede durar varios minutos.

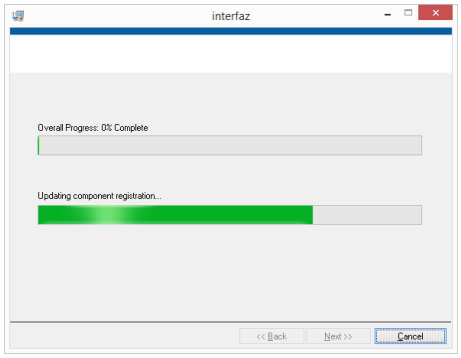


Figura A.18 Instalación en proceso

Una vez completada la instalación, la computadora personal debe ser reiniciada para que los controladores instalados funcionen con normalidad.

4. ¿Cómo empezar?

Para usar el ERM2GDL, es necesario tener **instalados** los controladores del DSC y la interfaz gráfica en una computadora personal, sino ha hecho esto, debe revisar las especificaciones e instrucciones de la sección de **Instalación del software** para realizar dichas tareas.

Una vez que se ha instalado la interfaz gráfica y los controladores del DSC, el siguiente paso es **conectar** el DSC con la computadora personal por medio del cable Mini-USB (Figura A.19), esto se debe hacer, antes de ejecutar la interfaz gráfica. En la Figura A.20 se muestran las conexiones. Una vez que se energiza la tarjeta, la comunicación esta deshabilitada. Por tanto hay que presionar en ésta, el botón de **Reset** para inicializar la comunicación por puerto serie.

Figura A.19 Mini-USB

Para que el ERM2GDL funcione, **no requiere** la conexión con la computadora. Por default la tarjeta ejecuta el modelo del sistema, leyendo el par de entrada con las entradas analógicas. Si no se tiene una computadora personal, el DSC puede ser alimentado con una fuente de voltaje externa de 5V mediante el conector USB.

Para mayor detalle en cuanto a las conexiones del ERM2GDL con la etapa de acondicionamiento de señales se debe revisar la sección **Conexión del CAS**.

Figura A.20 Conexiones USB

5. Interfaz gráfica

Se diseñó una interfaz gráfica para visualizar el movimiento del robot, guardar e historial con los datos de posición, velocidad y par de cada una de las articulaciones. Desde está se pueden configurar parámetros físicos del modelo dinámico. El intercambio de información se realiza por medio de un enlace de comunicación por puerto serie, entre éste y el ERM2GDL.

La interfaz gráfica consta de 3 secciones importantes que son el ambiente virtual, control de los parámetros de comunicación por puerto serie y gráficas de posición velocidad y par. Dichas secciones hacen un total de 5 pestañas, que se muestran en el siguiente orden: Ambiente virtual (Default), gráfica de posición, gráfica de velocidad, gráfica de par, control de comunicación por puerto serie. Los elementos y uso de cada pestaña se explican en detalle más adelante.

Antes de ejecutar la interfaz gráfica, es necesario presionar el botón de **Reset** del DSC (Figura A.21), lo que reinicializa la comunicación por puerto serie.

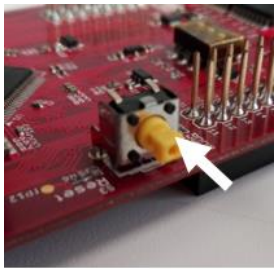


Figura A.21 Botón de RESET

5.1 Ambiente virtual

Una vez que se haya ejecutado la interfaz, se mostrará en pantalla una ventana que presenta la pestaña inicial o por defecto (Figura A.22), donde se pueden identificar los siguientes elementos:

- Ambiente virtual que muestra los movimientos del robot de 2 GDL.
- Pestañas para navegar entre diferentes presentaciones y control de datos, las cuales se explican en las subsecciones siguientes.
- Botón de paro para detener la interfaz.
- Panel de parámetros físicos, con el cual se modifican los parámetros físicos del modelo dinámico, se elige el método numérico y el modo de trabajo del DSC.
- Led indicador que se enciende cuando la interfaz está en ejecución.

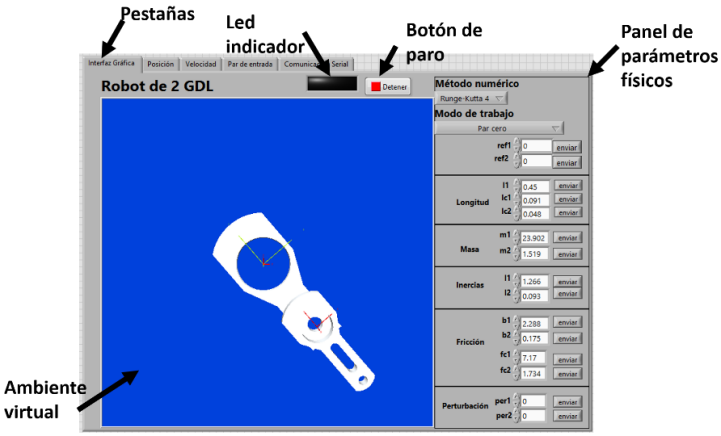


Figura A.22 Ambiente virtual

5.1.1 Panel de parámetros físicos

El panel de parámetros físicos, se usa para modificar los parámetros físicos del robot manipulador de 2 GDL, cambiar el modo de trabajo del DSC y el método numérico usado por el DSC. En esta sección se explica su uso.

5.1.1.1 Modificar parámetros físicos

Dicho panel, contiene grupos de parámetros físicos con el mismo diseño, como el que se muestra en la Figura A.23. Para modificar un parámetro, se tiene que identificar el tipo de parámetro con la **descripción**, el valor del **parámetro físico** que se modifica con un **control numérico** auxiliado

de un **indicador numérico**. Cuando se ha elegido el valor a modificar se presiona el botón de **enviar,** para generar el comando al DSC, el cual modifica el valor del parámetro seleccionado.

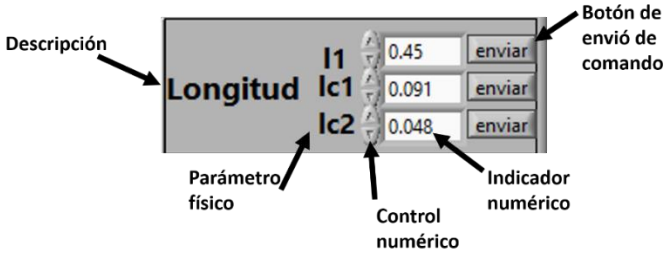


Figura A.23 Panel de parámetros físicos

5.1.1.2 Modo de trabajo

El modo de trabajo indica, la forma como se obtendrá el par de entrada para cada articulación del robot manipulador de 2 grados de libertad. Desde el panel de parámetros físicos se tiene la opción de cambiar el modo de trabajo del sistema empotrado, por medio de un menú desplegable, como se muestra en la Figura A.24. En la aplicación se cuenta con 3 modos de trabajo del DSC que son:

- **Par cero**, el sistema empotrado, ignora las entradas analógicas y establece el par de entrada en cero para ambas articulaciones. Este modo es útil cuando se requiere observar el comportamiento del sistema en lazo abierto con entradas de par igual a cero.
- **Par de PID interno**, el sistema empotrado, ignora las entradas analógicas y establece el par de entrada usando un controlador de posición interno. Dicho controlador es de tipo PID, el cual posiciona al robot de acuerdo al valor de las referencias deseadas. Este modo es útil cuando se requiere controlar de forma externa o simular la planta, a partir de una posición deseada. Este modo se establece por default cuando el sistema empotrado se conecta a la interfaz. En caso de no usar la interfaz gráfica por default es: **Par de entradas analógicas**.
- **Par de entradas analógicas**, el sistema empotrado, lee las entradas analógicas y las escala para obtener el par para ambas articulaciones. Este modo es útil para aplicar controladores externos.

5.1.1.3 Método numérico

El panel tiene un menú desplegable que permite elegir entre dos métodos numéricos que son Runge-Kutta de orden 4 y el método de Euler.

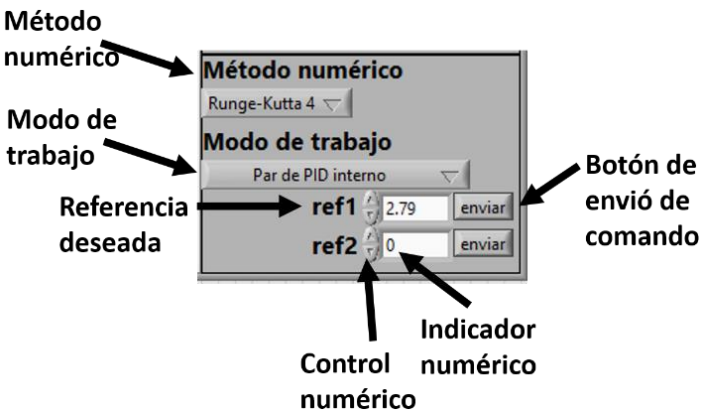


Figura A.24 Modos de trabajo

5.2 Pestañas de posición, velocidad y par

Para presentar los datos recibidos del DSC mediante gráficas, se diseñaron las pestañas de posición, velocidad y par, las cuales son parecidas en cuanto a su presentación. En la Figura A.25 se muestra la gráfica de posición angular de ambas articulaciones. Cada una de las gráficas tiene:

- **Unidad de medida** que se muestra en el Sistema Internacional Metro, Kilogramo y Segundo (MKS). La posición se presenta en $grados$, la velocidad en $\frac{grados}{segundo}$ y el par en $Newton \cdot metro$.
- **Tiempo** representa el eje de las abscisas en las gráficas.
- **Indicadores de valor** que muestra el último valor recibido por el DSC.
- **Gráficas** que representan la posición, velocidad o par en ambas articulaciones.

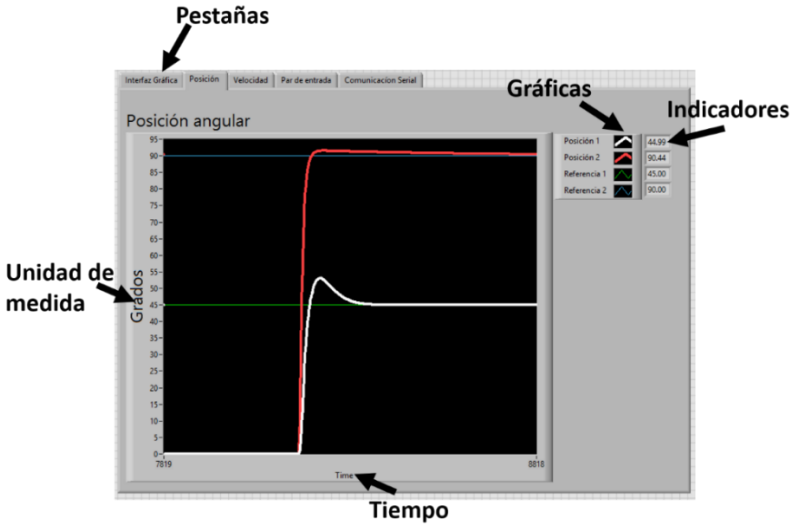


Figura A.25 Gráficas de posición

5.2.1 Exportar datos desde las gráficas

Se puede exportar los datos recibidos del DSC, dando click derecho sobre la gráfica, después ir al menú de Exportar y elegir entre copiar los datos al portapapeles, guardarlos en un archivo Excel o exportar como imagen como lo muestra la Figura A.26.

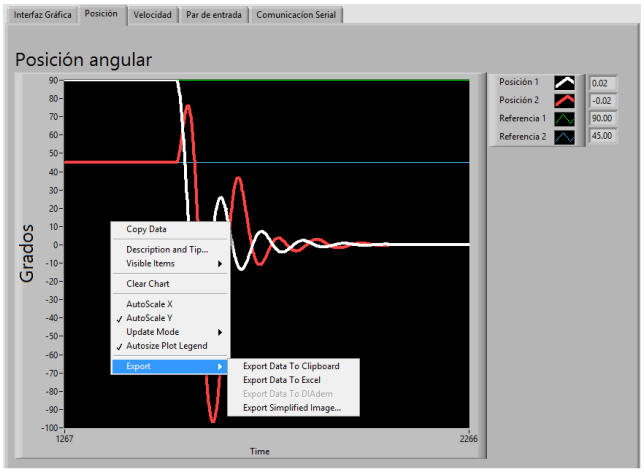


Figura A.26 Exportación de datos

5.3 Configuración de la comunicación serial

La pestaña de configuración de la comunicación serial, muestra los controles para modificar los parámetros de la comunicación con el DSC, en la Figura A.27 se ve el puerto COM, los parámetros de comunicación, carácter de terminación de recepción o envío y el buffer que muestra los datos recibidos desde el DSC.

La computadora asigna un puerto COM a cada dispositivo que maneje comunicación serial, el usuario debe ver el puerto COM que se le asignó al DSC y seleccionarlo usando el menú desplegable en esta pestaña. El sistema empotrado se configuró para una velocidad de 57600 baudios, de 8 bits de datos, sin paridad y con 1 bit de paro. El carácter de terminación de recepción o envío en este caso se estableció en un salto de línea por lo que la escritura y lectura terminan cuando se presenta un salto de línea. El buffer almacena los datos recibidos para ser procesados, los cuales se muestran en un indicador.

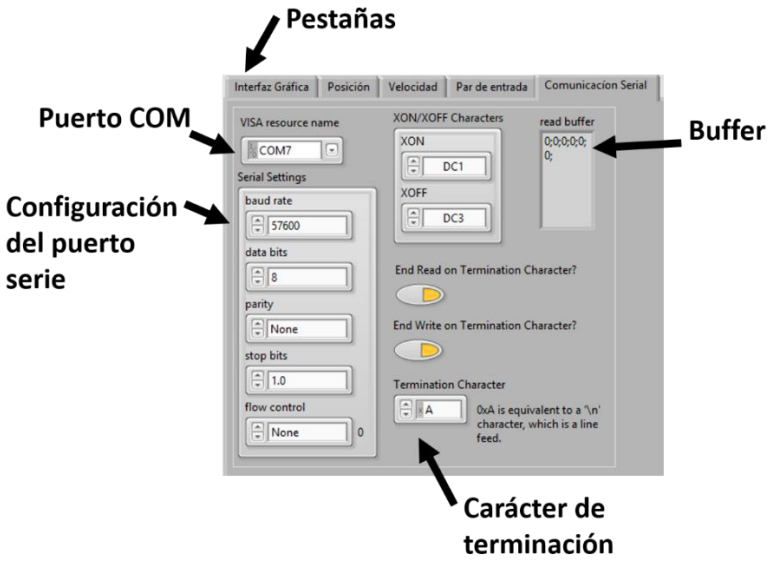


Figura A.27 Configuración de la comunicación serial

5.4 Descripción de errores

En esta sección se presentan 2 casos donde la IG muestra una ventana de error y se explican sus posibles causas. El primer caso se da cuando se ejecuta la interfaz sin tener conectada la tarjeta a la computadora, la ventana de la Figura A.28 se muestra cuando esto sucede.

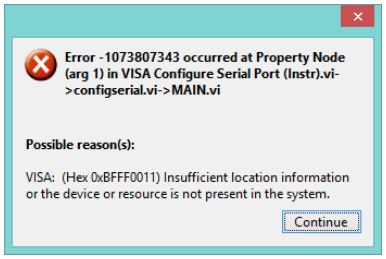


Figura A.28 Error de conexión

El segundo caso ocurre cuando se conecta la tarjeta sin presionar el botón de **Reset** o si la comunicación serial está mal configurada, es decir, si no se eligió el puerto COM correcto se muestra la ventana de la Figura A.29

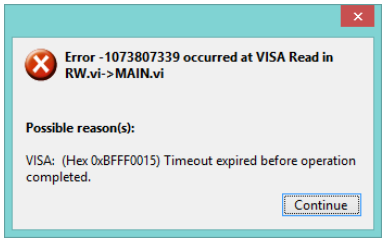


Figura A.29 Tiempo expirado

6. Conexión del CAS

Como se mencionó anteriormente, el ERM2GDL que emula la planta, consta de 3 elementos que son el Circuito Acondicionador de Señales (CAS), sistema empotrado e Interfaz gráfica. En la Figura A.30 se muestran sus elementos, donde el CAS se conecta al sistema empotrado mediante cables que contienen señales analógicas y digitales, y este a su vez, se conecta a la computadora mediante puerto USB para enviar y recibir datos de la interfaz gráfica. En las siguientes secciones se muestran las conexiones eléctricas del CAS a detalle.

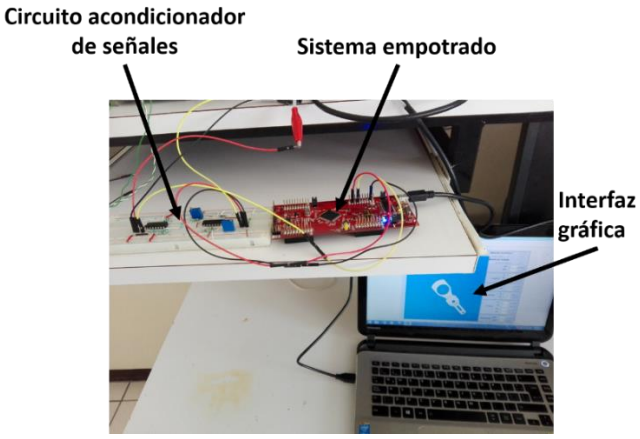


Figura A.30 Conexiones del ERM2GDL

El CAS escala el voltaje para que el DSC pueda leerlas sin sufrir daños. El cual consta de 2 elementos que son el **acondicionamiento analógico** y el **convertidor de niveles lógicos**. En esta sección se explican sus conexiones para que el DSC sea compatible con otras tarjetas de adquisición de datos. En la Figura A.31 se observan de forma general las salidas y entradas analógicas, salidas digitales del Encoder, botón de RESET y entrada mini-USB para la comunicación serial del sistema empotrado.

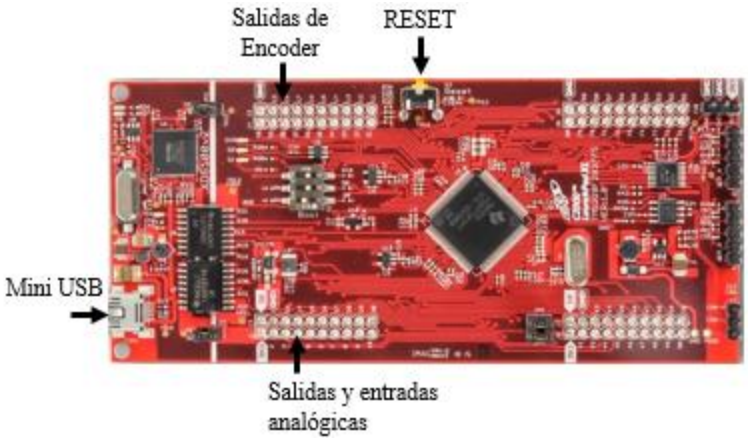


Figura A.31 Entradas y salidas

6.1 Circuito acondicionador de señales (CAS)

Para conectar eléctricamente el CAS al sistema empotrado, en la Figura A.32, se muestran los pines usados del DSC para esta aplicación, donde se ven las entradas/salidas analógicas y digitales. Los pines 24 y 29 con la etiqueta de $\boldsymbol{S_1}$ $\boldsymbol{y}$ $\boldsymbol{S_2}$ indican las salidas analógicas, los pines 27 y 28 con la etiqueta $\boldsymbol{E_1}$ $\boldsymbol{y}$ $\boldsymbol{E_2}$ indican las entradas analógicas. Los pines 40, 39 y 38 con las etiquetas de $\boldsymbol{D_0}, \boldsymbol{D_1}$ $\boldsymbol{y}$ $\boldsymbol{D_2}$ indican la salida digital de Encoder para la articulación 1 (A, B e Índex), los pines 37, 36 y 35 con las etiquetas de $\boldsymbol{D_3}, \boldsymbol{D_4}$ $\boldsymbol{y}$ $\boldsymbol{D_5}$ indican la salida digital de Encoder para la articulación 2 (A, B e Índex).

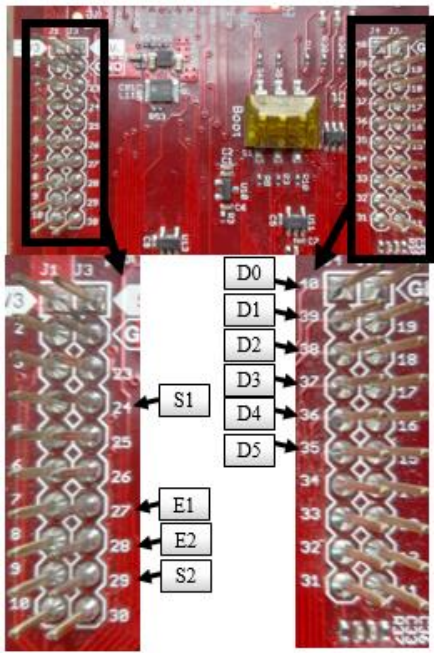


Figura A.32 Entradas/salidas analógicas y digitales

Para las conexiones analógicas se usan las entradas y salidas siguientes:

- $\boldsymbol{E_1}$ es una señal de entrada al sistema empotrado y representa el par para la articulación 1, con un voltaje en el rango de *0* a *3V*.
- $\boldsymbol{E_2}$ es una señal de entrada al sistema empotrado y representa el par para la articulación 2, con un voltaje en el rango de *0* a *3V*.
- $\boldsymbol{S_1}$ es una señal de salida del sistema empotrado y representa la posición angular de la articulación 1, con un voltaje en el rango de *0* a *3V*.
- $\boldsymbol{S_2}$ es una señal de salida del sistema empotrado y representa la posición angular de la articulación 2, con un voltaje en el rango de *0* a *3V*.

6.1.1 Acondicionamiento analógico

El acondicionamiento analógico sirve para emparejar el rango de voltaje del ERM2GDL ($0 - 3V$) a del controlador ($\pm 10V$). Se usa un CAS de subida, para ampliar el rango de voltaje y un CAS de bajada, para disminuirlo. Las entradas/salidas (Figura A.33) y rangos de voltaje del CAS analógico son:

- **El CAS de bajada** tiene como entrada un voltaje en el rango de $\pm 10V$ y en la salida tiene un rango de voltaje de $0 - 3V$.
- **El CAS de subida** tiene una entrada de voltaje en el rango de $0 - 3V$ y una salida con voltaje en el rango de $\pm 10V$.

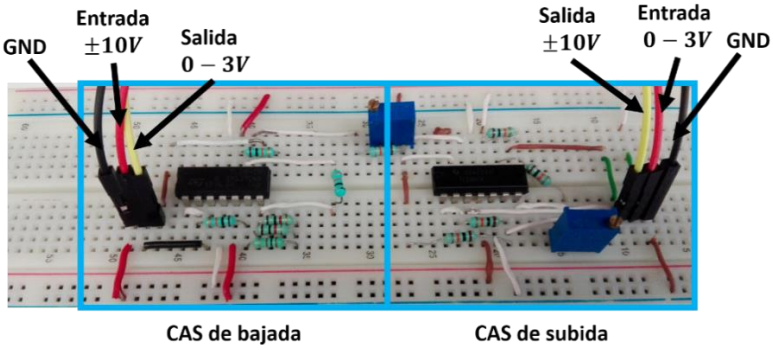


Figura A.33 Circuito acondicionador de señales analógico de bajada y subida

Las conexiones entre el sistema empotrado, acondicionamiento analógico y controlador, se basan en la Figura A.33 y Figura A.34, donde se muestran entradas/salidas del acondicionamiento analógico y el DSC, en los siguientes puntos se explican las conexiones.

- La salida del CAS de bajada ($0 - 3V$) se conecta a la entrada analógica del sistema empotrado ($\boldsymbol{E_1}$ y $\boldsymbol{E_2}$).
- La entrada del CAS de bajada ($\pm 10V$) se conecta a la salida del controlador ($\boldsymbol{\tau_1}$ y $\boldsymbol{\tau_2}$).
- La entrada del CAS de subida ($0 - 3V$) se conecta a la salida analógica del sistema empotrado ($\boldsymbol{S_1}$ y $\boldsymbol{S_2}$).
- La salida del CAS de subida ($\pm 10V$) se conecta a la entrada del controlador ($\boldsymbol{q_1}$ y $\boldsymbol{q_2}$).

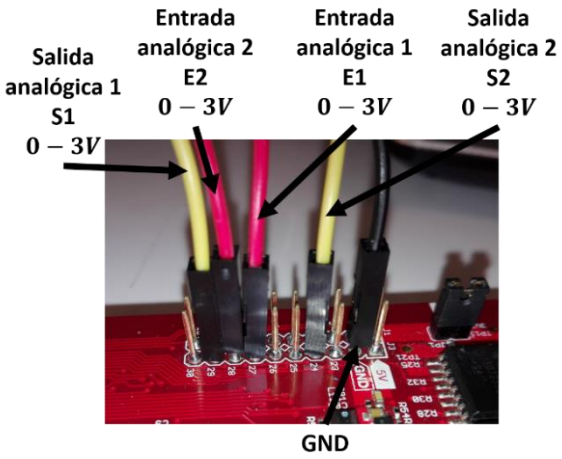


Figura A.34 Entradas/Salidas analógicas

6.1.2 Convertidor de niveles lógicos

El convertidor de niveles lógicos se usa para elevar o disminuir el voltaje máximo de la salida digital. En esta aplicación se usa para elevar el voltaje de $3V$ del DSC a $5V$ para que sea compatible con otras tarjetas. En esta sección se explican las conexiones referentes al convertidor de niveles lógicos para elevar el voltaje de la salida de Encoder.

Las conexiones referentes a la salida de Encoder se basan en la Figura A.35 y Figura A.36. Los pines de la salida del Encoder se muestran en la Figura A.36 donde el rango de voltaje es de $0 - 3V$, el cual se puede elevar al rango de $0 - 5V$ usando el convertidor de niveles lógicos bidireccional BOB-12009 que se muestra en la Figura A.35. El cual se alimenta con ambos niveles de voltaje, es decir, con 3V en LV y 5V en HV referenciados a tierra.

Los pines LVx o HVx (donde x va de 1-4) sirven como entradas/salidas, dependiendo del pin que sea alimentado. Es decir si se requiere que LVx sea la entrada y HVx la salida, se debe alimentar LVx y viceversa. Por ejemplo, para convertir de 3.3V a 5V, se conecta una de las salidas digitales del Encoder de 3.3V al pin LV1 y en el pin HV1 se tienen 5V.

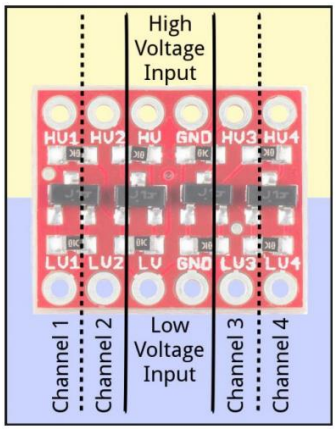


Figura A.35 Convertidor de niveles lógicos BOB-12009

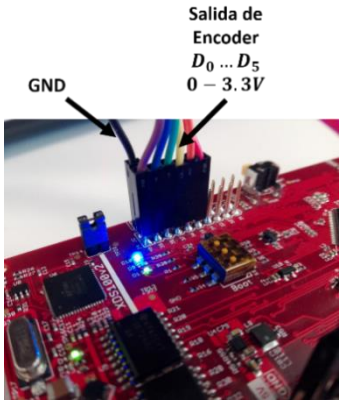


Figura A.36 Salida de Encoder

Apéndice B. Artículo publicado

Sintonizador fuera de línea de un controlador PID discreto usando un algoritmo genético multiobjetivo

David Bedolla-Martínez[1], Esther Lugo González[2],
Felipe Trujillo-Romero[1], F. Hugo Ramírez Leyva[3]

[1] División de Estudios de Posgrado, Universidad Tecnológica de la Mixteca,
Huajuapan de León, Oaxaca,
México

[2] CONACYT-Universidad Tecnológica de la Mixteca, Instituto de Electrónica y Mecatrónica, Universidad Tecnológica de la Mixteca,
Huajuapan de León, Oaxaca,
México

[3] Instituto de Electrónica y Mecatrónica, Universidad Tecnológica de la Mixteca,
Huajuapan de León, Oaxaca,
México

davidbedollamartinez@outlook.es, {elugog, ftrujillo, hugo}@mixteco.utm.mx

Resumen. Este trabajo presenta los resultados obtenidos, al usar un algoritmo genético multiobjetivo, para la sintonización de las ganancias de un controlador PID para el control de posición. Para evaluar este enfoque, la planta a controlar es un Robot Manipulador de 2 Grados de Libertad (GDL), que cuenta con las siguientes características: El modelo dinámico es No-Lineal, es un sistema de múltiples entradas y múltiples salidas (MIMO) y se tiene un acoplamiento en las dinámicas de cada articulación. Como resultado se obtuvo un conjunto de ganancias que estabilizan al sistema con un sobreimpulso y tiempo de establecimiento relativamente bajos.

Palabras clave: PID, algoritmo genético multiobjetivo, robot manipulador, sintonizador fuera de línea.

Offline tuner of a discrete PID controller using a multi-objective genetic algorithm

Abstract. This paper presents the results obtained when using a multi-objective genetic algorithm for tuning the gains of a PID controller for position control. To evaluate this approach, the plant to control is a robot manipulator with 2 degrees of freedom (DOF), which has the following characteristics: The dynamic model is not linear, it is a system of multiple input and multiple output (MIMO) and it has a coupling on the dynamics of each joint. As a result a set of gains that stabilize the system with an overshoot and settling time relatively low was obtained.