UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN MECATRÓNICA

TESIS PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN MECATRÓNICA

TEMA:

MÓDULO DIDÁCTICO PARA EL MODELAMIENTO DE SISTEMAS LINEALES CON MATLAB Y TARJETA COMPATIBLE USB

AUTOR:

JONATHAN JAVIER TAPIA MARROQUÍN

DIRECTOR:

Ing. ÁLVARO FUENTES

Ibarra – Ecuador

Diciembre 2013

DECLARACIÓN

Yo, JONATHAN JAVIER TAPIA MARROQUÍN, declaro que el trabajo aquí descrito es de mi autoría, no ha sido previamente presentado para ningún grado o calificación profesional y certifico la veracidad de las referencias bibliográficas que se incluyen en este documento.

Jonathan Javier Tapia Marroquín

CERTIFICACIÓN

En calidad de Director del Trabajo de Grado “Módulo didáctico para el modelamiento de sistemas lineales con Matlab y tarjeta compatible USB”, presentado por el señor Jonathan Javier Tapia Marroquín, para optar por el título de Ingeniero en mecatrónica, certifico que el mencionado proyecto fue realizado bajo mi dirección.

Ing. Álvaro Fuentes

DIRECTOR

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

La Universidad Técnica del Norte dentro del proyecto Repositorio Digital Institucional, determinó la necesidad de disponer de textos completos en formato digital con la finalidad de apoyar los procesos de investigación, docencia y extensión de la Universidad.

Por medio del presente documento dejo sentada mi voluntad de participar en este proyecto, para lo cual pongo a disposición la siguiente información:

DATOS DEL CONTACTO	
CÉDULA DE IDENTIDAD:	100341582-3
APELLIDOS Y NOMBRES:	JONATHAN JAVIER TAPIA MARROQUÍN
DIRECCIÓN:	GARCÍA MORENO 10-09 Y PEDRO RODRIGUEZ
E-MAIL:	juvenilforever@hotmail.com
TELÉFONO MOVIL:	0989153751

DATOS DE LA OBRA	
TÍTULO:	MÓDULO DIDÁCTICO PARA EL MODELAMIENTO DE SISTEMAS LINEALES CON MATLAB Y TARJETA COMPATIBLE USB
AUTOR:	JONATHAN JAVIER TAPIA MARROQUÍN
FECHA:	2013/12/16
PROGRAMA:	PREGRADO
TÍTULO POR EL QUE OPTA:	INGENIERO EN MECATRÓNICA
DIRECTOR:	ING. ÁLVARO FUENTES

2. AUTORIZACIÓN DE USO A FAVOR DE LA UNIVERSIDAD

Yo, Jonathan Javier Tapia Marroquín, con cédula de identidad Nro. 100341582-3, en calidad de autor y titular de los derechos patrimoniales de la obra o trabajo de grado descrito anteriormente, hago entrega del ejemplar respectivo en forma digital y autorizo a la Universidad Técnica del Norte, la publicación de la obra en el Repositorio Digital Institucional y uso del archivo digital en la Biblioteca de la Universidad con fines académicos, para ampliar la disponibilidad del material y como apoyo a la educación, investigación y extensión; en concordancia con la Ley de Educación Superior, Artículo 144.

3. CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 16 días del mes de Diciembre del 2013

EL AUTOR:

Jonathan Javier Tapia Marroquín

CI: 100341582-3

CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE GRADO A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

Yo, Jonathan Javier Tapia Marroquín, con cédula de identidad Nro. 100341582-3, manifiesto mi voluntad de ceder a la Universidad Técnica del Norte los derechos patrimoniales consagrados en la Ley de Propiedad Intelectual del Ecuador, Artículos 4,5 y 6, en calidad de autor de la obra o trabajo de grado denominado "Módulo didáctico para el modelamiento de sistemas lineales con Matlab y tarjeta compatible USB", que ha sido desarrollada para optar por el título de Ingeniero en Mecatrónica en la Universidad Técnica del Norte, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente. En mi condición de autor me reservo los derechos morales de la obra antes citada. En concordancia suscribo este documento en el momento que hago la entrega del trabajo final en formato impreso y digital en la Biblioteca de la Universidad Técnica del Norte.

(Firma)_____

Nombre: Jonathan Javier Tapia Marroquín

Cédula: 100341582-3

Ibarra, a los 16 días del mes de Diciembre del 2013

AGRADECIMIENTO

A mis padres y hermanos, por brindarme el apoyo constante en miras de alcanzar mis metas.

Especial reconocimiento al Ing. Álvaro Fuentes por su apoyo y guía en el desarrollo del presente trabajo.

A la Universidad Técnica del Norte, la Facultad de Ingeniería en Ciencias Aplicadas, y de manera especial al personal docente quienes me ilustraron con sus conocimientos, siempre útiles en la vida profesional.

A todos los amigos y amigas que influyeron de manera directa o indirecta en la elaboración del proyecto.

Jonathan T.

DEDICATORIA

A Dios, por ser mi fuerza de inspiración en la realización de objetivos.

Con infinito amor a mis padres y hermanas que con su apoyo incondicional y ejemplo, han sido un pilar fundamental en mi formación personal y el cumplimiento de sueños y objetivos profesionales.

Jonathan T.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS, TABLAS, DIAGRAMAS Y ECUACIONES

FIGURAS

TABLAS

DIAGRAMAS

ECUACIONES

LISTA DE SIGLAS

DAQ:	Tarjeta de adquisición de datos
DAC:	Conversor analógico a digital
MIP:	Módulo de interfaz de potencia
USB:	Bus de serie universal
LED:	Diodo emisor de luz
AR:	Auto-regresivo
MA:	Promedio móvil
ARMA:	Promedio móvil y auto-regresivo
ARX:	Auto-regresivo con entrada exógena
ARMAX:	Promedio móvil y auto-regresivo con entrada exógena
OE:	Error de salida
SISO:	Una entrada, una salida
ZOH:	Retenedor de orden cero
FOH:	Retenedor de primer orden
GUI:	Interfaz gráfica de usuario
DLL:	Librería dinámica
VID&PID:	Identificación del proveedor y del producto
RC:	Resistencia-capacitor

MÓDULO DIDÁCTICO PARA EL MODELAMIENTO DE SISTEMAS LINEALES CON MATLAB Y TARJETA COMPATIBLE USB

Autor: Jonathan Tapia

Tutor: Ing. Álvaro Fuentes

RESUMEN

El trabajo que se detalla a continuación, se fundamenta en la necesidad de la enseñanza y aprendizaje práctico por parte del profesor o instructor guía y el alumno, para adquirir criterios en cuanto se refiere a sistemas de control en un estudió centralizado en la identificación del funcionamiento de un objeto o sistema físico, pudiendo representarlo por una función que caracteriza el comportamiento del sistema anteriormente mencionado. En algunas ocasiones, al momento de realizar el modelo matemático de un sistema, no se conocen la mayoría de valores que caracterizan al mismo, siendo este un problema ya que no se podría implementar un circuito que interactúe con este para un fin específico; debido a ello se desarrolló un módulo didáctico para el modelamiento de estos sistemas, cuyo fin es el reconocimiento de las variables para el modelado en forma práctica. El objetivo que persigue el proyecto propuesto es el desarrollo de un módulo didáctico que simplifique el modelamiento de sistemas físicos para mejorar el entendimiento del estudiante con respecto a este campo. Debido a ello el proyecto se ha dividido en dos partes fundamentales las cuales son: la *adquisición de datos* en la cual el alumno debe establecer los parámetros de adquisición de valores representativos para la reconstrucción de la señal de respuesta del sistema, y el *modelado de datos* en la cual se realiza la aproximación de los valores obtenidos a una curva o señal que representa el modelo matemático del funcionamiento en cuanto a la respuesta del sistema. El presente proyecto también abarca el desarrollo de diferentes módulos con los cuales el estudiante podrá realizar prácticas de reconocimiento de la función característica de los mismos, partiendo de la base teórica que origina el comportamiento de los sistemas físicos implementados.

TRAINING MODULEFOR MODELINGLINEAR SYSTEMS WITHMATLAB ANDUSBCOMPATIBLE CARD

Author: Jonathan Tapia
Tutor: Ing.Álvaro Fuentes

SUMMARY

The work described below is based on the need of education and practical learning from the teacher or instructor and student, to acquire criteria regarding to control systems into a centralized study performance to identify a physical object or system, and may represent it as a function that characterizes the behavior of the system mentioned above. Sometimes when making a mathematical model of a system, the values that characterize it are do not know, being it a problem because a circuit that interacts with it for a specific purpose, could not be implemented. Due to that a training module for modeling these systems where developed, whose purpose is the recognition of variables for modeling in a practical way. The objective of the proposed project is the development of a training module that simplifies the modeling of physical systems to improve student understanding, with respect to this field. Due to that, the project has been divided into two main parts which are: *data acquisition* in which the student should set up the acquisition parameters for the reconstruction of the system response signal, and the *data modeling* in which it is made the approximation of the obtained values to a curve or signal that represent the mathematical model of the operation as to the system response. The present project also includes the development of different modules with which the student could do practices of recognition of the characteristic function, based on the same theoretical basis, that causes the behavior of the implemented physical systems.

PRESENTACIÓN

El proyecto del módulo didáctico para el modelamiento de sistemas lineales con Matlab y tarjeta compatible USB, está estructurado en seis capítulos: Análisis de la situación actual, marco teórico, diseño y construcción de la tarjeta de adquisición de datos, diseño de la interfaz gráfica, prácticas de laboratorio e implementación de módulos didácticos, conclusiones y recomendaciones.

En el primer capítulo se analiza la problemática que se está viviendo en cuanto a la enseñanza de profesor – estudiante planteando objetivos afines al desarrollo e implementación de un módulo que simplifique la enseñanza de sistemas de control en un tema específico, el cual es el modelado de sistemas físicos realizado en forma práctica. También se definen los límites que tiene el proyecto propuesto para el conocimiento a los usuarios o personas que van a utilizar el módulo anteriormente mencionado.

En el segundo capítulo se definen los conceptos básicos o generales con el cual se fundamenta el proyecto presente, dando a conocer reglas, leyes, circuitos varios y diagramas que ayudan en la comprensión con el cual se desarrolla y se implementa el módulo.

En el tercer capítulo se estudia los requerimientos que debe tener una DAQ para cumplir con las necesidades obtenidas en cuanto al funcionamiento de la misma; además se describe la selección, diseño e implementación de una DAQ en base a requerimientos planteados.

En el cuarto capítulo se desarrolla una interfaz gráfica la cual ayuda a cumplir con el fin del proyecto presente, realizando un software que interactúa con la DAQ para el modelamiento de sistemas físicos.

En el quinto capítulo se da a conocer el funcionamiento del módulo planteado a través de la realización de diferentes prácticas de laboratorio para que el estudiante comprenda el procedimiento de identificación y modelado además de la comprobación del mismo.

En el sexto capítulo se detallan las conclusiones y recomendaciones del proyecto presente.

CAPÍTULO I

ANÁLISIS DE LA SITUACIÓN ACTUAL

1.1. INTRODUCCIÓN

Los Seres humanos siempre hemos buscado nuevas formas y métodos que faciliten la enseñanza y el aprendizaje; si nos remontamos desde tiempos atrás hasta la actualidad vemos que el mundo ha ido evolucionando de manera inimaginable con la invención de la tecnología que ha dado comodidad facilitando las tareas realizadas por la gente, obligando a las personas a buscar y tomar nuevos rumbos para la subsistencia con la misión de especializarse en profesiones que el mundo demanda para su desarrollo.

Don Francisco Sánchez Ayuso dice: "oigo y olvido, veo y recuerdo, hago y comprendo". Por consiguiente, desde el punto de vista de estudiantes de carreras técnicas, no solamente es necesario abordar la teoría en clase, sino también reforzar con la enseñanza práctica, misma que permite a su vez, una aplicación más real de los conocimientos.

1.2. PLANTEAMIENTO DEL PROBLEMA

Para la preparación del ser humano en carreras técnicas como lo es la Ingeniería Mecatrónica, demanda de equipos para realizar prácticas en cada una de las materias o módulos que la profesión compete. El uso de equipos de laboratorio ayuda en el aprendizaje práctico del alumno, pero aún así se presentan dificultades al momento de realizar prácticas, debido a la falta de herramientas que faciliten hacerlo. Estos inconvenientes, generan en los estudiantes, dudas en cuanto a la aplicación, desarrollo de prácticas y la implementación de las mismas.

El proyecto propuesto, permite una mayor facilidad en el desarrollo de proyectos y prácticas relacionadas con el área de sistemas de control, a través de esto se da un mayor enfoque en donde el estudiante estará en capacidad de aplicar los

conocimientos aprendidos en clases, desarrollando prácticas y proyectos de forma más didáctica, al encontrar la ecuación característica de sistemas físicos (plantas) requeridos y poder aplicarlos en el modelamiento de la mismas utilizando el módulo propuesto.

1.3. OBJETIVOS DE LA INVESTIGACIÓN

1.3.1. OBJETIVO GENERAL

- Desarrollar un módulo didáctico de adquisición de datos mediante una tarjeta USB e interfaz gráfica en Matlab para la obtención y modelado de plantas lineales.

1.3.2. OBJETIVOS ESPECÍFICOS

- Implementar una tarjeta compatible con Matlab con comunicación USB para la adquisición de datos de la planta.
- Verificar la compatibilidad de la tarjeta a través de la realización de varias pruebas y ayudas didácticas para el estudio.
- Emplear un brazo robótico con sensores de posicionamiento angular para la realización de pruebas con la tarjeta anteriormente mencionada.
- Diseñar una interfaz gráfica que ayude al mejor entendimiento del módulo didáctico, incluyendo los pasos a seguir para la adquisición y modelado, y la ejecución de prácticas de laboratorio.
- Elaborar un manual de usuario para el uso de estudiantes y una guía de prácticas de laboratorio.

1.4. JUSTIFICACIÓN

Como ya se mencionó anteriormente, todo estudiante necesita de la implementación y estudio práctico para el refuerzo de lo aprendido en clases, además en el área de sistemas de control al momento de realizar una práctica es necesario la utilización de alguna herramienta de adquisición de datos para obtener la ecuación de la planta y la posterior implementación de un controlador. Como es un proceso largo que requiere de los materiales y equipos indicados para realizar

satisfactoriamente la práctica. El proyecto propuesto implica todo lo mencionado anteriormente, permitiendo a los estudiantes una mayor facilidad para la adquisición y modelado de sistemas lineales (plantas) y la realización de prácticas en esta materia.

1.5. ALCANCE

El desarrollo de proyecto, consta de lo siguiente: La implementación de una tarjeta USB tiene entradas analógicas necesarias con lectura de 0V a 5V, salidas digitales y PWM a disposición. Además de un transistor con salida relé de estado sólido para controlar el proceso de modelado de una planta que requiera potencia. La tarjeta tiene una frecuencia máxima de muestreo de 500 Hz. La utilización de la tarjeta USB sirve como un medio de comunicación entre el software y el hardware, permitiendo el ingreso de los datos obtenidos de la planta para su modelado.

Se utilizó el software Matlab 2012a, como medio de interfaz gráfica para la realización de las diferentes prácticas de laboratorio por su robustez en el área de reconocimiento y modelado de señales. El programa consta de dos partes fundamentales, las cuales son: adquisición de la señal, reconocimiento y modelado de la planta además de una guía de usuario para el manejo del software y conexión del hardware.

El sistema de adquisición y modelado sólo está enfocado a plantas lineales o estables que tengan una entrada y una salida, cuyo régimen transitorio será como mínimo 1s y máximo 30 minutos. El módulo consta con una guía de prácticas para laboratorio las cuales son adquisición y modelado de plantas de: temperatura, velocidad angular, posicionamiento angular, luminosidad.

CAPÍTULO II

MARCO TEÓRICO

2.1. INTRODUCCIÓN

En este capítulo se contemplará conceptos fundamentales para el desarrollo del proyecto.

2.2. ELECTRÓNICA[1]

La electrónica es parte de la física que estudia y utiliza el comportamiento de las señales eléctricas para obtener o transmitir información al medio, pudiendo ser: audio, sonido, mediciones de magnitudes físicas, etc.

Según Vallejo, 2003: "*La electrónica es el campo de la ingeniería que estudia el aprovechamiento del flujo de electrones en dispositivos semiconductores, para generar, recibir, almacenar y transmitir información en forma de señales eléctricas*".

2.2.1. ELECTRÓNICA ANALÓGICA

La electrónica analógica se encargada de estudiar los tipos de señales continuas (tensión, corriente), ya sea si estas varían o se mantienen estables en el tiempo, estudiando el comportamiento de dispositivos semiconductores como: el diodo, el transistor, los amplificadores operacionales, etc.

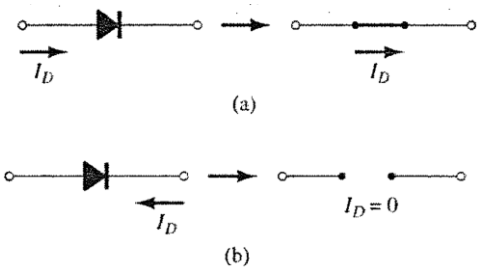
[1]Recopilación de información de varios autores

2.2.1.1. DIODO

El diodo es un dispositivo semiconductor que deja fluir la corriente en un solo sentido, está formado por la juntura de dos tipos de materiales: tipo P y tipo N, estos dispositivos están fabricados de silicio que presenta una barrera de umbral de activación de 0.7 V y de germanio con 0.3 V.

La característica principal de un diodo es: cuando se lo polariza directamente se comporta como un circuito cerrado y en polarización inversa tiene un comportamiento de circuito abierto.

Figura 1. Símbolo y Polarización del Diodo



Fuente: Electrónica. Teoría de circuitos y dispositivos electrónicos, 2009

Existen varios tipos de diodos, los cuales se presentan a continuación:

Figura 2. Tipos de Diodos

Tipos de Diodos
- Rectificadores
- LED
- Zener
- Varicap
- Láser
- Tunnel
- Schotty
- Pin
- Gunn
- Fotodiodo

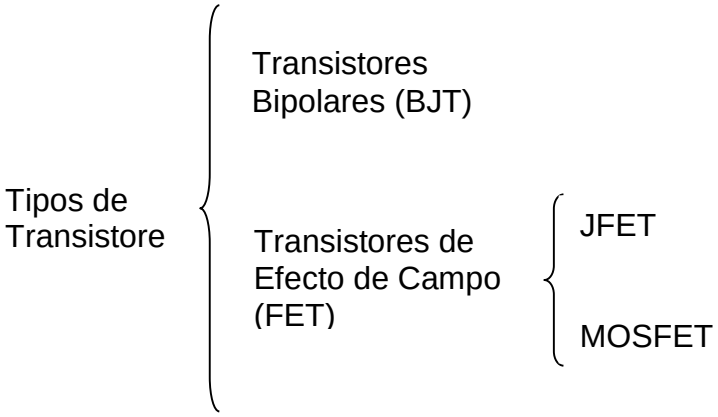
Fuente: Curso fácil de electrónica básica, CEKIT

2.2.1.2. TRANSISTOR

La palabra transistor es un acrónimo de los términos transfer y resistor (resistencia de transferencia) y designa a un componente electrónico de tres terminales cuya resistencia entre dos de ellos depende del nivel de corriente o voltaje aplicado al otro. (CEKIT)

Existen dos grandes grupos de transistores los cuales están presentados a continuación:

Figura 3. Tipos de Transistores



Fuente: Curso fácil de electrónica básica, CEKIT

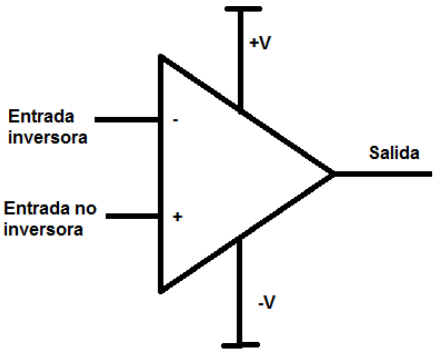
2.2.1.3. AMPLIFICADORES OPERACIONALES

El nombre de amplificadores operacionales viene de la utilización que se los da a estos dispositivos, como para la realización de operaciones matemáticas. (Sedra & Smith, 2011)

Presentan las siguientes características de funcionamiento:

- Alta impedancia de entrada
- Alta ganancia de voltaje
- Impedancia de salida cero
- Ancho de banda infinito

Figura 4. Símbolo del Amplificador Operacional



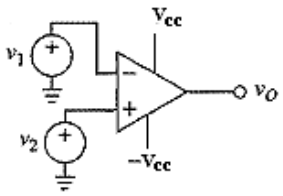
Fuente: Curso fácil de electrónica básica, CEKIT

Estos dispositivos vienen a ser una gran ayuda para el diseño e implementación de circuitos basados en amplificadores operacionales ya que con diferentes conexiones se pueden obtener varias aplicaciones con solo la utilización de componentes electrónicos. Algunas de las aplicaciones más comunes que tienen son:

- **Comparador**

  Permite diferenciar entre dos valores de voltajes; reconoce cuando uno es mayor que otro.

Figura 5. Comparador



$$Si\ v_2 > v_1 : v_0 = v_{cc}$$

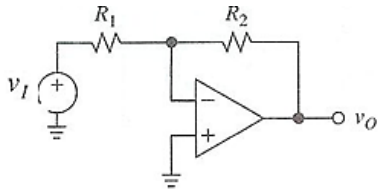
$$Si\ v_2 < v_1 : v_0 = -v_{cc}$$

Fuente: Diseño con amplificadores operacionales y circuitos integrados analógicos, 2005

- **Amplificador inversor**

  Se caracteriza porque el valor de voltaje a la salida es el valor de voltaje de la entrada multiplicado por un factor negativo, es decir, la salida es de signo contrario a la entrada.

Figura 6. Amplificador inversor



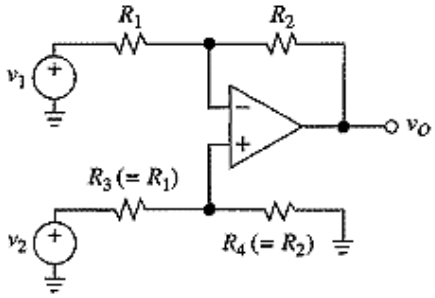
$$\frac{v_O}{v_I} = \left(-\frac{R_2}{R_1}\right)$$

Fuente: Diseño con amplificadores operacionales y circuitos integrados analógicos, 2005

- **Amplificador diferencial**

  Se caracteriza por realizar una resta entre dos valores de voltajes a cuyo resultado se le puede amplificar o reducir.

Figura 7. Amplificador diferencial



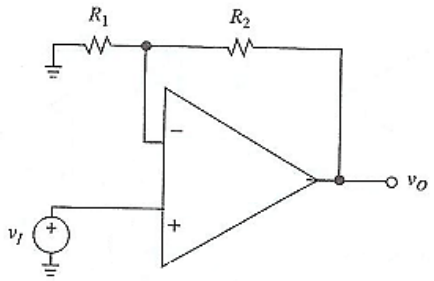
$$v_0 = \frac{R_2}{R_1}(v_2 - v_1)$$

Fuente: Diseño con amplificadores operacionales y circuitos integrados analógicos, 2005

- **Amplificador no inversor**

  Se caracteriza por realizar solamente amplificación del valor de voltaje de la entrada.

Figura 8. Amplificador no inversor



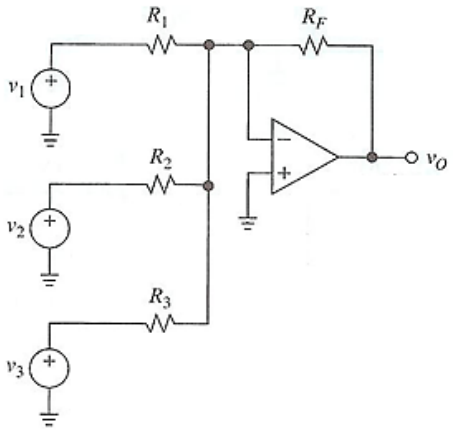
$$\frac{v_O}{v_I} = \left(1 + \frac{R_2}{R_1}\right)$$

Fuente: Diseño con amplificadores operacionales y circuitos integrados analógicos, 2005

- **Amplificador Sumador Inversor**

  Permite sumar varios valores de voltajes, multiplicar el resultado por un factor y cambiarlo de signo.

Figura 9. Amplificador Sumador Inversor



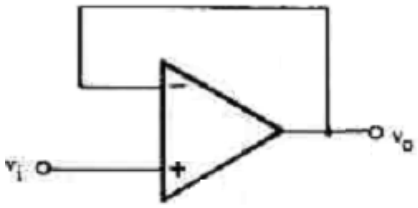
$$v_0 = -\left(\frac{R_F}{R_1}v_1 + \frac{R_F}{R_2}v_2 + \frac{R_F}{R_3}v_3\right)$$

Fuente: Diseño con amplificadores operacionales y circuitos integrados analógicos, 2005

- **Seguidor de voltaje**

  Su característica principal es: el valor a su salida es igual al valor de la entrada, permitiendo que en esta se consuma baja corriente.

Figura 10. Seguidor de voltaje



$$v_O = v_I$$

Fuente: Diseño con amplificadores operacionales y circuitos integrados analógicos, 2005

2.2.2. ELECTRÓNICA DIGITAL

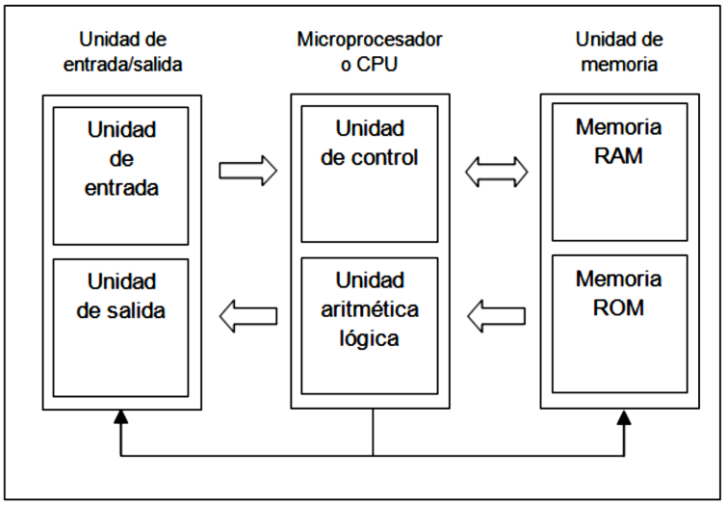
La electrónica digital a parte de la electrónica analógica se encarga de estudiar las señales digitales, es decir, señales cuya información están codificados en dos estados: valores altos (1 lógico) o valores bajos (0 lógico), originando una secuencia de números que vienen a representar la magnitud de la señal.

2.2.3. MICROCONTOLADORES

Un microcontrolador es un circuito programable que contiene todos los componentes de un computador pero de tamaño reducido. Se emplea para controlar el funcionamiento de una tarea determinada, es decir, es un computador dedicado, y todos los recursos complementarios disponibles tienen como única finalidad de atender sus requerimientos.(Collaguazo G. , 2007)

También se lo define como un circuito integrado que contiene toda la estructura de una minicomputadora, o sea: CPU (Unidad Central de Proceso), memoria RAM, memoria ROM, circuitos de entrada-salida (periféricos) y otros módulos con aplicaciones especiales. Su nombre nos indica sus principales características: *“micro”* por lo pequeño y *“controlador”* porque se lo utiliza principalmente para controlar otros dispositivos ya sean eléctricos, mecánicos, etc.(CEKIT, 2002)

Figura 11. Estructura general de un microcontrolador



Fuente: Microcontroladores CEKYT, 2002

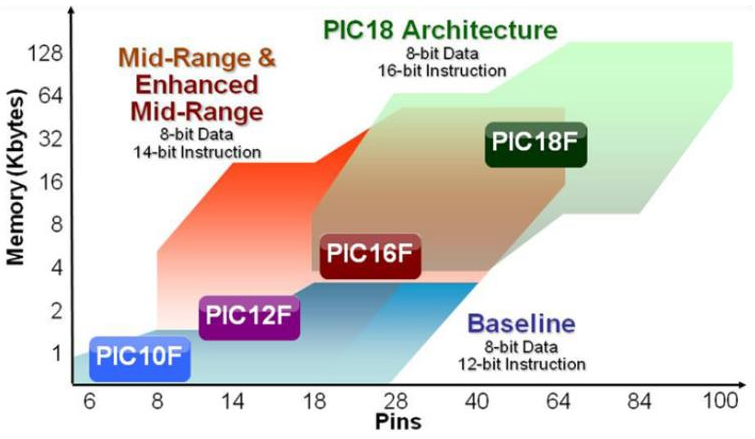
Un microcontrolador dispone de los siguientes recursos:

- ✓ Procesador o CPU encargada principalmente de la ejecución de instrucciones contenidas en el dispositivo.
- ✓ Memoria para guardar datos ya sea permanentemente (ROM) o temporalmente (RAM).
- ✓ Líneas de Entrada/Salida para comunicación e interacción con el exterior.
- ✓ Generador de pulsos de reloj que sincronizan el funcionamiento del sistema completo
- ✓ Diversos módulos para el control de periféricos (temporizadores, puertos serie y paralelo, conversores analógico/digital, conversores digital/analógico, etc.)

Existen en el mercado varias marcas de microcontrolador es siendo MICROCHIP una de ellas, con una amplia gama de estos dispositivos, los cuales se los clasifica de la siguiente manera:

- **Microcontroladores PIC de 8 bits**[2]

Figura 12. Microcontroladores PIC de 8 bits



Fuente: http://www.microchip.com/pagehandler/en-us/family/8bit/architecture/home.html

- **Microcontroladores Base**

- Ser de 33 instrucciones simple de 12-bit de ancho.
- 2 KW (3 KB) de memoria de programa direccionable.
- 144 bytes de memoria RAM máximo
- 2 nivel de pila hardware

- **Microcontroladores de Medio Rango**

- Set de 35 instrucciones de 14-bit de ancho.
- 8 KW (14 KB) de memoria de programa direccionable.
- 46 bytes de RAM máximo.
- 8 nivel de pila hardware.
- Manejo de interrupciones de hardware.

- **Microcontroladores Mejorados**

- Set de 49 instrucciones de14-bit de ancho.
- 32 KW (56 KB) de memoria de programa direccionable.

[2]Información obtenida de: http://www.microchip.com

- 4 KB de RAM máximo.
- 16 Nivel de pila de hardware.
- Manejo de interrupciones de hardware con ahorro de contenido.
- Conjunto de características avanzadas, múltiples comunicaciones en serie y la capacidad de control del motor.

  - **Microcontroladores PIC18**

- Procesamiento de hasta 16 MIPS.
- Set de 83 instrucciones de 16-bit de ancho.
- Hasta 2MB de memoria de programa direccionable.
- 4 KB RAM máximo.
- 32 niveles de pila hardware.
- El mayor rendimiento de la arquitectura de 8-bits.

- **Microcontroladores PIC de 16 bits[3]**
  - **Microcontroladores PIC24F**

- 16 MIPS rendimiento a 3.3 V.
- Consumo de corriente en baja como 20nA.
- 2.0 V a 3.6 V operación.
- Periféricos alta mente integrados para control de motores, pantallas de gráficos, USB OTG y aplicación táctil capacitiva.

  - **Microcontroladores PIC24H/E**

- Hasta 70MIPS funcionamiento a 3,3 V.
- Control de motores por periféricos y amplificadores operacionales integrados.
- Alto rendimiento de12-bit ADC y CAN.
- Amplia gama de opciones de memoria/ paquete.
- Apoyo a alta temperatura (150°C).

  - **Microcontroladores dsPIC30F**

[3] Información obtenida de: http://www.microchip.com

- 30MIPS DSC a 5 V.
- Ciclo único MAC.
- 40 bit acumulador y operando dual.
- Control de motores altamente integrado, SMPS y periféricos de audio.

  - **Microcontroladores dsPIC33F/E**

- Hasta 70MIPS DSC a 3.3 V.
- Ciclo MAC de 16x 16.
- Amplificadores Operacionales Integrados.
- 40 bit acumulador y operando dual.
- Control de motores altamente integrado, SMPS y periféricos de audio.

- **Microcontroladores PIC de 32 bits[4]**

- Hasta 80 MHz.
- Rango de temperatura: -40°C a 105°C.
- Hasta 512KB Flash.
- Hasta 128KBde SRAM.
- Buses para instrucciones y datos separados.
- Alta velocidad en dispositivos USB Host/OTG.
- Hasta 6UART, 5 I²C, 4 puertos SPI, CTMU y I²S.
- Hasta 8 canales adicionales de uso general DMA.

2.3. SENSORES Y TRASNDUCTORES

Un sensor es un elemento que produce una señal relacionada con la cantidad que se está midiendo y un transductor es el elemento que al someterlo a un cambio físico experimenta un cambio relacionado, es decir, un transductor es un sensor. Un sensor inteligente puede tener funciones como la capacidad de compensar errores al azar, adaptarse a cambios en el medio ambiente, ajustes para salida

[4]Información obtenida de: http://www.microchip.com

lineal, auto calibración y diagnosticar fallas.(Bolton, Mecatrónica. Sistemas de Control Electrónico en la Ingeniería Mecánica y Eléctrica, 2010)

2.3.1. CARACTERÍSTICAS DE FUNCIONAMIENTO

Antes de la utilización de cualesquier sensor para una función determinada debemos conocer cuáles son sus requerimientos para un excelente funcionamiento, para ello todos estos dispositivos cuentan con características que debemos conocer para la selección del mismo.

2.3.1.1. CARACTERÍSTICAS ESTÁTICAS

Se refieren a la actuación de un sensor cuando este presenta ciertas condiciones al estabilizarse, haciendo referencia a valores reales o exactos, se puede encontrar las siguientes características:

- Rango
- Exactitud
- Repetitividad
- Reproducibilidad
- Resolución
- Error
- No linealidad
- Sensibilidad

Son los valores obtenidos cuando se presentan condiciones de estado estable, es decir, valores obtenidos una vez que el transductor se asienta después de recibir cierta entrada. (Bolton, Mecatrónica. Sistemas de Control Electrónico en la Ingeniería Mecánica y Eléctrica, 2010)

2.3.1.2. CARACTERÍSTICAS DINÁMICAS

Nos indica el funcionamiento y tipo de respuesta al momento de pasar de un valor de entrada a otro diferente, tardándose un tiempo determinado en llegar a su valor estático, obteniendo las siguientes características:

- Tiempo de retardo
- Tiempo de su vida
- Tiempo de pico

- Pico de sobre-oscilación
- Tiempo de establecimiento

Se refiere al comportamiento del transductor entre el momento en que cambia el valor de la entrada y el tiempo en que el dispositivo logra su valor en estado estable. (Bolton, Mecatrónica. Sistemas de Control Electrónico en la Ingeniería Mecánica y Eléctrica, 2010)

2.3.2. CLASIFICACIÓN

Se los puede clasificar de acuerdo a su aplicación dándose a conocer que para cada una de ellas existen sensores análogos que me entregan un valor en cada instante de tiempo, del que se obtiene una señal continua; y sensores digitales que entregan valores en cada momento único y conocido de tiempo lo cual representa una señal discreta.

De acuerdo a su aplicación tenemos los siguientes tipos de sensores:

- Detectores de ultrasonidos
- Interruptores básicos
- Interruptores de fin de carrera
- Interruptores manuales
- Fibra óptica
- Infrarrojos
- Caudal
- Sensores de corriente
- Sensores de efecto Hall
- Sensores de humedad
- Sensores de posición de estado sólido
- Sensores de presión y fuerza
- Sensores de temperatura
- Sensores de turbidez
- Sensores magnéticos

2.4. ACTUADORES

Provocan una acción frente a un estímulo externo, es decir, son análogos a las extremidades de una persona, ellas se mueven si la persona quiere realizar una acción determinada o sino no realiza algún movimiento. Existen diferentes tipos de actuadores siendo los más conocidos:

- Actuadores mecánicos
- Actuadores eléctricos
- Actuadores hidráulicos
- Actuadores neumáticos
- O cualquier combinación de los anteriores

Los sistemas de actuación son los elementos de los sistemas de control que transforman la salida de un microprocesador o un controlador en una acción de control para una máquina o dispositivo.(Bolton, Mecatrónica. Sistemas de Control Electrónico en la Ingeniería Mecánica y Eléctrica, 2010)

2.5. SISTEMAS DE CONTROL

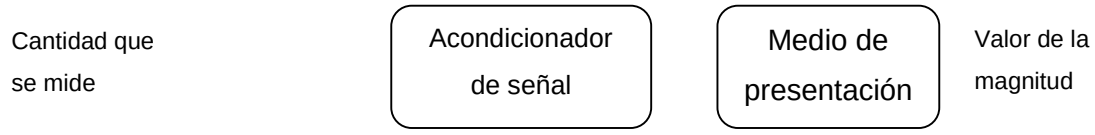
Antes de comenzar con la definición, primero debemos entender el funcionamiento, como por ejemplo: imaginémonos un objeto el cual sus características o conexiones internas en este momento pasan por alto, pero dicho objeto ante cualquier estímulo reacciona de diferente manera, ya sea que el estímulo fuera provocado por temperatura, luz, electricidad, humedad, etc. Entonces para cada una de estas reacciones con el mismo objeto puede representarse por un solo comportamiento que origina una respuesta diferente.

Los sistemas de control pueden ser definidos como la incorporación de una acción de control que permite que el sistema manipulado funcione de manera rápida o lenta, cambiando así su señal de respuesta haciéndolo más eficiente o deficiente según las características del controlador incorporado.

2.5.1. SISTEMA DE MEDICIÓN

A menudo se necesita saber como está respondiendo el elemento de proceso tomando como punto y verificación el elemento de medición para su respectiva visualización y conocimiento al ser humano.

Figura 13. Elementos de un sistema de medición



Fuente: Mecatrónica. Sistemas de Control Electrónico en la Ingeniería Mecánica y Eléctrica, 2010

a. Sensor: el cual permite relacionar la cantidad de respuesta del sistema con una respuesta eléctrica, entendible y utilizable por el circuito de control.
b. Acondicionamiento de la señal: muchas veces la señal provenida del sensor es muy débil y se necesita su manipulación para generar una señal del mismo tipo pero con mayores prestaciones.
c. Sistema de presentación visual: el cual permite al ser humano poder observar como está respondiendo el sistema frente a un estímulo al elemento de proceso.

2.5.2. TIPOS DE SISTEMAS

A pesar de su gran aplicación en el mundo actual se los puede organizar en dos grupos: sistemas de control en lazo abierto y en lazo cerrado, obteniendo una diferencia clara entre los dos, la cual es la configuración del sistema, siendo el sistema de lazo cerrado el más confiable.

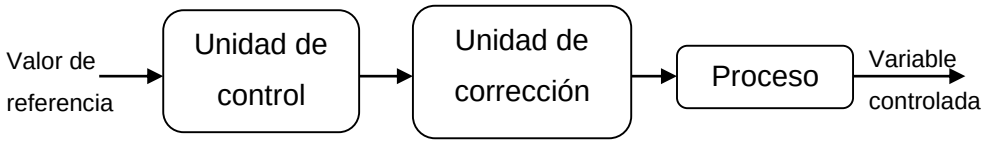
2.5.2.1. SISTEMAS EN LAZO ABIERTO

Su característica principal es que su señal de salida no influye en la señal de control del sistema, siendo un controlador con la suposición de que su salida está correcta con el valor deseado y su funcionamiento global no tiene errores, sin preocuparse por las perturbaciones existentes.

Los sistemas de control en lazo abierto son más sencillos de implementar, cuyo costo resulta bajo con una buena confiabilidad en cuanto no existan perturbaciones, pero son imprecisos con la existencia de errores no corregidos a la salida.

Los elementos mencionados a continuación son indispensables a la hora de diseñar y armar un sistema de control de este tipo, ya que con la falta de uno de ellos el funcionamiento del sistema sería incorrecto.

Figura 14. Elementos básicos de un control en lazo abierto



Fuente: Ingeniería de control, 2006

a. Elemento de control: Utiliza la señal de entrada para realizar una acción.
b. Elemento de corrección: Permite realizar un cambio en el proceso a fin de alterar la señal obtenida del controlador para producir a la salida del sistema el valor deseado.
c. Elemento de proceso: Es el elemento en el cual se va a controlar el valor de la señal de salida.

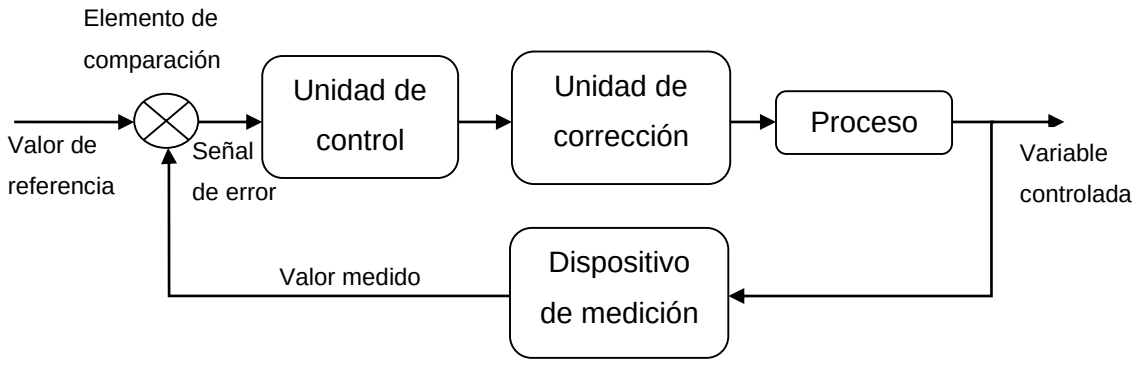
2.5.2.2. SISTEMAS EN LAZO CERRADO

La característica principal es que la señal de acción o salida del sistema está relacionada con la señal de referencia o valor deseado misma que se utiliza para que el sistema de control la corrija hasta llegar y mantenerla en el valor deseado.

Los sistemas de control en lazo cerrado son bastante precisos y si existen errores a su salida, estos son corregidos eliminando las perturbaciones, pero son más complejos y por ende más costosos porque tienen demasiados componente dando una mayor probabilidad de descompostura.

Los elementos mencionados a continuación son indispensables a la hora de diseñar y armar un sistema de control de este tipo, ya que con la falta de uno de ellos el funcionamiento del sistema sería incorrecto.

Figura 15. Elementos básicos de un control en lazo cerrado



Fuente: Mecatrónica. Sistemas de Control Electrónico en la Ingeniería Mecánica y Eléctrica, 2010

a. Elemento de comparación: Compara el valor de entrada deseada con el valor de salida obtenida, realizando una operación la cual permite conocer una señal de error utilizada para la compensación del sistema.

   Señal de error=señal del valor de referencia-señal del valor medido

b. Elemento de control: permite utilizar la señal de error para realizar una acción dependiendo si dicha señal es positiva o negativa, así pudiendo obtener una respuesta correctiva de acuerdo con la señal de error generada por el comparador.
c. Elemento de corrección: Permite realizar un cambio en el proceso a fin de alterar la señal obtenida del controlador para predecir que a la salida del sistema se genere la entrada propuesta.
d. Elemento de proceso: Es el elemento que se está controlando, el cual recibe la señal generada por el elemento anterior y reacciona generando una señal de salida.
e. Elemento de medición: Produce una señal utilizable y relacionada con la respuesta del sistema de control para la posterior verificación del punto de referencia.

2.5.3. FUNCIONES DE TRANSFERENCIA

Cualquier sistema lineal puede representarse por una función que permita relacionar la señal de salida con la señal de entrada, a esta ecuación también se la denomina “ecuación característica del sistema” dando así facilidad en el estudio y diseño de controladores.

Figura 16. Representación de la función de transferencia

$$G_{(t)} = \frac{Señal\ de\ salida}{Señal\ de\ entrada} = \frac{Y_{(t)}}{X_{(t)}}$$

Fuente: Ingeniería de control moderna, 2010

Al trabajar con la función de transferencia en ecuación de diferencias o en función del tiempo nos es un poco difícil y complejo estudiarla, por tal razón se utilizan otros métodos que lo faciliten como:

a. La trasformada de Laplace que hace posible estudiar cual sería el comportamiento de la salida del sistema con respecto a cualquier entrada, pudiéndose resolver solo por métodos algebraicos, utilizando como variable de análisis a “s”

Ecuación 1. Función de transferencia en transformada de Laplace

$$G_{(s)} = \frac{Y_{(s)}}{X_{(s)}}$$

Fuente: Ingeniería de control moderna, 2010

b. La transformada Z que evalúa como responde el sistema en tiempo discreto, utilizando como variable de análisis a “z” cuya representación viene a ser por retardos en periodos de tiempo, para dar paso al uso de microcontroladores o cualquier otros análogos para la implementación del “elemento de control”.

Ecuación 2. Función de transferencia en transformada Z

$$G_{(z)} = \frac{Y_{(z)}}{X_{(z)}}$$

Fuente: Sistemas de control en tiempo discreto, 1996

La función de transferencia de un sistema lineal se define como el cociente de la transformada de Laplace de la variable de salida entre la transformada de Laplace de la variable de entrada, suponiendo que todas las condiciones iniciales son cero.(Bolton, Ingeniería de control, 2006)

2.5.3.1. TRANSFORMADA DE LAPLACE

Se puede definir como un método en el cual permite pasar de una ecuación diferencial en función de tiempo a una ecuación algebraica en el dominio de “s”. Si definimos a $F_{(s)}$ como la función en Laplace y a $f_{(t)}$ como la función del sistema en el tiempo, podemos obtener la siguiente ecuación matemática para transformar del dominio del tiempo al de “s”.

Ecuación 3. Transformada de Laplace

$$F_{(s)} = \int_0^{\infty} f_{(t)} \times e^{-st} \partial t$$

Fuente: Ingeniería de control, 2006

A continuación se presentan las transformadas de Laplace de algunas funciones básicas:

Tabla 1. Transformadas de Laplace de algunas funciones básicas

Descripción	$f_{(t)}$	$F_{(s)}$
Impulso	$A \times \partial_{(t)}$	A
Escalón	$A \times u_{(t)}$	$\frac{A}{s}$
Rampa	$A \times t \times u_{(t)}$	$\frac{A}{s^2}$
Exponencial	$A \times e^{at}$	$\frac{A}{s-a}$
Seno	$A \times sen\ (\omega t)$	$\frac{A \times \omega}{s^2+\omega^2}$
Coseno	$A \times cos\ (\omega t)$	$\frac{A \times s}{s^2+\omega^2}$
Primera derivada	$\frac{d}{dt} f_{(t)}$	$s \times F_{(s)} + f_{(0)}$
Segunda derivada	$\frac{d^2}{dt^2} f_{(t)}$	$s^2 \times F_{(s)} + s \times f_{(0)} + \frac{d}{dt} f_{(0)}$
Integral	$\int_0^t f_{(t)} \times dt$	$\frac{F_{(s)}}{s} + f_{(0)}$

Fuente: Ingeniería de control moderna, 2010

2.5.3.2. TRANSFORMADA Z

Se puede definir como un método en el cual permite pasar de una señal en tiempo discreto a una ecuación en el dominio de "z". Si definimos a $X_{(z)}$ como la función en Z y a $x_{(kT)}$ como la función del sistema en tiempo discreto, podemos obtener la siguiente ecuación matemática.

Ecuación 4. Transformada Z unilateral

$$X_{(z)} = \sum_{k=0}^{\infty} x_{(kT)} \times z^{-\mathrm{k}}$$

Fuente: Sistemas de control en tiempo discreto. 1996

Siendo $x_{(kT)}= x_{(t)}\partial_{(k)}$, teniendo la observación de que la transformada Z unilateral supone: $x_{(t)}=0$ para t<0, o $x_{(kT)}=0$ para k<0.Sin embargo, si tenemos valores para t<0, se debe aplicar la siguiente ecuación denominada transformada Z bilateral.

Ecuación 5. Transformada Z bilateral

$$X_{(z)} = \sum_{k=-\infty}^{\infty} x_{(kT)} \times z^{-\mathrm{k}}$$

Fuente: Mechatronic system control, logic, and data acquisition. 2008

A continuación se presentan la transformada Z de algunas funciones básicas:

Tabla 2. Transformadas Z de algunas funciones básicas

Descripción	$X_{(kT)}$	$X_{(z)}$
Impulso	$\partial_{(kT)}$	1
Escalón	$u_{(kT)}$	$\frac{z}{z-1}$
Rampa	$kT \times u_{(kT)}$	$\frac{T \times z}{(z-1)^2}$
Exponencial	e^{akT}	$\frac{z}{z-e^{aT}}$
Seno	$sen\ (\omega kT)$	$\frac{zsen\ (\omega T)}{z^2-2z\ cos\ (\omega T)+1}$
Coseno	$cos\ (\omega kT)$	$\frac{z[z-cos\ (\omega T)]}{z^2-2z\ cos\ (\omega T)+1}$

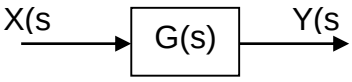
Fuente: Teoría de control. Diseño electrónico, 1999

2.5.4. DIAGRAMA DE BLOQUES

Los diagramas de bloques vienen a ser un enfoque más reducido de cómo trabaja un sistema lineal, suponiendo que el sistema global se puede representar por un bloque cerrado en el cual consta la función de transferencia del mismo. Permite estudiar cómo va a interactuar un sistema si lo agrupamos con otros.

Los diagramas son una representación gráfica del flujo de señales y de la función realizada por cada componente de un sistema, conteniendo información respecto a su comportamiento dinámico y no sobre su constitución interna.(Gomáriz Castro, Biél Solé, Matas Alcalá, & Reyes Moreno, 1999)

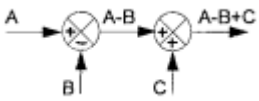
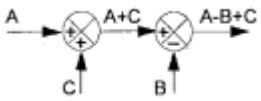
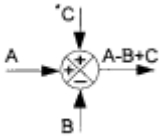
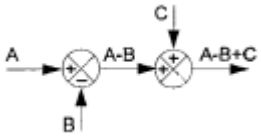
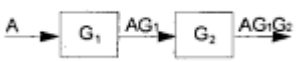
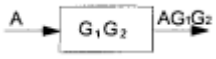
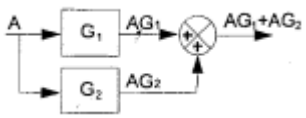
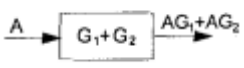
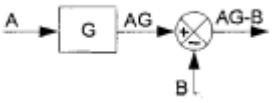
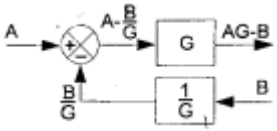
Figura 17. Diagrama de bloques de la función de transferencia

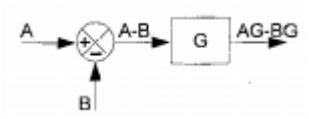
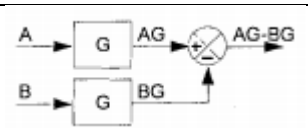
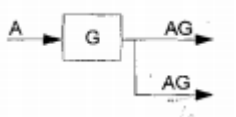
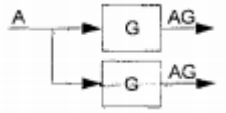
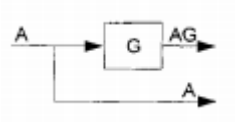
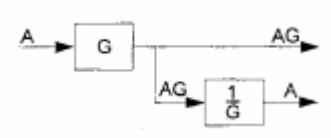
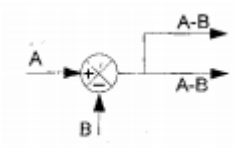
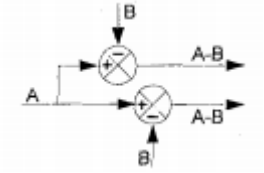
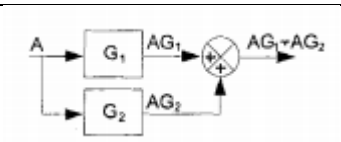
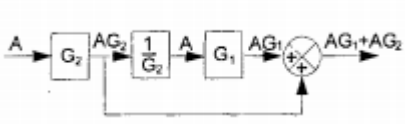
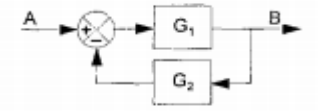
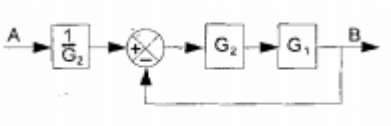
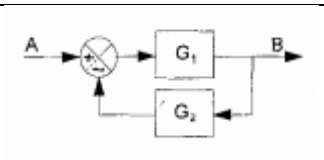
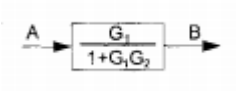


Fuente: Ingeniería de control, 2006

A continuación se presenta una tabla con las operaciones y equivalencias que se pueden hacer con estos diagramas:

Tabla 3. Equivalencias entre diagramas de bloques

	Diagrama original	Diagrama equivalente
1		
2		
3		
4		
5		
6		

7		
8		
9		
10		
11		
12		
13		

Fuente: Teoría de control. Diseño electrónico, 1999

2.5.5. MODELOS DE SISTEMAS DINÁMICOS

Muchos nos hacemos la misma pregunta: ¿Cuál es el primer paso?, puesto que si no es contestada no podemos proceder a encontrar la ecuación característica del sistema; pues para ello, lo primero que se debe hacer es representar al sistema por ecuaciones de diferencias y aplicando la transformada de Laplace, proceder a encontrar la función de transferencia.

Básicamente la mayoría de los sistemas lineales son de primero y segundo orden, denotando ecuaciones y transformadas que nos ayudan a describir su funcionamiento y principales características.

2.5.5.1. SISTEMAS DE PRIMER ORDEN

Algunos sistemas lineales se pueden representar por la siguiente ecuación de diferencias de primer orden, siendo a, b y c constantes.

Ecuación 6. Ecuación diferencial de primer orden

$$a\frac{d}{dt}y_{(t)} + by_{(t)} = cx_{(t)}$$

Fuente: Ingeniería de control, 2006

Aplicando la transformada de Laplace suponiendo condiciones iniciales cero y desarrollando la ecuación en una función de transferencia, reemplazando “A” por “c/b” y “τ” por “a/b” tenemos:

Ecuación 7. Función de transferencia de primer orden

$$\frac{Y_{(s)}}{X_{(s)}} = \frac{A}{\tau s + 1}$$

Fuente: Ingeniería de control, 2006

Teniendo las siguientes características con respuesta al escalón unitario:

1. “A” es la ganancia del sistema en condiciones de estado estacionario.
2. "τ" es la constante de tiempo.
3. Tiempo de establecimiento

Ecuación 8. Tiempo de establecimiento

$$t_s = 4\tau$$

Fuente: Ingeniería de control moderna, 2010

2.5.5.2. SISTEMAS DE SEGUNDO ORDEN

La mayoría de sistemas lineales también se puede representar por la siguiente ecuación de diferencias de segundo orden.

Ecuación 9. Ecuación diferencial de segundo orden

$$\frac{d^2}{dt^2}y_{(t)} + 2\varepsilon\omega_n\frac{d}{dt}y_{(t)} + \omega_n^2 y_{(t)} = b\omega_n^2 x_{(t)}$$

Fuente: Ingeniería de control, 2006

Aplicando la transformada de Laplace suponiendo condiciones iniciales cero y desarrollando la ecuación en una función de transferencia, tenemos:

Ecuación 10. Función de transferencia de segundo orden

$$\frac{Y_{(s)}}{X_{(s)}} = \frac{b\omega_n^2}{s^2 + 2\varepsilon\omega_n s + \omega_n^2}$$

Fuente: Ingeniería de control, 2006

Teniendo las siguientes características con respuesta al escalón unitario:

1. “b” es la ganancia del sistema en condiciones de estado estacionario.
2. "ω_n" es la frecuencia natural con la que oscila el sistema.
3. "ε" es el coeficiente de amortiguamiento

Figura 18. Tipo de respuesta dinámica

Respuesta del sistema	ε=0	Inestable
	0< ε<1	Sub amortiguada
	ε=1	Críticamente amortiguada
	ε>1	Sobre amortiguada

Fuente: Ingeniería de control moderna, 2010

4. Tiempo de establecimiento.

Ecuación 11. Tiempo de establecimiento

$$t_s = \frac{4}{\varepsilon\omega_n}$$

Fuente: Ingeniería de control moderna, 2010

5. Sobre impulso.

Ecuación 12. Máximo sobre impulso

$$M_p = e^{-\frac{\pi\varepsilon}{\sqrt{1-\varepsilon^2}}}$$

Fuente: Ingeniería de control moderna, 2010

6. Tiempo del máximo sobre impulso.

Ecuación 13. Tiempo de sobre impulso

$$t_p = \frac{\pi}{\omega_n\sqrt{1-\varepsilon^2}}$$

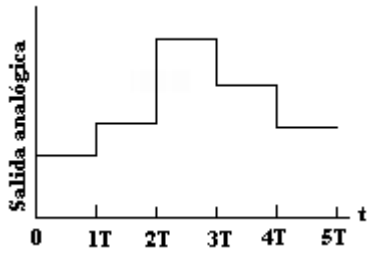
Fuente: Ingeniería de control moderna, 2010

2.5.6. MUESTREO Y RECONSTRUCCIÓN

La mayoría de veces existe la incertidumbre de que no se conocen todas las características del sistema para modelarlo de forma teórica, por el cual se utilizan el método de identificación de sistemas que consta principalmente del muestreo de datos y reconstrucción de la señal de salida debido a la acción de una señal de entrada, para ello se debe conocer: ¿cómo? y ¿cuáles? son los pasos o acciones a tomar para su respectiva obtención del modelo dinámico. (López Guillén, SN)

Una señal en tiempo continuo se reconstruye a partir de una señal en tiempo discreto mediante la retención del convertidor digital-analógico (DAC), hasta llegar al siguiente impulso convertido, resultando una señal analógica en forma de escalera. (Bolton, Ingeniería de control, 2006)

Figura 19. Señal analógica de una DAC

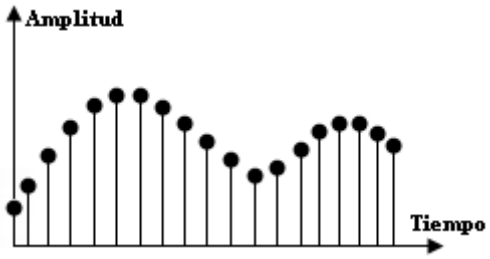


Fuente: Ingeniería de control, 2006

2.5.6.1. SEÑAL EN TIEMPO DISCRETO

Las señales en tiempo discreto no tienen una variación continua como las señales analógicas, sin embargo, existen en un determinado conjunto de valores tomados a instantes de tiempo denotándose una señal conformada de impulsos de amplitud igual al valor de la señal continua en ese instante de tiempo. (Gomáriz Castro, Biél Solé, Matas Alcalá, & Reyes Moreno, 1999)

Figura 20. Señal discreta



Fuente: Ingeniería de control, 2006

2.5.6.2. TEOREMA DE MUESTREO

Los sistemas en tiempo discreto conllevan a la reconstrucción de alguna señal requerida, siendo el teorema de muestreo fundamental para este tipo de sistemas y una aproximación más real del mismo; si el periodo de muestreo es muy largo se pierde información del sistema, y si es muy corto se incrementa la carga de procesamiento además de la necesidad de un convertidor digital-analógica (ADC)

que sea más rápido. Para ello se han ideado reglas empíricas para la elección del periodo de muestreo.

1. El teorema de Shannon: “La mínima frecuencia de muestreo para poder recuperar una señal previa al muestreo, a partir de una señal muestreada a través de un filtro pasa bajos ideal debe ser el doble de la frecuencia máxima de la señal a muestrear”. (Gomáriz Castro, Biél Solé, Matas Alcalá, & Reyes Moreno, 1999)
2. Si la planta o sistema tiene una constante de tiempo dominante, entonces se elige un período de muestreo que sea alrededor de la décima parte de dicha constante. (Bolton, Ingeniería de control, 2006)

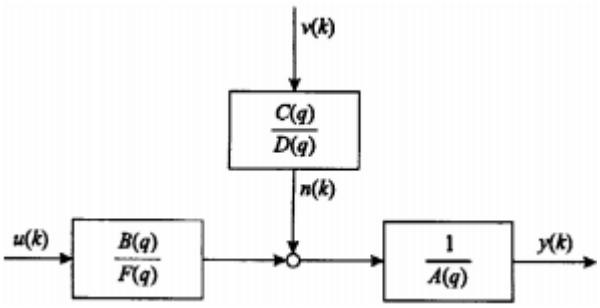
2.5.6.3. SISTEMAS DE ADQUISICIÓN DE DATOS

Son equipos que permiten tomar señal físicas y convertirlas en datos que posteriormente se los podrá utilizar para su procesamiento y visualización; por lo tanto se emplea para el muestreo de la señal que se va a modelar, entonces, el sistema constaría principalmente de tres elementos: transductor para la obtención de la señal, acondicionamiento adaptando la señal del transductor para su procesamiento, y el ADC que permite cuantificar la señal en valores que el sistema se adquisición de datos puede manipular; además de una etapa de salida para el envío de los datos muestreados, ya sea por comunicaciónRS232, USB, RS485, etc, para la conexión con un ordenador.(Meythaler Naranjo, 2005)

2.5.6.4. MÉTODOS DE REGRESIÓN

El presente trabajo sólo se enfoca a sistemas lineales, es decir, sistemas cuya señal de salida puede ser modelada, para ello existen procedimientos para el modelado de señales enfocándonos a los métodos de identificación paramétrica que analiza los modelos lineales.

Figura 21. Estructura general de los modelos discretos lineales



Fuente: Identificación con modelos discretos para sistemas lineales. Modelo matemático y aplicaciones. 2003

Describiendo al sistema global como: $u_{(k)}$ la entrada del sistema, $v_{(k)}$ ruido blanco acoplado, $y_{(k)}$ la salida o respuesta y los demás recuadros como funciones de transferencia en transformada Z denotando a $\frac{1}{A_q}$ como la función de transferencia de un filtro, describiendo la siguiente ecuación de modelado:

Ecuación 14. Ecuación general de los modelos discretos lineales

$$y_k = \frac{B_q}{F_q A_q} u_k + \frac{C_q}{D_q A_q} v_k$$

Fuente: Identificación con modelos discretos para sistemas lineales. Modelo matemático y aplicaciones. 2003

Existen diversos modelos para la aproximación y modelado paramétrico de sistemas lineales, algunos de los cuales son:

A. Modelos auto-regresivos

Especialmente caracterizados porque no dependen de la señal de entrada para el modelamiento del sistema, entonces $u_k = 0$ obteniendo los siguientes modelos:

1. **Modelo AR (Auto-regresivo):** modelo con serie de tiempo con sólo el polinomio del denominador $D_{(q)}$.

2. **Modelo MA (Promedio móvil):** modelo con serie de tiempo con sólo el polinomio del numerador $C_{(q)}$.
3. **Modelo ARMA (Promedio móvil y auto-regresivo):** modelo con serie de tiempo con sólo el polinomio del numerador $C_{(q)}$ y denominador $D_{(q)}$.

Un modelo basado en una serie de tiempo sin tener en cuenta la variable de entrada no puede ser preciso, estos modelos pertenecen a la clase de modelos de error de cuantificación. (Soriano, Escobar, & Peña, 2003)

B. Modelos auto-regresivos con variable exógena

Son modelos más precisos ya que en su cálculo incorporan la variable de entrada u_k llamada exógena, obteniendo los siguientes modelos:

1. **Modelo ARX (Auto-regresivo con entrada exógena):** solo es una extensión del modelo AR.

Ecuación 15. Ecuación del modelo ARX

$$y_k = \frac{B_q}{A_q} u_k + \frac{1}{A_q} v_k$$

Fuente: Identificación con modelos discretos para sistemas lineales. Modelo matemático y aplicaciones. 2003

2. **Modelo ARMAX (Promedio móvil y auto-regresivo entrada exógena):** es una extensión del modelo ARMA.

Ecuación 16. Ecuación del modelo ARMAX

$$y_k = \frac{B_q}{A_q} u_k + \frac{C_q}{A_q} v_k$$

Fuente: Identificación con modelos discretos para sistemas lineales. Modelo matemático y aplicaciones. 2003

Estos modelos son válidos si el ruido se introduce al modelo antes del filtrado, es decir, son razonables si el ruido se introduce al proceso primero. (Soriano, Escobar, & Peña, 2003)

C. Modelo de error de salida

Modelos cuya característica es que el ruido incorporado es independiente del modelo del proceso determinístico, es decir, el ruido no afecta a la dinámica del sistema así se asume que solo afecte a la salida del proceso directamente. (Soriano, Escobar, & Peña, 2003)

Ecuación 17. Ecuación del modelo OE

$$y_k = \frac{B_q}{F_q} u_k + v_k$$

Fuente: Identificación con modelos discretos para sistemas lineales. Modelo matemático y aplicaciones. 2003

2.5.6.4.1. IDENTIFICACIÓN DE PARÁMETROS

Una técnica básica de identificación paramétrica para sistemas SISO (single input, single output), es un método ampliamente utilizado por su sencillez, eficacia y eficiencia computacional. El método de los mínimos cuadrados es simple con la propiedad de ser lineal en sus parámetros, así pueden ser calculados analíticamente.

Ecuación 18. Ecuación de regresión

$$\theta = [X^T . X]^{-1} X^T . Y$$

Fuente: Identificación con modelos discretos para sistemas lineales. Modelo matemático y aplicaciones. 2003

Definiendo:

a. X es el vector de información de los datos obtenidos experimentalmente.

$X_{(k)}=[\ -Y_{(k-1)}\ -Y_{(k-2)}-Y_{(k-3)}\ \ .....\ \ -Y_{(k-n)}\ \ U_{(k-1)}\ \ U_{(k-2)}\ \ U_{(k-3)}\ \ .....\ \ U_{(k-n)}]$

b. Y representa al vector columna de valores de la señal de salida del sistema.
c. θ es el vector columna de parámetros correspondientemente al vector de información.

   $\theta = [a_1\ ;\ a_2\ ;a_3\ ;\ ....;a_n;;b_1\ ;\ b_2\ ;b_3\ ;\ ....;b_n]$
d. X^T es el vector traspuesto de X.
e. $[X^T.X]^{-1}$vector inverso de $[X^T.X]$

Correspondientemente a la siguiente ecuación de diferencias:

Ecuación 19. Ecuación de diferencia

$$Y_{(k-1)=} - a_1Y_{(k-1)} - a_2Y_{(k-2)} – a_3Y_{(k-3)} ...– a_nY_{(k-n)}+ b_1U_{(k-1)} + b_2U_{(k-2)} + b_3U_{(k-3)} ...+ b_nU_{(k-n)}$$

Fuente: Identificación con modelos discretos para sistemas lineales. Modelo matemático y aplicaciones. 2003

2.5.6.5. CONVERSIÓN ENTRE TRANSFORMADAS

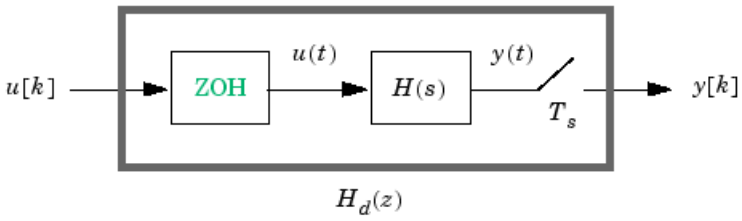
Existen diversos métodos de conversión para pasar de una transformada a otra, esencialmente se utiliza cuando el usuario pretende trabajar con una transformada u otra sin la necesidad de la utilización de la transformada inversa, o para la implementación de la función de transferencia en circuitos que emulen las características de dicho sistema.

Ocupándose de la transformada de Laplace con la implementación en circuitos analógicos, es decir, cuyos valores de la señal de salida sean válidos para todo instante de tiempo utilizando como elemento principal al amplificador operacional. Y la transformada Z con la implementación en circuitos digitales, es decir, cuyos valores de la señal de salida sean válidos solo en instante de tiempo determinados utilizando como elemento principal al microcontrolador o algún otro análogo.

A. Retenedor de orden cero (ZOH)

Proporciona una coincidencia exacta entre los sistemas continuos y de tiempo discreto en el dominio del tiempo para las entradas de escalera, denotando que u_t=u_k para el labzo de tiempo comprendido entre un periodo de muestreo.

Figura 22. Diagrama de bloques del ZOH



Fuente: Control System Toolbox. User's Guide.2013

B. Retenedor de primer orden (FOH)

Llamada también triángulo de aproximación en cual proporciona una coincidencia exacta entre los sistemas continuos y de tiempo discreto en el dominio del tiempo para las entradas lineales a trozos. Para activar las muestras de entrada $u_{(k)}$ en una continua de entrada $u_{(t)}$, FOH utiliza la interpolación lineal entre las muestras en un periodo de muestreo:

Ecuación 20. Ecuación del FOH

$$u(t) = u[k] + \frac{t - kT_s}{T_s}\left(u[k+1] - u[k]\right), \quad kT_s \le t \le (k+1)T_s$$

Fuente: Control System Toolbox. User's Guide.2013

C. Aproximación de Tustin

También llamada aproximación bilineal, se obtiene el mejor partido de dominio de frecuencia entre los sistemas continuos y discretos. Este método se refiere a las funciones de transferencia de dominio *Z* utilizando la aproximación *s-dominio* y:

Ecuación 21. Ecuación bilineal

$$H_d(z) = H(s'), \qquad s' = \frac{2}{T_s}\frac{z-1}{z+1}$$

Fuente: Control System Toolbox. User's Guide.2013

D. Equivalencia a juego ceros y polos

Se aplica sólo a los sistemas SISO. Los sistemas continuos y discretos se emparejan ganancias DC. Sus polos y ceros están relacionados por la transformación:

Ecuación 22. Ecuación a juego de ceros y polos

$$z_i = e^{s_i T_s}$$

Fuente: Control System Toolbox. User's Guide.2013

donde:

a. z_i es el th polo o cero del sistema de tiempo discreto *i.*
b. s_i es *i* th del polo o cero del sistema de tiempo continuo.
c. T_s es el tiempo de muestreo.

E. Mapeo de pulso invariante

Produce un modelo de tiempo discreto con la misma respuesta de impulso como el sistema de tiempo continuo. Por ejemplo, comparar la respuesta de impulso de un sistema continuo de primer orden con la discretización de impulso-invariante.(MathWorks, Inc, 2013)

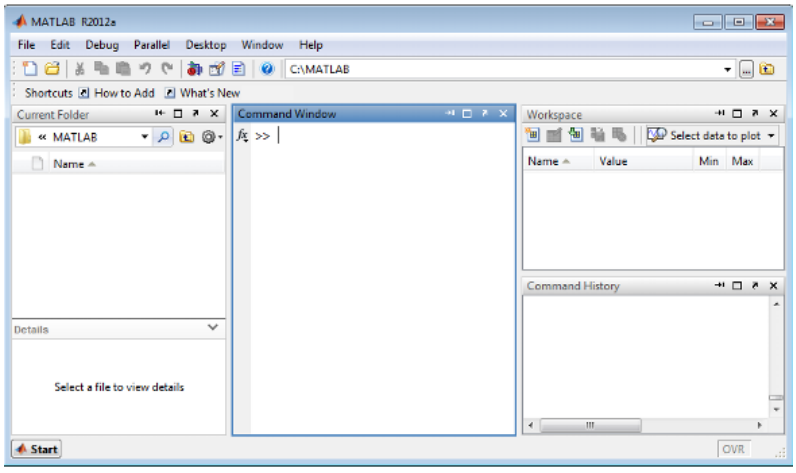
2.6. SOFTWARE DE PROGRAMACIÓN

En muchos casos para el modelamiento de sistemas lineales se hace necesario una interfaz gráfica la cual ayude en el procesamiento de los datos, para ello se ha elegido a MATLAB por su gran desempeño y aplicabilidad en varios ámbitos, exclusivamente en el área de sistemas de control para el procesamiento de señales y comunicaciones. En este proyecto se trabajará con la versión R2012a.

2.6.1. MATLAB

MATLAB es un lenguaje de alto nivel con ambiente interactivo para el cálculo numérico, visualización y programación. Con este software puede analizar los datos, crear algoritmos y crear modelos y aplicaciones. EL lenguaje, las herramientas y las funciones de matemáticas integradas le permiten explorar múltiples enfoque y llegar a una solución más rápida que con las hojas de cálculo o lenguajes de programación tradicionales como C/C++ o Java.(MathWorks, Inc, 2013)

Figura 23. Escritorio de Matlab



Fuente: MATLAB primer, 2013

El escritorio de MATLAB incluye:

1. **Curren folder:** permite acezar a tus archivos contenidos en la computadora.

2. **Command Window:** permite introducir comandos en la línea de comandos indicada por (>>).
3. **Workspace:** permite explorar los datos que se crean o se importan desde archivos.
4. **Comand history:** permite ver o volver a ejecutar los comandos que se han ejecutado en la línea de comandos.

2.6.1.1. ALGORITMOS

MATLAB cuenta con funciones internas que nos ayudan a la creación de algoritmos, teniendo en cuenta caracteres y palabras reservadas para su funcionamiento.

Tabla 4. Funciones y caracteres básicos

Función o carácter	Descripción
clc	Limpia la ventana de comandos
…	Permite continuar los comandos en la siguiente línea
;	Al final de la línea de comandos, inhibe la visualización de su resultado
help	Permite acceder a la ayuda general de Matlab
help *nombre*	Permite acceder a la ayuda de una función específica
clear all	Borra todas las variables anteriores utilizadas
pi	3.1415…
exp(1)	Exponencial e=2.71828…
i=j	$\sqrt{-1}$
inf	Infinito (1/0)
NaN	Indeterminación (0/0)
realmin	Menor número real utilizado por Matlab
realmax	Mayor número real utilizado por Matlab.
sin	Seno
cos	Coseno

tan	Tangente
sec	Secante
csc	Cosecante
cot	Cotangente
log	Logaritmo natural
sqrt	Raíz cuadrada
abs	Valor absoluto
input	Permite el ingreso de los datos desde el teclado asignándolo a una variable, este comando se lo puede emplear con un mensaje en la línea de comandos.
fprintf	Permite la visualización de un valor numérico o el resultado de una expresión guardada por el usuario.
disp('*mensaje*')	Visualiza el mensaje en la línea de comandos.

Fuente: Matlab herramienta para el aprendizaje.

Estos comandos se los puede utilizar y agrupar en scripts o archivos *.m que permite ejecutar una serie de funciones propias de MATLAB realizando un algoritmo o programa, comúnmente dichas funciones utilizan bucles para realizar más procesos o ahorrar código de programación.

A. Bucle for

Permite ejecutar de forma repetida uno o varios grupos de comandos varias veces. Su estructura es la siguiente:

for variable=valor_inicial : valor_final

Comandos

end

B. Bucle while

Permite ejecutar de forma repetida uno o varios grupos de comandos mientras se cumple una condición. Su estructura es la siguiente:

while *condición*

    Comandos

end

C. Bucle if-else

Permite ejecutar una secuencia de comandos si se cumplen ciertas condiciones. Su estructura es la siguiente:

if condición

    Comandos

elseif condición

    Comandos

else

    Comandos

end

if condición

    Comandos

else

    Comandos

end

D. Bucle switch y case

Permite ejecutar cierta secuencia de comandos basados en el valor de una variable o expresión. Su estructura es la siguiente:

swith expresión (escalar o cadena)

    case valor1

Comandos

case valor2

Comandos

…….

Otherwise

Comandos

end

Para más información sobre la funciones y comandos propios de MATLAB se recomienda la ayuda propia del programa, esta contiene la sintaxis de cada comando y ejemplos de como se los puede aplicar.

2.6.1.2. CONTROL SYSTEM TOOLBOX

Proporciona algoritmos y herramientas estándar para analizar de forma sistemática, el diseño y puesta a punto de sistemas de control lineales. Puede especificar el sistema en su función de transferencia, espacio de estado, ganancia-polos-ceros, o el modelo de respuesta en frecuencia.

Herramientas interactivas y funciones de línea de comandos, como respuesta al escalón y de Bode, permiten visualizar el comportamiento del sistema en el dominio del tiempo y el dominio de la frecuencia. (MathWorks, Inc, 2013)

Tabla 5. Funciones básicas del control system toolbox

Función	Descripción
ss	Crear modelos en espacio de estado.
tf	Crea modelos en funciones de transferencia.
zpk	Crea modelos en forma cero-polo-ganancia.

ssdata	Accede a los datos del modelo de espacio de estados
tfdata	Accede a los datos del modelo de la función de transferencia
zpkdata	Accede a los datos del modelo de cero-polo-ganancia
c2d	Convierte el modelos de forma continua a discreta
d2c	Convierte el modelo de forma discreta a continua
feedback	Permite conectar dos modelos en feedback
parallel	Permite conectar dos modelos en forma paralela
series	Permite conectar dos modelos en serie.
impulse	Grafica el sistema dinámico con respuesta al impulso
initial	Grafica la respuesta del sistema con condición inicial
step	Grafica la respuesta del sistema con entrada paso unitario
bode	Grafica la respuesta en diagramas de bode en magnitudes y fase
nichols	Grafica la respuesta en frecuencia de Nichols
nyquist	Grafica la respuesta en frecuencia de Nyquist
modred	Modelo de orden reducido
rlocus	Grafica el sistema en el lugar geométrico de las raíces
pidtune	Algoritmo de ajuste PID el control de la planta lineal

Fuente: Control System Toolbox. User's Guide. 2013

2.6.1.3. SYSTEM IDENTIFICATION TOOLBOX

Construye modelos matemáticos de sistemas dinámicos a partir de datos de entrada y salida de medición. Proporciona funciones de MATLAB, bloques de Simulink, y una herramienta interactiva para crear y utilizar modelos de sistemas dinámicos. Usted puede utilizar el dominio del tiempo y dominio de la frecuencia de los datos de entrada-salida para identificar las funciones de transferencia de tiempo continuo y de tiempo discreto, modelos de procesos y modelos de espacio de estado. (MathWorks, Inc, 2013)

Tabla 6. Funciones básicas del system identification toolbox

Función	Descripción
iddata	Datos en el dominio del tiempo o frecuencia
misdata	Reconstrucción de datos de entrada y salida perdidos
advice	Análisis y recomendación para datos o estimación de modelos
ar	Estimación de parámetros para modelo AR
armax	Estimación de parámetros para modelo ARMAX
arx	Estimación de parámetros para modelo ARX
fft	Transformación de iddata al dominio de la frecuencia
oe	Estimación de parámetros para modelo Output-Error
tfest	Estimación de parámetros para modelo de función de transferencia continua.
init	Envía o cambiar aleatoriamente los valores iniciales paramétricos
sim	Simularla respuesta de los modelos identificados a entradas arbitrarias
ident	Abre la interfaz gráfica de System Identification Tool
rarmax	Estimación recursiva de parámetros para modelo ARMAX
rarx	Estimación recursiva de parámetros para modelo ARX
roe	Estimación recursiva de parámetros para modelo Output-Error

Fuente: System Identification Toolbox. User's Guide. 2013

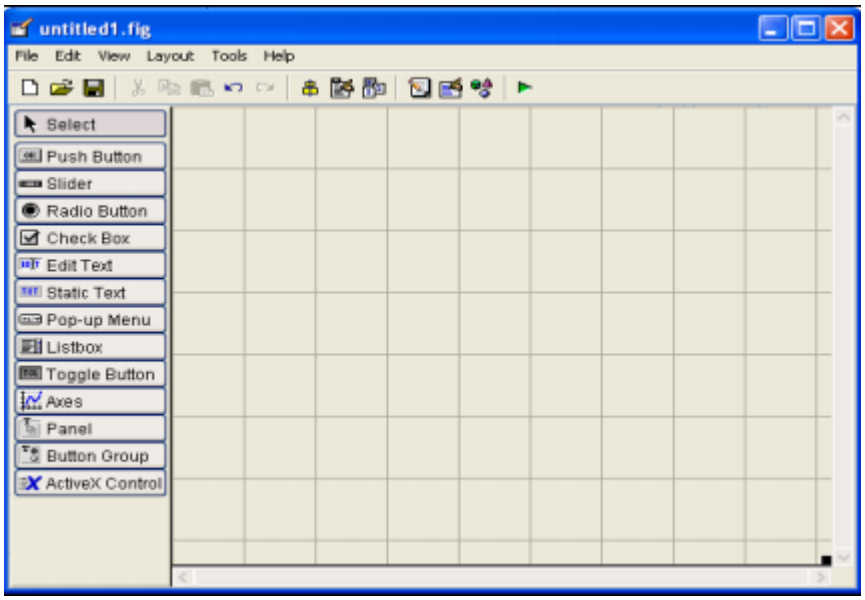
2.6.1.4. INTERFAZ GRÁFICA DE USUARIO

Es una visualización gráfica en una o más ventanas que contienen controles, denominados componentes, que permiten a un usuario realizar tareas interactivas. A diferencia decodificación de programas para realizar las tareas, el GUI no necesita conocer los detalles de cómo se realizan las tareas.

En los componentes del GUI pueden incluir menús, barras de herramientas, botones, botones de radio, cuadros de lista y deslizadores, sólo para nombrar unos

pocos. GUIs creados usando herramientas de MATLAB pueden también realizar cualquier tipo de cálculo, leer y escribir archivos de datos, comunicarse con otras interfaces gráficas de usuario y mostrar datos como tablas o gráficos. (MathWorks, Inc, 2013)

Figura 24. Entorno GUIDE



Fuente: MATLAB primer. 2013

Conteniendo los siguientes componentes:

a. **Check box**: 'checkbox'-Indica el estado de una opción o atributo.
b. **Editable Text**: 'edit'-Caja para editar texto
c. **Pop-up menu**: 'popupmenu'-Provee una lista de opciones
d. **List Box**: 'listbox'-Muestra una lista deslizable
e. **PushButton**: 'pushbutton'-Invoca un evento inmediatamente
f. **Radio Button**: 'radio'-Indica una opción que puede ser seleccionada
g. **ToggleButton**: 'togglebutton'-Solo dos estados, "on" o "off"
h. **Slider**: 'slider'-Usado para representar un rango de valores
i. **Static Text**: 'text'-Muestra un string de texto en una caja
j. **Panel button**: Agrupa botones como un grupo

k. **ButtonGroup**: Permite exclusividad de selección con los radio button

Cada componente tiene sus propias características; para su visualización se debe hacer click derecho en un componente y seleccionar "*Propiety Inspector*" que permite la manipulación delas características del objeto.

2.6.1.4.1. FUNCIONAMIENTO DE UNA APLICACIÓN GUI

La aplicación GUI creada por el usuario consta principalmente de dos archivos con el mismo nombre pero con diferente extensión:

a. El archivo *.m que contiene todo el código de programación correspondientemente a cada componente del sistema.
b. El archivo *.fig que contiene todos los elementos gráficos de la interfaz.

2.6.1.4.2. SENTENCIAS GET Y SET

Existen dos sentencias que son fundamentales para la programación en un GUI. Estas instrucciones son:

a. **GET:** Nos sirve para llamar a un dato de algún componente gráfico de la interfaz creada.
b. **SET:** Asigna el valor de una variable a un componente gráfico de la interfaz creada.

Para más información sobre la funciones, comandos y desarrollo de GUI's de MATLAB se recomienda la ayuda propia del programa, o además del libro "Manual de interfaz gráfica de usuario en Matlab" de Diego Barragán en el cual se explica y utiliza más propiedades y opciones sobre la programación en la interfaz gráfica.

CAPÍTULO III

DISEÑO Y CONSTRUCCIÓN DE LA TARJETA DE ADQUISICIÓN DE DATOS

3.1. INTRODUCCIÓN

En este capítulo se explica el proceso de diseño, programación y construcción de una tarjeta de adquisición de datos (DAQ) utilizando comunicación USB. Se muestra el proceso de selección y caracterización de la DAQ, como los puertos de entrada-salida que posee, incluido el algoritmo de recolección de datos; además de la zona de aplicación en la cual se pueden adquirir los valores del tiempo transitorio del sistema a modelar para la reconstrucción de su función de transferencia.

3.2. DISEÑO DE LA TARJETA

En la actualidad, se puede conseguir componentes electrónicos eficientes, con bajo consumo de energía y cuyos circuitos son más complejos e inteligentes; como por ejemplo: el microcontrolador en su empaquetado nos encontramos con circuitos de ayuda que mejoran las capacidades del mismo para la realización de grandes aplicaciones, una de las cuales y de alta necesidad es la interacción entre el microcontrolador con la PC, especialmente para la obtención de datos externos para su procesamiento y análisis.

3.2.1. CARACTERÍSTICAS DE DISEÑO

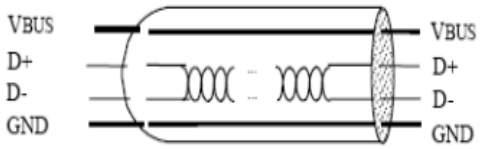
La acción o proceso del diseño de la tarjeta de adquisición de datos se lo debe realizar conforme a ciertas características que se lo debe definir conforme a la zona de aplicación en donde se va a desenvolver y facilidades de la obtención de los materiales o paquetes, ya sea para la construcción o simplemente la adquisición de la tarjeta. Como características importantes para su elección, tenemos:

1. Velocidad de comunicación y transmisión de datos
2. Tiempo de muestreo
3. Entradas y salidas análogas y digitales
4. Número de canales de entradas analógicas
5. Rangos de valores a adquirir
6. Precisión de lectura de las entradas analógicas
7. Pre-procesamiento de señales adquiridas
8. Costo

3.2.1.1. COMUNICACIÓN USB

La comunicación USB proporciona ventajas y facilidades que sus predecesores, sin embargo, por la utilización del protocolo de comunicación solo se puede tener un Host (dispositivo maestro USB de control primario) que permite la comunicación con otros dispositivos USB; la utilización del Hub permite la conexión de más dispositivos para la comunicación con el Host. (Pool, 2010)

Figura 25. Cable USB



Fuente: Comunicación entre MATLAB y PIC de MICROCHIP usando puerto USB. 2010

La comunicación USB acepta cuatro tipos de transferencias de datos, los cuales son:

A. Control Transfers

Es utilizado para configurar un dispositivo al momento en que se conecta, tiene alta prioridad con protección automática frente a un error obteniéndose una alta velocidad de transmisión de datos sin pérdida alguna. (Pool, 2010)

B. Bulk Data Transfers

Se emplea para la comunicación por transmisión de paquetes de datos, pudiendo variar el volumen del mismo dependiendo de otras actividades del bus con una ráfaga de datos secuencial. Es usado en scanner o cámaras. (Pool, 2010)

C. Interrupt Data Transfers

Utilizado para la entrega oportuna y fiable de datos, consiste típicamente en notificación de eventos, caracteres, o coordenadas que se organizan como uno o más bytes. Por ejemplo, características de respuesta de retroalimentación en la que se envía muy pocos datos. (Pool, 2010)

D. Isochronous Data Transfers

También llamada transferencia continua o en tiempo real, ocupa una cantidad predefinida de ancho de banda USB con una latencia de entrega negociada. Para la entrega de los datos en la práctica se asignan una porción de ancho de banda dedicado USB para asegurar que los datos pueden ser entregados a la velocidad deseada. El USB también está diseñado para un retraso mínimo de las transferencias de datos isócronos. (Pool, 2010)

3.2.1.2. PERIFÉRICOS

Al momento de definir la tarjeta de adquisición de datos a utilizar se debe tomar en cuenta los periféricos o pines de entrada y salida que posee, como por ejemplo: obteniendo una DAQ con “n” entradas y salidas análogas conjuntamente con “m” entradas y salidas digitales de las cuales se han de ocupar solamente un escaso número de entradas y salidas de cada uno, resultaría una pérdida en cuanto a hardware por el exceso de periféricos; además de las funciones internas no utilizadas de la DAQ como timers, interrupciones externas e internas, etc.

3.2.2. SELECCIÓN DE LA DAQ

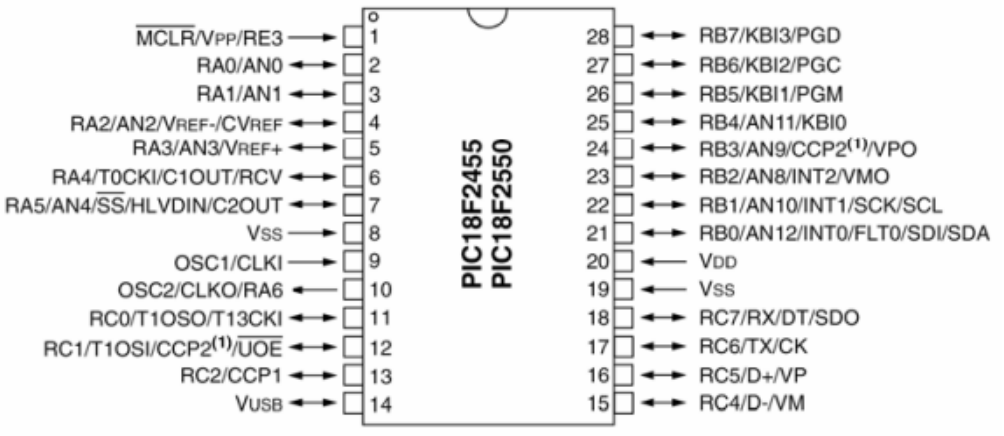
En vista de todo lo mencionado anteriormente se ha optado por el diseño y construcción de una DAQ con microcontrolador PIC18F2550 por su capacidad de comunicación USB, y la facilidad de adquirir dicho elemento, resultando un costo

total bajo en la implementación de la misma. Para el cual microchip en su página web proporciona ayuda para el desarrollo de esta aplicación facilitando librerías para la interfaz con la PC, además de software de programación para el desarrollo y modificación del programa final que se le va a incorporar al microcontrolador.

Los puntos importantes que ofrece este microcontrolador son:

1. Velocidad de comunicación USB:
   a. Low-Speed con 1.5 Mb/s
   b. Full-Speed con 12 Mb/s
2. Tipos transferencia de datos que soporta:
   a. Control transfers
   b. Interrupt transfers
   c. Isochronous transfers
   d. Bulk transfers
3. Periféricos:
   a. Entradas analógicas
   b. Entradas y salidas digitales
4. Precisión de lectura analógica de 10 bits de resolución
5. En modo normal permite una lectura analógica del ADC con límites de 0V a 5V
6. Utilizando oscilador externo de 20 MHz se tiene un tiempo de lectura máxima del ADC aproximadamente de 0.82 ms
7. Tiene hasta 10 canales de lectura ADC.

Figura 26. Diagrama de pines de microcontrolador 18F2550



Fuente: Hoja de datos, Microchip PIC18F2550

A continuación se presenta una tabla con las características y funciones generales del microcontrolador utilizado:

Tabla 7. Descripción y características del microcontrolador PIC 18F2550

Características	PIC18F2550
Frecuencia máxima de operación	48 MHZ
Memoria de programación (Bytes)	32768
Memoria de datos (Bytes)	2048
Memoria de datos EEPROM (Bytes)	256
Fuentes de interrupción	19
Puertos entrada/salida	Ports A, B, C, (E)
Temporizadores	4
Módulos CCP	2
Tipos de Comunicación	MSSP, USART, USB
Converso análogo a digital de 10 bits	10 entradas analógicas
Comparadores	2

Fuente: Hoja de datos, Microchip PIC18F2550

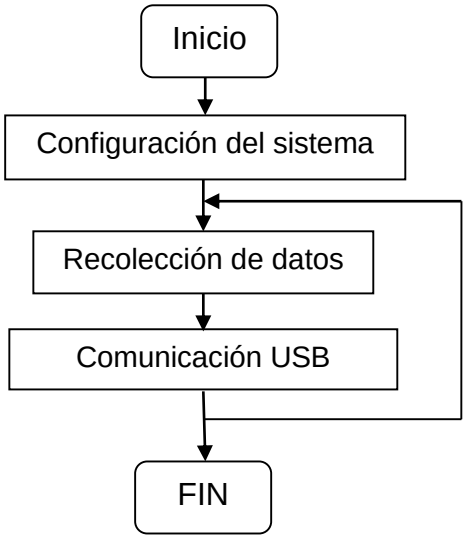
Microchip proporciona en su página web drivers y archivos necesarios para establecer la comunicación por puerto USB con la familia del microcontrolador mencionado. A través de ellos se puede efectuar transacciones de hasta 64 bytes por paquete cada milisegundo, por cada túnel de comunicación abierto. El PIC es programado mediante “PCWH Compiler de CCS”. Los descriptores utilizados para la comunicación USB están basados en los archivos que contiene el propio compilador. (Pool, 2010)

3.2.3. DIAGRAMA DE FLUJO PARA EL PROGRAMA DEL MICROCONTROLADOR

El funcionamiento general del microcontrolador para su configuración como una DAQ se presenta en el siguiente diagrama, indicando los pasos a seguir para su

desenvolvimiento para el fin indicado. El código fuente, se encuentra en el Anexo 3.

Diagrama 1. Diagrama general de la DAQ

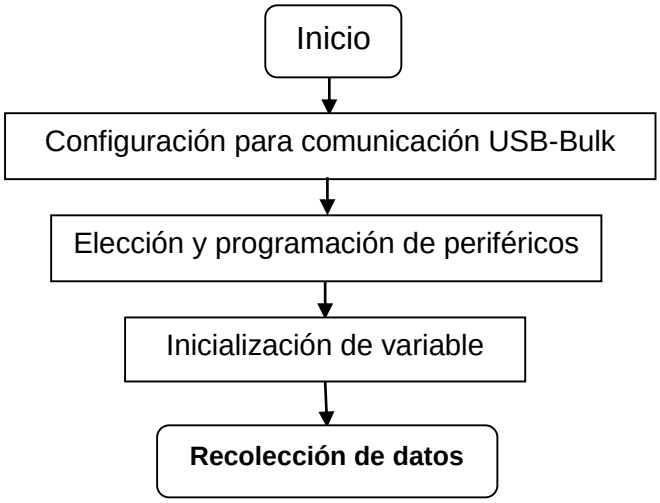


Fuente: Autor

3.2.3.1. CONFIGURACIÓN DEL SISTEMA

Es el primer paso para el desarrollo de cualquier algoritmo ya que se describe el comportamiento general del sistema, su importancia se hace notar en la elección o requerimientos iniciales que ha de tener la DAQ para un funcionamiento correcto a lo largo de su utilización. A continuación se presenta el diagrama de configuración inicial del sistema a construir.

Diagrama 2. Diagrama de configuración de la DAQ



Fuente: Autor

Es importante mencionar que el proceso de elección de periféricos se realizó en base a requerimientos de funcionamiento de la DAQ, dando a conocer la siguiente distribución de periféricos:

Tabla 8. Distribución de periféricos

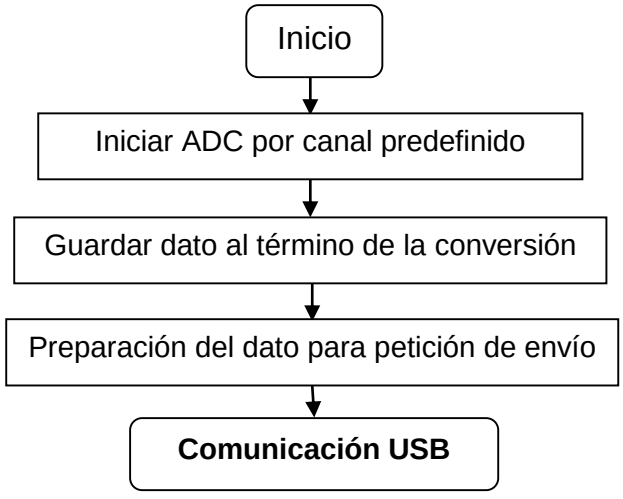
Función	Pines (etiquetados)
Entradas analógicas	A0, A1, A2, A3
Salidas PWM	A4, A5, C6, C7
Salidas digitales	B0, B1, B2, B3
Entradas digitales	B4, B5, B6, B7

Fuente: Autor

3.2.3.2. RECOLECCIÓN DE DATOS

El proceso de recolección de datos se lo realiza de manera que el sistema pueda leer una sola entrada analógica a la vez, con un rango de valores de lectura desde 0V hasta 5V, configurando al sistema para poder representar a los datos obtenidos con una resolución de 8 bits.

Diagrama 3. Diagrama de recolección de datos

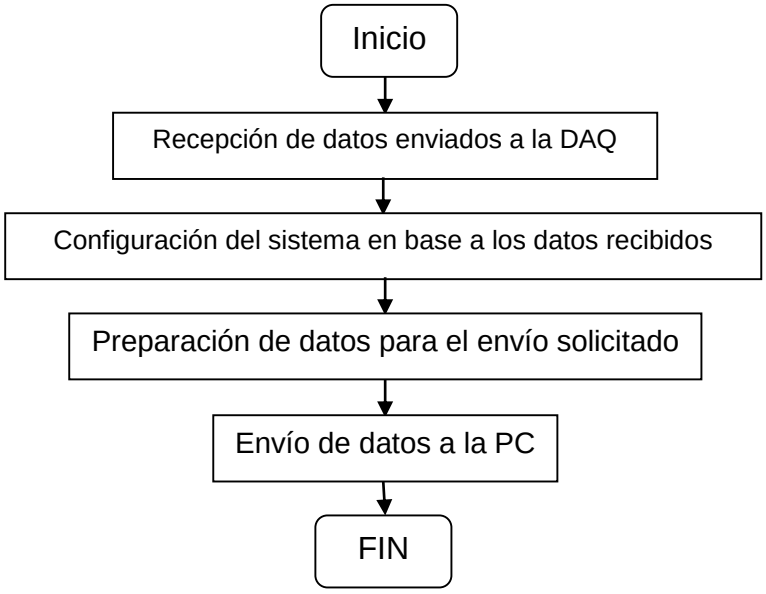


Fuente: Autor

3.2.3.3. COMUNICACIÓN USB

Esta parte es fundamental para el desenvolvimiento global del sistema como una DAQ, ya que permite configurar por medio de una interfaz gráfica las funciones implementadas a la tarjeta para que realice las peticiones encomendadas por el usuario. A continuación se presenta un diagrama que describe su funcionamiento.

Diagrama 4. Diagrama de interfaz DAQ-PC

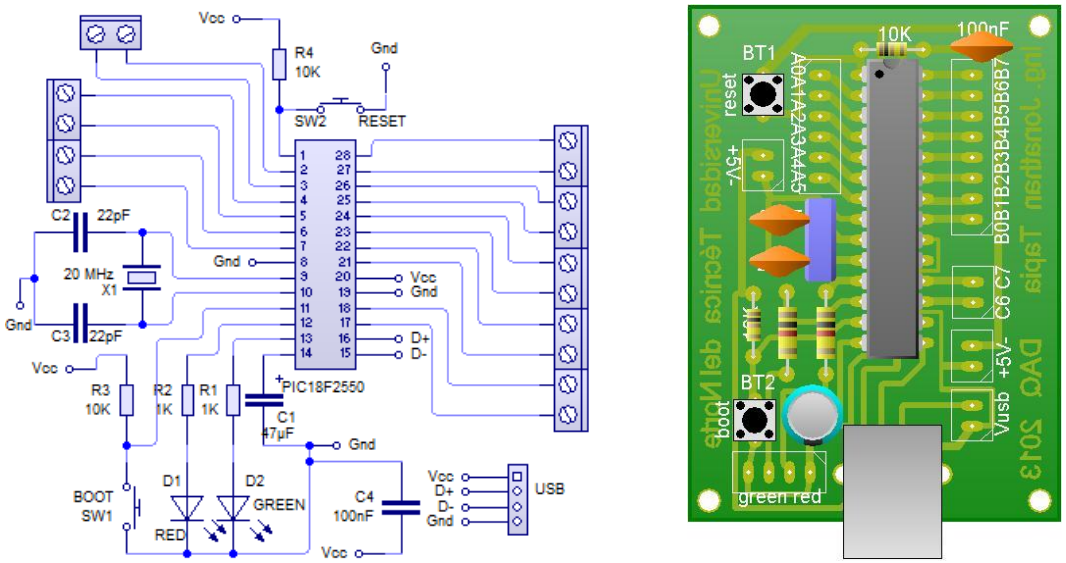


Fuente: Autor

3.2.4. MÓDULO DE LA TARJETA DE ADQUISICIÓN DE DATOS

Ya definido el funcionamiento y el algoritmo que van a ser implementados en la DAQ, se procede al diseño del hardware que va a interactuar con la programación del microcontrolador, teniendo en cuenta los pines de comunicación USB y sus requerimientos para el diseño físico final.

Figura 27. Circuito de la DAQ

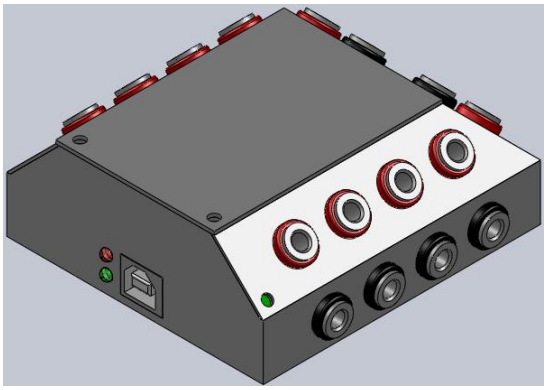


a) Diagrama de conexiones b) Circuito PCB

Fuente: Autor

Es importante mencionar que para su utilización de manera más didáctica, se ve necesaria la implementación de componentes que permitan el fácil manejo del circuito realizado, en la siguiente figura se muestra la DAQ finalizada.

Figura 28. Módulo de la DAQ



Fuente: Autor

CAPÍTULO IV

DISEÑO DE LA INTERFAZ GRÁFICA

4.1. INTRODUCCIÓN

Este capítulo se enfoca al desarrollo de una interfaz gráfica para la interacción con la DAQ construida, dividiéndola en partes fundamentales para su correcto funcionamiento y fácil entendimiento para el fin perseguido del proyecto presente, el cual es el modelamiento de sistemas lineales, es decir, modelar un sistema por métodos prácticos y experimentales dando a conocer al usuario las diferentes posibilidades al momento de la elección de una función de transferencia adecuada que va a caracterizar el funcionamiento del sistema el cual se quiere modelar. En el anexo 1 se proporciona el manual de usuario del software desarrollado.

4.2. CARACTERÍSTICAS DE LA INTERFAZ GRÁFICA

Antes de explicar el proceso de funcionamiento de la interfaz gráfica desarrollada, se debe identificar las partes y características fundamentales que debe tener el software, las cuales se describen a continuación:

1. Capacidad de comunicación USB
2. Manejo de datos digitales
3. Procesamiento de datos
4. Capacidad de modelamiento
5. Facilidad de manejo

La mayoría de programas que permiten la creación o desarrollo de alguna interfaz gráfica cuentan con funciones que permiten la utilización de varias comunicaciones, ya sea con el fin de la interacción con el medio exterior o comunicación con otros programas; estos también permiten el fácil manejo para el desarrollo de la interfaz, pero cuando se requiere procesar matemáticamente los datos obtenidos ya sea

graficándolos, aplicando operaciones estadísticas, procesamiento a través de filtros digitales, se vuelven literalmente inadecuados.

MATLAB resuelve el problema, pero existe un detalle, el hecho de importar los datos en tiempo real con alguna tarjeta de adquisición de datos, requiere compatibilidad con este programa sin contar adicionalmente con los recursos económicos para la compra de la tarjeta. Afortunadamente este software ha evolucionado al grado de que se puede conectar casi cualquier dispositivo, y su manejo es relativamente fácil e incluye una ayuda en línea. (Pool, 2010)

4.3. DESARROLLO DE LA INTERFAZ GRÁFICA

Hay que tomar en cuenta que para el desarrollo de esta aplicación, se ha visto necesario dividirlo en varias etapas que simplifiquen la creación de la aplicación tomando en cuenta los pasos fundamentales para su programación y entendimiento al momento de que algún usuario utilice la aplicación desarrollada.

4.3.1. COMUNICACIÓN CON LA DAQ

El primer paso fundamental en el desarrollo de la aplicación, es la definición de funciones, procesos y variables que ayuden en la comunicación del software desarrollado con la DAQ. El tipo de comunicación USB que se ha seleccionado es la transferencia de datos BULK. Como ya se ha mencionado microchip proporciona en su página web librerías para facilitar el desarrollo de dicha comunicación, en las cuales podemos encontrar una librería dinámica o DLL para la implementación de la transferencia de datos mencionada.

Dentro de los protocolos hay que especificar el tipo de transferencia de datos a usar, VID&PID, nombre y serie del producto que se conecta para que el host identifique al driver y pueda instalarlo con el fin de que el dispositivo pueda formar las “pipes” o túneles para su comunicación con el host. (Pool, 2010)

Es importante destacar que el uso y correcto funcionamiento de la librería dinámica se lo realiza en computadoras con sistema operativo Windows XP o Windows 7 de

32 bits ya que MATLAB proporciona un compilador propio para lenguaje C que solo puede ser utilizado en este tipo de sistemas operativos, el cual permite la interacción de la aplicación por medio de la comunicación USB-BULK sin ningún problema. Para elegir el compilador el usuario debe ejecutar la siguiente instrucción "*>> mex –setup*" y elegir el compilador con el nombre Lcc-Win32 C, solamente la primera vez que se utilice la aplicación.

4.3.1.1. DESCRIPCIÓN DE FUNCIONES DE MPUSBAPI.DLL DE MICROCHIP

Para una mayor facilidad de desarrollo de aplicaciones basados en el bus USB, microchip ha creado un archivo DLL en el que proporciona las funciones de acceso al puerto USB con un microcontrolador de la familia PIC18Fxx5x. Para un funcionamiento correcto, se necesita el driver mchpusb.sys. (Pool, 2010)

Las funciones de acceso al puerto USB utilizadas son las siguientes:

A. Verificación de conexión de dispositivo

```
MPUSBGetDeviceCount(PCHAR pVID_PID);
```

B. Abrir túneles de comunicación

```
MPUSBOpen(DWORD instance,    // Input
          PCHAR pVID_PID,    // Input
          PCHAR pEP,         // Input
          DWORD dwDir,       // Input
          DWORD dwReserved); // Input <Future Use>
```

C. Lectura de datos enviados del microcontrolador

```
MPUSBRead(HANDLE handle,            // Input
          PVOID pData,              // Output
          DWORD dwLen,              // Input
          PDWORD pLength,           // Output
          DWORD dwMilliseconds);    // Input
```

D. Envío de datos al microcontrolador

```
MPUSBWrite(HANDLE handle,           // Input
           PVOID pData,             // Input
           DWORD dwLen,             // Input
           PDWORD pLength,          // Output
           DWORD dwMilliseconds);   // Input
```

E. Cierre de túneles de comunicación

```
MPUSBClose(HANDLE handle);
```

POOL (2010) describe los siguientes caracteres para mayor comprensión de las funciones presentadas anteriormente:

1. **pVID_PID**: Cadena de caracteres que contiene el PID y VID del dispositivo con el que se va a trabajar. El formato es “vid_xxxx&pid_yyyy”, en donde “xxxx” es el valor del VID y el “yyyy” es del PID, ambos expresados en formato hexadecimal.
2. **instance:** Es el número de dispositivo al cual se va a ser referencia para abrir los túneles de comunicación.

3. **pEP:** Cadena de caracteres con el número de endpoints que se va a abrir. El formato es “\\MCHP_EPz” o “\MCHP_EPz” dependiendo del lenguaje de programación en donde z es el número del endpoint en decimal.
4. **dwDir:** Especifica la dirección del endpoint, en el cual se abre un solo pipe a la vez teniendo que utilizar la función dos veces.
   a. MP_READ para la función MPUSBRead con dwDir=1 se abre el pipe para leer
   b. MP_WRITE para la función MPUSBWrite con dwDir=0 se abre el pipe para escribir.
5. **dwReserved:** No asignado, por el momento el valor por omisión es cero. (Pool, 2010)
6. **handel:** Identifica el pipe del endpoint con el cual se va a trabajar, siendo MP_READ para la función MPUSBRead y MP_WRITE para la función MPUSBWrite.
7. **pData:** Puntero al buffer que recibe o envía los datos al pipe.
8. **dwLen:** Especifica el número de byte que se espera recibir o enviar al pipe.
9. **pLenght:** Puntero que proporcione el número de bytes que se espera enviar o recibir.
10. **dwMilliseconds:** Especifica el intervalo de tiempo de ejecución en milisegundos. La función vuelve si transcurre el intervalo aunque no se complete la operación. Si este es cero la función comprueba los datos del pipe y vuelve inmediatamente y si es infinito el intervalo de tiempo de ejecución nunca termina.

4.3.1.2. UTILIZACIÓN DE LAS FUNCIONES EN MATLAB

Al momento de proceder con el enlace de comunicación entre el software y la DAQ, se debe haber instalado el paquete del driver de microchip que permita el reconocimiento del dispositivo conectado, entonces, para realizar la conexión con Matlab simplemente se copiaron las instrucciones al programa y se acondicionaron las variables para que la DLL pueda reconocerlo. (Pool, 2010)

POOL (2010) describe los siguientes pasos a realizarse para una correcta comunicación con la DAQ:

1. Copiar los archivos “_mpusbapi.c” y “mpusbapi.dll” en la carpeta en donde se realizará la aplicación GUI, estos archivos son proporcionados por microchip.
2. Cargar los archivos en Matlab utilizando la siguiente instrucción:

- loadlibrary mpusbapi _mpusbapi.h alias librería

3. Luego se identifica el número de dispositivos conectados con VID&PID con:

- [conectado] = calllib ('libreria', 'MPUSBGetDeviceCount', vid_pid_norm)
  Donde: *vid_pid_norm = libpointer('int8Ptr', [uint8('vid_04d8&pid_000b') 0]);*

4. Seguidamente se abre los pipes la comunicación tanto de lectura como envío de datos:
   a. [my_out_pipe] = calllib('libreria', 'MPUSBOpen',uint8(0),vid_pid_norm, out_pipe, uint8(0), uint8 (0));
   b. [my_in_pipe]= calllib('libreria','MPUSBOpen',uint8 (0), vid_pid_norm, in_pipe, uint8 (1), uint8 (0));
      Donde: *out_pipe = libpointer('int8Ptr',[uint8('\MCHP_EP1') 0]);* y *in_pipe = libpointer ('int8Ptr', [uint8('\MCHP_EP1') 0]);*
5. Se procede a la lectura de datos enviados solamente si las pipes se encuentran abiertas, con:

- [aa,bb,data_in,dd]=calllib('libreria', 'MPUSBRead',my_in_pipe, data_in,uint8(64), uint8(64), uint8(10));
  Donde: *data_in = eye(1,64,'uint8');*

6. Se procede al envío de datos enviados solamente si las pipes se encuentran abiertas, con:

- calllib('libreria','MPUSBWrite', my_out_pipe, data_out, uint8(64), uint8(64),uint8(10));
  Donde: *data_out = eye(1,64,'uint8');*

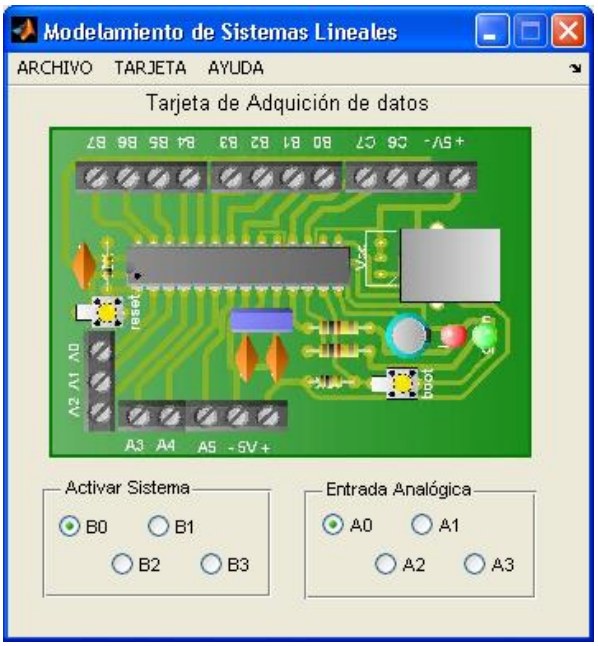
7. Se cierra las pipes abiertas al finalizar el programa, si no se implementa este paso Windows generará errores y se perdería la comunicación, se utiliza:
   a. calllib('libreria', 'MPUSBClose', my_in_pipe);
   b. calllib('libreria', 'MPUSBClose', my_out_pipe);

4.3.2. INICIALIZACIÓN DE LA APLICACIÓN

Al iniciar la aplicación nos aparecerá la ventana principal de la aplicación desarrollada permitiendo la definición de funciones y variables a utilizar, además de los requisitos básicos para la inicialización y preparación de la comunicación con la DAQ. El usuario puede realizar las siguientes acciones permitidas:

a. Definir pines a utilizar de la DAQ.
b. Conectar o desconectar la DAQ con el software.
c. Consulta de ayudas de usuario
d. Elección de proceso a realizarse:
    1. Adquisición de datos.
    2. Modelamiento de datos.

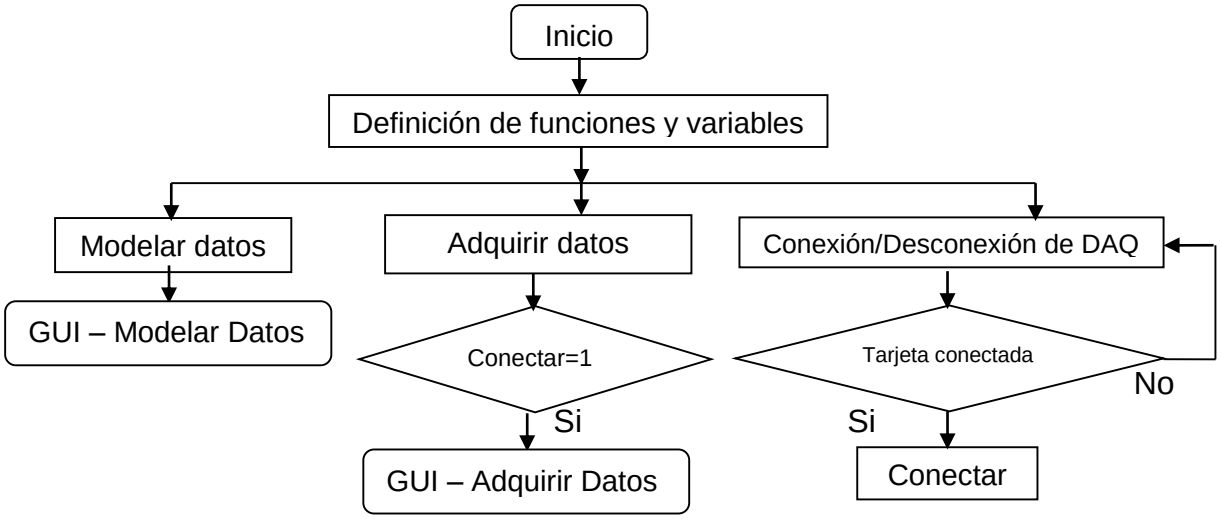
Figura 29. Ventana principal del software desarrollado



Fuente: Autor

A continuación se presenta un diagrama, el cual describe el funcionamiento de la aplicación presente. El código fuente se encuentra en el Anexo 4.

Diagrama 5. Diagrama de funcionamiento – Ventana principal



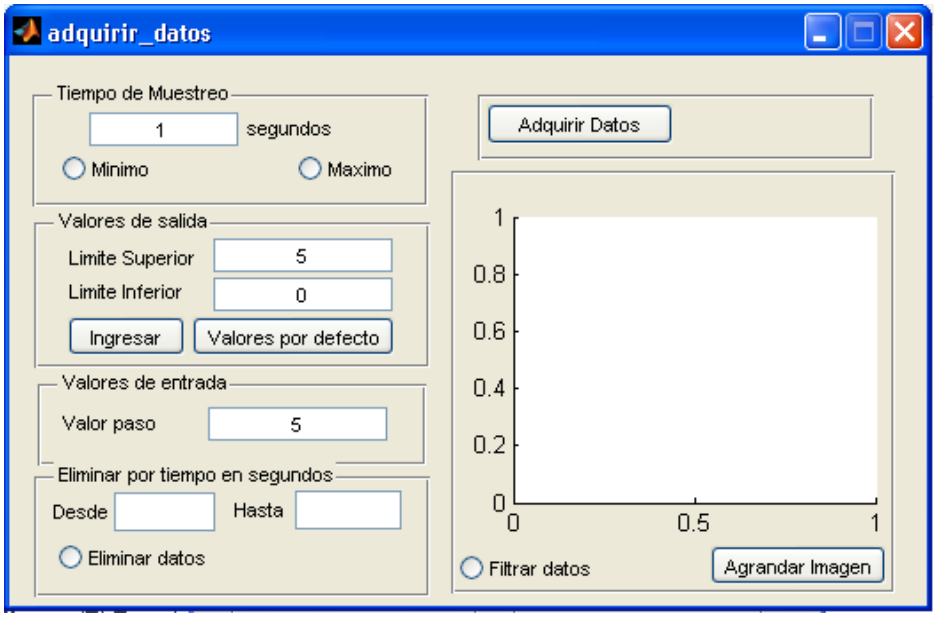
Fuente: Autor

4.3.3. ADQUISICIÓN DE LA SEÑAL

La aplicación presente permite al usuario la adquisición de datos provenientes del sistema con el cual vamos a trabajar para su modelado. Esta ventana permite las siguientes acciones a realizarse por el usuario:

a. Definición del tiempo de muestreo, configurándose internamente el periodo de adquisición de datos.
b. Iniciar adquisición de datos.
c. Definición de límites superior e inferior del sistema.
d. Definición del valor de entrada al sistema.
e. Eliminación de datos en un rango de tiempo.
f. Filtrado de datos para eliminación de ruido.
g. Guardar datos obtenidos.

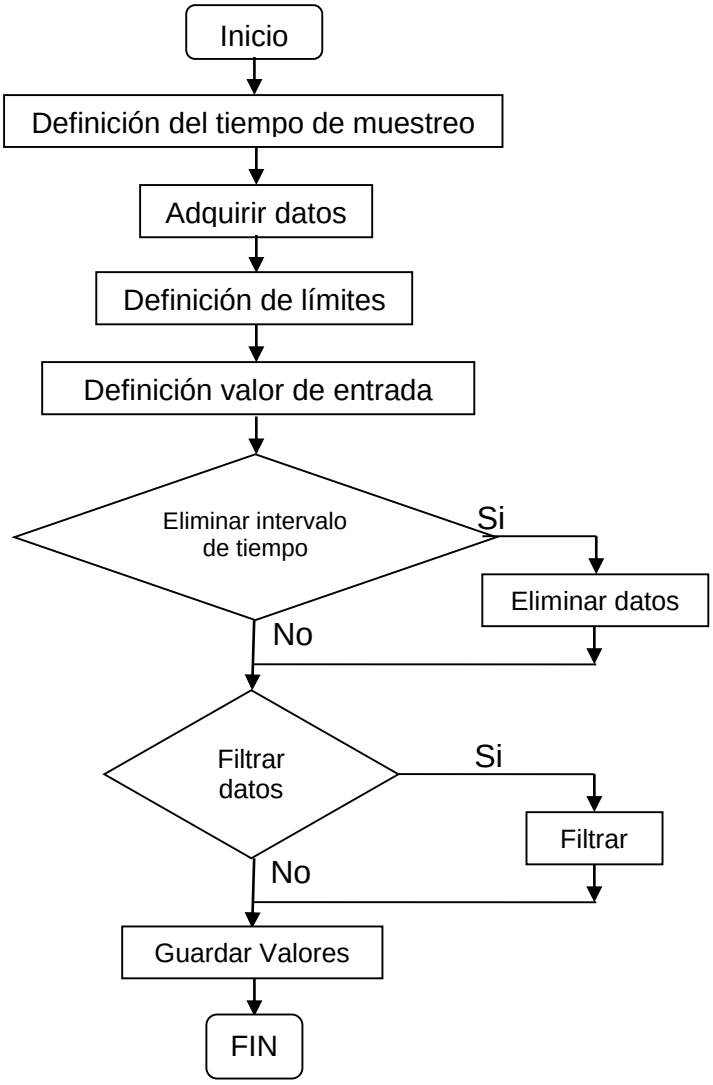
Figura 30. Ventana de adquisición de datos



Fuente: Autor

A continuación se presenta un diagrama, el cual describe el funcionamiento de la aplicación presente. El código fuente se encuentra en el Anexo 4.

Diagrama 6. Diagrama de funcionamiento – Adquisición de datos

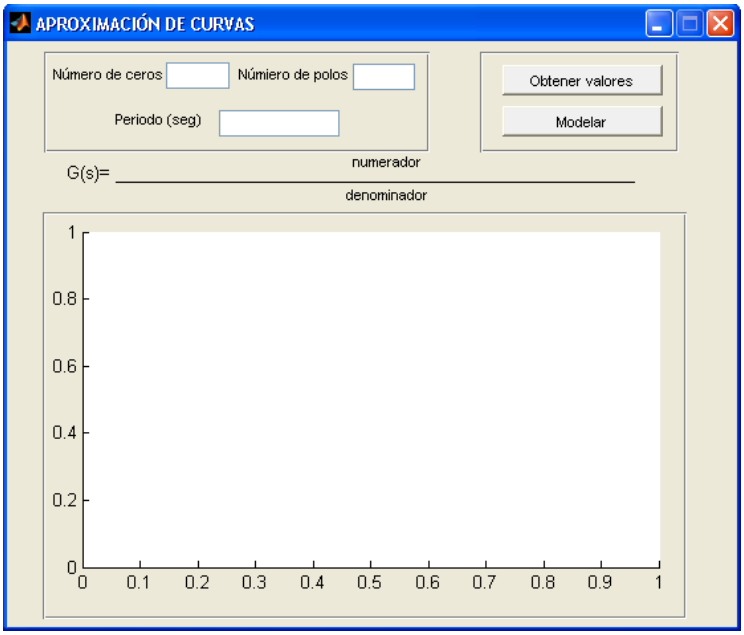


Fuente: Autor

4.3.4. MODELAMIENTO DE LA SEÑAL

Esta aplicación permite el ajuste de los datos adquiridos a un tipo de señal, entregando dicha función en relación a la transformada de Laplace definiéndola como función de transferencia, permitiendo al usuario la decisión del grado del polinomio tanto en el numerador como es denominador, es decir, el usuario decide cuantos ceros y polos tendrá la función característica del sistema que se está modelando; además permite la alteración del periodo de adquisición para la verificación de que este es un parámetro importante al momento del modelado de datos, demostrando que a periodos diferentes y manejando los mismos datos se tienen distintos tipos de funciones de transferencia.

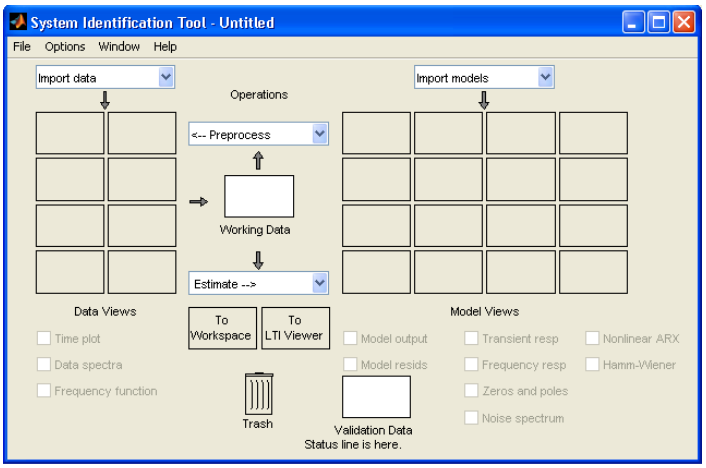
Figura 31. Ventana de identificación de curvas



Fuente: Autor

Además del modelado que ofrece la aplicación desarrollada, esta permite la utilización de la interfaz propia de Matlab para la identificación de curvas, para lo cual es necesario escribir en Command Window la siguiente línea de código “>> global x y T”.

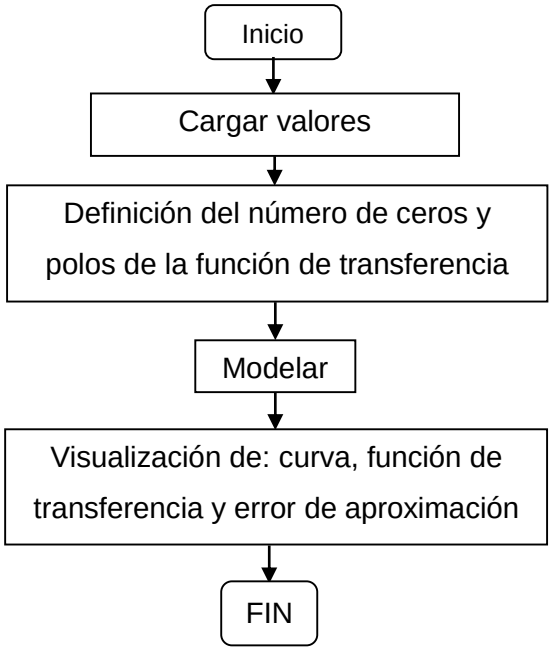
Figura 32. Ventana de aproximación de curvas propia de Matlab



Fuente: System Identification Toolbox. User's Guide. 2013

A continuación se presenta un diagrama, el cual describe el funcionamiento de la aplicación presente. El código fuente se encuentra en el Anexo 4.

Diagrama 7. Diagrama de funcionamiento – Modelado de datos



Fuente: Autor

CAPÍTULO V

PRÁCTICAS DE LABORATORIO - IMPLEMENTACIÓN DE MÓDULOS DIDÁCTICOS

5.1. INTRODUCCIÓN

En este capítulo se describen algunas prácticas de ejemplo para las personas que van a utilizar el proyecto aquí descrito, iniciando con una breve introducción a las utilidades que ofrece el módulo, hasta la realización de varias prácticas para el modelamiento y comprobación de la función obtenida a través de la aplicación de ecuaciones diferenciales para obtener la función de transferencia teórica. En el Anexo 2 se proporciona una guía de prácticas para el refuerzo del estudiante en la realización de las mismas.

También se procederá a explicar el proceso de diseño e implementación de cada módulo que se va a utilizar para la realización de cada práctica de adquisición y modelamiento, describiendo las características del módulo a implementar y acondicionamiento del sensor para que entregue la señal que representa la acción del sistema frente a un tipo de entrada.

5.2. TIPO SEÑAL DE ENTRADA Y SEÑAL DE SALIDA

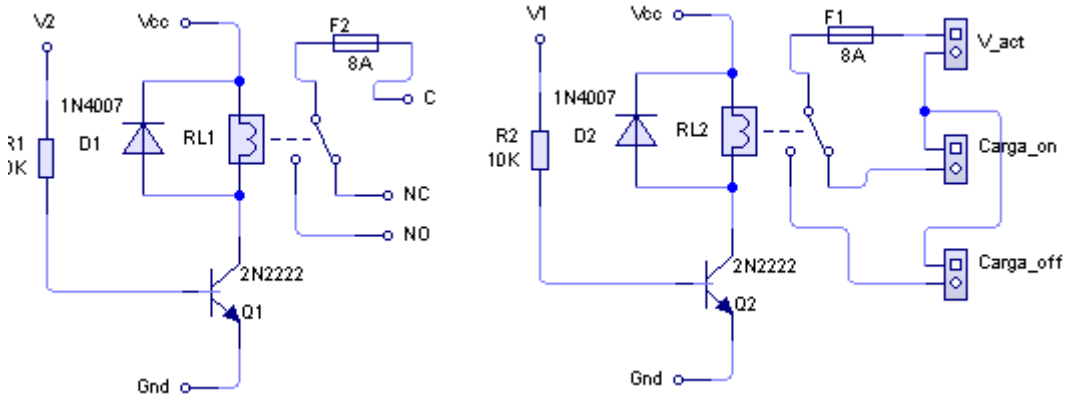
La tarjeta de adquisición de datos proporciona un voltaje de 5V por la razón de que se obtiene del puerto USB del computador, con la aclaración de que solamente debe ser utilizado en bajos consumos de energía para salidas de corriente inferiores a 500mA que es la máxima corriente que ofrece el puerto, como por ejemplo el acondicionamiento del sensor.

La señal del sensor debe ser acondicionada para que entregue una respuesta en voltaje entre valores desde 0V hasta 5V, debido a que la DAQ permite la lectura de voltaje en este rango de valores.

En la mayoría de prácticas no solamente vamos a necesitar el acondicionamiento del sensor, sino también mayor cantidad de energía para la activación de sistemas para que entreguen una señal adecuada en su modelamiento, por esta razón se ha visto en la necesidad de la implementación de un módulo que permita utilizar mayor energía para este tipo de sistemas.

En algunos módulos presentados solamente necesitamos su activación por medio de un switch, el cual debe ser controlado para la adquisición oportuna de los datos; dicho circuito se presenta en la figura 33a. En los módulos restantes se necesita su activación por medio del ingreso de alguna fuente de voltaje externa dependiendo de los requerimientos de los módulos a utilizar, presentado el circuito en la figura 33b. Ambos circuitos permiten la toma de datos desde el momento en que el sistema se encuentra desactivado o activado.

Figura 33. Circuito de interfaz de potencia



a) Activación por switch

a) Ingreso de voltaje externo

Fuente: Autor

En donde:

a. Vcc conexión a voltaje positivo, es igual a 5V.
b. Gnd conexión a voltaje negativo, es igual a 0V.
c. V1 y V2 es la salida activadora de la DAQ.
d. NC-C inicialmente se encuentran conectados, y tras la activación se desconectan.

e. NO-C inicialmente se encuentran desconectados, y tras la activación se conectan.
f. V_act es el voltaje de activación externo.
g. Carga_on permite que el sistema a modelas comience encendido.
h. Carga_off permite que el sistema a modelas comience apagado.

Es importante mencionar que para su utilización de manera más didáctica, se ve necesaria la implementación de componentes que permitan el fácil manejo del circuito realizado, en la siguiente figura se muestra la interfaz de potencia finalizada.

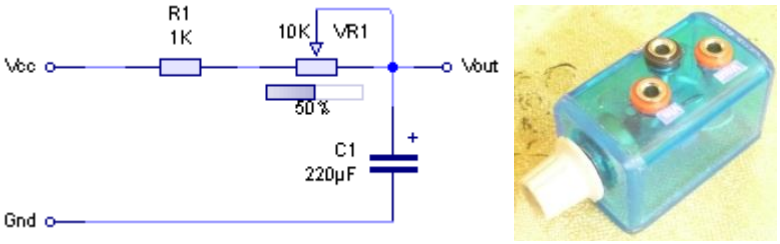
Figura 34. Módulo de interfaz de potencia

Fuente: Autor

5.3. PRÁCTICA 1: ADQUISICIÓN Y VISUALIZACIÓN DE DATOS UTILIZANDO UN POTENCIÓMETRO, CIRCUITO RC

El primer módulo es un circuito RC, aunque es el sistema más sencillo de construir y modelar nos da una breve introducción a la utilización del presente proyecto ya que no solamente es una resistencia y capacitor de valor fijo, sino también está adicionado un potenciómetro que permite cambiar el tiempo de establecimiento y por ende la función de transferencia.

Figura 35. Circuito RC



Fuente: Autor

5.3.1. MODELAMIENTO TEÓRICO

Para su modelamiento tenemos la misma intensidad de corriente para R1, VR1 y C1, siendo la siguiente ecuación diferencial que caracteriza al sistema de la figura anterior:

Ecuación 23. Ecuación diferencial del módulo RC

$$\frac{dVout}{dt} + \frac{Vout}{(R1 + VR1)C1} = \frac{Vin}{(R1 + VR1)C1}$$

Fuente: Autor

Aplicando la transforma de Laplace y suponiendo condiciones iniciales igual a cero, tenemos la función de transferencia representada en la ecuación 24a.

Ecuación 24. Función de transferencia del módulo RC

$$G_s = \frac{Vout_s}{Vin_s} = \frac{\frac{1}{(R1+VR1)C1}}{s + \frac{1}{(R1+VR1)C1}}$$

a) Función de transferencia general

$$G_s = \frac{4.546}{s + 4.545}$$

b) Tiempo de establecimiento mínimo

$$G_s = \frac{0.413}{s + 0.413}$$

c) Tiempo de establecimiento máximo

Fuente: Autor

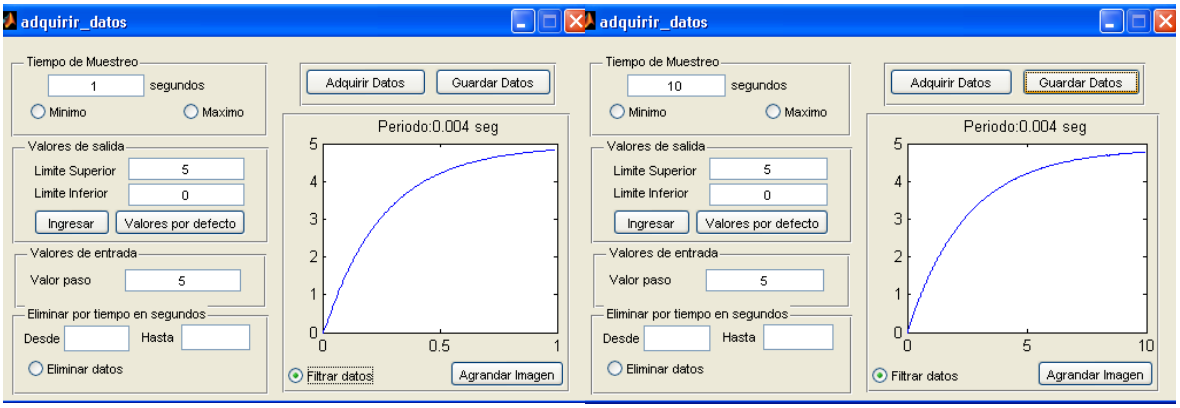
Teniendo una constante de tiempo $\tau = (R1 + VR1)C1$, y suponiéndose que los valores de los componentes son exactos se obtiene:

a. Tiempo de establecimiento mínimo con VR1 igual a 0Ω de $ts = 0{,}88\ segundos$ de la ecuación 24b.
b. Tiempo de establecimiento máximo con VR1 igual a 10kΩ de $ts = 9{,}68\ segundos$ de la ecuación 24c.

5.3.2. MODELAMIENTO PRÁCTICO

En base a la ecuación 24a vemos que está compuesta con una cantidad nula de ceros y con un polo, entonces, de la ecuación 24b podemos utilizar un tiempo de adquisición de 1 segundo y de la ecuación 24c utilizaremos un tiempo de adquisición de 10 segundos obteniendo los siguiente resultados:

Figura 36. Adquisición de datos para el circuito RC

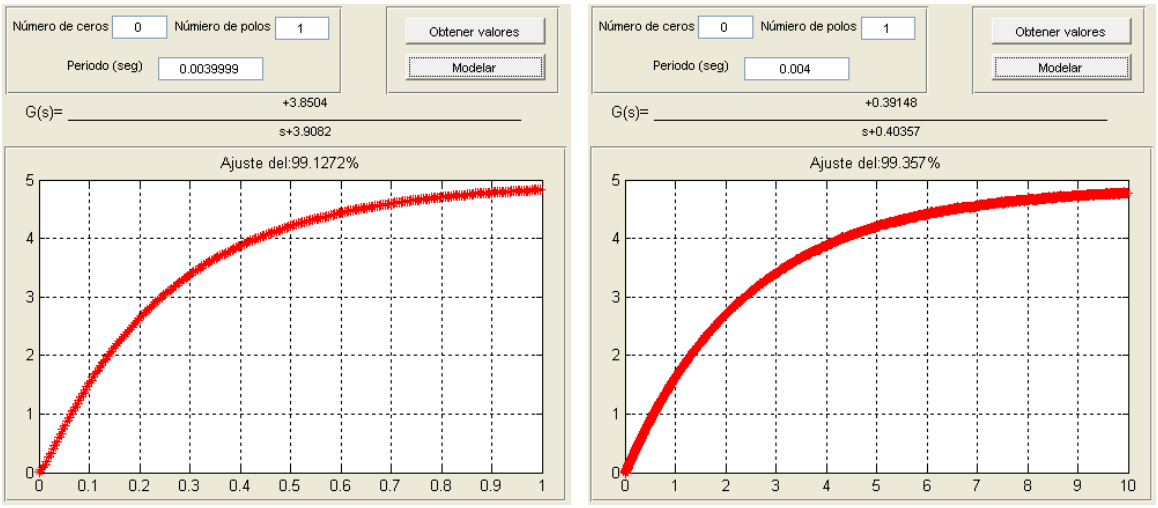


a) Tiempo de adquisición de 1 segundo
b) Tiempo de adquisición de 10 segundos

Fuente: Autor.

Aplicando el modelamiento de los sistemas y cuyo periodo de muestreo se carga automáticamente tenemos:

Figura 37. Modelamiento del circuito RC



a) Tiempo de adquisición de 1 segundo

b) Tiempo de adquisición de 10 segundos

Fuente: Autor

En la figura 37b podemos observar una ecuación de transferencia con un porcentaje de aproximación de 99.36%. En la figura 37a podemos observar una ecuación de transferencia con un porcentaje de aproximación de 99.13%.

Ecuación 25. Función de transferencia práctica del circuito RC

$$G_s = \frac{3.85}{s + 3.908}$$

a) Tiempo de establecimiento mínimo

$$G_s = \frac{0.391}{s + 0.403}$$

b) Tiempo de establecimiento máximo

Fuente: Autor

La principal característica por el cual las funciones de transferencia teóricas y prácticas no concuerdan son:

1. Debido a la tolerancia de valores de los componentes de resistencias y capacitor, la cual indica que los valores especificados en estos componentes son aproximadamente el valor real, debido a mediciones, se obtuvieron valores

de R1=1.04 kΩ, R1+VR1=10.55 kΩ, C1=228.98 μF, las fotografías se presentan en el anexo 5. Resultando las siguientes ecuaciones:

Ecuación 26. Función de transferencia con valores medidos

$$G_s = \frac{4.199}{s + 4.199}$$

a) Tiempo de establecimiento mínimo

$$G_s = \frac{0.414}{s + 0.414}$$

b) Tiempo de establecimiento máximo

Fuente: Autor

2. El porcentaje de ajuste de la curva el cual es inferior al 100% debido al ruido acoplado del ambiente exterior.

5.4. PRÁCTICA 2: ADQUISICIÓN Y MODELADO - PLANTA DE TEMPERATURA

El segundo módulo se trata de obtener la función de transferencia del calentamiento que se produce en el foco incandescente cuando este se prende, obteniéndose una función de temperatura con respecto al tiempo.

Figura 38. Módulo de temperatura

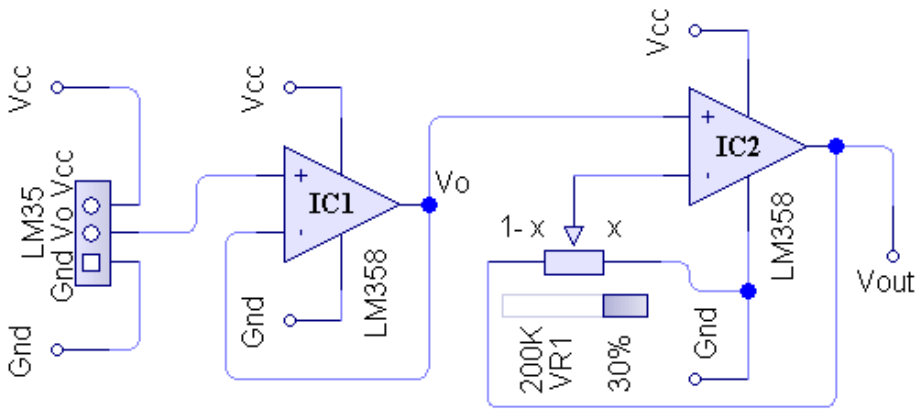
Fuente: Autor

El sensor elegido para esté módulo es el LM35, el cual tiene las siguientes características:

a. Está calibrado en grados centígrados.
b. Salida lineal con un valor en escala de +10 mV/ºC.
c. Rango de funcionamiento de -55 ºC hasta +150 ºC.
d. Adecuado para aplicaciones remostas.
e. Bajo costo.
f. Opera desde 4 a 30 voltios.
g. Corriente de drenaje inferior a 60 µA.
h. Impedancia de salida de 0.1 W para 1 mA.

Debido a lo anterior se decidió utilizar la fuente de 5V proporcionada por la DAQ y escogiendo un rango de operación desde 0 ºC hasta +150 ºC, con un rango de voltaje a la salida del sensor de 0 a 1.50 V; para ello se acondicionó la salida del sensor para que me entregue una un voltaje de 0 a 5 V respectivamente, obteniéndose el siguiente circuito:

Figura 39. Circuito acondicionador para sensor LM35



Fuente: Autor

Debido a que IC2 de la figura 39 es un amplificador no inversor, tenemos la siguiente ecuación, denotando que para Vo=1.50V el valor de salida debe ser Vout=5V.

Ecuación 27. Acondicionamiento del sensor LM35

$$\frac{5}{1.5} = \left(1 + \frac{1-x}{x}\right); \quad x = 30\%\, VR1$$

Fuente: autor

Definiendo a “x” como el porcentaje del valor total de VR1=200 kΩ, obteniendo un valor de resistencia en x de 60 kΩ.

5.4.1. MODELAMIENTO TEÓRICO

Para su modelamiento tenemos la siguiente función de transferencia:

Ecuación 28. Función de transferencia del módulo de temperatura

$$G_s = \frac{T_s}{q_s} = \frac{\frac{1}{C_t}}{s + (\frac{1}{C_t R_t} + \frac{QS}{C_t})}$$

Fuente: Sistemas de control moderno, 2005

Donde:

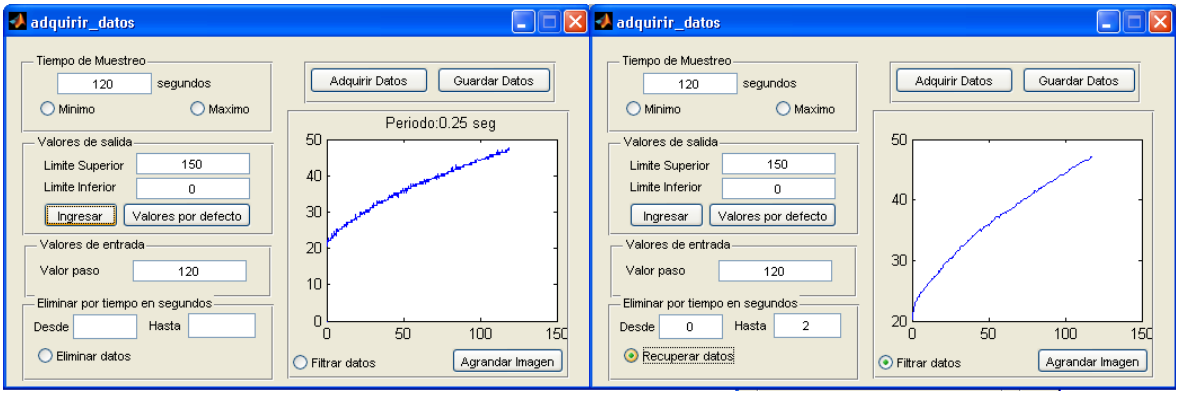
a. T es la diferencia de temperatura debido al proceso térmico.
b. C_t es la capacidad térmica, es una constante.
c. Q es el caudal del fluido a calentarse, es una constante.
d. S es el calor específico del fluido a calentarse, es una constante.
e. R_t es la resistencia térmica del aislamiento, es una constante.
f. q es el caudal de calor del elemento calentado

5.4.2. MODELAMIENTO PRÁCTICO

En base a la ecuación 28 vemos que está compuesta con una cantidad nula de ceros y con un polo, entonces, al modelar los datos obtenemos el siguiente

resultado con un tiempo de adquisición de 120 segundos, con límite superior de 150 ºC e inferior de 0 ºC y una entrada de 120 V_{AC} definida por la corriente eléctrica:

Figura 40. Adquisición de datos para el módulo de temperatura

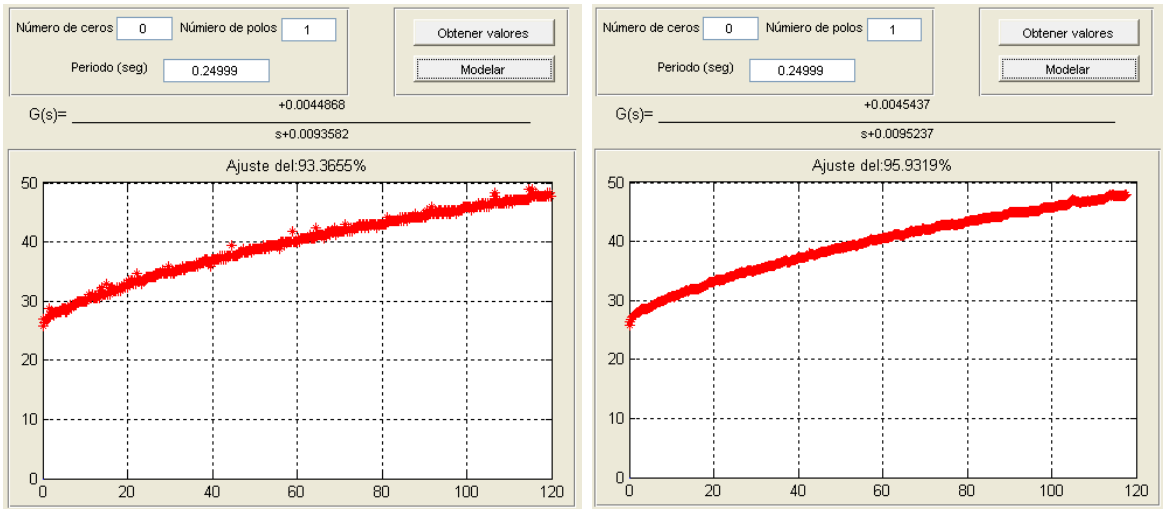


a) Señal con original b) Señal filtrada

Fuente: Autor.

En la figura 40b en la cual vemos la señal filtrada, hemos eliminado 2 segundos iniciales de la señal, debido a que en el procedo de filtrado los valores iniciales se encuentran por debajo del valor inicial de la señal representada en la figura 40a; aplicando el modelamiento del sistema y cuyo periodo de muestreo se carga automáticamente tenemos:

Figura 41. Modelamiento del módulo de temperatura



a) Función de trasferencia de la señal original b) Función de trasferencia de la señal filtrada

Fuente: Autor

En la figura 41a podemos observar la función de transferencia con un porcentaje de aproximación del 93.36% y en la figura 41b la función de transferencia tiene un porcentaje de aproximación del 95.93%. Con esto se demuestra que debido al filtrado se mejora el porcentaje de aproximación.

Ecuación 29. Función de transferencia práctica del módulo de temperatura

$$G_s = \frac{0.0044868}{s + 0.0093582} \qquad G_s = \frac{0.0045437}{s + 0.0095237}$$

a) Señal original b) Señal filtrada

Fuente: Autor

5.5. PRÁCTICA 3: ADQUISICIÓN Y MODELADO - PLANTA DE VELOCIDAD ANGULAR

El tercer módulo se trata de obtener la función de transferencia de la velocidad de un motor de corriente continua cuando este se enciende, obteniéndose una función de velocidad angular con respecto al tiempo.

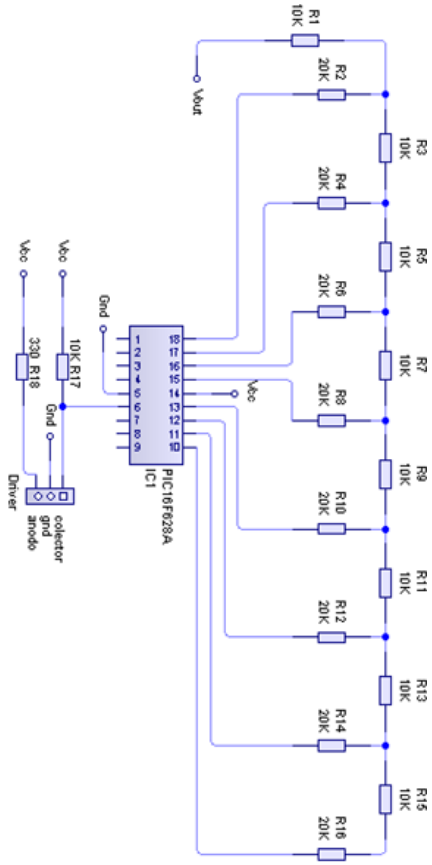
Figura 42. Módulo de velocidad angular

Fuente: Autor

El sensor elegido para este módulo es un encoder, el cual entrega una señal digital en frecuencia, figura 43a; como la DAQ acepta valores analógicos de voltaje, necesitamos un circuito acondicionador que permita pasar de frecuencia a voltaje, figura 43b.

Figura 43. Sensor de velocidad angular

a) Encoder-driver

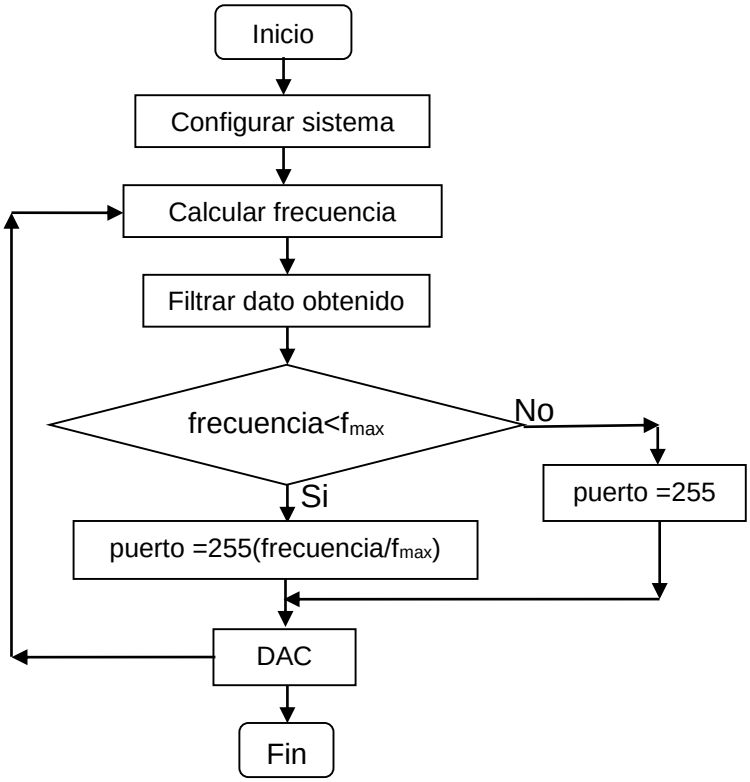


b) Circuito acondicionador

Fuente: Autor

Como se puede observar el circuito de la figura 43b está realizado con un microcontrolador, el cual permite leer la señal de frecuencia del encoder pasándola por un algoritmo que realiza la conversión de frecuencia a voltaje ayudado por un conversor digital-analógico DAC. A continuación se presenta un diagrama, el cual describe el funcionamiento del circuito cuyo código fuente se encuentra en el Anexo 6.

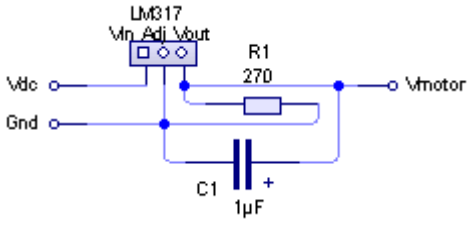
Diagrama 8. Diagrama de funcionamiento del conversor frecuencia voltaje



Fuente: Autor

También se ha implementado una fuente de 1.25V, utilizando el regulador variable LM317, el circuito implementado se puede ver en la siguiente figura.

Figura 44. Regulador de voltaje de 1.25V



Fuente: Autor

5.5.1. MODELAMIENTO TEÓRICO

Para su modelamiento tenemos la siguiente función de transferencia:

Ecuación 30. Función de transferencia del motor dc

$$G_s = \frac{\omega_s}{V_s} = \frac{\frac{K_m}{J \times L_f}}{s^2 + \left(\frac{R_f}{L_f} + \frac{b}{J}\right)s + \frac{R_f}{L_f} \times \frac{b}{J}}$$

Fuente: Sistemas de control moderno, 2005

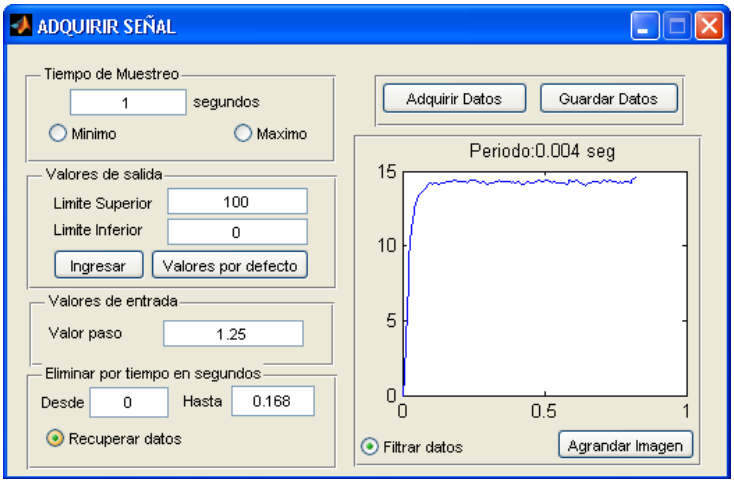
Donde:

a. K_m, constante característica del motor dc
b. b, valor constante de fricción del motor dc
c. J, valor constante de inercia del motor dc
d. R_f, resistencia interna del motor dc
e. L_f, valor de inductancia de la bobina del motor dc
f. ω, velocidad angular
g. V, voltaje aplicado al motor dc

5.5.2. MODELAMIENTO PRÁCTICO

En base a la ecuación 30 vemos que está compuesta con una cantidad nula de ceros y con dos polos, entonces, al modelar los datos obtenemos el siguiente resultado con un tiempo de adquisición de 1 segundo debido a su tiempo de establecimiento que es inferior al tiempo mínimo de adquisición del proyecto descrito, con límite superior de 100 e inferior de 0 vueltas por segundo; los límites son definidos debido al circuito conversor de frecuencia a voltaje, y una entrada de 1.25 V:

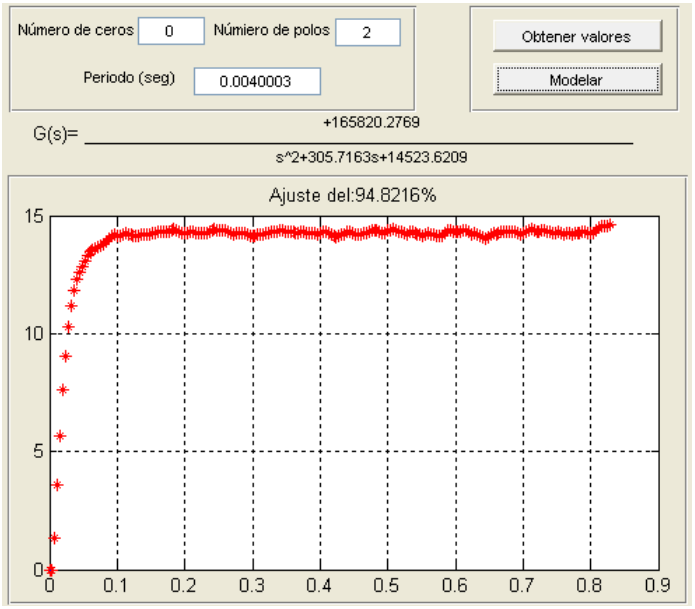
Figura 45. Adquisición de datos para el módulo de velocidad angular



Fuente: Autor.

En la figura 45 en la cual vemos la señal filtrada, hemos eliminado 0.168 segundos iniciales de la señal, debido a la demora de encendido en el módulo de potencia; aplicando el modelamiento del sistema y cuyo periodo de muestreo se carga automáticamente tenemos:

Figura 46. Modelamiento del módulo de velocidad angular



Fuente: Autor

En la figura 46 podemos observar la función de transferencia con un porcentaje de aproximación del 94.82%.

Ecuación 31. Función de transferencia práctica del módulo de velocidad angular

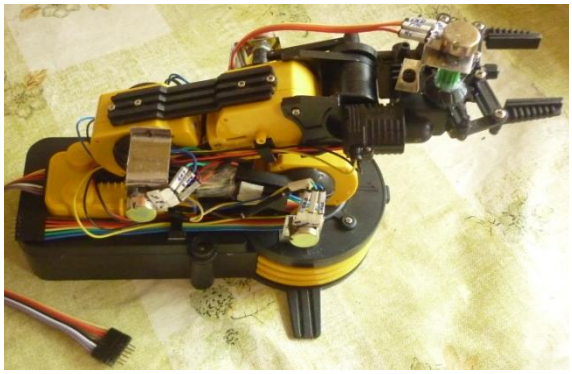
$$G_s = \frac{165820.277}{s^2 + 305.716s + 14523.62}$$

Fuente: Autor

5.6. PRÁCTICA 4: ADQUISICIÓN Y MODELADO - PLANTA DE POSICIONAMIENTO ANGULAR

El cuarto módulo se trata de obtener la función de transferencia del ángulo de giro de un motor de corriente continua cuando este se enciende, obteniéndose una función de posicionamiento angular con respecto al tiempo.

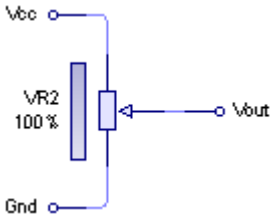
Figura 47. Módulo de posicionamiento angular



Fuente: Autor

El sensor elegido para este módulo es un potenciómetro lineal, este entrega una señal de resistencia en relación al ángulo de giro, para su acondicionamiento solo se necesita la polarización de voltaje en sus extremos teniendo una variación de voltaje con respecto al ángulo de giro; se a implementado cuatro potenciómetros para cuatro motores.

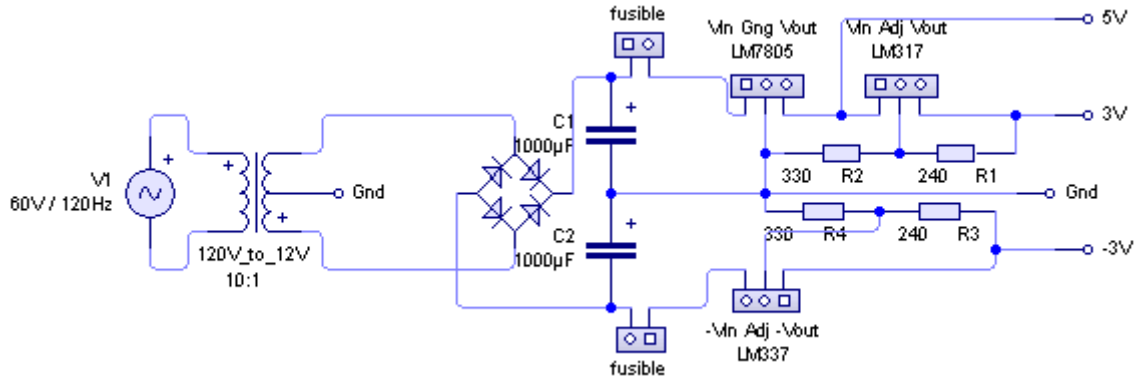
Figura 48. Acondicionamiento del potenciómetro lineal



Fuente: Autor

Como se ve en la figura 47, este módulo se implementó en un brazo robótico el cual necesita una alimentación entregada por cuatro pilas dando un voltaje de ±3V; para evitar el uso de las pilas debido a que si se acaba su energía tendríamos que conseguir más, se realizó una fuente de voltaje de ±3V y 5V, el circuito se muestra en la siguiente figura.

Figura 49. Fuente de voltaje



Fuente: Autor

5.6.1. MODELAMIENTO TEÓRICO

Para su modelamiento tenemos la siguiente función de transferencia:

Ecuación 32. Función de transferencia del motor dc en ángulo

$$G_s = \frac{\omega_s}{V_s} = \frac{\frac{K_m}{J \times L_f}}{s^3 + \left(\frac{R_f}{L_f} + \frac{b}{J}\right)s^2 + (\frac{R_f}{L_f} \times \frac{b}{J})s}$$

Fuente: Sistemas de control moderno, 2005

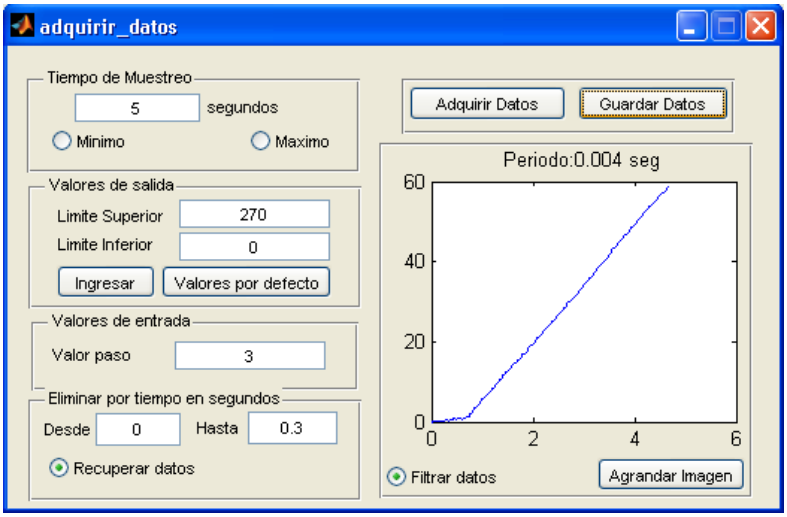
Donde:

a. K_m, constante característica del motor dc
b. b, valor constante de fricción del motor dc
c. J, valor constante de inercia del motor dc
d. R_f, resistencia interna del motor dc
e. L_f, valor de inductancia de la bobina del motor dc
f. ω, velocidad angular
g. V, voltaje aplicado al motor dc

5.6.2. MODELAMIENTO PRÁCTICO

En base a la ecuación 32 vemos que está compuesta con una cantidad nula de ceros y con tres polos, entonces, al modelar los datos obtenemos el siguiente resultado con un tiempo de adquisición de 5 segundos debido al rango de movimiento permitido por el motor a modelar seleccionado, con límite superior de 270º e inferior de 0º; los límites son definidos debido al rango de movimiento del potenciómetro lineal, y una entrada de 3V:

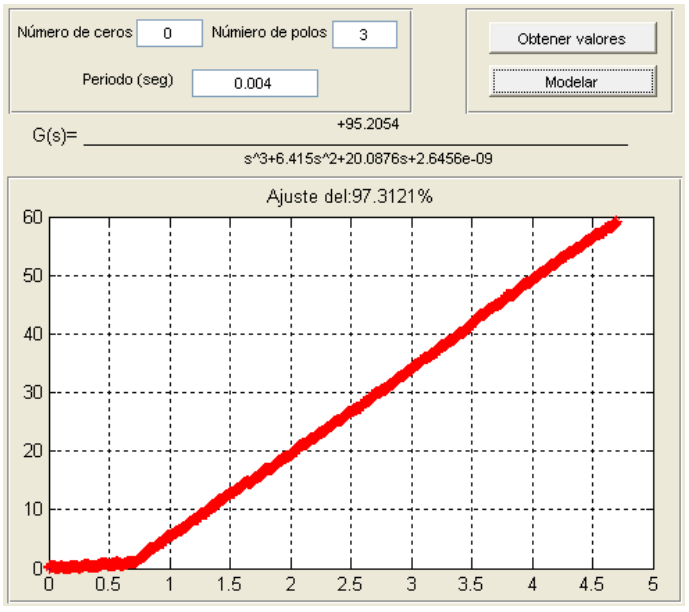
Figura 50. Adquisición de datos para el módulo de posicionamiento angular



Fuente: Autor.

En la figura 50 en la cual vemos la señal filtrada, hemos eliminado 0.3 segundos iniciales de la señal, debido a la demora de encendido en el módulo de potencia; aplicando el modelamiento del sistema y cuyo periodo de muestreo se carga automáticamente tenemos:

Figura 51. Modelamiento del módulo de posicionamiento angular



Fuente: Autor

En la figura 51 podemos observar la función de transferencia con un porcentaje de aproximación del 97.31%, hay que tomar en cuenta el último término del polinomio del denominador el que tiende a cero o prácticamente es cero.

Ecuación 33. Función de transferencia práctica del módulo de posicionamiento angular

$$G_s = \frac{95.2054}{s^3 + 6.415s^2 + 20.0876s}$$

Fuente: Autor

5.7. PRÁCTICA 5: ADQUISICIÓN Y MODELADO - PLANTA DE LUMINOSIDAD

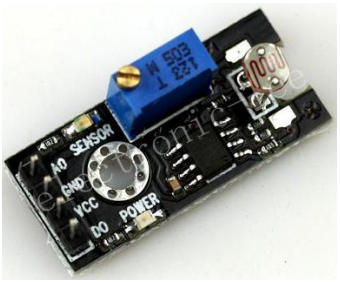
El quinto módulo se trata de obtener la función de transferencia de la cantidad de luz producida por una lámpara cuando este se enciende, obteniéndose una función de luminosidad con respecto al tiempo, cabe recalcar que la fuente de voltaje utilizada para esta práctica es proporcionada por el módulo mismo, teniendo que accionarle con el módulo de potencia.

Figura 52. Módulo de luminosidad

Fuente: Autor

El sensor utilizado en este módulo se lo puede observar en la siguiente figura, el cual ya viene acondicionado y permite convertir la cantidad de luz existente en el ambiente a una variación de voltaje.

Figura 53. Sensor utilizado



Fuente: http://www.ebay.com

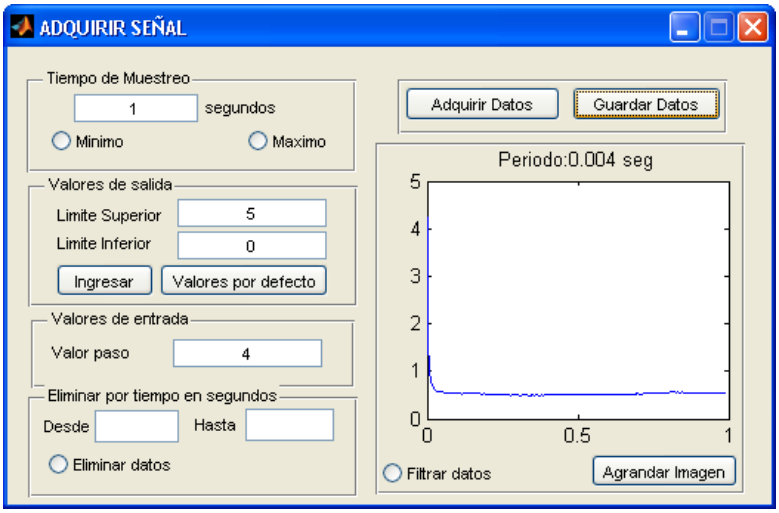
Tiene las siguientes características:

a. Dos salidas: análoga y digital.
b. Fuente de alimentación de 3-24V_{DC}.
c. Tamaño reducido.
d. Detecta de la intensidad de la luz.

5.7.1. MODELAMIENTO PRÁCTICO

Debido a que no se ha encontrado una función teórica, existen muchas posibilidades para este módulo; entonces, debemos escoger la más adecuada teniendo en cuenta el grado de aproximación como el orden de la función de transferencia. Si encontramos una función que tenga bajo orden y grado de aproximación en coherencia con las ecuaciones de orden superior, es recomendable escoger el primero mencionado.

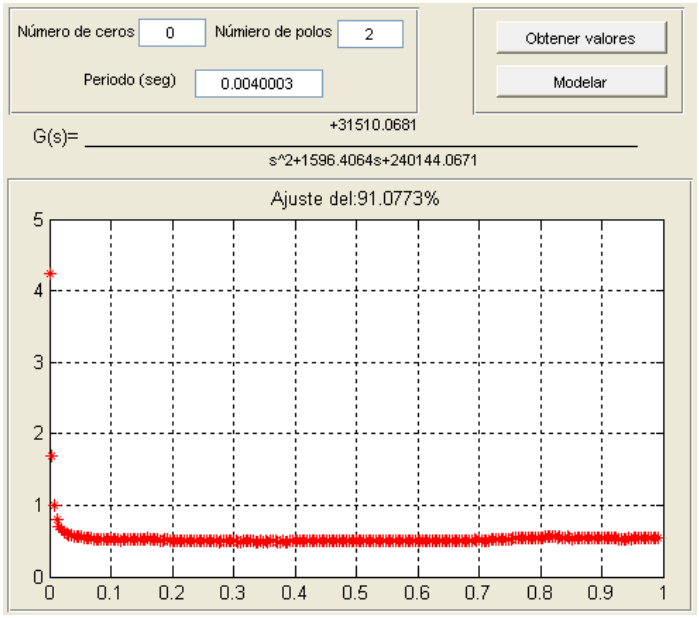
Figura 54. Adquisición de datos para el módulo de luminosidad



Fuente: Autor.

Aplicando el modelamiento del sistema con una cantidad nula de ceros y un polo además de cuyo periodo se carga automáticamente, tenemos:

Figura 55. Modelamiento del módulo de luminosidad



Fuente: Autor

En la figura 55 podemos observar la función de transferencia con un porcentaje de aproximación del 89.25%.

En base a pruebas realizadas, se determinó que la mejor relación con respecto al número de polos y ceros corresponde a una función de primer orden, con una cantidad nula de ceros y un polo. Además se aclara que el módulo fue probado en la noche en un cuarto oscuro.

Ecuación 34. Función de transferencia práctica del módulo de luminosidad

$$G_s = \frac{35.163}{s + 267.194}$$

Fuente: Autor

5.8. VALIDACIÓN DEL PROYECTO

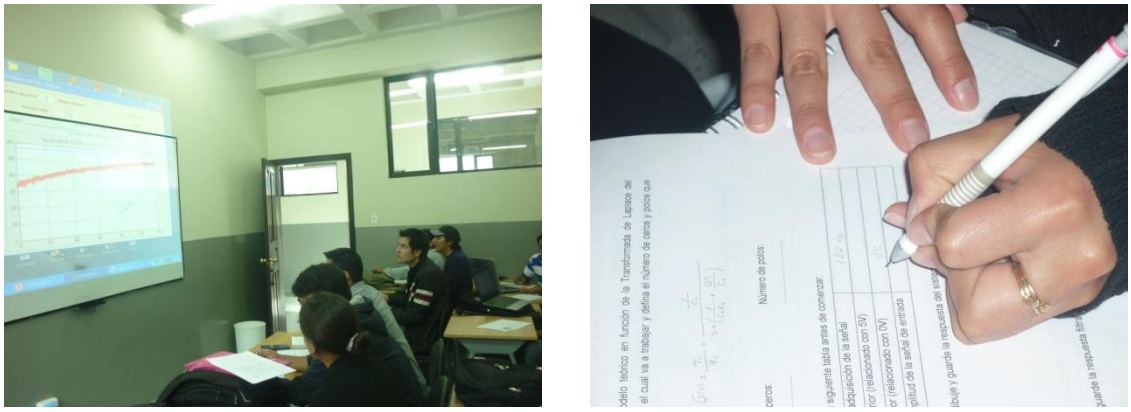
El proyecto desarrollado fue probado en un curso de la Universidad Técnica del Norte facultad FICA en la materia de “Sistemas de Control I y II” en el mes de julio del período académico marzo-agosto 2013. En el Anexo 10 se muestran fotografías que respaldan lo anteriormente mencionado.

Las actividades desarrolladas para la validación del sistema consistieron en:

1. Presentación del funcionamiento del sistema
2. Pruebas utilizando el módulo de temperatura
3. Análisis de resultados obtenidos de la práctica
4. Implementación de sistemas de control para los módulos de temperatura, posicionamiento angular, luminosidad y RC.

En la siguiente figura se puede observar a los estudiantes realizando la práctica y demostración del sistema funcionando:

Figura 56. Validación del proyecto



Fuente: Autor

Con la presentación del funcionamiento del sistema al docente y estudiantes de la asignatura de Sistemas de Control, se ha comprobado la utilidad del proyecto desarrollado como complemento a la aplicación de los conocimientos en la práctica dentro del proceso enseñanza aprendizaje; permitiendo así tanto al docente facilitar la explicación referente al tema, como a los estudiantes para que adquieran habilidades en el modelamiento de sistemas lineales a través de los módulos de prácticas realizados, además quedando abierta la posibilidad de realizar diferentes módulos que los presentados en este capítulo.

CAPÍTULO VI

CONCLUSIONES Y RECOMENDACIONES

6.1. CONCLUSIONES

- En el presente trabajo se ha logrado desarrollar un módulo didáctico para el modelamiento de sistemas lineales, el cual brinda ayuda en lo que se refiere a la adquisición y modelamiento de plantas, con la incorporado módulos de temperatura, posicionamiento angular, velocidad angular, y luminosidad, para la realización de prácticas de laboratorio, brindando una facilidad en la manipulación y conexión de estos con la Tarjeta DAQ.

- Este proyecto ofrece una ventaja en la elección del período de muestreo y el número de muestras a obtener, ya que el usuario solamente debe definir el tiempo de adquisición de datos, obteniéndose automáticamente los valores anteriormente mencionados.

- La aplicación desarrollada toma en cuenta los datos obtenidos en amplitud como el tiempo de adquisición de los mismos, para definir una función de transferencia la cual supone y predice el comportamiento del sistema frente a cualquier tipo de señal con la cual va a interactuar.

- Se dispone de un software con robustez en procesamiento matemático para realizar el proceso de aproximación y modelamiento de los datos obtenidos a una señal continua, lo cual tiene eficiencia en el modelado de los mismos.

- El proyecto cuenta con el desarrollo de una tarjeta DAQ dando la ventaja de la implementación de otro dispositivo con las mismas características, a un costo económico relativamente bajo a diferencia de las DAQ comerciales y alcanzable para que el estudiante la realice.

- El funcionamiento del sistema desarrollado fue presentado y validado al docente y estudiantes de la asignatura “Sistemas de Control I y II” en la UTN. Comprobándose así su utilidad como complemento a la aplicación de los conocimientos en la práctica dentro del proceso enseñanza aprendizaje; permitiendo tanto al docente facilitar la explicación referente al tema, como a los estudiantes para que adquieran habilidades en el modelamiento de sistemas lineales a través de los módulos de prácticas realizados.

6.2.RECOMENDACIONES

- Al momento de modelar un sistema o planta debemos guiarnos por la función teórica de la misma, ya que nos entrega una guía del funcionamiento y una clara definición de las futuras acciones del sistema, en las cuales va a responder frente a un tipo de señal de entrada diferente a la función paso.

- Frente a la señal obtenida del sistema a modelar, es necesario un tratamiento de datos para la reducción en amplitud del ruido acoplado o la eliminación del mismo, para obtener una señal más nítida y aproximada a la real del sistema con el cual se va a interactuar.

- Para una buena visualización y representación de la señal característica de un sistema o planta, es necesario tener ventajas en la DAQ que se está utilizando, ya que existen diferentes parámetros como la velocidad y tipo de comunicación así como la frecuencia de adquisición de datos que definen su elección.

- La mayoría de sistemas lineales o plantas necesitan grandes cantidades de energía, misma que la DAQ no puede suministrar, entonces, se recomienda un tipo de interfaz que permita manejar potencia y cuya característica principal debe ser la velocidad de respuesta como la potencia máxima soportada.

- Para cada sistema con la cual el usuario desee trabajar, se recomienda utilizar una fuente de energía externa para la DAQ funcione adecuadamente, debiéndose utilizar la energía proporcionada por esta última en procesos que

requieran el consumo de bajas cantidades de energía como el circuito acondicionador de la señal del sensor.

- El proyecto desarrollado ofrece la posibilidad de realizar diferentes módulos de prácticas de laboratorio que las incorporadas actualmente, para lo cual en su implementación se debe tener en cuenta los requerimientos específicos o limitados por la DAQ.

BIBLIOGRAFÍA

1. Alciatore, D. G., & Histand, M. B. (2008). *Introducción a la mecatrónica y los sistemas de medición.* McGraw-Hill.
2. Barragán, D. (5 de Agosto de 2008). *Manual de interfaz gráfica de usuario en matlab.* Recuperado el Agosto de 2012, de MATPIC.COM: http://www.matpic.com
3. Bishop, R. H. (2008). *Mechatronic system control, logic, and data acquisition* (2 ed.). U.S.A.
4. Bolton, W. (2006). *Ingeniería de control* (2 ed.). México: Alfaomega.
5. Bolton, W. (2010). *Mecatrónica. Sistemas de Control Electrónico en la Ingeniería Mecánica y Eléctrica* (4 ed.). México: Alfaomega.
6. Bolzern, P., Scattolini, R., & Schiavoni, N. (2009). *Fundamentos de control automático.* McGraw-Hill.
7. Boylestad, R. L., & Nashelsky, L. (2009). *Electrónica: teoría de circuitos y dispositivos electrónicos* (10 ed.). Madrid, España: Prentice Hall.
8. CEKIT. (2002). *Microcontroladores* (Vol. TOMO I y II). Colombia.
9. CEKIT. (s.f.). *Curso fácil de electrónica básica* (Vol. TOMO 1 y 2). Pereira, Colombia.
10. Collaguazo, G. (2007). *Sistemas Microprocesados.* Ibarra, Ecuador: Inédito.
11. Collaguazo, G. (SN). *Matlab herramienta para el aprendizaje.* Ibarra, Ecuador: Inédito.
12. Coughlin, R. F., & Driscoll, F. F. (1999). *Amplificadores Operacionales Y Circuitos Integrados Lineales* (5 ed.). México: Pearson Educación.
13. Creus Solé, A. (2011). *Instrumentación Industrial.* México: Alfaomega.

14. Dorf, R., & Bishop, R. (2005). *Sistemas de control moderno* (10 ed.). Madrid: Pearson.

15. Franco, S. (2005). *Diseño con amplificadores operacionales y circuitos integrados analógicos* (3 ed.). México: McGraw-Hill.

16. Gomáriz Castro, S., Biél Solé, D., Matas Alcalá, J., & Reyes Moreno, M. (1999). *Teoría de control. Diseño electrónico* (2 ed.). México: Alfaomega.

17. Kunusch, C. (2003). *Identificación de sistemas dinámicos.* Inédito.

18. Kuo, B. C. (1996). *Sistemas de control automático.* Prentice Hall.

19. Kuo, B. C. (1999). *Sistemas de control digital.* Compañia Editorial Continental.

20. López Guillén, E. (SN). *Identificación de sistemas. Aplicación al modelo de un motor de continua.* SN: Inédito.

21. Malvino, A. P., & Bates, D. J. (2007). *Principios de electrónica* (7 ed.). Madrid, España: McGraw-Hill.

22. MathWorks, Inc. (Marzo de 2013). *Control System Toolbox. User's Guide.* Recuperado el Abril de 2013, de MathWorks: http://www.mathworks.com/help/index.html

23. MathWorks, Inc. (Marzo de 2013). *MATLAB Primer.* Recuperado el Abril de 2013, de MathWorks: http://www.mathworks.com/help/index.html

24. MathWorks, Inc. (Marzo de 2013). *System Identification Toolbox. User's Guide.* Recuperado el Abril de 2013, de MathWorks: http://www.mathworks.com/help/index.html

25. Meythaler Naranjo, A. (2005). Diseño e implementación de un sistema de adquisición de datos para instrumentación virtual utilizando un microcontrolador PIC. *Proyecto de grado para la obtención del título en Ingeniería Electrónica e Instrumentación*. Latacunga.

26. MICROCHIP. (s.f.). *MICROCHIP*. Recuperado el 26 de Marzo de 2013, de http://www.microchip.com

27. Ogata, K. (1996). *Sistemas de control en tiempo discreto* (2 ed.). México: Prentice Hall.

28. Ogata, K. (2010). *Ingeniería de control moderna* (5 ed.). Pearson Educación.

29. Pool, G. (8 de Abril de 2010). *Comunicación entre MATLAB y PIC de MICROCHIP usando puerto USB*. Recuperado el 2012, de MathWorks: http://www.mathworks.com/matlabcentral/fileexchange/24417-comunicacion-entre-matlab-y-pic-de-microchip-usando-puerto-usb

30. Reinoso García, O., Sebastián y Zuñiga, J. M., & Torres Medina, F. (2004). *Control de sistemas discretos.* McGraw-Hill.

31. Sedra, A. S., & Smith, K. C. (2011). *Circuitos microelectrónicos* (5 ed.). México: McGraw-Hill.

32. Soriano, J., Escobar, A., & Peña, R. (2003). Identificación con modelos discretos para sistemas lineales. Modelo matemático y aplicaciones. *Ingeniería, 8*(2), 47-55.

33. Vallejo, D. H. (2003). Enciclopedia de electrónica básica. *Saber Electrónica, TOMO 1*, 4.

ANEXOS

ANEXO 1

Manual de usuario de la interfaz gráfica

INTRODUCCIÓN

El presente documento ofrece una información para el uso de la aplicación desarrollada, siguiendo pasos y opciones predeterminadas que ayudan al usuario a realizar un buen trabajo al momento de utilizar la aplicación aquí descrita.

La interfaz gráfica permite al usuario la facilidad de recopilar datos de sistemas físicos lineales con varias opciones facilitando la relación de valores como la manipulación de los mismos generando una señal adecuada para luego proceder a la representación a una ecuación que representa una señal continua obtenida bajo el método de aproximación de curvas con datos experimentales.

La aplicación utiliza como medio de interacción con el medio exterior una tarjeta de adquisición de datos la cual usa comunicación USB-BULK que facilita la acción anteriormente mencionada.

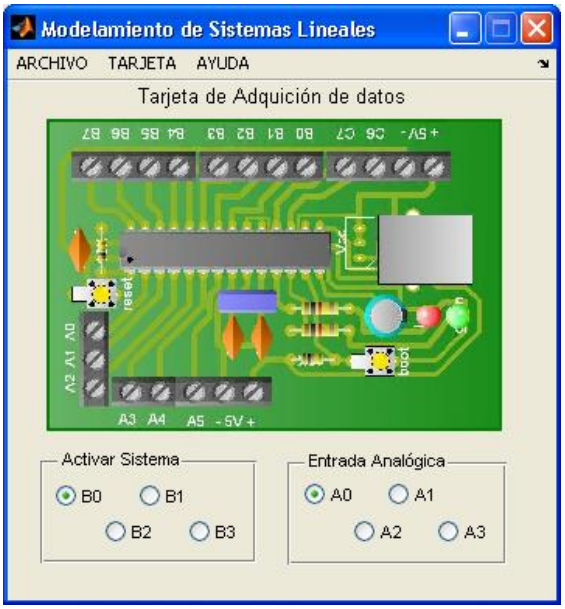
DESCRIPCIÓN

En este apartado se realiza una breve descripción de las partes que contiene la interfaz, explicando la función que efectúa cada una de las opciones mencionadas pertenecientes a tres aplicaciones desarrolladas definiendo a una de ellas como menú principal o ventana de inicio.

VENTANA DE INICIO

Al momento de iniciar la aplicación desarrollada en MATLAB, aparecerá una ventana de inicio como se muestra en la figura inferior, la cual contiene menús que se explican a continuación:

Figura A1. Ventana principal



Fuente: Autor

A. Archivo

1. Adquirir datos

Opción que permite al usuario ingresar a la aplicación de adquisición de datos, con teclas de acceso rápido “Ctrl+A”.

2. Modelar planta

Contiene dos ventanas que permiten al usuario ingresar a una aplicación para el modelamiento de datos aproximando a una curva.

2.1. Predeterminado

Opción que permite al usuario ingresar a una aplicación propia del software, con teclas de acceso rápido “Ctrl+B”.

2.2. MATLAB

Opción que permite al usuario ingresar a la aplicación de aproximación de curvas propia de MATLAB, con teclas de acceso rápido “Ctrl+C”.

3. Salir

Opción que permite al usuario cerrar la aplicación, también se puede realizar la misma acción con teclas de acceso rápido “Ctrl+D”.

B. Tarjeta

Que contiene una única opción que permite al usuario iniciar o terminar la comunicación de la aplicación desarrollada con la tarjeta de adquisición de datos, reconociendo como teclas de acceso rápido “Ctrl+E”.

C. Ayuda

1. **Autor**

   Opción que permite al usuario conocer los datos del autor, con teclas de acceso rápido “Ctrl+F”.

2. **Referencias de la obra**

   Opción que permite al usuario conocer los datos de la obra, con teclas de acceso rápido “Ctrl+G”.

3. **Manual de usuario**

   Opción que permite al usuario abrir el documento de ayuda para el manejo de la aplicación, con teclas de acceso rápido “Ctrl+H”.

4. **Guía de prácticas**

   Opción que permite al usuario abrir un documento para el desarrollo de prácticas de laboratorio, con teclas de acceso rápido “Ctrl+I”.

D. Otros

Además de los menús descritos anteriormente, la ventana también contiene dos opciones que permite la elección de entrada y salida a ser utilizada para la adquisición de datos.

1. **Activar Sistema**

   Opción que permite la elección de la salida digital a ser utilizada por la tarjeta de adquisición de datos para la activación del sistema físico lineal a ser modelado.

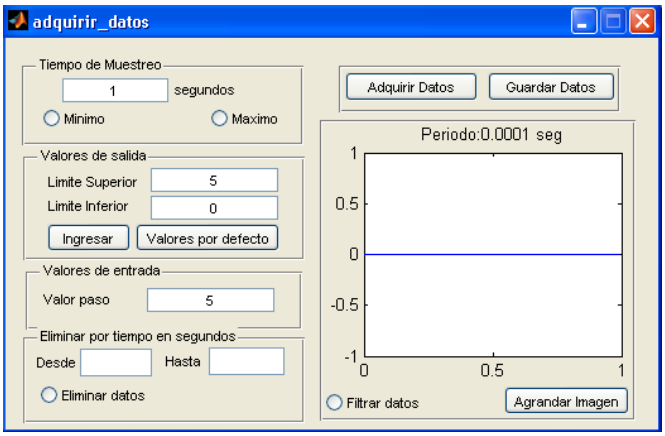
2. **Entrada Analógica**

   Opción que permite la elección de la entrada analógica a ser utilizada por la tarjeta de adquisición de datos para el ingreso de la señal del sistema físico activado por la salida digital de la tarjeta anteriormente mencionada.

VENTANA DE ADQUISICIÓN DE LA SEÑAL

La ventana de la figura inferior contiene varias opciones que permiten la adquisición de datos formando una señal, permitiendo al usuario la manipulación de la misma para la obtención de una señal mejorada, también permite guardar los valores ya definidos en un archivo para su posterior modelado. Las opciones que contiene esta aplicación se explican a continuación:

Figura A2. Ventana de adquisición de datos



Fuente: Autor

A. Tiempo de Muestreo

Utilizado para establecer el tiempo total de adquisición de datos para la reconstrucción de la señal a muestrear, definiendo como límite inferior a 1 segundo y como límite superior a 30 minutos.

B. Adquirir Datos

Permite al usuario iniciar la adquisición de datos en el tiempo determinado en el literal A, al terminarse el tiempo de muestreo aparecerá automáticamente la gráfica de la señal adquirida

C. Valores de entrada

Permite al usuario definir el valor en amplitud de la entrada paso del sistema a modelar, teniendo un valor por defecto de cinco voltios el cual es proporcionado por la salida digital de la DAQ.

D. Valores de salida

Representa los valores límites mínimo y máximo que tiene la señal, comparando al límite mínimo con un valor de voltaje de cero y al límite máximo con un valor de cinco. Esta opción actúa mediante dos botones, los cuales son:

1. **Ingresar**

   Permite al usuario ingresar nuevos valores para configurar la señal ya obtenida, ampliado o reduciendo su amplitud.

2. **Valores por defecto**

   Definido para regresar a los valores de voltaje mencionados anteriormente.

E. Eliminar por tiempo en segundos

Ya obtenida la señal, el usuario puede eliminar parte de ella definiendo los límites superior e inferior de tiempo en los cuales se quiere realizar dicha acción. La acción se realizará dando click a “Eliminar datos”.

Una vez eliminado los datos la opción cambiará de nombre a “Recuperar datos” para que el usuario pueda recuperarlos, dando un click la opción antes mencionada.

F. Filtrar datos

Permite al usuario una mejor visualización de la señal; está opción es utilizada cuando se requiera atenuar el ruido acoplado a la señal perteneciente al sistema físico con el que se está trabajando.

G. Agrandar imagen

Permite al usuario agrandar la imagen para una mejor visualización de la señal obtenida.

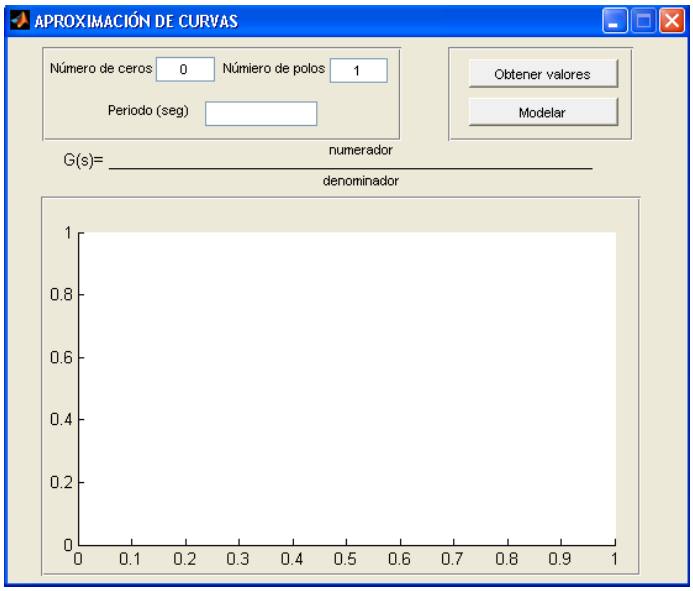
H. Guardar Datos

Esta opción permite guardar permanentemente los datos obtenidos y manipulados de la señal de salida del sistema físico. Los valores se guardan en un archivo con la extensión "*.mat", la dirección es especificada por el usuario.

VENTANA DE MODELAMIENTO DE LA SEÑAL

La ventana de la figura inferior permite al usuario la elección de valores en los cuales se requiere que se realice la aproximación de la curva a los datos obtenidos experimentalmente, entregando la aproximación a una ecuación en función de la transformada de Laplace. Las opciones que contiene la aplicación se explican a continuación:

Figura A3. Ventana de modelamiento de la señal



Fuente: Autor

A. Obtener valores

Permite al usuario elegir un archivo para cargar los valores de la señal previamente guardada en la ventana de adquisición de la señal, también se carga el valor del periodo de muestreo.

B. Número de ceros

Permite elegir el grado del polinomio del numerador contenida en la función de la transformada de Laplace a aproximar. Ejemplos, siendo k_1 y k_2 constantes:

- Si es 0, el polinomio será "k_2".
- Si es 1, el polinomio será "s+k_2".
- Si es 2, el polinomio será "s^2 + k_1s+k_2", etc.

C. Número de polos

Permite elegir el grado del polinomio del denominador contenida en la función de la transformada de Laplace a aproximar. Ejemplos, siendo k_3 y k_4 constantes:

- Si es 0, el polinomio será "k_4".
- Si es 1, el polinomio será "s+k_4".
- Si es 2, el polinomio será "s^2 + k_3s+k_4", etc.

D. Modelar

Opción que permite al usuario la aproximación a una curva, con los valores definidos por el usuario de los literales mencionados anteriormente.

E. Periodo

Opción que permite al usuario cambiar el periodo de muestreo de la señal cargada, pudiéndose comprobar la necesidad del valor del mismo para una correcta aproximación.

GUÍA DE USO

En este apartado se describen pasos a seguir para una correcta utilización de la interfaz gráfica. Antes de empezar con la descripción de dichos pasos, es necesario recalcar que la interfaz está dividida en tres partes:

A. PASOS PARA LA VENTANA PRINCIPAL

1. Definir la salida digital a ser usada.
2. Definir la entrada analógica a ser usada.
3. Conectar la tarjeta.
4. Click en archivo y luego en adquirir datos, para ingresar a la ventana de adquisición de la señal.
5. Click en archivo y luego en modelar datos; en esta opción se pueden elegir dos ventanas a ser abiertas.
   5.1. Para ingresar a la ventana de modelamiento de la señal proporcionada por la interfaz presente, debe hacer click en predeterminado.
   5.2. Para ingresar a la interfaz de aproximación de datos proporcionada por MATLAB, debe hacer click en MATLAB.
6. Desconectar la tarjeta.
7. Salir de la aplicación.

Los literales 4, 5.1 y 5.2 son opcionales, es decir, se puede ingresar a cualquiera de ellos automáticamente, siendo el literal 3 un paso esencial para cumplir con el literal siguiente.

B. PASOS PARA LA VENTANA DE ADQUISICIÓN DE LA SEÑAL

1. Definir el tiempo de muestreo en el cual se va a adquirir la señal.
2. Pulsar el botón “Adquirir Datos”.
3. Esperar hasta que se cumpla el tiempo de adquisición, su finalización se presenta debido a que aparecerá la imagen de la señal con el periodo de muestreo a la que fue adquirido
4. Establecer el valor de entrada paso al cual fue conectado al sistema físico a modelar.

5. Establecer los valores límites a ser utilizados para el cambio de amplitud en la señal muestreada.
6. Definir un rango de tiempo para eliminar los valores de la señal en dicho rango; para elegirlo puede ayudarse con el botón de agrandar imagen. Este paso es opcional.
7. Pulsar “Filtrar datos” para atenuar el ruido acoplado en la señal. Este paso es opcional.
8. Pulsar el botón “Guardar Datos” para guardar los valores de la señal adquirida.
9. Repetir los pasos del 1 al 8 para realizar otra adquisición de datos.
10. Cerrar la aplicación.

C. PASOS PARA LA VENTANA DE MODELAMIENTO DE LA SEÑAL

1. Pulsar el botón “Obtener valores” para cargar un archivo guardado que contiene los datos de una señal adquirida.
2. Establecer el número de ceros para la función a aproximar.
3. Establecer el número de polos para la función a aproximar.
4. Pulsar el botón “Modelar” para ajustar los valores a una curva con los parámetros ya definidos.
5. Esperar a que finalice el ajuste de los valores, al momento de que se termina aparecerá automáticamente la función aproximada con el porcentaje de ajuste.
6. Si el usuario lo considera necesario puede cambiar el periodo de muestreo para visualizar un modelo diferente.
7. Repetir los pasos de 2 al 6, para la obtención de una curva diferente. Este paso es opcional.
8. Repetir los pasos del 1 al 7 para cargar y aproximar otros valores de señales diferentes.
9. Salir de la aplicación.

ANEXO 2

Guía de prácticas

INTRODUCCIÓN

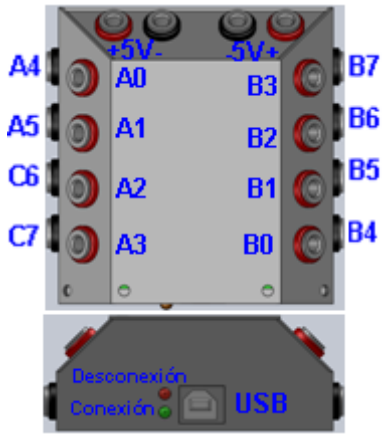
El presente documento realiza una descripción de los pasos a seguir para la realización de prácticas de laboratorio utilizando el software “Modelamiento de sistemas lineales”, contando con cinco prácticas descritas en el proyecto de trabajo de grado cuyo tema es: “Módulo didáctico para el modelamiento de sistemas lineales con MATLAB y tarjeta compatible USB”.

Además de las prácticas proporcionadas en este documento se proporciona una guía general para construcción de nuevos módulos para ser utilizados tomando en cuenta los requisitos definidos por el proyecto anteriormente mencionado. Principalmente se explica las características de uso y conexión de: la DAQ y el módulo de interfaz de potencia para el uso adecuado de los mismos.

UTILIZACIÓN DE LA DAQ

Como se puede observar en la figura inferior la DAQ está compuesta por:

Figura A4. Distribución de pines de la DAQ



Fuente: Autor

1. 16 pines para interacción con el medio exterior.
2. Dos pares de pines que ofrece salida de voltaje aproximadamente de 5V.
3. Un LED verde que indica la conexión de la DAQ con la PC.
4. Un LED rojo que indica la desconexión de la DAQ con la PC.

5. Un puerto USB hembra tipo-B.

De los 16 pines solamente ocho son utilizados como medio de comunicación entre el sistema a modelar y el software nombrado anteriormente. Los pines mencionados están divididos en dos grupos:

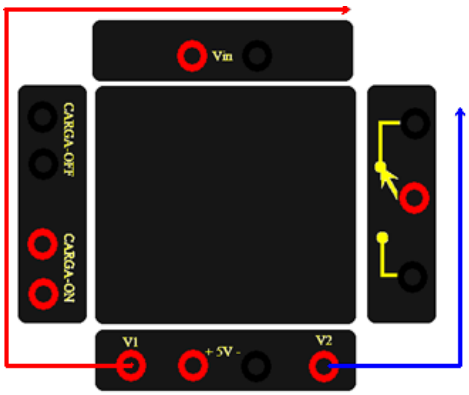
a. **Salidas activadoras:** Estos pines permiten activar el sistema a modelar utilizando una entrada paso, cuyo valor por defecto es 5V, pudiéndolo modificar utilizando un módulo de potencia el cual se describe en el siguiente tema. Los pines que permiten realizar dicha acción son: B0, B1, B2 y B3.

b. **Entrada de valores:** Estos pines permiten introducir los valores de respuesta a la salida del sistema a modelar a la PC por medio de la lectura de voltaje; cabe importante destacar que el rango de valores de respuesta del sistema debe estar comprendido en voltaje con un rango de amplitud de cero a cinco, con gran relevancia en la lectura de un pin a la vez. Los pines que permiten realizar dicha acción son: A0, A1, A2, A3.

UTILIZACIÓN DEL MÓDULO DE INTERFAZ DE POTENCIA

Como se puede observar en la figura inferior, el módulo de potencia necesita de una entrada fuente de voltaje de 5V la cual se puede obtener de la DAQ, además está compuesto por dos grupos:

Figura A5. Distribución de pines del módulo de interfaz de potencia



Fuente: Autor

a. **Activación por switch:** Controlado por el pin V2, el cual permite el control de la parte del módulo mostrado por la flecha de color azul. Normalmente se utilizan los pines del switch normalmente abierto.
b. **Activación voltaje:** Controlado por el pin V!, el cual permite el control de la parte del módulo mostrado por la flecha de color rojo. El par de pines Vin permite introducir voltajes diferentes con mayores cantidades de energía, El par CARGA-OFF permite iniciar con la carga apagada y el par CARGA-ON inicia con la carga prendida. Normalmente se utilizan los pares de pines Vin en conjunto con CARGA-OFF.

PRÁCTICAS DE LABORATORIO

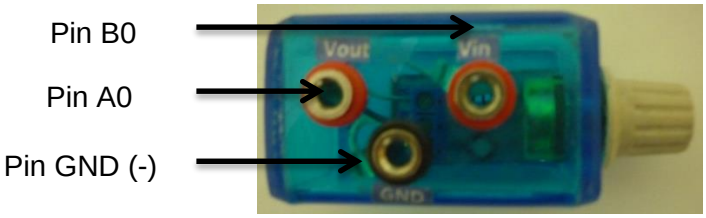
El proyecto cuenta con cinco diferentes sistemas lineales con los cuales se puede realizar prácticas de laboratorio para que el estudiante se introduzca en la materia de sistemas de control obteniendo conocimientos prácticos en base al modelamiento de mencionados sistemas y la elección de una función adecuada que represente el comportamiento del mismo.

A continuación se enuncian los sistemas desarrollados para su correcto uso: tanto en conexión y manipulación, para la realización sin ninguna dificultad de las distintas prácticas de laboratorio, tomando como pines definidos por defecto a B0 (activar sistema) y A0 (señal de salida del sistema).

1. Circuito RC

El siguiente sistema está compuesto de tres componentes: un capacitor en el cual se realiza la adquisición de datos, una resistencia de 1 kΩ para limitar la corriente y un potenciómetro de 10 kΩ para cambiar el tiempo de establecimiento del sistema en sí. En la siguiente imagen se muestra el diagrama de conexión del sistema a la DAQ.

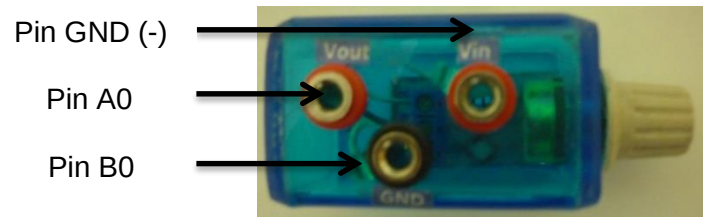
Figura A6. Conexión del módulo R-C



Fuente: Autor

La conexión de la figura A6 nos interesa para obtener la respuesta del voltaje en el capacitor, pero si queremos obtener la respuesta del resistor realice la siguiente conexión.

Figura A7. Conexión del módulo C-R

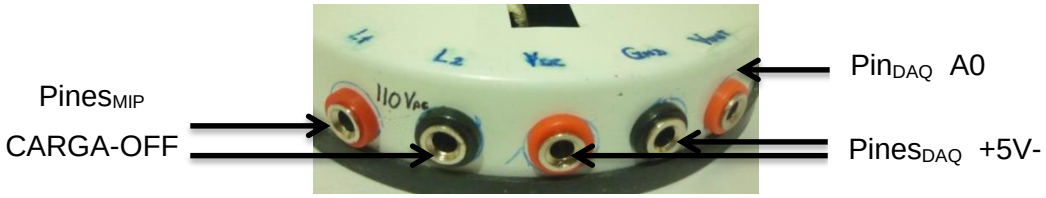


Fuente: Autor

2. Temperatura

El siguiente módulo tiene como requisito una señal de entrada de 110 Vac, la cual se la puede suministrar realizando la siguiente conexión a través del módulo de interfaz de potencia (MIP).

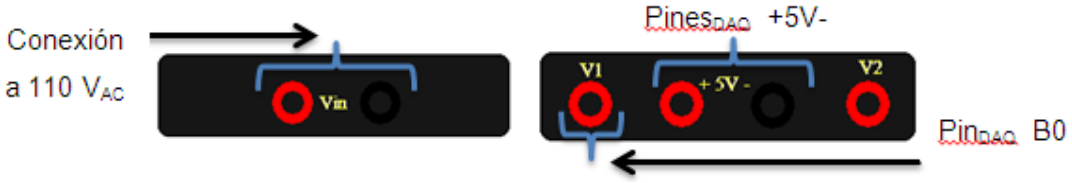
Figura A8. Conexión del módulo de temperatura



Fuente: Autor

En la figura inferior se muestra como debe ser conectado el MIP con la DAQ para este caso.

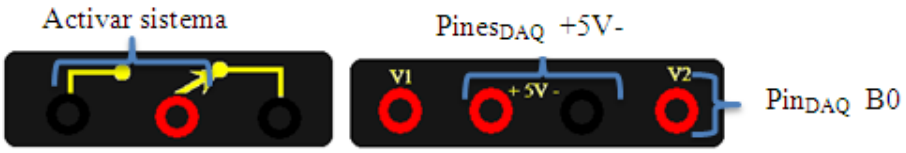
Figura A9. Conexión del MIP en V1



Fuente: Autor

Para los siguientes módulos también se necesita del uso del MIP para su funcionamiento, teniendo en común que para activarlos se lo realiza por medio de la conmutación de un switch. En la siguiente figura se muestra la conexión que se debe realizar la DAQ con el MIP.

Figura A10. Conexión del MIP en V2



Fuente: Autor

3. Velocidad angular

El presente módulo cuenta con un circuito el cual permite cambiar el valor de amplitud de la señal de estrada del sistema en 1.25 V, teniendo como requisito una fuente de voltaje externa superior a 1.25V.

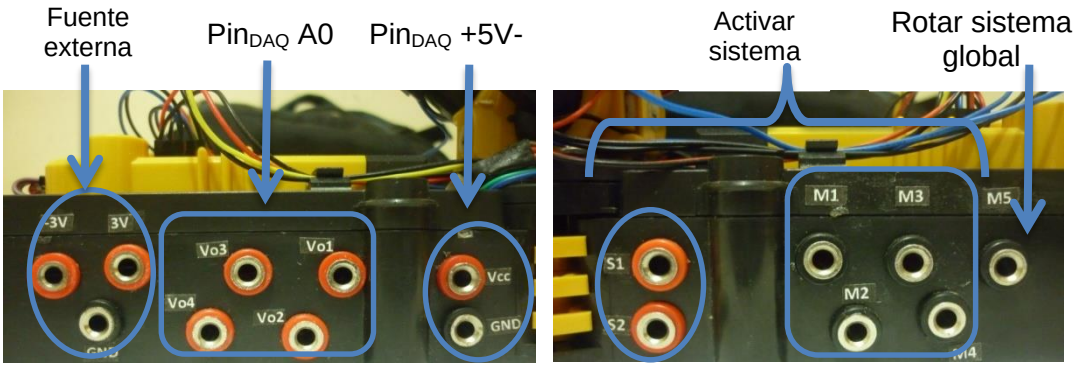
Figura A11. Conexión del módulo de velocidad angular.

Fuente: Autor

4. Posicionamiento angular

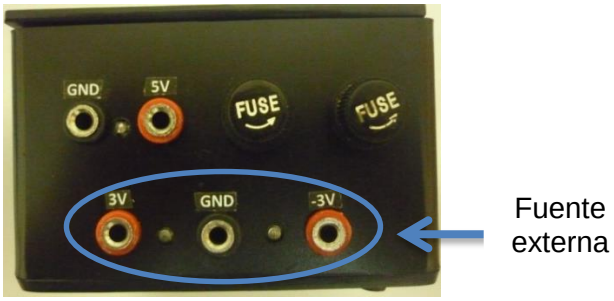
El presente módulo cuenta con cuatro sistemas incluidos para su modelamiento en posicionamiento angular, los cuales son: M1, M2, M3 y M4 teniendo como señal de salida a Vo1, Vo2, Vo3 y Vo4 respectivamente. Cabe importante recalcar que se debe realizar una práctica a la vez, es decir, activar un sistema seleccionando S1 (adelanto) o S2 (retroceso) para conmutar una de ellos con M1 a M4; el pin M5 conjuntamente con S1 o S2 realizan la acción de rotación del sistema global. El módulo cuenta con una fuente de voltaje externa de ±3 V aproximadamente.

Figura A12. Conexión del módulo de posicionamiento angular



Fuente: Autor

Figura A13. Conexión de la fuente externa

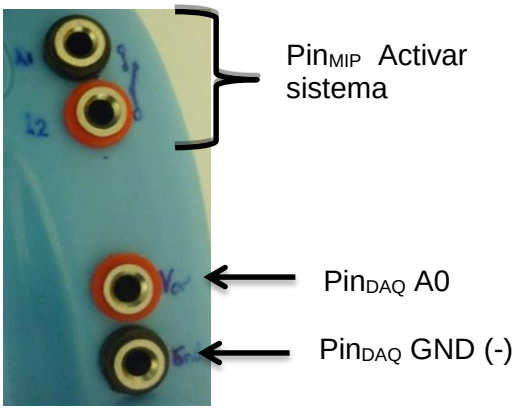


Fuente: Autor

5. Luminosidad

El presente módulo cuenta con una batería interna recargable, la cual se la utiliza ya que proporciona la energía necesaria para hacer funcionar el presente módulo. El valor de voltaje de la batería puede ser medido entre los pines L2 y GND del módulo. En la siguiente figura se muestra su diagrama de conexión.

Figura A14. Conexión del módulo de luminosidad



Fuente: Autor

Para la realización de una práctica de laboratorio es necesario realizar los siguientes pasos que se mencionan a continuación:

1. Seleccione el sistema con el que va a realizar la práctica.
2. Encuentre el modelo teórico del mismo.
3. Conecte del módulo con la DAQ y el MIP si fuere necesario.

4. Inicie la aplicación antes mencionada y siga los pasos descritos en el manual de usuario del mismo.
5. Llene la hoja de datos proporcionada a continuación.

Nota: La hoja de datos es la misma para todos los módulos, para ello se recomienda sacar una copia por cada práctica que va a realizar.

HOJA DE DATOS

PRÁCTICA DE LABORATORIO

A. Anote el modelo teórico en función de la Transformada de Laplace del sistema con el cual va a trabajar y defina el número de ceros y polos que contiene.

Número de ceros: ……..

Número de polos: ……..

B. Complete la siguiente tabla antes de comenzar.

Tiempo de adquisición de la señal	
Límite superior (relacionado con 5V)	
Límite inferior (relacionado con 0V)	
Valor de amplitud de la señal de entrada	

C. Adquiera, dibuje y guarde la respuesta del sistema.

D. Dibuje y guarde la respuesta filtrada del sistema.

E. Escriba el periodo de adquisición con la cual se obtuvo la señal.

……

F. En base al literal A, complete la siguiente tabla utilizando la ventana de “modelamiento de la señal”.

Modelo obtenido con la señal inicial	**Porcentaje de aproximación**
Modelo obtenido con la señal filtrada	**Porcentaje de aproximación**

G. Realice cada uno de los siguientes literales, defina su respuesta y escriba el modelo obtenido en cada uno.

   a. Aumente al doble el período de adquisición y obtenga el modelo. ¿Qué ocurrió con el modelo inicial? Realice la misma para T=T/3, T/2, 3T, 4T.

   b. Con periodo inicial, realice lo siguiente (escriba el porcentaje de aproximación) en base al modelo obtenido en el literal F:

      1. Aumente 1 cero y 1 polo.
      2. Aumente 2 ceros y 2 polos.
      3. Aproxime a un modelo de primer orden.
      4. Aproxime a un modelo de segundo orden.
      5. Aproxime a un modelo de tercer orden.

   c. Para cada modelo Gs obtenido en el literal b, grafique su respuesta utilizando MATLAB frente a una señal de entrada:

      1. **Paso unitario**
         Utilice la función “step(Gs)”
      2. **Impulso**
         Utilice la función “impulse(Gs)”
      3. **Rampa:** defina “ r = tf([1],[1 0]); ”
         Utilice la función “step(Gs*r)”
      4. **Seno:** defina “ seno = tf([1 0],[1 0 1]); ”
         Utilice la función “step(Gs*seno)”

NO RESPONDE LA DAQ

Esta situación se puede presentar por los siguientes motivos, después de haber conectado con el software de modelado:

1. En el proceso de la adquisición de datos, el usuario desconecto el cable USB de la DAQ o de la PC.
2. La DAQ se colgó por algún motivo interno (saturación de energía) o externo (conexión en el sistema a modelar)

SOLUCIÓN:

Para los motivos mencionados anteriormente debe seguir los siguientes pasos.

1. Diríjase a la ventana principal del software.
2. Desconecte la tarjeta del software (Menú Tarjeta/ Desconectar)
3. Desconecte el cable USB de la PC o DAQ y vuelva a conéctela
4. Conecte la DAQ con el software (Menú Tarjeta/ Conectar)
5. Listo para usarse.

MODELAR SISTEMAS CON DATOS ADQUIRIDOS MANUALMENTE

Para este caso en particular el software también permite realizar el modelo del sistema; el usuario solamente tendrá que realizar los siguientes pasos en la ventana de comandos del software MATLAB denotando a “n” como en número de datos obtenidos:

1. Introduzca los valores de la salida del sistema como se indica a continuación:

   >> data_yout=[valor1, valor 2, valor 3, valor 4, valor 5, valor 6,… valor n] ';

2. Introduzca los valores de la entrada del sistema como se indica a continuación:

   >> data_xin=[valor 1, valor 2, valor 3, valor 4, valor 5, valor 6,… valor n] ';

3. Introduzca el valor del período de adquisición de datos de la siguiente forma:

   >> T_xy=período*ones(n,1);

4. Agrupe los datos en una variable de la siguiente forma:

   >> nombre_variable=[data_xin, data_yout, T_xy];

5. Utilizar el siguiente comando para guardar:

   >> uisave('nombre_variable');

6. Listo para modelar

CONSTRUCCIÓN DE MÓDULOS-SISTEMAS LINEALES

Si se desea desarrollar nuevos módulos para su aplicación en diferentes prácticas que las mencionadas en el documento presente, se debería regirse a las limitaciones que contiene el proyecto mencionado; de acuerdo con ello se recomienda seguir los pasos detallados a continuación que son una guía general para la implementación de sistemas a construir:

1. Diseñar el sistema con la capacidad de facilitar la conexión y desconexión para agilitar el desarrollo de prácticas de laboratorio. Como recomendación se propone utilizar conectores jacks hembra.
2. Diseñar y construir el sistema con la capacidad de soportar una señal de entrada paso ejecutada principalmente por una señal de cinco voltios enviada por la DAQ.
3. Utilizar la DAQ para consumos de energía inferiores a 40 mA de corriente eléctrica y disponer del módulo de interfaz de potencia, en caso de necesitar una cantidad mayor de energía que la que puede proporcionar la DAQ con el fin de activar la señal de entrada.
4. La señal del sensor el cual es la señal salida del sistema a construir debe comprenderse en un rango de amplitud de cero a cinco en magnitud de voltaje en valores positivos.
5. En caso de utilizar algún sensor que no proporcionare el anterior requisito, se debe proceder al acondicionamiento adecuado para cumplir con el mismo ya que la DAQ admite variaciones de señales en los valores anteriormente mencionados.

Es recomendable utilizar la energía en forma de voltaje proporcionada por la DAQ para el circuito acondicionador, si en caso fuere a necesitarse una fuente aparte, es necesario conectar la misma referencia de voltaje entre la fuente externa y la DAQ.

ANEXO 3

Código fuente del microcontrolador

/* PROGRAMA PRINCIPAL PARA LA ADQUISICIÓN DE DATOS

Elaborado por: Jonathan Tapia */

// <<BULK.c>>

```
#include <.\BULK_definiciones.h>
#include <.\BULK_funciones.h>

void main(void) {

   configurar_sistema();
   conectar:
   conexion_usb();
   while (1)
   {
      leer_adc();
      if(usb_enumerated()){                    // si el Pic está configurado via USB
         if (usb_kbhit(1)){// si el endpoint de salida contiene datos del host
            leer_datos_usb(dato_in);
            programa();
            escribir_datos_usb(dato_out);
         }
      }
      else{
         goto conectar;
      }
    }
}
```

/* PROGRAMA CABECERA PARA DEFINICIÓN DE VARIABLES A USAR

Elaborado por: Jonathan Tapia */

// <<BULK_definiciones.h >>

```
#include <18F2550.h>
#device ADC=8         //Descomente ésta opción en caso de usar el ADC a 10bits

#fuses hspll,nowdt,pll5,cpudiv2,nodebug,nolvp,mclr,nopbaden,vregen,usbdiv
//             PLL1 = Para un Xtal de 4Mhz
//             PLL2 = Para un Xtal de 8Mhz
//             PLL3 = Para un Xtal de 12Mhz
//             PLL4 = Para un Xtal de 16Mhz
//             PLL5 = Para un Xtal de 20Mhz , etc
#use delay(clock=48M)
```

```
//Para usar bootloader se debe agregar las 2 ineas siguiente
#build (reset=0x1000, interrupt=0x1008)   // redireccionamiento del reset e interrupciones
#org 0x0000, 0x0FFF {}                    //Espacio reservado para el bootloader

#define USB_HID_DEVICE     FALSE              //deshabilitamos el uso de las directivas HID
#define USB_EP1_TX_ENABLE  USB_ENABLE_BULK    //turn on EP1(EndPoint1) for IN bulk/interrupt transfers
#define USB_EP1_RX_ENABLE  USB_ENABLE_BULK    //turn on EP1(EndPoint1) for OUT bulk/interrupt transfers
#define USB_EP1_TX_SIZE    64                 //size to allocate for the tx endpoint 1 buffer
#define USB_EP1_RX_SIZE    64                 //size to allocate for the rx endpoint 1 buffer

#include <pic18_usb.h>             //Microchip PIC18Fxx5x Hardware layer for CCS's PIC USB driver
#include <.\usb_monitor_bulk.h>    //descriptors del Pic USB
#include <usb.c>                   //handles usb setup tokens and get descriptor reports

////////////////////////////////////////////////////////////////////////////
/*  EN USB_MONITOR_BULK.H
Avance hasta la sección start device descriptors (aprox en la linea 132) y reemplace los valores del vendor id, el product id y el device release number como sigue ( puede copiar las tres líneas siguiente y pegar en el archivo del descriptor) :

      0xD8,0x04,          //vendor id (0x04D8 is Microchip)
      0x0B,0x00,          //product id
      0x01,0x00,          //device release number

ESTO ES IMPORTANTE HACERLO CORRECTAMENTE DE LO CONTRARIO, EL DISPOSITIVO NO SERA RECONOCIDO POR EL DRIVER.
*/

#define  LEDV    PIN_C1
#define  LEDR    PIN_C0
#define  PULSADOR   input(PIN_C2)

#define  LED_ON  output_high
#define  LED_OFF output_low

// Direcciones de memoria válidas para PIC18F2455/2550/4455/4550
#BYTE TRISA  = 0x0F92    // Registro de control de E/S del puerto A
#BYTE TRISB  = 0x0F93    // Registro de control de E/S del puerto B
#BYTE TRISC  = 0x0F94    // Registro de control de E/S del puerto C
#BYTE PORTA  = 0x0F80    // Registro del puerto A
```

```
#BYTE PORTB  = 0x0F81    // Registro del puerto B
#BYTE PORTC  = 0x0F82    // Registro del puerto C

// Para modelado del sistema a modelar
#define ACTIVAR_PLANTA PORTB
#DEFINE IN_DIGITALS PORTB/16      // Estados de entradas digitales,
                          //      nibble alto del Puerto B

#BYTE ADCON0  = 0x0FC2
#BYTE ADCON1  = 0x0FC1                    // Registro de control del ADC
#BYTE CMCON   = 0x0FB4                    // Registro del modulo comparador
```

/* PROGRAMA DE DEFINICIÓN DE FUNCIONES UTILIZADAS POR EL PROGRAMA PRINCIPAL

Elaborado por: Jonathan Tapia */

// <<BULK_fuciones.h >>

```
int8 dato_in[64]; // vector de recepcion de datos provenientes de la PC
/*
   dato_in[0]==> Activar la salida de la planta
   dato_in[1]==> Eleccion del canal de lectura analogico ADC
   dato_in[2]==> Eleccion de salida PWM
*/
int8 dato_out[64]; // vector de envio de datos hacia la PC
/*
   dato_out[0]==> Estado del "PULSADOR" PIN_C2 //Bootloader
   dato_out[1]==> Estado de las entradas digitales B4-a-B7
   dato_out[2]==> Byte bajo del ADC  // Adquisicion
   dato_out[3]==> Byte alto del ADC  //         de datos
*/

void conexion_usb(){
   LED_ON(LEDR);      //encender led en RC1 para indicar presencia de energia
   LED_OFF(LEDV);

   usb_init();               // inicializamos el USB
   usb_task();                // habilita periferico usb e interrupciones
   usb_wait_for_enumeration(); // espera hasta sea configurado por el host

   LED_OFF(LEDR);
   LED_ON(LEDV);      // encender led en RC0 al establecer contacto con la PC
}

void configurar_sistema(){
   // Configuracion de funcion de puertos I/O
   // 1-entrada;    0-salida
```

```
   TRISA  = 0x8F;                         // Se declara el puerto A
   // A0 - A3 == entradas analogicas
   // A4 - A5 == salidas pwm
   TRISB  = 0xF0;                         // Se declara el puerto B
   // B0 - B3 == salidas digitales
   // B4 - B7 == entradas digitales
   TRISC  = 0x0C;                          // Se declara el puerto C
   // C0 - C2 == visualizacion y bootloader_comunicacion usb
   // C4 - C5 == comunicacion usb
   // C6 - C7 == salidas pwm

   ADCON1 = 0x0F;                  // Apaga el ADC inicialmente
   CMCON  = 0x07;                  // Apaga los comparadores inicialmente

   // Pines de salida a cero
   PORTA=0x00;
   PORTB=0x00;
   PORTC=0x00;
   dato_in[0]=0;
   dato_in[1]=0;
   dato_in[2]=0;

   //      ***************  CONFIGURACIÓN DEL ADC   **************
   setup_comparator(NC_NC_NC_NC);
   // Selección de AN0_TO_AN3 como entradas analógicas
   setup_adc_ports(AN0_TO_AN3);
   // Se indica el rango de voltaje que tendrá la entrada análoga
   setup_adc( VSS_VDD );
   // Indica de que pin se hará la conversion por defecto AN0
   set_adc_channel( 0 );
   // Indica la frecuencia que se usará el reloj del ADC ADC_CLOCK_DIV_16
   setup_adc( ADC_CLOCK_DIV_16);
}

void leer_datos_usb(int8 dato){
   usb_get_packet(1, dato, 64); // Recibe el paquete de datos enviados de la PC
}

void leer_adc(){
   unsigned int dato_analogo=0;
   // TRABAJANDO CON ENTRADAS ANALOGAS
   read_adc(ADC_START_ONLY);
   while (bit_test(ADCON0,1));
   dato_analogo = read_adc(ADC_READ_ONLY);
   //dato_out[2] = make8(dato_analogo,0); //Byte bajo
   //dato_out[3] = make8(dato_analogo,1); //Byte alto
   dato_out[2] = dato_analogo; //Byte bajo
   dato_out[3] = 0; //Byte alto
}
```

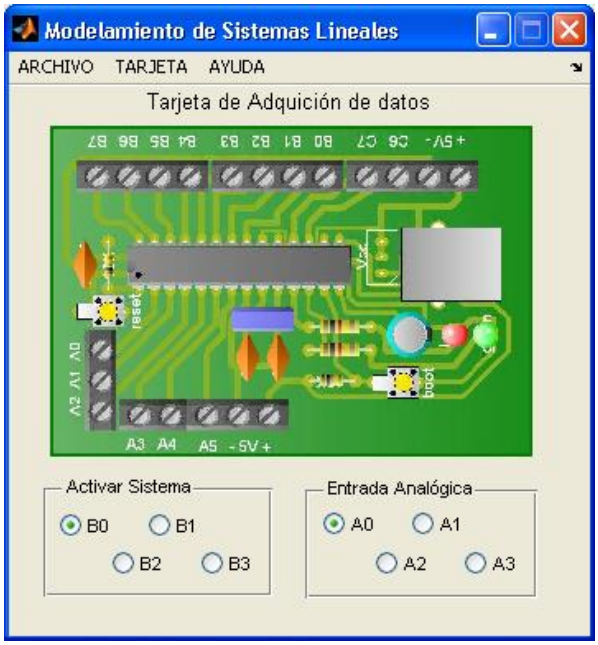
```
void programa(){
   // TRABAJANDO CON IN/OUT DIGITALES
   dato_out[0] = PULSADOR;       // Estado del PULSADOR Bootloader
   dato_out[1] = IN_DIGITALS;    // Estados de entradas digitales
   ACTIVAR_PLANTA=dato_in[0];    // Activacion del sistema

   // LECTURA DEL ADC
   set_adc_channel( dato_in[1] );// Indica de que pin se hará la conversion
   delay_us(10);
}

void escribir_datos_usb(int8 dato){
   usb_put_packet(1, dato, 64, USB_DTS_TOGGLE); // Envío el paquete de datos
                                                // a la PC
}
```

ANEXO 4

Código fuente de la interfaz gráfica

PROGRAMA PRINCIPAL

Elaborado por: Jonathan Tapia




```
% --------------------------------------------------------------------
% ---------------                TÍTULO DE LA OBRA                -----
% --------------- MÓDULO DIDÁCTICO PARA EL MODELAMIENTO DE SISTEMAS -----
% ---------------     LINEALES CON MATLAB Y TARJETA COMPATIBLE USB  -----
% ---------------                                                  -----
% ---------------           UNIVERSIDAD TÉCNICA DEL NORTE          -----
% ---------------                   FICA - CIME                    -----
% ---------------                                                  -----
% --------------- AUTOR: JONATHAN JAVIER TAPIA MARROQUÍN           -----
% --------------------------------------------------------------------

% --------------------------------------------------------------------
% --------------------  Modelamiento_Sistemas_Lineales  ---------------
% --------------------------------------------------------------------

function varargout = Modelamiento_Sistemas_Lineales(varargin)
% MODELAMIENTO_SISTEMAS_LINEALES MATLAB code for
Modelamiento_Sistemas_Lineales.fig
%      MODELAMIENTO_SISTEMAS_LINEALES, by itself, creates a new
MODELAMIENTO_SISTEMAS_LINEALES or raises the existing
%      singleton*.
%
%      H = MODELAMIENTO_SISTEMAS_LINEALES returns the handle to a new
MODELAMIENTO_SISTEMAS_LINEALES or the handle to
%      the existing singleton*.
%
%
MODELAMIENTO_SISTEMAS_LINEALES('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in MODELAMIENTO_SISTEMAS_LINEALES.M with
the given input arguments.
%
```

```
%      MODELAMIENTO_SISTEMAS_LINEALES('Property','Value',...) creates a
new MODELAMIENTO_SISTEMAS_LINEALES or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before
Modelamiento_Sistemas_Lineales_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to
Modelamiento_Sistemas_Lineales_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
Modelamiento_Sistemas_Lineales

% Last Modified by GUIDE v2.5 08-Jul-2013 16:36:01

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
@Modelamiento_Sistemas_Lineales_OpeningFcn, ...
                   'gui_OutputFcn',
@Modelamiento_Sistemas_Lineales_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Modelamiento_Sistemas_Lineales is made
visible.
function Modelamiento_Sistemas_Lineales_OpeningFcn(hObject, eventdata,
handles, varargin)
    % This function has no output args, see OutputFcn.
    % hObject    handle to figure
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    % varargin   command line arguments to Modelamiento_Sistemas_Lineales
(see VARARGIN)

    global data_in data_out vid_pid_norm out_pipe in_pipe conectado
handles1
    global out_planta
    imagen=imread('Placa_tarjeta.jpg');
```

```
    imshow(imagen);
    title('Tarjeta de Adquición de datos');

    handles1 = handles;       % para referencia global en el metodo
tiempo

    %SE DECLARA LA RUTINA QUE DEBE DE EJECUTARSE AL CERRAR LA APLICACIÓN
    set(handles.figure1,'CloseRequestFcn',@closeGUI);

    %%%%%%%%%%%%%%%%%%%%COMUNICACION CON LA TARJETA%%%%%%%%%%%%%%%%%%%%%%%
    % data_out(1)==> Activar la salida de la planta
    % data_out(2)==> Eleccion del canal de lectura analogico ADC
    % data_out(3)==> Eleccion de salida PWM
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % data_in(1)==> Estado del "PULSADOR" de la DAQ
    % data_in(2)==> Estado de entradas digitales de la DAQ
    % data_in(3)==> Byte bajo del ADC    % Adquisicion
    % data_in(4)==> Byte alto del ADC    %     de datos
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % variable inicial para reconociemiento y busqueda de la DAQ
    conectado = 0;

    %%%%%%%%            LOS DATOS SE DECLARAN COMO "UINT8"       %%%%%%%%
    %%%%%%%%           de lo contrario no hay comunicación.      %%%%%%%%
    % Declaracion del vector de datos de entrada (se recibe de la DAQ)
    data_in = eye(1,64,'uint8');
    % Declaración del vector de datos de salida (se envia a la DAQ)
    data_out = eye(1,64,'uint8');

    %%%%%%%%           Inicializacion de vectores               %%%%%%%%
    out_planta=1;     % PIN Digital 1 (Pin B0)
    data_out(1)= 0;   % Desactivar planta
    data_out(2)= 0;   % Eleccion entrada analogica 0 (Pin A0)

    % mex -setup     % instrucción de elección de compilador para
librerías dll
    vid_pid_norm = libpointer('int8Ptr',[uint8('vid_04d8&pid_000b') 0]);
    out_pipe = libpointer('int8Ptr',[uint8('\MCHP_EP1') 0]);
    in_pipe = libpointer('int8Ptr',[uint8('\MCHP_EP1') 0]);

    loadlibrary mpusbapi _mpusbapi.h alias libreria;
    clc;                                    % limpiar documentos cargados
    % calllib('libreria','MPUSBGetDLLVersion');

    %Choose default command line output for
Modelamiento_Sistemas_Lineales
    handles.output = hObject;
    % Update handles structure
    guidata(hObject, handles);
    % UIWAIT makes Modelamiento_Sistemas_Lineales wait for user
    %        response (see UIRESUME)
    % uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Modelamiento_Sistemas_Lineales_OutputFcn(~,
eventdata, handles)
    % varargout  cell array for returning output args (see VARARGOUT);
```

```
    % hObject    handle to figure
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)

    % Get default command line output from handles structure
    varargout{1} = handles.output;


function closeGUI(hObject, eventdata, handles)
    % hObject    handle to figure1 (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    global my_in_pipe my_out_pipe conectado

    selection = questdlg('Desea cerrar la aplicación?',...
                    'Confirmación',...
                    'Si','No','Si');
    switch (selection)
        case 'Si'
           if conectado == 1
               % Se cierra el tunel de recepción
               calllib('libreria', 'MPUSBClose', my_in_pipe);
               % Se cierra el tunel de envio
               calllib('libreria', 'MPUSBClose', my_out_pipe);
           end
           % Descarga la librería de memoria ( para no generar errores )
           unloadlibrary libreria
           delete(hObject)
        case 'No'
            return
    end
    % Hint: delete(hObject) closes the figure

% --------------------------------------------------------------------
% ------------------------  MENU ARCHIVO  ----------------------------
% --------------------------------------------------------------------

% --------------------------------------------------------------------
function mn_archivo_Callback(hObject, eventdata, handles)
% hObject    handle to mn_archivo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
function mm_Adquirir_datos_Callback(hObject, eventdata, handles)
% hObject    handle to mm_Adquirir_datos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    global conectado
    if(conectado==1)
        adquirir_datos;  % Abre la ventada de adquisición de datos
    else
        errordlg('Conecte la tarjeta primero');
    end


% --------------------------------------------------------------------
function mn_modelar_planta_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to mn_modelar_planta (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function mn_predeterminado_Callback(hObject, eventdata, handles)
    % hObject    handle to mn_predeterminado (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    modelo_aproximacion; % Abre la ventada de modelación de datos


% --------------------------------------------------------------------
function mm_matlab_Callback(hObject, eventdata, handles)
    % hObject    handle to mm_matlab (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    global y x T
    [f,p]=uigetfile('*.mat','PLANTA DE APROXIMACION');

    if or(isequal(f,0),isequal(p,0))
        warndlg('Elija el archivo que contiene la planta','AVISO');
        return;
    else
        %questdlg  warndlg
        warndlg({'Ecriba en "Command Window" de matlab:';...
                 '>> global x y T'},'AVISO');
        planta=open(strcat(p,f));
        planta=struct2cell(planta);
        planta=cell2mat(planta);
        x=planta(:,1);
        y=planta(:,2);
        T=planta(1,3);
        ident;
    end


% --------------------------------------------------------------------
function mn_salir_Callback(hObject, eventdata, handles)
    % hObject    handle to mn_salir (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    close(Modelamiento_Sistemas_Lineales);


% --------------------------------------------------------------------
% ----------------------  MENU TARJETA  ------------------------------
% --------------------------------------------------------------------

% --------------------------------------------------------------------
function lb_tarjeta_Callback(hObject, eventdata, handles)
    % hObject    handle to lb_tarjeta (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
```

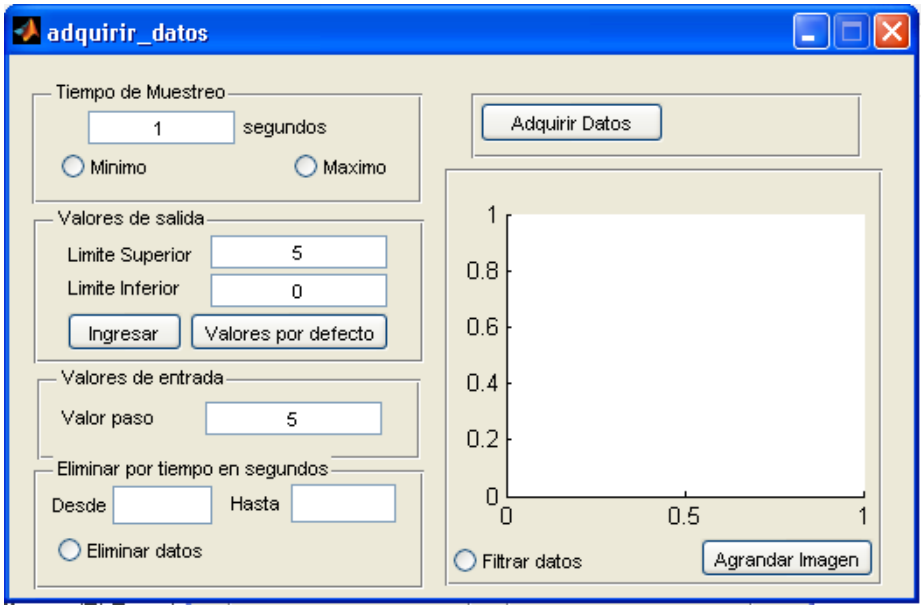
```
function mn_conectar_Callback(hObject, eventdata, handles)
    % hObject    handle to mn_conectar (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    global vid_pid_norm out_pipe in_pipe conectado my_out_pipe my_in_pipe

    name=get(handles.mn_conectar,'Label');
    if isequal(name,'Conectar') % Tarjeta conectada
       while (conectado == 0)
            [conectado] =
calllib('libreria','MPUSBGetDeviceCount',vid_pid_norm);
            if (conectado ==0)
                selection = questdlg('La tarjeta de evaluación no se
encuentra',...
                        'Tarjeta no encontrada',...
                        'Buscar de nuevo','Cancelar','Cancelar');
                switch selection,
                    case 'Cancelar',
                        return % sale del ciclo while
                    case 'Buscar de nuevo'
                        conectado = 0;
                end
            end
        end

        if conectado == 1
            helpdlg('Tarjeta de evaluación conectada','AVISO');
            % Se abre el tunel de envio
            [my_out_pipe] = calllib('libreria', 'MPUSBOpen',uint8 (0),
vid_pid_norm, out_pipe, uint8(0), uint8 (0));
            % Se abre el tunel de recepción
            [my_in_pipe] = calllib('libreria', 'MPUSBOpen',uint8 (0),
vid_pid_norm, in_pipe, uint8 (1), uint8 (0));
            set(handles.mn_conectar,'Label','Desconectar');
        end

    else  % Tarjeta desconectada
       set(handles.mn_conectar,'Label','Conectar');
       if conectado == 1
            helpdlg('Tarjeta desconectada','AVISO');
            % Se cierra el tunel de recepción
            calllib('libreria', 'MPUSBClose', my_in_pipe);
            % Se cierra el tunel de envio
            calllib('libreria', 'MPUSBClose', my_out_pipe);
        end
        conectado=0;
    end

% --------------------------------------------------------------------
% ---------------------------  PANEL GUI  ----------------------------
% --------------------------------------------------------------------


% --- Executes when selected object is changed in pnl_salida_analogica.
function pnl_salida_analogica_SelectionChangeFcn(hObject, eventdata,
handles)
% hObject    handle to the selected object in pnl_salida_analogica
% eventdata  structure with the following fields (see UIBUTTONGROUP)
%   EventName: string 'SelectionChanged' (read only)
```

```
%   OldValue: handle of the previously selected object or empty if none
was selected
%   NewValue: handle of the currently selected object
% handles    structure with handles and user data (see GUIDATA)
    global data_out
    % Elección del canal analógico de la DAQ
    if hObject == handles.rdbtn_A0
        data_out(2)= 0;
    elseif hObject == handles.rdbtn_A1
        data_out(2)= 1;
    elseif hObject == handles.rdbtn_A2
        data_out(2)= 2;
    else %Entrana A3
        data_out(2)= 3;
    end

% --- Executes when selected object is changed in pnl_activar_sistema.
function pnl_activar_sistema_SelectionChangeFcn(hObject, eventdata,
handles)
% hObject    handle to the selected object in pnl_activar_sistema
% eventdata  structure with the following fields (see UIBUTTONGROUP)
%   EventName: string 'SelectionChanged' (read only)
%   OldValue: handle of the previously selected object or empty if none
was selected
%   NewValue: handle of the currently selected object
% handles    structure with handles and user data (see GUIDATA)
    global out_planta
    % Elección de la salida digital de la DAQ
    if hObject == handles.rdbtn_B0
        out_planta=1;   % Pin digital 0
    elseif hObject == handles.rdbtn_B1
        out_planta=2;   % Pin digital 1
    elseif hObject == handles.rdbtn_B2
        out_planta=4;   % Pin digital 2
    else  % Salida PIN B3
        out_planta=8;   % Pin digital 3
    end


% --------------------------------------------------------------------
% ----------------------  MENU AYUDA  --------------------------------
% --------------------------------------------------------------------

% --------------------------------------------------------------------
function mn_ayuda_Callback(hObject, eventdata, handles)
% hObject    handle to mn_ayuda (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function mn_autor_Callback(hObject, eventdata, handles)
% hObject    handle to mn_autor (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
warndlg({'La aplicación fue desarrollada por:',...
         ' Jonathan Javier Tapia Marroquín',...
         ' ',...
         '        Estudiante de la UTN',...
```

```
        'Carrera de Ingeniería en Mecatrónica'},'Referencias del Autor')

% --------------------------------------------------------------------
function mn_referencias_de_la_obra_Callback(hObject, eventdata, handles)
% hObject    handle to mn_referencias_de_la_obra (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
warndlg({'Para más referencias consulte:',...
         '',...
         'MÓDULO DIDÁCTICO PARA EL MODELAMIENTO DE SISTEMAS',...
         'LINEALES CON MATLAB Y TARJETA COMPATIBLE USB',...
         '',...
         '      Tema de trabajo de grado',...
         'Disponible en la biblioteca de la UTN'}...
         ,'Referencias del Módulo')

% --------------------------------------------------------------------
function mn_manual_de_usuario_Callback(hObject, eventdata, handles)
% hObject    handle to mn_manual_de_usuario (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 open('MANUAL DE USUARIO.pdf');

% --------------------------------------------------------------------
function mn_gui_de_practicas_Callback(hObject, eventdata, handles)
% hObject    handle to mn_gui_de_practicas (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 open('GUIA DE PRACTICAS.pdf');
```

PROGRAMA DE ADQUISICIÓN DE DATOS

Elaborado por: Jonathan Tapia




```
function varargout = adquirir_datos(varargin)
% ADQUIRIR_DATOS MATLAB code for adquirir_datos.fig
%      ADQUIRIR_DATOS, by itself, creates a new ADQUIRIR_DATOS or raises
the existing
%      singleton*.
%
%      H = ADQUIRIR_DATOS returns the handle to a new ADQUIRIR_DATOS or
the handle to
%      the existing singleton*.
%
%      ADQUIRIR_DATOS('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in ADQUIRIR_DATOS.M with the given input
arguments.
%
%      ADQUIRIR_DATOS('Property','Value',...) creates a new
ADQUIRIR_DATOS or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before adquirir_datos_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to adquirir_datos_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help adquirir_datos
```

```
 % Last Modified by GUIDE v2.5 08-Apr-2013 09:32:05

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @adquirir_datos_OpeningFcn, ...
                   'gui_OutputFcn',  @adquirir_datos_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before adquirir_datos is made visible.
function adquirir_datos_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to adquirir_datos (see VARARGIN)
    global muestras periodo_de_muestreo y
    y='';
    muestras=500;
    % INICIALIZAR VENTANA GUI
    set(handles.btn_guardar_datos,'Visible','off');
    set(handles.txt_limite_inferior,'String','0');
    set(handles.txt_limite_superior,'String','5');
    set(handles.txt_entrada,'String','5');

    periodo_de_muestreo=1/muestras;
    set(handles.txt_display_tiempo,'String','1');

% Choose default command line output for adquirir_datos
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes adquirir_datos wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = adquirir_datos_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: delete(hObject) closes the figure
delete(hObject);

% ------------------------------------------------------------------------
% ---------------------------- GUI ---------------------------------------
% ------------------------------------------------------------------------
function txt_display_tiempo_Callback(hObject, eventdata, handles)
% hObject    handle to txt_display_tiempo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    global muestras periodo_de_muestreo

    set(handles.rdbtn_minimo,'Value',0);
    set(handles.rdbtn_maximo,'Value',0);
    tiempo=str2double(get(handles.txt_display_tiempo,'String'));
    if isnan(tiempo)
    else
        if tiempo<1
            tiempo=1;
        elseif tiempo>1800
            tiempo=1800;
        end
        set(handles.txt_display_tiempo,'String',num2str(tiempo));
        tiempo=tiempo/muestras;
        periodo_de_muestreo=tiempo;
    end
% Hints: get(hObject,'String') returns contents of txt_display_tiempo as
text
%        str2double(get(hObject,'String')) returns contents of
txt_display_tiempo as a double


% --- Executes during object creation, after setting all properties.
function txt_display_tiempo_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txt_display_tiempo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in rdbtn_minimo.
```

```
function rdbtn_minimo_Callback(hObject, eventdata, handles)
% hObject    handle to rdbtn_minimo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    global muestras periodo_de_muestreo

    set(handles.rdbtn_maximo,'Value',0);

    tiempo=1;
    set(handles.txt_display_tiempo,'String',num2str(tiempo));
    tiempo=tiempo/muestras;
    periodo_de_muestreo=tiempo;
% Hint: get(hObject,'Value') returns toggle state of rdbtn_minimo


% --- Executes on button press in rdbtn_maximo.
function rdbtn_maximo_Callback(hObject, eventdata, handles)
% hObject    handle to rdbtn_maximo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    global muestras periodo_de_muestreo

    set(handles.rdbtn_minimo,'Value',0);

    tiempo=1800;
    set(handles.txt_display_tiempo,'String',num2str(tiempo));
    tiempo=tiempo/muestras;
    periodo_de_muestreo=tiempo;

% Hint: get(hObject,'Value') returns toggle state of rdbtn_maximo

function txt_limite_superior_Callback(hObject, eventdata, handles)
% hObject    handle to txt_limite_superior (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of txt_limite_superior as
text
%        str2double(get(hObject,'String')) returns contents of
txt_limite_superior as a double


% --- Executes during object creation, after setting all properties.
function txt_limite_superior_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txt_limite_superior (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function txt_limite_inferior_Callback(hObject, eventdata, handles)
% hObject    handle to txt_limite_inferior (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of txt_limite_inferior as
text
%        str2double(get(hObject,'String')) returns contents of
txt_limite_inferior as a double


% --- Executes during object creation, after setting all properties.
function txt_limite_inferior_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txt_limite_inferior (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in btn_ingresar_limites.
function btn_ingresar_limites_Callback(hObject, eventdata, handles)
% hObject    handle to btn_ingresar_limites (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    global y y_aux y_fil time Periodo_real relacion
    % ' [ ]
    lim_inf=str2double(get(handles.txt_limite_inferior,'String'));
    lim_sup=str2double(get(handles.txt_limite_superior,'String'));
    if(or(isnan(lim_inf),isnan(lim_sup)))
        errordlg('Ingrasar valores reales','ERROR');
    else
        y=lim_inf+(lim_sup-lim_inf)*(y_aux)/relacion;
        plot(0,0); hold on
        plot(time,y); hold off
        title(strcat('Periodo: ',num2str(roundn(Periodo_real,-4)),'
seg'));
        set(handles.rdbtn_filtar_datos,'Value',0);
        y_fil=y;
    end

% --- Executes on button press in btn_limites_defecto.
function btn_limites_defecto_Callback(hObject, eventdata, handles)
% hObject    handle to btn_limites_defecto (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    global y y_aux y_fil time Periodo_real relacion
    % ' [ ]
    set(handles.txt_limite_inferior,'String','0');
    set(handles.txt_limite_superior,'String','5');
    lim_inf=0;
```

```
    lim_sup=5;
    y=lim_inf+(lim_sup-lim_inf)*(y_aux)/relacion;
    plot(0,0); hold on
    plot(time,y); hold off
    title(strcat('Periodo: ',num2str(roundn(Periodo_real,-4)),' seg'));
    set(handles.rdbtn_filtar_datos,'Value',0);
    y_fil=y;


% --- Executes on button press in rdbtn_filtar_datos.
function rdbtn_filtar_datos_Callback(hObject, eventdata, handles)
% hObject    handle to rdbtn_filtar_datos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    % SELECCION PARA ACTIVAR FILTRADO DE DATOS
% Hint: get(hObject,'Value') returns toggle state of rdbtn_filtar_datos
    global y time Periodo_real y_fil

    if isnan(y)==0
        filtrar=get(handles.rdbtn_filtar_datos,'Value');
        y=y_fil;
        plot(time,y,'b'); % Grafica valores antes de filtrar
        switch filtrar
            case 1
                plot(0,0); hold on
                fmuestreo=1/Periodo_real;
                fcorte=10*fmuestreo;
                [b,a]=butter(1,fmuestreo/fcorte);
                y=filter(b,a,y);
                plot(time,y,'b'); hold off   % Grafica valores filtrados
        end
        title(strcat('Periodo: ',num2str(roundn(Periodo_real,-4)),'
seg'));
        hold off
    end


    % --- Executes on button press in btn_adquirir_datos.
function btn_adquirir_datos_Callback(hObject, eventdata, handles)
% hObject    handle to btn_adquirir_datos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    global y time Periodo_real periodo_de_muestreo muestras
    global y_aux y_fil stop relacion

    relacion=255; %255 ó 1023;
    set(handles.btn_guardar_datos,'Visible','on');
    clc
    stop=0;
    % Eleccion del metodo de adquicion de datos
    texto=get(handles.btn_adquirir_datos,'String');
    if(isequal(texto,'Adquirir Datos'))
        set(handles.btn_adquirir_datos,'String','Detener');
        t_muestreo=periodo_de_muestreo*muestras;

[y,time,Periodo_real]=DAQ_adquirir_datos_funcion(periodo_de_muestreo,t_mu
estreo);
        y_aux=y;
```

```
        lim_inf=0;
        lim_sup=5;
        y=lim_inf+(lim_sup-lim_inf)*(y_aux)/relacion;
        y_fil=y;
        if stop==0
            plot(0,0); hold on
            plot(time,y); hold off
            title(strcat('Periodo: ',num2str(roundn(Periodo_real,-4)),'
seg'));
            set(handles.rdbtn_filtar_datos,'Value',0);
        end
        set(handles.btn_adquirir_datos,'String','Adquirir Datos');
        % Condiciones iniciales de metodo eliminar
        set(handles.rdbtn_eliminar,'String','Eliminar datos');
        set(handles.rdbtn_eliminar,'Value',0);
    else
        set(handles.btn_adquirir_datos,'String','Adquirir Datos');
        stop=1;
    end

% --- Executes on button press in btn_agrandar_imagen.
function btn_agrandar_imagen_Callback(hObject, eventdata, handles)
% hObject    handle to btn_agrandar_imagen (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    global time y %y_fil
    % y       contiene los valores actuales, ya sea filtrados o nó
    % y_fil   contiene los valores de y sin filtrar
    clc
    if isequal(y,'')==0
        figure
        plot(0,0); hold on; grid on
        plot(time,y);
        hold off
    end

function txt_entrada_Callback(hObject, eventdata, handles)
% hObject    handle to txt_entrada (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of txt_entrada as text
%        str2double(get(hObject,'String')) returns contents of
txt_entrada as a double


% --- Executes during object creation, after setting all properties.
function txt_entrada_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txt_entrada (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in btn_guardar_datos.
function btn_guardar_datos_Callback(hObject, eventdata, handles)
% hObject    handle to btn_guardar_datos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    global y x Periodo_real

    amplitud_x=str2double(get(handles.txt_entrada,'String'));
    if isnan(amplitud_x)
        errordlg('Ingrese el valor de la entrada correctamente','ERROR');
        set(handles.txt_entrada,'String','5');
    else
        clc;
        x=amplitud_x*ones(size(y,1),1);
        periodo=Periodo_real*ones(size(y,1),1);
        planta=[x y periodo];
        uisave('planta');
    end

function txt_eliminar_inf_Callback(hObject, eventdata, handles)
% hObject    handle to txt_eliminar_inf (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of txt_eliminar_inf as
text
%        str2double(get(hObject,'String')) returns contents of
txt_eliminar_inf as a double


% --- Executes during object creation, after setting all properties.
function txt_eliminar_inf_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txt_eliminar_inf (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function txt_eliminar_sup_Callback(hObject, eventdata, handles)
% hObject    handle to txt_eliminar_sup (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of txt_eliminar_sup as
text
%        str2double(get(hObject,'String')) returns contents of
txt_eliminar_sup as a double

 % --- Executes during object creation, after setting all properties.
```

```
function txt_eliminar_sup_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txt_eliminar_sup (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in rdbtn_eliminar.
function rdbtn_eliminar_Callback(hObject, eventdata, handles)
% hObject    handle to rdbtn_eliminar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    global y time y_eli time_eli
    global y_aux y_fil relacion
    clc
    activado=get(handles.rdbtn_eliminar,'Value');
    if activado==1
        % Obtencion de valores
        t_inf=str2double(get(handles.txt_eliminar_inf,'String'));
        t_sup=str2double(get(handles.txt_eliminar_sup,'String'));
%

        if or( or( isnan(t_inf),isnan(t_sup) ), isequal(y,'') )
            % Valores ingresados incorrectos
            errordlg('Ingrese valores correctos','ERROR');
            set(handles.rdbtn_eliminar,'Value',0);
        elseif  and(time(1)<=t_inf,t_inf<t_sup)

            if time(size(time,1),1)<t_sup
                t_sup=time(size(time,1),1);
                set(handles.txt_eliminar_sup,'String',num2str(t_sup));
            end

            % Guarda los valores anteriores
            y1=y;  % y_fil
            y_eli=y1;
            time_eli=time;

            % Selecciona y muestra los valores a eliminar
            figure
            plot(time,y1); hold on; grid on
            x_t=[t_inf t_inf t_sup t_sup]';
            y_t=[min(y1) max(y1) max(y1) min(y1)]';
            plot(x_t,y_t,'g');
            hold off;

            eliminar = questdlg('Desea eliminar esta parte de datos?',...
                     'CONFIRMACIÓN',...
                     'Si','No','Si');
            switch (eliminar)
                case 'Si' % Elimina y grafica los valores restantes
```

```
                    [y,time]=eliminar_datos(y1,time,t_inf,t_sup);

                    plot(handles.axes2,0,0); hold on;
                    plot(handles.axes2,time,y);

                    hold on; grid on;
                    plot(0,0);
                    plot(time,y,'r');


                    y_fil=y;

lim_inf=str2double(get(handles.txt_limite_inferior,'String'));

lim_sup=str2double(get(handles.txt_limite_superior,'String'));
                    y_aux=round(relacion*(y-lim_inf)/(lim_sup-lim_inf));

                    set(handles.rdbtn_eliminar,'String','Recuperar
datos');

                case 'No' % No realiza ninguna accion
                    close;
                    set(handles.rdbtn_eliminar,'Value',0);
                    return
            end
        else
            errordlg('Límites incorrectos','ERROR');
            set(handles.rdbtn_eliminar,'Value',0);
        end
    else
        % Recupera los valores eliminados
        y=y_eli;
        y_fil=y;
        lim_inf=str2double(get(handles.txt_limite_inferior,'String'));
        lim_sup=str2double(get(handles.txt_limite_superior,'String'));
        y_aux=round(relacion*(y-lim_inf)/(lim_sup-lim_inf));
        time=time_eli;
        plot(handles.axes2,time,y);
        set(handles.rdbtn_eliminar,'String','Eliminar datos');
    end
% Hint: get(hObject,'Value') returns toggle state of rdbtn_eliminar
```

SUBRUTINA DE ADQUISICIÓN DE DATOS

Elaborado por: Jonathan Tapia

```
function
[y_planta,t_planta,T_real]=DAQ_adquirir_datos_funcion(T_muestreo,t_muestr
eo)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % y_planta     ===> datos de la respuesta de la planta
    % t_planta     ===> vector de datos del tiempo en relacion con "y"
    % T_real  ===> periodo de muestreo obtenido por la máquina
    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % T_muestreo===> periodo de muestreo definido por el usuario
```

```
    % t_muestreo===> tiempo máximo para muestrear
    % ylim_sup  ===> Valor superior en realcion a la entrada
    % ylim_inf  ===> Valor inferior en realcion a la entrada
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%COMUNICACION CON LA TARJETA%%%%%%%%%%%%%%%%%%%%%%%%%
    % data_out(1)==> Activar la salida de la planta
    % data_out(2)==> Eleccion del canal de lectura analogico ADC
    % data_out(3)==> Eleccion de salida PWM
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % data_in(1)==> Estado del "PULSADOR" de la DAQ
    % data_in(2)==> Estado de entradas digitales de la DAQ
    % data_in(3)==> Byte bajo del ADC    % Adquisicion
    % data_in(4)==> Byte alto del ADC    %     de datos
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % El tamanio de datos de "x" e "y" deben ser iguales
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % Definicion de variables globales
    global conectado my_out_pipe my_in_pipe data_in data_out stop
    global y_aux time Periodo_real out_planta
    % Borrar espacio de trabajo
    clc
    %Inicializacion de variables requieridas
    planta=0;               % vector de la planta (datos digitales)
("planta" es vector)
    n_planta=1;             % posicion de dato de entrada
    inicio_vector=2;        % Inicio de posición para toma de datos para
modelado ("planta")

    %Adquisicion de datos
    if conectado == 1
        periodo(n_planta,1)=0;
        if T_muestreo>=0.12
            pausa=T_muestreo;
        else
            pausa=0;
        end
        % Envio de comienzo de adquisicion de datos
        data_out(1)=out_planta;
        calllib('libreria', 'MPUSBWrite',my_out_pipe, data_out,
uint8(64), uint8(64), uint8(200));
        tic
        while (and(periodo(n_planta,1)<t_muestreo , stop==0))
            % Recepcion de datos del Mircrocontrolador
            [aa,bb,data_in,dd] = calllib('libreria',
'MPUSBRead',my_in_pipe, data_in, uint8(64), uint8(64), uint8(10)); % Se
recibe el dato que envia el PIC

            %  Programe aqui
            planta(n_planta,1)=data_in(4);   % guardo dato analogo de
entrada de la planta
            planta(n_planta,2)=data_in(3);   % guardo dato analogo de
entrada de la planta
            n_planta=n_planta+1;

            calllib('libreria', 'MPUSBWrite',my_out_pipe, data_out,
uint8(64), uint8(64), uint8(10));
            %   Periodo de muestreo
```

```
            pause (pausa);   % En caso de que se desee ver la operación
más lenta
            periodo(n_planta,1)=toc;  %Tiempo real
        end
        % Envio de fin de adquisicion de datos
        data_out(1)=0;
        calllib('libreria', 'MPUSBWrite',my_out_pipe, data_out,
uint8(64), uint8(64), uint8(10));

        if(stop==1)
            y_planta=y_aux;
            t_planta=time;
            T_real=Periodo_real;
            return;  % Retorna si fue parada la adquisición de datos
        end

        % Pasar de tiempo a vector de periodo
        periodo_anterior=periodo(1,1);
        for n=2:n_planta
            periodo_actual=periodo(n,1);
            periodo(n,1)=periodo_actual-periodo_anterior;
            periodo_anterior=periodo_actual;
        end

        % Definicion del periodo de adquisicion
        periodo=periodo(inicio_vector:n_planta);
        suma_periodo=0;
        for n=1:n_planta-inicio_vector
            suma_periodo=suma_periodo+periodo(n,1);
        end
        T_real=suma_periodo/(n_planta-inicio_vector);   % Definicion de
Periodo

        % Union de 2 bytes en una sola variable digital
        y_planta=double(planta(:,1))*256+double(planta(:,2));
        y_planta=y_planta(inicio_vector:n_planta-1);
        t_planta=[0:T_real:T_real*(n_planta-1-inicio_vector)]';
    end
```

SUBRUTINA DE ELIMINACIÓN DE DATOS

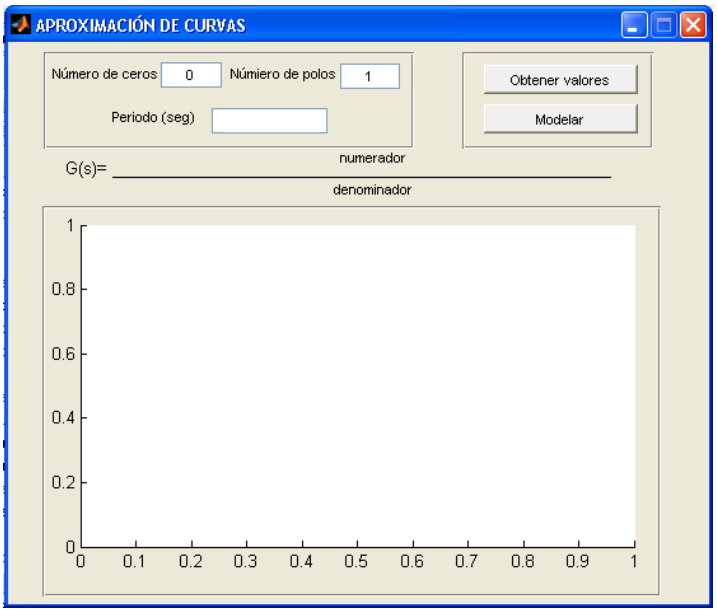
Elaborado por: Jonathan Tapia

```
function[y_muestreada,t_muestreada]=eliminar_datos(y_muestreada,t_muestre
ada,t_inf,t_sup)
    nt=size(t_muestreada,1);
    for n=1:nt
        if t_inf>=t_muestreada(n)
            n_inf=n;
        end
        if t_sup>=t_muestreada(n)
            n_sup=n;
        end
    end
```

```
    y_muestreada=[y_muestreada(1:n_inf-1,1) ;
y_muestreada(n_sup:size(y_muestreada,1),1)];
    nt=nt-n_sup+n_inf;
    t_muestreada=t_muestreada(1:nt);
```

PROGRAMA DE MODELAMIENTO DE LA SEÑAL

Elaborado por: Jonathan Tapia




```
function varargout = modelo_aproximacion(varargin)
% MODELO_APROXIMACION MATLAB code for modelo_aproximacion.fig
%      MODELO_APROXIMACION, by itself, creates a new MODELO_APROXIMACION
or raises the existing
%      singleton*.
%
%      H = MODELO_APROXIMACION returns the handle to a new
MODELO_APROXIMACION or the handle to
%      the existing singleton*.
%
%      MODELO_APROXIMACION('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in MODELO_APROXIMACION.M with the given
input arguments.
%
%      MODELO_APROXIMACION('Property','Value',...) creates a new
MODELO_APROXIMACION or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before modelo_aproximacion_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to modelo_aproximacion_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help modelo_aproximacion
```

```
% Last Modified by GUIDE v2.5 15-May-2013 14:39:44

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @modelo_aproximacion_OpeningFcn, ...
                   'gui_OutputFcn',  @modelo_aproximacion_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before modelo_aproximacion is made visible.
function modelo_aproximacion_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to modelo_aproximacion (see VARARGIN)
    set(handles.txt_orden_ceros,'String','0');
    set(handles.txt_orden_polos,'String','1');
% Choose default command line output for modelo_aproximacion
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
% UIWAIT makes modelo_aproximacion wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = modelo_aproximacion_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes during object deletion, before destroying properties.
function figure1_DeleteFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in btn_obtener_valores.
function btn_obtener_valores_Callback(hObject, eventdata, handles)
% hObject    handle to btn_obtener_valores (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    global y x
    [f,p]=uigetfile('*.mat','PLANTA DE APROXIMACION');

    if or(isequal(f,0),isequal(p,0))
    else
        clc;
        planta=open(strcat(p,f));
        planta=struct2cell(planta);
        planta=cell2mat(planta);
        x=planta(:,1);
        y=planta(:,2);

        periodo=planta(1,3);
        t=[0:periodo:periodo*(size(y,1)-1)]';
        plot(0,0); hold on;
        plot(t,y,'r*');
        set(handles.txt_periodo,'String',num2str(periodo));

        set(handles.lb_numerador,'String','LISTO');
        set(handles.lb_denominador,'String','LISTO');
        plot(0,0);
        grid on
        hold off
    end


% --- Executes on button press in btn_modelar.
function btn_modelar_Callback(hObject, eventdata, handles)
% hObject    handle to btn_modelar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    clc
    global y x

    periodo=str2double(get(handles.txt_periodo,'String'));
    n_ceros=str2double(get(handles.txt_orden_ceros,'String'));
    n_polos=str2double(get(handles.txt_orden_polos,'String'));

    if or(or(isnan(periodo),isnan(n_ceros)),isnan(n_polos))
        errordlg({'Ingresar solo numeros enteros','los decimales
seperados por punto'},'ERROR');
    elseif or(isequal(y,''),isequal(x,''))
        errordlg('Archivo no especificado','ERROR');
    else
        try
            % optencion del modelo de la planta

[lb_num,lb_den,error]=aproximar_curva(y,x,n_ceros,n_polos,periodo);
            title(strcat('Ajuste del: ',num2str(error),'% '));
            set(handles.lb_numerador,'String',lb_num);
            set(handles.lb_denominador,'String',lb_den);
```

```
        catch
            warndlg('Modelo no obtenido','AVISO');
        end
    end


function txt_orden_ceros_Callback(hObject, eventdata, handles)
% hObject    handle to txt_orden_ceros (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of txt_orden_ceros as
text
%        str2double(get(hObject,'String')) returns contents of
txt_orden_ceros as a double

% --- Executes during object creation, after setting all properties.
function txt_orden_ceros_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txt_orden_ceros (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function txt_orden_polos_Callback(hObject, eventdata, handles)
% hObject    handle to txt_orden_polos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of txt_orden_polos as
text
%        str2double(get(hObject,'String')) returns contents of
txt_orden_polos as a double

% --- Executes during object creation, after setting all properties.
function txt_orden_polos_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txt_orden_polos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function lb_numerador_CreateFcn(hObject, eventdata, handles)
% hObject    handle to lb_numerador (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns
called

% --- Executes during object creation, after setting all properties.
function lb_denominador_CreateFcn(hObject, eventdata, handles)
% hObject    handle to lb_denominador (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

function txt_periodo_Callback(hObject, eventdata, handles)
% hObject    handle to txt_periodo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    set(handles.ppmn_metodo,'Value',4);
% Hints: get(hObject,'String') returns contents of txt_periodo as text
%        str2double(get(hObject,'String')) returns contents of
txt_periodo as a double

% --- Executes during object creation, after setting all properties.
function txt_periodo_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txt_periodo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

SUBRUTINA DE APROXIMACIÓN DE CURVA

Elaborado por: Jonathan Tapia

```
function [lb_num,lb_den,error]=aproximar_curva(y,x,nc,np,periodo)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % y         ===> datos de la respuesta de la planta
    % x         ===> datos de la entrada de la planta
    % nc        ===> numero de ceros de la funcion de transferencia
    % np        ===> numero de polos de la funcion de transferencia
    % periodo   ===> periodo
    %
    % lb_num    ===> Polígono String del numerador
    % lb_den    ===> Polígono String del denominador
    % error     ===> Erros de aproximación
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % El tamanio de datos de "x" e "y" deben ser iguales
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % Aproximación a modelos continuos
    data = iddata(y,x,periodo);
    g=tfest(data,np,nc);
    % Separa polígono numerador y denominador
```

```
    num=g.num;
    den=g.den;
    % Muestra la aproximacion del la senial muestreada a la modelada
    error=g.Report.Fit.FitPercent;

    % Convierte el polígono a string
    lb_num='';
    lb_den='';

    n=size(num,2);
    for i=1:n
        if num(i)==0
            num(i)=0;
        elseif and(i==1,i==n-1)
            if i==n-1
                lb_num=strcat(num2str(num(i)),'s');
            else
                lb_num=strcat(num2str(num(i)),'s^',num2str(n-i));
            end
        elseif num(i)>0;
            if i<n-1
                lb_num=strcat(lb_num,'+',num2str(num(i)),'s^',num2str(n-
i));
            elseif i==n-1
                lb_num=strcat(lb_num,'+',num2str(num(i)),'s');
            elseif i==n
                lb_num=strcat(lb_num,'+',num2str(num(i)));
            end
        else
            if i<n-1
                lb_num=strcat(lb_num,num2str(num(i)),'s^',num2str(n-i));
            elseif i==n-1
                lb_num=strcat(lb_num,num2str(num(i)),'s');
            elseif i==n
                lb_num=strcat(lb_num,num2str(num(i)));
            end
        end
    end

    n=size(den,2);
    for i=1:n
        if den(i)==0
            den(i)=0;
        elseif i==1
            if i==n-1
                lb_den=strcat('s');
            else
                lb_den=strcat('s^',num2str(n-i));
            end
        elseif den(i)>0;
            if i<n-1
                lb_den=strcat(lb_den,'+',num2str(den(i)),'s^',num2str(n-
i));
            elseif i==n-1
                lb_den=strcat(lb_den,'+',num2str(den(i)),'s');
            elseif i==n
                lb_den=strcat(lb_den,'+',num2str(den(i)));
            end
        else
```

```
        if i<n-1
            lb_den=strcat(lb_den,num2str(den(i)),'s^',num2str(n-i));
        elseif i==n-1
            lb_den=strcat(lb_den,num2str(den(i)),'s');
        elseif i==n
            lb_den=strcat(lb_den,num2str(den(i)));
        end
    end
end
```

ANEXO 5

Mediciones de valores de componentes del módulo RC

Mediciones de valores de componentes del módulo RC

Los valores obtenidos de los componentes de resistencia fija y resistencia variable, son las siguientes:

Figura A15. Valores medidos de Resistencia

a) Valor del resistor R1

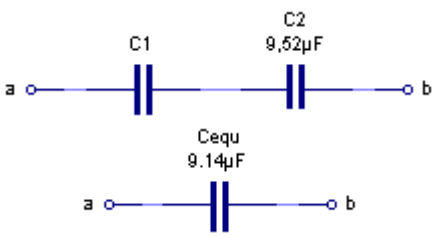
b) Valor de resistores R1 y VR1 en serie

Fuente: Autor

En el caso del condensador no se logró medir directamente su valor, entonces se midió el valor del capacitor C2 de 10uF, teniendo un valor de medición de 9.52uF; luego se conectó en serie con otro el capacitor C1 necesitado y se obtuvo un valor equivalente a 9.14uF. Véase la figura siguiente:

Figura A16. Valores medidos de capacitancia

a) Valor del capacitor C2



b) Circuito equivalente

c) Valor del capacitor equivalente Cequ

Fuente: Autor

Aplicando la fórmula de capacitores en serie se obtuvo el valor del capacitor C1=228.98uF

Ecuación A1. Fórmula de capacitancia equivalente de capacitores en serie

$$C_{equ} = \frac{C_1 C_2}{C_1 + C_2}$$

Fuente: Autor

ANEXO 6

Código fuente de conversión frecuencia a voltaje

PROGRAMA PRINCIPAL

Elaborado por: Jonathan Tapia

```
#include <.\Frecuencia_a_voltaje_complemento.h>

void main(){
  configurar_sistema();
  while(1){
    cancular_frecuencia();

    //rpm=filtrar(rpm,rpm_anterior);
    //rpm_anterior=rpm;
    frecuencia=filtrar(frecuencia,frecuencia_anterior);
    frecuencia_anterior=frecuencia;

    fvc();
  }
}
```

PROGRAMA DE SUBRUTINAS

Elaborado por: Jonathan Tapia

<<Frecuencia_a_voltaje_complemento.h>>

```
#include <16F628A.h>
//#device ADC=10                    //Descomente ésta opción en caso de usar el
ADC a 10bits
#fuses INTRC_IO, NOWDT, NOPUT, NOMCLR, NOBROWNOUT, NOLVP,
#use delay(clock=4M)

//#use rs232(baud=9600, xmit=PIN_B2,rcv=PIN_B1, stop=1, PARITY=N)
// printf("F= %7.2g Hz ",frecuencia);   // en un metodo para ver en el virtual
terminal
#BYTE PORTA=0x05
#BYTE TRISA=0x85
//A0-->I/O          //A4-->I/O
//A1-->I/O          //A5-->I
//A2-->I/O          //A6-->I/O
//A3-->I/O          //A7-->I/O
#BYTE PORTB=0x06
#BYTE TRISB=0x86
//B0_INT-->I/O       //B4-->I/O
//B1-->I/O          //B5-->I/O
//B2-->I/O          //B6-->I/O
//B3-->I/O          //B7-->I/O
```

```
#BYTE OPTION_REG=0x81
#BIT INTEDG=0x81.1
#BYTE INTCON=0x0B
#BIT GIE=0x0B.7
#BIT PEIE=0x0B.6
#BIT INTE=0x0B.4
#BIT INTF=0x0B.1

// INTCON
//BIT 7 -> GIE
//BIT 6 -> PEIE
//BIT 4 -> INTE
//BIT 1 -> INTF

//TIMER 0
#DEFINE F_PRUEBA PIN_A2
#BIT T0IE=0x0B.5
#BIT T0IF=0x0B.2

//TIMER 1
#BYTE PIR1=0x0C         //BIT's = 0 -> TMR1IF
#BIT TMR1IF=0x0C.0

#BYTE PIE1=0x8C         //BIT's = 0 -> TMR1IE
#BIT TMR1IE=0x8C.0

#BYTE TMR1L=0x0E        //BIT's = 7-0
#BYTE TMR1H=0x0F        //BIT's = 7-0
#BYTE T1CON=0X10        //BIT's = 5-0
#BIT TMR1ON=0x10.0

/*
   1bit -> entrada de frecuencia                              INT(B0) - CCP1(B3)
   1bit -> salida de frecuencia para el multiplicador de voltaje  A2-A3-A4-B1-B2
   8bit -> salida de frecuencia a voltaje (converssor R2R)        A1-A0-A7-A6-B7-B6-
B5-B4
*/
#DEFINE activar 1
#DEFINE desactivar 0
#DEFINE flanco_subida 1
#DEFINE flanco_bajada 0

int1 falg_rpm=0;                  //1->timer1_desbordado
                            //0->timer1_no_desbordado
unsigned int16 periodo=0;
float frecuencia=0;
float rpm=0;
float frecuencia_anterior=0;
float rpm_anterior=0;
```

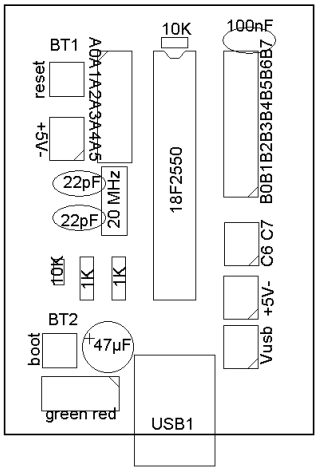
```
float fmotor=100;  // frecuencia con un diente
float fmax=125000.0;
int lagfactor=70;
int8 fre_timer0=149;  // para Frecuencia de 52.43 Hz

void configurar_sistema(){
   // Para los puertos: 1_entrada 0_salida
   //Bit   76543210
   TRISA=0b00000000;
   TRISB=0b00001001;   //B0-B3 -> Bits de entrada
   // Inicializacion de puertos en cero
   PORTA=0x00;
   PORTB=0x00;
   // Avilitar Interrupciones globales
   GIE=activar;
   PEIE=activar;
   // Configuracion de INT(B0)
   INTEDG=flanco_subida;
   INTE=activar;     //Activa metodo de interrupcion
    //Configuracion de TIMER0 para frecuencia
   SETUP_TIMER_0(RTCC_INTERNAL|RTCC_DIV_64);
   T0IE=activar;
   // Configuracion de TIMER1
   T1CON=T1_DIV_BY_8|T1_INTERNAL; //0x00;
   TMR1ON=activar;
   TMR1IE=activar;   //Activa metodo de interrupcion
}

void DAC_8BIT(int voltaje_digital){
   //A1-A0-A7-A6-B7-B6-B5-B4
   int level=0;

   level=(voltaje_digital&0b10000000)/0b10000000;
   output_bit(PIN_A1,(int1)(level));

   level=(voltaje_digital&0b01000000)/0b01000000;
   output_bit(PIN_A0,(int1)(level));

   level=(voltaje_digital&0b00100000)/0b00100000;
   output_bit(PIN_A7,(int1)(level));

   level=(voltaje_digital&0b00010000)/0b00010000;
   output_bit(PIN_A6,(int1)(level));

   level=(voltaje_digital&0b00001000)/0b00001000;
   output_bit(PIN_B7,(int1)(level));

   level=(voltaje_digital&0b00000100)/0b00000100;
   output_bit(PIN_B6,(int1)(level));
```

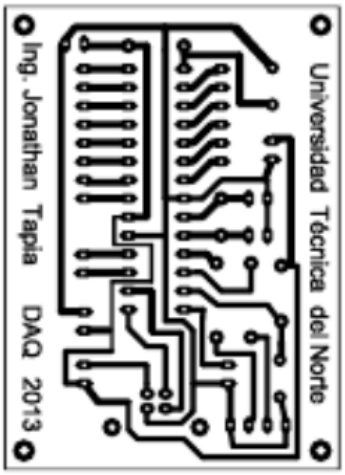
```
   level=(voltaje_digital&0b00000010)/0b00000010;
   output_bit(PIN_B5,(int1)(level));

   level=(voltaje_digital&0b00000001);
   output_bit(PIN_B4,(int1)(level));
}

void fvc(){
   int voltaje=0;
   float aux=0;
   if (frecuencia<=fmotor){
      aux=255*(frecuencia/fmotor);
      voltaje=(int)(aux);
   }
   else{
      voltaje=255;
   }
   DAC_8BIT(voltaje);
}

void cancular_frecuencia(){
   if (periodo==0){
      rpm=0;
      frecuencia=0;
   }
   else{
      //frecuencia=(Fosc)/(4*periodo)
      frecuencia=fmax/((float)(periodo));
      //rpm=(Fosc*60*2pi)/(4*2pi*n_dientes*periodo)
      //  con n_dientes=1 y Fosc=4 MHz
      //rpm=60*frecuencia
      rpm=60*frecuencia;
   }
}

float filtrar(float actual,float anterior){
   float acumulador1,acumulador2,acumulador3;

   acumulador1=(float)(actual); //Este obtiene el valor real de RMP medidas en el
reluctor
   //aqui comienza el filtro digital iir o de factor de retardo - lag factor
   acumulador2=(float)(anterior);
   acumulador3=(acumulador1-acumulador2)*(float)(lagfactor)/100;//aquí se aplica
filtro iir (50/100)=1/2-->lag factor=50 según la fórmula:
   //  Valor nuevo= valor previo+{(valor medido-valor previo)*[lagfactor/100]}
   acumulador3+=acumulador2;

   actual=acumulador3;
```

```
   return actual;
}

#INT_EXT
void entrada_frecuencia(){
   // Programe desde aqui
   //GET_TIMER1, SET_TIMER1
   if (falg_rpm==desactivar){
       periodo=get_timer1();
   }
   else{
      falg_rpm=desactivar;
      periodo=0;
   }
   set_timer1(0);
   // hasta qui
   //INTF=desactivar;
}

#INT_TIMER1
void timer1_desborde(){
   // Programe desde aqui
   falg_rpm=activar;
   periodo=0;
   set_timer1(0);
   // hasta qui
   TMR1IF=desactivar;
}

#INT_TIMER0
void timer0_desborde(){
   // Programe desde aqui
   output_toggle(F_PRUEBA);
   set_timer0(255-fre_timer0);
   // hasta qui
   T0IF=desactivar;
}
```

ANEXO 7

PCB de la DAQ

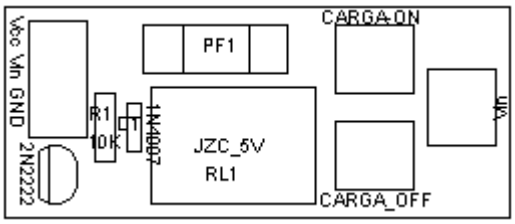

BT1
reset
+5V-
A0A1A2A3A4A5
22pF
22pF
20 MHz
10K
1K
1K
18F2550
10K
100nF
B0B1B2B3B4B5B6B7
C6 C7
+5V-
Vusb
BT2
boot
47µF
green red
USB1

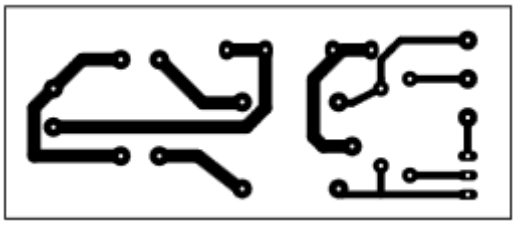

Universidad Técnica del Norte
Ing. Jonathan Tapia
DAQ 2013

ANEXO 8

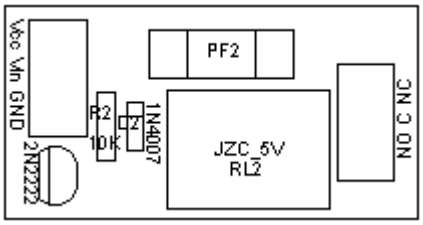
PCB circuito de interfaz de potencia

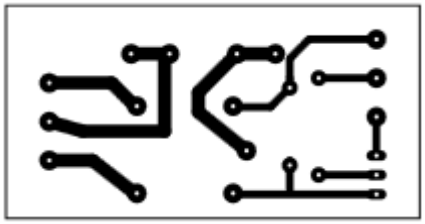
SUMINISTRADOR DE FUENTE EXTERNA





ACTUADOR POR SWITCH

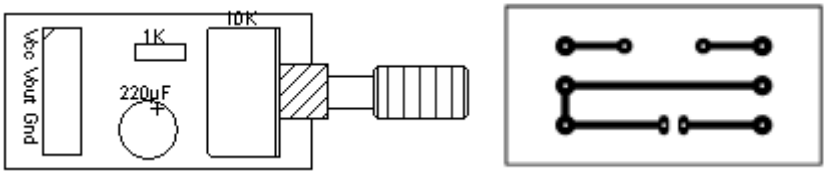




ANEXO 9

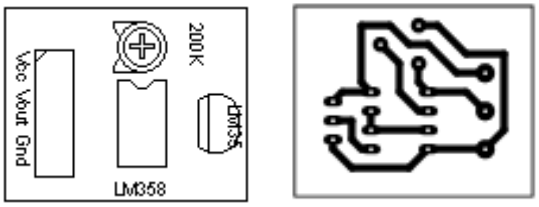
PCB circuitos varios

MÓDULO RC



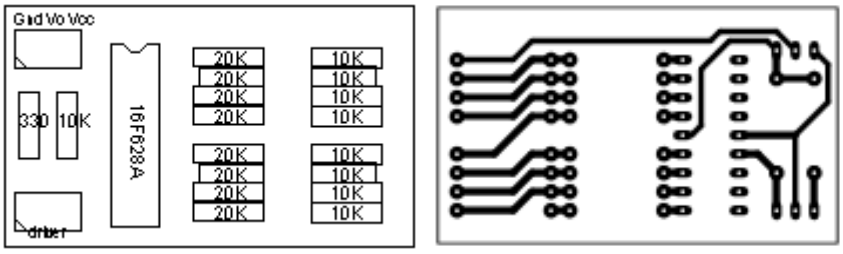
MÓDULO DE TEMPERATURA

Circuito acondicionador de sensor LM35

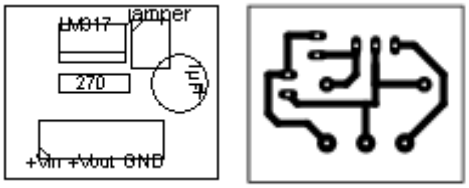


MÓDULO DE VELOCIDAD ANGULAR

Circuito acondicionador de conversión frecuencia a voltaje

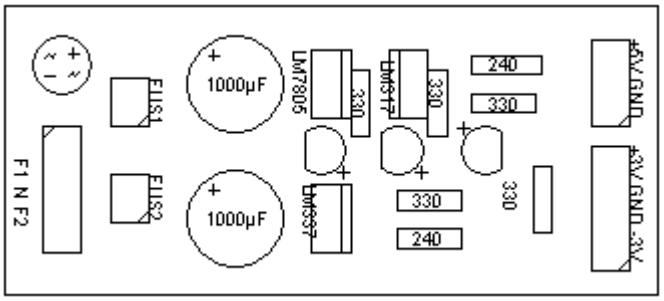


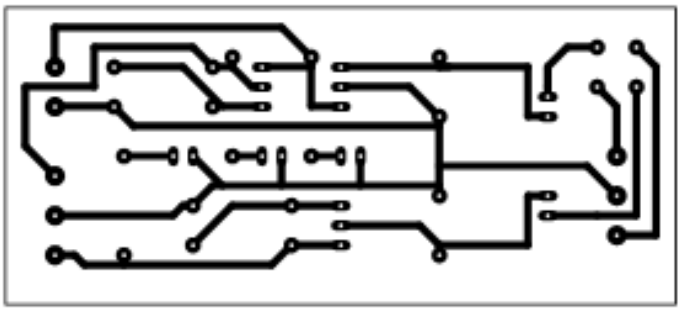
Circuito de voltaje de 1.25V



MÓDULO DE POSICIONAMIENTO ANGULAR

Circuito de fuente externa

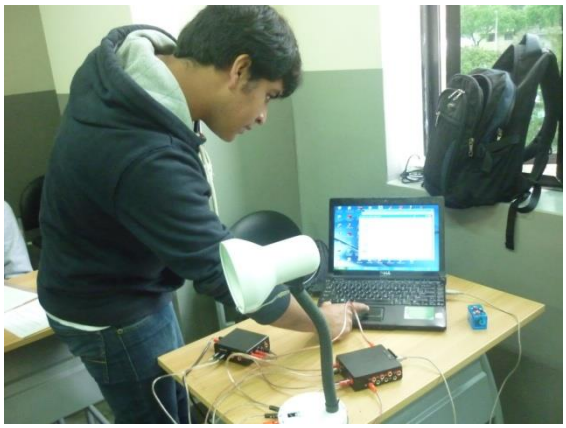


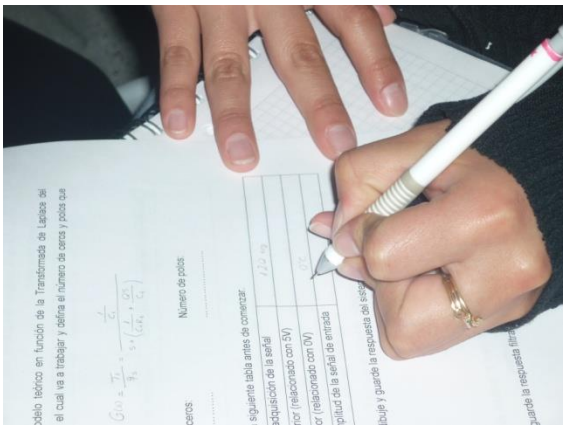


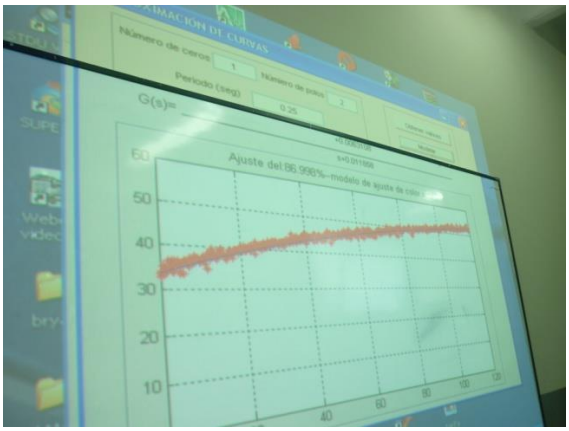
ANEXO 10

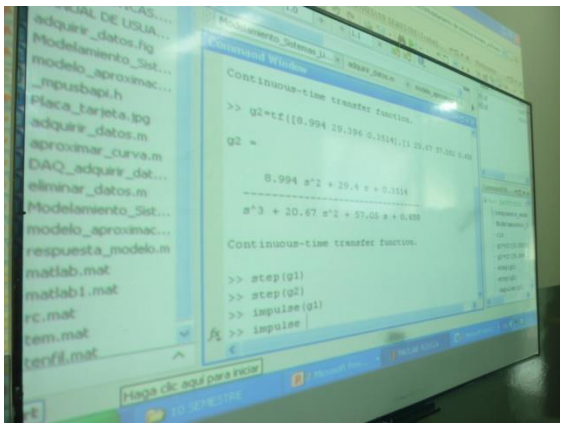
Fotografías para la validación del proyecto

VALIDACIÓN DEL PROYECTO CON EL DOCENTE Y ESTUDIANTES DE LA ASIGNATURA DE SISTEMAS DE CONTROL








Command Window
Continuous-time transfer function.
>> g2=tf([8.994 29.396 0.1514],[1 20.67 57.05 0.65
g2 =
8.994 s^2 + 29.4 s + 0.1514
s^3 + 20.67 s^2 + 57.05 s + 0.658
Continuous-time transfer function.
>> step(g1)
>> step(g2)
>> impulse(g1)
>> impulse
adquirir_datos.fig
modelo_aproximac...
Placa_tarjeta.jpg
adquirir_datos.m
aproximar_curva.m
DAQ_adquirir_dat...
eliminar_datos.m
Modelamiento_Sist...
modelo_aproximac...
respuesta_modelo.m
matlab.mat
matlab1.mat
Haga clic aquí para iniciar