Universidad Nacional Autónoma de México

FACULTAD DE INGENIERÍA

Optimización de la implementación de controladores a un equipo neumático

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO ELÉCTRICO ELECTRÓNICO

P R E S E N T A:

LEONARDO RENÉ DE LA CRUZ TRUJILLO

Dirigida por:

M.I. JUAN CARLO RIVERA DUEÑAS

MÉXICO, D.F. **2010**

Jurado:

PRESIDENTE: M.I. Ricardo Garibay Jiménez

VOCAL: M.I. Juan Carlo Rivera Dueñas

SECRETARIO: Dr. Luis Agustín Álvarez Icaza Longoria

1ER. SPTE.: Dr. Marco Antonio Arteaga Pérez

2DO. SPTE.: Dr. Juan Carlos Martínez Rosas

Esta tesis fue elaborada en el Laborotario de Robótica del Posgrado de la Facultad de Ingeniería.

Agradecimientos

A mi familia

Al PROGRAMA UNIVERSITARIO MÉXICO NACIÓN MULTICULTURAL por el apoyo a lo largo de la licenciatura

A Juan Carlo al Dr. Marco Arteaga y a Jens por el apoyo y la confianza en la realización de este trabajo

A los compañeros del Laboratorio de Robótica que me apoyaron con las dudas surgidas en el día a día.

Índice general

1. Introducción **7**
- 1.1. Estado del arte . . . 7
- 1.2. Conceptos Fundamentales . . . 8
- 1.3. Objetivos del Trabajo . . . 14

2. Problemática de la optimización del sistema **15**
- 2.1. Rediseño e ingeniería inversa . . . 15
- 2.2. Expectativas de adaptación empleando cRIO-MATLAB, cRIO-CVI . . . 16
- 2.3. Problemática de adaptación y metodología de solución . . . 17

3. Adaptación de un sistema neumático con cRIO **21**
- 3.1. Identificación de señales y características del sistema neumático . . . 21
- 3.2. Descripción del sistema cRIO y módulos empleados . . . 25
- 3.3. Acoplamiento de señales para el sistema cRIO . . . 27

4. Análisis de Controladores **31**
- 4.1. Generalidades del control de trayectorias . . . 31
- 4.2. Métodos para el diseño de controladores no lineales . . . 34
- 4.3. Modelo dinámico . . . 35
- 4.4. Control PID . . . 37
- 4.5. Control por Modos deslizantes . . . 38
- 4.6. Control PID no lineal . . . 42

5. Implementación de los Controladores **51**
- 5.1. CVI y programación entrada/salida . . . 51
- 5.2. LabView 2009 y programas de entrada/salida . . . 54
- 5.3. Interfaz para el control del sistema neumático . . . 57
- 5.4. Resultados experimentales . . . 59

6. Conclusiones **69**

A. CAN y CANopen **71**
- 1.1. CAN . . . 71
- 1.2. CANopen . . . 72

B. Especificaciones del hardware **77**

C. Glosario **83**

Capítulo 1

Introducción

La optimización de procesos es uno de los objetivos principales en la industria. El Control de Sistemas puede ser interpretado como una especialización de tal proceso, puesto que el mejoramiento de los procesos implica muchas veces la optimización de los métodos de control. El control ha sido vital en el avance de la tecnología y la ciencia, además es uno de los indicadores con el cual es posible observar la evolución de la Ingeniería, específicamente en Ingeniería Eléctrica.

Los sistemas físicos ya sean mecánicos, térmicos, hidráulicos, etc, están acompañados por elementos de control y monitoreo que permiten un análisis posterior del comportamiento del sistema. Uno de los sistemas que ha sido utilizado desde la antigüedad es el Neumático cuyo origen se remonta a los trabajos de Herón de Alejandría. La neumática es el uso de gas comprimido para generar movimiento mecánico. La neumática hoy en día es una de las tecnologías más utilizadas en la industria para aplicar fuerza, mover máquinas o herramientas. En el área de la robótica se encuentran muchas herramientas neumáticas, sobre todo en aplicaciones de manipulación de objetos y en pinzas para soldadura.

Este trabajo se desarrolla sobre un sistema neumático y la implementación de controladores, la optimización de la programación del sistema y el mejoramiento del tiempo de muestreo de tal forma que sea posible realizar pruebas de controladores no lineales. Se plantea el problema de optimización de la adquisición de señales, el mejoramiento del software de control y la utilización de conceptos fundamentales de la Teoría de Control así como la Teoría de Sistemas y Señales.

1.1. Estado del arte

Los sistemas neumáticos son controlados en la industria generalmente por sistemas que integran Controladores de Lógica Programable (PLC's por sus siglas en Inglés), válvulas, sensores de presión y en ocasiones sensores de posición basados en Transformadores diferenciales de variación lineal (LVDT's por sus siglas es inglés). Otros dispositivos más recientes utilizan microcontroladores diseñados específicamente para leer la posición del pistón y controlar únicamente el desplazamiento del pistón. Además por la facilidad de manipulación

y por el costo se utilizan válvulas on-off por lo que el pistón tiene dos estados que pueden ser manipulados con señales digitales para realizar una gran variedad de procesos.

En el campo de la investigación el estudio se ha dividido en tres grupos. El primero trata del desarrollo de un modelo dinámico que considera únicamente al cilindro neumático, un modelo dinámico para las diferentes válvulas y otros modelos aún más complejos consideran la dinámica de la tubería. El segundo grupo trata de los controladores para el desplazamiento y/o fuerza. En este grupo están los controladores PID, linealización exacta, entre otros. El último grupo trata el análisis de observabilidad de dichos sistemas.

El control avanzado de los sistemas neumáticos implica la obtención de señales de presión y desplazamiento. Para obtener la posición del pistón se usan encoders o bien, sensores de presión y desplazamiento internos. La utilización de sensores basados en el protocolo de comunicación CANopen es algo usual en la industria automotriz debido a que es posible obtener además de la posición, la velocidad y la aceleración con un sólo sensor. El uso de PAC's[1] (Controladores de Automatización Programable, por sus siglas en Inglés) para la adquisición y control de las señales provenientes de sistemas neumáticos, es un campo poco desarrollado.

1.2. Conceptos Fundamentales

1.2.1. Definición de un sistema de adquisición de datos DAQ

La adquisición de datos (DAQ por sus siglas en inglés) es el proceso por el cual un fenómeno físico es transformado en señales eléctricas, las cuales son medidas y convertidas a un formato digital para el procesamiento, análisis y almacenamiento por una computadora [Park and Mackay, 2003].

En la mayoría de las aplicaciones, el sistema adquisición de datos (DAQ) es diseñado no sólo para adquirir datos sino también para actuar en el sistema mismo. Cuando se define un *Sistema de Adquisición* es usual extender la definición para incluir aspectos de control del sistema. Control, en el ámbito de los sistemas DAQ, es el proceso por el cual las señales digitales de control provenientes de los sensores son convertidas a señales analógicas para que éstas sean usadas por dispositivos actuadores o relevadores. Se dice que estos dispositivos *controlan* el proceso o el sistema.

1.2.2. Conceptos fundamentales de adquisición

Un sistema de adquisición y control puede construirse a partir de la versatilidad de la PC y depende de las necesidades específicas de nuestro sistema y de los objetivos planteados para su desarrollo. Los elementos básicos de sistema de adquisición de datos son los siguientes:

[1] Como apéndice se agrega un glosario de los términos más usados a lo largo de este trabajo

1.2.2.1. Sensores y comunicación

Sensores y transductores son la interfaz entre el mundo real y el sistema de adquisición ya que convierten el fenómeno físico en señales eléctricas que el hardware puede interpretar. Los transductores pueden llevar a cabo casi cualquier tipo de medición y proveer la correspondiente señal eléctrica, como ejemplo de sensores transductores están: termopares, detectores resistivos de temperatura (RTD's), termistores, etc.

El cableado de comunicación representa la conexión física desde los sensores y transductores hacia el hardware de acondicionamiento de señal o al hardware de adquisición. Cuando alguno de éstos elementos está localizado de forma remota con respecto a la PC, entonces el cableado de campo provee el enlace entre los elementos de hardware y la PC. Si el enlace físico es una interfaz RS-232 o bien, RS-485 entonces el cableado de campo es denominado cable de comunicación puesto que su configuración requiere la utilización de un protocolo como es el caso de CANopen .

Puesto que el cableado de campo y el cable de comunicación a menudo representan la parte más extensa del total del sistema, éste es más susceptible a los efectos de ruido externo, especialmente en ambientes industriales. Este fenómeno puede generar errores de precisión en las mediciones que se obtienen del sistema por lo cual es necesario acondicionar la señal proveniente de los sensores.

1.2.2.2. Acondicionamiento de señal

Las señales eléctricas generadas por los transductores a menudo deben ser convertidas a una forma aceptable para el hardware de adquisición, particularmente la conversión a un formato digital.

Además muchos transductores requieren alguna forma de excitación para una operación apropiada y precisa. Algunas formas de acondicionamiento de señal son:

1. Filtrado
2. Amplificación
3. Linealización
4. Aislamiento
5. Excitación

1.2.2.3. Hardware de adquisición

El hardware de adquisición y control puede ser definido como el componente del sistema de adquisición y control que lleva a cabo las siguientes funciones:

1. La conversión y procesamiento de señales en un formato digital utilizando ADC's (Convertidor Analógico-Digital por sus siglas en inglés), provenientes de entradas analógicas del sistema o proceso, los datos son de esta forma transmitidos a la computadora para el análisis, el almacenamiento y presentación.

2. La entrada de señales digitales, que contienen información del sistema o el proceso.

3. La conversión y procesamiento de señales digitales a analógicas por medio de DAC's (Convertidor Digital-Analógico por sus siglas en inglés) provenientes de la computadora que son utilizadas para *controlar* el proceso o sistema.

1.2.2.4. Software de adquisición

El software de adquisición ubicado por lo general en una PC es el último elemento necesario para la manipulación de la información proveniente de los sensores; dicho software nos permite el análisis, control y presentación de datos.

El software de aplicación corre en la PC bajo un sistema operativo que puede ser uni-tarea (single-tasking) como DOS, multi-tarea (multitasking) como LINUX, WINDOWS, UNIX, etc, que nos permite ejecutar mas de una aplicación de forma simultánea.

El software de aplicación puede ser un panel interactivo, un programa de control de entradas y salidas, entrada de datos, un control de comunicación o una combinación de éstos.

1.2.3. Controladores de automatización programable (PAC)

Los sistemas de control tipo DAQ están basados en la PC puesto que dependen de la versatilidad del software que se utiliza para realizar la adquisición y la compatibilidad del hardware con la PC. A la par de los sistemas DAQ están los sistemas basados en PLC's que son los sistemas más usados en la industria para el control de procesos y máquinas.

Los Controladores de Automatización Programables (PAC's por sus siglas en Inglés) combinan la funcionalidad de una PC y la confiabilidad de un PLC y resuelve la mayoría de los problemas que se relacionan con el uso de los PLC's y las PC's [OPTO22, 2006].

Un PAC, de acuerdo a ARC Advisory Group (los que introdujeron el término), es un Controlador de Automatización Programable que debe satisfacer los siguiente requerimientos:

- Funciona en una plataforma sencilla en múltiples dominios que incluyen movimiento, controladores, procesos de control y lógica de control.

- Utiliza una plataforma de desarrollo sencilla utilizando etiquetado y bases de datos para tareas de desarrollo que incluya varias disciplinas.

- Una integración fuerte de hardware y software.

- Debe ser programable usando herramientas que puedan diseñar programas de control que admitan procesos multiplataforma.

- Utiliza estándares de facto para interfaces de red, lenguajes y protocolos permitiendo intercambio de datos como parte de una red de sistemas de diferentes fabricantes.

- Provee un procesamiento eficiente y escaneo de entrada/salida (I/O).

El sistema que es objetivo de esta trabajo puede ser definido como un sistema DAQ o PAC puesto que cumple con las características de ambos modelos. **Puesto que el término PAC es relativamente nuevo y está asociado a dispositivos como el cRIO es posible denominar a nuestro sistema PAC.**

1.2.4. Modelo OSI de redes

Los sistema de adquisición y control no son bidireccionales. En la industria es común monitorear varios sistemas a la vez, cada uno de los cuales es un sistema de adquisición y control por sí mismo, lo que hace necesaria una administración del sistema completo. Para acceder a muchos dispositivos a la vez se utilizan redes de comunicaciones o simplemente redes.

Una red de comunicaciones es un conjunto de medios que permite la comunicación entre equipos autónomos para transmitir datos por ondas electromagnéticas a través de diversos medios (aire, vacío, cable de cobre, cable de fibra óptica, etc.). Debido a que los medios de transmisión son variados, así como los dispositivos que se conectan entre sí, se definió el modelo OSI de redes en 1984.

El modelo OSI define un estándar de comunicaciones entre programas de distintos equipos. El modelo OSI divide en siete capas el proceso de transmisión de la información entre equipos informáticos, donde cada capa se encarga de ejecutar una determinada parte del proceso global. La Figura 1.1 presenta la estructura de capas que conforman el modelo OSI en forma de pirámide que ilustra la estructura del modelo, en el que los datos con un formato bastante complejo pasan a convertirse en una secuencia simple de bits cuando alcancen el cable de red [Tanenbaum, 2003].

Las capas del modelo OSI describen el proceso de transmisión de los datos dentro de una red. Las dos únicas capas del modelo con las que, de hecho, interactúa el usuario son la capa Física (primera capa) y la capa de Aplicación (la última capa).

- La **capa física** abarca los aspectos físicos de la red (es decir, los cables, hubs y el resto de dispositivos que conforman el entorno físico de la red).
- La **capa de aplicación** proporciona la interfaz que utiliza el usuario en su computadora para enviar mensajes de correo electrónico o ubicar un archivo en la red.

1.2.4.1. La capa de aplicación

La capa de aplicación proporciona la interfaz y servicios que soportan las aplicaciones de usuario. También se encarga de ofrecer acceso general a la red. Esta capa proporciona las herramientas para el usuario y los servicios de red relacionado con las aplicaciones de usuario, como la gestión de mensajes, la transferencia de archivos y las consultas de las bases de datos. En la figura 1.2 muestra el flujo de datos desde la computadora receptora a la computadora emisora y viceversa.

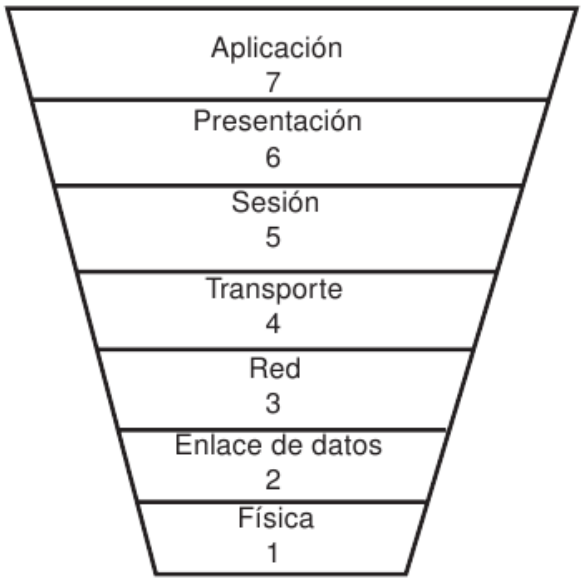


Figura 1.1: Modelo OSI que explica el modo en que se desplazan los datos de un dispositivo a otro.

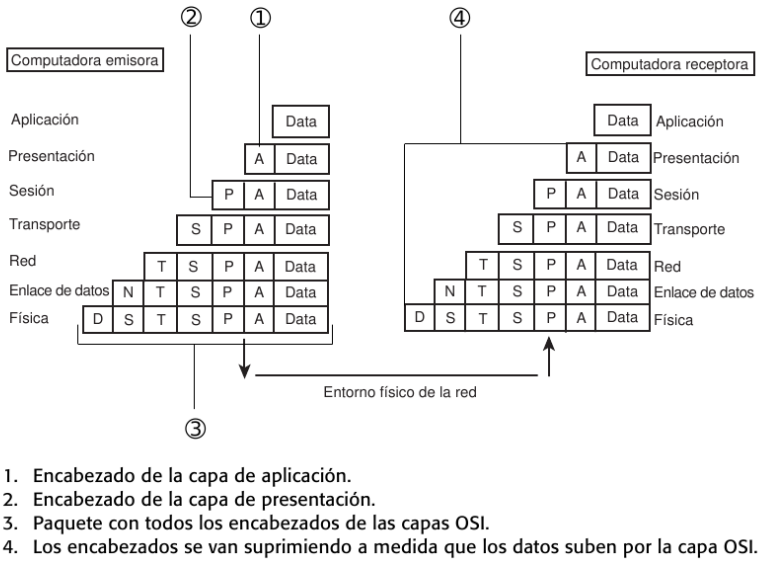


Figura 1.2: Los datos bajan por la pila de la computadora emisora y suben por la pila de la receptora

1.2.4.2. La capa de presentación

El nivel de presentación está relacionado con la sintaxis y la semántica de la información. Las funciones realizadas por este nivel son:

Traducción. Debido a que cada dispositivo usa un sistema de codificación distinto, el nivel de presentación es responsable de la interoperabilidad entre los distintos métodos de codificación.

Cifrado. Para transportar información sensible el sistema debe ser capaz de asegurar la privacidad.

Autenticación. La verificación de la identidad del emisor.

Compresión. La compresión reduce el número de bits que se deben transmitir.

1.2.4.3. La capa de sesión

Permite a los usuarios sesionar entre sí, permitiendo acceder a un sistema de tiempo compartido a distancia o transferir un archivo entre dos maquinas.

Uno de los servicios de esta capa es la del seguimiento de turnos en el tráfico de información, como así también la administración de tareas, sobre todo para los protocolos.

Otra tarea de esta capa es la de sincronización de operaciones con los tiempos de caída en la red.

1.2.4.4. La capa de transporte

La función principal es de aceptar los datos de la capa superior y dividirlos en unidades más pequeñas, para pasarlos a la capa de red, asegurando que todos los segmentos lleguen correctamente, esto debe ser independiente del hardware en el que se encuentre.

1.2.4.5. La capa de red

Es responsable de la entrega de un paquete desde el origen al destino, y posiblemente, a través de múltiples redes (enlaces).

1.2.4.6. La capa de enlace de datos

Transforma el nivel físico, un simple medio de transmisión, en un enlace fiable y es responsable de la entrega nodo a nodo. Hace que el nivel físico aparezca ante el nivel superior (nivel de red) como un medio libre de errores.

1.2.4.7. La capa física

Coordina las funciones necesarias para transmitir un flujo de datos a través de un medio físico. Trata con las especificaciones eléctricas y mecánicas de la interfaz y del medio de transmisión. El nivel físico se relaciona con lo siguiente:

- Características físicas de las interfaces y el medio de transmisión.
- Representación de los bits.
- Tasa de transmisión de los datos.
- Sincronización de los bits.
- Configuración de la linea.
- Modo de transmisión

1.3. Objetivos del Trabajo

El objetivo principal de este trabajo es realizar modificaciones en el último nivel del sistema de adquisición y control, mejorar el esquema de comunicación y desarrollar una aplicación que permita implementar diversos controladores.

Debido a que el último nivel es el nivel de software es preciso analizar la viabilidad de lenguajes de programación existentes y la compatibilidad de éstos con en el sistema físico. Una vez elegido el software conveniente para el sistema es preciso realizar experimentos que comprueben la viabilidad del software en conjunto con el equipo y si éste obtuvo el rendimiento esperado.

Además se espera minimizar el tiempo de muestreo actual del sistema en función de las posibilidades del equipo actual para que puedan probarse algoritmos de control complejos; además mejorar la manera en que se maneja la información.

Los objetivos específicos del trabajo se presentan a continuación:

1. Mejorar la programación para que el programa desarrollado sea capaz de almacenar de forma eficiente los datos para el análisis posterior del funcionamiento de los controladores.
2. Optimizar el uso del sensor de posición que actualmente utiliza una tasa de transferencia de datos de 500Kbaud a una tasa de transferencia de 1Mbaud.
3. Mejorar la programación para que permita la implementación de esquemas de control diferentes de tipo no-lineal como modos deslizantes, adaptables, pasivos, robustos, óptimos, etc.
4. Disminuir el tiempo de muestreo.
5. Utilizar, si es posible, una interfaz distinta a LabVIEW , como CVI, MATLAB o C++.

Capítulo 2

Problemática de la optimización del sistema

2.1. Rediseño e ingeniería inversa

El sistema neumático que se encuentra instalado en el Laboratorio de Robótica del edificio de posgrado de la Facultad de Ingeniería está diseñado en función de las necesidades de los proyectos de investigación que ahí se realizan. El sistema consta de un dispositivo de control programable denominado CompactRIO (cRIO) y la programación en el dispositivo es lo que se debe diseñar y optimizar. Dicho sistema ha sido utilizado para realizar algunas pruebas de controles PID, sin embargo el tiempo de muestreo ha sido muy alto para implementar otros controles; además, controladores mas sofisticados no son soportados por la estructura del sistema de adquisición.

2.1.1. Breve descripción del programa base

El programa existente que realiza la adquisición es limitado, sin embargo su análisis es necesario para mejorarlo y evitar repetir errores que puedan ser imputados al esquema de programación. Algunas de las características del programa son las siguientes:

- El protocolo CANopen se implementa en dos niveles de programación, el nivel del FPGA y el nivel de microprocesador.
- La adquisición de las señales de salida y entrada se realizan en el nivel FPGA.
- El tiempo de muestreo en el nivel del FPGA es de 500 [μs].
- La tasa de transferencia del sensor de posición es de 500 Kbaud.

En la Figura 2.1 se muestra el panel del programa que implementa únicamente un control PID sin almacenamiento de datos, con algunas opciones de modificación en los parámetros comunes en un control PID.

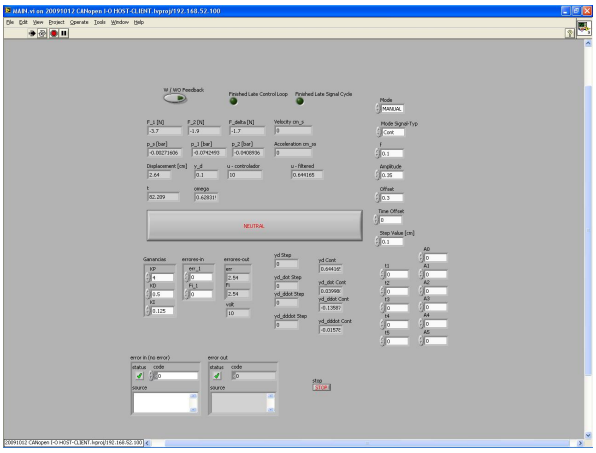


Figura 2.1: Programa implementado en LabVIEW

La versatilidad del programa es limitada lo que plantea un diseño nuevo casi en su totalidad, aunque algunas características se preservarán, puesto que son requerimientos del sistema de control:

1. La posibilidad de establecer una posición inicial.
2. La opción de establecer las constantes de control.
3. La posibilidad de modificar la señal utilizada para el seguimiento y distintas frecuencias.

2.2. Expectativas de adaptación empleando cRIO-MATLAB, cRIO-CVI

El hecho de que las modificaciones que se deben hacer son de software, sugiere que es posible utilizar otros lenguajes de programación debido a que los problemas que presenta la implementación del sistema se asocian al uso de LabVIEW como entorno de programación.

2.2.1. Compact RIO - MATLAB

MATLAB es un software matemático que nos permite realizar operaciones analíticas y que integra herramientas de simulación, a saber, SIMULINK y herramientas para realizar alguna interfaz gráfica . Es posible además utilizar el lenguaje de programación C, C++ en MATLAB. El desarrollo de una aplicación utilizando únicamente MATLAB implica la existencia de controladores para el dispositivo que deseamos utilizar (cRIO), que actualmente no existen.

Sin embargo ésta opción sería la ideal puesto que los algoritmos de control se simulan con esta herramienta y lo ideal es utilizar el algoritmo simulado en MATLAB directamente

en el sistema. El desarrollo de un controlador compatible con el sistema cRIO se plantea como una solución viable, pero en la practica ésto implicaría el conocimiento del sistema operativo Vx Works para sistemas embebidos y el funcionamiento del controlador privativo de National Instruments o bien, generar un entorno de programación completamente nuevo que utilice bibliotecas dinámicas que compartan información entre MATLAB y LabVIEW, lo cual aumentaría el tiempo de muestreo del sistema.

2.2.2. cRIO - LabwindowsCVI

NI Labwindows/CVI (CVI) es un entorno de programación que nos permite realizar programas utilizando el lenguaje C ya que es posible utilizar las herramientas de desarrollo para instrumentación y control, análisis de datos y desarrollo de interfaces para el usuario. Debido a que esta herramienta es de National Instruments, se planteó como el primer acercamiento a la programación que emplea herramientas convencionales de programación.

Esta opción representa el paso natural puesto que es la herramienta proporcionada por el fabricante del dispositivo del cual se quiere obtener el máximo rendimiento. Como se verá adelante esta opción es llevada hasta la implementación de un control PID.

Una vez planteado lo anterior es posible simplificar las expectativas de programación a: **si es posible desarrollar una aplicación utilizando CVI pues así la migración al lenguaje C++ es casi inmediata**. Esto es debido a la experiencia que se tiene en el Laboratorio de Robótica trabajando con dicho software.

2.3. Problemática de adaptación y metodología de solución

El algoritmo de programación que se utiliza en el programa base es limitado como se explicó anteriormente, por lo cual es necesario realizar un diseño completamente nuevo que disminuya el tiempo de muestreo y la versatilidad del programa. Las únicas características que se preservarán corresponden a la interfaz de usuario, es decir, variables que son manipuladas por el usuario final.

Con base en lo anterior se plantea un nuevo diseño que debe ser independiente del lenguaje de programación que se utilice, puesto que el dispositivo que se utilizará para realizar el control es el cRIO y que como se explicará posteriormente ofrece tres niveles de programación, los cuales son: Nivel FPGA, Nivel de microprocesador y Nivel host (PC). Esta versatilidad nos permite realizar programas que se ejecutan en tiempo real en los primeros dos niveles de programación. En la Figura 2.2 se muestran los niveles de programación que se utilizarán para el diseño del programa.

La “velocidad” de procesamiento en cada uno de éstos niveles es distinta. El nivel más rápido corresponde al del FPGA, en segundo lugar el Nivel de microprocesador y el más lento es el nivel de host (PC). Con base en lo anterior es necesario plantear el modelo que se utilizará para aprovechar las características del equipo con respecto a la velocidad de procesamiento y las necesidades específicas del sistema de adquisición y control.

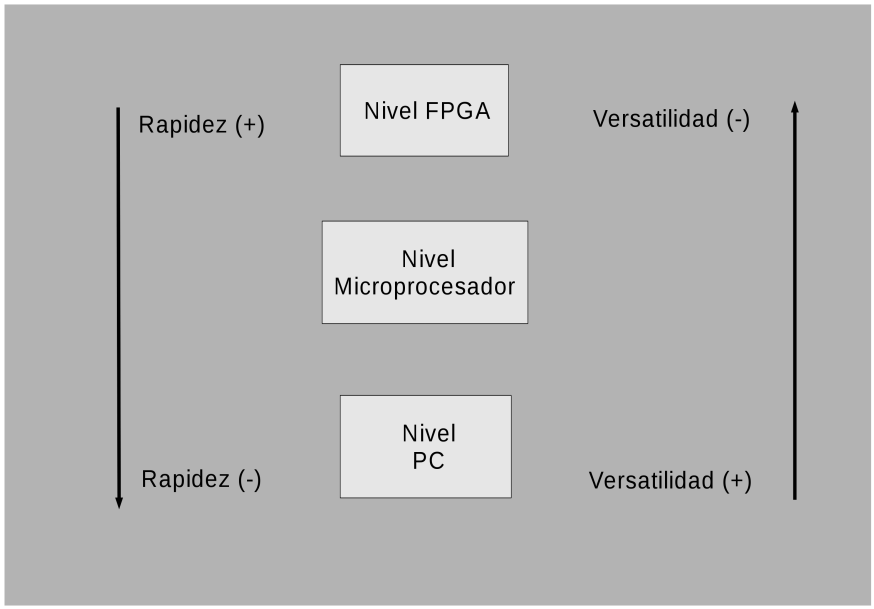


Figura 2.2: Niveles de programación

Además de la rapidez en la cuál se realizará el proceso, es necesario plantear la relación entre las etapas de nuestro sistema de adquisición y control y el nivel de programación que administrará cada etapa, es decir, se debe definir en qué nivel se realizará el control, la adquisición de las señales, y las operaciones relacionadas con ellas incluyendo el almacenamiento de datos tomando en cuenta que algunos lenguajes de programación no realizan procesos paralelos.

2.3.1. Paradigmas de programación

Las posibilidades de programación que se tienen plantean dos paradigmas de programación, a saber, programación estructurada (CVI) y procesos paralelos (LabVIEW) lo que puede generar dos posibilidades de implementación. La programación estructurada, que únicamente realiza una operación por ciclo, y la programación paralela que ejecuta varias operaciones en un mismo ciclo. El programa base utiliza el segundo paradigma pero los algoritmos de control son estructurados, además basados en experiencias anteriores se consideró que podría tener mejores resultados. por lo cual el diseño se realizará utilizando el primer paradigma y deberá también ser independiente del lenguaje utilizado, es decir, se buscará que los procesos se realicen en forma secuencial.

El diagrama de flujo que se muestra en la Figura 2.3 representa los procesos que ejecutará el programa diseñado y la prioridad de éstos. Estos procesos están divididos en cinco partes:

Adquisición. En esta parte del programa se obtendrán los parámetros que se necesitan para la ejecución del control.

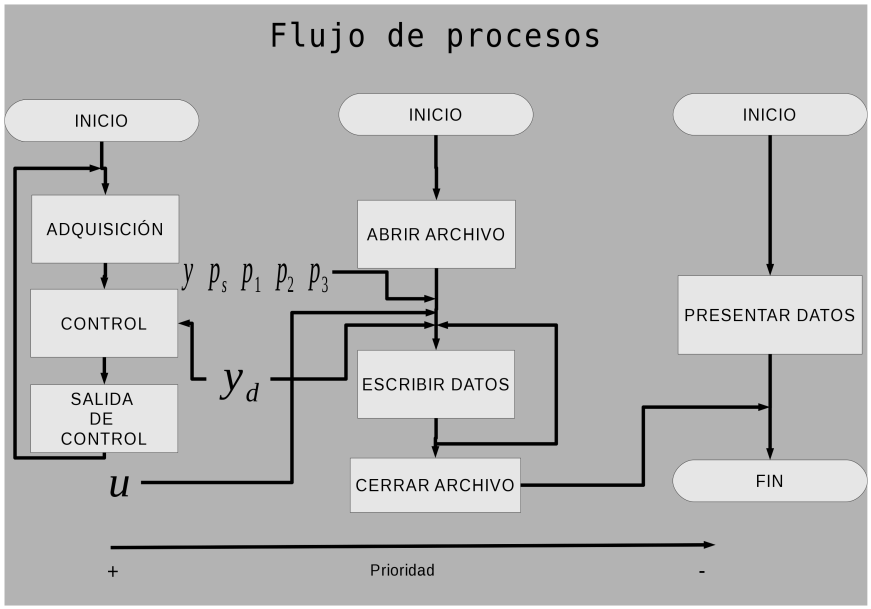


Figura 2.3: Diagrama de flujo

Control. En esta etapa se realizan las operaciones matemáticas para cada algoritmo de programación.

Salida de control. En esta etapa se acondiciona el voltaje que realiza el control.

Almacenamiento de datos. En esta etapa se deben almacenar los datos que se hayan utilizado para la realización del control.

Presentación de datos. Esta etapa muestra los datos en la PC para el monitoreo en tiempo real del proceso.

Cada una de estas etapas debe tener un prioridad para mejorar la eficiencia de los procesos. Por ejemplo, el proceso de almacenamiento tiene una prioridad secundaria a la realización del control. La prioridad en los algoritmos de programación implica la existencia de ciclos determinísticos (con un tiempo de muestreo fijo) y ciclos deterministicos pero de baja prioridad. El proceso del control y adquisición son determinísticos mientras que el almacenamiento de datos y la presentación de datos son determinísticos con menos prioridad.

2.3.2. Planteamiento de solución

Con base en los factores que se han mencionado es posible plantear los pasos necesarios para la realización del programa objetivo que cumpla con todos los requerimientos en orden de prioridad. La Figura 2.4 muestra el diagrama de flujo para la solución del problema de programación.

En el diagrama se observan las siguientes etapas:

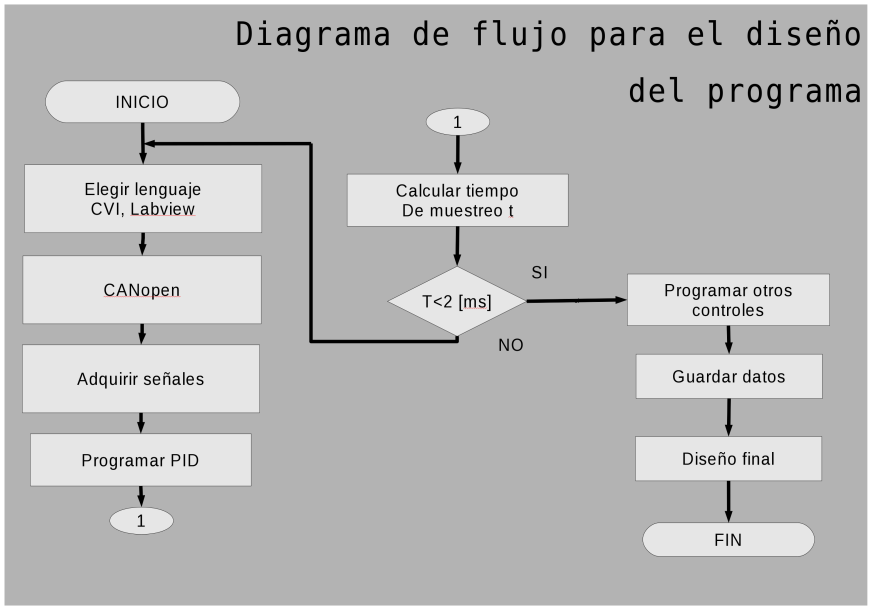


Figura 2.4: Diagrama de flujo para la realización del programa

1. Realizar la programación del protocolo CANopen para la obtención del desplazamiento.
2. Realizar la adquisición de las señales del sistema con el menor tiempo de muestreo posible utilizando alguno delos lenguajes de programación propuesto.
3. Realizar la medición del tiempo de muestreo.
4. Programar los algoritmos de programación que se utilizarán en el sistema.
5. Realizar la programación del almacenamiento de datos.
6. Diseñar una interfaz de usuario.
7. Si el desempeño no es adecuado utilizar otro lenguaje de programación.

Capítulo 3

Adaptación de un sistema neumático con cRIO

El sistema instalado se puede dividir en dos grupos. El primer grupo corresponde al sistema físico que está constituido por el pistón, las electroválvulas y los sensores de presión y de posición. El segundo constituido por el sistema cRIO en conjunto con la alimentación y los módulos adicionales que permiten obtener los voltajes de los sensores de presión y la comunicación con el sensor de posición (CANopen).

De acuerdo al modelo de un sistema DAQ el primer grupo que definimos corresponde a las dos primeras fases en la etapa de adquisición, es decir, la correspondiente al sistema físico (sistema neumático) y a las etapas del sensor y transductor. El segundo grupo corresponde al hardware de adquisición y al acondicionamiento de las señales de los sensores y el cableado utilizado. Si nos basamos en el modelo PAC entonces el cRIO en conjunto con los módulos es el PAC, mientras que la PC corresponde a la interfaz del operador y el pistón y la válvula corresponden a las entradas y salidas analógicas.

3.1. Identificación de señales y características del sistema neumático

Las señales necesarias para la implementación de los controladores son: la presión de cada una de las cámaras del pistón (p_1 y p_2), la presión de alimentación o suministro (p_s), el voltaje de control que será aplicado a la válvula o señal de control (u) y el desplazamiento (y). Estas señales serán acondicionadas por medio de los módulos de salida y entrada analógicos que a su vez están conectados al cRIO .

El sistema neumático se observa en la Figura 3.1. En la Figura se muestra que está conformado por un cilindro (a), los sensores de presión (b) para p_1 y (c) para p_2, el sensor de posición (d) y la electroválvula (e).

El pistón es un modelo Telemeqanique PAEA5280760. Cuenta con dos cámaras divididas por la base del pistón y su vástago (Ver Figura 3.2). La base del pistón mide 80mm de diámetro y se puede desplazar hasta 760mm.

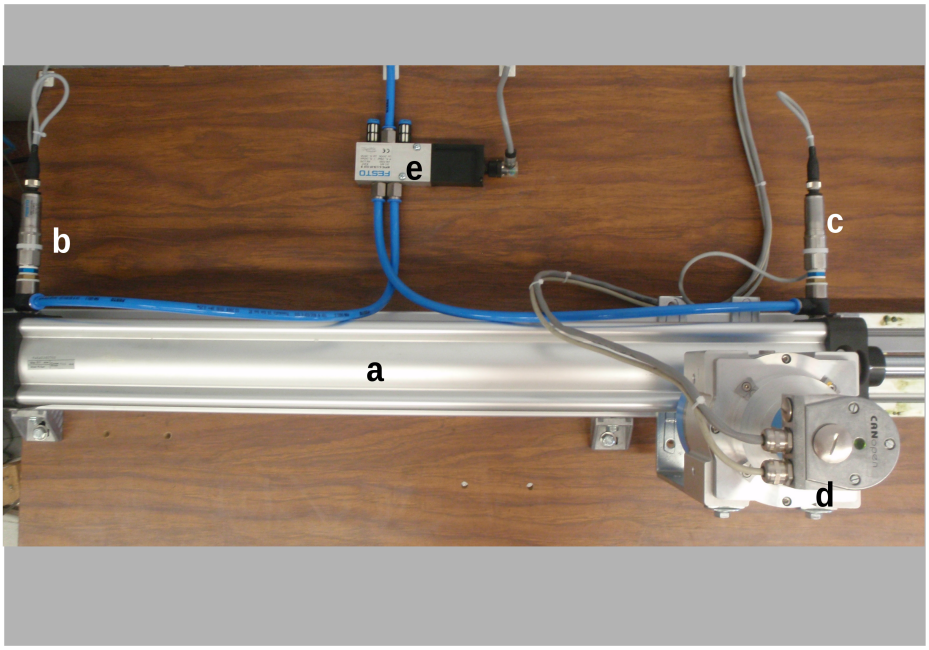


Figura 3.1: Sistema neumático

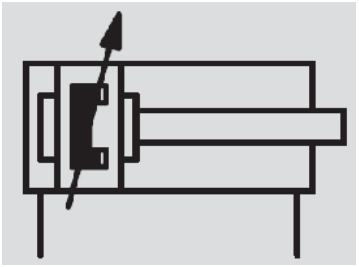


Figura 3.2: Símbolo de un pistón diferencial con vástago y amortiguamiento

El sistema completo se muestra en la Figura 3.3. Además del pistón se muestra la unidad de mantenimiento (a) en conjunto con el sensor de la presión de suministro (b) y el cRIO con los módulos empleados (c).

3.1.1. Electroválvula

La electroválvula es de tipo proporcional 5/3 Festo MPYE-5-$^{1}/_{8}$HF-010-B, que propone un flujo de aire máximo [1] de $700\frac{l}{\text{min}}$, controlada por un voltaje en la entrada de 0[V] a 10[V] (ver apéndice) . La válvula tiene una entrada (a), cuatro salidas (escapes al ambiente (b), (c) y hacia las cámaras 1 (d) y 2 (e) del cilindro) y el spool (f) se mueve entre 3 posiciones claves (ver Figura 3.4). Si la entrada (g) recibe 0[V], el spool conecta el suministro de aire con la

[1] con una presión de suministro de 10 [bar]

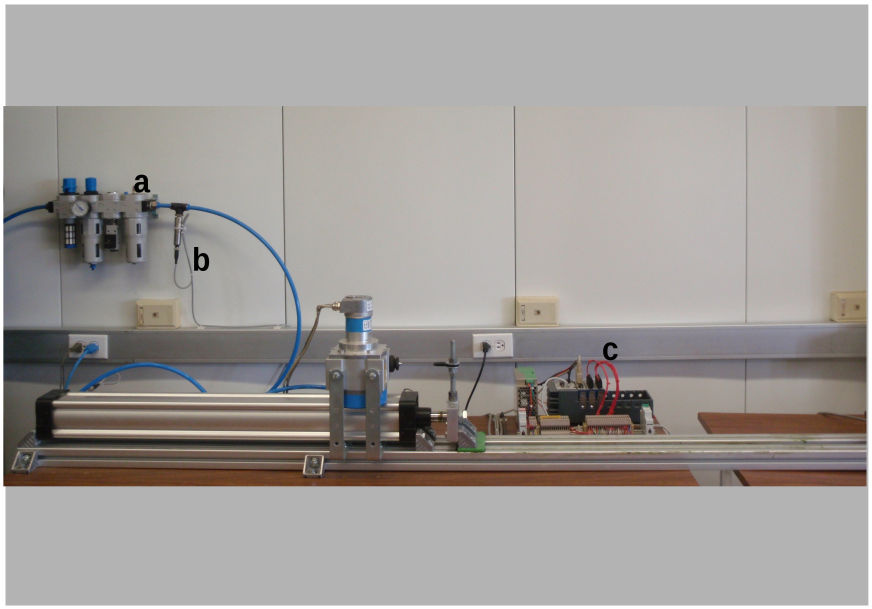


Figura 3.3: Sistema completo en conjunto con cRIO

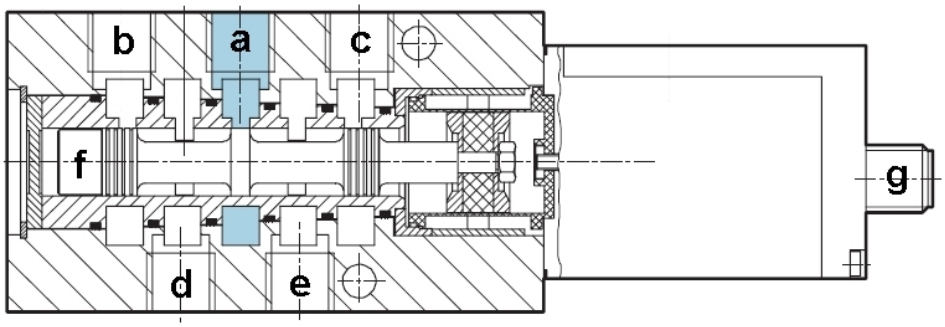


Figura 3.4: Corte transversal de una válvula proporcional

cámara 1 del pistón a su máximo y deja escapar el volumen de la cámara 2 a su máximo, o bien, lo iguala a la presión atmosférica p_{atm}. Lo contrario pasa para 10[V] en la entrada. La posición neutral del spool se encuentra aplicando 5[V] en la entrada. En la Figura (3.5) se observa el símbolo de la válvula.

Para la medición de la presión del suministro p_{s} y las dos presiones de las cámaras p_1 y p_2 del pistón, se usa un sensor Festo SDET-22T-D16-G14-U-M12, que tiene un rango de medición de la presión de 0 a 16 bares, y dos del tipo SDET-22T-D10-G14-U-M12, que tienen un rango de medición de la presión de 0 a 10 bares. Los tres sensores tienen una exactitud de 1 % del valor absoluto máximo y entregan la señal lineal de voltaje de 0.1[V] a 10[V]. Como estos sensores trabajan en forma analógica se puede despreciar el desfase hasta que sea medible la señal en forma de un voltaje en la salida (ver apéndice).

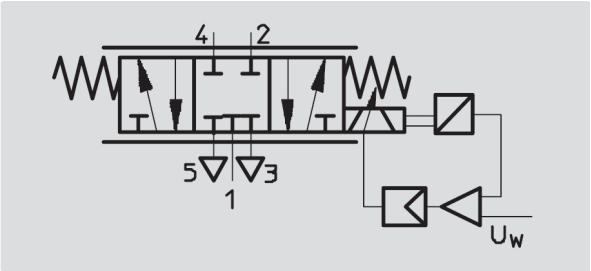


Figura 3.5: Símbolo de una válvula proporcional

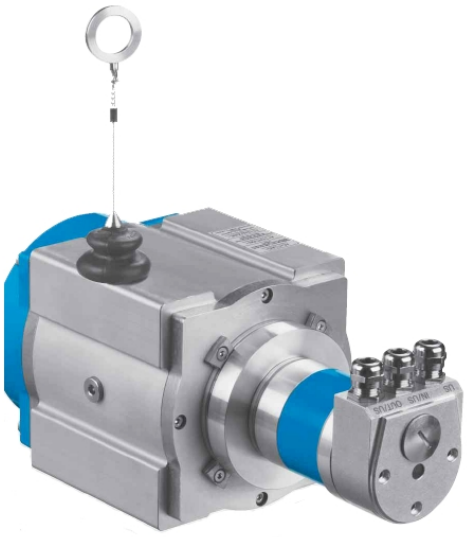


Figura 3.6: Sensor de desplazamiento Sick/Stegmann

3.1.2. Sensor de posición

Para la medición del desplazamiento de la biela se usa un encoder de hilo absoluto del tipo Sick/Stegmann BTF08-C1HM0241 (Figura 3.6). Dicho sensor transmite utilizando el protocolo CAN y tiene una precisión de 0.025 [mm]. El comportamiento es lineal (0.05 %) y la electrónica interna tarda 0.15 [ms] para obtener la posición (ver apéndice).

3.1.3. Comunicación

Para la comunicación de los dispositivos se utilizaron distintos tipos de cableado de comunicación y de alimentación. A continuación se describen brevemente las conexiones entre los dispositivos y el cableado utilizado.

PC-cRIO: Esta conexión permite el desarrollo de la aplicación de control y la descarga del programa a el dispositivo objetivo que es el cRIO y una transmisión de datos para el almacenamiento y la interacción con el usuario. Esta conexión se realiza vía cable ethernet.

cRIO-Módulos: Esta conexión es directa ya que los módulos tienen adaptadores que se "sincronizan" para la comunicación con el cRIO .

Modulo NI 9853-CANopen: Esta conexión nos permite conectar el módulo de cRIO y el sensor de posición Sick/Stegmann. La conexión se realiza vía cable serial.

E/S: Los sensores de presión y la electroválvula se conectan directamente a los módulos E/S del cRIO y a una fuente de alimentación externa de 18 [V].

Conexión neumática: Estas conexiones consisten en mangueras que están conectadas a cada una de las cámaras del pistón y a la estación de control neumática, así como a la electroválvula.

3.2. Descripción del sistema cRIO y módulos empleados

El sistema de adquisición está conformado por un cRIO 9014 de National Instruments que es un controlador en tiempo real, los módulos NI 9853, NI 9215 y NI 9263 que nos permiten utilizar el protocolo CANopen, leer los voltajes de los sensores y "escribir" voltajes a la electroválvula, respectivamente [National-Instruments, 2010].

3.2.1. NI cRIO-9014

El compactRIO es un dispositivo PAC[2], reconfigurable para control y adquisición de señales que combina arquitectura embebida con un tamaño pequeño y robusto, además utiliza módulos adicionales industriales certificados que son intercambiables. NI CompactRIO está basado en la tecnología FPGA de E/S reconfigurable (RIO por sus siglas en inglés) .

CompactRIO combina un procesador embebido en tiempo real, un FPGA de alto rendimiento y módulos de E/S intercambiables en vivo. Cada módulo de E/S se conecta directamente al FPGA, proporcionando personalización para temporización, disparo, sincronización, control y procesamiento de señales para E/S analógicas y digitales.

El FPGA es conectado al procesador embebido en tiempo real vía un bus PCI de alta velocidad. LabVIEW contiene mecanismos integrados para transferencia de datos desde los módulos de E/S al FPGA y también desde el FPGA al procesador embebido para análisis en tiempo real, procesamiento posterior, registro de datos o comunicación a un servidor conectado en red. En la Figura 3.7 se muestra la arquitectura del cRIO .

3.2.1.1. Real Time

El sistema embebido CompactRIO tiene un procesador industrial Freescale MPC5200 de 400 MHz que ejecuta de manera determinística las aplicaciones de LabVIEW Real-Time en el sistema operativo Wind River VxWorks en tiempo real. LabVIEW tiene funciones integradas

[2]Controlador de Automatización Programable

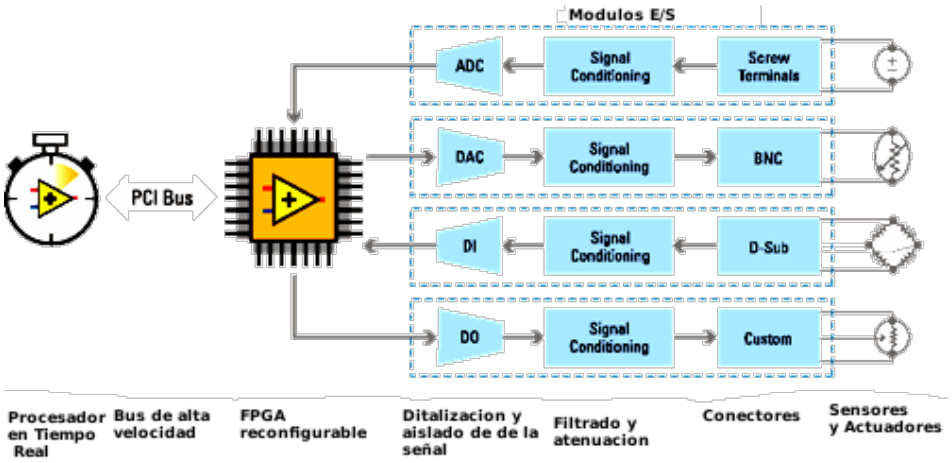


Figura 3.7: Arquitectura del sistema crio

Figura 3.8: Módulo CAN

para transferir datos entre el FPGA y el procesador en tiempo real en el sistema embebido CompactRIO.

3.2.2. Módulos

NI 9853: El NI 9853 es un módulo de dos puertos de alta velocidad para el protocolo CAN con dos formas de alimentación, a saber, externa e interna. Este módulo está diseñado para ser utilizado como parte del chassis del cRIO. La característica principal es que puede transmitir o recibir datos con una tasa de transferencia de hasta 1MB/s. En la Figura 3.8 se muestra el modulo 9853.

NI 9263: El NI 9263 es un modulo de salidas analógicas con cuatro salidas que son actualizadas con una tasa de muestreo de 100 [kS/s] [3]. La resolución de éste modulo es de 16 bits y como rango ± 10 [V]. En la Figura 3.9 se muestra el modulo 9263.

NI 9215: El NI 9215 es un módulo de entradas analógicas con cuatro entradas que son actualizadas con una tasa de muestreo de 100 kS/s. La resolución de este módulo es de 16 bits y como rango de salida ± 10[V]. Este modulo es similar al de la Figura 3.9.

[3] kilo muestras por segundo

Figura 3.9: Módulo de salidas analógicas

3.3. Acoplamiento de señales para el sistema cRIO

Las señales que se utilizan en el control son variables numéricas en un rango específico. Para poder analizar los datos de forma posterior se deben utilizar rangos que coincidan con las variables del sistema físico.

Las señales que provienen del programa son:

1. Presiones p_1, p_2 , p_3[4] y p_s.

2. Desplazamiento y

3. Voltaje de control u

Cada una de las señales mencionadas tiene una función distinta. Mientras que las presiones p_1 a p_3 son señales para la realización de control, p_s tiene como función principal el monitoreo de la la presión de suministro. Por este motivo el estudio de las señales se dividirá en dos grupos. El primer grupo corresponde a las variables de entrada como las presiones y el desplazamiento y el segundo a las variables de salida, que en este caso únicamente es el valor del voltaje u.

Las operaciones que se realicen con las señales deben ser independientes del lenguaje de programación que se utilice, puesto que la adaptación de las señales únicamente configura el protocolo CANopen y las presiones.

3.3.1. Señales de entrada y salida

3.3.1.1. Presiones

El sensor de presión entrega un voltaje en el rango de 0-10 [V] que equivale a presiones de 0-10 bar, sin embargo el comportamiento del sensor es ligeramente distinto al que se presenta en las especificaciones del fabricante [5] ya que para una presión de 1 bar el sensor debe

[4]Esta señal es adicional y será usado en futuros proyectos

[5]ver hoja de datos

"entregar" 1 volt, lo cual no ocurre con la precisión que se desea. Por lo anterior algunas mediciones permitieron obtener ecuaciones que relacionan de una forma más precisa el voltaje y la presión.

Las ecuaciones que nos relacionan al voltaje y a la presión son:

$$p_s = 1.6161V_1 - 0.1616 \tag{3.1}$$

$$p_1 = 1.0101V_2 - 0.1010 \tag{3.2}$$

$$p_2 = 1.0101V_3 - 0.1010 \tag{3.3}$$

donde V_i con $i = 1, 2, 3$ es el voltaje que obtenemos del módulo para cada entrada.

La electroválvula recibe voltajes entre 0-10 [V] que serán provistos directamente por el módulo de salidas analógicas, este voltaje debe cumplir únicamente con el requisito de no ser mayor a 10 [V] (o menor a -10[V]) para no dañar la electroválvula.

3.3.1.2. Desplazamiento

La variable que representa el desplazamiento es y, la cual debe cuantificarse en metros se obtiene implementando el protocolo CANopen (ver apéndice). Lo que se realiza en esta etapa es, desde el punto de vista de redes, una red entre el cRIO y el sensor de posición para intercambiar información de distinto tipo. Una de las ventajas de este tipo de conexión es que es posible conectar un numero grande de dispositivos distintos de tipo CANopen con una configuración similar (o con cambios sencillos) lo que incrementaría la versatilidad del sistema.

A grandes rasgos CANopen funciona enviando mensajes o recibiéndolos en forma tal que no hay posibilidad de pérdida de estos mensajes independientemente del número de dispositivos conectados al host. Además que un dispositivo CANopen puede ser interconectado fácilmente con otros dispositivos como PLC's por medio de la red. El sensor utilizado para este trabajo nos permite configurar opciones de velocidad o aceleración, sin embargo la configuración de estos datos hacia más complicada la configuración de la red por lo cual no se realizó.

En términos generales el protocolo CANopen se configura de la siguiente forma[6]:

1. Se envía un mensaje NMT para que el nodo se ponga en modo pre operacional.

2. Se envía un mensaje SDO para definir el tipo de objeto que se leerá (desplazamiento) y el lugar del diccionario de objetos donde se encuentra el objeto.

3. Se envía un mensaje NMT operacional para iniciar el envío de la posición por el nodo.

4. Se configura la transmisión de los objetos PDO para que se transmita la posición en alta prioridad (este proceso es recursivo).

[6]ver apéndice para detalles sobre los mensajes

5. Una vez obtenido el valor del diccionario de objetos indicado es necesario cambiar el formato del dato que obtenemos cuyo formato es big-endian a otro formato denominado little endian [7].

6. Se multiplica el valor obtenido en bits por un factor para obtener el resultado en metros.

3.3.2. Señal de control

El único parámetro de salida que tenemos es el voltaje de control u. El principal requerimiento es que éste voltaje este en un rango comprendido entre 10 [V] y -10 [V] para utilizar únicamente voltajes que puedan ser aplicados a la electroválvula del sistema y para no dañar el módulo de salidas analógicas.

[7] ver apéndice

Capítulo 4

Análisis de Controladores

En la teoría de control de sistemas no lineales el conocimiento del modelo puede o no ser determinante dependiendo del sistema y del controlador que se utilice. La linealización depende del desarrollo de un modelo que describa la dinámica del sistema para después transformar el modelo en una forma equivalente y más simple mientras que en un control robusto se utilizan algunas características del modelo, pero se pueden agregar incertidumbres no consideradas en el modelo y se puede realizar la medición de algunos estados. Como en el caso del control de sistemas lineales para cada sistema pueden haber varios algoritmos de control aptos para su implementación.

4.1. Generalidades del control de trayectorias

Es posible definir el objetivo del diseño de controladores de la siguiente forma: *dado un sistema físico que debe ser controlado y las especificaciones del comportamiento deseado, se construye una "ley de control" para hacer que el sistema en lazo cerrado presente el comportamiento deseado* [Jean-Jaques E.~Slotine, 1991].

Generalmente las tareas del control de sistemas pueden ser divididas en dos categorías: estabilización (o regulación) y seguimiento (o servo). En los problemas de estabilización, el sistema de control, denominado estabilizador (o regulador), se debe diseñar de tal forma que los estados del sistema en lazo cerrado se estabilicen en torno a un punto de equilibrio. Como ejemplos se pueden mencionar el control de temperatura, control de altitud, de posición, etc. En problemas de seguimiento el objetivo del diseño es construir un controlador denominado "seguidor" de tal forma que la salida del sistema "siga" una trayectoria variante en el tiempo dada. Problemas como el seguimiento de una trayectoria por un robot o un avión son tareas comunes del control de seguimiento.

4.1.1. Definición del problema de seguimiento y el problema de estabilización

El problema que se planteará en este trabajo es un problema de seguimiento por lo cual es necesario definir las diferencias entre uno y otro. Los algoritmos de control deberán estar enfocados al problema de seguimiento para un sistema que se sabe es altamente no lineal. Las definiciones de los problemas de estabilización y seguimiento se presentan a continuación:

Problema de estabilización asintótica: dado un sistema dinámico no lineal descrito por:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, u, t) \tag{4.1}$$

encontrar una ley de control tal que, comenzando en cualquier lugar de una región en $\boldsymbol{\Omega}$ el estado $\boldsymbol{x}$ tienda a 0 cuando $t \to \infty$.

Si la ley de control depende de la medición de la señales directamente, se dice que es una ley de control estática. Si ésta depende de la medición a través de una ecuación diferencial, se dice que la ley de control es una ley de control dinámica, es decir, hay una dinámica en la ley de control.

Problema de seguimiento asintótico: Dado un sistema dinámico no lineal descrito por:

$$\begin{aligned} \dot{\boldsymbol{x}} &= f(\boldsymbol{x}, u, t) \\ \boldsymbol{y} &= h(\boldsymbol{x}) \end{aligned} \tag{4.2}$$

y una trayectoria de salida deseada $\boldsymbol{y}_d$, encontrar una ley de control para la entrada u tal que, comenzando desde cualquier estado inicial en una región $\boldsymbol{\Omega}$ los errores de seguimiento $y(t) - y_d(t)$ se vuelvan cero mientras todo el estado $\boldsymbol{x}$ permanezca acotado.

Las condiciones para las cuales lo anterior se cumple son analizadas en el caso del control PID no lineal, mientras que en los dos primeros casos que se analizan es necesario consultar las referencias.

4.1.2. Implementación en tiempo discreto

Los sistemas físicos no lineales son continuos en la naturaleza, por esta razón, el diseño de controladores se hace en tiempo continuo pero la ley de control es implementada digitalmente. La necesidad de un tiempo de muestreo bajo tiene en este punto su justificación.

La diferenciación e integración numérica son, en ocasiones, partes explícitas del diseño del controlador. La diferenciación numérica puede evitar la complejidad de construir los estados del sistema basado en mediciones parciales (como los observadores). En el caso del controlador PID no lineal lo anterior se observa claramente en el caso del observador de velocidad.

En este trabajo se utilizará la diferenciación e integración numérica como se explica a continuación:

Diferenciación numérica: Para diferenciar se utiliza una expansión en serie de Taylor y después se trunca para obtener una aproximación aceptable [Chapra, 1998]. A partir de la expansión en serie de Taylor de una función $f(t)$

$$f(t_{i+1}) = f(t_i) + f'(t_i)h + \frac{f''(t_i)h^2}{2} + \dots \tag{4.3}$$

Resolviendo para $f'(t_i)$,

$$f'(t_i) = \frac{f(t_{i+1}) - f(t_i)}{h} - \frac{f''(t_i)h}{2} + O(h^2) \tag{4.4}$$

Si se truncan los términos derivativos de orden dos y mayores, entonces es posible expresar lo anterior como:

$$f'(t_i) = \frac{f(t_{i+1}) - f(t_i)}{h} + O(h) \tag{4.5}$$

Es decir nuestra mejor aproximación se obtiene utilizando más términos en la serie de Taylor o bien mejorando h que es el periodo de muestreo. En este punto se justifica el hecho de disminuir este valor para obtener un calculo más preciso.

Integración numérica: para el proceso de integración se utiliza la regla del trapecio [Chapra, 1998]. La integral se divide en intervalos de la siguiente forma

$$I = \int_{t_0}^{t_1} f(t)dt + \int_{t_1}^{t_2} f(t)dt + ... + \int_{t_{n-1}}^{t_n} f(t)dt \tag{4.6}$$

y aplicando sucesivamente la regla del trapecio, entonces

$$I = \frac{h}{2}\left[f(t_0) + 2\sum_{i=1}^{n-1} f(t_i) + f(t_n)\right] \tag{4.7}$$

con $h = \dfrac{b-a}{n}$. En este caso el error se obtiene sumando los errores individuales por cada aplicación de la regla del trapecio, i.e.,

$$E_t = -\frac{1}{12}\frac{(b-a)^3}{n^3}\sum_{i=1}^{n} f''(\xi_i) = -\frac{1}{12}h^3\sum_{i=1}^{n} f''(\xi_i) \tag{4.8}$$

De esta expresión es claro que el periodo de muestreo también influye en el calculo de la integral.

4.2. Métodos para el diseño de controladores no lineales

Como ocurre en el análisis de sistemas de control lineales, no hay un método general para el diseño de controladores no lineales. En lugar de ello, hay una conjunto de alternativas y técnicas complementarias que son aplicables a casos particulares de control. Para cada sistema no lineal existe un método de control que se adapta mejor a la dinámica del sistema, por lo cual al realizar la implementación es necesario evaluar varios métodos antes de elegir el adecuado.

En este trabajo se implementan controles de tipo robusto, específicamente de tipo Modo Deslizante, PID aplicado a un sistema no lineal y una combinación de ambos. Sin embargo se presentan otros métodos que existen para sistemas no lineales tales como:

Linealización: El primer paso para el diseño de un sistema de control para una planta es desarrollar un modelo que describa de manera apropiada la dinámica de la planta en el rango de interés. Los modelos de los sistemas presentan formas variadas dependiendo de las suposiciones que se hagan con respecto al comportamiento del mismo. Algunas de estas formas conducen a un diseño más simple del controlador. La linealización usa técnicas para transformar el modelo original del sistema en un modelo equivalente más simple.

La linealización puede ser usada como una metodología de diseño no lineal. La idea básica es, primero, transformar el sistema no lineal en un sistema lineal (completamente lineal o parcialmente) y después usar una de las técnicas conocidas del diseño lineal para realizar el control. Este método ha sido usado para resolver un conjunto de problemas prácticos de control en sistemas no lineales. Típicamente requieren la medición de todos los estados. Sin embargo, éste método no garantiza robustez en presencia de incertidumbres o perturbaciones de los parámetros.

Las técnicas de linealización pueden ser usadas para simplificar el modelo y pueden ser empleados en aplicaciones de controladores robustos, adaptables, pasivos, etc.

Control Robusto: En un control no lineal basado puramente en el modelo (como la linealización), la ley de control es diseñada basada en un modelo nominal del sistema físico. La forma en que el sistema se comportará en presencia de perturbaciones o incertidumbres no es clara. En el control robusto no lineal (como el control por modos deslizantes), por otro lado, el controlador es diseñado basado en el modelo nominal y la caracterización de algunas incertidumbres del modelo.

Las técnicas de control robusto has sido probadas de forma eficiente en una variedad específica de problemas prácticos y generalmente requiere de mediciones de los estados del sistema.

Control adaptable: El control adaptable es una aproximación del trato con incertidumbres del sistema o de sistemas variantes en el tiempo. Los diseños de control adaptable se aplican a sistemas con una estructura dinámica conocida, pero parámetros desconocidos o que tienen variaciones lentas. Los controladores adaptables que se desarrollan para sistemas lineales o no lineales son inherentemente no lineales.

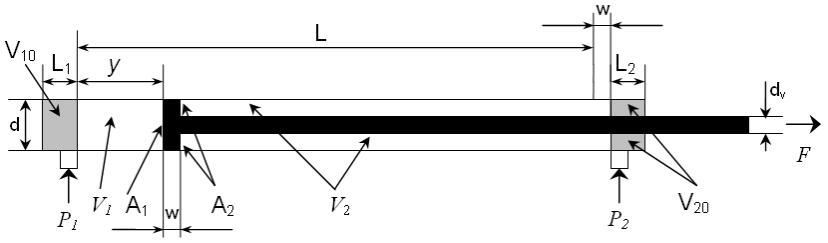


Figura 4.1: Esquema de un pistón neumático diferencial

En este capítulo se estudiarán los controladores que se implementarán posteriormente. Cada uno de los algoritmos presentados tendrá una breve explicación, así como la simulación de las señales más importantes del sistema, además se presentarán los resultados experimentales.

4.3. Modelo dinámico

El modelo dinámico considerado se encuentra formado por los subsistemas pistón y electroválvula. El orden del sistema en conjunto con la electroválvula es de sexto orden, sin embargo es posible realizar algunas simplificaciones para obterner un sistema de orden cuatro. El modelo dinámico del sistema neumático ha sido publicado en varios artículos [Goettert and Neumann, 1999, Sobczyk and Perondi, 2006, Beater, 2007, Kothapalli and Y.Hassan, 2007]. La Figura 4.1 muestra el esquema del pistón.

Los estados del sistema están dados por

$$\boldsymbol{x} = \begin{bmatrix} y \\ \dot{y} \\ p_1 \\ p_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \tag{4.9}$$

con y=x_1 como el desplazamiento del pistón [m], $\dot{y}$=x_2 como la velocidad del pistón [m/s], p_1=x_3 como la presión de la cámara 1 [Pa] y p_2=x_4 como la presión de la cámara 2 [Pa]. Un diagrama de bloques simplificado del sistema es mostrado en la Figura (4.2).

El modelo está basado principalmente en la segunda ley de Newton y la primera ley de la termodinámica para un proceso adiabático, la dinámica se muestra en las siguientes ecuaciones,

$$\dot{x}_1 = x_2, \tag{4.10}$$

$$\dot{x}_2 = \frac{\mathrm{A}_1}{\mathrm{M}} x_3 - \frac{\mathrm{A}_2}{\mathrm{M}} x_4 - \frac{\mathrm{A}_1 - \mathrm{A}_2}{\mathrm{M}} \mathrm{p}_{\mathrm{atm}} - \frac{F_{\mathrm{V}}}{\mathrm{M}} x_2 \tag{4.11}$$

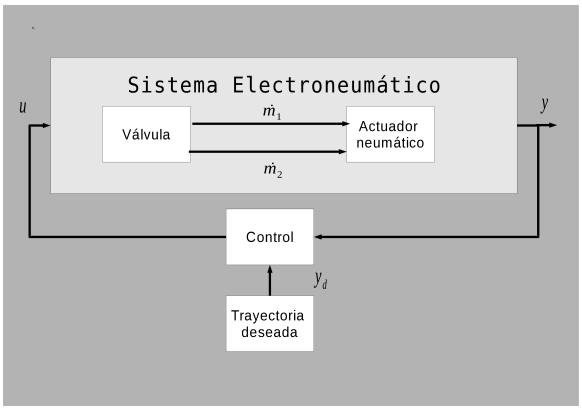


Figura 4.2: Diagrama de bloques del sistema electroneumático

donde A_1 = sección transversal de la cámara 1 ($5.0265 \cdot 10^{-4}\mathrm{m}^2$), A_2 = sección transversal de la cámara 2 ($4.5357 \cdot 10^{-4}\mathrm{m}^2$), M = masa en movimiento (5.8kg), $\mathrm{p}_{\mathrm{atm}}$ = presión atmosférica ($1 \cdot 10^5$Pa) y F_{V} = Fricción de Coulomb.

La dinámica de las presiones en las cámaras son

$$\dot{x}_3 = -\frac{\gamma}{\mathrm{V}_{10} + \mathrm{A}_1 x_1}\big(x_2 x_3 \mathrm{A}_1 - \mathrm{RT}\dot{m}_1(x_3, u)\big), \tag{4.12}$$

$$\dot{x}_4 = -\frac{\gamma}{\mathrm{V}_{20} + \mathrm{A}_2(\mathrm{L} - x_1)}\big(x_2 x_4 \mathrm{A}_2 - \mathrm{RT}\dot{m}_2(x_4, u)\big) \tag{4.13}$$

donde γ = índice adiabático (1.4), V_{10} = zona muerta de la cámara 1 incluyendo tubos ($1.5 \cdot 10^{-5}\mathrm{m}^3$), V_{20} = zona muerta 2 incluyendo tubos ($1.5 \cdot 10^{-5}\mathrm{m}^3$), L = longitud del vástago del pistón (0.76m), R = constante del gas (287.05 $\frac{\mathrm{J}}{\mathrm{kg \cdot K}}$), T = temperatura ($293.15$K) y u = señal de control.

Para resolver las Ecuaciones (4.12) y (4.13) es necesario calcular la dinámica de cada uno de los flujos de masa

$$\dot{m}_1 = \begin{cases} \mathrm{C}(u)\rho_0 \mathrm{p_s}\sqrt{1 - \left(\frac{\left(\frac{x_3}{\mathrm{p_s}} - \mathrm{b}\right)}{1-\mathrm{b}}\right)^2} & , u \geq 0 \text{ y } \frac{x_3}{\mathrm{p_s}} \geq b \\ \mathrm{C}(u)\rho_0 \mathrm{p_s} & , u \geq 0 \text{ y } \frac{x_3}{\mathrm{p_s}} < b \\ \mathrm{C}(u)\rho_0 x_3 & , u \leq 0 \end{cases} \tag{4.14}$$

y

$$\dot{m}_2 = \begin{cases} -\mathrm{C}(u)\rho_0 \mathrm{p_s}\sqrt{1 - \left(\frac{\left(\frac{x_4}{\mathrm{p_s}} - \mathrm{b}\right)}{1-\mathrm{b}}\right)^2} & , u \leq 0 \text{ y } \frac{x_4}{\mathrm{p_s}} \geq b \\ -\mathrm{C}(u)\rho_0 \mathrm{p_s} & , u \leq 0 \text{ y } \frac{x_4}{\mathrm{p_s}} > b \\ -\mathrm{C}(u)\rho_0 x_4 & , u \geq 0 \end{cases} \tag{4.15}$$

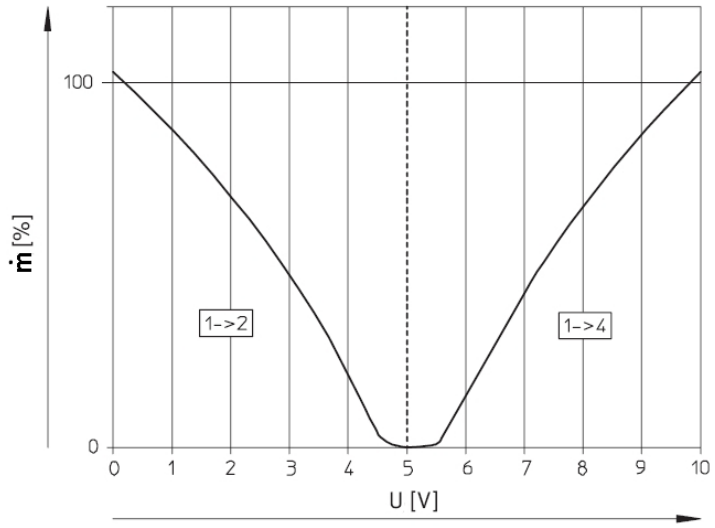


Figura 4.3: Conductancia

con $\dot{m}_1$ = flujo de masa de la cámara 1 ($\frac{\text{kg}}{\text{s}}$), $\dot{m}_2$ = flujo de masa de la cámara 2 ($\frac{\text{kg}}{\text{s}}$), ρ_0 = densidad ($1.204\ \frac{\text{kg}}{\text{m}^3}$), C($u$) = función de flujo de masa, p_s = presión de suministro (Pa) y $\text{b} = (\frac{2}{\gamma+1})^{\frac{\gamma+1}{\gamma-1}}$ = factor de presión crítica (0.582). La dependencia en u de la función flujo de masa se debe a la conductancia que es la que lo determina como se observa en la Figura (4.3) y de hecho se toma como $C(u) = 0.8u$.

En la Figura (4.4) se muestra el diagrama de bloques que se utilizó para la simulación de la planta.

4.4. Control PID

El control PID ha sido uno de los controles más populares en la industria. Esto se debe a la simplicidad en la implementación. Los controladores PID han sido utilizados en una amplia variedad de sistemas dinámicos, desde procesos industriales hasta aviones. Aunque un controlador PID con ganancias fijas es adecuado para controlar un proceso lineal, los requerimientos para un control de alto desempeño con cambios en las condiciones de operación, o en los parámetros, están más allá de las capacidades de un controlador PID simple [Ogata, 1998].

Los controladores PID se expresan de una forma general como sigue:

$$u(t) = K_p e(t) + K_i \int e(\tau)d\tau + K_d \dot{e}(t), \tag{4.16}$$

con K_p,K_i, K_d ganancias del controlador, las cuales que permiten la reducción del error en régimen permanente, que el error en régimen permanente tienda a cero y que el error tienda a cero más rápido respectivamente. $u(t)$ es la entrada del sistema y $e(t)$ es el error del sistema.

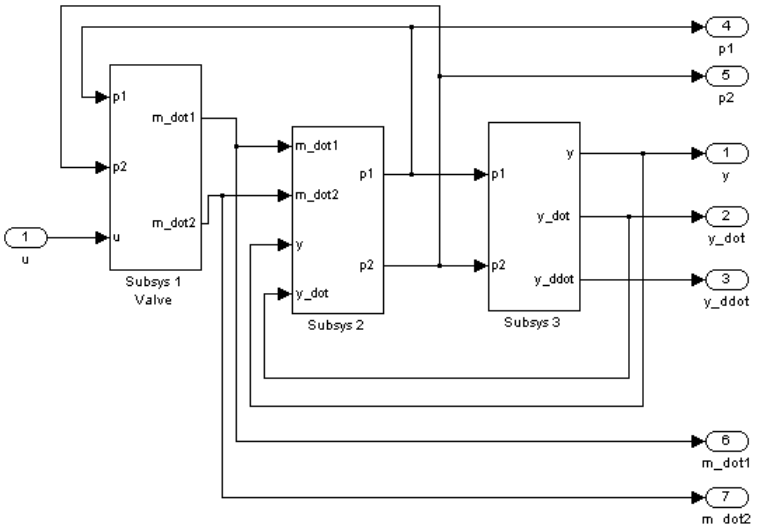


Figura 4.4: Diagrama de bloques para la simulación del comportamiento de la planta

Para este caso, el error $e(t)$ está dado por $e(t) = y - y_d$ con y como el desplazamiento real y y_d el desplazamiento deseado. La derivada de la señal de error se calcula de la siguiente forma:

$$\dot{e}(t) \equiv F_d = \frac{e_k - e_{k-1}}{h}, \tag{4.17}$$

donde la derivada se calcula de forma discreta utilizando el tiempo de muestreo h que se fija para la simulación en 1[ms]. La integral se calcula utilizando un proceso similar.

$$\int e(\tau)d\tau \equiv F_i = F_{i-1} + \frac{h(e_k + e_{k-1})}{2}. \tag{4.18}$$

En el caso de la integral se utiliza la regla del trapecio. La ley de control puede escribirse como

$$u = -K_p e_k + K_i F_i + K_d F_d \tag{4.19}$$

Las Figuras (4.5), (4.6), (4.7) muestran el comportamiento de una control PID para una señal senoidal en una tarea de seguimiento con una frecuencia de 0.1 [Hz] y una amplitud de 0.36[m].

4.5. Control por Modos deslizantes

4.5.1. Modelo linealizado del sistema

Para el desarrollo del control con modos deslizantes (SMC) se utilizó un modelo linealizado del sistema servoneumático. El desarrollo del modelo linealizado se presenta en [Ning and

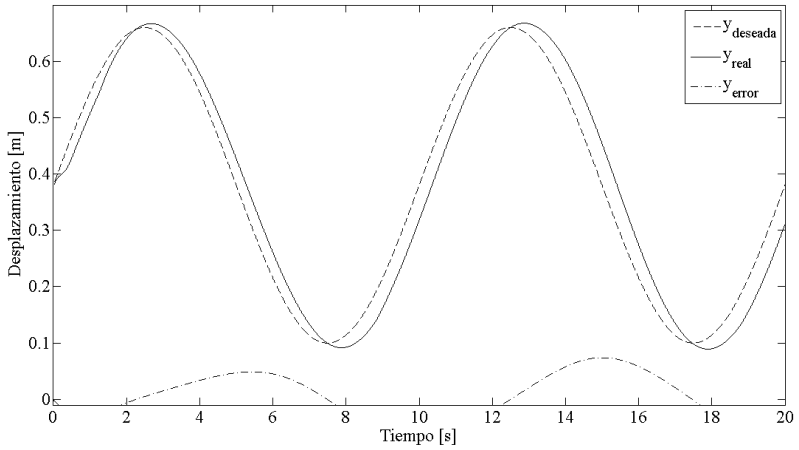


Figura 4.5: Seguimiento

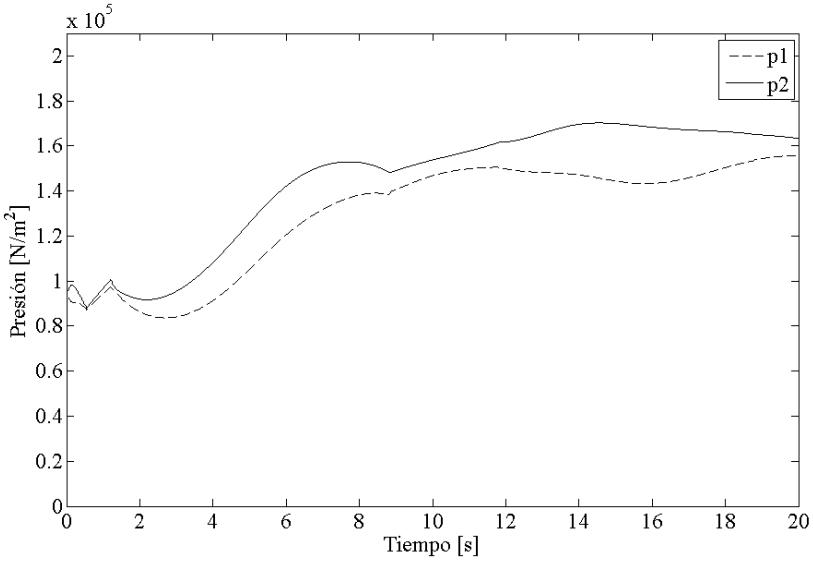


Figura 4.6: Presiónes de las cámaras

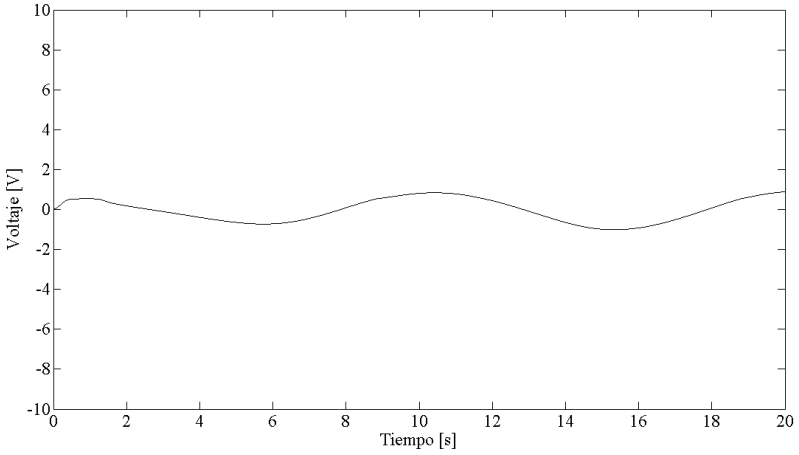


Figura 4.7: Señal de control

Bone, 2005] y consiste en expandir en serie de Taylor el modelo no lineal con un polinomio de tercer grado y obtener la función de transferencia para el modelo simplificado,

$$\frac{Y(s)}{U(s)} = \frac{n_1(s+a)}{s(s+b)(s^2+d_2s+d_1)}, \tag{4.20}$$

donde y es la posición del pistón y u es la entrada de la válvula. A partir de la dinámica del cero en $-a$ y el polo $-b$ se cancelan de tal forma que el modelo puede reducirse, i.e.,

$$\frac{Y(s)}{U(s)} = \frac{n_1\dfrac{a}{b}}{s(s^2+d_2s+d_1)} = \frac{n_0}{s(s^2+d_2s+d_1)}. \tag{4.21}$$

4.5.2. Diseño del Control por Modos Deslizantes

Para el diseño del control se utiliza el método que se presenta en [Jean-Jaques E.~Slotine, 1991]. El modelo nominal de la Ecuación (4.21) se reescribe de la siguiente forma:

$$\dddot{y} = -d_2\ddot{y} - d_1\dot{y} + n_0u. \tag{4.22}$$

Una superficie deslizante apta para la planta es

$$S = \ddot{y} - \ddot{y}_d + 2\lambda(\dot{y} - \dot{y}_d) + \lambda^2(y - y_d), \tag{4.23}$$

con

$$\dot{S} = \dddot{y} - \dddot{y}_d + 2\lambda(\ddot{y} - \ddot{y}_d) + \lambda^2(\dot{y} - \dot{y}_d). \tag{4.24}$$

Para obtener la ley de control se requiere una señal de control equivalente que mantendrá los estados del sistema en la superficie deslizante una vez que éstos la alcancen. Esto ocurre cuando $\dot{S}$=0. Por lo tanto, la señal equivalente del sistema, u_{eq}, se obtiene despejando $\dddot{y}$ de la Ecuación (4.24) con $\dot{S} = 0$ y sustituyendo en la Ecuación (4.22), i.e.,

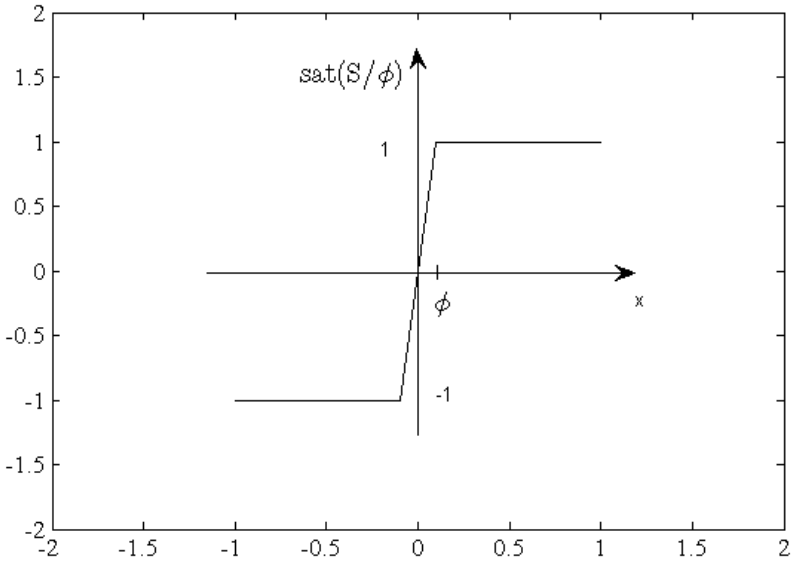


Figura 4.8: Comportamiento de la función de saturación $sat(S/\phi)$

$$u_{eq} = \frac{1}{n_0}[\dddot{y}_d + d_2\ddot{y} + d_1\dot{y} - 2\lambda(\ddot{y} - \ddot{y}_d) - \lambda^2(\dot{y} - \dot{y}_d)] \tag{4.25}$$

La señal de control deslizante que compensa las incertidumbres del modelo y lleva a la superficie deslizante los estados del sistema que se alejan es:

$$u_s = -k_s \text{sat}(\frac{S}{\phi}), \tag{4.26}$$

donde

$$\text{sat}(\frac{S}{\phi}) = \left\{ \begin{array}{cc} \frac{S}{\phi} & \text{if } \frac{S}{\phi} \leq 1 \\ \text{sign}(\frac{S}{\phi}) & \text{if } \frac{S}{\phi} > 1 \end{array} \right\}$$

$$\text{sign}(x) = \left\{ \begin{array}{cc} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{array} \right\}.$$

La función de saturación es mostrada en la Figura (4.8). Lo anterior conduce a la siguiente Ley de Control

$$u = u_{eq} + u_s \tag{4.27}$$

Para reducir el chattering se modifica el parámetro ϕ, mientras que los parámetros λ y k_s se sintonizan manualmente.

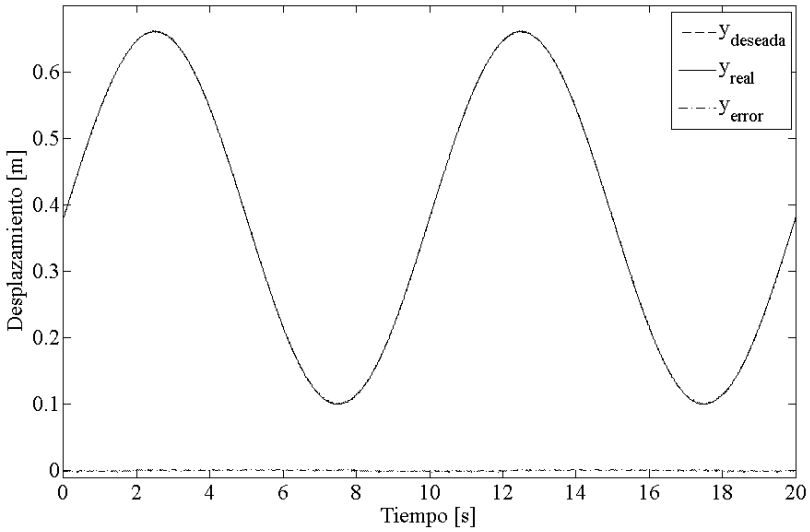


Figura 4.9: Seguimiento

En la simulación se utiliza la Ecuación (4.17) para obtener el error de seguimiento y de forma análoga se obtienen las derivadas de segundo y tercer orden. Las Figuras (4.9), (4.10), (4.11) muestran el comportamiento del control SMC para una señal deseada senoidal con una frecuencia de 0.1 [Hz] y una amplitud de 0.36[m].

4.6. Control PID no lineal

La idea básica del controlador PID no lineal es controlar la diferencia de fuerzas en cada una de las cámaras que conforman el pistón (Capitulo 3). El controlador está diseñado bajo la idea de un controlador PID con un Modo Deslizante en la parte integral [Arteaga et~al.].

Para obtener la ley de control se utiliza el modelo matemático del sistema (4.3). Las Ecuaciones (4.9) a (4.15) y las siguientes consideraciones :

1. $\mathrm{p_{atm}} \leq x_3 \leq \mathrm{p}_s$
2. $\mathrm{p_{atm}} \leq x_4 \leq \mathrm{p}_s$
3. $C(u)$ es una función monótona y $C(0) = 0$
4. Como $C(u)$ es una función monótona existe una constante $\mathrm{K_u}$ que satisface

$$|C(u)| > \mathrm{K_u}|u| \tag{4.28a}$$

Las primeras dos consideraciones provienen de aspectos físicos mientras que las siguientes dos son definiciones.

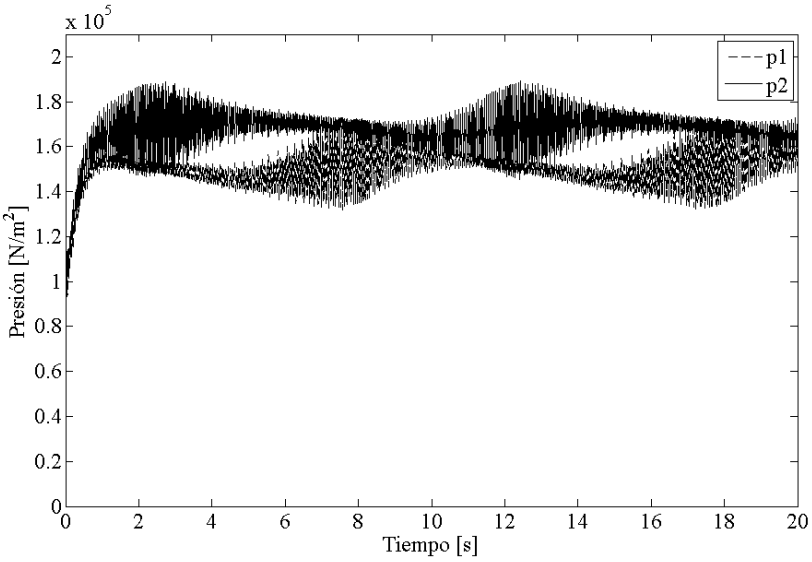


Figura 4.10: Presiones de las cámaras

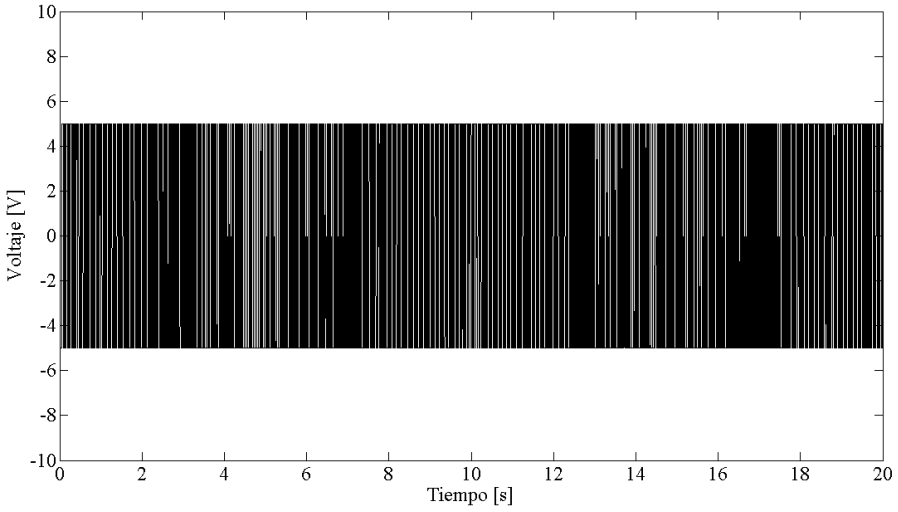


Figura 4.11: Señal de control

Adicionalmente, si $|u| < \infty$ existe $0 \leq J(u) < \infty$ tal que

$$C(u) = J(u)u, \tag{4.28b}$$

y $J(u) = 0$ sólo sí $u = 0$. Como $C(u)$ es una función monótona conduce a que

$$C(u_1) = J(u_1)u_1 > J(u_2)u_2 = C(u_2) \tag{4.28c}$$

si $u_1 > u_2$.

Teorema 1 *Sea*

$$\sigma(s) = \int_0^t (K_\beta s + \mathrm{sign}(s))\mathrm{d}\delta \qquad\qquad \sigma(0) = 0 \tag{4.29}$$

donde $\sigma, s \in \mathbb{R}^{\mathrm{n}}$. *Si*

$$e = s + K_\gamma \sigma(s) \tag{4.30}$$

con e *acotado, entonces* s, σ *y* $\dot{\sigma}$ *también están acotados, si* $K_\beta, K_\gamma > 0$ *con* $K_\beta, K_\gamma \in \mathbb{R}^{\mathrm{n\,x\,n}}$.

Adicionalmente, si $\dot{e}$ *esta acotado por* $\| \dot{e} \| \leq \mathrm{e_{max}}$ *y sí*

$$\phi = \lambda_{\min}(K_\gamma) - e_{\max} > 0 \tag{4.31}$$

donde $s = 0$ *en un tiempo finito, entonces*

$$K_\gamma \leq \frac{\| s(0) \|}{\phi}. \tag{4.32}$$

[Arteaga et~al.]

Para el diseño del algoritmo de control, con base en la Ecuación (4.11), se define

$$\begin{aligned} x_{\mathrm{f}} &\equiv \mathrm{A}_1(x_3 - \mathrm{p_{atm}}) - \mathrm{A}_2(x_4 - \mathrm{p_{atm}}) \\ &= \mathrm{A}_1 x_3 - \mathrm{A}_2 x_4 - \mathrm{p_{atm}}(\mathrm{A}_1 - \mathrm{A}_2), \end{aligned} \tag{4.33}$$

de esta forma se obtiene

$$\dot{x}_{\mathrm{f}} = \mathrm{A}_1 \dot{x}_3 - \mathrm{A}_2 \dot{x}_4 \tag{4.34}$$

$$\begin{aligned} &= \frac{\mathrm{A}_1 \gamma \mathbf{RT}}{\mathrm{V}_{10} + \mathrm{A}_1 x_1} \dot{m}_1 - \frac{\mathrm{A}_2 \gamma \mathbf{RT}}{\mathbf{V}_{20} + \mathrm{A}_2(\mathrm{L} - x_1)} \dot{m}_2 \\ &\quad - \left(\frac{\mathrm{A}_1^2\, \gamma\, x_3}{\mathrm{V}_{10} + \mathrm{A}_1 x_1} \dot{m}_1 - \frac{\mathrm{A}_2^2\, \gamma\, x_4}{\mathbf{V}_{20} + \mathrm{A}_2(\mathrm{L} - x_1)} \dot{m}_2 \right) x_2. \end{aligned} \tag{4.35}$$

Con lo cual, es posible definir

$$\dot{m}_1 \equiv \bar{\gamma}_1(x_3) C(u) \tag{4.36}$$

$$\bar{\gamma}_1(x_3) = \begin{cases} \rho_0 \mathrm{p_s}\sqrt{1-\left(\frac{\left(\frac{x_3}{\mathrm{p_s}}-\mathrm{b}\right)}{1-\mathrm{b}}\right)^2} & , u \geq 0 \text{ y } \frac{x_3}{\mathrm{p_s}} \geq b \\ \rho_0 \mathrm{p_s} & , u \geq 0 \text{ y } \frac{x_3}{\mathrm{p_s}} < b \\ \rho_0 x_3 & , u \leq 0 \end{cases} \tag{4.37}$$

$$\dot{m}_2 \equiv \bar{\gamma}_2(x_4) C(u) \tag{4.38}$$

$$\bar{\gamma}_2(x_4) = \begin{cases} -\rho_0 \mathrm{p_s}\sqrt{1-\left(\frac{\left(\frac{x_4}{\mathrm{p_s}}-\mathrm{b}\right)}{1-\mathrm{b}}\right)^2} & , u \leq 0 \text{ y } \frac{x_4}{\mathrm{p_s}} \geq b \\ -\rho_0 \mathrm{p_s} & , u \leq 0 \text{ y } \frac{x_4}{\mathrm{p_s}} > b \\ -\rho_0 x_4 & , u \geq 0 \end{cases} \tag{4.39}$$

De acuerdo a los aspectos físicos que consideramos

$$\bar{\gamma}_1(x_3) \text{ y } \bar{\gamma}_2(x_4) \tag{4.40}$$

están acotados.

Por eso la Ecuación (4.35) se puede escribir en la forma

$$\dot{x}_\mathrm{f} = \bar{\gamma}\, C(u) + b(t) x_2 \tag{4.41}$$

con

$$\begin{aligned} \bar{\gamma} &= \frac{\mathrm{A_1}\gamma \mathrm{RT}}{\mathrm{V_{10}} + \mathrm{A_1} x_1} \bar{\gamma}_1(x_3) - \frac{\mathrm{A_2}\gamma \mathrm{RT}}{\mathrm{V_{20}} + \mathrm{A_2}(\mathrm{L} - x_1)} \bar{\gamma}_2(x_4) \\ \bar{\gamma} &> 0, \end{aligned} \tag{4.42}$$

y

$$b(t) = -\left(\frac{\mathrm{A_1^2}\,\gamma\, x_3}{\mathrm{V_{10}} + \mathrm{A_1} x_1} \dot{m}_1 - \frac{\mathrm{A_2^2}\,\gamma\, x_4}{\mathrm{V_{20}} + \mathrm{A_2}(\mathrm{L} - x_1)} \dot{m}_2, \right) \tag{4.43}$$

donde $b(t)$ es acotado puesto que x_3 y x_4 están acotados y $\bar{\gamma}$ es acotado ya que $\bar{\gamma}_1(x_3)$ y $\bar{\gamma}_2(x_4)$

Finalmente de las Ecuaciones (4.9) , (4.11), (4.33) y (4.35) se obtiene

$$\mathrm{M}\ddot{y} + F_V \dot{y} = x_\mathrm{f} + p_{\mathrm{b}1} \tag{4.44}$$

$$\dot{x}_\mathrm{f} = \bar{\gamma}\, C(u) + b(t)\dot{y} + p_{\mathrm{b}2} \tag{4.45a}$$

$$\bar{\gamma} > 0. \tag{4.45b}$$

donde se han agregado $p_{\mathrm{b}1}$ y $p_{\mathrm{b}2}$ como perturbaciones acotadas.

De esta forma se puede usar directamente [Arteaga et~al.] para redefinir la ley de control como

$$x_{\mathrm{fd}} = -\mathrm{K}_{\mathrm{py}} s_0 \tag{4.46}$$

donde

$$\begin{aligned}
s_0 &= \dot{y}_0 - \dot{y}_{\mathrm{r}} && (4.47)\\
\dot{y}_0 &= \dot{\hat{y}} - \lambda_{\mathrm{z}} z && (4.48)\\
\dot{y}_{\mathrm{r}} &= \dot{y}_{\mathrm{d}} - \lambda_y(\hat{y} - y_{\mathrm{d}}) + s_{\mathrm{d}} - K_{\gamma y}\sigma_{\mathrm{y}} && (4.49)\\
\sigma_{\mathrm{y}} &= \int_0^t \Big(\mathrm{K}_{\beta y} s_{\mathrm{y1}}(\vartheta) + \mathrm{sign}\big(s_{\mathrm{y1}}(\vartheta)\big)\Big)\mathrm{d}\vartheta \qquad \sigma_{\mathrm{y}}(0) = 0 && (4.50)\\
s_{\mathrm{y1}} &= s_{\mathrm{y}} - s_{\mathrm{d}} && (4.51)\\
s_{\mathrm{d}} &= s_{\mathrm{y}}(0)e^{-\mathrm{K_t}} && (4.52)\\
&= \dot{\bar{y}} + \lambda_{\mathrm{y}}\bar{y} && (4.53)\\
s_{\mathrm{y}} &= \dot{\hat{y}} - \dot{y}_{\mathrm{d}} + \lambda_{\mathrm{y}}(\hat{y} - y_{\mathrm{d}}) && \\
\dot{\bar{y}} &= \dot{\hat{y}} - \dot{y}_{\mathrm{d}} && (4.54)\\
\bar{y} &= \hat{y} - y_{\mathrm{d}} && (4.55)\\
\dot{\hat{y}} &= \dot{y}_{\mathrm{d}} - \lambda_{\mathrm{y}}(\hat{y} - y_{\mathrm{d}}) + s_{\mathrm{d}} + \mathrm{K_d}\lambda_{\mathrm{z}}\int_0^t z(\vartheta)\mathrm{d}\vartheta + \lambda_{\mathrm{z}} z + K_{\mathrm{d}} z && (4.56)\\
z &= y - \hat{y}, && (4.57)\\
& && (4.58)
\end{aligned}$$

donde z es el error de observación y y_d es la posición deseada [Arteaga et~al.]. El control debe satisfacer

$$x_{\mathrm{f}} \rightarrow x_{\mathrm{fd}} \tag{4.59}$$

donde

$$\Delta x_{\mathrm{f}} \triangleq x_{\mathrm{f}} - x_{\mathrm{fd}} \rightarrow 0 \tag{4.60}$$

Ahora se debe elegir u para la Ecuación (4.45a). Para lograr esto es necesario definir

$$\begin{aligned}
\Delta y &\equiv y - y_{\mathrm{d}}, && (4.61)\\
\Delta\dot{y} &\equiv \dot{y} - \dot{y}_{\mathrm{d}}. && (4.62)
\end{aligned}$$

Si se sustituye $\dot{y}$ en la Ecuación (4.45a) resulta

$$\dot{x_{\mathrm{f}}} = \bar{\gamma}\, C(u) + b(t)\Delta\dot{y} + b(t)\dot{y}_{\mathrm{d}} + p_{\mathrm{b2}} \tag{4.63}$$

Basado en la suposición de que $\Delta\dot{y}$ está acotado y usando el Teorema 1 es posible definir

$$\begin{aligned}
s_{\mathrm{f}} &= \Delta x_{\mathrm{f}} + K_{\gamma f}\sigma_{\mathrm{f}} \;=\; x_{\mathrm{f}} - x_{\mathrm{fd}} - K_{\gamma f}\sigma_{\mathrm{f}}, && (4.64)\\
\sigma_{\mathrm{f}} &= \int_0^t \big(K_{\beta f}\Delta x_{\mathrm{f}}(\vartheta) + \mathrm{sign}(\Delta x_{\mathrm{f}}(\vartheta)\big)\mathrm{d}\vartheta \qquad \sigma_{\mathrm{f}}(0) = 0, && (4.65)\\
u &= -\mathrm{K_{pf}} s_{\mathrm{f}}. && (4.66)
\end{aligned}$$

De ese modo, se obtiene

$$\dot{x}_{\mathrm{f}} = \bar{\gamma}\, C(-\mathrm{K}_{\mathrm{pf}} s_{\mathrm{f}}) + b(t)\Delta\dot{y} + \bar{p}_{\mathrm{b2}}, \tag{4.67a}$$

$$\bar{p}_{\mathrm{b2}} = b(t)\dot{y}_{\mathrm{d}} + p_{\mathrm{b2}} \tag{4.67b}$$

con $\bar{p}_{\mathrm{b2}}$ acotado.

Después de un tiempo suficientemente grande, $z = 0$ y $s_d = 0$ y la ley de control puede verse como un controlador PID no lineal como se explica a continuación. Puesto que el error de observación es cero, se puede reescribir la Ecuación (4.58) como

$$\begin{aligned}
x_{\mathrm{fd}} &= -\mathrm{K}_{\mathrm{py}}\left(\dot{y}_0 - \dot{y}_{\mathrm{r}}\right) \\
&= -\mathrm{K}_{\mathrm{py}}\left(\dot{y} - \dot{y}_{\mathrm{d}} - \lambda_y \Delta y - K_{\gamma y}\sigma_{\mathrm{y}}\right) \\
&= -\mathrm{K}_{\mathrm{py}}\left(\dot{y} - \dot{y}_{\mathrm{d}} - \lambda_y \Delta y - K_{\gamma y}\int_0^t \Big(\mathrm{K}_{\beta y} s_{\mathrm{y1}}(\vartheta) + \mathrm{sign}\big(s_{\mathrm{y1}}(\vartheta)\big)\Big)\mathrm{d}\vartheta\right) \\
&= -\mathrm{K}_{\mathrm{py}}\left(\Delta\dot{y} - \lambda_y \Delta y - K_{\gamma y}\int_0^t \Big(\mathrm{K}_{\beta y}(\Delta\dot{y} + \lambda_y \Delta y)(\vartheta) + \mathrm{sign}\big(\Delta\dot{y} + \lambda_y \Delta y\big)\Big)\right) \\
&= \underbrace{\mathrm{K}_{\mathrm{py}}\lambda_y \Delta y}_{\mathrm{P}} + \underbrace{\mathrm{K}_{\mathrm{py}} K_{\gamma y}\int_0^t \big(\mathrm{K}_{\beta y}(\Delta\dot{y} + \lambda_y \Delta y)\big)}_{\mathrm{I}} - \underbrace{\mathrm{K}_{\mathrm{py}}\Delta\dot{y}}_{\mathrm{D}} \\
&\quad -\mathrm{K}_{\mathrm{py}}\mathrm{sign}\big(\Delta\dot{y} + \lambda_y \Delta y\big)
\end{aligned} \tag{4.68}$$

Para simular el control PID no lineal se utilizaron las Ecuaciones (4.43-4.62), utilizando la misma metodología de los primeros dos algoritmos anteriores. Sin embargo, en este caso se utiliza un observador como se puede ver en la Ecuación (4.56), lo que produjo una modificación sustancial en cuanto a los valores iniciales. En los primeros dos controladores el valor inicial de y, correspondiente al primer ciclo se fijo en cero, mientras que en el PID no lineal el valor inicial se debe fijar como la posición en la cual se encuentra el pistón en el momento de iniciarse la ejecución del programa. La Figuras (4.12) a (4.14) muestran el comportamiento del control PID no lineal para una señal senoidal con una frecuencia de 0.1 [Hz] y una amplitud de 0.36[m].

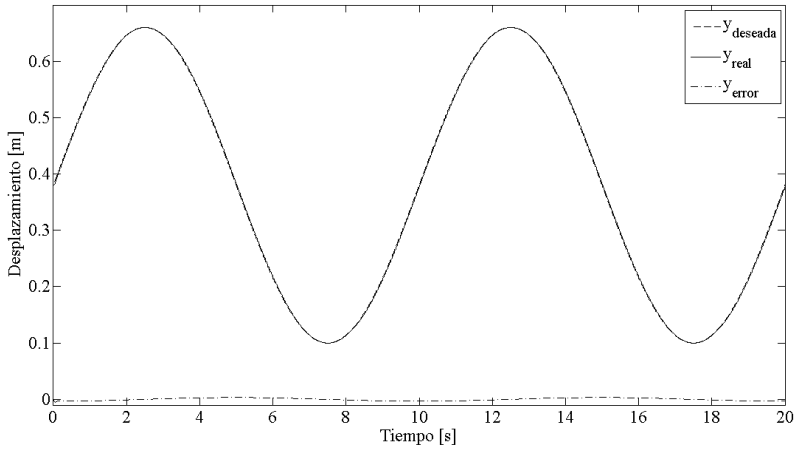


Figura 4.12: Seguimiento

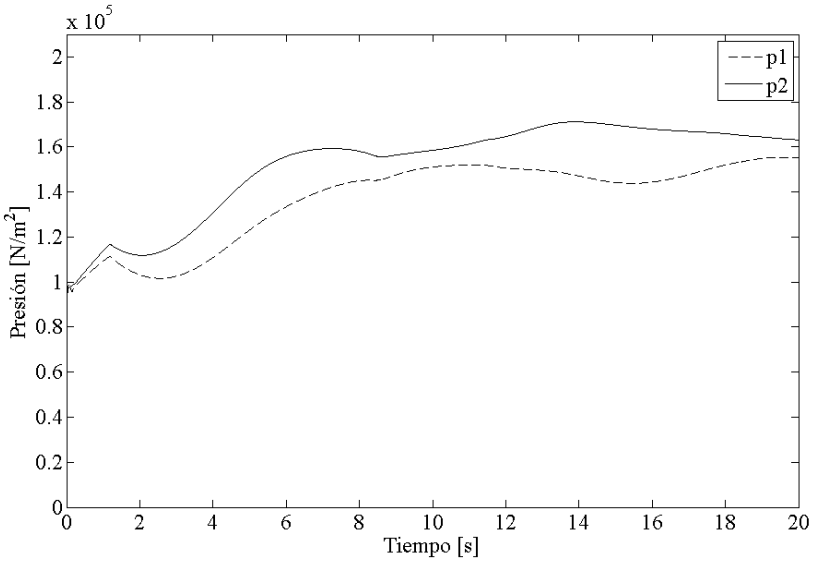


Figura 4.13: Presiónes de las cámaras

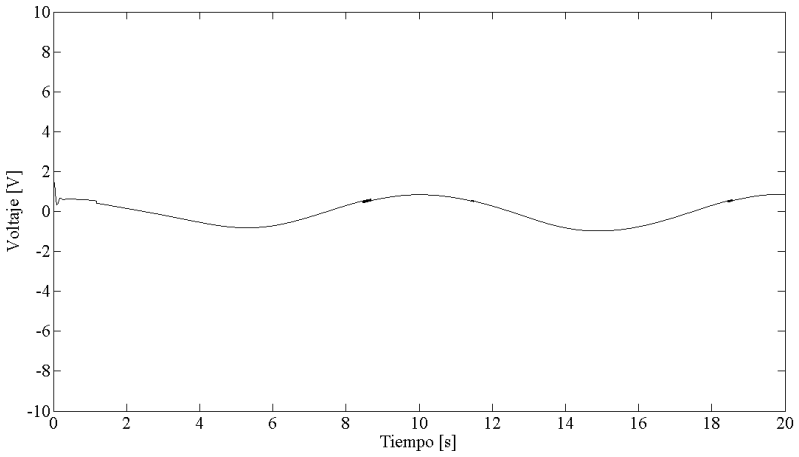


Figura 4.14: Señal de control

Capítulo 5

Implementación de los Controladores

Tomando en cuenta las restricciones del diseño (Sección 2.3) y priorizando la implementación de controladores que requieren bajo tiempo de muestreo de tal forma que sea posible implementar algoritmos de control como Control por Modos Deslizantes (SMC) o una combinación de Control por Modos Deslizantes con PID. Esto conduce a la realización de pruebas en dos plataformas de programación. La primera (siguiendo la idea inicial del trabajo) utilizando CVI y la segunda utilizando LabVIEW .

Como se verá adelante se llevo a cabo la implementación de los controladores PID en ambas plataformas para realizar las primeras pruebas de adquisición, desempeño, tiempo de muestreo y almacenamiento de datos. Los resultados en cada una de las pruebas que se realizaron con el control PID, determinaron la viabilidad de cada una de las plataformas y, por tanto se decidió por una de ellas para llevar a cabo la programación de los siguientes algoritmos de control.

5.1. CVI y programación entrada/salida

5.1.1. Programación a nivel FPGA

Como se planteó en la Sección (3.3.1.2), implementar el protocolo CANopen es el principal problema que se presenta en la adquisición. Además, la adaptación de cada una de las señales provenientes de los sensores de presión se realiza en este nivel de programación. El proceso de implementación de CANopen implica el envío de distintos mensajes que configuran el sensor y que inician el proceso de envío de datos que contienen el valor del desplazamiento en cuentas que se encuentran en la escala de metros (ver apéndice).

La Figura (5.1) muestra los primeros mensajes que configuran el protocolo programada en LabVIEW.

La configuración del protocolo CANopen se realiza de la siguiente forma (ver Sección 3.3.1.2):

1. Se inicia la configuración del mensaje NMT enviando el mensaje “80 01” que indica que el mensaje es preoperacional (80) y que éste se dirige al nodo 01 (el sensor).

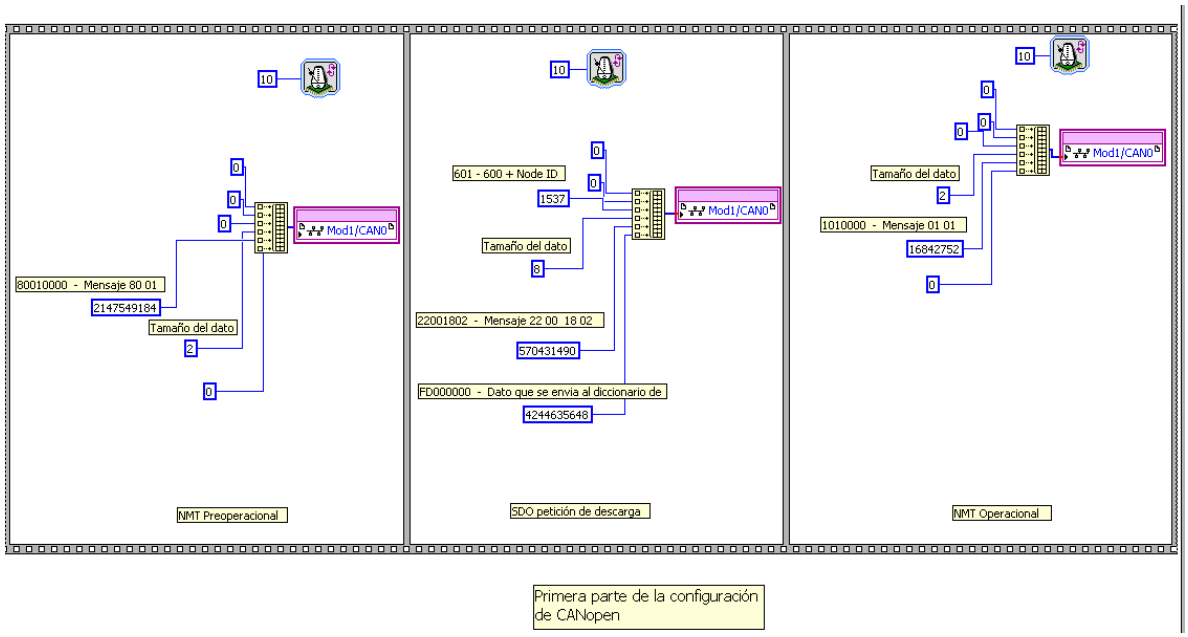


Figura 5.1: Configuración inicial del protocolo

2. Se envía el mensaje 22 00 18 02 que indica que es un mensaje SDO (22), y que se debe escribir un dato que es “FD” al diccionario de objetos con el índice (1800) y subíndice (02), que en el caso de el sensor que se utiliza en este trabajo estos índices corresponden al PDO1.

3. Una vez configurado el sensor se envía un mensaje para comenzar la operación del dispositivo. Es enviado un mensaje NMT 01 01 que hace que el nodo 01 se ponga en operación (01).

El proceso de configuración se realiza una vez con un retraso de 10[μs] puesto que el sensor no procesa operaciones a la misma velocidad del FPGA. En este momento es posible recibir mensajes PDO que están contenidos en el diccionario de objetos. Para poder realizar este proceso de forma continua es necesario configurar el proceso de transmisión de tal forma que el nodo 01 (el sensor) envíe los datos.

La Figura (5.2) muestra el proceso de envío de los datos de proceso (PDO’s) que corresponden al valor de la posición.

El proceso de envío de PDO’s se realiza de la siguiente forma:

1. Se envía el mensaje 10 00 00 00 para que el sensor transmita de forma remota y en forma continua, es decir que se envíe con la velocidad máxima y sin mensaje de confirmación.

2. Se escribe dentro del mensaje 181, para indicar que el PDO que se quiere leer es el PDO1 y el nodo es el 01.

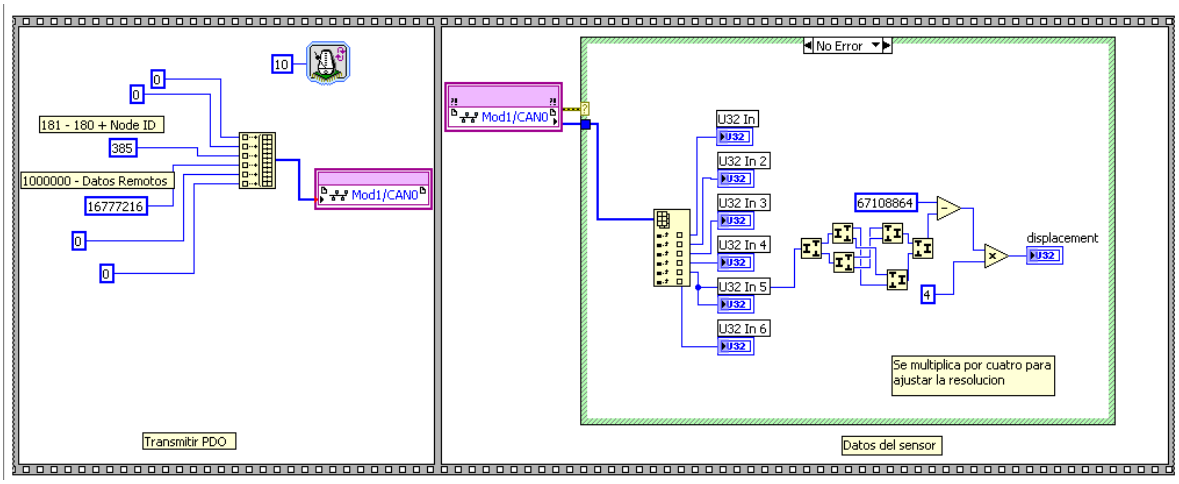


Figura 5.2: Configuración de la transmisión del los PDO

Este proceso de envío de PDO's se encuentra en un ciclo que se realiza con un tiempo de aproximadamente 190 [μs]. De esta forma es posible obtener finalmente la posición, ésta se encuentra dentro de la trama de datos que nos envía el sensor. Para obtener el dato de la posición se realiza un cambio de formato del dato de big-endian a little-endian [1]. Por último éste valor se multiplica por 4 debido a que la resolución del sensor es de 0.4 [mm].

Las operaciones de ajuste de las presiones (ver Sección 3.3.1.1) se realizan en el mismo ciclo donde se realiza el envío de PDO's puesto que éste proceso requiere únicamente 10 [μs]. La Figura (5.3) muestra el ajuste que se realiza con las presiones para obtener éstas en bares.

Además se muestra el cálculo de algunos valores necesarios que dependen de las presiones, como la fuerza diferencial que ejerce el pistón y las fuerzas 1 y 2 que son necesarias para calcular la fuerza diferencial y que es la base del funcionamiento del control PID no lineal.

5.1.2. Programación del controlador PID con CVI

Para analizar la viabilidad del programa utilizando C, pero ejecutándose en la PC, se realizó un programa que implementaba un controlador PID.

Con el uso de las herramientas que nos proporciona LabVIEW es posible generar una interfaz de programación de aplicaciones (API [2] por sus siglas en Inglés) que contiene las funciones necesarias en lenguaje C para leer los indicadores que se encuentran en el código de LabVIEW.

De esta forma es posible leer las señales de entrada y salida de forma similar en ambas

[1] Este cambio de formato se refiere a un cambio en el orden en el cual se ordenan los bytes en un dato de tipo entero sin signo de 32 bits

[2] Un API es un conjunto de funciones y métodos que ofrece cierta biblioteca para ser utilizado por otro software. En este caso el API es generado por LabVIEW para ser utilizado en C.

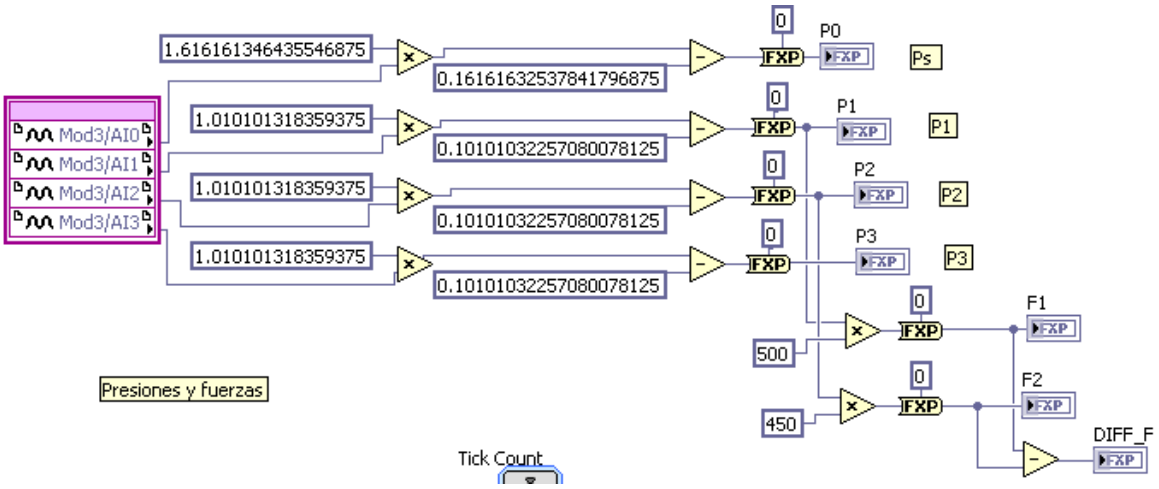


Figura 5.3: Acondicionamiento del valor de las presiones y obtención de los valores de fuerza

```
NiFpga_MergeStatus(&status, NiFpga_ReadI32(session,
                   NiFpga_IOCLIENT_CAN_IndicatorI32_Mod3AI0,
                   &presion0));
```

Figura 5.4: Lectura de la presión de suministro

plataformas (CVI y LabVIEW), es decir, el proceso de adquisición no cambia. En la Figura (5.4) se muestra la lectura de la presión utilizando las funciones que nos proporciona el API con código de C. De la misma forma se leen las señales restantes y el voltaje de control. Sin embargo este programa no se ejecuta en el cRIO, por lo cual el tiempo de muestreo que se obtiene en un programa en C y que se ejecuta en la PC, utilizando las funciones del API es de 4[ms]. Esto es debido a que la PC trasmite los datos por medio de ethernet primero al cRIO y después al FPGA utilizando el reloj del sistema operativo, en este caso Windows, cuya resolución máxima es de 1[ms].

La Figura (5.5) es la interfaz de usuario para el programa con el lenguaje C. El tiempo de muestreo es elevado considerando que el cRIO puede ejecutar procesos en el orden de [μs]. Por este motivo es necesario considerar otro lenguaje para implementar los controladores SMC, PID y SMC PID para que los controles se ejecuten completamente en el cRIO.

5.2. LabView 2009 y programas de entrada/salida

Una vez descartada la posibilidad de implementar el programa utilizando el lenguaje C se recurre a la programación en niveles como se planteó en el algoritmo de solución (ver Sección 2.3) y ahora utilizando el paradigma de programación en LabVIEW . Puesto que la etapa de adquisición se encuentra programada en LabVIEW desde un inicio, no es necesaria

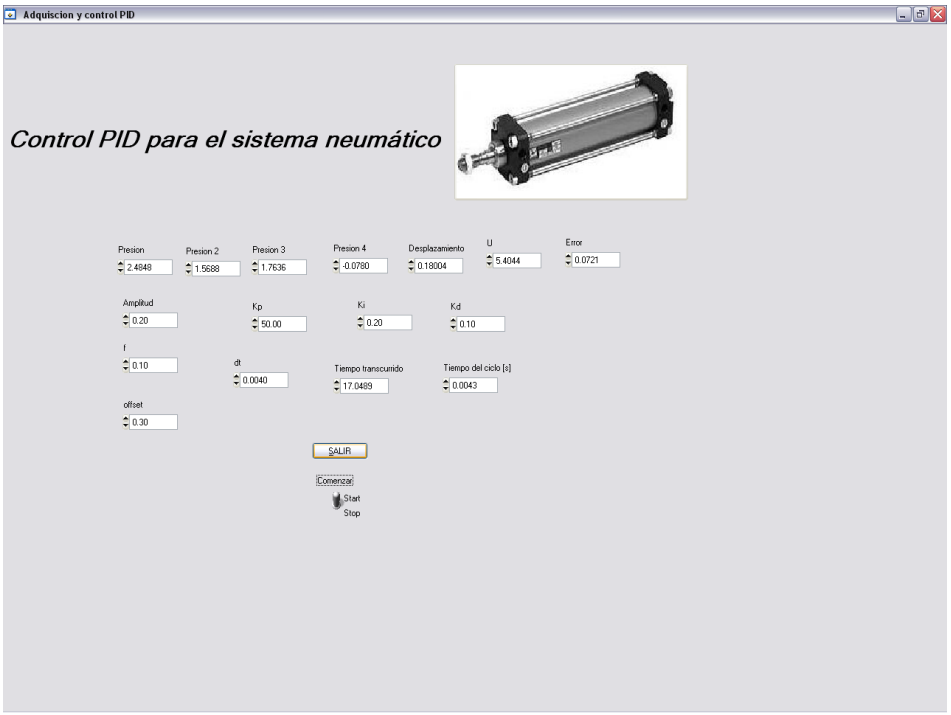


Figura 5.5: Programa para el controlador PID utilizando lenguaje C

una reprogramación del protocolo CANopen ni la lectura de presiones y escritura de voltaje. Por lo cual éste es idéntico al de las Figuras (5.1), (5.2) y (5.3).

5.2.1. Programación del microprocesador

En el caso de la programación con LabVIEW el proceso de implementación de los controladores se realiza por medio del algoritmo presentado en la Sección (2.3.2). Es decir, se adquieren las señales, se ejecuta el algoritmo de control, se escribe el voltaje y se guardan los datos. Cabe destacar que los procesos de control y adquisición se realizan en un ciclo separado al ciclo de grabación de datos. La Figura (5.6) muestra la secuencia de control y adquisición.

El ciclo de almacenamiento de datos se muestra en la Figura (5.7), junto con la secuencia que permite abrir y cerrar un archivo distinto cada vez que se inicia el control deseado.

En este punto es necesario hacer notar que los procesos de almacenamiento y ejecución del control son de naturaleza distinta. Mientras que el control es determinístico y se ejecuta en 1.5[ms], el almacenamiento es determinístico y se ejecuta en un ciclo con menor prioridad y a la máxima velocidad posible sin interferir el proceso de control. Como se observa en la Figura (5.6) el proceso es dividido en tres etapas: lectura de datos, ejecución del control y escritura del voltaje de control. Como se observa es posible implementar algoritmos de control adicionales simplemente sustituyendo el "subvi"[3] por otro utilizando las mismas entradas y

[3] SubVI se refiere a una funcion en código de LabVIEW donde VI se refiere a Instrumento Virtual por sus

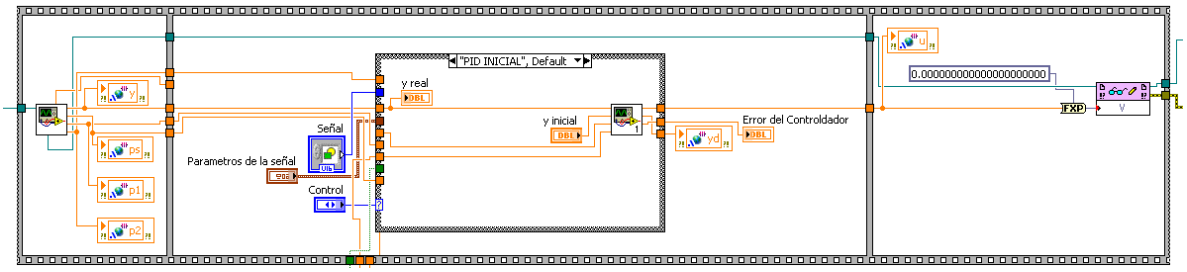


Figura 5.6: Control y adquisición

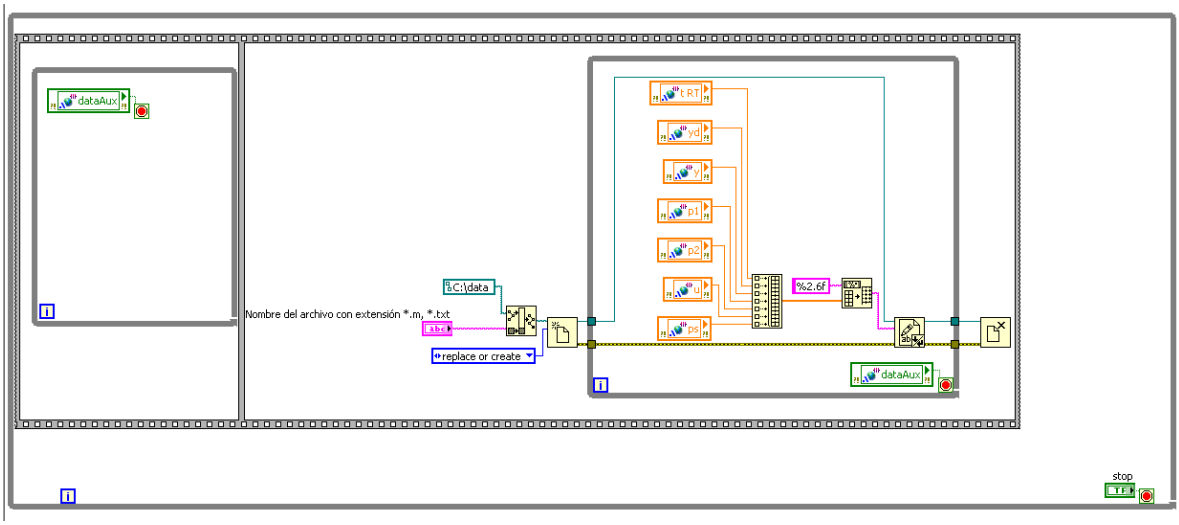


Figura 5.7: Ciclo de almacenamiento de datos

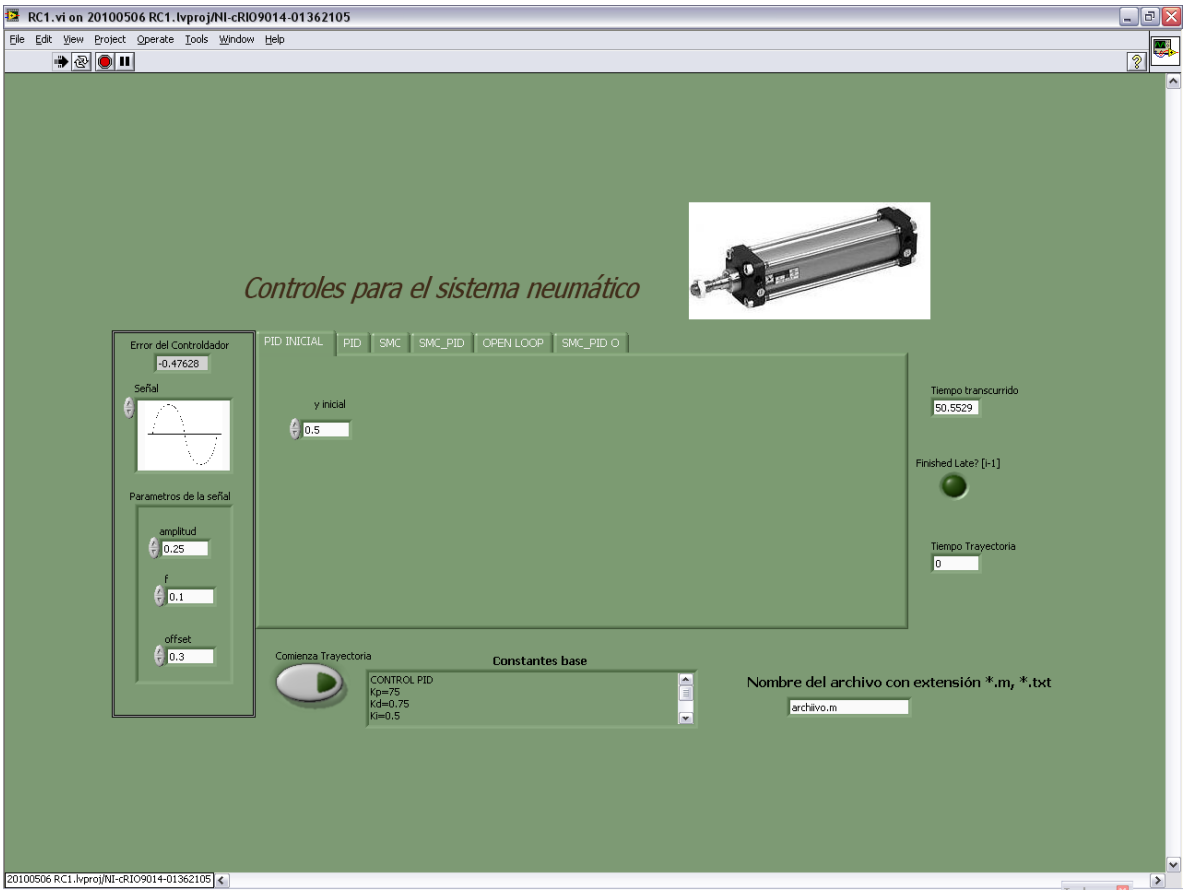


Figura 5.8: Vista general del panel

salidas.

5.3. Interfaz para el control del sistema neumático

El programa que se utilizó para la programación de controladores se muestra en la Figura (5.8). La interfaz contempla la elección de varios controladores, la elección de dos tipos de señal deseada (cuadrada o senoidal), así como la configuración de las características de la señal deseada.

5.3.1. Posición inicial

Cuando el programa se ejecuta, es posible llevar al pistón a una posición base o inicial (Figura 5.9). La posición inicial se establece con el control que aparece al seleccionar la pestaña

siglas en Inglés

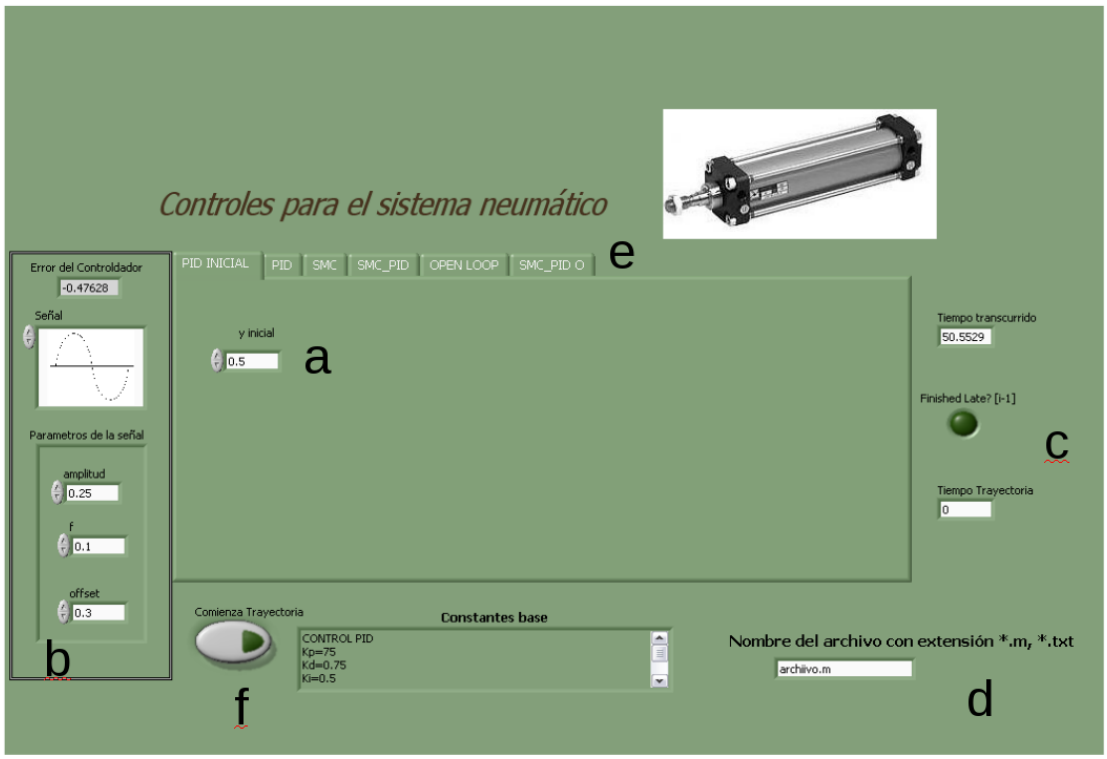


Figura 5.9: Panel con las partes que lo conforman

PID INICIAL (a), que se refiere a que esta posición inicial se obtiene utilizando un control PID, como el de la Sección (4.4) aplicado a un problema de estabilización. Cuando un experimento de seguimiento se realiza ésta posición inicial debe corresponder al valor del **offset** (b) para asegurarnos de que la trayectoria comience efectivamente en la posición inicial. Para realizar experimentos con cualquier control, el primer paso corresponde precisamente en llevar el pistón a la posición inicial.

Para las señales deseadas se debe considerar que la válvula únicamente puede seguir señales de 0.5[Hz] como máximo y que el desplazamiento máximo es de 0.76 [m]. Además debe considerarse que en el caso de la pestaña **OPEN LOOP** los valores de la amplitud se convierten en valores de voltaje y que el rango de voltaje es de ± 10[V].

5.3.2. Elección del controlador y configuración del archivo

Una vez hecha la configuración inicial para posicionar el pistón y ajustar la señal deseada se elije el control que se desea ejecutar. Los controles aparecen en cada una de las pestañas de la Figura (5.9) son: PID, SMC (Sliding Mode Control), SMC-PID (PID no lineal sin observador), OPENLOOP (Lazo abierto) y SMC-PID O (PID no lineal con observador) (e).

En las pestañas es posible elegir las ganancias relacionadas con cada uno de los controladores mencionados.

Para ejecutar el control deseado con almacenamiento de datos se debe escribir el nombre del archivo con una extensión "*.txt" o bien "*.m" (d) . Si no se desea almacenar datos únicamente se deja el espacio en blanco y se ejecuta el control. La ejecución del control se realiza pulsando el botón **Comienza Trayectoria**(f).

Es importante verificar el estado del indicador **Finished Late? [i-1]** (c) que muestra si el ciclo se está ejecutando con un tiempo de 1.5 [ms]. De la misma forma el indicador **tiempo transcurrido** muestra el tiempo que ha pasado desde la ejecución del programa, mientras que el indicador **tiempo trayectoria** muestra cuanto tiempo ha estado funcionando el controlador y la trayectoria deseada.

5.4. Resultados experimentales

Existía un versión anterior en la cual se había programado un control PID. En este trabajo se desarrollo una versión nueva y se implementaron los algoritmos PID, SMC y PID no lineal. En esta sección se muestran los resultados experimentales del programa anterior para realizar una comparación en el algoritmo en el cual esto es posible (PID). Además de lo anterior se presentarán los datos para las presiones de las cámaras del pistón y las señales de control, que también son producto de este trabajo.

5.4.1. PID

En las Figuras (5.10), (5.11) se presentan los resultados del controlador anterior implementado con un tiempo de muestreo de 10[ms]. En el primer caso, la trayectoria deseada es una senoidal con frecuencia de 0.1 [Hz] y amplitud de 0.3 [m]. Como se observa la señal no comienza en 0[s]. En la Figura (5.11) se muestra una senoidal con la misma amplitud que la otra pero con distinta frecuencia (0.05[Hz]) y con las mismas complicaciones que la anterior.

Para realizar la comparación se utilizó una señal senoidal de frecuencia de 0.1[Hz] y amplitud de 0.28[m]. Una vez desarrollada la nueva versión del algoritmo PID, se resolvieron los problemas que se tenian al inicio de la ejecución del programa, como se muestra en la Figura (5.12). En la Figura se observa que el seguimiento inicial no tiene cambios bruscos. El programa desarrollado resolvió el problema que se tenía al inicio cambiando las prioridades en que se ejecutaban los procesos.

En la Figura (5.13) se muestra una señal senoidal con una frecuencia de 0.2[Hz] y su comportamiento, en este caso no hubo comparación puesto que anteriormente éste tipo de pruebas no podían realizarse. Para este experimento se utilizaron como constantes $K_p = 100$, $K_i = 0.5$, $K_d = 0.05$.

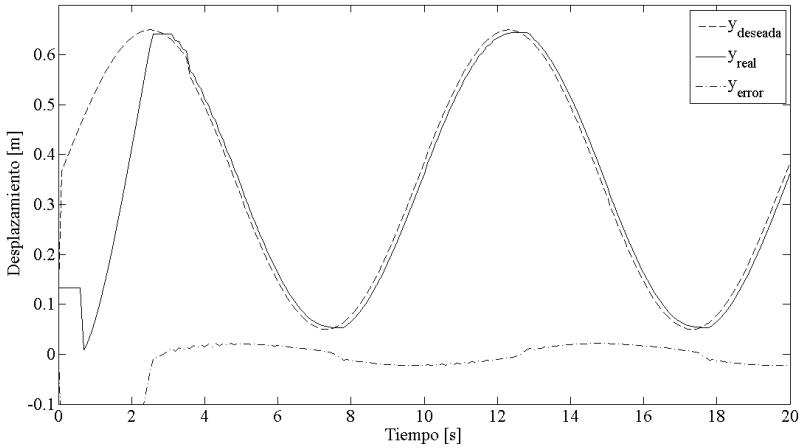


Figura 5.10: Resultados del PID con el programa anterior, frecuencia 0.1[Hz], amplitud 0.3[m]

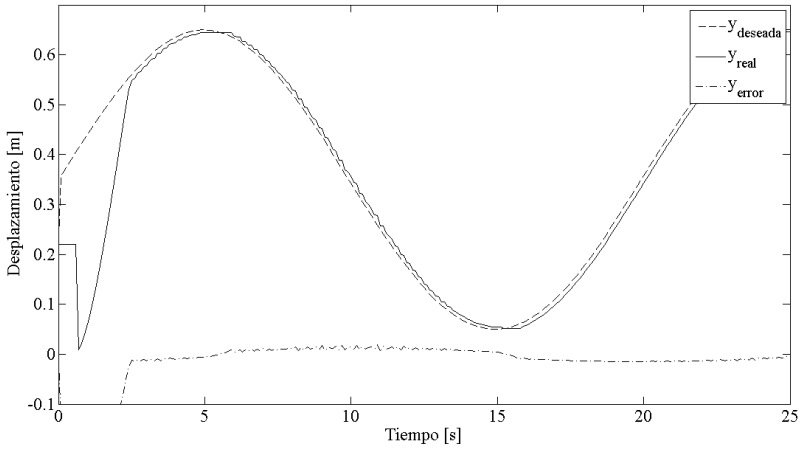


Figura 5.11: Resultados del PID con el programa anterior, frecuencia 0.05[Hz], amplitud 0.3[m]

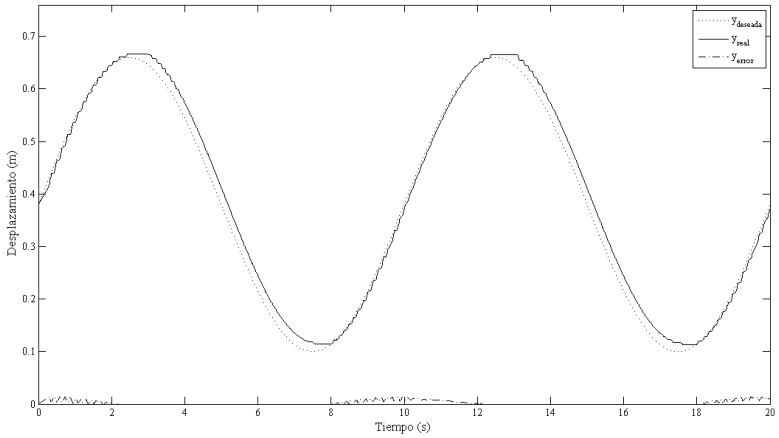


Figura 5.12: Seguimiento en el control PID, frecuencia 0.1[Hz], amplitud 0.28[m]

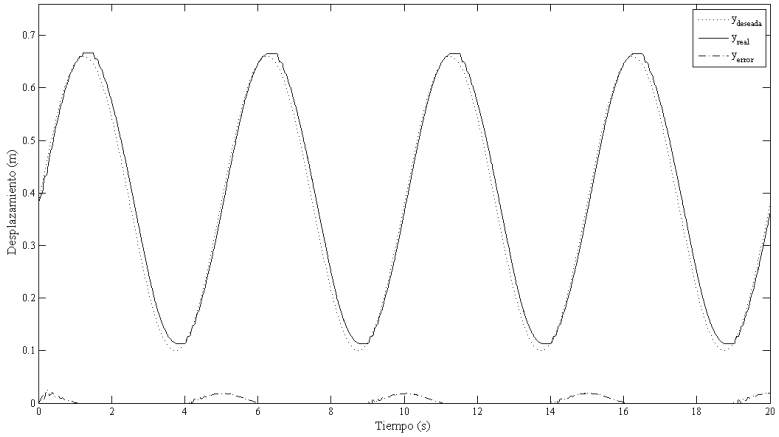


Figura 5.13: Seguimiento en el control PID, frecuencia 0.2[Hz], amplitud 0.28[m]

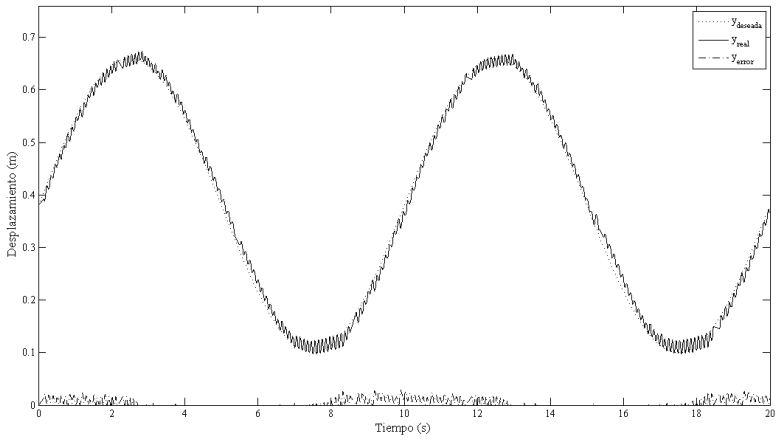


Figura 5.14: Seguimiento en el control SMC, 0.1[Hz] y amplitud de 0.28[m]

5.4.2. Control Por Modos deslizantes

En este apartado se presentan los resultados obtenidos en la implementación del algoritmo de control por modos deslizantes (SMC) presentado en la Sección (4.5). En la Figura (5.14) se muestra el comportamiento del control para una señal de seguimiento senoidal de 0.1[Hz] y amplitud de 0.28[m]. Las constantes que se utilizaron son las siguientes: $\lambda = 2500$, $k_s = 120$ y $\phi = 4$.

Como en el caso del control PID también se muestra una señal senoidal de frecuencia 0.2[Hz] (Figura 5.15). En este caso las constantes se fijaron en $\lambda = 3500$, $k_s = 120$ y $\phi = 4$.

5.4.3. Control PID no lineal

En este apartado se presentan los resultados experimentales para el control PID no lineal presentado en la Sección (4.6). Las constantes ajustadas en este caso son (para la señal senoidal): $K_{\beta f} = 5$, $K_{\beta y} = 150$, $K_{\gamma f} = 0.01$, $K_{\gamma y} = 0.03$,$K_{py} = 50$, $\lambda_y = 200$, $\lambda_z = 125$, $K_d = 350$, $K_t = 1$ y $K_{pf} = 0.025$. En la Figura (5.16) se puede apreciar el comportamiento de este algoritmo de control.

Para la senoidal de frecuencia 0.2[Hz] se utilizaron las siguientes constantes: $K_{\beta f} = 5$, $K_{\beta y} = 150$, $K_{\gamma f} = 0.01$, $K_{\gamma y} = 0.03$, $K_{py} = 95$, $\lambda_y = 200$, $\lambda_z = 125$, $K_d = 350$, $K_t = 1$ y $K_{pf} = 0.025$. La Figura (5.17) muestra el comportamiento del control para esta señal.

Adicionalmente al seguimiento y el correspondiente error, es necesario presentar el comportamiento de las presiones de las cámaras para cada una de los algoritmos empleados. Estas señales forman parte de las ocho señales que se obtienen de la grabación de datos.

Para la señal senoidal de 0.1[Hz], las presiones de las cámaras se muestran en la Figura (5.18) en la cual la primera columna corresponde a la presión de la cámara 1 y la segunda a la presión de la cámara 2. Las constantes en cada caso corresponden a las mencionadas en cada algoritmo de control. En el caso de la onda senoidal de 0.2[Hz] las presiones de cada cámara

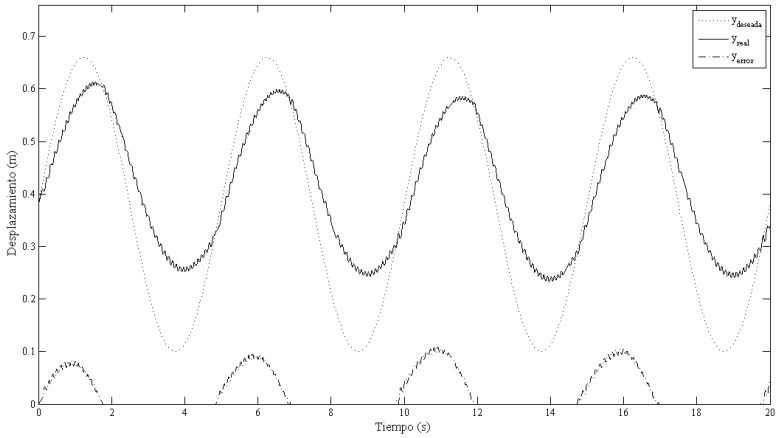


Figura 5.15: Seguimiento en el control SMC, 0.2[Hz] y amplitud de 0.28[m]

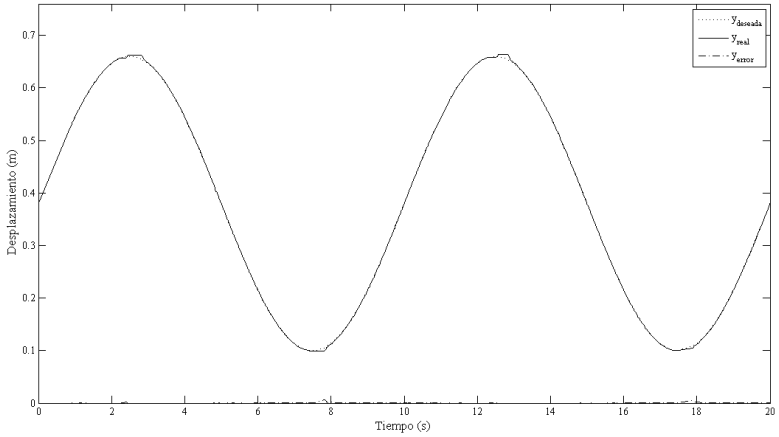


Figura 5.16: Seguimiento en el control PID no lineal, 0.1[Hz] y amplitud de 0.28[m]

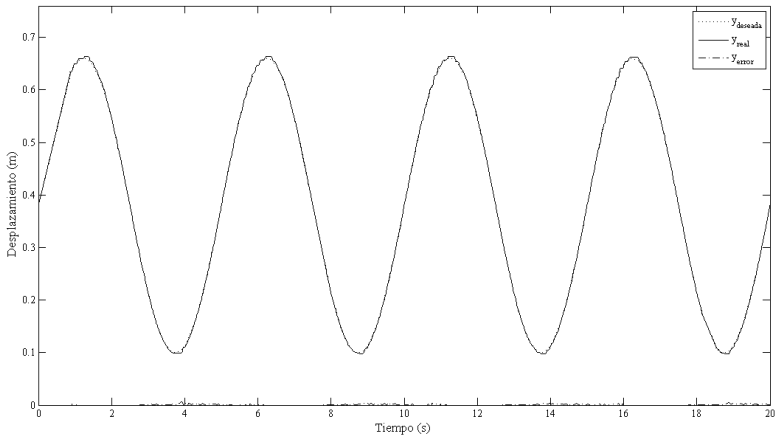


Figura 5.17: Seguimiento en el control PID no lineal, 0.2[Hz] y amplitud de 0.28[m]

se presentan en la Figura (5.19).

Finalmente, como parte importante del proceso de estudio de los algoritmos de control es importante presentar el comportamiento de la señal de control. En las Figuras (5.20), (5.21) se presentan las gráficas para la señal senoidal de 0.1[Hz] y 0.2[Hz] respectivamente. Como se observa las señales de control están en el rango de 0-10 [V] esto es debido al comportamiento de la electroválvula (Sección 3.1.1). Se eligió el rango 0-10[V] en lugar del -10-0 [V] puesto que el comportamiento es el mismo salvo un cambio en el “sentido” del flujo del aire.

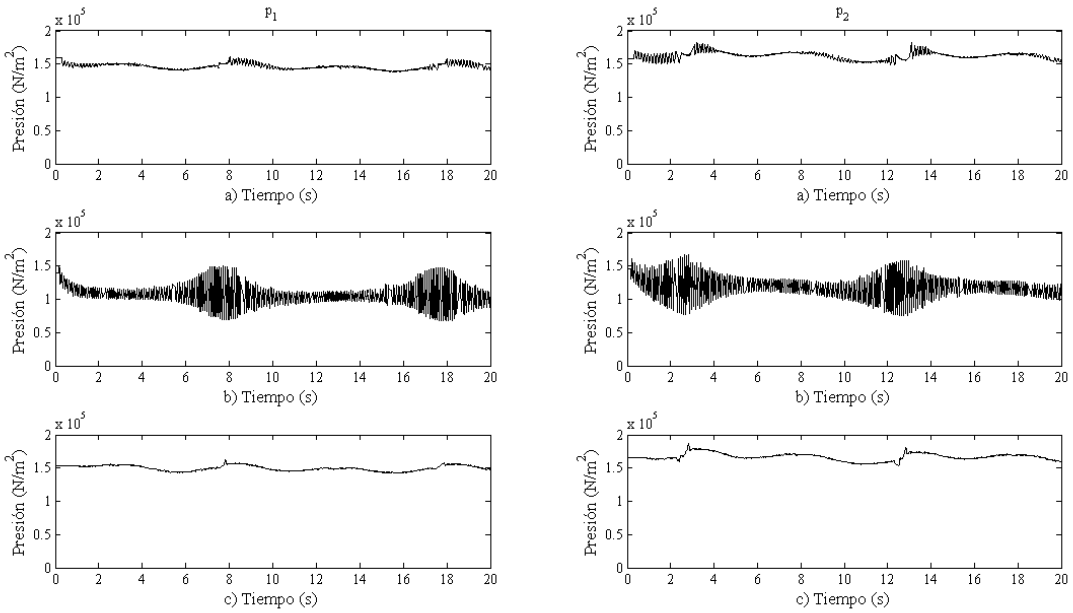


Figura 5.18: Comportamiento de las presiones para la senoidal, 0.1[Hz]. a) PID , b) SMC y c) PID no lineal

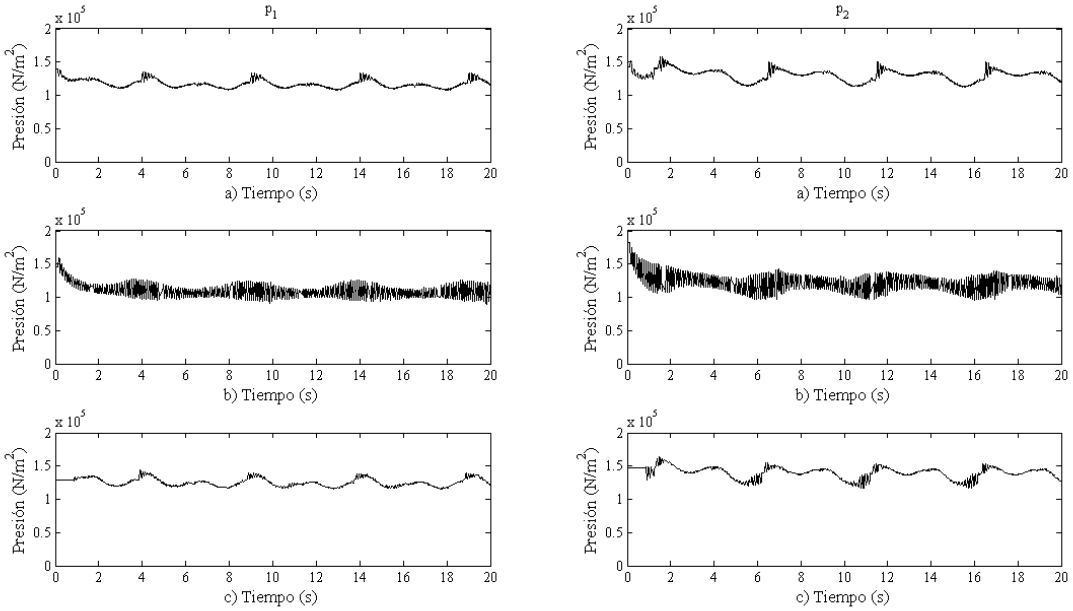


Figura 5.19: Comportamiento de las presiones para la senoidal, 0.2[Hz]. a) PID , b) SMC y c) PID no lineal

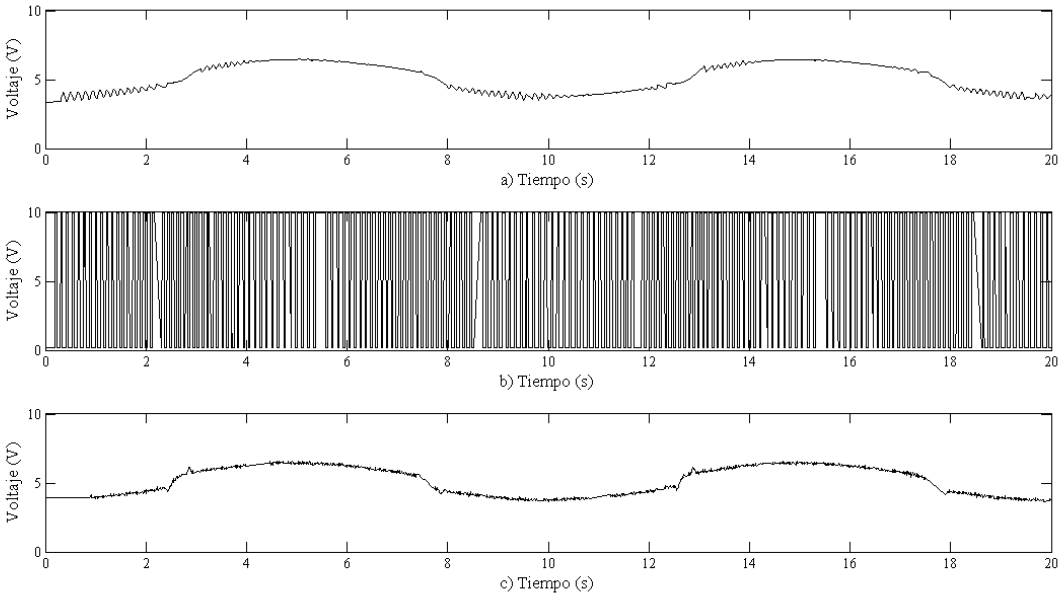


Figura 5.20: Señal de control para la onda senoidal de 0.1[Hz]. a) PID , b) SMC y c) PID no lineal

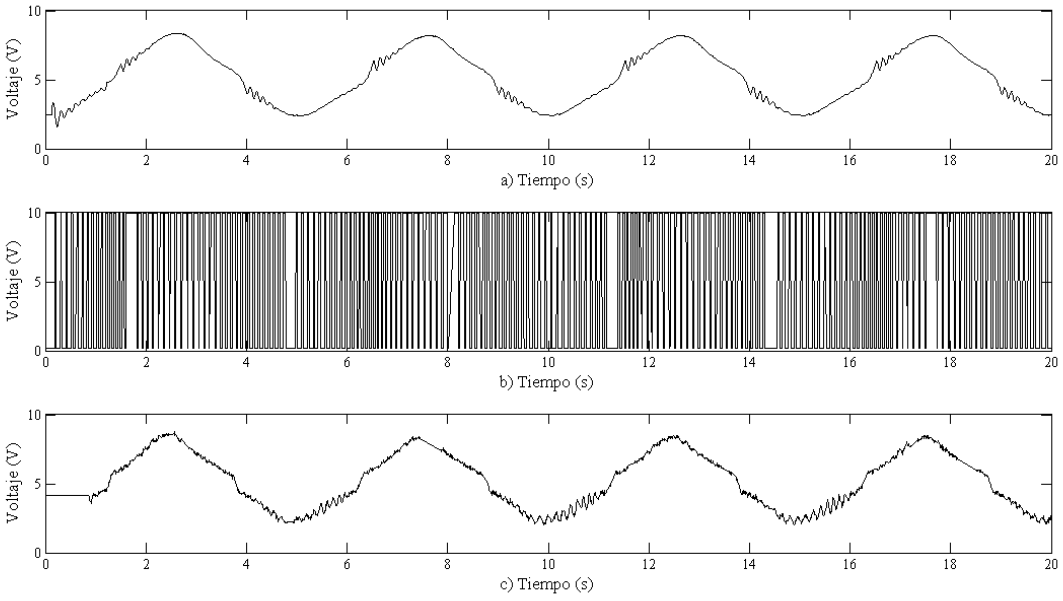


Figura 5.21: Señal de control para la onda senoidal de 0.2[Hz]. a) PID , b) SMC y c) PID no lineal

Capítulo 6

Conclusiones

En este trabajo se desarrolló un programa que implementó controles que requieren un periodo de muestreo bajo para su implementación. Se cambio el orden de prioridad de los procesos de control y almacenamiento de datos, se programó el protocolo CANopen y se implementaron tres algoritmos de control para el sistema neumático no lineal. En el desarrollo del programa para la implementación se lograron resolver los siguientes problemas:

Periodo de muestreo. Se disminuyó en una proporción de 2 a 1, esto es de 3[ms] a 1.5[ms].

Almacenamiento. Fue posible guardar datos de todas las señales que forman parte del sistema, aunque no todas se utilizaron en el proceso de control.

Interfaz. Se desarrolló una interfaz gráfica para el usuario.

En la comparación con la aplicación anterior, ésta se mejoró notablemente ya que con el periodo de muestreo de 3[ms] no era posible hacer el almacenamiento de datos, puesto que la aplicación se caía y se necesitaba aumentar el tiempo de muestreo a 10[ms] para lograr guardar los datos. Si se compara éste comportamiento con el actual, la mejora en el tiempo de muestreo es aproximadamente de 6 a 1.

Aunado a la mejora en el periodo de muestreo se lograron implementar tres algoritmos de control y se desarrolló una plataforma versátil que permite implementar otros algoritmos simplemente añadiendo un bloque de programación de forma análoga a como se realizaría en SIMULINK , dejando las bases para poder experimentar con otros algoritmos en trabajos de investigación futuros del Laboratorio de Robótica; cubriendo uno de los objetivos que se planteó en este trabajo. La implementación de los controladores se realizó a partir del programa desarrollado y simulado en MATLAB contenido en el archivo *.m y utilizando la mayoría del código, para establecer un proceso que se pueda repetir en caso de que se quiera utilizar la interfaz para experimentar con otros algoritmos de control de tal forma que no sea necesario el desglose del funcionamiento de la aplicación.

Uno de los principales problemas con lo que se comenzó el desarrollo de la aplicación de control fue el mejoramiento en la implementación del protocolo CANopen para lograr utilizar todo el potencial que implica la utilización de los sensores basados en éste protocolo. Para

lograrlo se rediseñó la programación en el nivel más bajo del modelo de programación, es decir, la adquisición de las señales. En este nivel se realizó toda la adquisición aprovechando las ventajas que implica la utilización del FPGA del cRIO. La posición que se obtiene por medio de CANopen se logró implementar en un tiempo de muestreo de 190[μs] y en conjunto con la obtención de las señales de presión y la escritura del voltaje de control el proceso no consumió más de 200[μs]. Esto dio como resultado un programa mejor estructurado y más eficiente considerando las limitantes del nivel FPGA, esto es, la dificultad de realizar operaciones complejas características de los algoritmos de control más avanzados.

La realización de la aplicación utilizando el lenguaje de programación C resultó desfavorable en gran medida por las limitaciones que impone el fabricante del hardware, puesto que para lograr implementar una aplicación utilizando el lenguaje C era necesario desarrollar controladores o bien una interfaz que nos permitiera descargar los programas escritos en C directamente al sistema operativo del cRIO (Vx Works). La otra posibilidad era adquirir software diseñado para realizar lo anterior pero que implicaba un aumento en el precio del proyecto lo que no concordaba con los objetivos del trabajo y por lo cual se desechó, además de que la disminución del tiempo de muestreo no estaba garantizada. La aplicación que se desarrolló en C a pesar de tener un tiempo de muestreo relativamente bajo (4[ms]), era ejecutada en la PC desperdiciando las capacidades del cRIO por lo cual también se desechó como posibilidad pero fue de gran ayuda para el replanteamiento del paradigma de programación y el entendimiento de los límites del equipo.

Para la experimentación se partió siempre del funcionamiento del algoritmo de control PID, para todos los casos. Una vez montado el control PID se implementaron los demás y las pruebas de funcionamiento del programa se realizaron utilizando este control y bajo el lenguaje de programación LabVIEW. Partiendo del hecho conocido de las limitantes de LabVIEW se diseñó un programa que conjuntara algunas características de la programación en C y las de la programación paralela de LabVIEW. Utilizando éste hecho se utilizaron dos tipos de procesos: deterministicos de baja y alta prioridad. El primero corresponde al proceso de control y el segundo al almacenamiento de datos. En conjunto fue posible relacionar el hecho de que a medida que aumentaba el número de señales almacenadas el tiempo del proceso deterministico aumentaba y por lo cual se hizo necesario decidir en qué medida era importante monitorear señales y controlar. El resultado de lo anterior fue que se almacenaron todas la señales del sistema(8 señales) y se obtuvo un periodo de muestreo de 1.5[ms].

En cuanto a la eficacia de los controladores, el controlador denominado *PID no lineal* fue el de mejor desempeño como se observa en las gráficas comparativas de los voltajes de control y en los errores de seguimiento que para el caso senoidal no superó los 5[mm]. Debido a las limitantes de la electroválvula el control por modos deslizantes no fue adecuado en este sistema por las complicaciones que presenta este tipo de algoritmo (chattering). El control PID clásico, con errores del orden de 1[cm] mostró su simplicidad y fácil programación, sin embargo al ser el sistema altamente no lineal también demostró las deficiencias del mismo. El monitoreo de todas las señales ayudó en gran medida a conocer mejor las variaciones tanto de las presiones de las cámaras como la presión de suministro que se supuso constante para la simulación, aunque en realidad no lo fue.

Apéndice A

CAN y CANopen

1.1. CAN

Las redes de campo desde el punto de vista del modelo de redes OSI usualmente sólo tienen la capa 1 (Capa Física), 2 (Capa de Enlace) y la 7 (Capa de Aplicación) implementada.

El bus de campo CAN (Controller Area Network) define únicamente las capas 1 y 2 (ISO11898); es decir éstas están completamente incluidas en el hardware CAN, lo que reduce de forma significativa la dificultad en la implementación para desarrolladores del firmware de un nodo.

Sin embargo es necesario un protocolo de alto nivel para definir de qué forma son usados el identificador de 11-bit del mensaje CAN y los 8 bytes de datos. Para ello se definió el protocolo CANopen, que está basado en CAN e implementa la capa de aplicación. Actualmente está ampliamente extendido y ha sido adoptado como un estándar internacional.

La construcción de sistemas basados en CAN que puedan garantizar la interoperabilidad entre dispositivos de diferentes fabricantes requiere una capa de aplicación y unos "perfiles" que estandaricen la comunicación en el sistema, la funcionalidad de los dispositivos y la administración del sistema:

Capa de aplicación (Application Layer) . Proporciona un conjunto de servicios y protocolos de dispositivos de red.

Perfil de comunicación (Communication Profile) . Define cómo configurar los dispositivos y los datos y la forma de intercambiarlos entre ellos.

Perfiles de dispositivos (Device Profiles) . Añade funcionalidad específica a los dispositivos.

La relación entre el modelo OSI y los estándares CAN y CANopen se muestran en la Figura (A.1).

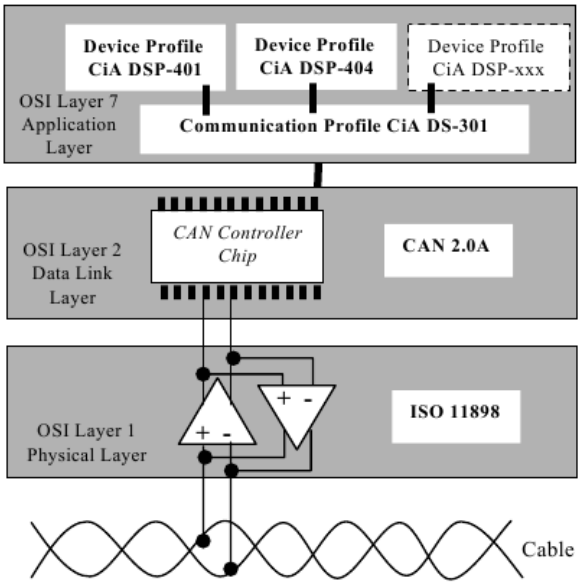


Figura A.1: Relación entre los protocolos OSI, CAN y CANopen. CANopen se encuentra en el nivel 7 que es la capa de aplicación y está contenido en un estándar denominado CiA DS-301

1.2. CANopen

CANopen define una capa de aplicación y un perfil de comunicación basado en CAN [in~Automation, 2002]. CANopen define los siguientes objetos de comunicación (mensajes):

- Objeto de datos de proceso o PDO(Process Data Object)
- Objeto de datos de servicio o SDO(Service Data Object)
- Objeto de gestión de red o NMT(Network Managment Object)
- Objeto de funciones especiales (SYNC, EMCY, TIME)

1.2.1. Diccionario de objetos

El diccionario de objetos es la interfaz entre el programa de aplicaciones y el dispositivo de comunicación (Figura A.2). Describe completamente y de forma estandarizada la funcionalidad de cada dispositivo y permite su configuración mediante mensajes (SDO) a través del propio bus. Cada objeto se direcciona utilizando un índice de 16 bits. Para permitir el acceso a elementos individuales de las estructuras de datos también existe un subíndice de 8 bits. En el Cuadro (A.1) podemos ver la estructura general del diccionario.

El rango relevante de objetos va desde el índice 1000 al 9FFF. Para cada nodo de la red existe un diccionario de objetos (OD por sus siglas en inglés), que contiene todo los parámetros que describen el dispositivo y su comportamiento en la red.

Índice	Objeto	Descripción
0x0000	No usado	
0x0001 - 0x001F	Tipos de dato estáticos	Contiene definiciones de tipos de dato estándar como boolean ,enteros, string, punto flotante , etc. Se incluyen como referencia, no pueden ser leídas ni escritas.
0x0020 - 0x003F	Tipos de dato complejo	Contiene definiciones de estructuras predefinidas, compuestas de tipos estáticos, comunes a todos los dispositivos.
0x0040 - 0x005F	Tipos de dato especificas del fabricante	Contiene definiciones de estructuras predefinidas, compuestas de tipo estático, específicas del dispositivo en particular.
0x0060 - 0x007F	Tipos de dato específicos del perfil del dispositivo	Definiciones de dato básicos específicos para el perfil del dispositivo.
0x0080 - 0x009F	Tipos de dato complejos específicos del perfil del dispositivo	Definiciones de estructuras específicas para el perfil del dispositivo.
0x00A0 - 0x0FFF	Reservado	
0x1000 - 0x1FFF	Rango para el perfil de comunicaciones	Contiene parámetros de configuración del bus CAN. Estas entradas del diccionario son comunes a todos los dispositivos
0x2000 - 0x5FFF	Rango para el perfil específico del fabricante	Contiene las extensiones al perfil estándar, realizadas por el fabricante.
0x6000 - 0x9FFF	Rango para perfiles de dispositivos estandarizados	Contiene todos los objetos de datos comunes a un tipo de perfil que pueden ser leídos o escritos por la red. Algunas de las entradas son obligatorias (funcionalidad requerida) mientras que otras son opcionales (funcionalidad opcional).
0xA000 - 0xFFFF	Reservado	

Cuadro A.1: Estructura general de un diccionario en CANopen

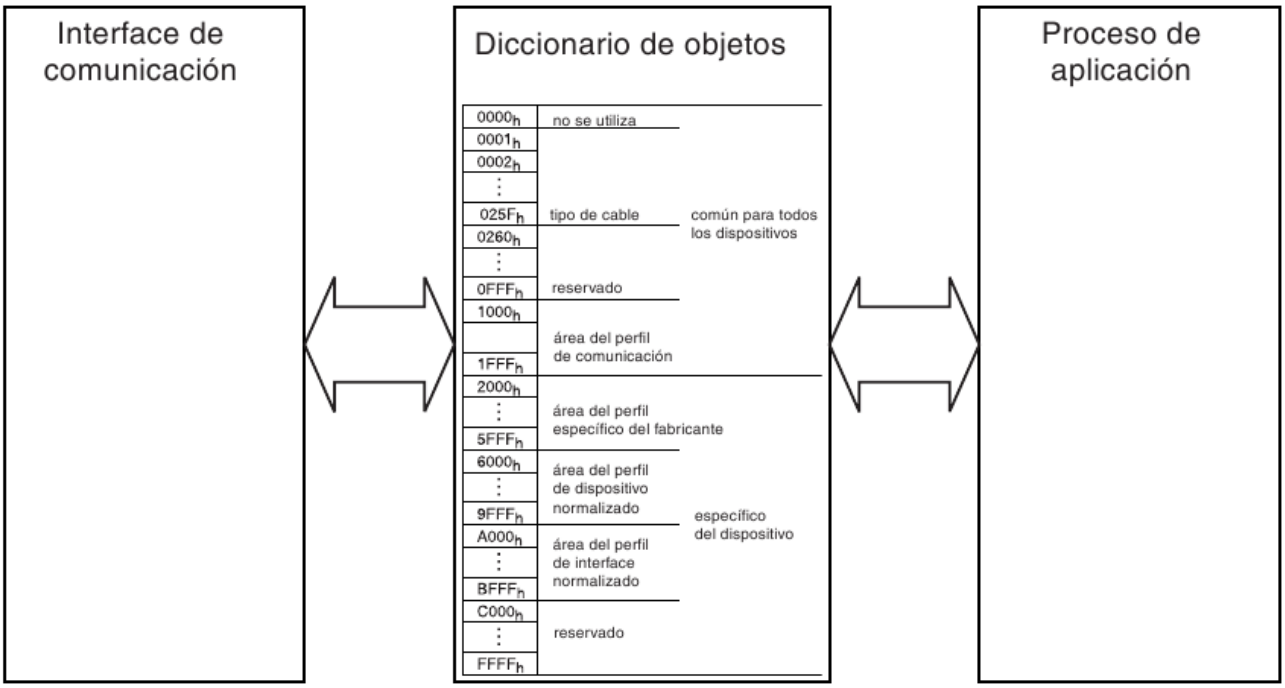


Figura A.2: Interacción del diccionario de objetos en los procesos y los espacios de memoria que se muestran en el Cuadro A.1

← Número de bit										
10	9	8	7	6	5	4	3	2	1	0
Function code				Node-ID						

Cuadro A.2: Estructura del identificador de mensajes CAN

En CANopen hay documentos que describen perfiles. Hay un perfil de comunicaciones (communication profile) donde están descritos todos los parámetros relacionados con las comunicaciones. Además hay varios perfiles de dispositivos (device profiles) donde se definen los objetos de un dispositivo en particular.

Un perfil define para cada objeto del diccionario su función, nombre, índice, subíndice, tipos de datos, si es obligatorio u opcional, si es de "sólo lectura", "sólo escritura" o "lectura-escritura", etc.

1.2.2. Identificadores de mensaje

CANopen define la distribución de los identificadores de mensaje (Predefined Connection Set) de manera que hay un mensaje de emergencia por nodo, mensajes de sincronización y time stamp, un SDO (ocupando dos identificadores) , mensajes NMT y cuatro PDOs de transmisión y cuatro de recepción por dispositivo.

El identificador de once bits se divide en dos partes (Cuadro A.2):

1. 4 bits para el código de función
2. 7 bits para el identificador de nodo (Node-ID)

1.2.3. Objeto de datos de proceso (PDO)

Los objetos de datos de proceso se utilizan para transmitir rápidamente datos de proceso. Un PDO puede transportar una carga útil de 8 bytes, que es la carga máxima de una trama CAN. La transmisión de un PDO utiliza el modelo de productor y consumidor de CAN ampliado mediante transferencias síncronas. La transferencia síncrona de PDO se basa en la transferencia de mensajes de SYNC en el bus CAN.

Un PDO se envía en modo cíclico tras un número configurable (de 1 a 240) de mensajes SYNC recibidos. También es posible esperar a que se encuentren disponibles datos del proceso de aplicación y enviar un PDO tras el siguiente mensaje de SYNC recibido. Esto se denomina transferencia síncrona acíclica.

1.2.4. Objeto de datos de servicio (SDO)

Los objetos de datos de servicio tienen con fin la transmisión de parámetros. Los SDO ofrecen acceso al diccionario de objetos de los dispositivos remotos. Un SDO no tiene límites de longitud. Si la carga útil no cabe en la trama CAN, se dividirá en varias tramas CAN.

Se acusa el recibo de cada SDO. La comunicación de SDO emplea una comunicación de homólogo a homólogo, con uno de los homólogos que actúa como servidor y el otro como cliente.

1.2.5. Objetos de gestión de red (NMT)

Los objetos de gestión de red o NMT (network management objects) cambian el estado o supervisan el estado de un dispositivo CANopen. Un mensaje de NMT es un mensaje con identificador 0 de CAN. Esto otorga al mensaje NMT la máxima prioridad posible. El mensaje NMT siempre consta de una carga útil de dos bytes en la trama CAN. El segundo byte contiene el ID del nodo al que se dirige. Es posible dirigirse a todos los dispositivos con un mensaje NMT con el ID 0 de nodo reservado. El primer byte codifica el comando NMT.

Un dispositivo CANopen comienza el estado de Inicialización tras encender la alimentación. A continuación, el dispositivo realiza su inicialización. Cuando el dispositivo ha completado su inicialización, emite un objeto NMT de arranque para notificar al maestro. Además existe un protocolo denominado **Heartbeat** que permite saber el estado de un nodo y éste se implementa con objetos NMT.

1.2.6. Objetos de funciones especiales

CANopen puede contar con un productor de SYNC para sincronizar las acciones de los nodos de CANopen. Un productor de SYNC emite (periódicamente) el objeto SYNC. El objeto SYNC contiene el identificador CAN 128. Esto puede introducir algo de fluctuación, debido a la prioridad de este mensaje.

Un error interno de dispositivo puede activar un objeto de emergencia o EMCY (emergency). La reacción de los consumidores de EMCY depende de la aplicación. El estándar CANopen define varios códigos de emergencia. El objeto EMCY se transmite en una sola trama de CAN con ocho bytes.

Una trama CAN con el ID 256 de CAN y con una carga útil de seis bytes se puede utilizar para transmitir la hora del día a varios nodos de CANopen. Este objeto de marcaje de tiempo (TIME) contiene el valor de la hora en el tipo de datos de hora del día o TOD (Time-of-Day).

Apéndice B

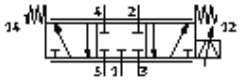
Especificaciones del hardware

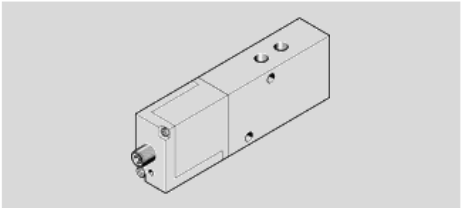
En las siguiente páginas se incluyen las especificaciones del hardware que son necesarias para conocer las limitantes de cada dispositivo. Se incluyen las especificaciones por orden de aparición de la electroválvula, del sensor de presión y del sensor de desplazamiento.

Data sheet: Proportional directional control valve MPYE-5-1/8-HF-010-B – 15169

FESTO

Function





Feature	values
Nominal size	6 mm
Type of actuation	electrical
Sealing principle	Hard
Assembly position	Any
Design structure	Piston slide
Type of reset	Magnetic spring
Safety instructions	MPYE safety position: If the power supply cable is interrupted, travel occurs to the closed mid-position
Type of piloting	direct
Flow direction	non reversible
Valve function	5/3 closed
Polarity protected	for all electrical connections
Working pressure	0 - 10 bar
b value	0.26
C value	3.1 l/sbar
Standard nominal flow rate	700 l/min
Max. hysteresis	0.4 %
Operating voltage range DC	17 - 30 V
Residual ripple	5 %
SETPOINT/ACTUAL values	Voltage type 0 - 10 V
Operating medium	Filtered, unlubricated compressed air, 5 µm filtration
CE symbol (see declaration of conformity)	according to EU-EMV guideline
Corrosion resistance classification CRC	2
Medium temperature	5 - 40 °C
Protection class	IP65
Ambient temperature	0 - 50 °C
Authorization	C-Tick
Product weight	330 g
Electrical connection	4-pin M12x1 Plug Round design
Mounting type	with through hole
Pneumatic connection, port 1	G1/8
Pneumatic connection, port 2	G1/8
Pneumatic connection, port 3	G1/8
Pneumatic connection, port 4	G1/8
Pneumatic connection, port 5	G1/8
Materials information for cover	ABS coated
Materials information for seals	NBR
Materials information, housing	Aluminum Anodized

SDET Pressure Sensor

All-Fluid Sensor

Universal Pressure Sensor
The SDET pressure sensor is universal due to its flexible applications; the ease and quickness of installation; and its excellent price to performance ratio.

SDET sensors can be used to monitor the pressure of all kinds of gaseous and liquid mediums, including compressed air. Possible pressure measuring ranges are from -1 ... 1 bar to 0 ... 100 bar, depending on the variant chosen.

Design Features
- Stainless-steel housing
- Mediums-resistant Viton seals
- Integrated ceramic sensor element with thick-film technology

Faster Installation
The installation is quicker and easier due to industry-standard interfaces such as:
- G1/4 threaded connections
- 4-pin M12 plugs
- Standard analog outputs
- Included interlinking mounting brackets

Total Application Flexibility
The sensors are resistant to water and oils, which makes them ideal for applications in all industries, especially:
- Process industry
- Automation technology
- Automobile industry
- Food and packaging industry

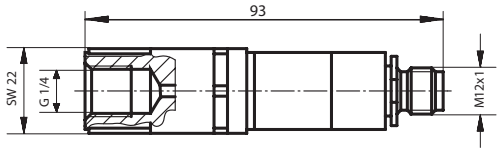
For more information: **www.festo.com/us/sensors/sde**

SDET.PSI.US

Product Short Information

SDET Pressure Sensor

Dimensions



All dimensions in millimeters

Technical Data																		
Part No.		547476	547478	547480	547482	547484	547486	547488	547490	547475	547477	547479	547481	547483	547485	547487	547489
Type		SDET-22T-B2-G14-U-M12	SDET-22T-D2-G14-U-M12	SDET-22T-D6-G14-U-M12	SDET-22T-D10-G14-U-M12	SDET-22T-D16-G14-U-M12	SDET-22T-D25-G14-U-M12	SDET-22T-D50-G14-U-M12	SDET-22T-D100-G14-U-M12	SDET-22T-B2-G14-I-M12	SDET-22T-D2-G14-I-M12	SDET-22T-D6-G14-I-M12	SDET-22T-D10-G14-I-M12	SDET-22T-D16-G14-I-M12	SDET-22T-D25-G14-I-M12	SDET-22T-D50-G14-I-M12	SDET-22T-D100-G14-I-M12
Pressure measuring range Start value	[bar]	-1	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0
Pressure measuring range Full-scale value	[bar]	1	2	6	10	16	25	50	100	1	2	6	10	16	25	50	100
Overload range	[bar]	5	5	10	20	40	40	100	200	5	5	10	20	40	40	100	200
Bursting pressure	[bar]	6	6	12	25	50	50	120	250	6	6	12	25	50	50	120	250
Accuracy	[% FS]	1.5	1.5	1.0	1.0	1.0	1.0	1.0	1.0	1.5	1.5	1.0	1.0	1.0	1.0	1.0	1.0
Analog output		0.1 - 10 V (3-wire)								4 - 20 mA (2-Wire)							
Load resistance		RL >10 k								RL <= (UB-8 V)/0.02 A							
Operating voltage UB		14 - 30 V DC								8 - 30 V DC							
Idle current		Max. 4 mA								-							
Residual ripple		10% UB															
Linearity error BFSL		0.40% FS															
Repeatability of analog value FS		0.25% FS															
Long-term drift		0.3% FS per year															
Ambient temperature		0 - 80°C															
Medium temperature		-10 - 100°C															
Operating temperature		-20 - 85°C															
Temperature coefficient Zero signal		Typically 0.2% FS, max. 0.4% FS /10K (pressure measuring range 0...16 bar, typically 0.25% FS, max. 0.5% FS/10K)															
Temperature coefficient Output range		Typically 0.15% FS, max.0.25% FS/10K															
CE symbol		In accordance with EU EMC Directive															
Short-circuit protection		Yes															
Polarity reversal protection		Within operating voltage range															
Degree of protection		IP65															
Sensor element material		Al203 96%															
Sealing ring material		FKM (Viton)															
Housing material		CrNi steel 1.4305															
Product weight		100 g															

Festo Corporation

For ordering assistance, or to find your nearest Festo Distributor,
Call: 1.800.99.FESTO
Fax: 1.800.96.FESTO
Email: customer.service@us.festo.com
Visit: www.festo.com/usa

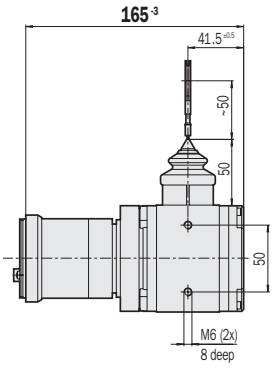
For technical support,
Call: 1.866.GO.FESTO
Fax: 1.800.96.FESTO
Email: product.support@us.festo.com

Resolution up to 0.025 mm

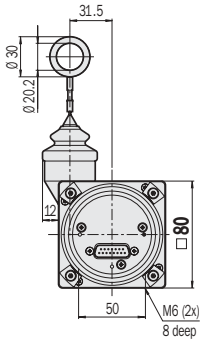
Absolute Wire Draw Encoders

- Linear path measurement using a wire draw mechanism
- High resolution
- Easy to mount
- High-precision measurement drum
- Extremely stable spring return
- Highly flexible steel wire
- Dirt remover made of steel

Dimensional drawing wire draw encoder BTF08 Profibus, CANopen, DeviceNet, measuring length 2 m



1

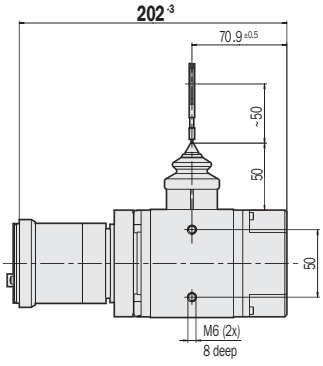


1

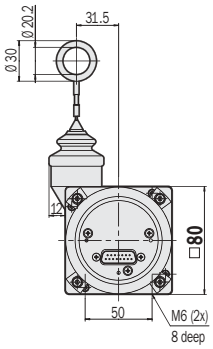
1 Threaded blind hole for mounting

General tolerances to DIN ISO 2768-mk

Dimensional drawing wire draw encoder BTF08 Profibus, CANopen, DeviceNet, measuring length 3 m



1



1

1 Threaded blind hole for mounting

General tolerances to DIN ISO 2768-mk

Accessories

Bus adaptors

Wire draw mechanisms

Mounting systems

Connection systems

Profibus adaptor with PIN and wire allocation see pages 18 / 19

CANopen adaptor with PIN and wire allocation see pages 20 / 21

DeviceNet adaptor with PIN and wire allocation see pages 22 to 24

Technical data	BTF08	PB 2 m	CO 2 m	DN 2 m	PB 3 m	CO 3 m	DN 3 m				
Drum housing	Anodised Aluminium	■	■	■	■	■	■				
Spring housing	Die-cast zinc	■	■	■	■	■	■				
Measuring wire (stainless)	Highly flexible stranded steel, Ø 1.35 mm	■	■	■	■	■	■				
Measuring length	2 m max.	■	■	■							
	3 m max.				■	■	■				
Mass	1.9 kg approx.	■	■	■							
	2.1 kg approx.				■	■	■				
Measuring step (recommended)	0.025 mm 1	■	■	■	■	■	■				
Linearity	0.05 % typ.	■	■	■	■	■	■				
Repeatability	± 1 measuring step	■	■	■	■	■	■				
Operating speed	4 m/sec.	■	■	■	■	■	■				
Position forming time	0.25 ms	■	■	■	■	■	■				
Spring return force (typ.)											
start/finish [1]	6 N/14 N	■	■	■	■	■	■				
Working temperature range	– 20 … + 70 °C	■	■	■	■	■	■				
Storage temperature range	– 40 … + 100 °C	■	■	■	■	■	■				
Life of wire draw mechanism [2]	1 million cycles	■	■	■	■	■	■				
EMC [3]		■	■	■	■	■	■				
Resistance											
to shocks [4]	100/6 g/ms	■	■	■	■	■	■				
to Vibration [5]	20/10 … 2,000 g/Hz	■	■	■	■	■	■				
Protection to IEC 60529	IP 64 (wire draw mechanism)	■	■	■	■	■	■				
	IP 67 (encoder)	■	■	■	■	■	■				
Operating voltage range (U_s)	10 … 32 V	■	■	■	■	■	■				
Power consumption max.	2.0 W	■	■	■	■	■	■				
Initialisation time [6]	1,250 ms	■	■	■	■	■	■				
Bus interface											
Electronic adjustment (Number SET)	Via PRESET switch or protocol	■	■	■	■	■	■				
Bus termination [7]	Via DIP switch	■	■	■	■	■	■				
Electrical connection	Connection adaptor	■	■	■	■	■	■				
Electrical interface [8]	RS485	■			■						
Electrical interface [9]	ISO-DIS 11898		■	■		■	■				
Protocol	Profile for encoders (07_{hex}) – Class 2	■			■						
	Communication Profile DS 301 V4.0		■			■					
	Device Profile DSP 406 V2.0		■			■					
	DeviceNet Specification, Release 2.0			■			■				
Address setting (node no.)	0 … 127 (DIP switch or protocol)	■			■						
Address setting (Node ID)	0 … 63 (DIP switch or protocol)		■	■		■	■				
Data transmission rate (Baud rate) [10]	9.6 kBaud … 12 MBaud	■			■						
(DIP switch or protocol)	(10, 20, 50, 125, 250, 500) kB, 1 MB		■			■					
(DIP switch or protocol)	(125, 250, 500) kB			■			■				
Status information	Running (LED green), bus activity (LED red)	■			■						
	2-coloured LED for CAN Controller Status		■			■					
	Network status LED (NS), 2-coloured			■			■				

[1] These values were measured at an ambient temperature of 25 °C. The values may be different at other temperatures.

[2] Average values, which depend on the loading. At high operating speeds over long lengths, this figure can decrease; at slow operating speeds over short lengths, it can increase.

[3] To DIN EN 61000-6-2 and DIN EN 61000-6-3

[4] To DIN EN 60068-2-27

[5] To DIN EN 60068-2-6

[6] From the moment the supply voltage is applied, this is the time which elapses before the data word can be correctly read in.

[7] Connection for terminal device only

[8] To EN 50 170-2 (DIN 19245 parts 1-3), galvanically separated using an opto-coupler

[9] (CAN High Speed) and CAN specification 2.0 B, galvanically separated

[10] Automatic detection

1 When the customer configures the encoder to 8,000 steps x 16 revolutions, via the Bus Master. (Factory entry in GSD or EDS file: 8,192 steps x 8,192 revolutions).

Order information

BTF08; U_s 10 … 32 V; field buses

Type	Part no.	Description
BTF08-P1HM0241	1034305	Profibus, measuring length 2 m
BTF08-D1HM0241	1034311	DeviceNet, measuring length 2 m
BTF08-C1HM0241	1034317	CANopen, measuring length 2 m
BTF08-P1HM0341	1034893	Profibus, measuring length 3 m
BTF08-D1HM0341	1034894	DeviceNet, measuring length 3 m
BTF08-C1HM0341	1034895	CANopen, measuring length 3 m

Please note: connection adaptor must be ordered separately (see pages 18 to 24)

Apéndice C

Glosario

API: Un API (Interfaz de programación de aplicaciones por sus siglas en Inglés) es un conjunto de bibliotecas y funciones que contienen las operaciones necesarias para programar una determinada aplicación. En este trabajo se refiere a las bibliotecas generadas por LabVIEW.

Big-endian: Es la representación de los bytes utilizando un cierto orden de tal forma que el dato 0x4A3B2C1D se codificaría en memoria en la secuencia 4A, 3B, 2C, 1D. En este trabajo se realizo una conversión de datos de este tipo a little-endian.

CANopen: Es un protocolo de comunicación que extiende el protocolo CAN para implementar más niveles en el modelo OSI.

cRIO: El CompactRIO (cRIO) es un controlador industrial de tiempo real desarrollado por National Instruments. El CompactRIO es una combinación de un controlador en tiempo real, modulos E/S reconfigurables, un módulo FPGA y un chasis de expansión Ethernet.

CVI: Es abreviación del software denominado LabwindowsCVI y es un compilador de C para Instrumentos Virtuales.

FPGA: Es un arreglo de lógica programable que requiere un archivo denominado "bitfile" que se descarga en el cRIO después de un proceso de compilación. Éste archivo se genera por medio de LabVIEW que simplifica las operaciones que se encuentran en el diagrama de bloques. Los FPGA's generalmente se programan utilizando VHDL, sin embargo en el caso de los FPGA's que se encuentran en los cRIO son únicamente compatibles con la compilación vía LabVIEW.

LabVIEW: Es una plataforma de programación gráfica desarrollada por National Instruments y enfocada a la programación de Instrumentos Virtuales.

Little-endian: Es la representación de los bytes, adoptado por Intel (entre otros) en el cual el dato 0x4A3B2C1D se codificaría en memoria en la secuencia 1D, 2C, 3B, 4A de tal forma que la codificación es más intuitiva.

PLC: Controlador de Lógica Programable.

Protocolo: Un protocolo es un conjunto de reglas que hacen que la comunicación en una red sea más eficiente.

VI: Instrumento Virtual (Por sus siglas en inglés) es la extensión de un programa realizado en LabVIEW y es la denominación al programa en sí. Dentro de un VI es posible generar un "subvi" que es el equivalente a un función en lenguajes de programación como C.

Bibliografía

John Park and Steve Mackay. *Practical Data Acquisition for Instrumentation and Control Systems*. Elsevier, Great Britain, 2003.

OPTO22, 2006. Understanding Programmable Automation Controllers (PACs) in Industrial Automation.

Andrew S. Tanenbaum. *Redes de Computadoras*. Pearson Educación, México, 2003.

National-Instruments, 2010. NI CompactRIO High-Performance Real-Time Controllers NI cRIO-9012, NI cRIO-9014.

Weiping Li Jean-Jaques E. Slotine. *Applied nonlinear control*. Prentice-Hall, USA, 1991.

Steven C. Chapra. *Numerical Methods for Engineers: with programming and software applications*. Mc Graw-Hill, USA, 1998.

M. Goettert and R.Ñeumann. Nichtlineare regelungskonzepte fÜr servopneumatische roboter. In *3. Deutsch-Polnisches Seminar Inovation und Fortschritt in der Fluidtechnik, Zakopane*, September 1999.

M. R. Sobczyk and E. A. Perondi. Variable structure cascade control of a pneumatic positioning system. *ABCM Symposium Series in Mechatronics*, 2:27–34, 2006.

P. Beater. *Pneumatic Drives*. Springer-Verlag Berlin Heidelberg, 2007.

G. Kothapalli and M. Y.Hassan. Design of a Neural Network Based Intelligent PI controller for a Pneumatic system. *IAENG International Journal of Computer Science*, 35:69–74, 2007.

Katsuhiko Ogata. *Ingeniería de Control Moderna*. PRENTICE-HALL, México, 1998.

Shu Ning and Gary M. Bone. Experimental comparison of two pneumatic servo position control algorithms. *IEEE*, 0-7803-9044-X(05):p. 37–42, 2005.

Marco A. Arteaga, Adrián Castillo-Sánchez, and Vicente Parra-Vega. Cartesian control of robots without dynamic model and observer design, journal = Automatica, volume = 42, pages = 473-480, year = 2006.

Can in Automation, 2002. CANOPEN, Application Layer and Communication Profile.