ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"DISEÑO E IMPLEMENTACIÓN DE UNA PLANTA PILOTO COMO HERRAMIENTA DIDÁCTICA PARA ADQUISICIÓN Y CONTROL"

TESIS DE GRADO

Previa a la obtención del título de:

MAGISTER EN AUTOMATIZACIÓN Y CONTROL INDUSTRIAL

Presentado por:

VÍCTOR MANUEL ASANZA ARMIJOS

Guayaquil – Ecuador

2013

AGRADECIMIENTO

A mi familia, quienes han estado conmigo en todo momento. Gracias por ser mis padres, mis consejeros y mis amigos; por darme amor, confianza y comprensión.

A las autoridades de mi facultad, quienes supieron direccionarme a lo largo de mis estudios de postgrado; a mi director de tesis, quien con sus sabias observaciones me supo dirigir en el desarrollo de la presente tesis; a mis colegas docentes de la ESPOL, personal administrativo y de servicio, quienes han sido un ejemplo de cortesía en los trámites de presentación del presente trabajo de tesis.

A todos aquellos que en el camino de la vida han sabido ganarse mi aprecio, estima y respeto, mis amigos. Porque ellos han sabido enseñarme entre tantas lecciones que las cosas a medias no sirven.

DEDICATORIA

Este proyecto de tesis lo dedicado con mucho cariño a mi hijo Jean Asanza, quien es la más grande bendición y la mejor inspiración que Dios me ha dado para alcanzar mis metas en la vida, entre las cuales está el ser un padre ejemplar en esta nueva faceta de mi vida. *"Hijo mío cuando estés más grande y leas estas líneas quiero que sepas que me hace fuerte el pensar cada día en ti; nunca dejes que nada te separe de tus objetivos en esta vida por más grandes o imposibles que parezcan de alcanzar; da lo mejor de ti cada día como si este fuese el último por vivir"*.

A mi Mamá que siempre estará en mi corazón y que desde el cielo siempre ha sabido guiar mis pasos en la vida y

me demostró una de las lecciones más importantes, que hay que darlo todo en la vida por aquellos a quienes amamos.

A mi Abuelita y tíos que en mi infancia me han sabido formar y me han hecho ver que lo más importante es la familia, ese amor que es incondicional es el que nos fortalece en los momentos difíciles y nos permite levantarnos y seguir luchando.

A mis primos que me han sabido dar su apoyo y ánimos cual si fuesen hermanos, enseñándome así que la familia va mucho más allá de apellidos, se trata de dar afecto y respeto al prójimo.

TRIBUNAL DE SUSTENTACIÓN

Ph. D. Boris Vintimilla Burgos	M.Sc. Carlos Salazar López
SUB-DECANO DE LA FACULTAD FIEC	**DIRECTOR DE TESIS**
M. Sc. Holger Cevallos U.	M. Sc. Carlos Valdivieso A.
MIEMBRO PRINCIPAL	**MIEMBRO SUPLENTE**

DECLARACIÓN EXPRESA

"La responsabilidad por los hechos, ideas y doctrinas expuestas en este proyecto me corresponden exclusivamente, y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL"

_____

VÍCTOR MANUEL ASANZA ARMIJOS

RESUMEN

Este proyecto consiste en el desarrollo de distintas guías de prácticas usando la "PLANTA DE ADQUISICIÓN Y CONTROL" en conjunto con el software MATLAB (abreviatura de MATrix LABoratory, "laboratorio de matrices"), MATLAB es un software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio disponible en la plataforma Windows.

Este proyecto presenta el uso de los toolbox necesarios para realizar cada práctica y además se indican los detalles del hardware involucrado.

En el Capítulo 1, se realiza el análisis del problema donde se identifica, define y descompone el problema a tratar, cuales son las principales antecedentes, causales y efectos del problema.

En el Capítulo 2, se realiza el análisis de herramientas y conocimientos disponibles, aquí revisaremos los conocimientos y conceptos teóricos que disponemos para dar solución al tema planteado. Además revisaremos las herramientas de software y hardware disponibles en el mercado que se usarán en el desarrollo de la tesis.

En el Capítulo 3, se realiza el diseño de la solución, aquí explicaremos como se resolverá el problema. Se harán los diseños de las distintas prácticas a ser realizadas respetando los objetivos planteados. También se hará el diseño físico de como estarán ubicadas los distintos subsistemas dentro de la planta de adquisición y control.

En el Capítulo 4, se realiza la Implementación, aquí veremos cómo pasar de los diseños propuestos a una implementación real del sistema, especificaremos que requerimientos de hardware y software son necesarios para poder desarrollar las aplicaciones. Se desarrollará las guías de prácticas, manual de uso de la planta, códigos de MATLAB y MIKROBASIC (MikroBasic PRO para PIC es un compilador BASIC con todas las características para microcontroladores PIC de Microchip).

En el Capítulo 5, se realiza las pruebas, aquí explicaremos la lista de prácticas a realizar con la planta de adquisición y control. Se indicará la metodología seguida para realizar cada prueba, incluyendo los resultados experimentales y simulaciones.

Finalmente se presentan las conclusiones y sus recomendaciones.

OBJETIVOS:

a) Realizar prácticas de las diferentes técnicas de ADQUISICIÓN DE DATOS usando microcontroladores como hardware y MATLAB como software siendo de mucho provecho para los estudiantes.
b) Ejecutar prácticas de los conocimientos teóricos de IDENTIFICACIÓN DE SISTEMAS para hallar la ecuación característica de los diferentes sistemas que vienen en la PLANTA DE ADQUISICIÓN Y CONTROL.
c) Hacer los diseños de los diferentes tipos de control en MATLAB usando las distintas herramientas ToolBox y en simulink poder comprobar su eficiencia

previa a su implementación en hardware del controlador usando microcontroladores.

d) Adquirir los datos ya en lazo cerrado con el controlador y poder así verificar la aproximación de los resultados reales a los resultados de las simulaciones.

METODOLOGÍA:

Las estrategias planteadas servirán para implementar los proyectos propuestos:

a) Describir el problema a resolver, se analiza la situación actual del problema.
b) Analizaremos las herramientas disponibles entre ellos recursos teóricos, software, hardware.
c) Se diseñará la solución, con las herramientas analizadas que se utilizarán y con ello daremos solución al problema analizado.
d) Se desarrollarán prácticas y pruebas que nos permitirán evaluar la solución con los objetivos planteados.
e) Finalmente se realizará la implementación de la solución y analizaremos los resultados con las prácticas desarrolladas.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

GENERALIDADES

a) El elaborar una herramienta académica para complementar ciertos conocimientos teóricos será de interés tanto de profesores y alumnos de electrónica.

b) Guías de prácticas debidamente documentadas ayudará a los estudiantes a entender las aplicaciones del desarrollo teórico con MATLAB.

c) El usar a MATLAB como software de adquisición de datos, posteriormente el análisis y aplicación conceptos fundamentales de la teoría de control a esos datos adquiridos, le darán mayor aplicabilidad en el mundo real a esos conocimientos.

d) La robusta plataforma en hardware de desarrollo y de aplicación de la empresa ecuatoriana IDETEC CIA. LTDA. facilitará enormemente la integración de varias plantas o sistemas en una sola herramienta.

ALCANCE

El uso de una herramienta que permita a los estudiantes de electrónica a reforzar los conocimientos teóricos en aplicaciones académicas en aula de clase.

Esta herramienta tendrá valiosas aplicaciones en prácticas como:

- Adquisición de datos.
- Identificación de sistemas.
- Diseño de controladores.
- Aplicación del controlador.

El desarrollo de distintas guías de prácticas que harán uso de la herramienta llamada "PLANTA DE ADQUISICIÓN Y CONTROL" en conjunto con el software MATLAB, le ayudará a los estudiantes a aprovechar los toolbox que dispone MATLAB como complemento al aprendizaje de ciertos conocimientos teóricos.

Esta planta piloto contará con 2 subsistemas:

- Planta con un motor DC.
- Planta con una fuente generadora de calor.

DESCRIPCIÓN DEL PROBLEMA A RESOLVER

Lo que se busca es dar una herramienta que permita realizar prácticas de adquisición de datos de un proceso de un motor DC con encoder óptico, un proceso con generador de calor y sensor de temperatura, un proceso con generador de luz y sensor de intensidad lumínica.

Los sensores de cada proceso brindan la información necesaria para medir el comportamiento de la planta. Esta información se la digitaliza y procesa con la ayuda del software MATLAB.

Las plantas nos permiten utilizar datos reales para las prácticas que se desarrollan en los capítulos siguientes presentados en el presente proyecto, permitiéndonos utilizar los conceptos teóricos estudiados.

SITUACIÓN ACTUAL DEL PROBLEMA A RESOLVER

En la actualidad, se busca aportar el aprendizaje usando plantas académicas que nos brinden datos reales para poder aplicar los conocimientos teóricos aprendidos.

La aplicación práctica de los conceptos teóricos muchas veces en la industria implica el uso de: maquinarias, procesos de automatización, equipos costosos que agregado al consumo de materia prima implica riesgos costosos para hacer pruebas de conceptos teóricos aprendidos.

El objetivo del presente trabajo consiste en el desarrollo de una herramienta accesible al estudiante y que, agregando prácticas que se realicen con esta estación de trabajo básica, los cuales contienen tres plantas con las que se podrán obtener datos y analizarlos aplicando el contenido teórico de control.

CAPÍTULO 1

ANÁLISIS DE HERRAMIENTAS Y CONOCIMIENTOS DISPONIBLES

1.1 RECURSOS TEÓRICOS DISPONIBLES

Dentro de los recursos disponibles cabe recalcar la importancia del hardware y el software que en conjunto nos permiten desarrollar la solución planteada en el presente proyecto, dentro de las herramientas disponibles de hardware tenemos:

Sensores, estos elementes son transductores que convierten un fenómeno físico en una señal eléctrica medible. Dentro de los sensores que se utilizaran en el presente trabajo tenemos:

- Encoder Óptico para la planta con un motor DC.
- Sensor de temperatura para la planta con una fuente generadora de calor.
- Sensor de LDR para la planta con una fuente de luz.

Drivers, estos módulos nos permiten controlar las cargas o actuadores en cada planta. Los drivers a utilizar en el presente trabajo son:

- Módulo control de carga AC I&T.
- Módulo disparador de relé I&T.
- Módulo encoder óptico I&T.

Controlador, este módulo es el principal de todos los componentes de hardware utilizados ya que nos permite configurarlos y programarlo en función de la planta que se desee analizar. Este módulo es:

- Módulo de desarrollo MEI&T04.

Dentro del importante componente de software tenemos la herramienta de MATLAB, que con sus librerías nos ayuda enormemente la aplicación práctica en conjunto con el hardware de los conceptos teóricos aprendidos.

1.1.1 INSTRUMENTACIÓN INDUSTRIAL

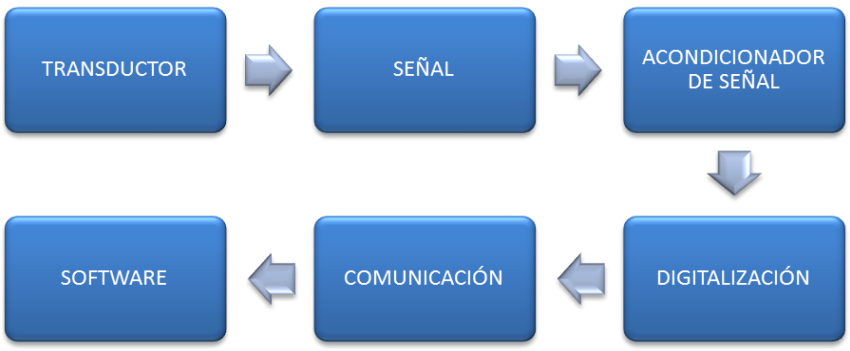

FIGURA 1.1: COMPONENTES EN LA INSTRUMENTACION

TRANSDUCTOR:

Un transductor convierte un fenómeno físico en otro tipo de energía, en especial que sea una señal eléctrica medible.

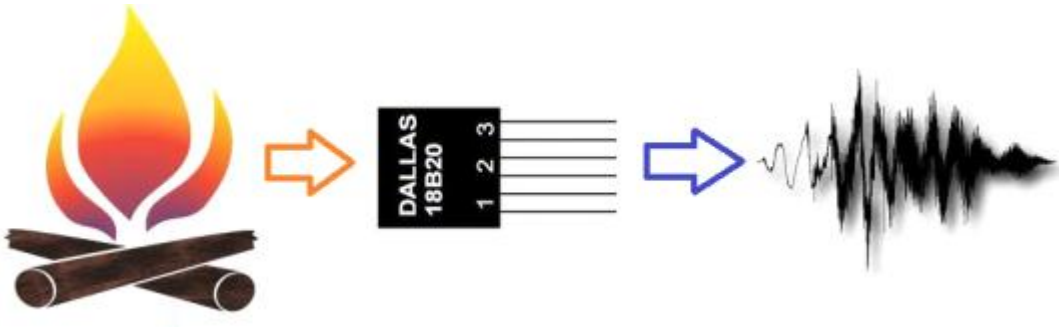

FIGURA 1.2: TRANSDUCTOR

TIPOS DE TRANSDUCTORES:

Fenómeno	Transductor
Temperatura	Termopares RTDs (Resistance Temperature Detector) Termistores
Luz	Tubos de Vacio Foto Sensores
Sonido	Micrófono
Force and Pressure	Galgas extensiométricas Celdas de carga
Posición y Desplazamiento	Potenciómetros LVDT (Linear voltage differential transformer)
Flujo	Medidor de flujo rotacional
pH	Electrodos de pH

TABLA 1.1: TIPOS DE TRANSDUCTORES

SEÑAL:

Una señal eléctrica es un tipo de señal generada por algún fenómeno electromagnético. Estas señales pueden ser analógicas, si varían de forma continua en el tiempo, o digitales si varían de forma discreta (con valores dados como 0 y 1).

SEÑAL DIGITAL:

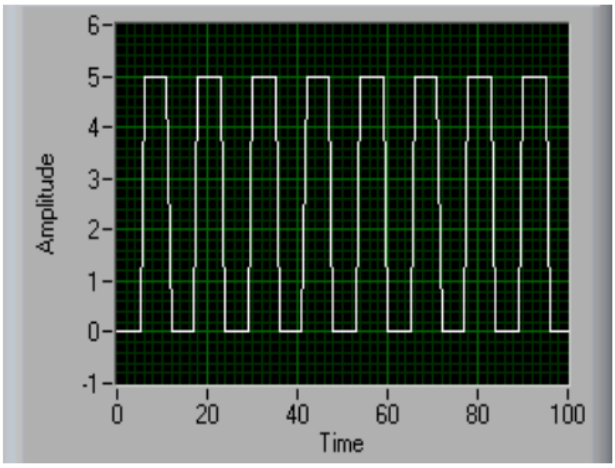

FIGURA 1.3: SEÑAL DIGITAL

La información que nos puede brindar este tipo de señales son dos valores posibles:

- Alto/On (3 - 5 Volts).
- Bajo/Off (0 – 0.8 Volts).

Información:

- Estado.
- Tasa de Cambio.
- Flanco de subida y bajada.
- Frecuencia y periodo.

SEÑAL ANALÓGICAS:

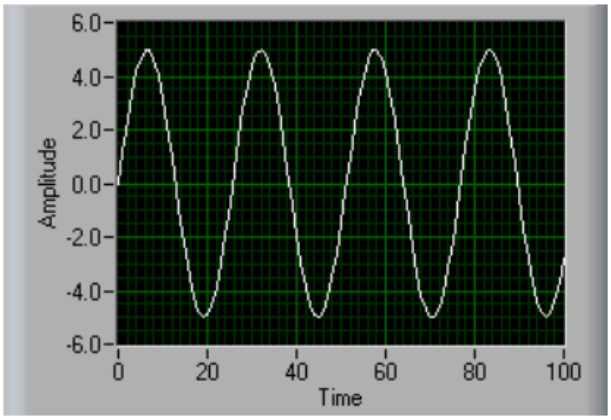

FIGURA 1.4: SEÑAL ANALÓGICA

Pueden tomar cualquier valor con respecto al tiempo.

Información:

- Nivel.
- Forma.
- Frecuencia y Periodo.

ACONDICIONADOR DE SEÑAL:

FIGURA 1.5: ACONDICIONADOR DE SEÑAL

El Ac. (Acondicionador) de Señales toma una señal que es difícil de leer por nuestro sistema DAQ y la hace más sencilla de leer. El Ac. de Señales no es siempre requerido depende del tipo de señal que se lee.

TIPOS DE ACONDICIONADOR DE SEÑAL:

Amplificación.-

- Excitación del Transductor.
- Voltaje o corriente externa aplicada al transductor.
- Aplicada por equipo de acondicionamiento de señales.

Linealización.-

- La mayoría de los transductores no se comportan de manera lineal.
- Se puede realizar en hardware o software.

Aislamiento.-

- Protege el equipo de alto voltaje.
- Utilizado en equipos con alto voltaje de modo común.

Filtrado.-

- Elimina ruido o señales no deseadas.
- Un filtro de 4 Hz elimina ruido de 60 Hz AC de señales muestreadas lentamente.
- Se puede realizar en hardware o software.

Ruido.-

Siempre que se tenga un “electrical loop of wire”, como todo los sistemas DAQ lo tienen en alguna forma, se introduce ruido eléctrico porque actúa como una radio antena.

DIGITALIZACIÓN:

Número de bits que el ADC utiliza para representar la señal, el ancho de palabra sirve para determinar cuántos cambios distintos de voltaje pueden ser medidos.

Más bits = representación más precisa de la señal

$$\#\ \text{niveles} = 2^{\text{número de bits}} = 2^{12} = 4{,}096\ \text{niveles}$$

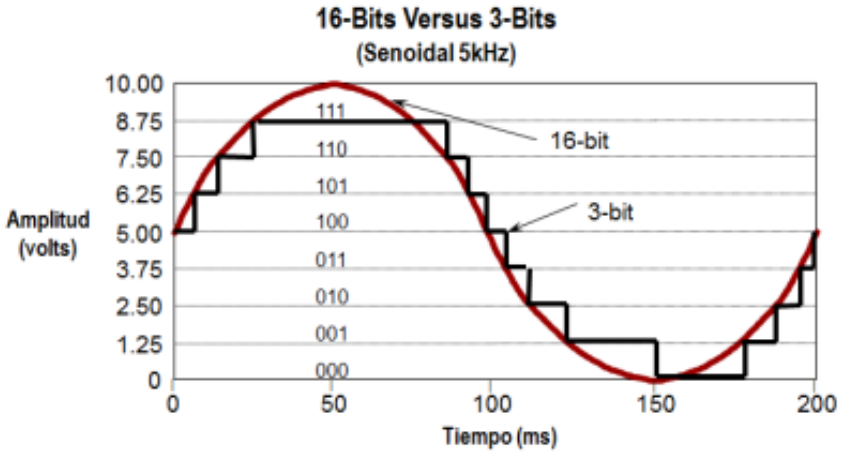

FIGURA 1.6: DIGITALIZACIÓN

RESOLUCIÓN:

La resolución se define como el cambio más pequeño de voltaje que el sistema puede detectar (depende del ancho de palabra, rango y ganancia).

$$\text{cambio mínimo} = \frac{\text{rango}}{\text{ganancia} * 2^{\text{número de bits}}}$$

Mayor resolución = representación más precisa de la señal

$$\frac{\text{rango}}{\text{ganancia} * 2^{\text{número de bits}}} = \frac{10}{1 * 2^{12}} = 2.4\ \text{mV}$$

$$\text{Incremento en el rango} = \frac{20}{1 * 2^{12}} = 4.8\ \text{mV}$$

$$\text{Incremento en la ganancia} = \frac{10}{100 * 2^{12}} = 24\ \mu\text{V}$$

FIGURA 1.7: PROCESO DE DIGITALIZACIÓN

CONSIDERACIONES AL MUESTREAR:

- La señal analógica original es continua respecto al tiempo.
- La señal muestreada es una serie de muestras discretas adquiridas a una velocidad de muestreo específica.
- A mayor velocidad de muestreo, más se parecerá la señal muestreada a la original.
- Si no se muestrea de manera correcta, un fenómeno llamado aliasing se puede presentar en la señal.

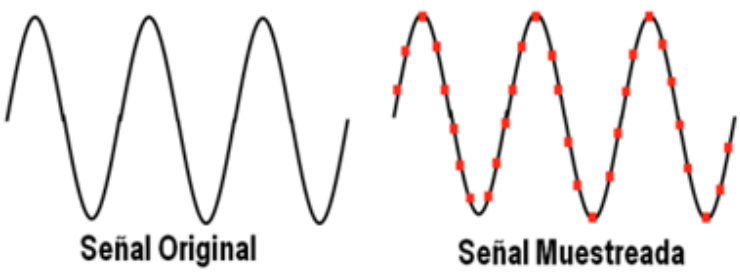

FIGURA 1.8: MUESTREO DE LA SEÑAL

ALIASING:

Es una representación incorrecta de una señal, cuyas posibles causas son:

- Ruido.
- Armónicas.

FIGURA 1.9: EFECTO ALIASING

COMUNICACIÓN:

La comunicación serial es un protocolo muy común (no hay que confundirlo con el Bus Serial de Comunicación, o USB) para comunicación entre dispositivos que se incluye de manera estándar en prácticamente cualquier computadora. La comunicación serial es también un protocolo común utilizado por varios dispositivos para instrumentación; En nuestro caso en particular la comunicación serial puede ser utilizada para adquisición de datos si se usa en conjunto con un dispositivo remoto de muestreo.

El concepto de comunicación serial es sencillo. El puerto serial envía y recibe bytes de información un bit a la vez.

Típicamente, la comunicación serial se utiliza para transmitir datos en formato ASCII. Para realizar la comunicación se utilizan 3 líneas de transmisión: (1) Tierra (o

referencia), (2) Transmitir, (3) Recibir. Debido a que la transmisión es asincrónica, es posible enviar datos por una línea mientras se reciben datos por otra.

Velocidad de transmisión (baud rate): Indica el número de bits por segundo que se transfieren, y se mide en baudios (bauds). Por ejemplo, 300 baudios representan 300 bits por segundo. Cuando se hace referencia a los ciclos de reloj se está hablando de la velocidad de transmisión.

Bits de datos: Se refiere a la cantidad de bits en la transmisión. Cuando la computadora envía un paquete de información, el tamaño de ese paquete no necesariamente será de 8 bits. Las cantidades más comunes de bits por paquete son 5, 7 y 8 bits. El número de bits que se envía depende en el tipo de información que se transfiere. Por ejemplo, el ASCII estándar tiene un rango de 0 a 127, es decir, utiliza 7 bits; para ASCII extendido es de 0 a 255, lo que utiliza 8 bits. Si el tipo de datos que se está transfiriendo es texto simple (ASCII estándar), entonces es suficiente con utilizar 7 bits por paquete para la comunicación. Un paquete se refiere a una transferencia de byte, incluyendo los bits de inicio/parada, bits de datos, y paridad. Debido a que el número actual de bits depende en el protocolo que se seleccione, el término paquete se usar para referirse a todos los casos.

Bits de parada: Usado para indicar el fin de la comunicación de un solo paquete. Los valores típicos son 1, 1.5 o 2 bits.

Paridad: Es una forma sencilla de verificar si hay errores en la transmisión serial. Existen cuatro tipos de paridad: par, impar, marcada y espaciada. La opción de no usar

paridad alguna también está disponible. Para paridad par e impar, el puerto serial fijará el bit de paridad (el último bit después de los bits de datos) a un valor para asegurarse que la transmisión tenga un número par o impar de bits en estado alto lógico.

SOFTWARE:

El software recibe los datos por comunicación serial y las almacenas para ser procesadas, es importante para ello que el software sea capaz de manejar puerto de comunicación serial.

La comunicación serial entre un dispositivo 16F628Ade Microchip con Matlab es sencilla. En esta tesis se ensayará el código necesario para la tx de datos desde Matlab para encender un led con el microcontrolador 16F628A. Para esta simulación, se configuran dos puertos virtuales usando el demo del programa Virtual Serial Port Driver v.6.

1.1.1.1 ENCODER ÓPTICO PARA LA PLANTA CON UN MOTOR DC

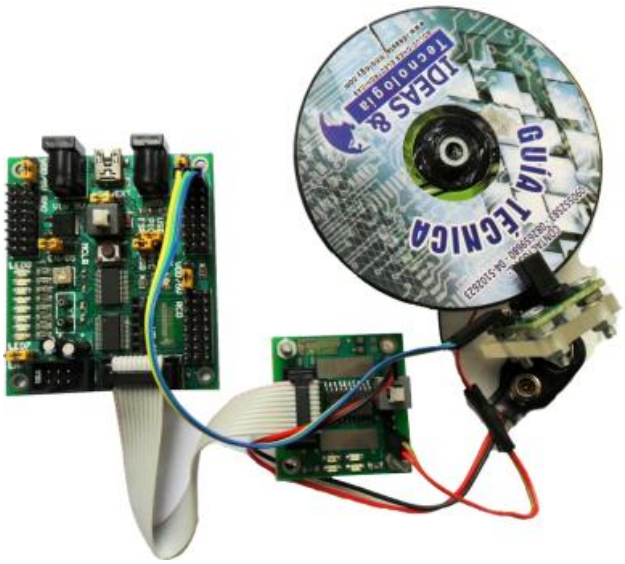

FIGURA 1.10: CONEXIÓN DEL ENCODER ÓPTICO

El encoder MOC70T3 es uno de los sensores más utilizados y de fácil adquisición.

Este sensor tiene muchas aplicaciones entre ellas:

- **Encendido Óptico**: Se lo puede usar como interruptor, usado el foto transistor y activar una bobina de un relé.
- **Tacómetro**: Haciendo uso de un disco con ranura se puede detectar cuando la luz infrarroja del diodo infrarrojo llega al foto transistor por medio de la ranura, esto nos otorga un tren de pulsos donde el medir el periodo nos da el tiempo que demora el robot del motor en dar una vuelta.

Materiales: Los foto transistores se construyen con silicio o germanio, como cualquier tipo de transistor bipolar.

La parte exterior del fototransistor está hecho de epoxy, que es una resina que permite el ingreso de radiación infrarroja hacia la base del fototransistor.

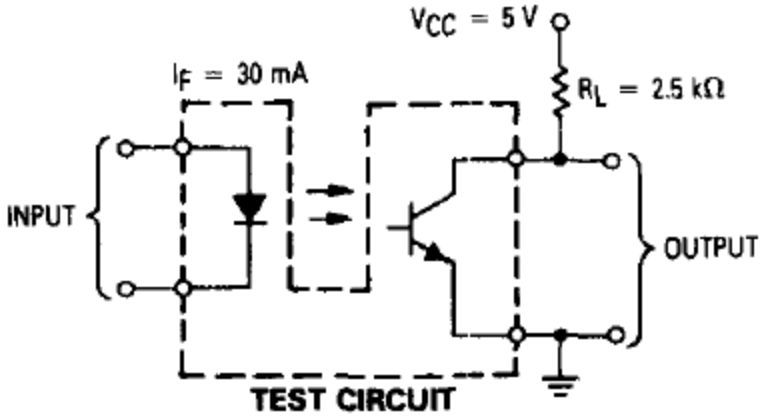

FIGURA 1.11: ENCODER ÓPTICO DE MOTOROLA SEMICONDUCTOR

Algunas de las características técnicas son:

Tipo	Fototransistor
Tipo de salida	Colector abierto NPN
Anchura de ranura	5mm
Profundidad de ranura	8mm
Tiempo de ascenso/descenso	13//17Us
Rango de temperaturas de funcionamiento	-40 a +85°C

TABLA 1.2: CARACTERÍSTICAS DEL ENCODER ÓPTICO

1.1.1.2 SENSOR DE TEMPERATURA PARA LA PLANTA CON UNA FUENTE *GENERADORA DE CALOR*

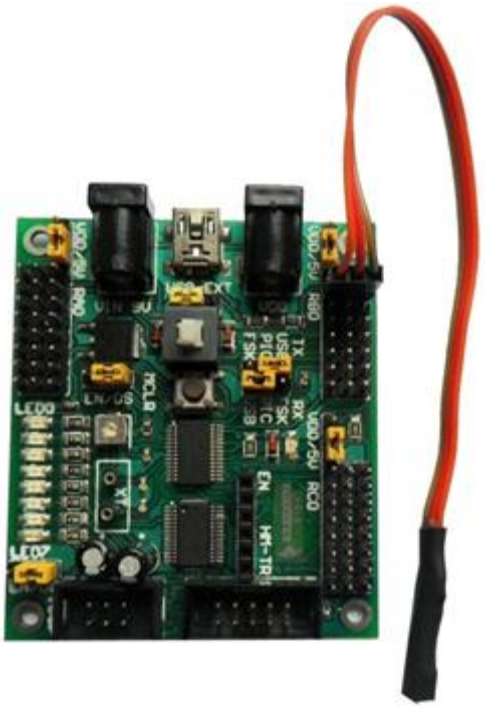

FIGURA 1.12: CONEXIÓN DEL SENSOR DE TEMPERATURA

EL termómetro DS18B20 tiene una resolución de 9-bit a 12-bit y las mediciones de temperatura son en grados Celsius. Tiene Una función de alarma programable en la memoria volátil. El DS18B20 se comunica a través de diferentes una 1-Wire ® autobús que por definición, solamente requiere una línea de Datos (vcc y tierra) para la comunicación con un microprocesador. Trabaja en un rango de -55 ° C a +125 ° C y con una exactitud de ± 0,5 ° C en el rango de -10 ° C a +85 ° C. Cada DS18B20 tiene un único código de 64 bits en serie, lo que le permite ejecutar múltiples DS18B20s en el mismo bus 1-Wire. Las Aplicaciones que pueden beneficiarse de esta característica incluyen sistemas de controles ambientales, los sistemas de monitorización de la temperatura en el interior de los edificios, la maquinaria equipos / procesos y supervisión de los sistemas de control.

El presente proyecto se utiliza el sensor de temperatura DS18B20 usando la comunicación one-wire para formar con el microcontrolador las tramas adecuadas que luego serán enviadas al software Matlab a través del puerto serial. Ver referencia [2].

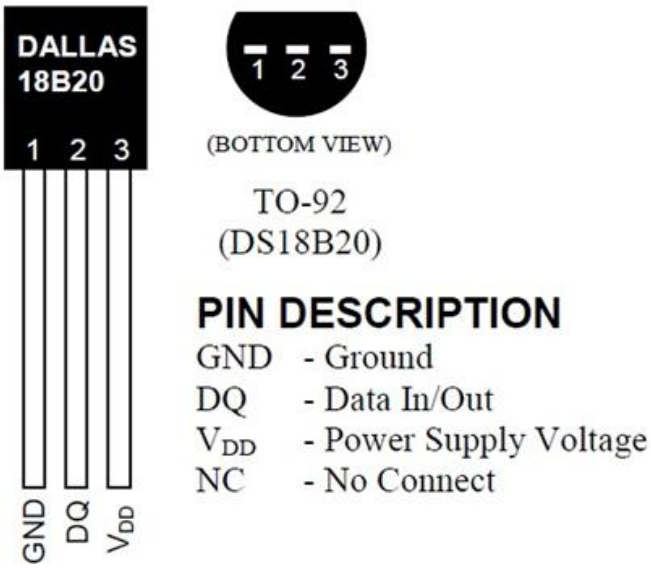

FIGURA 1.13: ENCAPSULADO Y PINES DEL DS18B20

1.1.1.3 SENSOR DE LDR PARA LA PLANTA CON UNA FUENTE DE LUZ

Una fotorresistencia es un componente electrónico cuya resistencia disminuye con el aumento de intensidad de luz incidente. Puede ser llamado foto-resistor, fotoconductor, célula fotoeléctrica o resistor dependiente de la luz "LDR". Su cuerpo está formado por una célula o celda y dos patillas. En la siguiente imagen se muestra su símbolo eléctrico.

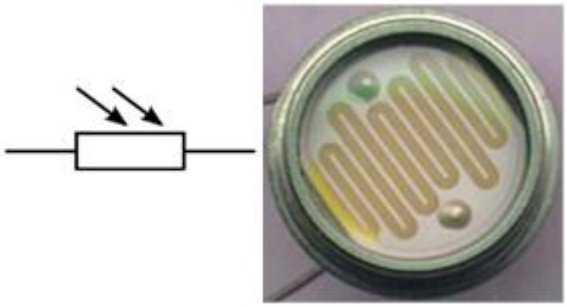

FIGURA 1.14: SENSOR LDR (LIGHT DEPENDENT RESISTOR)

El valor de resistencia eléctrica de un LDR es bajo cuando hay luz incidiendo en él (puede descender hasta 50 ohmios) y muy alto cuando está a oscuras (varios mega ohmios).

Las células de sulfuro del cadmio se basan en la capacidad del cadmio de variar su resistencia según la cantidad de luz que incide la célula.

La variación del valor de la resistencia tiene cierto retardo, diferente si se pasa de oscuro a iluminado o de iluminado a oscuro. Esto limita a no usar los LDR en aplicaciones en las que la señal luminosa varía con rapidez. El tiempo de respuesta típico de un LDR está en el orden de una décima de segundo. Esta lentitud da ventaja en algunas aplicaciones, ya que se filtran variaciones rápidas de iluminación que podrían hacer inestable un sensor (ej. tubo fluorescente alimentado por corriente alterna).

El sensor LDR está en una mini PCB que contiene el juego de resistencias que forman un divisor de tensión que en conjunto con la LDR nos da una señal analógica inversamente proporcional a la intensidad de luz incidente en la superficie de la LDR. Una vez conectado este sensor en el módulo programable, dicho sensor se energiza usando los pines (+vcc, signal, gnd).

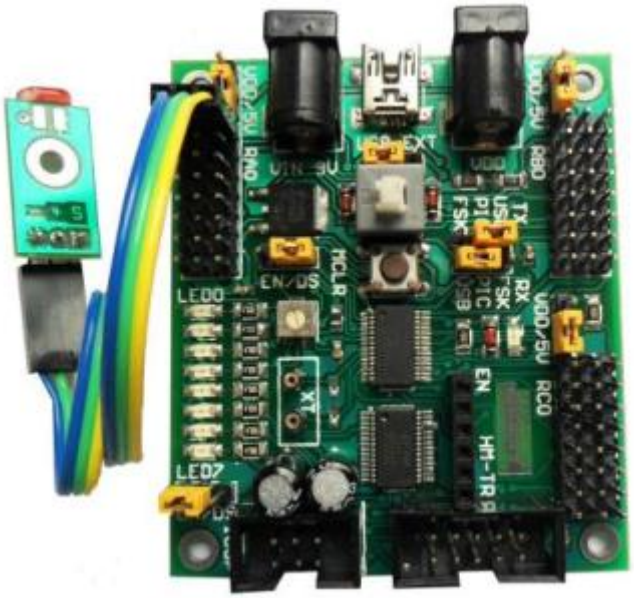

FIGURA 1.15: CONEXIÓN DEL SENSOR LDR

1.1.2 IDENTIFICACIÓN DE SISTEMAS

En aplicaciones de automatización en la industria es necesario contar con la documentación necesaria del proceso a automatizar, en muchos casos en la industria no existe esa información por motivo de que los equipos son muy antiguos o el proceso necesita un levantamiento de diagrama eléctrico y mecánico, lo cual incrementa el costo de la automatización de dicho proceso.

Una opción aceptable es utilizar la herramienta de Matlab que nos permita identificar la ecuación característica de cualquier sistema haciendo uso de medición de la señal de entrada y salida del sistema en cuestión.

Identificación de Sistemas consiste en la construcción de modelos de sistemas dinámicos (Objeto donde interactúan variables en el cual se producen variables observables), considerando datos medidos y observaciones de un sistema.

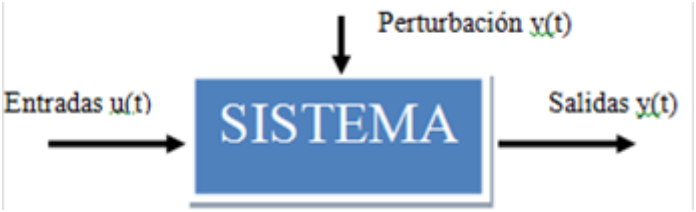

FIGURA 1.16: SISTEMA A IDENTIFICAR

Las formas de encontrar la ecuación característica de un sistemas dinámicos son: Modelación matemática e Identificación de Sistemas.

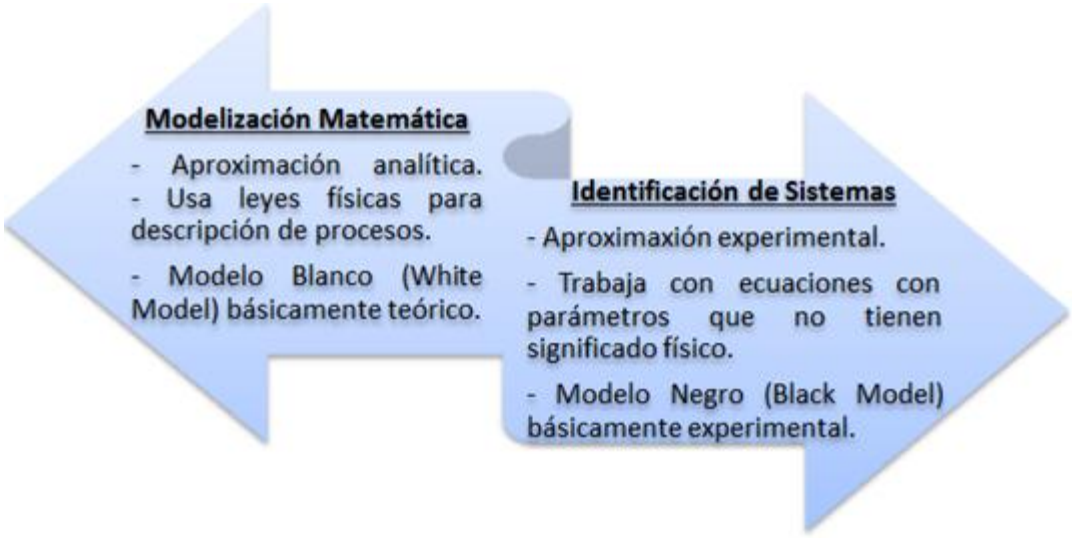

FIGURA 1.17: COMPARACIÓN ENTRE MODELIZACIÓN E IDENTIFICACIÓN

Aplicaciones de Modelo Matemática:

- Procesos Sencillos.
- Parámetros con significado físico, preciso.
- Planta no existe o no se puede hacer experimentos.

Aplicaciones de Identificación de Sistemas:

- Sistemas Complejos.
- No requiere conocimientos del proceso, fáciles de construir.
- Validez limitada.

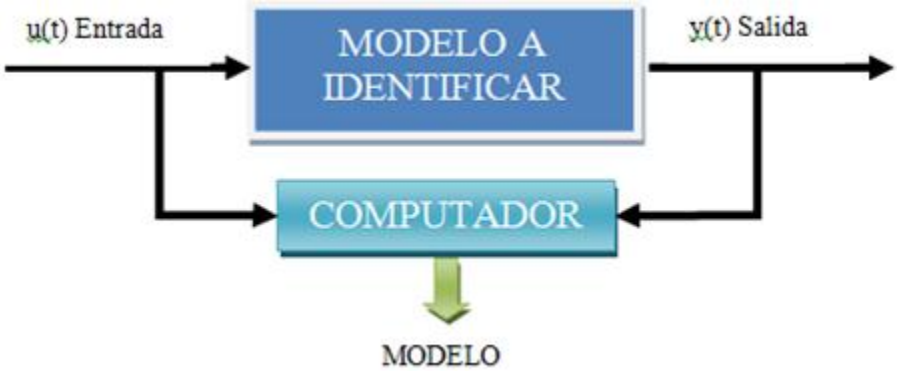

FIGURA 1.18: SISTEMA DE IDENTIFICACIÓN CON COMPUTADOR

Las formas de hacer identificación son:

- Recolectar datos y estimar parámetros (ulterior).
- Estimación de parámetros en el transcurso de la toma de datos (recursiva).
- Criterios para seleccionar las variables del diseño:
- Selección de la señal de entrada: Señal paso, señal pulso, señal ruido blanco, señal PRBS?.
- Parámetros de la señal de entrada: duración de la señal, número de datos, período de muestreo.

Pre-procesamiento de la señal: eliminación de picos, eliminación de tendencias, filtrado, eliminación de tiempos muertos, etc.

Selección de la estructura del modelo y la estimación de parámetros: AR, ARX, ARMAX, Output Error, Box-Jenkis.

Validación del modelo: Simulación, auto-correlación y correlación cruzada de los residuales, examinación de las respuestas paso, impulso y en frecuencia.

Estructura de un modelo de identificación:

- u: Señal de entrada (Aleatoria o determinística).
- v: Señal de perturbación (Aleatoria, auto-correlacionada).
- y: Señal de Salida (Aleatoria, auto-correlacionada).

$$y(k) = G(q^{-1})u(k) + H(q^{-1})e(k)$$

Señal de entrada:

- Rango de frecuencia de excitación debe ser amplio.
- No debe correlacionarse con la señal de perturbación.
- Son funciones de transferencia dadas en tiempo discreto.

$$G(q^{-1})\ y\ H(q^{-1})$$

Modelos para sistemas LTI:

Para parametrizar las funciones de transferencia lineal , tomamos las funciones de transferencia Racionales, en las cuales el numerador y el denominador son POLINOMIOS, y los coeficientes de estos polinomios, serán los parámetros (Θ) que se buscan.

$$y(k) = G(q^{-1}, \Theta)u(k) + H(q^{-1}, \Theta)e(k)$$

La relación de entrada-salida de un sistema cualquiera LTI se describe por la ecuación:

$$y(k) + a_1 y(k-1) + \cdots + a_{na} y(k-n_a) = b_1 u(k) + \cdots + b_{nb} u(k-n_b) + e(k)$$

El término de ruido blanco e(k) sirve aquí como un error directo en la ecuación diferencial y por ello se llama modelo de ecuación de error. El vector de parámetros Θ es:

$$\Theta = [a_1 a_2 \dots a_{na} \quad b_1 b_2 \dots b_{nb}]^T$$

Ecuación de error Modelo AR (Autoregresive):

$$A(q^{-1})y(k) = e(k)$$

$$A(q^{-1}) \qquad \Theta = [a_1 a_2 \dots a_{na}]^T$$

Buscar:

Ecuación de error Modelo ARX (Autoregresive eXogenous):

$$y(k) = G(q^{-1})u(k) + H(q^{-1})e(k)$$

Dónde:

$$G(q^{-1}) = \frac{B(q^{-1})}{A(q^{-1})}; \qquad H(q^{-1}) = \frac{1}{A(q^{-1})}$$

$$A(q^{-1}), B(q^{-1}) \qquad \Theta = [a_1 a_2 \dots a_{na} \quad b_1 b_2 \dots b_{nb}]^T$$

Buscar:

Modelo ARMAX (Auto Regresive Moving Average eXogenous):

$$y(k) = G(q^{-1})u(k) + H(q^{-1})e(k)$$

Dónde:

$$G(q^{-1}) = \frac{B(q^{-1})}{A(q^{-1})}; \qquad H(q^{-1}) = \frac{C(q^{-1})}{A(q^{-1})}$$

$$A(q^{-1}), B(q^{-1}), C(q^{-1}) \qquad \Theta = [a_1 a_2 \dots a_{na} \quad b_1 b_2 \dots b_{nb} \quad c_1 c_2 \dots c_{nc}]^T$$

Buscar:

Modelo Output Error (OE):

$$y(k) = G(q^{-1})u(k) + H(q^{-1})e(k)$$

Dónde:

$$G(q^{-1}) = \frac{B(q^{-1})}{F(q^{-1})}; \qquad H(q^{-1}) = 1$$

$$A(q^{-1}) = C(q^{-1}) = F(q^{-1})$$

Este es un modelo ARMAX con:

Modelo BJ (Box-Jenkins):

$$y(k) = G(q^{-1})u(k) + H(q^{-1})e(k)$$

Dónde:

$$G(q^{-1}) = \frac{B(q^{-1})}{F(q^{-1})}; \qquad H(q^{-1}) = \frac{C(q^{-1})}{D(q^{-1})}$$

$$B(q^{-1}), C(q^{-1}), D(q^{-1}), F(q^{-1})$$

Buscar:

Modelo General (PEM):

$$A(q^{-1})y(k) = G(q^{-1})u(k) + H(q^{-1})e(k)$$

Dónde:

$$G(q^{-1}) = \frac{B(q^{-1})}{F(q^{-1})}; \qquad H(q^{-1}) = \frac{C(q^{-1})}{D(q^{-1})}$$

$$A(q^{-1}), B(q^{-1}), C(q^{-1}), D(q^{-1}), F(q^{-1})$$

Buscar:

1.1.3 CONTROL CLÁSICO

Diagrama de bloques de un sistema de control automático de lazo cerrado:

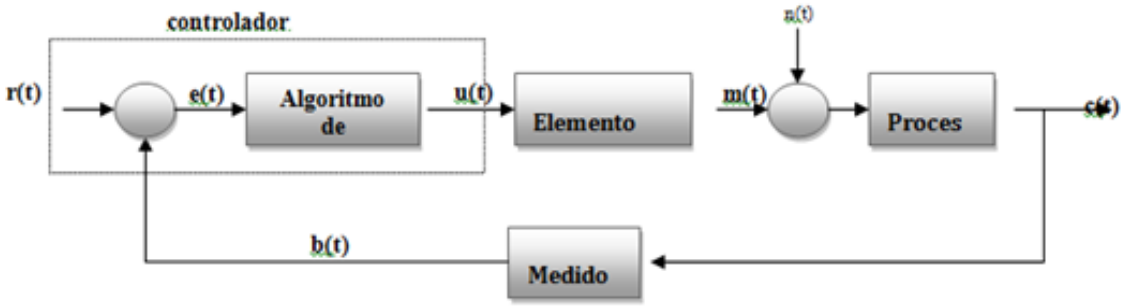

FIGURA 1.19: DIAGRAMA DE SISTEMA EN LAZO CERRADO

- **r(t):** señal de referencia, valor deseado, punto de consigna, valor prescrito (set point).
- **e(t):** señal de error e(t) = r(t) – b(t).
- **u(t):** señal de control.
- **m(t):** variable manipulada.
- **c(t):** variable controlada.
- **b(t):** variable medida.
- **n(t):** señal perturbadora, perturbación, carga.

Formas de funcionar los sistemas de control en lazo cerrado:

- Control regulatorio (que la variable controlada se mantenga en todo instante igual a la referencia).
- Control lógico secuencial (control de operaciones por tiempo o por valores límites).

El control regulatorio puede actuar como:

- Servomecanismo: la referencia es variable y la variable controlada debe seguirla en todo momento. Ejemplos: controles de posición, perfiles de temperatura, etc.
- Regulador: el sistema debe compensar el efecto de la perturbación sobre la variable controlada, haciéndola retornar al valor de la referencia.

  Ejemplos: controles de temperatura, presión, caudal, nivel, pH, etc., en la industria de procesos.

Etapas para el diseño de un sistema de control:

- Definición de los objetivos del sistema de control.
- Selección de variables.
- Obtención de modelo.
- Selección de la configuración y estrategia de control.
- Definición de la estructura y el algoritmo del controlador.

Las leyes de control más utilizadas en los controladores convencionales son:

- Proporcional (P).
- Integral (I).
- Proporcional-integral (PI).
- Proporcional-derivativa (PD).

- Proporcional-integral-derivativa (PID).

CONTROL PROPORCIONAL (P):

- u(t) = Kce(t) + U0.
- Kc: ganancia.
- U0: valor de salida del controlador cuando e(t) = 0.

La función transferencial (variables de desviación):

$$\Delta u(t) = u(t) - U_0 = K_c\left[e(t) - 0\right]$$
$$\Delta u(t) = K_c \Delta e(t)$$
$$U(s) = K_c E(s)$$
$$\frac{U(s)}{E(s)} = K_c \Leftarrow \text{ parámetro del controlador : ganancia}$$

Hay controladores en los que este parámetro no viene dado como ganancia, sino como banda proporcional:

$$\%\text{Banda proporcional (BP)} = \frac{100}{K_c}$$

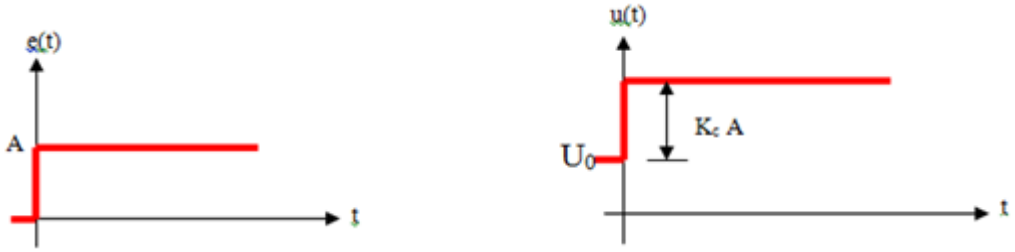

FIGURA 1.20: CONTROL PROPORCIONAL (P)

Dependiendo de las características del sistema, a un paso escalón en la referencia, puede tener un error de estado estacionario (e(t), $t \to \infty$) no nulo.

En ese caso se puede disminuir el Ess (Error de Estado Estacionario) aumentando la ganancia.

Un control P con Kc muy grande se comporta similar a un ON-OFF.

CONTROL INTEGRAL (I):

$$u(t) = \frac{1}{\tau_I}\int_0^t e(t)dt + U_0 \qquad \tau_I : \text{tiempo de acción integral}$$

$$\frac{U(s)}{E(s)} = \frac{1}{\tau_I s}$$

Para e(t) = 0, la salida del controlador mantiene su valor anterior (de antes de anularse).

Esto hace que el Ess = 0.

Introduce fase negativa, lo que provoca que el sistema sea menos estable.

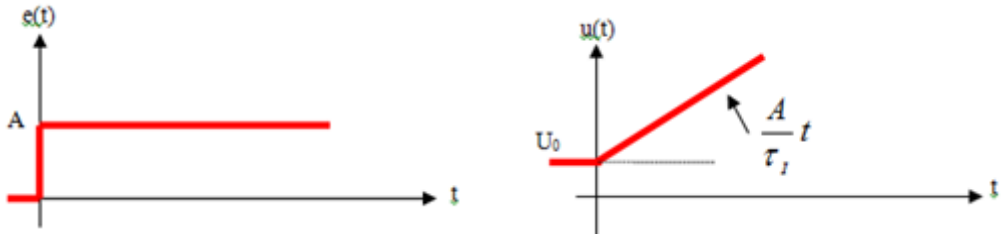

FIGURA 1.21: CONTROL INTEGRAL (I)

Mientras menor tI, mayor efecto de la acción integral.

El sistema presenta un Ess = 0 para un paso en la referencia.

CONTROL PROPORCIONAL-INTEGRAL (PI):

$$u(t) = K_c\left[e(t) + \frac{1}{\tau_I}\int_0^t e(t)dt\right] + U_0 \qquad \text{parámetros} : K_c, \tau_I$$

$$\frac{U(s)}{E(s)} = K_c\left(1 + \frac{1}{\tau_I s}\right)$$

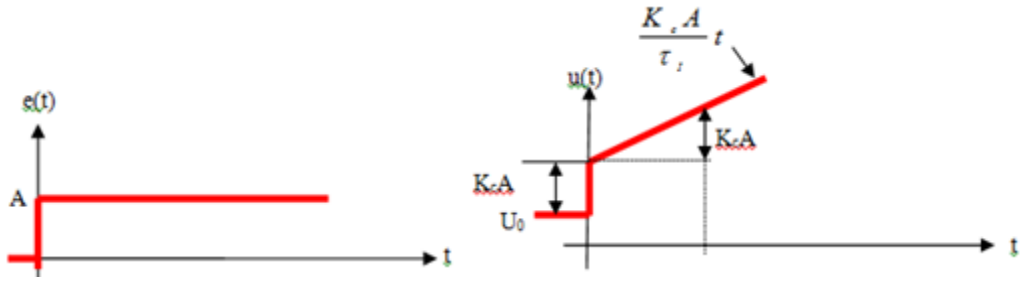

FIGURA 1.22: CONTROL PROPORCIONAL - INTEGRAL (PI)

$$u(t) = K_c\left[e(t) + \tau_D \frac{de(t)}{dt}\right] + U_0 \qquad \text{parámetros} : K_c, \tau_D$$

$$\frac{U(s)}{E(s)} = K_c\left(1 + \tau_D s\right)$$

CONTROL PROPORCIONAL-DERIVATIVO (PD):

Introduce fase positiva, lo que hace al sistema más estable. Amplía el ancho de banda del sistema, lo que tiene dos efectos:

- La respuesta será más rápida.
- Más posibilidad de que entre ruido.
- La acción derivativa no se emplea sola, pues no tiene efecto en estado estacionario.

CONTROL PROPORCIONAL-INTEGRAL-DERIVATIVO (PID):

$$u(t) = K_c\left[e(t) + \frac{1}{\tau_I}\int_0^t e(t)dt + \tau_D \frac{de(t)}{dt}\right] + U_0 \qquad \text{parámetros} : K_c, \tau_I, \tau_D$$

$$\frac{U(s)}{E(s)} = K_c\left(1 + \frac{1}{\tau_I s} + \tau_D s\right)$$

La acción integral hace que el Ess = 0 a un paso en r(t).

La acción derivativa, a un paso en r(t) provoca un impulso en t = 0.

A continuación veremos las respuestas del sistema con diferentes algoritmos de control, ante una señal estímulo paso escalón:

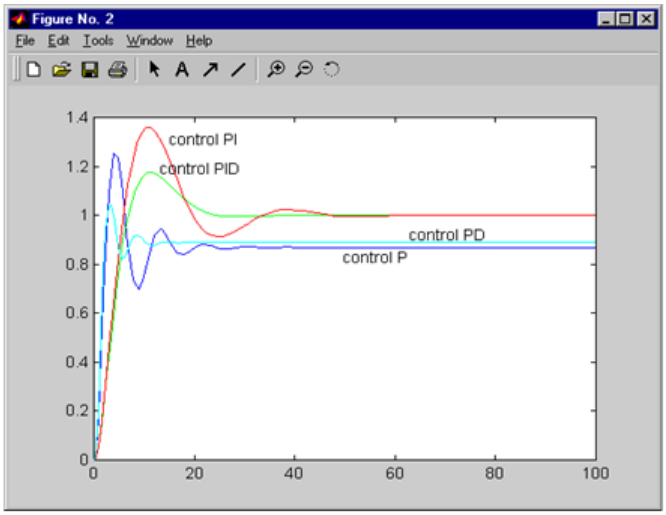

FIGURA 1.23: CONTROL PROPORCIONAL – INTEGRAL – DERIVATIVO

Análisis de estabilidad de los sistemas de control en lazo cerrado:

El análisis del comportamiento de los sistemas de control se hace teniendo en cuenta:

- Respuesta temporal { transitoria (Mp, tr, ts, td)
  Estacionaria (Ess)
- Estabilidad

Definición de estabilidad:

- Un sistema es estable si su salida retorna a un estado de equilibrio cuando es sometido a una perturbación.
- Un sistema es estable si a un estímulo acotado presenta una respuesta acotada.

La respuesta a un paso escalón de un sistema (de orden n) presenta términos del tipo:

$$e^{(\sigma \pm j\omega_d)t} \qquad \text{o} \qquad e^{\sigma t}$$

- De modo que si σ (parte real de sus raíces) es negativa, dichos términos $\rightarrow 0$ cuando $t \rightarrow \infty$.

- Luego, para un sistema estable, todas las raíces de su ecuación característica deben estar en el semiplano izquierdo del plano s, o sea, su parte real debe ser negativa.
- Raíces de la ecuación característica = polos de lazo cerrado

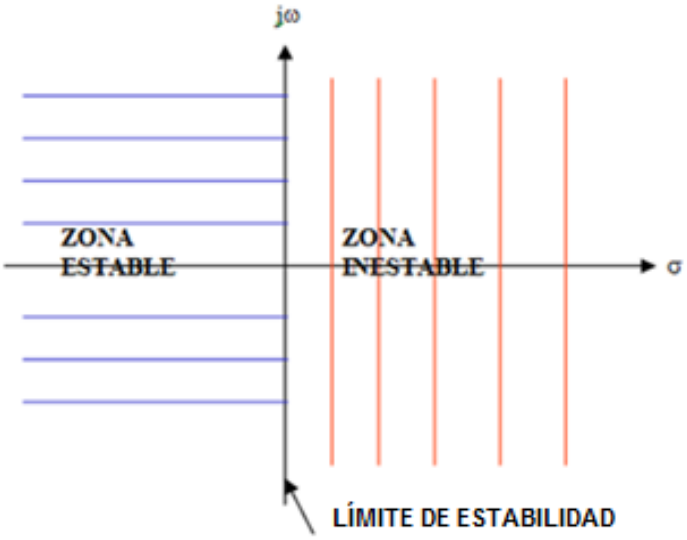

FIGURA 1.24: PLANO COMPLEJO S

- Estabilidad absoluta se refiere a si un sistema es estable o no.
- Estabilidad relativa se refiere a cuan estable es el sistema, o sea, a su grado de estabilidad.
- Es hecho de que un sistema sea estable no garantiza que la respuesta será satisfactoria. Raíces muy cercanas al eje jw pueden dar lugar a respuestas muy oscilatorias o muy lentas.

Concepto del Lugar Geométrico de las Raíces (L.G.R.):

El L.G.R. es el gráfico de la ubicación en el plano complejo s de las raíces de la ecuación característica del sistema, cuando varía un parámetro del mismo.

Efecto de la adición y/o corrimiento de polos y ceros.

$$\mathbf{G(s)H(s)} = \frac{\mathbf{K}}{(\mathbf{s}+2)(\mathbf{s}+4)}$$

Suponga un sistema cuya F.T.L.A. es:

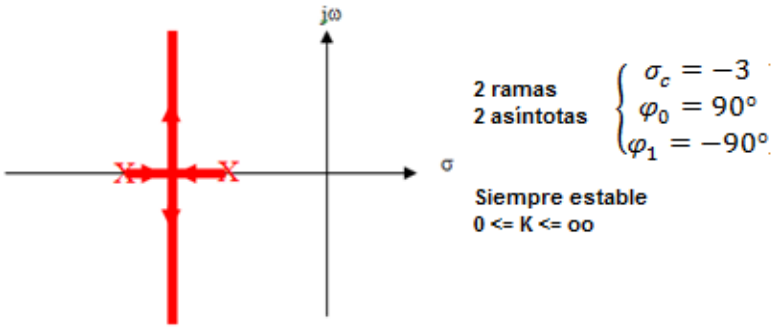

FIGURA 1.25: LUGAR GEOMÉTRICO DE LAS RAÍCES

$$\frac{1}{\tau_i s} \qquad G(s)H(s) = \frac{K'}{s(s+2)(s+4)} \qquad K' = \frac{K}{\tau_i}$$

Se introduce acción integral:

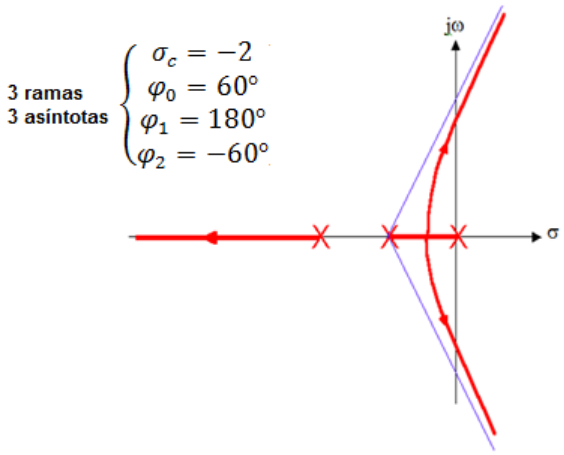

FIGURA 1.26: LUGAR GEOMÉTRICO DE LAS RAÍCES CON ACCION INTEGRAL

$$G_c(s) = K_c\left(\tfrac{1}{6}s + 1\right) = \tfrac{K_c}{6}(s+6)$$

$$G(s)H(s) = \frac{K'(s+6)}{(s+2)(s+4)} \qquad K' = \frac{KK_c}{6}$$

Efecto del movimiento de un polo:

$$G(s)H(s) = \frac{K}{(s+2)(s+4)(s+a)}$$

Mientras más alejado está el polo del eje, Kcrit aumenta con lo que el sistema gana en estabilidad.

Efecto del movimiento de un cero:

$$\mathbf{G(s)H(s)} = \frac{\mathbf{K(s+a)}}{\mathbf{(s+2)(s+4)}}$$

$$\mathbf{a} = \frac{1}{\tau_{\mathbf{d}}}$$

Por ejemplo:

Si aumenta τd, el cero va acercándose a los polos, y "la bolita" va cerrándose, por lo que la ωd máxima será menor.

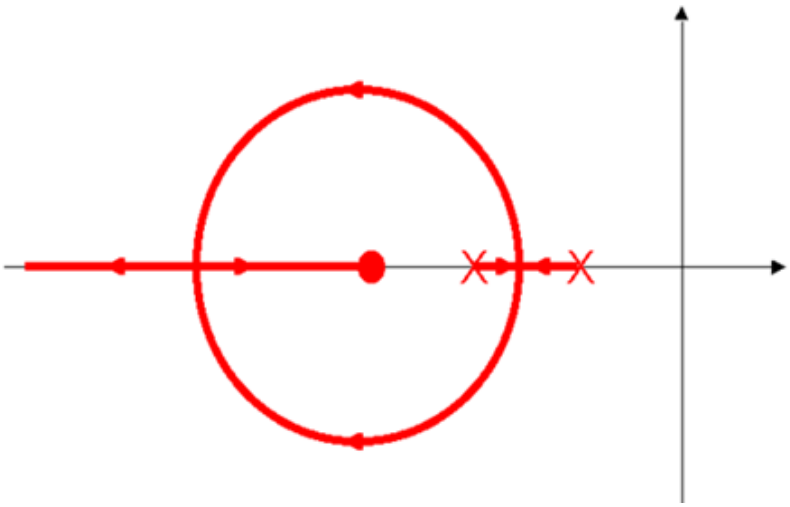

FIGURA 1.27: LUGAR GEOMÉTRICO DE LAS RAÍCES CON ACCION DERIVATIVA

1.1.4 CONTROL DISCRETOS

Un sistema en tiempo discreto viene caracterizado por magnitudes que varían solo en instantes específicos de tiempo. Estas magnitudes o señales en tiempo discreto toman valores r(k), r(t), r(t), r(nt).

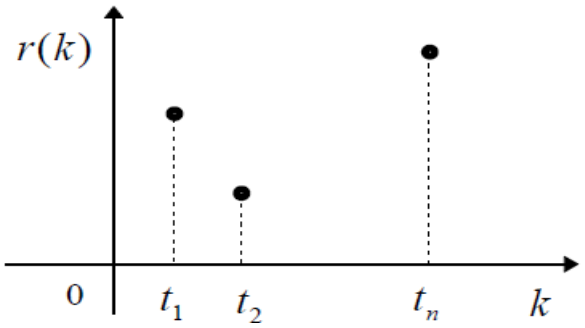

FIGURA 1.28: DISCRETIZACIÓN DE UNA SEÑAL

Además de los sistemas inherentemente discretos, se incluyen también en esta categoría a los sistemas continuos muestreados con r(k) formada por r(T), r(2T),…,r(nT)

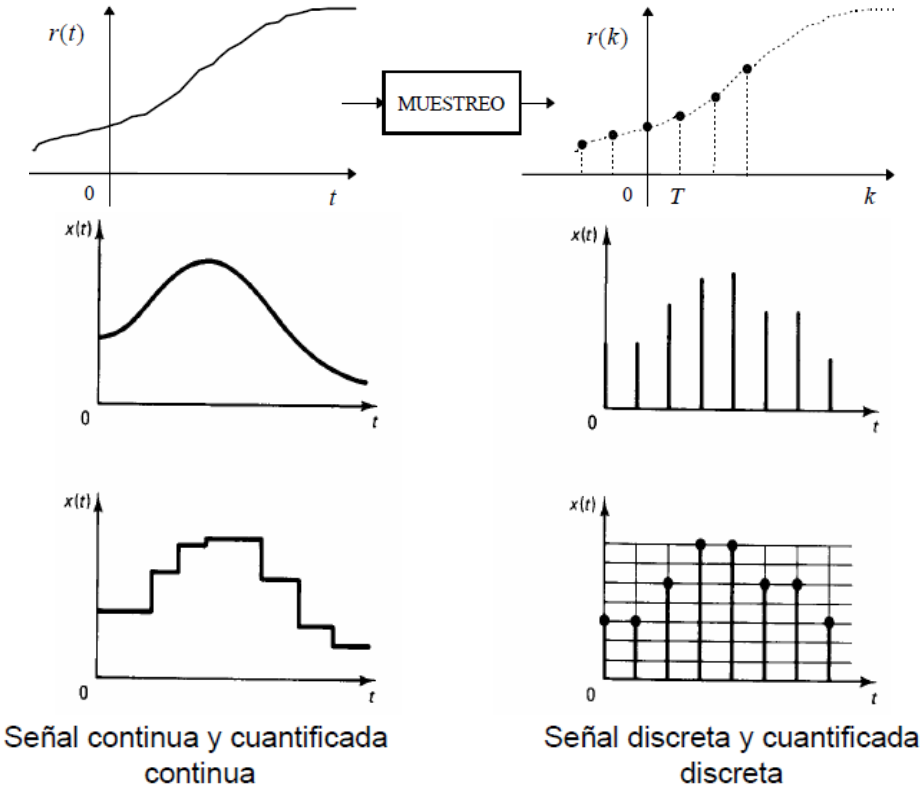

FIGURA 1.29: DISCRETIZACIÓN Y MUESTREO DE UNA SEÑAL

Sistema de Control Discreto:

Un sistema de control discreto es aquel que incluye un computador digital en el bucle de control para realizar un procesamiento de señal.

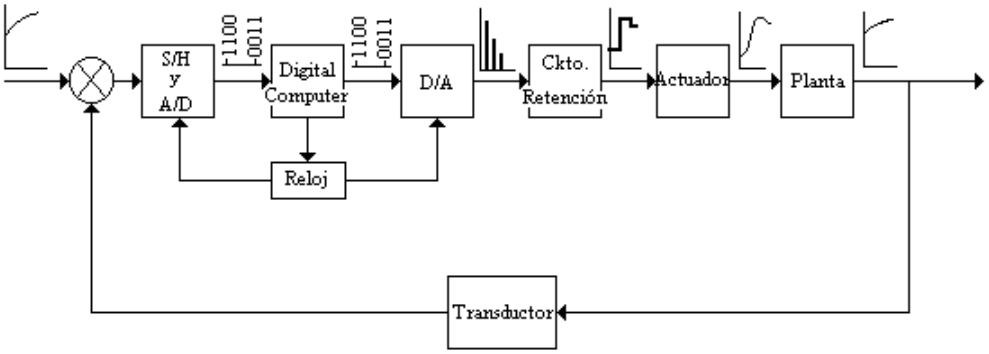

FIGURA 1.30: SISTEMA DE CONTROL DISCRETO

La salida de la planta es continua y es realimentada a través de un transductor que convierte la señal de salida en señal eléctrica. La señal de error continuo es convertida a señal digital a través del circuito de muestreo (periodo T) y reconstrucción S&H (Sample and Hold) y del convertidor A/D, proceso llamado codificación.

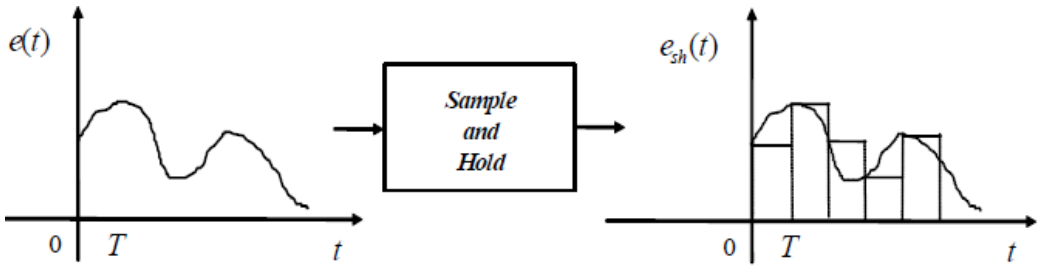

FIGURA 1.31: EFECTO SAMPLE AND HOLD

El convertidor A/D (encoder) convierte la señal continua en señal digital binaria, mediante un proceso de cuantificación.

El computador digital procesa la secuencia de valores de entrada digital a través de un algoritmo y produce una salida digital, según establezca la ley de control.

Esta señal de control habrá de ser transformada de nuevo a señal continua como entrada a la planta (acción de filtrado de la planta). Ver referencia [1].

El convertidor D/A y el circuito de reconstrucción (Hold) convierten la secuencia de valores en una señal continua, proceso llamado decodificación.

El circuito de reconstrucción (Hold) retiene el valor de salida del convertidor D/A justo un periodo T.

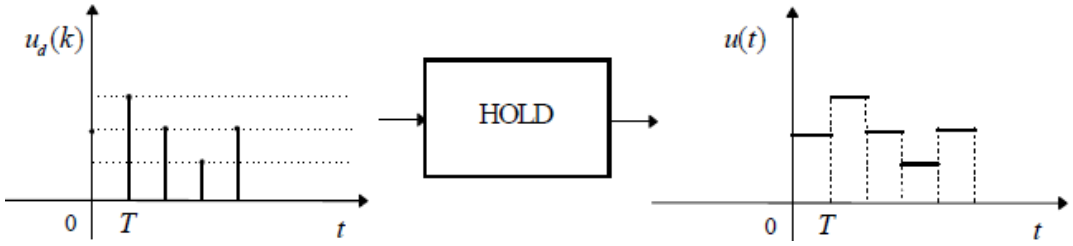

FIGURA 1.32: EFECTO HOLD

El uso del control discreto presenta ventajas e inconvenientes frente al control analógico. Hay diversos campos de aplicación del control digital.

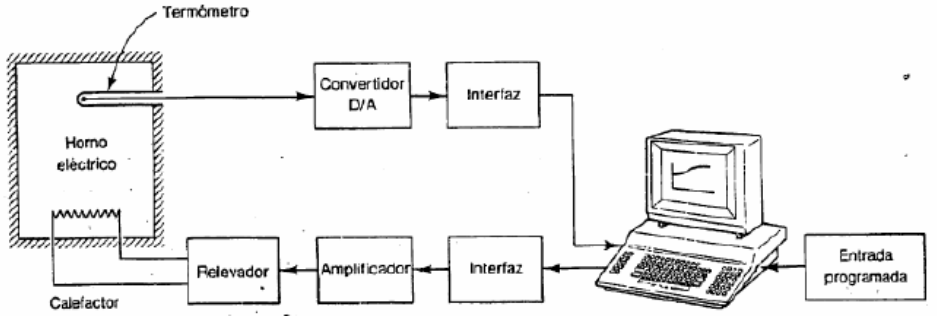

FIGURA 1.33: SISTEMA DE CONTROL DISCRETO LAZO CERRADO

La descripción de los sistemas en tiempo discreto viene definida por ecuaciones en diferencia, que relacionan la señal de salida con la señal de entrada.

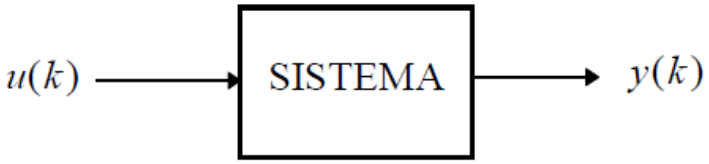

FIGURA 1.34: SISTEMA A DISCRETIZAR

$$u(kT) = u(k) = \{u(0), u(T), u(2T), \ldots, u(kT), \ldots\}$$

$$y(kT) = y(k) = \{y(0), y(T), y(2T), .., u(kT), \ldots\}$$

El algoritmo de control del computador digital podrá ser expresado como una ecuación en diferencias.

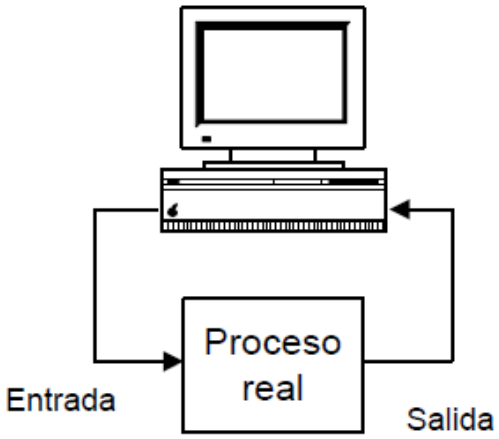

FIGURA 1.35: SISTEMA DE CONTROL POR COMPUTADOR

Ejemplo: controlador PI discreto

$$u(kT) = k_p . e(kT) + k_i . x(kT)$$

$$x(kT) = x\big((k-1)T\big) + T.e(kT)$$

1.1.5 CONTROL MODERNO FUZZI

En general, la lógica difusa provee una estructura de inferencia que posibilita apropiadas capacidades de razonamiento humano. Usa la teoría de la probabilidad para explicar si un evento ocurre o no, mide la posibilidad de que un evento dado pueda ocurrir. La utilidad de los conjuntos difusos está en su habilidad para modelar incertidumbre o datos ambiguos, que a menudo se encuentran en la vida real.

Los controles lógicos difusos utilizan la lógica difusa para convertir las estrategias de control lingüísticas basadas en el conocimiento experto en una estrategia de control automática.

La lógica Difusa básicamente propone variables del mundo real no de forma discreta o exacta sino en forma de grados de pertenencia a determinados conjuntos difusos.

Los modelos difusos proporcionan una manera aproximada pero efectiva de describir el compartimiento de los sistemas muy complejos, mal definidos o no fácilmente analizables.

Para utilizar la lógica difusa con propósito de control es necesario introducir los siguientes pasos:

- **FUSIFICACIÓN:** Proceso que convierte un valor preciso a un valor difuso.
- **INFERENCIA DIFUSA:** Mecanismo responsable de sacar conclusiones a partir de la base de conocimiento. Este proceso se basa en la aplicación de reglas del tipo IF…THEN.
- **DESFUSIFICACIÓN:** Proceso que convierte la acción de control difusa a una acción de control precisa.

Las técnicas de lógica difusa se han aplicado en: análisis de datos, predicción, modelado y control.

En el área de control se ha aplicado en:

- Elevadores.
- Equipos electrodomésticos.
- Control automático de trenes.
- Cámaras de video.

- Sistema de aterrizaje de aviones.
- Control de procesos.

En la teoría de los conjuntos difusos está dirigida a tratar con fenómenos complejos que no pueden ser analizados mediante métodos clásicos basados en la lógica bivalente o la teoría de probabilidades.

Conjunto difuso.- Un conjunto difuso, a diferencia de un conjunto clásico, es un tipo especial de conjunto que permite la pertenencia parcial de sus elementos.

Universo de discurso U.- Determina la gama de valores que pueden tomar los elementos que poseen la propiedad expresada por la variable lingüística.

Etiquetas.- Son las diferentes clasificaciones que se efectúan sobre la variable lingüística. Cada etiqueta tendrá un conjunto difuso asociado.

Función de pertenencia o membresía u(x).- Es una relación que asocia cada elemento en un conjunto con su grado de pertenencia al mismo (un número real en el intervalo [0, 1]). Puede expresarse como un grupo de valores discretos o como una función continua.

Soporte.- Proporciona el rango de definición de la función de pertenencia. Es una manera de restringir el universo de discurso parara a cada etiqueta.

Es el subconjunto de los elementos del conjunto universal que pertenecen al conjunto difuso con un grado de pertenencia mayor o igual que α. El valor α se denomina nivel - α.

Punto de cruce.- Es el punto de la función de membresía donde toma el valor de 0.5.

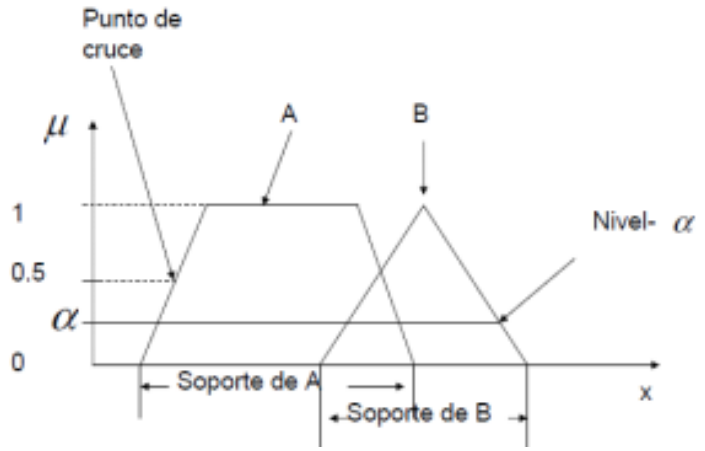

FIGURA 1.36: PUNTO DE CRUCE

Convexidad.- Una función de membresía puede ser uni-modal (tiene un máximo global), o multimodal (si tiene varios máximos). Los conjuntos difusos uni-modales se denominan conjuntos difusos convexos.

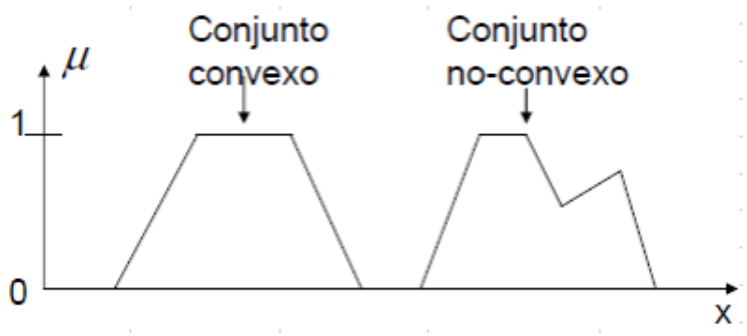

FIGURA 1.37: CONVEXIDAD

Conjuntos difusos comunes en ingeniería:

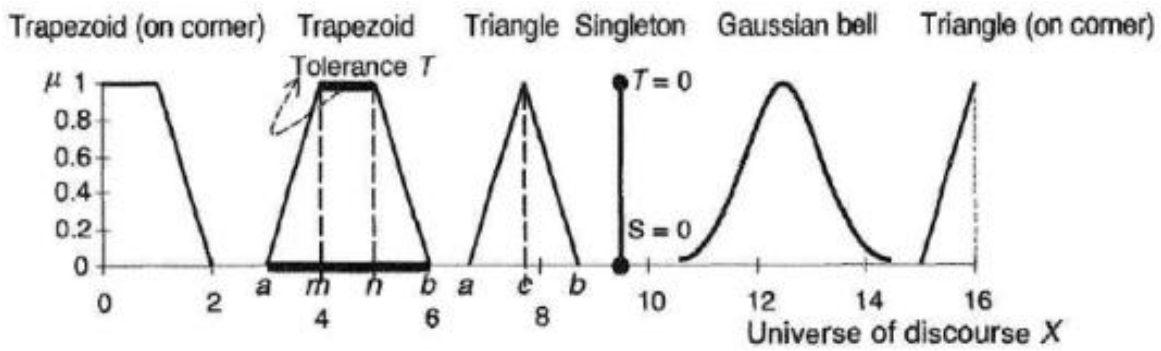

FIGURA 1.38: TIPOS DE CONJUNTOS DIFUSOS

Operaciones sobre conjuntos difusos:

Una vez que se tengan definidos los conjuntos difusos, deben plantearse las relaciones entre ellos y las operaciones que se pueden realizar.

Esto es importante pues las reglas al igual que en la lógica básica evalúan condiciones de tipo IF (expresión) THEN (relación, conclusión).

- **AND:** T-Norma T(ìA(x), ìB(x))
- **OR:** T-Conorma T(ìA(x), ìB(x))
- **Mínimo:** MIN(μA(x), μB(x))
- **Máximo:** MAX(ìA(x), ìB(x))
- **Producto Algebraico:** ìA(x)ìB(x)
- **Suma Algebraica:** ìA(x) + ìB(x) . ìA(x)ìB(x)
- **Lukasiewicz AND:** MAX(0, μA(x) + μB(x) − 1)
- **Lukasiewicz OR:** MIN(1, μA(x) + μB(x))

Sistemas basados en reglas: En los sistemas difusos basados en reglas, las relaciones entre las variables son representadas por reglas del tipo IF-THEN de la forma siguiente: IF proposición del antecedente THEN proposición del consecuente.

Modelo difuso lingüístico (Zadeh, 1973, Mamdani, 1977): Los antecedentes y los consecuentes son modelos difusos. El modelo difuso solitario es un caso especial donde los consecuentes son conjuntos solitarios (singleton).

Modelo relacional difuso (Pedrycz, 1984 y Yi y Chung, 1994): Que puede considerarse una generalización del modelo lingüístico, permite que una proposición del antecedente en particular este asociada a varias proposiciones diferentes del consecuente, por medio de una relación difusa.

Modelo lingüístico: El modelo lingüístico ha sido introducido como una manera de capturar conocimiento en forma de reglas IF-THEN.

Ri: IF x es Ai THEN y es Bi i= 1,2,…,K

Aquí x es la variable lingüística de entrada (antecedente) y Ai es la etiqueta.

Similarmente, y es la variable lingüística de salida (consecuente) y Bi es la etiqueta lingüística del consecuente.

Una variable lingüística L es definida como un quíntuple: L = (x,A,X,g,m), x es la variable base (tiene el mismo nombre que la variable lingüística).

A= {A1, A2…,An} es el conjunto de términos lingüísticos o etiquetas.

X es el dominio o universo de discurso de x.

g es una regla sintáctica para genera etiquetas.

m es una regla semántica que asigna a cada etiqueta su significado (un conjunto difuso en X).

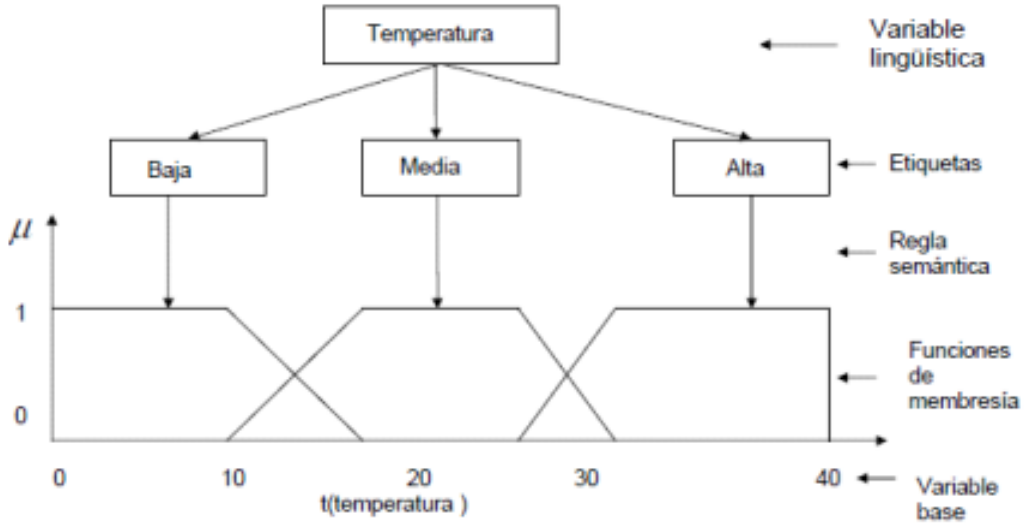

FIGURA 1.39: MODELO LINGUÍSTICO

Composición de relaciones difusas: Las relaciones en diferentes espacios productos pueden ser combinadas entre ellas por medio de la composición. Además, los conjuntos difusos pueden ser combinados con relaciones difusas de la misma manera.

Existen diferentes versiones del operador composicional. La más conocida de todas es una composición MAX-MIN. Otras como MAXPROD y MAX-AVERAGE también pueden ser utilizadas.

La inferencia difusa es el proceso de formular el mapeo de una entrada dada hacia una salida usando la lógica difusa. La inferencia en sistemas difusos basados en reglas es el proceso de obtener un conjunto difuso de salida, dada las reglas y las entradas. El mecanismo de inferencia en el modelo lingüístico se basa en la regla composicional de inferencia (Zadeh). Esto es, ser capaces de inferir una conclusión cuando el antecedente es parcialmente conocido o cuando el hecho es similar pero no igual al conocido.

Inferencia en el modelo lingüístico: La inferencia, en sistemas difusos basados en reglas, es el proceso de derivar un conjunto difuso de salida, dada las reglas y las entradas.

El mecanismo de inferencia en el modelo lingüístico se basa en la regla composicional de inferencia de Zadeh.

Cada regla puede ser considerada como una relación difusa, que es calculada como:

$$\mu_R(x,y) = I(\mu_A(x),\mu_B(y))$$

El operador I puede ser una implicación difusa o un operador de conjunción (una t-norma).

Las implicaciones difusas son usadas cuando la regla está relacionada con una implicación Ai Bi, es decir, Ai implica Bi.

Cuando se utiliza una conjunción, A B, la interpretación de las reglas IF – THEN es ¨es cierto que A y B simultáneamente se cumplen¨.

1.1.6 CONTROL PREDICTIVO

El control predictivo tiene como objetivo resolver de forma efectiva, problemas de control y automatización de procesos industriales que se caractericen por presentar un comportamiento dinámico complicado, multivariable, y/o inestable. La estrategia de control en que se basa este tipo de control, utiliza el modelo matemático del proceso a controlar para predecir el comportamiento futuro de dicho sistema, y en base a este comportamiento futuro puede predecir la señal de control futura.

El control predictivo integra disciplinas como el control óptimo, control estocástico, control de procesos con retardo de tiempo, control multivariable, control con restricciones.

El tipo de control predictivo tratado, es el Control Predictivo Basado en Modelo (CPBM), conocido también como Model Based Predictive Control (MBPC) o simplemente Model Predictive Control (MPC). Esta estrategia también se conoce como control por horizonte deslizante, por ser ésta la forma en la que se aplican las señales de actuación. Existen muchos algoritmos de control predictivo que han sido aplicados con éxito: GPC, IDCOM, DMC, APC, PFC, EPSAC, RCA, MUSMAR, NPC, UPC, SCAP, HPC, etc.

El control predictivo basado en modelo se puede definir como una estrategia de control que se basa en la utilización de forma explícita de un modelo matemático interno del proceso a controlar (modelo de predicción), el cual se utiliza para predecir la evolución de las variables a controlar a lo largo de un horizonte temporal de predicción especificado por el operador, de este modo se puede calcular las variables manipuladas futuras (señal de control futura) para lograr que en dicho horizonte, las variables controladas converjan en sus respectivos valores de referencia.

El MPC se enmarca dentro de los controladores óptimos, es decir, aquellos en los que las actuaciones responden a la optimización de un criterio. El criterio a optimizar, o función de coste, está relacionado con el comportamiento futuro del sistema, que se predice gracias a un modelo dinámico del mismo, denominado modelo de predicción.

El intervalo de tiempo futuro que se considera en la optimización se denomina horizonte de predicción. Dado que el comportamiento futuro del sistema depende de las actuaciones que se aplican a lo largo del horizonte de predicción, son éstas las variables de decisión respecto a las que se optimiza el sistema. La aplicación de estas actuaciones sobre el sistema conduce a un control en bucle abierto.

La posible discrepancia entre el comportamiento predictivo y el comportamiento real del sistema crean la necesidad de imponer cierta robustez al sistema incorporando realimentación del mismo. Esta realimentación se consigue gracias a la técnica del horizonte deslizante que consiste en aplicar las actuaciones obtenidas durante un periodo de tiempo, tras el cual se muestrea el estado del sistema y se resuelve un nuevo problema de optimización. De esta manera, el horizonte de predicción se va deslizando a lo largo del tiempo.

Una de las propiedades más atractivas del MPC es su formulación abierta, que permite la incorporación de distintos tipos de modelos de predicción, sean lineales o no lineales, mono-variables o multi-variables, y la consideración de restricciones sobre las señales del sistema. Esto hace que sea una estrategia muy utilizada en diversas áreas del control. El CPBM es una de las pocas técnicas que permiten controlar sistemas con restricciones incorporando éstas en el propio diseño del controlador.

Estas características han hecho del control predictivo una de las escasas estrategias de control avanzado con un impacto importante en problemas de ámbito industrial Por tal motivo es importante resaltar que el control predictivo se ha desarrollado en el mundo de la industria, y ha sido la comunidad investigadora la que se ha esforzado en dar un soporte teórico a los resultados prácticos obtenidos.

Merece la pena destacar que el control predictivo es una técnica muy potente que permite formular controladores para sistemas complejos y con restricciones. Esta potencia tiene un precio asociado: el coste computacional y la sintonización del controlador. Recientes avances en el campo del MPC proveen un conocimiento más profundo de estos controladores, obteniéndose resultados que permiten relajar estos requerimientos. Así por ejemplo, se han establecido condiciones generales para garantizar la estabilidad (Mayne 2001), condiciones bajo las cuales se puede relajar el carácter optimal del controlador garantizando su estabilidad (Scokaert & Mayne 1998).

Ventajas y desventajas del Control Predictivo.-

Entre las ventajas se pueden citar:

- Formulación en el dominio del tiempo, lo cual le permite ser una técnica flexible, abierta e intuitiva.
- Permite tratar con sistemas lineales y no lineales, mono-variables y multi-variables utilizando la misma formulación para los algoritmos del controlador.
- La ley de control responde a criterios de optimización.
- Permite la incorporación de restricciones en la síntesis o implementación del controlador.
- Brinda la posibilidad de incorporar restricciones en el cálculo de las actuaciones

Entre las desventajas se pueden citar:

- Requiere el conocimiento de un modelo dinámico del sistema suficientemente preciso.
- Requiere un algoritmo de optimización, por lo que solo se podría implementarse por medio de una computadora.

- Requiere un alto coste computacional, lo que hace difícil su aplicación a sistemas rápidos.
- Hasta hace relativamente poco, no se podía garantizar la estabilidad de los controladores, especialmente en el caso con restricciones. Esto hacía que el ajuste de estos controladores fuese heurístico y sin un conocimiento de cómo podían influir los parámetros en la estabilidad del sistema.

ELEMENTOS DEL CONTROL PREDICTIVO:

Hay una serie de elementos comunes a todos los controladores predictivos:

- El uso de un modelo matemático del proceso que se utiliza para predecir la evolución futura de las variables controladas sobre un horizonte de predicción.
- La imposición de una estructura en las variables manipuladas futuras.
- El establecimiento de una trayectoria deseada futura, o referencia, para las variables controladas.
- El cálculo de las variables manipuladas optimizando una cierta función de costo.
- La aplicación del control siguiendo una política de horizonte móvil.

1.2 RECURSOS DE HARDWARE DISPONIBLES

Los módulos utilizados son de la firma comercial IDETEC CIA. LTDA., los mismos que fueron escogidos por la facilidad de uso en aplicaciones de electrónica con microcontroladores. Entre los principales productos que IDETEC CIA. LTDA. ofrece a sus clientes están:

- **ENTRENADORAS:**
    - MEI&T04 (PIC16F886): Módulo entrenamiento con PIC16F886, sin interfaz USB.
    - MEI&T04 (PIC16F886) USB: Módulo entrenamiento con PIC16F886, con interfaz USB.
    - MEI&T887 (PIC16F887): Módulo entrenamiento con PIC16F887.

- **PROGRAMADOR:**
    - P.PICI&T04: Programador de Microcontroladores de la familia Microchip.

- **COMUNICACIÓN:**
    - XBEE-USB I&T: Interfaz USB para módulos XBEE.
    - XBEE-TTL I&T02: Interfaz TTL para módulos XBEE.
    - PSCONTROL I&T02: Interfaz para palanca PS2 con conexión XBEE para comunicación inalámbrica.
    - ETHERNET I&T 02: Módulo con interfaz Ethernet y comunicación SPI.
    - USB-UART I&T 02: Módulo convertidor USB a interfaz TTL serial.

- **SENSORES:**
    - QRD1114 I&T 02: sensor infrarrojo de corto alcance para detectar colores.
    - BANCO QRD1114 I&T 02: Banco de sensores infrarrojos de corto alcance.
    - ENCODER OPTICO I&T.
    - DS18B20 I&T: Sensor de temperatura.
    - LDR I&T: Resistencia dependiente de LUZ.

- **DRIVERS/ACTUADORES:**
    - MD8RELE I&T 02: Disparador 8 relés.
    - MD1RELE I&T 02: Disparador 1 Relé.
    - PH2A I&T 02: Puente H - 2A.
    - PH600mA I&T 04: Puente H - 600mA.
    - MC8SERVO I&T: Controlador de hasta 8 servomotores.
    - MC-AC-IND. I&T: Control de carga AC Inductivo.
    - MC-AC-RES. I&T: Control de carga AC Resistivo.

1.2.1 MÓDULO DE DESARROLLO MEI&T04

M.E. I&T04 es un módulo de entrenamiento y desarrollo que nos permite realizar múltiples tareas con el microcontrolador 16F886.

FIGURA 1.40: MÓDULO DE ENTRENAMIENTO MEI&T04

- Comunicación serial asíncrona UART.
- Comunicación serial síncrona SPI e I2C.
- Comunicación ONE WIRE y USART.
- Comunicación inalámbrica RX y TX con módulos FSK y ASK.

- Potenciómetro integrado.
- 10 entradas analógicas.
- 24 entradas y salidas digitales.
- 8 leds indicadores de salidas digitales.
- Control para 4 servomotores.
- Control para 2 motores DC (Dirección y Velocidad).
- Programación ICSP in circuit.
- Reset manual.
- Switch de ON/OFF.
- Led indicador de power.
- Regulador integrado.

Aplicaciones:

- Construcción de robots (seguidores de líneas, sumobot, teleoperados, exploradores, soccer, etc.).
- Aplicaciones de Telemetría y radio control.
- Implementación de sistemas de control.
- Tarjeta de adquisición de datos.
- Placa de desarrollo de ejercicios de Programación con microcontroladores.

Fuente de Alimentación:

- Alimentación desde (5 - 25) VDC en el EXT (VIN 9V).
- Interruptor ON/OFF para energizar o des energizar al módulo de entrenamiento.

- Jumper (USB/ EXT) permite seleccionar la fuente que va alimentar al módulo la cual puede ser mediante el EXT (VIN 9V) o USB que es la energía que se obtiene a través del puerto USB.

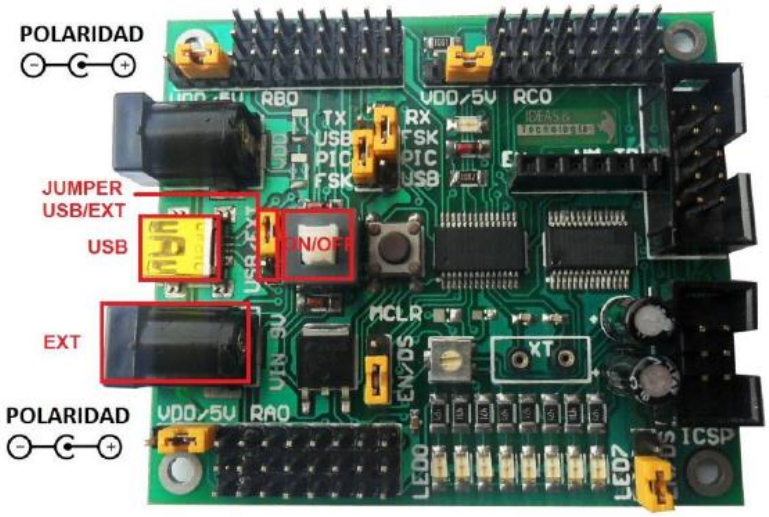

FIGURA 1.41: FUENTE DE ALIMENTACIÓN DEL MÓDULO MEI&T04

Microcontrolador PIC16F886:

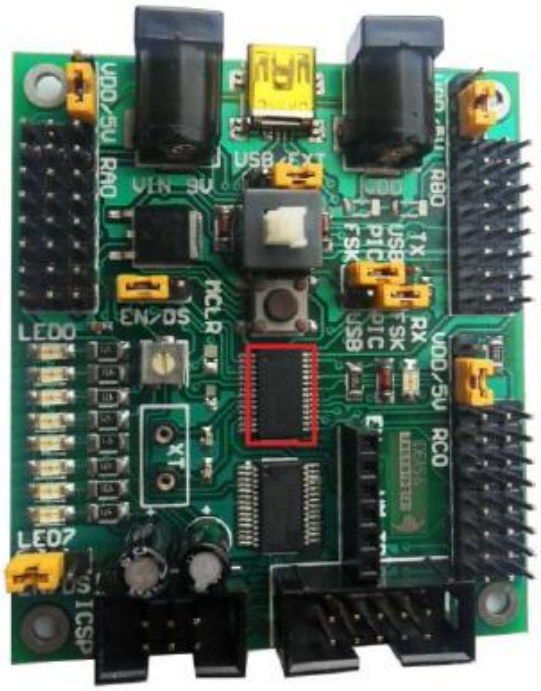

FIGURA 1.42: PIC 16F886 DEL MÓDULO MEI&T04

- 25/24 Pines Entradas/Salidas.
- Puertos (PORT A, B, C, E).
- Oscilador interno seleccionable entre (31KHZ – 8MHZ).
- Rango de voltaje de Operación (2 – 5.5) VDC.
- 11 entradas analógicas con 10 bit de resolución.
- 3 Timers (Timer0 8bits, Timer1, 2 16bits).
- 2 PWM (CCP) de 10bits, frecuencia máx. 20KHZ.
- Comunicaciones seriales sincrónicas MSSP (SPI (4 modos), I2C).
- Módulo USART (RS-485, RS-232 and LIN 2.0). Ver referencia [5].

Reset (MCLR):

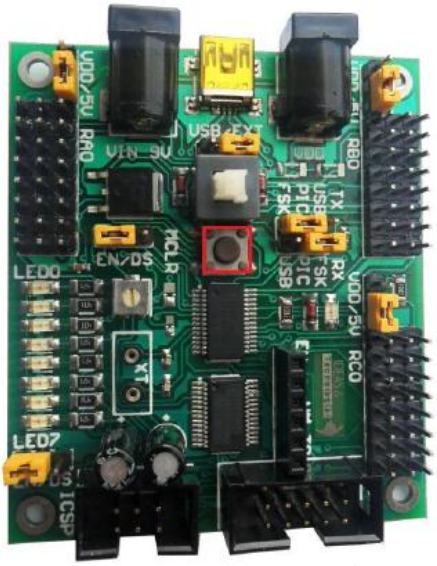

FIGURA 1.43: MCLR DEL MÓDULO MEI&T04

Este botón posee un resistor pull up y está conectado al PIN MCLR.

Para utilizar este botón es necesario que se lo habilite mediante software.

ICSP (Programación serial en circuito):

FIGURA 1.44: CONECTOR ICSP DEL MÓDULO MEI&T04

Este conector IDC 3X2 (conector por desplazamiento del aislante) se lo utiliza para cargar el código en el microcontrolador usando cualquier programador con terminales ICSP.

El módulo de entrenamiento puede ser alimentado con el programador, habilitando en power target en el software.

Entradas/Salidas en Puertos A, B, C:

Cada puerto tiene 8 pines correspondientes a los 8 bits, a cada bit se denomina Señal I/O acompañada de pines de +Vcc y Gnd, donde Vcc puede ser seleccionable es decir utilizar el voltaje interno del módulo entrenamiento o externo mediante el Jack VDD.

PORTA (0, 1, 2, 3, 4, 5, 6, 7)

PORTB (0, 1, 2, 3, 4, 5, 6, 7)

PORTC (0, 1, 2, 3, 4, 5, 6, 7)22,4+3,95+0,8

Entradas Analógicas:

AN0=RA0, AN1=RA1, AN2=RA2, AN3=RA3, AN4=RA5, AN8=RB2, AN9=RB3, AN10=RB1, AN11=RB4, AN12=RB0, AN13=RB5.

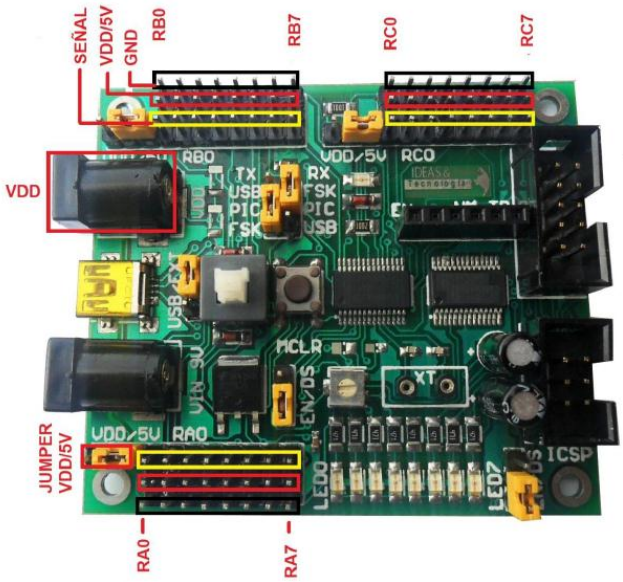

FIGURA 1.45: POLARIZACIÓN DE PUERTOS A - B - C DEL MÓDULO MEI&T04

Led en PORTB:

Para utilizar los led del PORTB, debemos setear los pines de este puerto como salidas, además debemos de habilitar el jumper LED (EN/DS). De esta manera quedan habilitados todos los indicadores led del módulo de entrenamiento.

Conexiones:

- LED0-> RB0 LED7-> RB7.
- Jumper Led: EN Habilita todos los led conectándolos a GND.
- Jumper Led: DS Deshabilita todos los led.

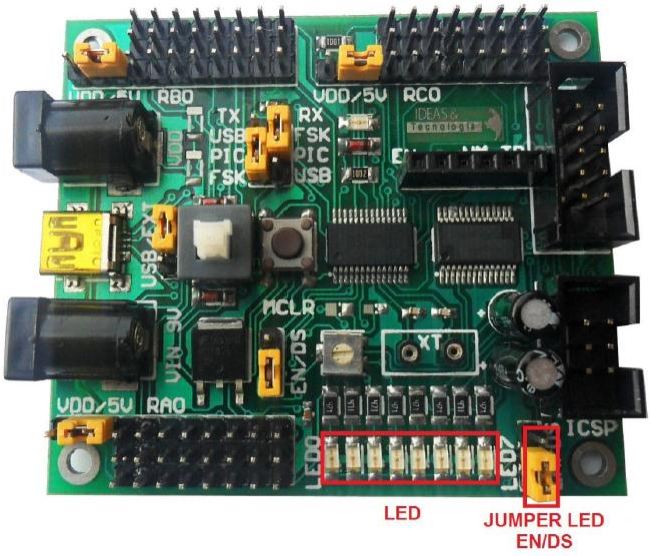

FIGURA 1.46: LEDS DEL MÓDULO MEI&T04

Potenciómetro en PORTA:

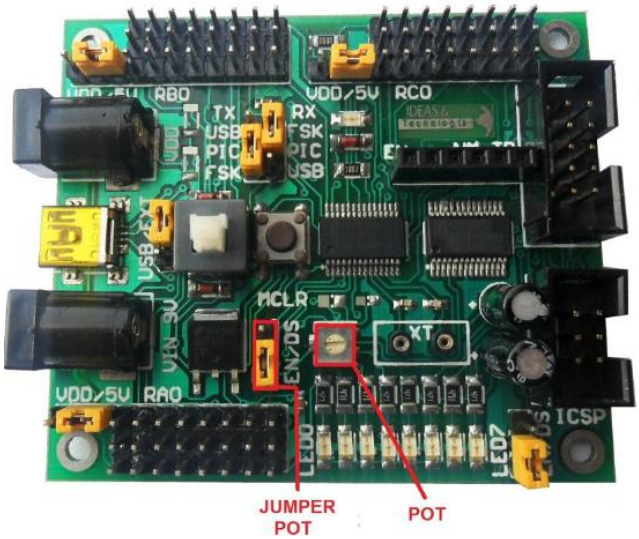

FIGURA 1.47: POTENCIÓMETRO DEL MÓDULO MEI&T04

Para utilizar el potenciómetro del módulo de entrenamiento se debe colocar el Jumper POT (EN/DS) tal como se indica en la figura de esta manera, el cual permite conectar el Pin RA0 del PIC con el potenciómetro.

Mediante la variación del POT se producirá un voltaje analógico entre (0-5VDC) el cual será enviado al PIN A0.

Control de motor DC:

El módulo de entrenamiento M.E I&T 04 permite controlar la dirección y velocidad de motores DC.

Para aquello se agregado un conector IDC de 5X2 compatible eléctricamente con los módulos P. H I&T04, P.H.2A I&T (Puente H para motores DC),y otros módulos desarrollados por IDEAS&TECNOLOGIA.

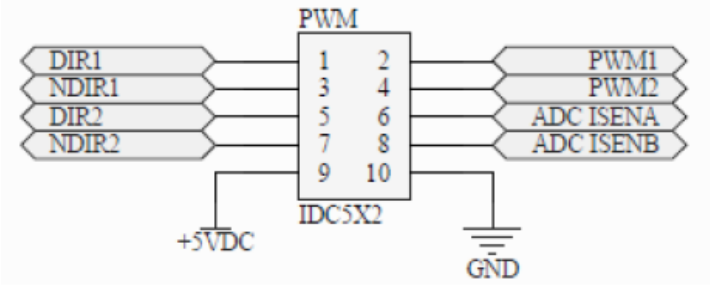

FIGURA 1.48: CONECTOR PARA CONTROL DE MOTORES DC

DIR1: RA2 NDIR1: RA4 DIR2: RA3

NDIR2: RA5 PWM1: RC1 PWM2: RC2

ADCISENA: RB1 ADCISENB: RB2

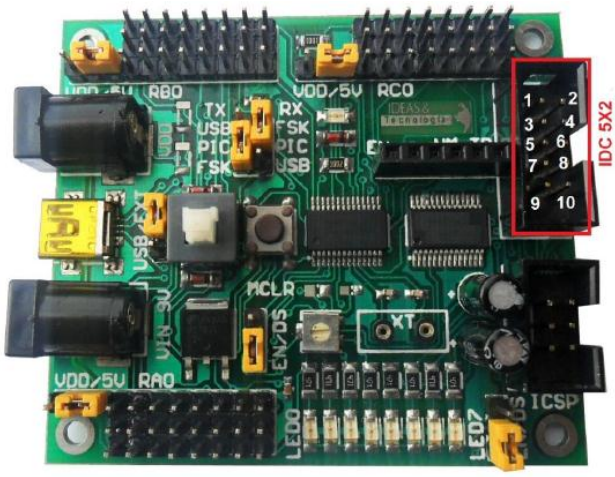

FIGURA 1.49: CONECTOR PARA CONTROL DE MOTORES DC DEL MÓDULO MEI&TO4

Control de Servomotor:

Podemos conectar hasta 24 servomotores, debido a que a cada puerto se le agrego pines de +Vcc y Gnd según el estándar de los servomotores los cuales poseen 3 señales.

Señal: Este pin necesita una señal PPM.

VDD/5V: Alimentación a través del jack VDD si se requiere más corriente o se utiliza el voltaje del módulo de entrenamiento 5V.

GND: Tierra (0 VDC).

Jumper VDD/5V: Permite seleccionar la fuente de alimentación para los servomotores la cual puede ser la del módulo de entrenamiento o una externa a través de JACK VDD.

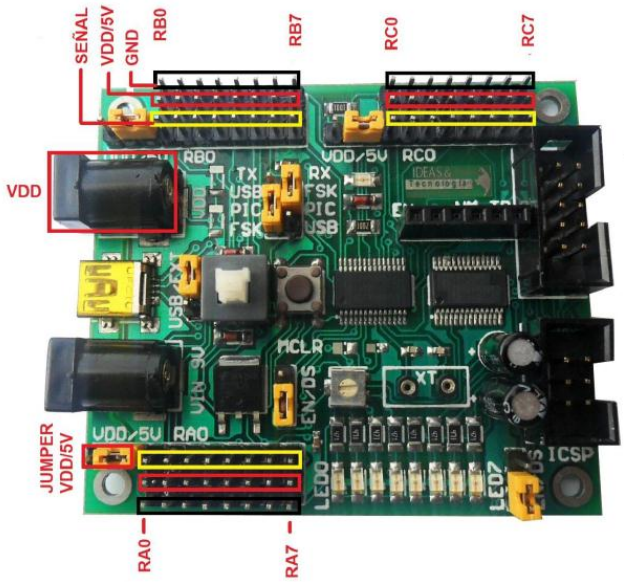

FIGURA 1.50: PUERTOS PARA CONTROL DE SERVOMOTORES DEL MÓDULO MEI&T04

Comunicación Serial UART/USB/FSK:

Este módulo nos permite realizar varias aplicaciones con comunicación serial por este motivo se le incorporo un módulo de comunicación UART-USB y de radiofrecuencia UART-FSK.

Mediante el Jumper USB-PIC-FSK podemos realizar varias selecciones para diferentes configuraciones.

PIC TX: Pin RC6 de transmisión de datos seriales UARTPIC RX: Pin RC7 de recepción de datos seriales UART. Ver referencia [8].

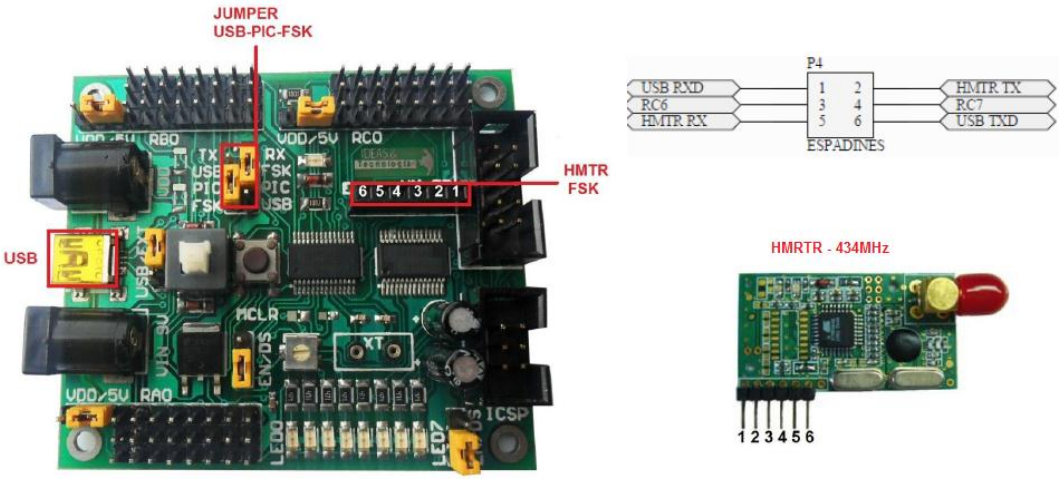

FIGURA 1.51: PUERTO USB Y CONECTOR PARA COMUNICACIÓN INALÁMBRICA

1.2.2 MÓDULO DE CONTROL DE CARGA AC I&T (RESISTIVO)

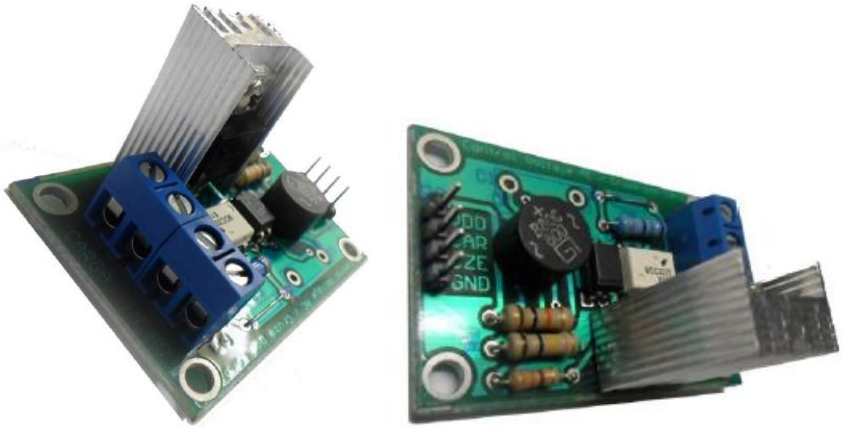

FIGURA 1.52: MÓDULO DE CONTROL DE CARGA AC RESISTIVO

El módulo Control AC Carga Resistiva nos permite controlar la potencia suministrada a una carga AC. Ver referencia [9].

Especificaciones:

- Bus de señales para control y detección cruce por cero.
- Conectores para entrada y salida 110VAC.
- Potencia máxima para carga AC de 600w.

Aplicaciones:

- Control de potencia para resistencia térmica.
- Control de luminarias incandescentes.

Características:

- Fuente de Alimentación.
- IN AC: Alimentación 110VAC.
- VDD: 5VDC.
- GND: 0V.

Señales de Control:

- CAR: Entrada disparadora para triac.
- CZE: Salida señal de sincronización de cruce por cero.

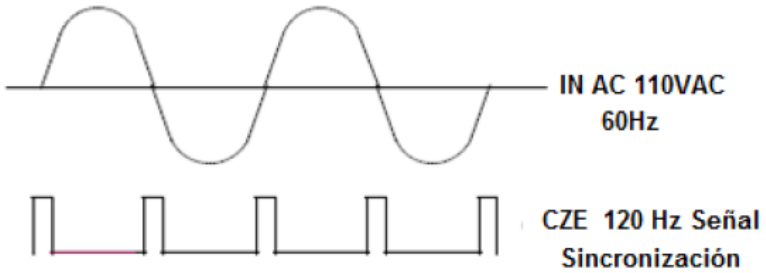

FIGURA 1.53: SEÑAL DE SINCRONIZACIÓN CRUCE POR CERO

Salidas:

- CARGA: Salida para carga AC Resistiva.

1.2.3 MÓDULO DISPARADOR DE RELÉ I&T

Módulo Disparador de Relé, sirve para el manejo de cargas de gran Potencia.

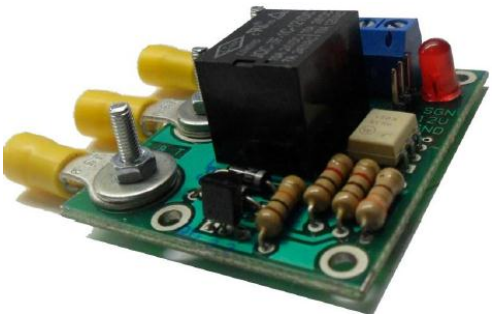

FIGURA 1.54: MÓDULO DISPARADOR DE RELÉ

Especificaciones:

- Led indicador de activación de relé.
- Alimentación independiente para relé.
- Salidas mediante terminales de potencia.

Aplicaciones:

- Encendido o Apagado de motores AC/DC.
- Control ON/OF de luces.

Características:

- Fuente de Alimentación.
- La alimentación se la puede realizar por dos maneras:
    - 12V de alimentación directa.

- 12V de alimentación por la señales de control.

Señales de Control:

- SGN: Señal TTL/CMOS para controlar la conmutación de los relés.
- EN/DS: AL seleccionar EN escogemos que la alimentación de 12V sea por el control o DS por el control.

Salidas:

- Normalmente Abierto: Cuando el Relé está en estado abierto.
- Normalmente Cerrado: Cuando el Relé está en estado cerrado.
- Común: Pin común para la referencia en la conmutación del relé.
- La carga en la salida puede ser con los siguientes parámetros: 110V/10A; 28V/10A; 24V/15A; 240V/7A. Ver referencia [10].

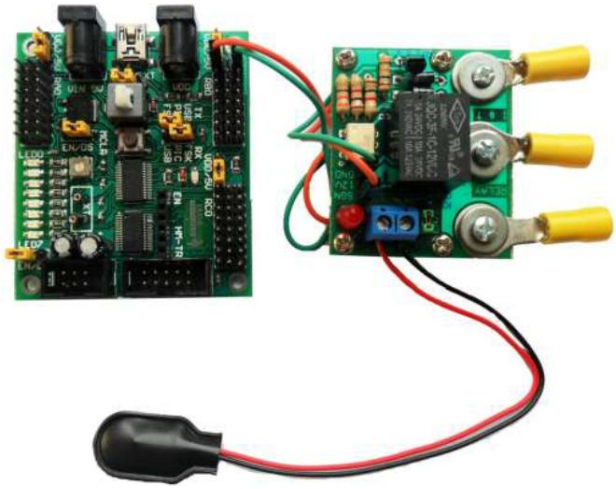

FIGURA 1.55: CONEXIÓN DEL MÓDULO DISPARADOR DE RELÉ

1.2.4 MÓDULO ENCODER ÓPTICO I&T

EL encoder que aquí se describe y que se utiliza en el presente proyecto está hecho a base de una barrera infrarroja, a este tipo de dispositivos se los llaman encoders ópticos. Están

compuestos por un emisor y un receptor de infrarrojos (IR) enfrentados a corta distancia. Ver referencia [6].

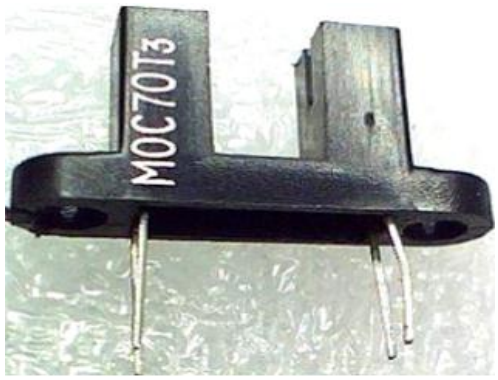

FIGURA 1.56: ENCODER MOC70T3

Especificaciones:

- Posee 1 encoder MOC70T3.
- Independiente con conector de 3 hilos tipo extensión servomotor.
- Alimentación 5VDC.

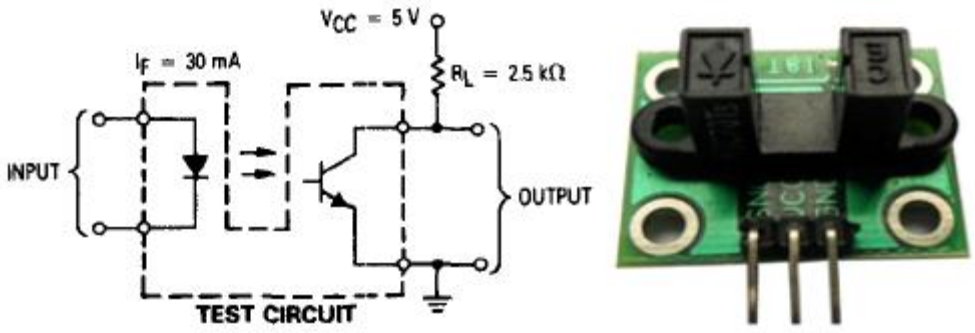

FIGURA 1.57: MÓDULO CON ENCODER MOC70T3

Aplicaciones:

- Tacómetro digital para motores.
- Sensores de fin de carrera.

Fuente de Alimentación:

- Alimentación de 5VDC.

- Señales de Control.
- -: GND Tierra.
- +: VCC Entrada de voltaje +5VDC.
- S: Señal de sensor MOC70T3.

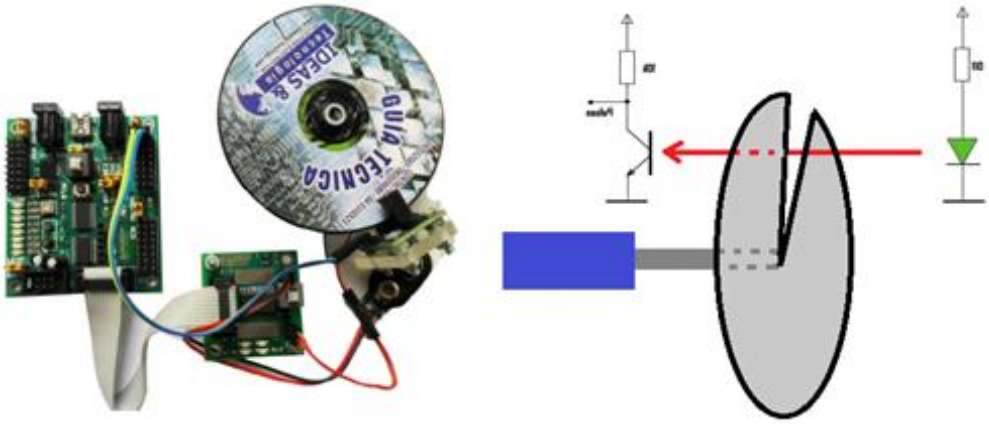

FIGURA 1.58: CONEXIÓN DEL MÓDULO ENCODER ÓPTICO

1.2.5 FUENTE UNIPOLAR 9VDC I&T

Esta fuente de voltaje nos permite suministrar dos voltajes regulados DC a saber:

- 9 VDC.
- 5VDC.

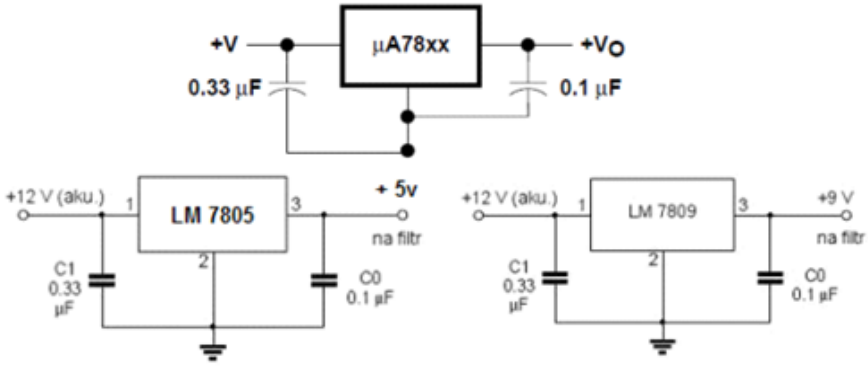

FIGURA 1.59: REGULADOR LM7805 – LM7809

Los componentes principales son los reguladores:

- LM7805, para suministrar un voltaje estable de 5VDC con una corriente máxima de 1A.
- LM7809, para suministrar un voltaje estable de 9VDC con una corriente máxima de 1A.

El encapsulado utilizado es el TO-220, el mismo que permite colocar un disipador al regulador.

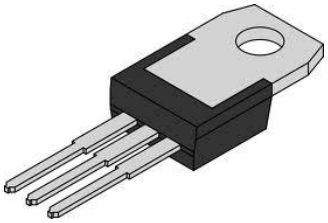

FIGURA 1.60: ENCAPSULADO TO-220

La fuente tiene ambas reguladores colocados en una misma PCB, permitiéndonos utilizar un voltaje regulado de 5VDC o 9VDC. Ver referencia [4]

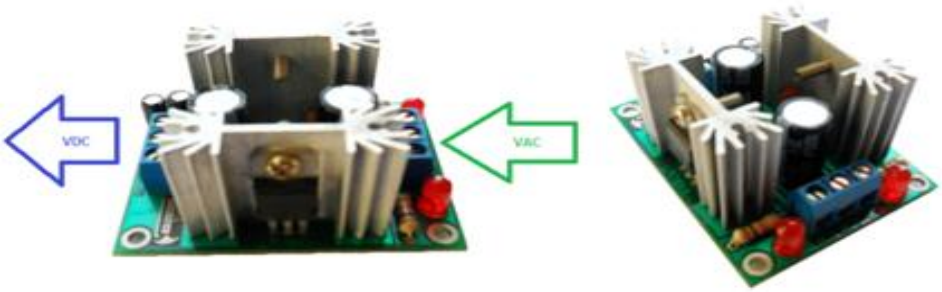

FIGURA 1.61: ENCAPSULADO TO-220

1.2.6 MÓDULO LDR I&T

LDR (LIGHT DEPENDENT RESISTORS) o también conocida con el nombre de Foto celda o Foto resistor, son sensores muy utilizados en detección de intensidad de

luz ambiental es decir actúan dependiendo el entorno luminoso en el que se encuentra este sensor.

El LDR es un tipo de sensor foto resistivo formado por un área sensible a la luz o luminosidad, contiene una pista de Sulfuro de Cadmio, y dos terminales de conexión, cuya resistencia óhmica entre estas terminales cambia dependiendo de la intensidad de luz o luminosidad incidente sobre el área sensible que la percibe.

En la gráfica siguiente se puede observar en la parte superior izquierda cómo está construida el área sensible del LDR y en el extremo superior derecho está el símbolo eléctrico que se utiliza para identificar a este elemento.

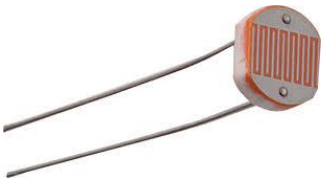

FIGURA 1.62: SENSOR LDR

La resistencia del LDR varia de forma inversamente a la cantidad de luz incidente que cae sobre ella, en la parte inferior de la imagen anterior se puede ver de forma gráfica esta relación, en el eje vertical de la Resistencia entre sus terminales y en el eje Horizontal la cantidad de luz incidente (luxes).

Habiendo analizado la gráfica podemos concluir que a mayor cantidad de luz menor es la resistencia entre sus terminales, esta resistencia llega a tener valores del orden de décimas de ohmio, dependiendo del LDR.

En total oscuridad se presenta una resistencia entre terminales del orden de las centenas de miles de ohmios.

Especificaciones:

- Posee 1 sensor LDR.
- Independiente con conector de 3 hilos tipo extensión servomotor.
- Alimentación 5VDC.

Aplicaciones:

- Seguidor Solar para paneles fotovoltaicos.
- Sensor crepuscular.
- Activador automático de encendido de luz de jardín.

Características:

- Fuente de Alimentación de 5VDC.
- Señales de Control.
- **S**: Señal de la LDR en divisor de tensión.
- **+**: VCC Entrada de voltaje +5VDC.
- **-**: GND Tierra.

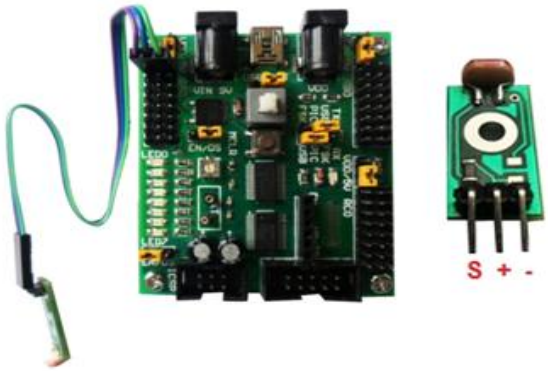

FIGURA 1.63: CONEXÍON DEL MÓDULO LDR

1.2.7 MÓDULO SENSOR DE TEMPERATURA I&T

Uso del sensor de temperatura ds18b20 que opera en el rango de -55°C a +125°C, usando la comunicación one-wire.

Este sensor de temperatura utiliza comunicación por un solo cable o PIN, es necesario consultar la hoja de especificaciones para una mayor información.

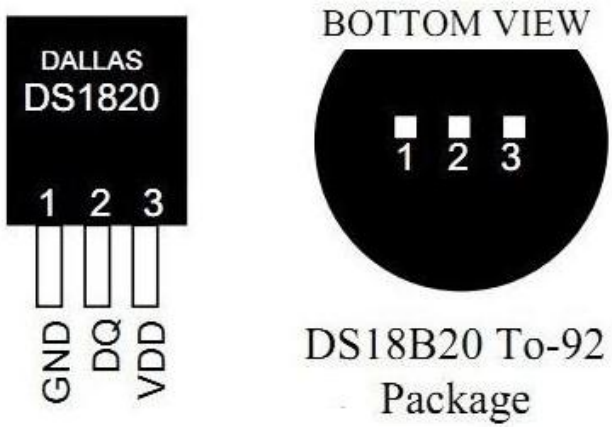

FIGURA 1.64: SENSOR DE TEMPERATURA DS18B20

Sensor fabricado por Maxim es un pequeño sensor de temperatura con una precisión de 9 a 10 bits, nos permite medir temperatura en un amplio margen desde -55C a 125C con un margen de error de tan solo 0.5 grados. No requiere de componentes externos para su funcionamiento, es decir no necesita de un módulo de digitalización ya que internamente digitaliza la temperatura y la envía en protocolo ONE WIRE.

FIGURA 1.65: MÓDULO SENSOR DE TEMPERATURA

Cada sensor incorpora de fábrica un número de serie de 64 bits que permite conectar múltiples sensores en paralelo usando sólo una patilla como bus de datos.

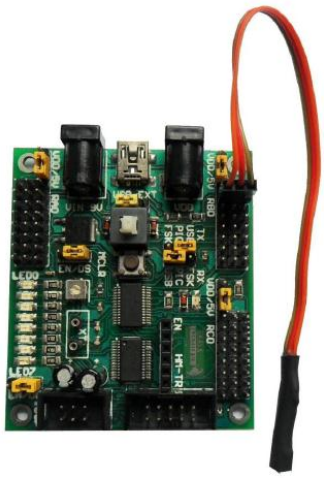

FIGURA 1.66: CONEXIÓN DEL MÓDULO SENSOR DE TEMPERATURA

1.3 RECURSOS DE SOFTWARE DISPONIBLES

Los recursos de software que se utilizan tienen mucho que ver con software de programación de microcontroladores y con software de procesamiento de datos que tengan interfaz de comunicación serial para la adquisición de datos a analizar.

1.3.1 MIKROBASIC:

FIGURA 1.67: SOFTWARE MIKROBASIC

Es una poderosa herramienta de desarrollo que dispone de muchas herramientas para PIC microcontroladores. Está diseñado para proporcionar al usuario la más fácil solución para el desarrollo de aplicaciones para sistemas embebidos, sin comprometer el rendimiento o el control. MikroBasic implementa un número predefinido de Variables globales y

constantes. Todos los Registros SFR del PIC están declarados implícitamente como variables globales del tipo byte, y son visibles en todo el proyecto.

Cuando se crea un proyecto, MiKrobasic incluirá el archivo apropiado con extensión .def, el cual contiene las declaraciones disponibles del SFR y constantes (como PORTB, TMR1, etc.). Para el set completo de constantes y globales predefinidas, Busque "DEFS" en el directorio de instalación de MikroBasic, o pruebe el asistente de código para letras especificas (ctrl.+espacio en el editor de código).

DIRECTIVAS PARA EL COMPILADOR:

- Program< >: Es una palabra reservada por el compilador, va siempre al principio del programa y sirve para indicarle al compilador el comienzo de las instrucciones ejem: program MyProgram.

- Include< >: Esta directiva se la emplea para añadir librerías o módulos externos donde existan funciones a ser utilizadas desde el programa principal, además siempre va después de la directiva Program<> Ejemplo: include "utils".

- Symbol < >: Se la utiliza para reemplazar alguna línea de código con algún identificador alias. Ejemplo: symbol MYDELAY = Delay_ms(1000).

- Const < >: Se la utiliza para declarar constantes, las cuales no se pueden modificar sus contenidos, el tipo de dato es opcional. const MIN = 1000 , const MAX as longint = 10000.

- Dim < > as < >: Permite la declaración de variables con su respectivo tipo de dato. Ejemplo: dim i, j, k as byte.

- Procedure: Sirve para declarar un procedimiento o subrutina a realizar. Ejemplo: sub procedure procedure_name (parameter_list).

  [local declarations] , procedure body

  end sub

- Parameter_list: Parámetros del procedimiento los cuales pueden o no existir, sin existen deben de ser por referencia.

- Function: Es similar a procedure con la diferencia que esta si retorna un valor y además sus parameter_list no son por referencia. Ejemplo: sub function function_name(parameter_list) as return_type.

  [local declarations]

  function body

  end sub

- Main: Es una palabra reservada por el compilador la cual indica el comienzo del programa principal (código).

  Ejemplo: main: ' Place program code here

  end.

TIPOS DE DATOS:

Type	Size	Range
bit	1-bit	0 or 1
sbit	1-bit	0 or 1
byte, char	8-bit	0 .. 255
short	8-bit	-127 .. 128
word	16-bit	0 .. 65535
integer	16-bit	-32768 .. 32767
longword	32-bit	0 .. 4294967295
longint	32-bit	-2147483648 .. 2147483647
float	32-bit	$\pm 1.17549435082 * 10^{-38}$ .. $\pm 6.80564774407 * 10^{38}$

TABLA 1.3: TIPOS DE DATOS USADOS EN MIKROBASIC

Arrays:

```
dim weekdays as byte[7]     '1-dimensional array of size 7
dim samples as word[50]     '1-dimensional array of size 50
dim m as byte[50][20]       '2-dimensional array of size 50x20
const MONTHS as byte[12] = (31,28,31,30,31,30,31,31,30,31,30,31)
const NUMBER as byte[4][4] = ((0, 1, 2, 3), (5, 6, 7, 8), (9, 10, 11,12), (13,14, 15, 16))
```

Structures:

```
structure structname
   dim member1 as type1 ...
   dim membern as typen
end structure
```

Ejem:

```
structure Dot
   dim x as float
   dim y as float
end structure
```

1.3.2 MATLAB:

FIGURA 1.68: SOFTWARE MATLAB

MATLAB es el nombre abreviado de "MATrixLABoratory".

MATLAB es un programa para realizar cálculos numéricos con vectores y matrices. Puede trabajar con números escalares, tanto reales como complejos.

Realiza una amplia variedad de gráficos en 2D y 3D. MATLAB tiene también un lenguaje de programación propio.

Posee Toolboxes de apoyo a diferentes especialidades. Cada uno de ellos, supone una interpretación diferente del concepto de Vector ó Matriz. Ver referencia [3].

Formatos numéricos:

Los formatos de salida en la ventana principal de MATLAB se pueden controlar fácilmente a partir del cuadro de diálogo que se abre con el comando Preferences del menú File luego en Variable Editor. Respecto a los formatos numéricos con que MATLAB muestra los resultados (siempre calcula con la máxima precisión), se pueden activar por medio de comandos tecleados en la línea de comandos de MATLAB. Los más importantes de estos comandos son:

- **format short:** coma fija con 4 decimales (defecto).
- **format long:** coma fija con 15 decimales.

- **format hex:** cifras hexadecimales.
- **format bank:** números con dos cifras decimales.
- **format short e:** notación científica con 4 dec.
- **format short g:** notación científica o decimal, dependiendo del valor.
- **format long e:** notación científica o decimal, dependiendo del valor.
- **format loose:** introduce algunas líneas en blanco en la salida (defecto).

Funciones de Biblioteca:

- MATLAB tiene un gran número de funciones incorporadas. Algunas son funciones intrínsecas, esto es, funciones incorporadas en el propio código ejecutable del programa. Estas funciones son particularmente rápidas y eficientes.
- Existen además funciones definidas en ficheros *.m y *.mex que vienen con el propio programa o que han sido aportadas por usuarios del mismo.
- Una función tiene nombre, valor de retorno y argumentos. Una función se llama utilizando su nombre en una expresión o utilizándolo como un comando más.
- Las funciones se definen en ficheros de texto *.m en la forma que se verá más adelante.
- Una diferencia importante con otros lenguajes es que en MATLAB las funciones pueden tener valores de retorno matriciales múltiples.
- Una característica de MATLAB es que las funciones que no tienen argumentos no llevan paréntesis, por lo que a simple vista no siempre son fáciles de distinguir de las simples variables.

Programación:

MATLAB es una aplicación que permite programar con facilidad.

MATLAB posee un lenguaje de programación que dispone de sentencias para realizar bifurcaciones y bucles. Las bifurcaciones permiten realizar una u otra operación según se cumpla o no una determinada condición. La Figura muestra tres posibles formas de bifurcación. Además se muestran lazos (o bucles) con control al principio y al final.

Scripts; En MATLAB se puede escribir un script, con cuales quiera instrucciones válidas y agruparlas en un fichero .m

Al salvar el fichero, se pueden invocar todas estas instrucciones desde la ventana de comandos utilizando el nombre con que se salvó el fichero.

Para crear un fichero debe ejecutar:

- File->New -> M-File

1.3.2.1 ADQUISICIÓN DE DATOS

Es una lectura y escritura interactiva de variables. La función input permite imprimir un mensaje en la línea de comandos de MATLAB y recuperar como valor de retorno un valor numérico o el resultado de una expresión tecleada por el usuario.

Cualquier expresión válida de MATLAB es aceptada por este comando.

- n = input('Teclee el número de ecuaciones').

Otra posible forma de esta función es la siguiente (obsérvese el parámetro ’s’):

- nombre = input('¿Cómo te llamas?','s').

La función Disp permite imprimir en pantalla un mensaje de texto o el valor de una matriz, pero sin imprimir su nombre. En realidad, disp siempre imprime vectores y/o matrices: las cadenas de caracteres son un caso particular de vectores.

Considérense los siguientes ejemplos de cómo se utiliza:

- » disp('El programa ha terminado')
- » A=rand(4,4);
- » disp(A)

Funciones:

- Este concepto permite definir funciones análogas a las de MATLAB:
    - nombre,
    - argumentos.
    - valores de retorno.
- Los ficheros *.m que definen funciones permiten extender las posibilidades de MATLAB;
- Existen bibliotecas (ficheros *.m) que se venden (toolkits) o se distribuyen gratuitamente.

Estructura de definición:

- Las funciones se definen en ficheros *.m.
- Function.
    - [valores de retorno, separados por comas].
    - el signo igual (=).
    - nombre de la función.
    - (argumentos entre paréntesis y separado por comas).

Características:

- Un fichero *.m puede llamar a otros ficheros *.m,

- Puede llamarse a sí mismo de forma recursiva.
- Los ficheros de comandos se pueden llamar también desde funciones.
- Las variables que se crean pertenecen a espacio de trabajo de la función.
- El espacio de trabajo de una función es independiente del espacio de trabajo base y del espacio de trabajo de las demás funciones.
- Puede haber funciones: Sin valor de retorno y sin argumentos.

Recuérdese que los argumentos son los datos de la función y los valores de retorno sus resultados.

- Si no h ay valores de retorno se omiten los corchetes y el signo igual (=);
- Si sólo hay un valor de retorno no hace falta poner corchetes.
- No ha ce falta poner paréntesis si no hay argumentos.
- En MATLAB una función no modifica nunca los argumentos que recibe.
- Los resultados se obtienen a través de los valores de retorno.
- Tanto el número de argumentos como el de valores de retorno no tienen que ser fijos, dependiendo de cómo el usuario llama a la función.

Funciones FSCANF, SSCANF, FPRINTF y SPRINTF:

- Permiten leer y escribir en ficheros ASCII, es decir, en ficheros formateados.
- La forma general de la función fscanf es la siguiente:
    - [var1, var2,...] = fscanf (fi, ’cadena de control’,size), donde:
    - fi es el identificador del fichero (devuelto por la función fopen),
    - size es un argumento opcional que puede indicar el tamaño del vector o matriz a leer.

- La cadena de control va encerrada entre apóstrofos simples, y contiene los especificadores de formato para las variables:
    - %s para cadenas de caracteres.
    - %d para variables enteras.
    - %f para variables de punto flotante.
    - %lf para variables de doble precisión.
- La función fprintf dirige su salida formateada hacia el fichero indicado por el identificador. Su forma general es:
    - Fprintf (fi,'cadena de control', var1, var2,...).

1.3.2.2 TOOLBOX IDENT DE MATLAB PARA LA IDENTIFICACIÓN DE SISTEMAS

- datos= [XT FT] % Configuración de los datos. Se coloca primera la variable de salida XT y después la variable de entrada FT. Deben tener el mismo tamaño.
- tam= length (FT) % Cantidad de datos de la variable de entrada FT.
- datos_ident= [XT(1:60) FT(1:60)] % Cantidad de datos tomados para la validación del sistema. Para este caso, se toman los siguientes 60 datos tanto de entrada como de salida.
- datos_val= [XT(61:tam) FT(61:tam)] % Cantidad de datos tomados para la identificación del sistema. Para este caso, se toman los primeros 61 datos tanto de entrada como de salida.
- idplot(datos_ident) % Visualizar los datos tomados para identificación.

- datos_ident= dtrend(datos_ident) % % Remueve las tendencias lineales de los datos de identificación, manteniendo la información de la dinámica del sistema, pero no su comportamiento estático.
- datos_val= dtrend(datos_val) % Remueve las tendencias lineales de los datos de validación, manteniendo la información de la dinámica del sistema, pero no su comportamiento estático.
- idplot(datos_ident) % Visualiza los datos de identificación sin tendencia lineal.
- idplot(datos_val) % Visualiza los datos de validación sin tendencia lineal.
- th=arx(datos_ident,[2 7 6]) % Aplicación del modelo posible. Para este caso es ARX. Puede ser ARMAX, OE y BJ. Se debe tener presente los parámetros que maneja cada uno.
- Discrete-time IDPOLY model: A(q)y(t) = B(q)u(t) + e(t)
- th=sett(th,300) % Representación del modelo en términos de q-1, con el tiempo de muestreo del sistema.
- present(th) % Presenta el modelo obtenido en q-1.
- [numd1,dend1]=th2tf(th) % Transforma los polinomios en format q-1 en expresiones numerador y denominador. Presenta cada coeficiente de los polinomios obtenidos.
- roots(dend1) % Se encuentran las raíces del polinomio denominador para ubicación de los polos.
- compare(datos_val,th) % Compara los datos de validación con el modelo obtenido. En la gráfica obtenida se muestra una comparación entre las salidas de los modelos simulados y la salida medida cuando son aplicados los datos de validación. sysd=tf(numd1,dend1,300).

Transfer function:

$$\frac{0.1531\,z^6 + 0.07232\,z^5 + 0.02384\,z^4 + 0.05164\,z^3 + 0.1027\,z^2 - 0.008651\,z - 0.03379}{z^{12} - 0.3144\,z^{11} - 0.3001\,z^{10}}$$

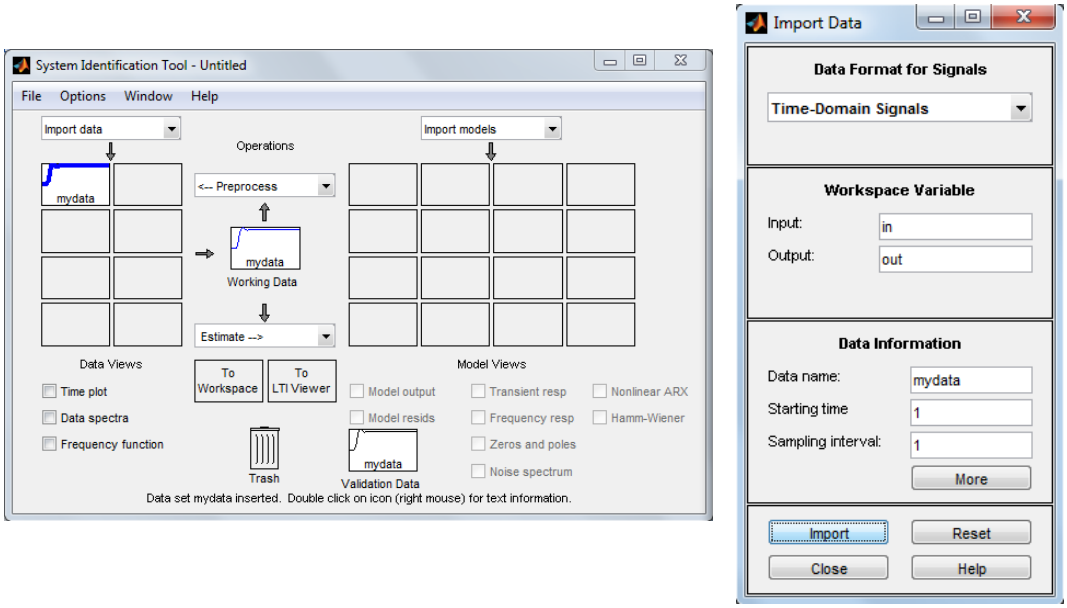

FIGURA 1.69: TOOLBOX IDENT DE MATLAB

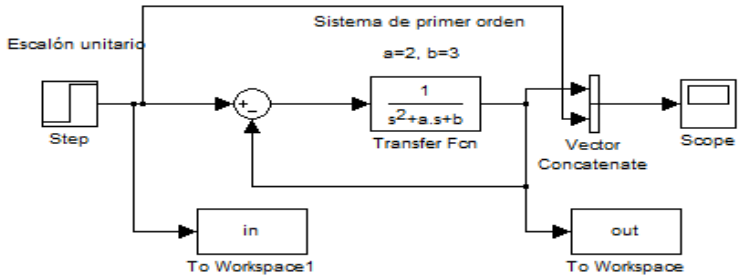

FIGURA 1.70: SIMULINK DE MATLAB

MODELOS DE ESTIMACIÓN PARAMÉTRICOS:

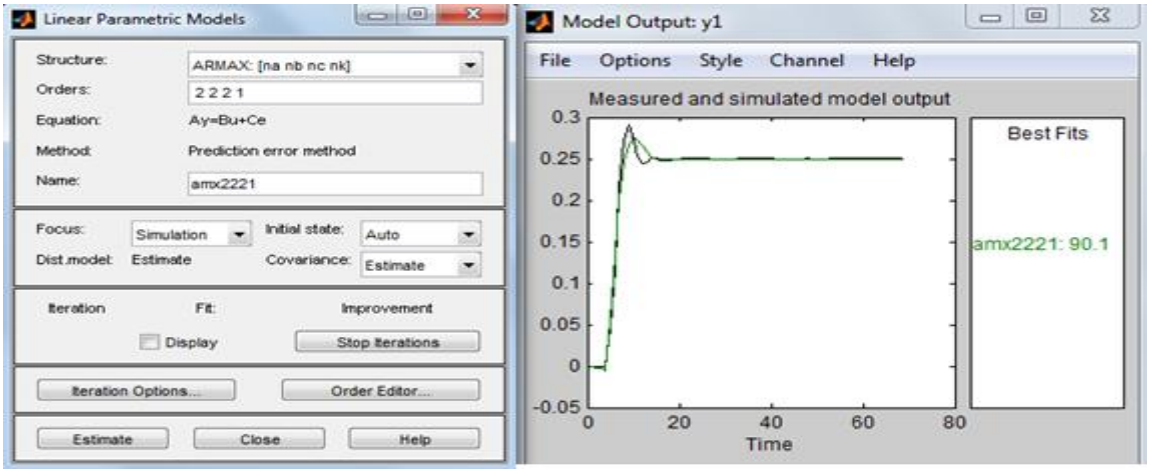

FIGURA 1.71: MODELOS DE ESTIMACIÓN PARAMÉTRICOS

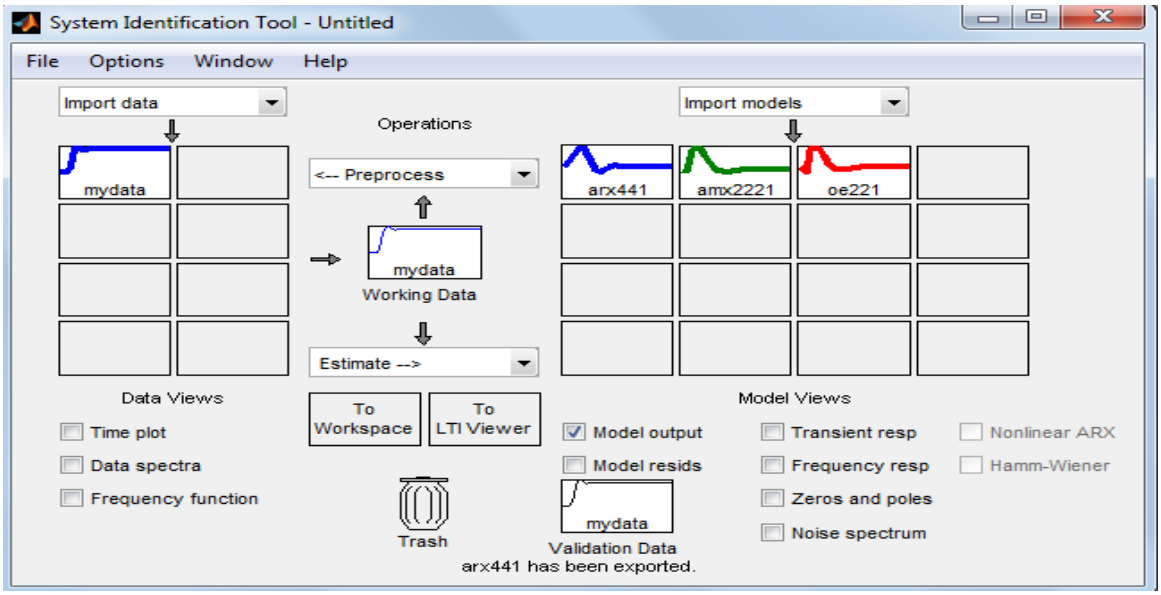

FIGURA 1.72: ENVÍO AL WORKSPACE DEL MODELO PARAMÉTRICO SELECCIONADO

```
>> G=tf(d2c(zpk(arx441)))
```

Transfer function from input "u1" to output "y1":

$$\frac{0.1103\ s^4 - 0.8686\ s^3 + 2.383\ s^2 - 2.128\ s + 3.973}{s^5 + 6.534\ s^4 + 25.84\ s^3 + 38.69\ s^2 + 31.6\ s + 15.88}$$

1.3.2.3 TOOLBOX SISOTOOL DE MATLAB PARA EL DISEÑO DE CONTROLADORES CLÁSICOS

Sisotool es una herramienta gráfica que permite el análisis de sistemas lineales para obtener y analizar el lugar de las raíces de un sistema.

- Primero definimos la función de transferencia del sistema:

```
>> s = tf('s'); G = 1/(s+2),
```

Transfer function:

```
     1
   -----
   s + 2
```

```
>> sisotool(G)

>>
```

La ventana del sisotool nos muestra a la izquierda el lugar de las raíces del sistema G(s) cuando lo realimentamos. A la derecha nos muestra el diagrama de Bode en cadena abierta, tanto de amplitud como de fase, correspondiente a la ganancia indicada en la ventana 'Current compensator' ('controlador actual').

Los polos y ceros del sistema en cadena abierta se muestran como " × " y " o " respectivamente. Los polos en cadena cerrada se muestran como cuadrados " □ ". En el ejemplo de la figura, realimentando el sistema G(s), con un regulador proporcional de ganancia uno ('Current compensator'), nos daría un polo en cadena cerrada en -3. El lugar de las raíces es una línea que parte del polo en cadena abierta y termina en $-\infty$.

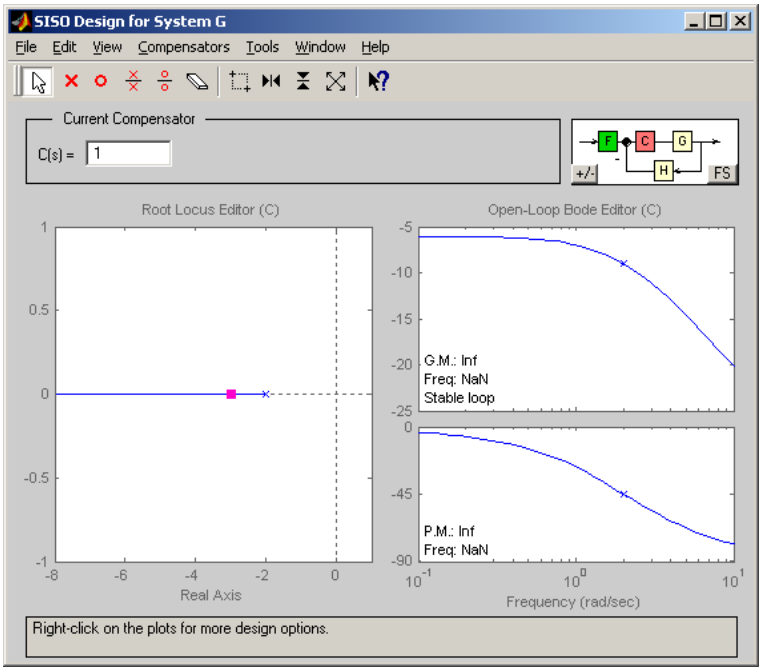

FIGURA 1.73: TOOLBOX SISOTOOL DE MATLAB

Modificación de la ganancia del controlador: Si queremos ver qué posición toman los polos en cadena cerrada cuando variamos la ganancia lo podemos hacer cambiando el valor de la ganancia del controlador. En este caso, al variar la ganancia a 3.49, cambia automáticamente la posición del polo en cadena cerrada. Otra opción es arrastrar el polo en cadena cerrada hasta la posición deseada, y que sisotool calcule el valor de la ganancia para que en cadena cerrada el polo tome esa posición.

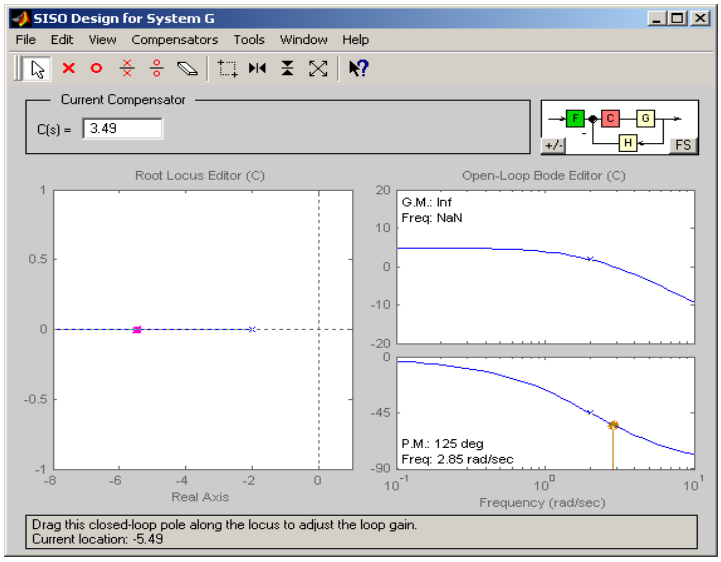

FIGURA 1.74: GANACIA DEL CONTROLADOR EN SISOTOOL

Representación factorizada del controlador: Como puede verse, en la parte superior de las figuras anteriores aparece la función de transferencia del controlador para la situación actual definida para la ganancia, polos y ceros del controlador, de la que resultan los polos en lazo cerrado descritos por los cuadrados " □ " de color fucsia. Una opción interesante es hacer que la función de transferencia del controlador aparezca en formato ceros-polos-ganancia, donde numerador y denominador aparecen factorizados y en formato Mónico (común cuando se utiliza el lugar de las raíces). Esto puede hacerse como se muestra en las siguientes figuras.

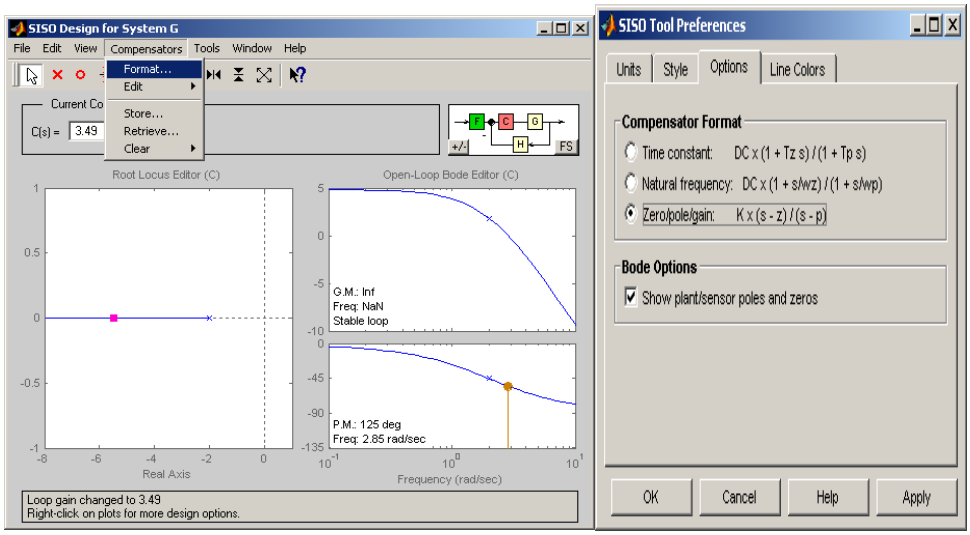

FIGURA 1.75: REPRESENTACIÓN FACTORIZADA DEL CONTROLADOR EN SISOTOOL

REPRESENTACIÓN FACTORIZADA DEL CONTROLADOR EN SISOTOOL

Visualización de respuestas del sistema: La herramienta sisotool también nos permite obtener las respuestas en lazo cerrado del sistema realimentado resultante. Hay varias posibilidades, pudiendo definirse cada una de eligiendo la entrada (referencia, perturbación, etc.), la salida (acción de control, señal de error, salida del proceso) así como el tipo de representación (Bode, respuesta al escalón, respuesta impulsional, Nyquist, etc.). En la figura siguiente se muestra la forma de obtener la respuesta al escalón en bucle cerrado:

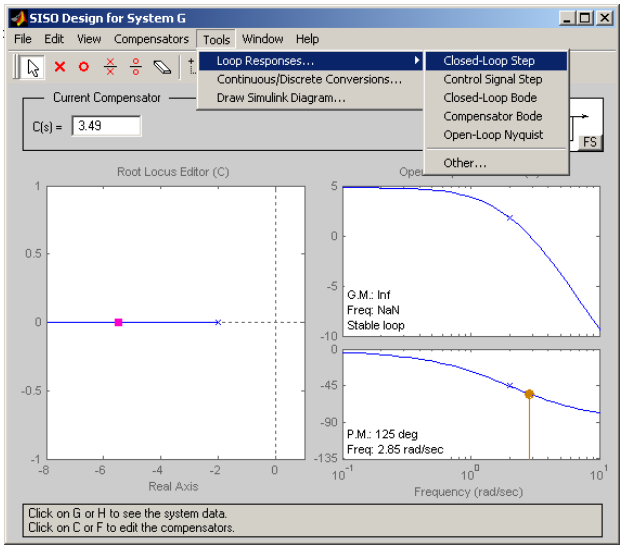

FIGURA 1.76: RESPUESTA DEL SISTEMA EN EL SISOTOOL

Edición numérica del controlador: También es posible editar directamente el controlador, definiendo numéricamente la ganancia, los polos y los ceros accediendo a la opción edit compensator. A esta ventana se puede acceder haciendo doble click sobre el área que contiene al regulador.

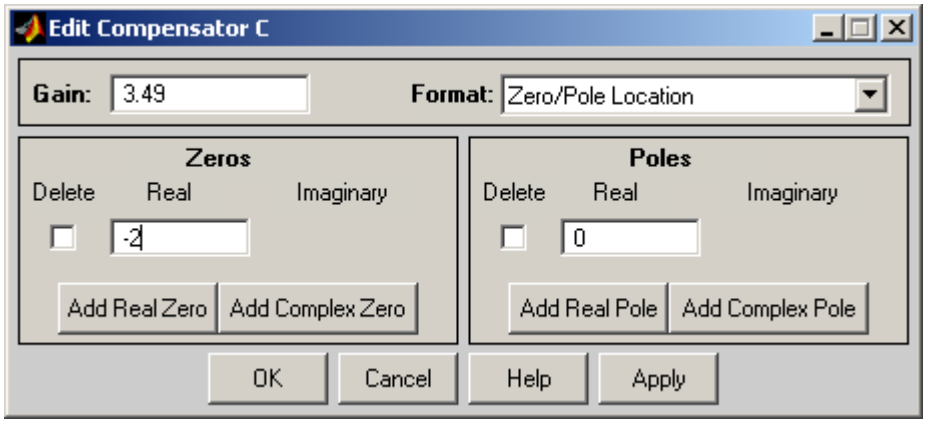

FIGURA 1.77: EDICIÓN DEL CONTROLADOR EN EL SISOTOOL

Selección interactiva del controlador: En la figura de abajo se muestra, finalmente, el lugar de las raíces para el controlador (ventana de la izquierda) junto con la respuesta al escalón (ventana de la derecha). En este punto, es posible "arrastrar" las raíces (cuadrados fucsia) por el lugar, lo que produce un cambio automático en la ganancia del controlador. También se pueden arrastrar el polo y el cero del controlador, lo que produce una modificación en la geometría del lugar de las raíces. En ambos casos, los "arrastres" producen también el cambio de la respuesta al escalón en el LTI viewer (ventana de la derecha). Además, en el LTI viewer, es posible obtener propiedades de la respuesta (valores de pico, valor en régimen permanente, etc.) pulsando en el botón derecho del ratón. Si hubiésemos definido cualquier otra respuesta (respuesta en frecuencia de una función de sensibilidad, por ejemplo, también reflejaría de forma simultánea los cambios). Esto permite analizar de forma interactiva las consecuencias de los cambios en nuestro diseño.

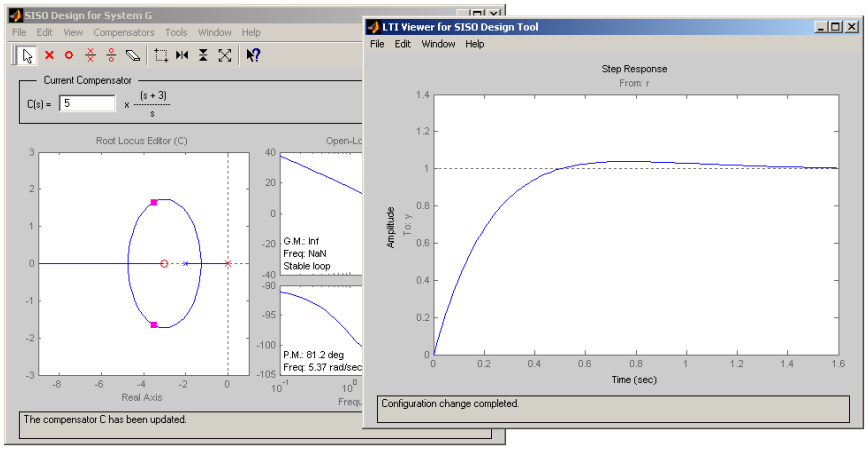

FIGURA 1.78: EDICIÓN INTERACTIVA DEL CONTROLADOR

1.3.2.4 TOOLBOX FUZZY DE MATLAB PARA EL DISEÑO DE CONTROLADORES INTELIGENTES FUZZY

Acceso:

Para acceder al toolbox fuzzy se debe digitar la palabra fuzzy en la línea de comandos y luego oprimir enter. En el caso de encontrar un error, por no hallarse cargado el toolbox se debe agregar el CD de instalación de Matlab. El menú al cual se debería acceder es el siguiente:

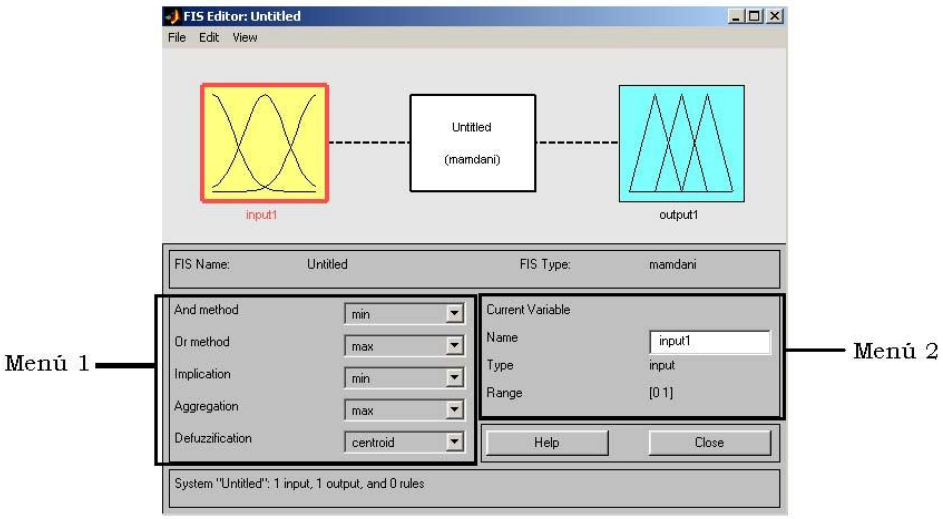

FIGURA 1.79: MENÚ PRINCIPAL DEL FUZZY TOOLBOX, FIS EDITOR

En el Menú 1, se podrá modificar los métodos de los operadores lógicos and y or, los métodos de implicación, de agregación y de defuzificación.

En el Menú 2, se podrá cambiar el nombre de la variable que se encuentre seleccionada, por ejemplo, modificar el nombre “input1” por “flujo de agua”.

Elección del Modelo:

Para elegir el tipo de modelo a usar, Sugeno o Mamdani, se debe acceder al menú File -> New FIS… -> Mamdani (Sugeno).

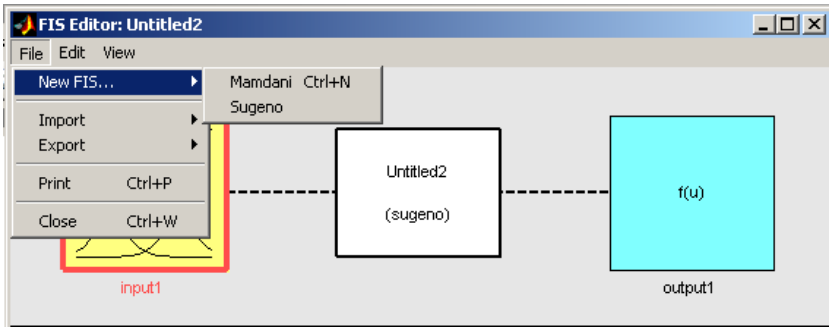

FIGURA 1.80: ELECCIÓN DEL MODELO

Variables y Funciones de Pertenencia:

Para agregar alguna variable, ya sea de entrada o de salida, se debe seleccionar el menú Edit -> Add Variable -> Input (Output).

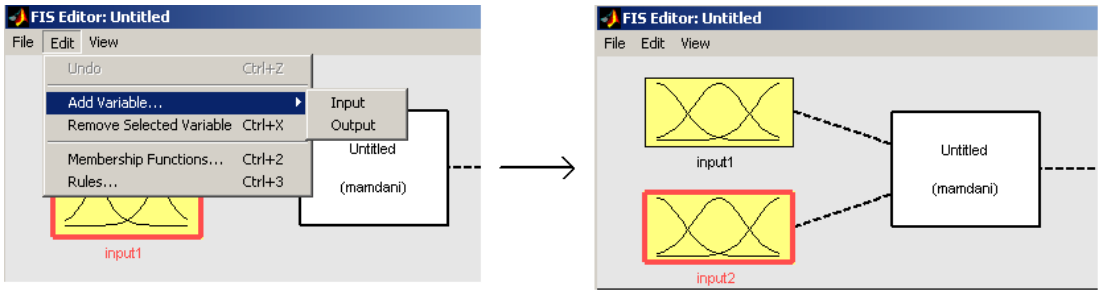

FIGURA 1.81: AGREGANDO UNA VARIABLE EN EL MENÚ GRÁFICO

Las funciones de pertenencia, tanto para las variables de entrada como para las de salida, se modifican en un menú especial Membership Function Editor que aparece al hacer doble click en la variable de interés.

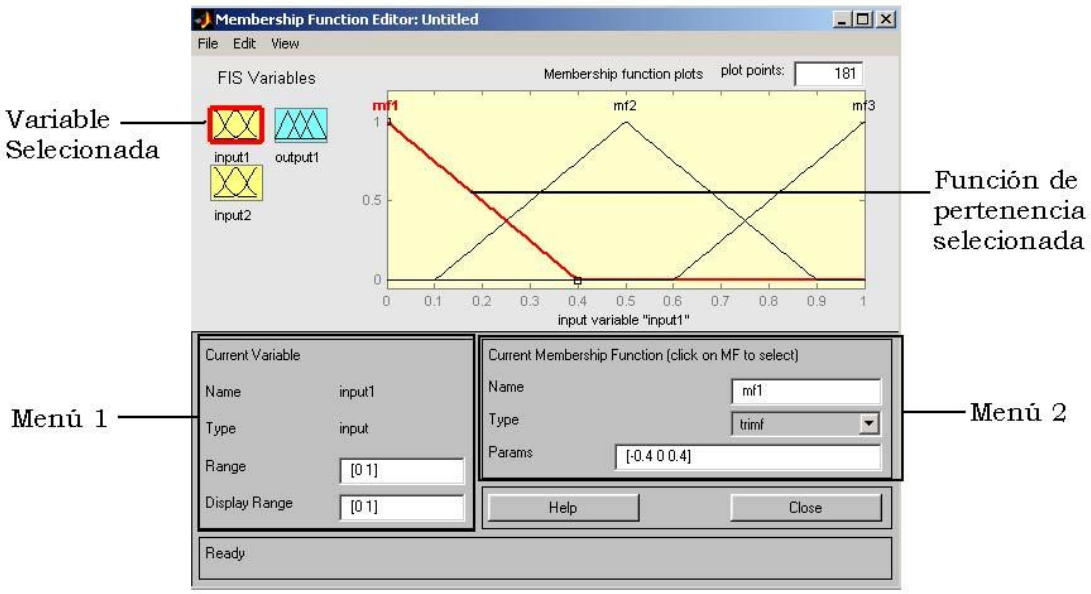

FIGURA 1.82: EDITOR DE FUNCIONES DE PERTENENCIA, MEMBERSHIP EDITOR

En el Menú 1, se puede modificar el rango de la función de pertenencia, en el cual la estará definida.

En el Menú 2, es posible modificar el nombre de la función de pertenencia, los parámetros de la función de pertenencia y también su forma, la cual está seleccionada triangular en este caso, siendo ésta la más común.

Reglas del Modelo:

Para poder modificar las reglas del modelo se debe acceder al Rule Editor, haciendo doble click sobre el modelo.

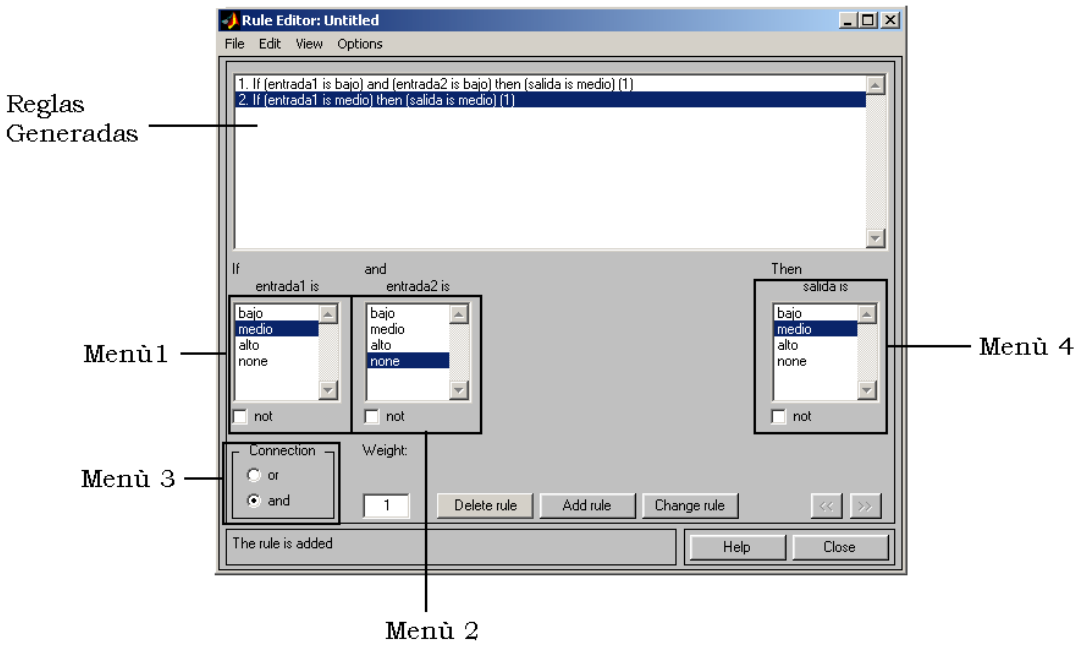

FIGURA 1.83: EDITOR DE REGLAS, RULE EDITOR

Según el número de variables de entrada y salida que existan y sus funciones de pertenencia será el número de reglas que es posible generar. En el Menú 1 se selecciona el valor que toma la primera variable de entrada, en el Menú 2, el valor que toma la segunda variable de entrada (si es necesario es posible negarla marcando not). En el Menú 3, se selecciona el tipo de conexión lógica entre ambos valores seleccionados (and, or), finalmente, en el Menú 4, se selecciona la salida que deberá entregar el controlador para los valores de entrada ya indicados. Luego, se presiona el botón Add rule, y la regla es agregada.

Para Eliminar una regla basta seleccionarla y apretar el botón Delete rule. Para modificarla se debe hacer click en el botón Change rule.

Implementación:

- Para poder implementar el controlador es necesario guardar el trabajo realizado en 1 a 4, con el menú File -> export to… -> Disk (del FIS Editor), guardando así el trabajo. Luego, es necesario importar el archivo al workspace, para que luego Matlab lo pueda reconocer y pueda ser implementado en Simulink, para eso se debe acceder al menú File -> export to… -> workspace.
- Si se desea trabajar con un modelo ya guardado se debe importar desde el menú
- fuzzy primero, y luego exportarlo al workspace.
- Luego de esta etapa, se debe cargar el controlador en Simulink, lo cual se hace dentro de un bloque llamado fuzzy logic controller.

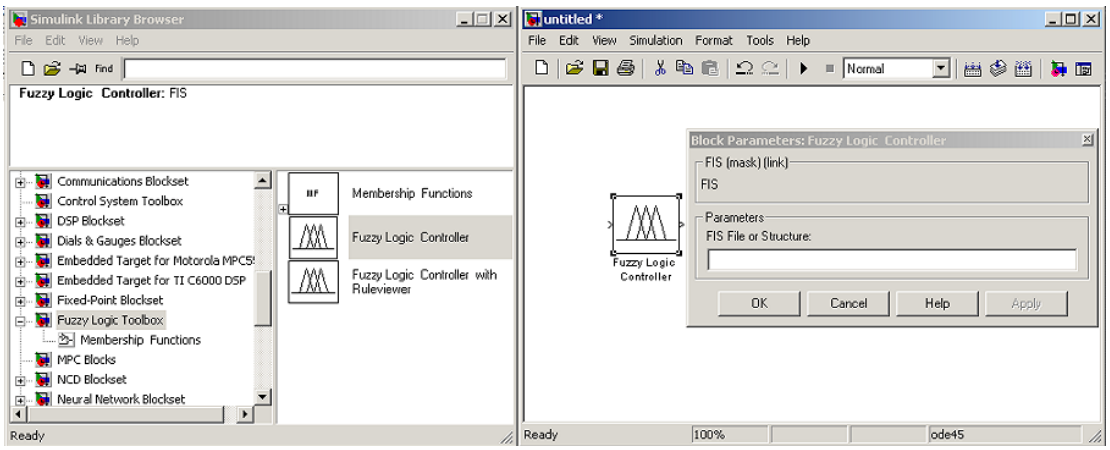

FIGURA 1.84: BLOQUE FUZZY LOGIC CONTROLLER

Por último, para implementar el controlador difuso, se hace doble click en el bloque Fuzzy Logic Controller, y se escribe el nombre del archivo con que se guardó el controlador.

1.4 TÉCNICAS DISPONIBLES

- Paso 1.- Diseñar el experimento de identificación. Determinar limites, entradas, salidas, perturbaciones, sensores, adquisición de datos, tiempos de muestreo, apoyo gerencial y operativo.
- Paso 2.- Pre procesamiento de señal. Debe eliminarse de los datos de entrada y salida las perturbaciones y no linealidades antes de ser usados en el algoritmo de identificación.
- Paso 3.- Elección del conjunto de modelos. Escoger el óptimo dentro de este conjunto.
- Paso 4.- Validación. Verificación si el modelo seleccionado tiene uso en la aplicación propuesta (control, predicción, análisis).

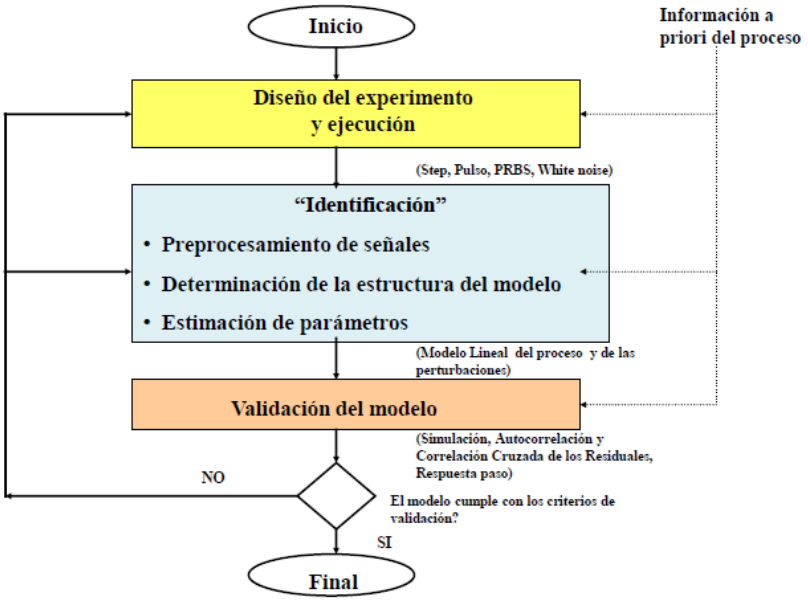

FIGURA 1.85: TÉCNICAS DISPONIBLES

CAPÍTULO 2

DISEÑO DE LA SOLUCIÓN

2.1 DISEÑO DE LA PLANTA DE ADQUISICIÓN Y CONTROL:

La estructura de la planta de adquisición consiste en los siguientes elementos:

- Fuente de alimentación para todos los circuitos electrónicos.
- Tarjeta principal con microcontrolador.
- Planta con moto DC, que deberá incluir encoder óptico y puente H.
- Planta con una fuente generadora de calor, que deberá incluir sensor de temperatura, sensor de luz y dimerizador de carga AC.

FUENTE DE ALIMENTACIÓN:

La fuente de energía suministra voltaje DC a los distintos módulos a utilizar. Esta fuente DC deberá alimentar al módulo principal con el microcontrolador (MEI&T04), 3 sensores (LDR, DS18B20, Encoder Óptico), driver puente h (Puente H I&T04) y el

motor DC, driver para control AC con carga resistiva y el dicroico. Para lo cual se usará una fuente DC de 12 voltios a 2A.

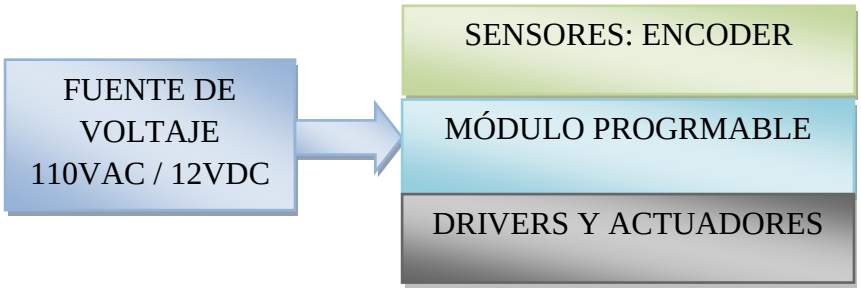

FIGURA 2.1: FUENTE DE ALIMENTACIÓN

PLANTA CON MOTOR DC:

Esta planta utiliza un actuador de motor DC de 200W con un torque de 15Kg/cm de 12VDC y un sensor encoder óptico para detectar el número de vueltas del rotor. La orden del ciclo de trabajo de la señal PWM para alimentar al motor se la genera desde el computador con el software de Matlab, esta orden se la envía por puerto serial hacia el módulo programable MEI&T04 que decodifica la trama de comunicación y toma la información de PWM para el motor. Al rotor del motor DC deberá estar conectado un disco con una ranura que al momento de pasar por el sensor permitirá el paso de la señal infrarroja enviando un pulso al módulo programable por cada vuelta del rotor del motor DC, el módulo programable calcula el tiempo entre pulso y pulso y se calcula las revoluciones por segundo que dará determinando así las RPS (Revoluciones por Segundo). La información de las RPS se envía al computador por el puerto serial en una trama de comunicación que será validada y almacenada por Matlab.

FIGURA 2.2: PLANTA CON MOTOR DC

PLANTA CON FUENTE GENERADORA DE CALOR Y LUZ:

Esta planta utiliza un actuador dicroico de 50W DC y dos sensores uno de temperatura y uno de luz para detectar el número de vueltas del rotor. La orden de la potencia para energizar el dicroico se la genera desde el computador con el software de Matlab, esta orden se la envía por puerto serial hacia el módulo programable MEI&T04 que decodifica la trama de comunicación y toma la información de potencia para el dicroico. El dicroico deberá estar en una caja de acrílico para evitar perturbaciones al momento de generar calor y en su interior también deberá estar los sensores de temperatura y luz. La información de la temperatura se envía al computador por el puerto serial en una trama de comunicación que será validada y almacenada por Matlab, el sensor de luz permite validar que el dicroico esté encendido en el código del módulo programable.

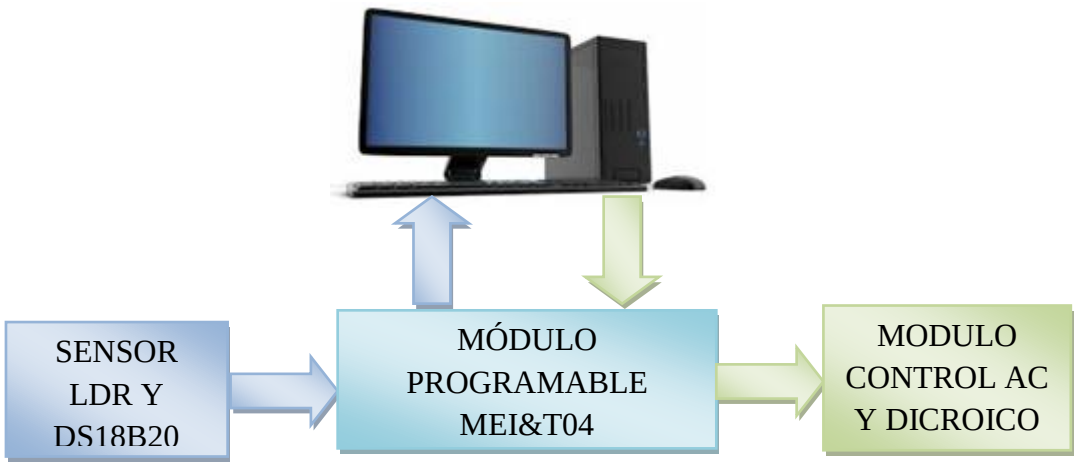

FIGURA 2.3: PLANTA CON FUENTE DE LUZ Y CALOR

2.2 DISEÑO DE LAS GUÍAS DE PRÁCTICAS

Esta planta nos permite realizar prácticas en un entorno académico de laboratorio específicamente en temas de adquisición de datos, identificación de sistemas y diseño de controlador para las plantas de motor DC y la planta de luz-calor.

PRÁCTICA DE ADQUISICIÓN DE DATOS:

Para la implementación de esta práctica se debe realizar un código en Matlab que nos permita adquirir, monitorear y almacenar los datos de la planta, el mismo que podrá ser realizado con la herramienta GUIDE de Matlab. Además en esta práctica el software interactúa con la planta conectada por puerto serial para recibir y enviar datos.

Los sensores en el diseño de la planta ya debieron ser seleccionados adecuadamente para poder tener mediciones aceptables y que nos permitan hacer una correcta medición. El tiempo de muestreo se configurará con un simple time de 1 segundo en el firmware del módulo programable MEI&T04.

Una vez adquiridos los datos en el WorkSpace de Matlab se procede a almacenarlos para su posterior análisis en las siguientes prácticas.

El tiempo de adquisición de datos dependerá del proceso muestreado, justamente el proceso con el motor DC demora segundos para medir todo su comportamiento en lazo abierto. El proceso con la fuente de calor es mucho más lento incluso necesitando horas para una adecuada medición del comportamiento del proceso.

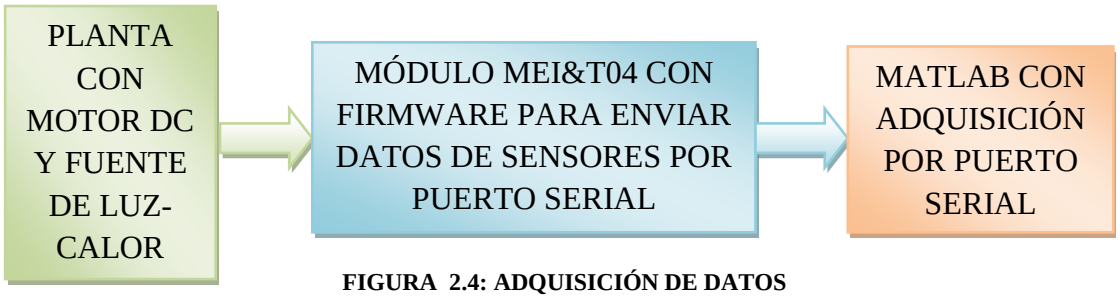

FIGURA 2.4: ADQUISICIÓN DE DATOS

PRÁCTICA DE IDENTIFICACIÓN DEL SISTEMA:

Para la implementación de esta práctica se utilizan los datos adquiridos en la práctica adquisición de datos realizada anteriormente. En esta práctica se utiliza solamente la herramienta Matlab con el System Identification Toolbox para determinar la ecuación característica de cada sistema a identificar para su posterior diseño de controlador. Para la implementación de esta práctica no necesitaremos el uso de la planta física lo que más bien nos interesa es alcanzar buen porcentaje de aproximación al sistema medido.

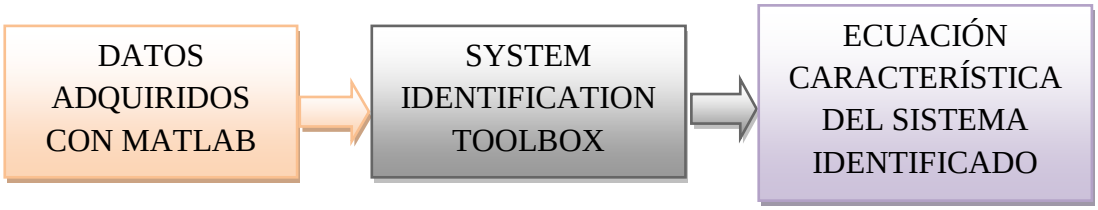

FIGURA 2.5: IDENTIFICACIÓN DEL SISTEMA

PRÁCTICA DE DISEÑO DEL CONTROLADOR ADECUADO:

Para la implementación de la práctica del diseño del controlador debemos tomar la ecuación característica de la planta identificada y hallar un controlador adecuado en función de los requerimientos de control propuestos para cada proceso. Para esta actividad usaremos la herramienta de Matlab con el Sisotool. El controlador dependerá de cada sistema debido a que ambos sistemas de la planta tiene comportamiento diferente y variables distintas. Luego de hallar el controlador procedemos a comprobar el sistema en lazo cerrado (planta, controlador y retroalimentación) sin necesidad de utilizar la planta sino haciendo uso de la herramienta Simulink, esto es una ventaja ya que no es necesario tener el proceso real o tener que detener procesos en la industria desperdiciando materia prima para evaluar el controlador, previo a su implementación en la planta o proceso real.

FIGURA 2.6: DISEÑO DEL CONTROLADOR ADECUADO

CAPÍTULO 3

IMPLEMENTACIÓN

En este capítulo se cubrirá el tema de implementación de las plantas que integran el sistema. Todas las plantas usan en común el módulo programable MEI&T04, lo que diferencia una planta de la otra es el sensor, driver y actuador.

En los enunciados siguientes trataremos en detalle la implementación de cada planta.

3.1 IMPLEMENTACIÓN DE LA PLANTA DE ADQUISICIÓN Y CONTROL

PLANTA CON MOTOR DC:

Esta planta está constituida por un sensor que nos permite detectar cuando el rotor del motor ha girado una vuelta completa, esto es gracias a que en el rotor se acopla un disco con una ranura que al momento de pasar la ranura por el sensor

éste último genera un pulso de unos cuantos microsegundos suficientes para ser detectados por microcontrolador.

Cada vez que se detecta el pulso generado por el sensor se incrementa un contador que durante un segundo contará el número de vueltas que ha girado el rotor del motor o RPS (Revoluciones Por Segundo).

Una vez calculado las revoluciones por segundo del motor se forma una trama de datos que incluye Byte de Inicio “P” y luego un byte de revoluciones medidas “RPS”. Esta trama de datos se escribe por el puerto serial del microcontrolador hacia el computador donde estará corriendo Matlab, esta información de las RPS es importante tanto para operaciones con la planta en lazo abierto y cerrado.

Desde el software Matlab se envía una trama de datos “IT+M=XXX” donde el valor de “XXX” puede iniciar con una velocidad mínima “000” hasta una velocidad máxima “255”, este valor representa el ciclo de trabajo de la señal PWM generada por el microcontrolador para controlar el motor.

Una vez que el módulo programable recibe la trama de datos por el puerto serial te toma el valor del ciclo de trabajo para generar la señal PWM con el ciclo de trabajo que le indica Matlab.

Debido a que la señal de PWM generada por el microcontrolador es muy débil para controlar cargas como motores, se utiliza un driver capaz de controlar el motor DC con un consumo de corriente máximo de 2A con el rotor sin carga.

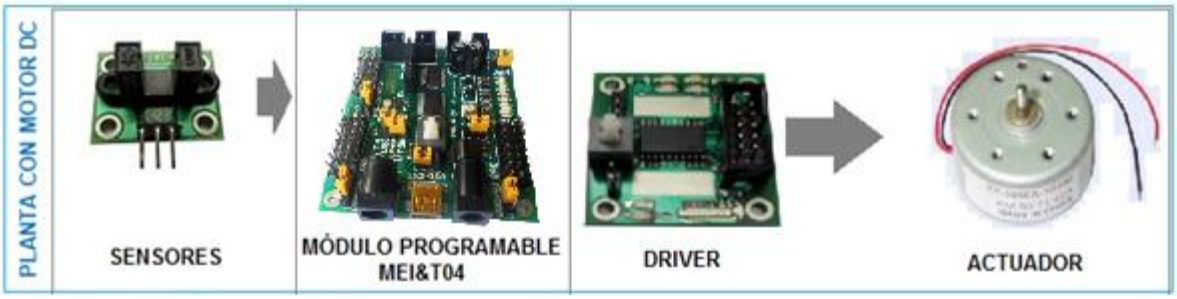

FIGURA 3.1: COMPONETES DE LA PLANTA CON MOTOR DC

PLANTA CON FUENTE GENERADORA DE CALOR Y LUZ:

Esta planta está constituida por un sensor de luz y un sensor de temperatura que nos permiten detectar si el dicroico está encendido y cuál es la temperatura generada, el escenario donde se alojan los sensores y el dicroico están confinados en una caja de acrílico para evitar lo más posible perturbaciones externas.

El sensor de temperatura usado es el DS18B20 que envía por comunicación OneWire al módulo programable la temperatura medida hasta con 4 decimales de precisión.

Una vez medida la temperatura se forma una trama de datos que incluye Byte de Inicio “T” y luego un byte de temperatura medida “Temp”. Esta trama de datos se escribe por el puerto serial del microcontrolador hacia el computador donde estará corriendo Matlab, esta información de la temperatura es importante tanto para operaciones con la planta en lazo abierto y cerrado.

Desde el software Matlab se envía una trama de datos “IT+L=XXX” donde el valor de “XXX” puede iniciar con una potencia mínima “000” hasta una potencia máxima “255”, este valor representa el porcentaje de la señal de corriente alterna (“000” que equivale 0% y el “255” que equivale 100%) que alimentará al dicroico, estos

porcentajes también se pueden interpretar como la potencia que genera el dicroico sabiendo que con un 100% de la fuente de corriente alterna el dicroico generará los 50W y en un 0% el dicroico se apagará.

Una vez que el módulo programable recibe la trama de datos por el puerto serial toma el valor del porcentaje de potencia para controlar la carga de corriente alterna resistiva, en este caso el dicroico.

Debido a que la señal generada por el microcontrolador es muy débil para controlar cargas como resistivas de 50W y de corriente alterna, se utiliza un driver capaz de controlar este tipo de cargas.

FIGURA 3.2: COMPONETES DE LA PLANTA CON FUENTE DE LUZ Y CALOR

COMPONENTES DEL SISTEMA COMPLETO:

El sistema completo con ambas plantas comparten un mismo módulo programable donde el firmware detecta con qué tipo de planta (sensores y actuador) se desea trabajar, dándonos como resultado de que la implementación nos permite trabajar con una sola planta a la vez.

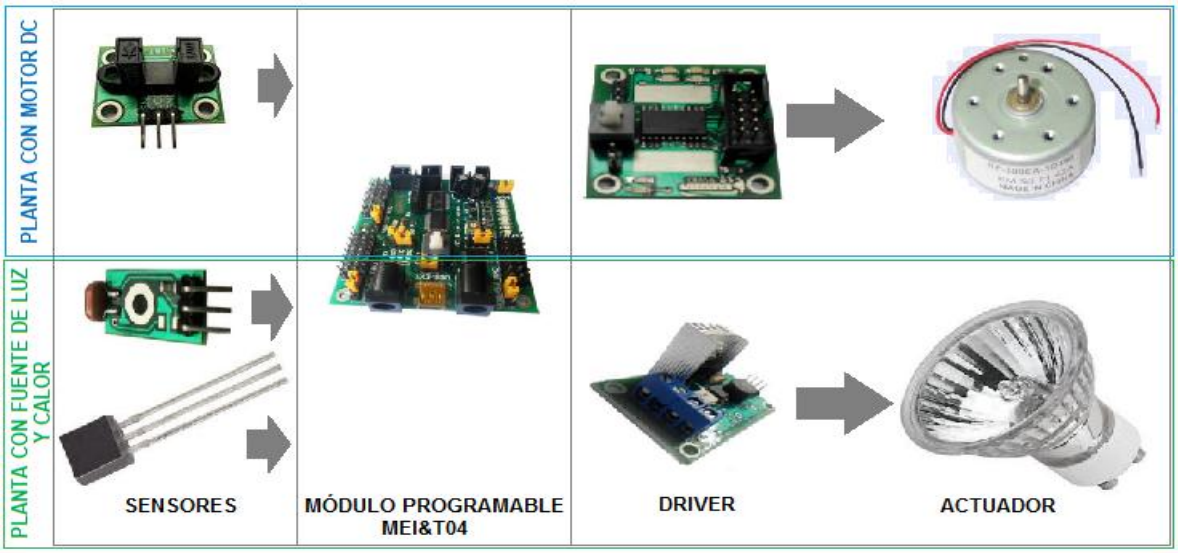

FIGURA 3.3: COMPONENTES DE LA PLANTA DE ADQUISICIÓN

110VAC. Al momento de implementar la planta hemos colocado breaker de protección antes de energizar la fuente que alimenta todos los circuitos. La salida del breaker se lo conecta a cuatro borneras para distribuir la conexión de corriente alterna.

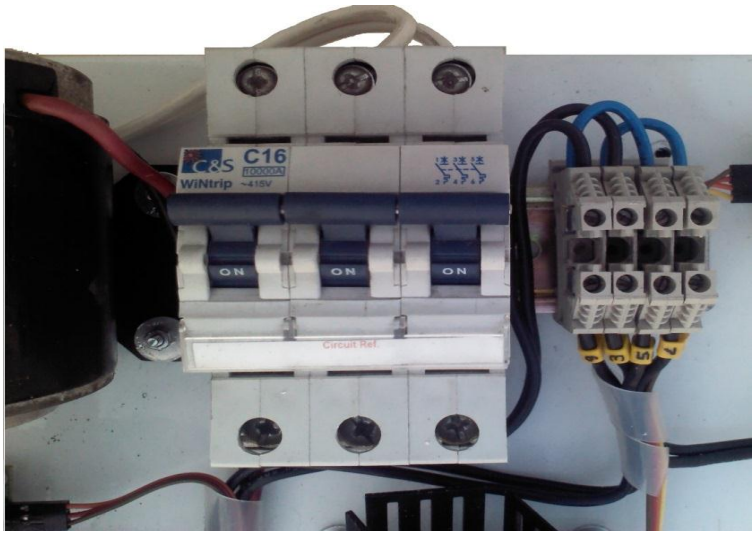

FIGURA 3.4: IMPLEMENTACIÓN DE PROTECCIÓN ELÉCTRICA

La fuente que energiza toda la planta tiene una entrada 110VAC de alimentación y entrega 12 VDC / 2A. Por medio de borneras se lleva el voltaje para dar energía a módulo MEI&T04 que a su vez energiza a todos los sensores y polariza el módulo de control de carga AC. Otra bornera alimenta el puente H para suministrar energía al motor DC.

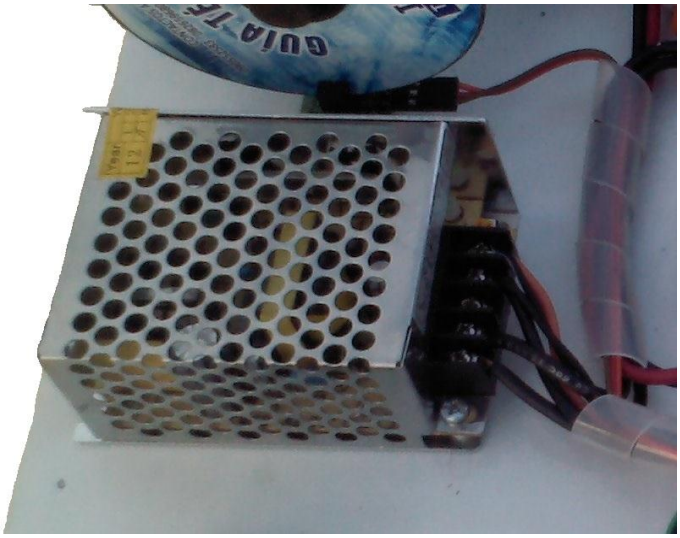

FIGURA 3.5: FUENTE DE ALIMENTACIÓN

El módulo programable MEI&T04 permanece conectado a los sensores: Encoder óptico, sensor de temperatura y sensor de luz. Además está conectado al puerto de comunicación serial y a los driver del motor y de la carga resistiva AC. Este módulo programable toma energía directamente de la fuente.

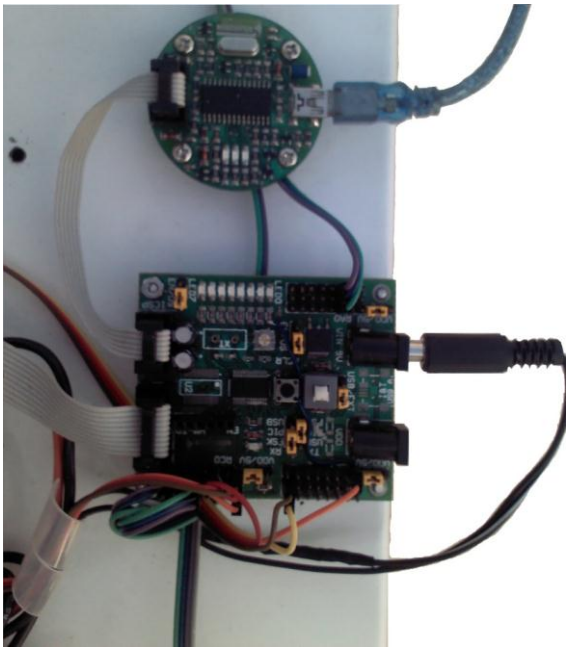

FIGURA 3.6: MÓDULO PROGRAMABLE MEI&T04

La planta con motor DC como vemos en la figura 3.6, posee instalado un motor de 200W y en su rotor tiene acoplado un disco con ranura que permite al encoder óptico detectar cuando el motor ha dado un giro completo. En los terminales del motor DC están conectados a la bornera del módulo puente H para suministrar el voltaje del motor para su control. Y este puente h al igual que el sensor, está conectado el módulo programable.

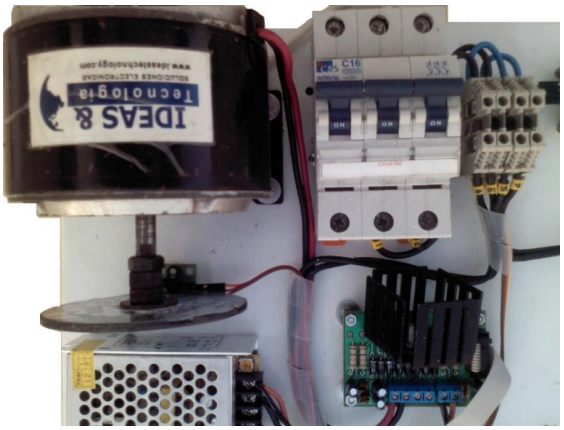

FIGURA 3.7: PLANTA CON MOTOR DC

El disco acoplado con el rotor del motor está centrado para evitar que este toque el sensor e introduzca ruido por causa de las vibraciones. Además el driver tiene un disipador de calor para proteger el driver del motor.

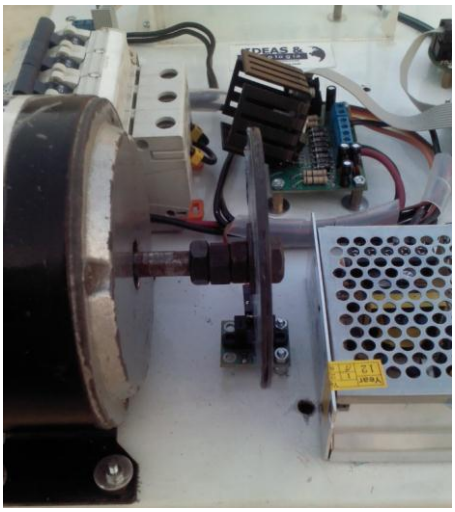

FIGURA 3.8: PLANTA CON MOTOR DC

La planta con fuente de luz y calor está confinada en una caja acrílica transparente que nos permite evitar perturbaciones externas de temperatura, lo sensores también están al interior de la caja. El dicroico genera el calor que es medido por el sensor DS18B20 que por comunicación one wire envía esta información al módulo programable por tanto posee un bus de datos que lo conecta y energiza para su correcto trabajo. El dicroico se conecta a una boquilla de cerámica que es capaz de soportar temperaturas elevadas; esta carga se energiza desde el driver para control de carga AC resistivo.

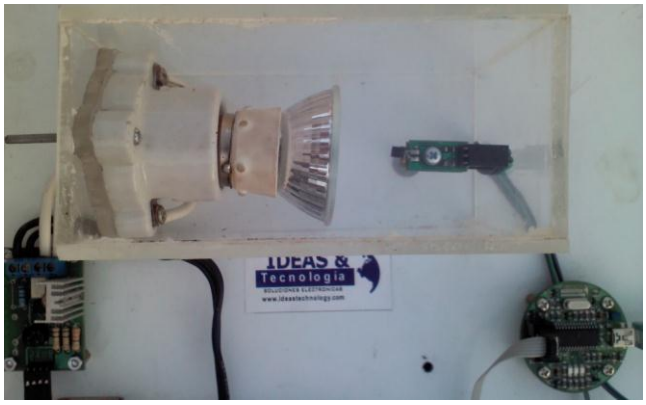

FIGURA 3.9: PLANTA CON FUENTE DE LUZ Y CALOR

Tanto el actuador como el sensor están conectados al módulo programable. EL dicroico no puede ser conectado directamente al módulo programable por lo que hay un bus de conexión entre el módulo programable y el driver de control AC, este driver logra dimerizar y controlar la potencia del dicroico por medio de la segmentación de la onda de corriente alterna y la salida del driver se conecta al dicroico.

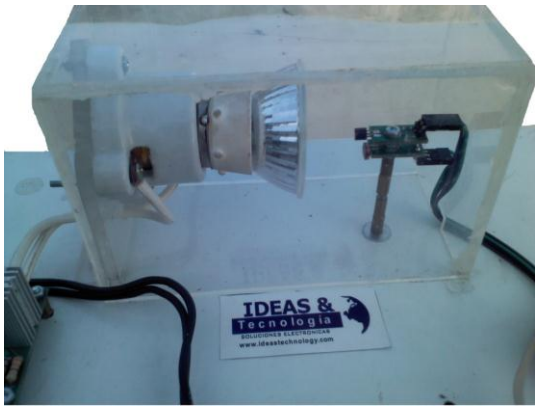

FIGURA 3.10: PLANTA CON FUENTE DE LUZ Y CALOR

Como se observa en la figura 3.10 la planta completa poseen conexiones entre los sensores y drivers directamente al módulo programable que es el cerebro donde está

el firmware que permite administrar los recursos del sistema en función de la práctica a realizar y con el tipo de planta seleccionada.

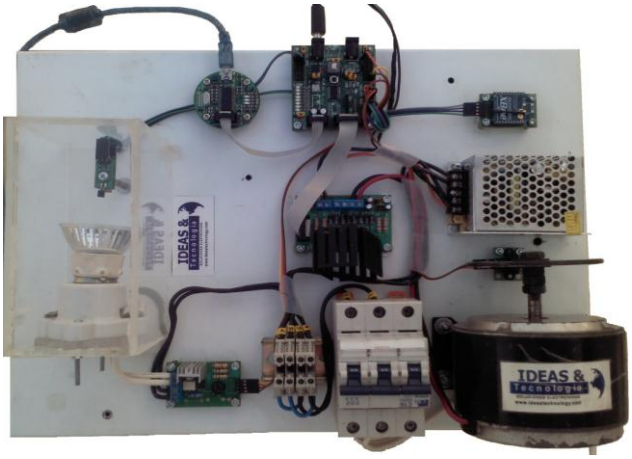

FIGURA 3.11: IMPLEMENTACIÓN COMPLETA

La distribución de los componentes de la planta está de tal forma que nos permite apreciar y diferenciar cada planta y sus componentes principales que son: sensores, módulo programable y driver con el actuador.

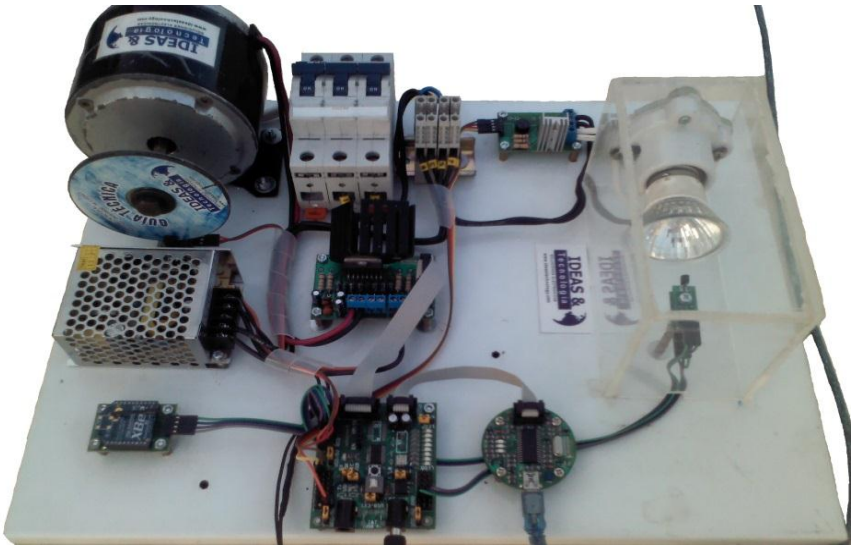

FIGURA 3.12: IMPLEMENTACIÓN COMPLETA

3.2 IMPLEMENTACIÓN DE LAS PRÁCTICAS Y SIMULACIONES

3.2.1. INTRODUCCIÓN

Software de adquisición de datos Matlab

Aquí podemos observar el software de adquisición desarrollado en Matlab, este software fue desarrollado para que el estudiante pueda adquirir de una manera fácil, económica y rápida datos de las plantas de I&T ya que envés de usar una tarjeta de adquisición de datos (DAQ) se hizo el uso del puerto serial, y así poder hacer una identifcación aproximada de la planta y desarrollar un controlador para la misma.

3.2.1.1. INTRODUCCIÓN BÁSICA ENTORNO GRÁFICO DE MATLAB (GUIDE)

GUIDE es un entorno de programación visual disponible en MATLAB para realizar y ejecutar programas que necesiten ingreso continuo de datos. Tiene las características básicas de todos los programas visuales como Visual Basic o Visual C++.

Inicio: Para iniciar nuestro proyecto, debemos dar clic en la figura.

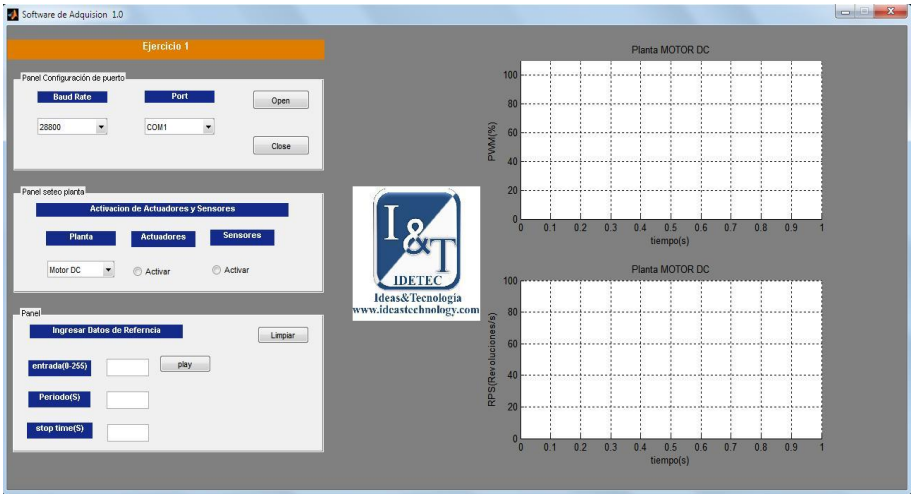

FIGURA 3.13: SOFTWARE PARA ADQUISICIÓN DESARROLLADO

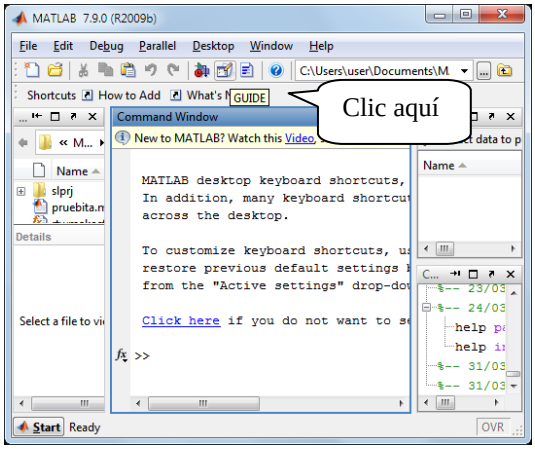

FIGURA 3.14: COMO LLAMAR AL GUIDE DE MATLAB

- Luego se nos abrirá el siguiente cuadro de dialogo.

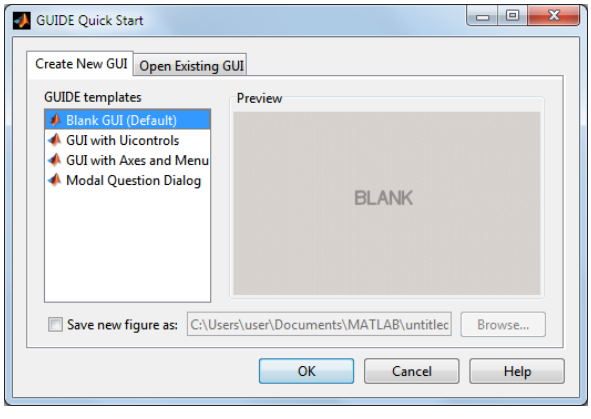

FIGURA 3.15: HERRAMIENTA GUIDE DE MATLAB

- Escogemos la opción: **Blank GUI (Default).**
- La opción de interfaz gráfica de usuario en blanco (viene predeterminada), nos presenta un formulario nuevo, en el cual podemos diseñar nuestro programa.
- Luego de haber escogido la opción mencionada tenemos:

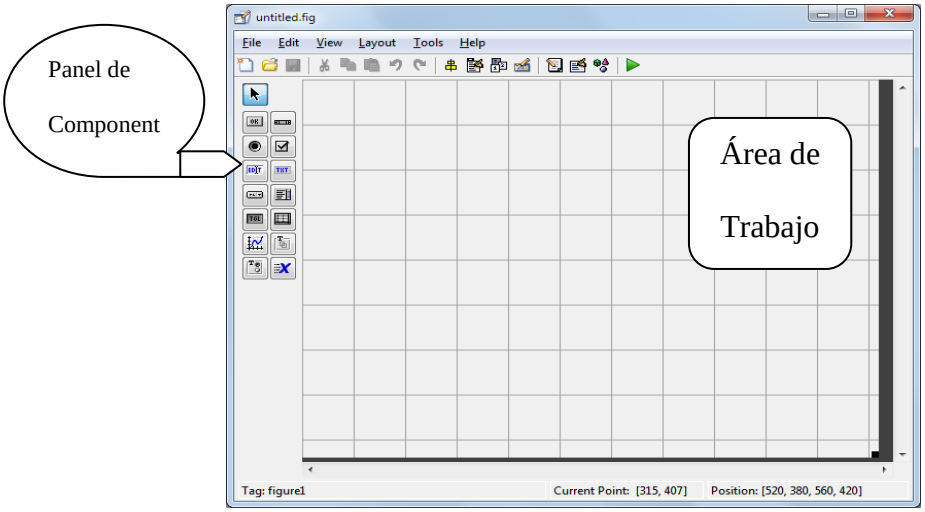

FIGURA 3.16: PANEL DE TRABAJO Y COMPONENTES

Aquí podemos ver la siguiente barra de herramientas:

	Alinear objetos.
	Editor de menú.
	Navegador de objetos.
	Editor de Barra de Herramientas

	Accede al panel de componentes
	Grabar y Ejecutar
	Editor del M-file.
	Editor de orden de etiqueta.

TABLA 3.1: LISTA DE HERRAMIENTAS EN PANEL FRONTAL

Para obtener la etiqueta de cada elemento de la paleta de componentes ejecutamos: *File>>Preferentes* y seleccionamos *>>Show names in component palette.*

- Tenemos la siguiente presentación:

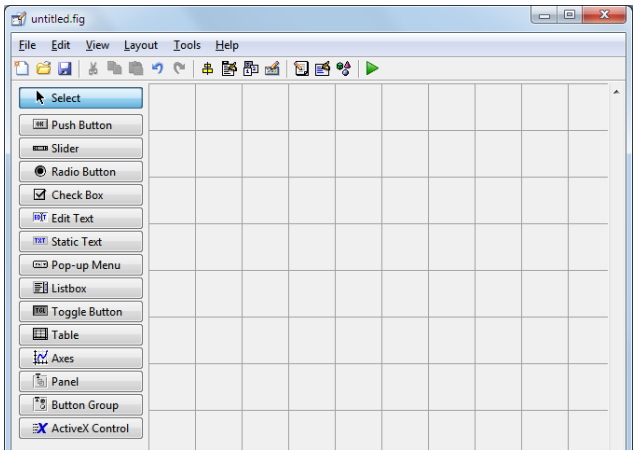

FIGURA 3.17: PALETA DE COMPONENTES

La siguiente tabla muestra una descripción de los componentes:

Control	*Valor de estilo*	*Descripción*
Check box	checkbox’	Indica el estado de una opción o atributo
Editable Text	‘edit’	Caja para editar texto
Pop-up menu	‘popupmenu’	Provee una lista de opciones
List Box	‘listbox’	Muestra una lista deslizable
Push Button	‘pushbutton’	Invoca un evento inmediatamente
Radio Button	‘radio’	Indica una opción que puede ser seleccionada
Toggle Button	‘togglebutton’	Solo dos estados, “on” o “off”
Slider	‘slider’	Usado para representar un rango de valores
Static Text	‘text’	Muestra un string de texto en una caja
Panel button		Agrupa botones como un grupo
Button Group		Permite exclusividad de selección con los radio button

TABLA 3.2: DESCRIPCIÓN DE CADA COMPONENTES

3.2.1.2 PROPIEDAD DE LOS COMPONENTES

Cada uno de los elementos de GUI, tiene un conjunto de opciones que podemos acceder con clic derecho.

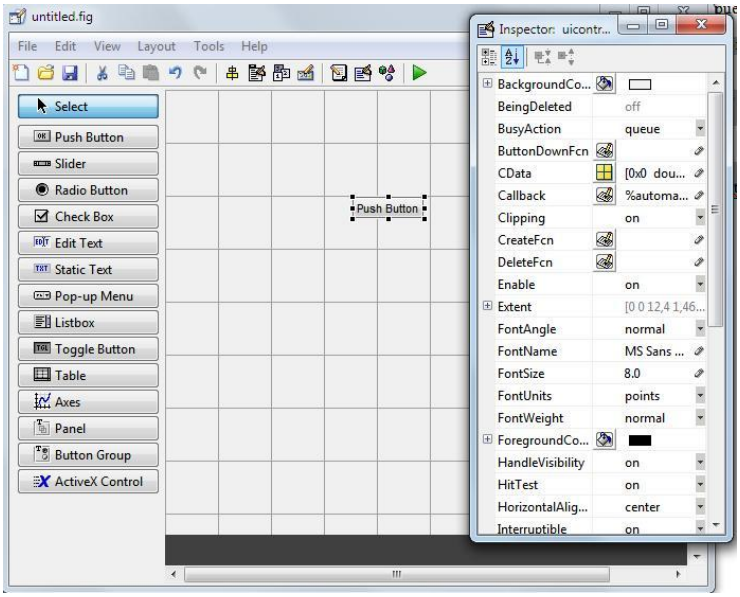

FIGURA 3.18: PROPIEDAD DE LOS COMPONENTES

Al hacer click derecho en el elemento ubicado en el área de diseño, una de las opciones más importantes es View Callbacks, la cual, al ejecutarla, abre el archivo .m asociado a nuestro diseño y nos posiciona en la parte del programa que corresponde a la subrutina que se ejecutará cuando se realice una determinada acción sobre el elemento que estamos editando.

Por ejemplo, al ejecutar View Callbacks>>Callbacks en el Push Button, nos ubicaremos en la parte del programa:

```
function pushbutton1_Callback(hObject, eventdata, handles)
%hObject handle to pushbutton1 (see GCBO)
%eventdata reserved-to be defined in a future version of MATLAB
%handles structure with handles and user data (see GUIDATA)
```

3.2.1.3 FUNCIONAMIENTO DE UNA APLICACIÓN GUI

Una aplicación GUIDE consta de dos archivos: *.m* y *.fig*. El archivo .m es el que contiene el código con las correspondencias de los botones de control de la interfaz y el archivo *.fig* contiene los elementos gráficos.

Cada vez que se adicione un nuevo elemento en la interfaz gráfica, se genera automáticamente código en el archivo *.m*.

Para ejecutar una Interfaz Gráfica, si la hemos etiquetado con el nombre *curso.fig*, simplemente ejecutamos en la ventana de *comandos* >> *curso*. También podemos acceder haciendo clic derecho en el m-file y seleccionando la opción RUN.

3.2.1.4 MANEJO DE DATOS ENTRE LOS ELEMENTOS DE LA APLICACIÓN Y EL ARCHIVO ".M"

Todos los valores de las propiedades de los elementos (color, valor, posición, string…) y los valores de las variables transitorias del programa se almacenan en una estructura, los cuales son accedidos mediante un único y mismo *identificador* para todos éstos. Tomando el programa listado anteriormente, el identificador se asigna en:

- handles.output = hObject;
    - *handles*, es nuestro identificador a los datos de la aplicación. Esta definición de identificador es salvada con la siguiente instrucción:
- guidata(hObject, handles);
    - *guidata*, es la sentencia para salvar los datos de la aplicación.

Aviso: *guidata* es la función que guarda las variables y propiedades de los elementos en la estructura de datos de la aplicación, por lo tanto, como regla general, en cada subrutina se debe escribir en la última línea lo siguiente:

- guidata(hObject,handles);
    - Esta sentencia nos garantiza que cualquier cambio o asignación de propiedades o variables quede almacenado.

Por ejemplo, si dentro de una subrutina una operación dio como resultado una variable *diego* para poder utilizarla desde el programa u otra subrutina debemos salvarla de la siguiente manera:

- handles.variable=variable;
- guidata(hObject,variable);

La primera línea crea la variable *variable* a la estructura de datos de la aplicación apuntada por *handles* y la segunda graba el valor.

3.2.1.5 SENTENCIAS GET Y SET

La asignación u obtención de valores de los componentes se realiza mediante las sentencias *get* y *set*. Por ejemplo si queremos que la variable *utpl* tenga el valor del *Slider* escribimos:

- utpl= get(handles.slider1,'Value');
    - Notar que siempre se obtienen los datos a través de los identificadores *handles*.

Para asignar el valor a la variable *utpl* al *statictext* etiquetada como *text1*escribimos:

- set(handles.text1,'String',utpl);%Escribe el valor del Slider...
- %en static-text

3.2.1.6 ABRIR EL SOFTWARE DE ADQUISICIÓN DE DATOS

- **Paso 1:** Vamos abrir nuestra carpeta donde tengamos los archivos de Matlab, y abrimos el *archivo.m*.

FIGURA 3.19: ABRIENDO EL ARCHIVO ".M" DEL GUIDE

- **Paso 2:** Se nos abrirá la siguiente ventana,para correr el programa simplemente le damos clic en la figura que dice (Run).

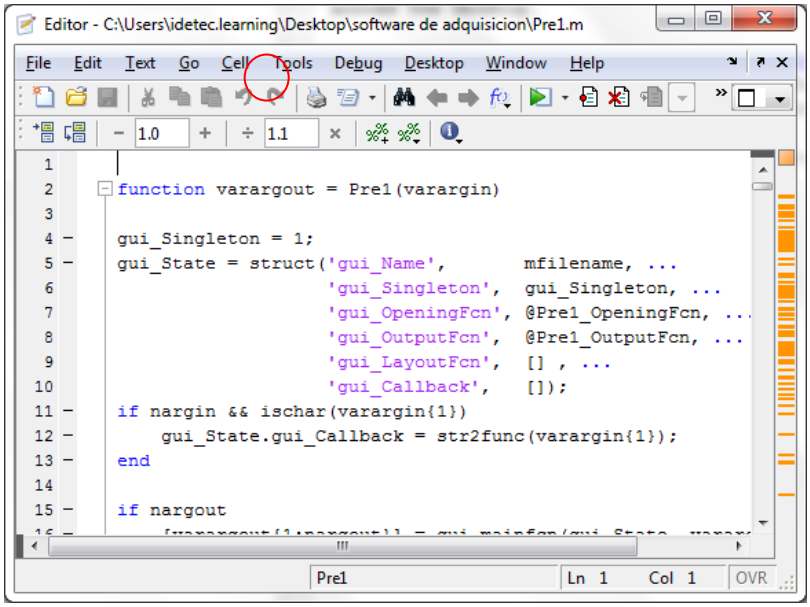

FIGURA 3.20: EJECUTANDO EL ARCHIVO ".M" DEL GUIDE

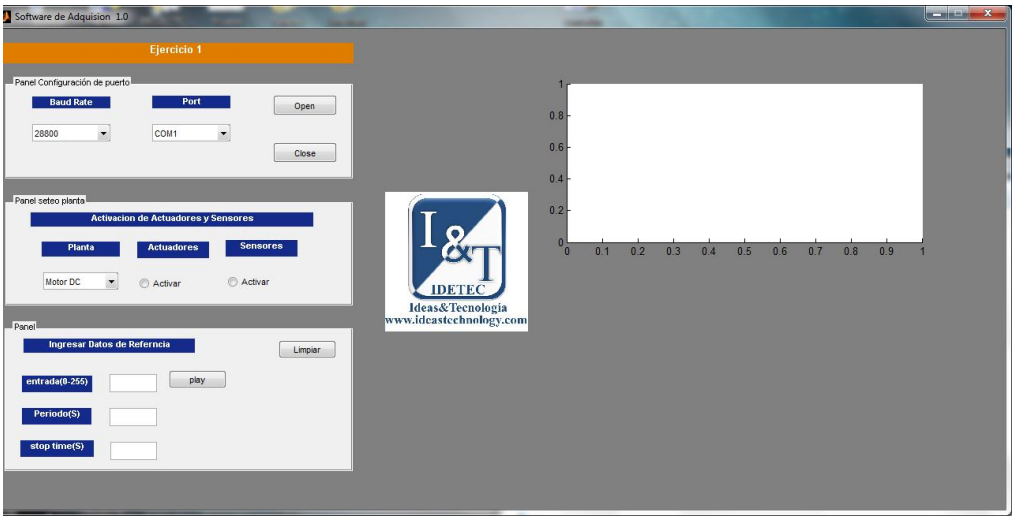

FIGURA 3.21: SOFTWARE GUIDE LISTO PARA USAR

- **Paso 3:** Se nos abrirá el programa GUIDE para usarlo con la planta.

3.2.2. ADQUISICIÓN DE DATOS DE LA PLANTA

3.2.2.1. ANTECEDENTES

Hay diversas formas de adquirir datos, ya sea por una (DAQ), por puerto serial, por puerto paralelo, nuestra planta de adquisición envía datos a nuestro computador por medio del puerto serial.

3.2.2.2. OBJETIVOS

- Usar el software Adquisición 1.0 del programa MATLAB herramienta (GUIDE).
- Aprender consideraciones básicas para tomar datos de una planta para identificarla.

3.2.2.3. TEORÍA:

Ciclo de Trabajo (Duty Cycle): Una manera posible para la identificación experimental en un sistema, es someterlo a una prueba dinámica utilizando una señal de prueba del tipo tren de pulsos centrados co. un ciclo de trabajo (duty cycle) del 50%. En este caso es necesario ajustar el periodo de la señal de prueba de tal manera que el sistema tenga tiempo de estabilizarse antes que el siguiente pulso se aplique. (Ver figura).

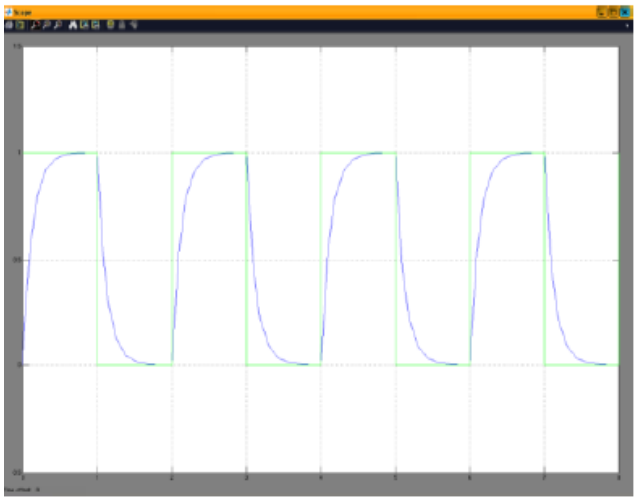

FIGURA 3.22: CICLO DE TRABAJO DEL PROCESO

Por esta razón, es necesario establecer el Tiempo de Estabilización del sistema a ser probado por una señal de entrada tipo escalón. Este tiempo se lo define como el tiempo necesario para que la magnitud de la señal de respuesta a la prueba de escalón sea del 98%. Ver figura.

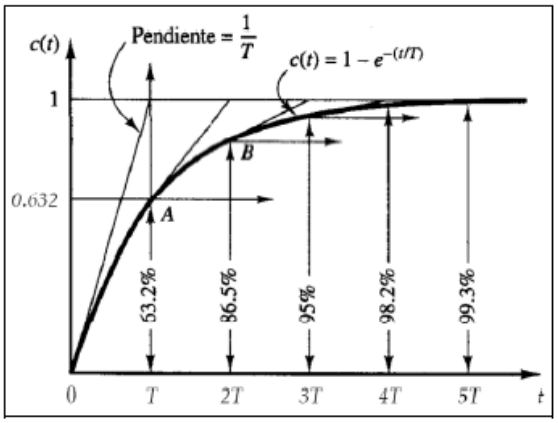

FIGURA 3.23: CICLO DE TRABAJO DEL PROCESO

De acuerdo al gráfico, el Tiempo de Estabilización “Ts” se lo logra en cuatro Constantes de Tiempo “T”.

$$T_s = 4T \qquad Ecuación\ 1$$

El periodo de la señal de prueba de se lo fijaría de acuerdo a la siguiente relación:

$$P_{Tren\ Pulsos} \geq 2T_s \qquad Ecuación\ 2$$

3.2.2.4. PRÁCTICA

En esta práctica procederemos a identificar la planta motor DC de I&T, la velocidad del motor está controlada por medio del PWM que se le dé como referencia y podemos medir la velocidad por medio de un encoder el cual nos dará la información de cuantas RPS (Revoluciones /segundo) tendrá mi motor en tiempo real, cabe recalcar que el tiempo de muestreo de la planta es aproximadamente de 1 segundo.

- **Paso 1:** Para la adquisición de datos debemos abrir el puerto serial por el cual vamos a comunicar la planta con la pc, y también debemos fijar un tasa de baudios, para las plantas (I&T) trabajamos a 9600 baudios y el puerto serial depende de que puerto COM serial se halla habilitado, después proceda a presionar el botón (Open).

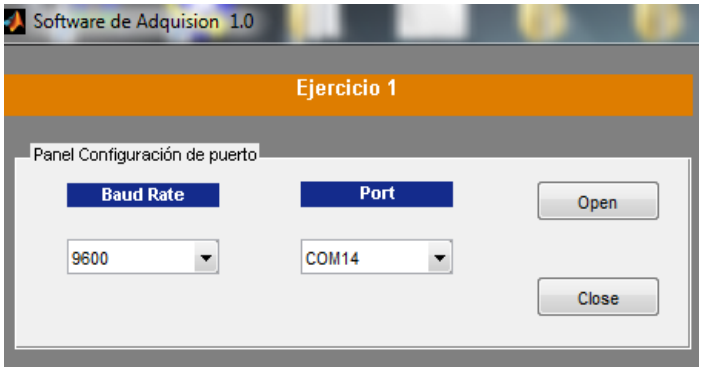

FIGURA 3.24: SET PARA COMUNICACIÓN SERIAL

Como vemos en administrador de dispositivos vemos el puerto COM14 que se nos habilito, y en software también esta habilitados hasta el puerto 15, si se les abre un puerto mayor al com15, pueden proceder a cambiarlo por otro menor en propiedades donde dice (USB Serial Port(COM14)).

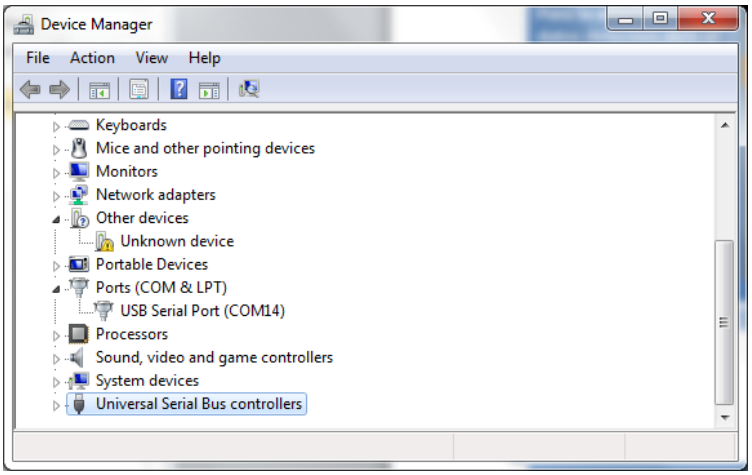

FIGURA 3.25: IDENTIFICANDO EL NÚMERO DE PUERTO COM

- **Paso 2:** Ahora tenemos que elegir cual planta vamos a adquirir los datos, para esta práctica elegimos Motor DC y después procedemos a activar actuadores y sensores. Cuando elijamos Motor DC veremos que se nos mostrara dos plot en el primero se graficara la referencia y en el segundo lo que está midiendo en tiempo real.

FIGURA 3.26: SELECCIÓN DE LA PLANTA

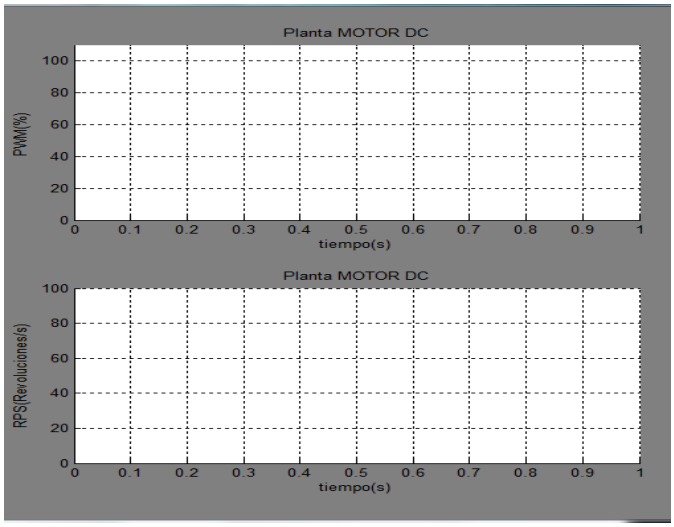

FIGURA 3.27: PLOT DE VISUALIZACIÓN DE SEÑAL DE ESTIMULO

- **Paso 3:** Aquí ingresamos la entrada la cual va de 0-255 la cual representará en nuestra grafica de 0 al 100% de PWM que tendrá mi Motor DC. El periodo que ingresamos es el tiempo en alto y en bajo de nuestra señal de referencia, con un duty cicle de 50%, y el stop time es el tiempo el cual para nuestra adquisición de datos.

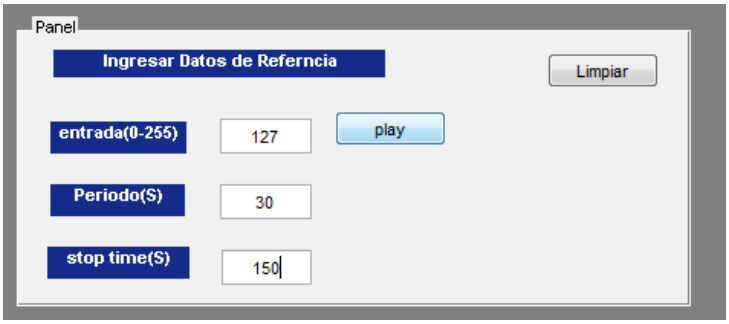

FIGURA 3.28: CONFIGURACIÓN DE CICLO DE TRABAJO Y DURACIÓN DEL TIEMPO DE MUESTREO

NOTA: En esta práctica procederemos a identificar la planta con una referencia de 127 lo cual implica el 50% de PWM de mi motor DC, con un periodo de 30 segundos

como tenemos un duty cicle de 50% tendremos aproximadamente 15s en alto y 15s en bajo, podemos ver también un stop time de 150 segundos, lo cual quiere decir que tendremos aproximadamente 5 periodos.

Si se desea cambiar la referencia se lo puede hacer sin ningún problema, debido a que su tiempo de estabilización es rápido, y tampoco hay problema que se pueda afectar el sensor como lo veremos en la planta del dicroico donde mediremos temperatura.

- **Paso 4:** Una vez llenado todos los datos de referencia procederemos a darle PLAY y podremos ver como se toman los datos y se grafican en tiempo real.

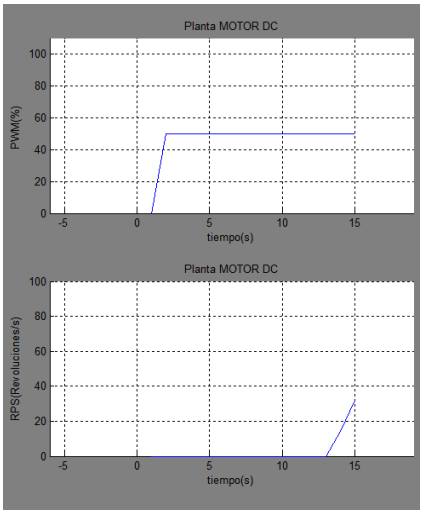

FIGURA 3.29: MUESTREOS DEL PROCESO

NOTA: Si vemos en la segunda grafica (RPS vs tiempo), comienza muy alejado, proceda a cerrar y volver hacer los pasos anteriores desde el incio, esto ocurre debido a un error de procesamiento, debido a que estamos tomando los datos de manera inlambrica en el cual el buffer del puerto serial se congestiona.Solo en la planta del

motor es aceptable que comience aproximadamente a partir de t=10 como lo vemos en la figura.

Cuando se esten tomando los datos no utilizar matlab para hacer algo más, una vez dado PLAY, solo usted puede ver las gráficas en el software.

Cuando se corra el programo no hacerlo desde un pendrive, si el proceso es lento, fijarse que su computadora no se suspedanda o no se apague porque interumpiria la toma de datos.

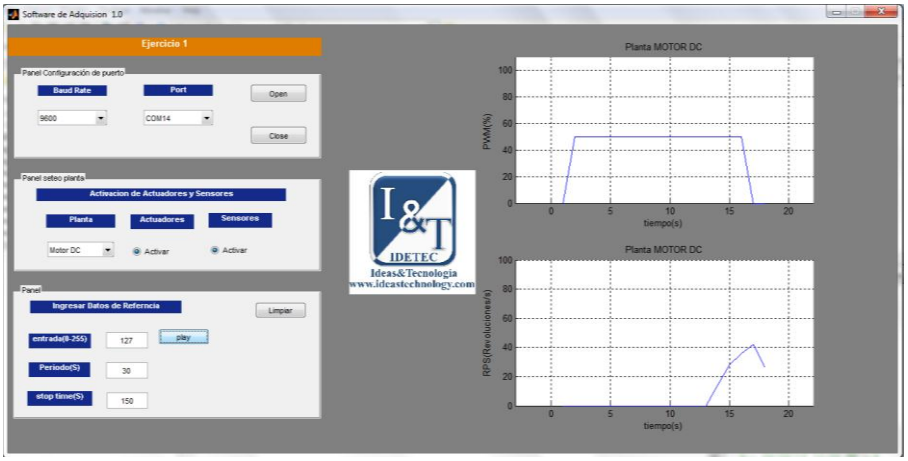

FIGURA 3.30: ADQUIRIENDO DATOS DEL PROCESO

- **Paso 5:** Cuando llegue al final la adquisición procedemos a ver los datos obtenidos, como podemos saber que llego al final de la adquisición, solo tenemos que ver en el grafico que se haya detenido hasta el tiempo que se indicó en el stop time, en nuestro ejemplo tenemos 150 s.

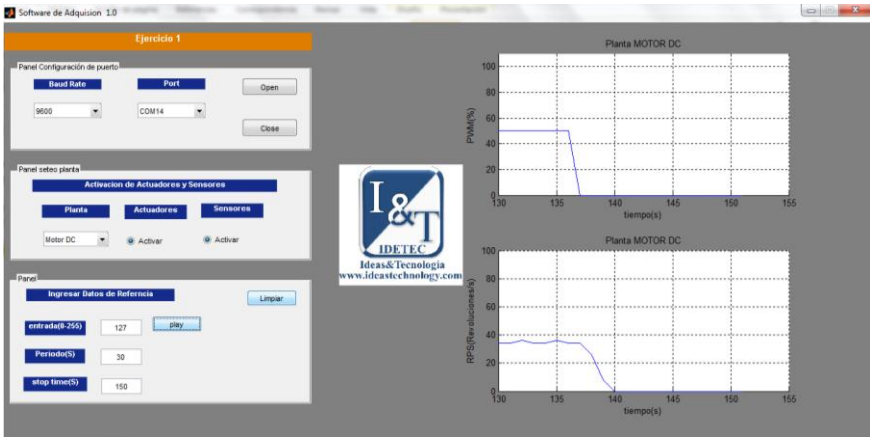

FIGURA 3.31: FINALIZADO EL PROCESO DE ADQUISICIÓN DE DATOS

- **Paso 6:** Procedemos a ir al workspace y ver el archivo que tiene como nombre DATOS, en el cual están guardados los datos de entrada y salida.

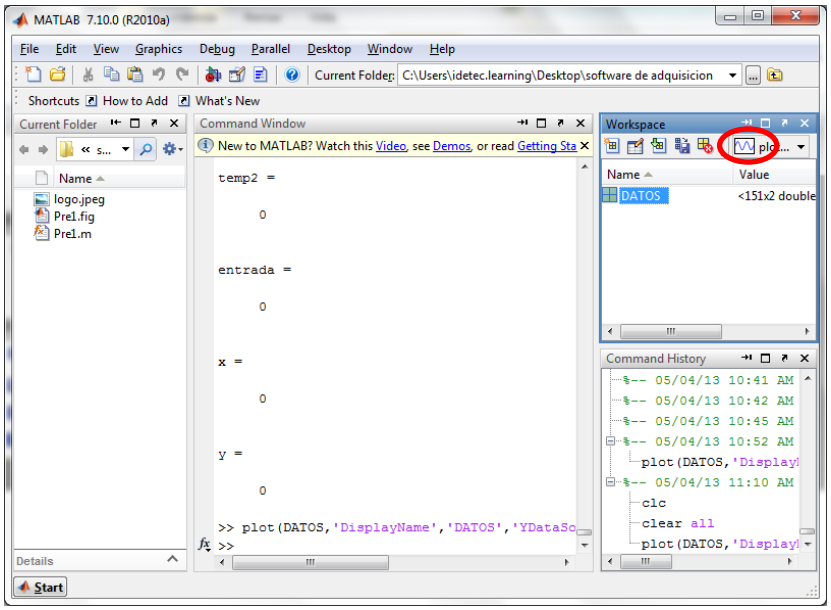

FIGURA 3.32: DATOS ADQUIRIDOS EN EL WORKSPACE

- **Paso 7:** Si deseamos ver los datos procederemos a señalar el archivo y luego darle clic en PLOT y veremos los datos de nuestra adquisición.

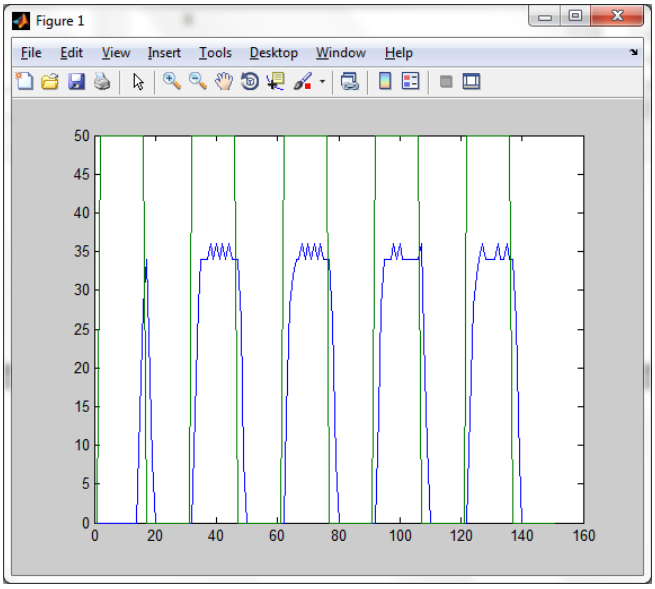

FIGURA 3.33: PLOTEO DE LOS DATOS ADQUIRIDOS

3.2.3. IDENTIFICACIÓN DE LA PLANTA

3.2.3.1. ANTECEDENTES

En muchas ocasiones no se puede obtener teóricamente (mediante un modelo matemático) la función de transferencia de una planta, debido a diversos factores. En estos casos podemos hacer una identificación del sistema a partir de mediciones hechas a la entrada de la planta y a la salida de la planta. Para este efecto, el programa MATLAB tiene una herramienta llamada SYSTEM IDENTIFICATION.

3.2.3.2. OBJETIVOS

- Usar la herramienta SYSTEM IDENTIFICATION del programa MATLAB.
- Estimar la función de transferencia de una planta real.

3.2.3.3. TEORÍA

En todo sistema podemos distinguir tres tipos de señales que son:

- **Señales de entrada:** Son aquellas señales que pueden ser controladas y de las cuales depende básicamente el funcionamiento del sistema.
- **Señales de salida:** Son señales que nos indican como se está comportando el sistema.
- **Señales de perturbación:** Son señales que afectan el comportamiento del sistema pero que no pueden ser controladas.

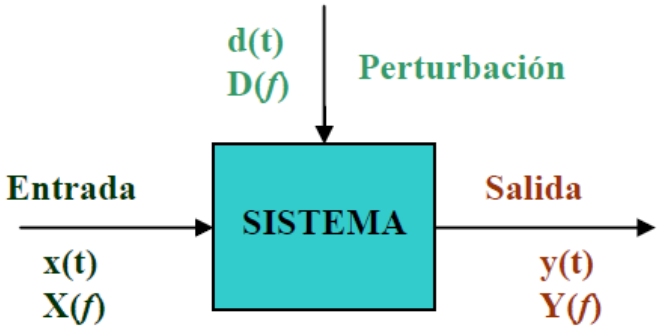

FIGURA 3.34: SEÑALES PRESENTES EN EL SISTEMA

Las señales del sistema están en el dominio del tiempo pero pueden ser manipuladas matemáticamente para llevarlas al dominio de la frecuencia. Aunque, para efecto de identificación las señales son muestreadas solo a tiempos discretos que usualmente están igualmente distanciados en unidades de tiempo. En consecuencia el problema del modelamiento es describir como están relacionadas las señales entre sí. La relación básica entre las señales es una ecuación diferencial lineal.

Matemáticamente notamos que la salida al instante t puede ser calculada como una combinación lineal de las entradas y salidas anteriores. Esta dependencia de lo que sucedió anteriormente es lo que se entiende por dinámica. En consecuencia, el problema de la identificación de un sistema consiste en determinar los coeficientes de cualquiera de las dos ecuaciones previas. Ver referencia [7].

3.2.3.4. PRÁCTICA

Una vez adquirido los datos a continuación se encuentran los pasos para realizar la identificación de un sistema.

- **Paso 1:** Una vez adquiridos los datos procedemos a realizar una identificación de sistemas, esto lo realizaremos con la herramienta <<ident>> como podemos ver en la siguiente figura.

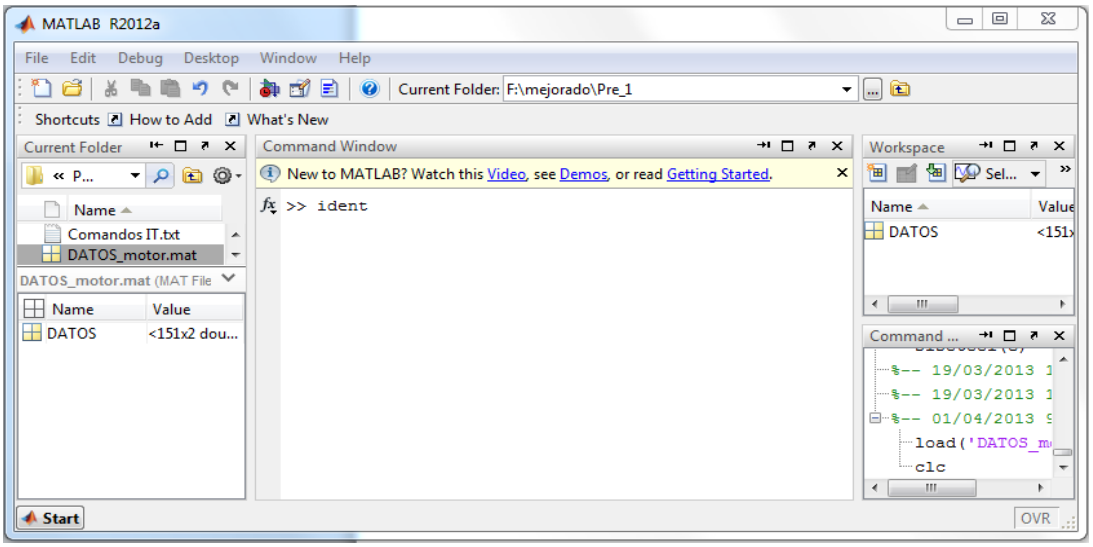

FIGURA 3.35: COMANDO IDENT PARA LLAMAR LA HERRAMIENTA DE IDENTIFICACIÓN EN MATLAB

- **Paso 2:** Se nos abrirá una ventana la cual tendremos que dale clic en la lista donde dice import data y seleccionar <<time domain data>>.

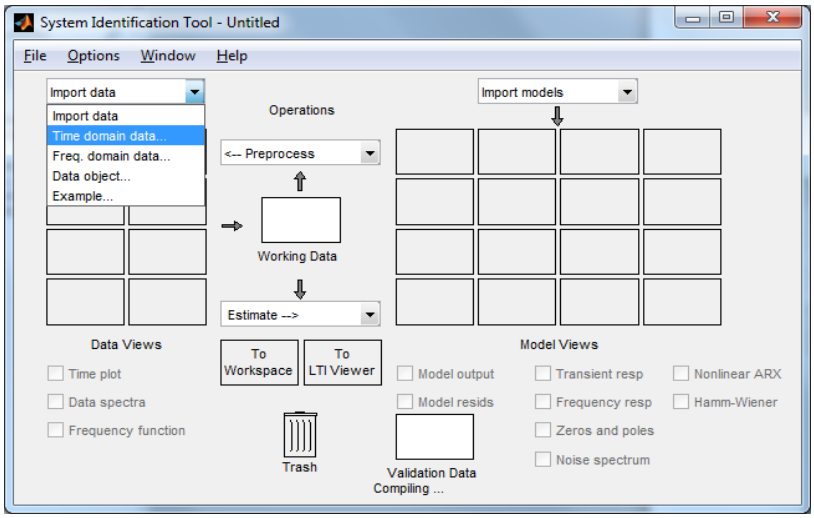

FIGURA 3.36: SELECCIÓN DEL TIPO DE DATOS A IMPORTAR

- **Paso 3:** Aquí procederemos a poner el nombre del archivo que está en el *workspace* el cual están guardados los datos de entrada y salida de mi planta, en este caso nuestro archivo se llama “DATOS” el cual es una matriz de n filas y 2 columnas. Donde dice Input tenemos que poner los datos de entrada de la matriz lo cuales están en este caso en la segunda columna, y donde dice Output de salida están en la primera columna.
- **NOTA:** Los otros datos como son *Starting time* ponemos 0 y *Sample interval* debemos poner el tiempo de muestreo el cual es 1 segundo.
- Finalmente damos clic en Import.

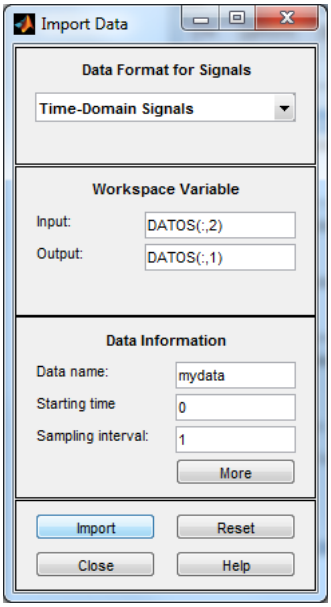

FIGURA 3.37: IMPORTANDO LOS DATOS

- **Paso 4:** Se nos abrirá una ventana en el cual se importaron los datos de la adquisición, en esta ventana procederemos a modelar nuestro sistema. <<Recordar que siempre debe estar sombreada la figura con la cual vamos a trabajar>>

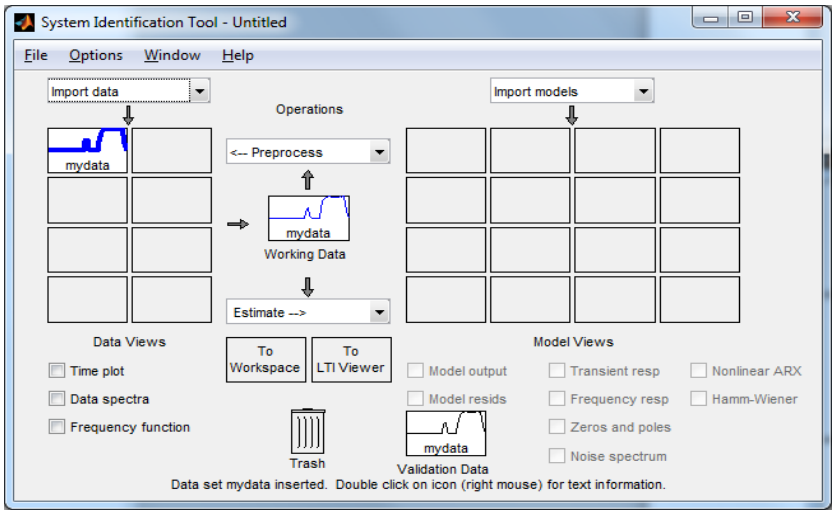

FIGURA 3.38: SELECCIÓN DE LOS DATOS DE TRABAJO EN EL IDENT

- **Paso 5:** Para poder ver nuestros datos obtenidos y con los cuales vamos a trabajar puesto que esta sombreada la figura de nombre <<mydata>> damos visto donde dice <<time plot>>.

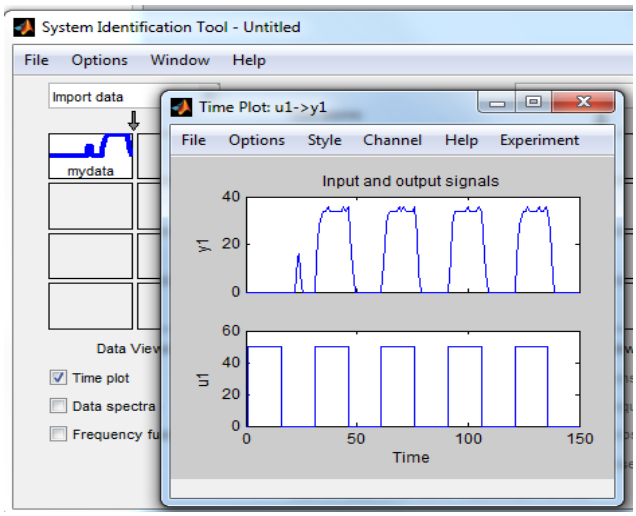

FIGURA 3.39: SELECCIÓN DE LOS DATOS DE TRABAJO EN EL IDENT

- **Paso 6:** Procederemos a seleccionar un rango de todos nuestros datos y vamos a elegir los mejores periodos para hacer una buena identificación.

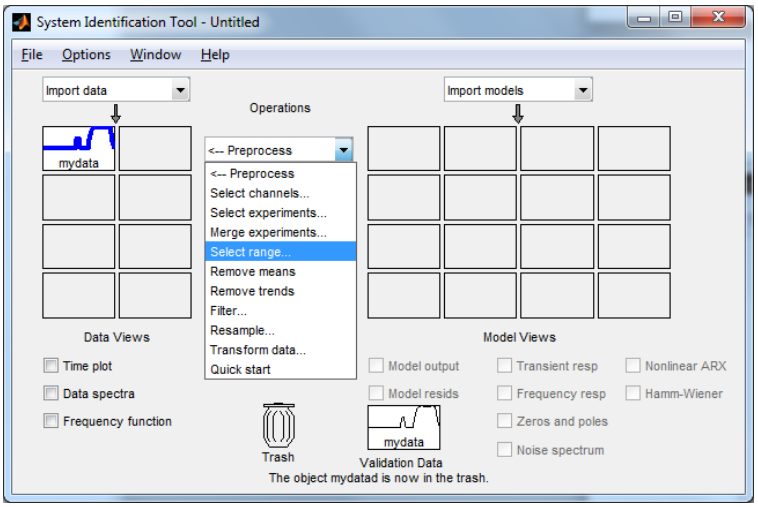

FIGURA 3.40: SELECCIÓN DE RANGO DE DATOS VÁLIDOS

Como podemos ver se escogieron los dos últimos periodos una vez seleccionado damos clic en insertar.

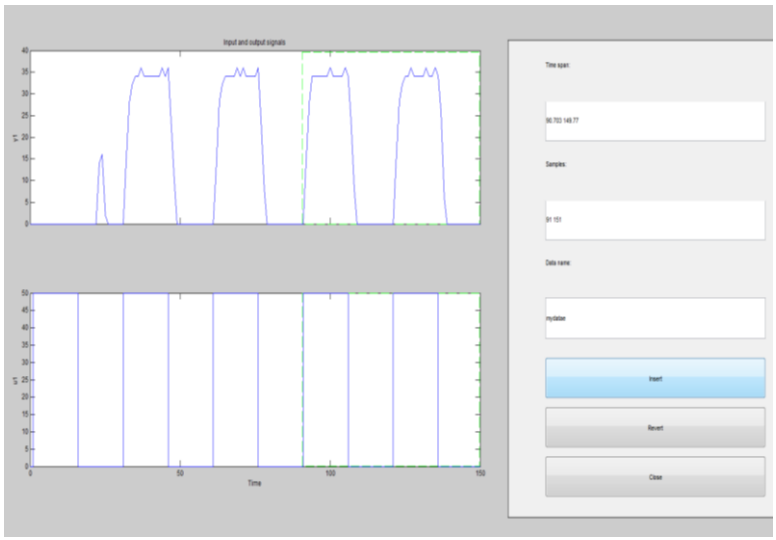

FIGURA 3.41: RANGO DE DATOS PARA EL WORKING DATA

- **Paso 7:** Podemos ver que se abre otra figura la cual se llama <<mydatae>>.

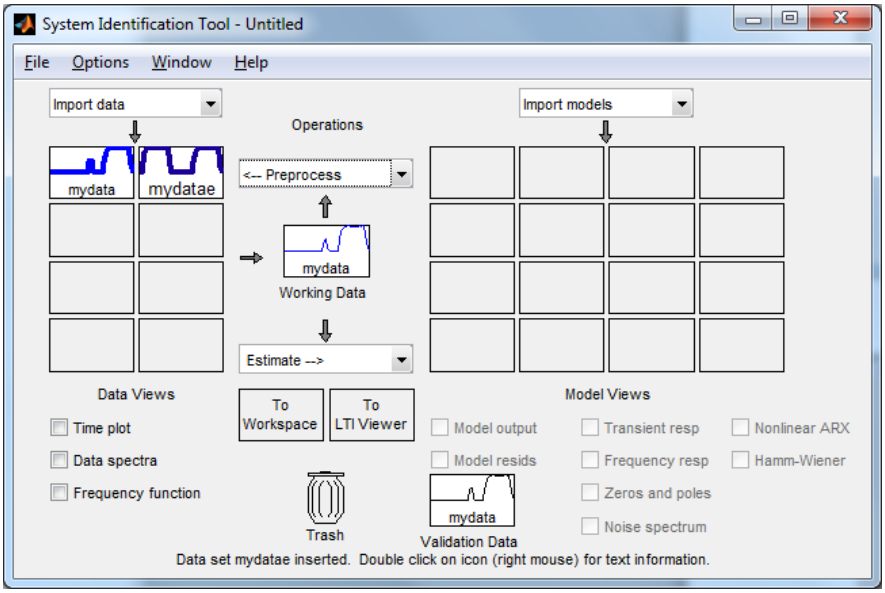

FIGURA 3.42: "MYDATAE" EN EL DATA VIEWS

Para proceder a trabajar con los datos seleccionados tenemos que arrastrar la figura que dice <<*mydatae*>> donde dice *working Data* y donde dice *validation Data*.

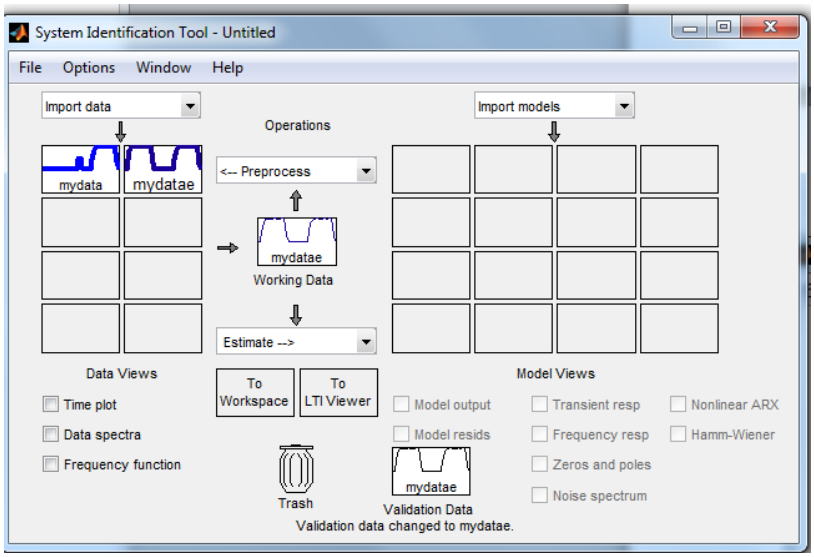

FIGURA 3.43: "MYDATAE" SELECCIONADO COMO DATOS DE TRABAJO

- **Paso 8:** Procedemos a dar clic donde está la pestaña que dice *estimate* y seleccionamos <<*Process Model*>>.

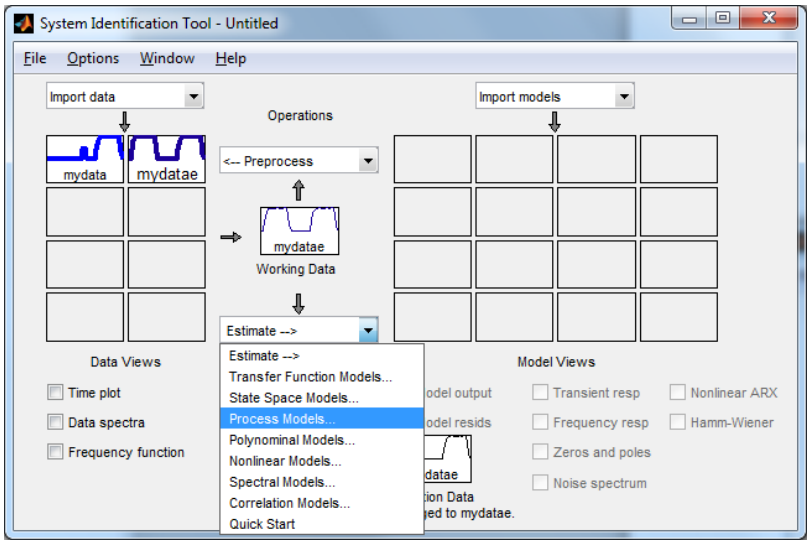

FIGURA 3.44: ESTIMACIÓN POR MODELOS DE PROCESOS

- **Paso 9:** Se nos abrirá una ventana la cual nosotros tenemos que poner que función de transferencia que podría ser nuestra planta podemos elegir <<1 polo, 2 polos, 1 polo con 1 zero, etc.>> tendremos que elegir de acuerdo a la experiencia q tengamos con nuestra planta. Y sabremos que nuestro modelo elegido es correcto cuando el porcentaje de aproximación sea mayor igual que 85%. Escogeremos <<1 polo>> procederemos a estimar dando clic en <<Estimate>>.

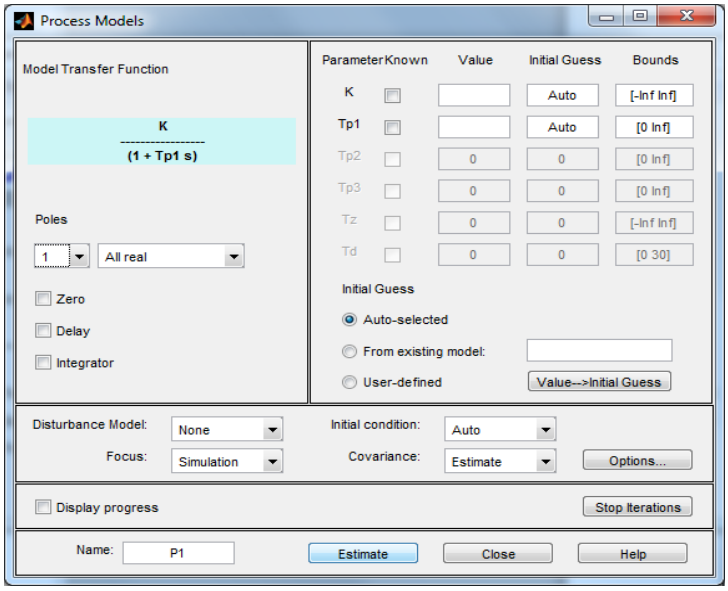

FIGURA 3.45: SELECCIÓN DEL ORDEN DE LA ECUACIÓN DEL SISTEMA A IDENTIFICAR

- **Paso 10:** Como podemos ver el porcentaje de aproximación es de 87% por lo tanto es un buen modelo que hemos elegido, se nos presentara en la parte derecha una figura llamada <<P1>> y una ventana donde podemos ver la aproximación los datos reales.

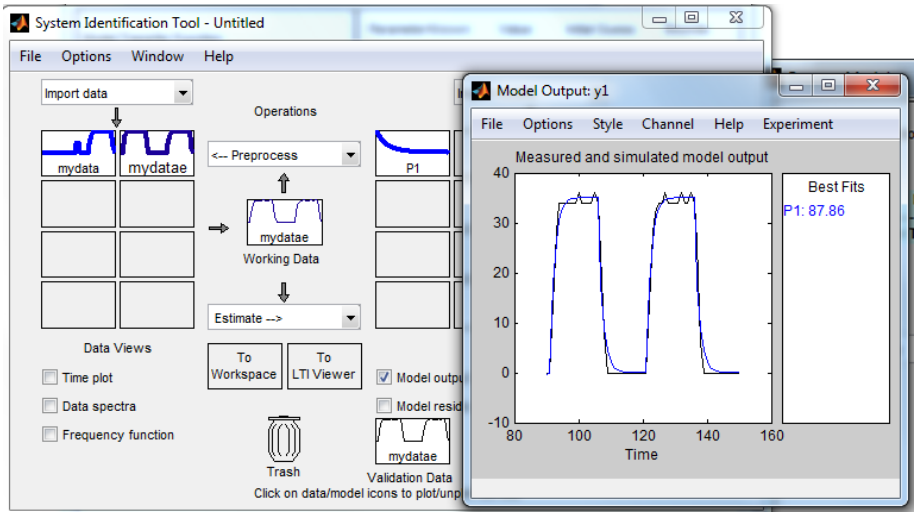

FIGURA 3.46: PORCENTAJE DE APROXIMACIÓN DE MODELOS DE ESTIMACIÓN

Aquí tenemos un ejemplo si deseamos probar con otro modelo por ejemplo (2 polos y un *zero*), luego de elegir ponemos *Estimate*.

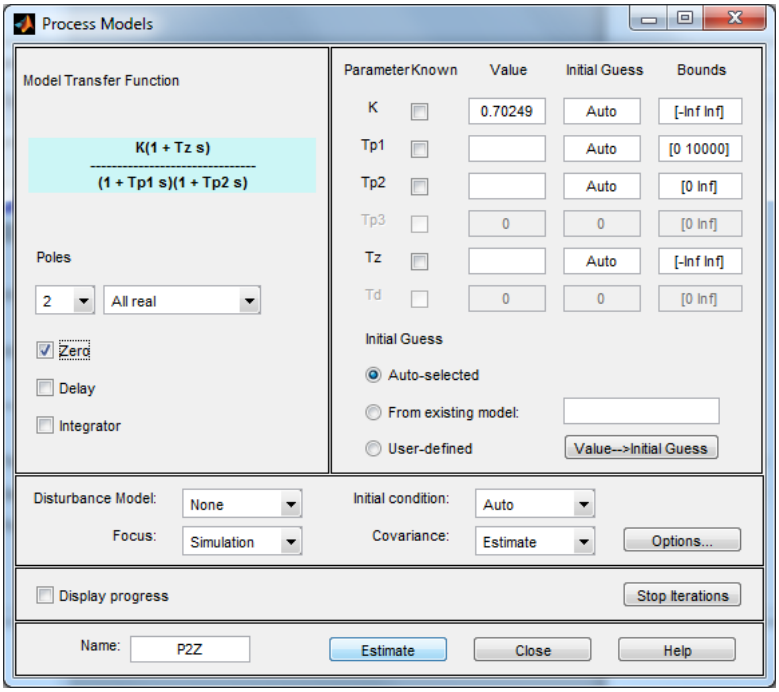

FIGURA 3.47: CAMBIO DEL ORDEN DE LA ECUACIÓN DEL SISTEMA A IDENTIFICAR

- **Paso 11:** Aquí vemos que se nos formó la cual es otra figurita llamada (P2Z) la cual es el segundo modelo que probamos y vemos que su porcentaje de aproximación es muy bajo por lo tanto seleccionamos el modelo (P1). Para ver la función de transferencia de mi modelo aproximado tenemos que seleccionar la figura (P1) y arrastrarla donde está un cuadrito llamado <*to Workspace*>.

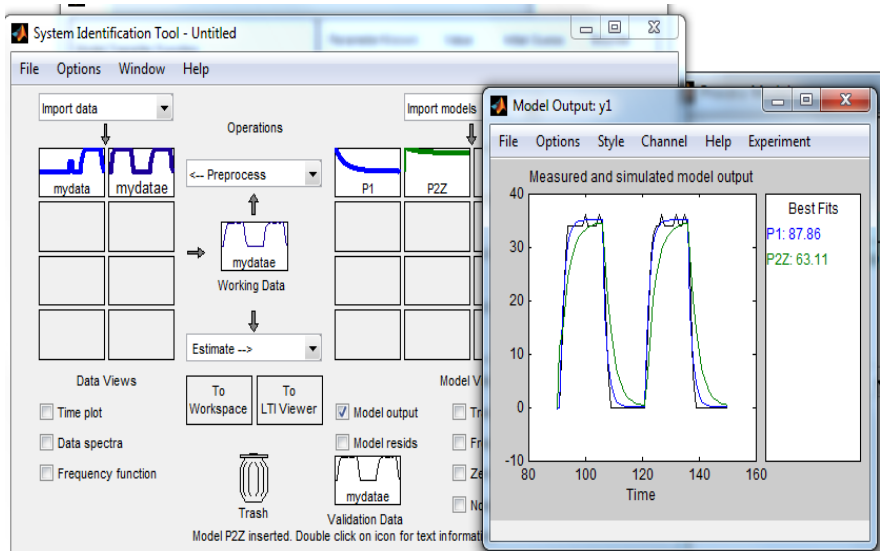

FIGURA 3.48: PORCENTAJE DE APROXIMACIÓN DE AMBAS ESTIMACIONES

- **Paso 12:** Una vez hecho el paso anterior tenemos nuestro modelo en el *workspace* entonces podemos trabajarlo en el *command window*. Ejecutamos la siguiente instrucción G=tf(P1) la cual me devuelve la función de trasferencia de mi modelo aproximado P1, donde G es una variable donde voy a guardar mi modelo y donde el comando <*tf*> significa transfer *function* (función de transferencia).

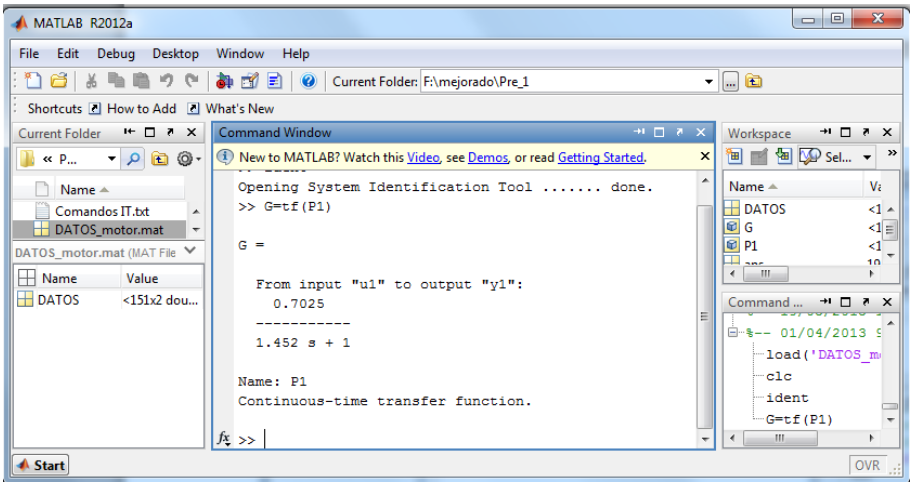

FIGURA 3.49: FUNCIÓN DE TRANSFERENCIA DEL MODELO

Finalmente hemos modelado nuestra planta, ahora se puede proceder a encontrar el controlador para mi sistema modelado, eso se realizara en la práctica siguiente.

Nota: Recordar que se hizo la identificación para un punto de operación dado es decir para una referencia específica, por lo tanto el controlador que hallemos solo podrá controlar nuestro sistema alrededor de mi punto operación.

3.2.4. CONTROLADOR DE LA PLANTA MOTOR DC

3.2.4.1 ANTECEDENTES

En la siguiente práctica se analizará una planta en la que se desea controlar la velocidad de un motor DC, tenemos que seleccionar un controlador adecuado para ajustar la respuesta del sistema con realimentación negativa a un comportamiento deseado.

3.2.4.2 OBJETIVOS

- Aplicar herramientas de simulación como el “SisoTool” de Matlab para obtener el ajuste del controlador del sistema.
- Determinar el tipo de controlador necesario para ajustar la respuesta del sistema en función de señales de prueba definidas de tal manera que tanto su error de estado estacionario como su dinámica respondan a especificaciones prefijadas.

3.2.4.3 TEORÍA

A continuación se muestra un diagrama de bloques típico de un sistema con realimentación negativa del tipo SISO (una entrada una salida).

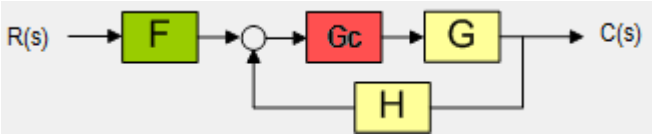

FIGURA 3.50: LAZO DE CONTROL CON RETROALIMENTACIÓN

De donde:

- **R(s)** es la señal de referencia que define el punto de operación de la planta,
- **C(s)** es la señal de salida o variable que se controla,
- **F** es el Pre filtro,
- **Gc** es el Controlador,
- **G** es la función de transferencia de la planta,
- **H** es la retroalimentación.

Los controladores pueden ser básicamente de los siguientes tipos (aunque existen variaciones de los mismos dependiendo de las condiciones):

PROPORCIONAL	INTEGRADOR	DIFERENCIADOR
$\mathbf{G_c(s) = K_1}$	$\mathbf{G_c(s) = \frac{K_1}{s}}$	$\mathbf{G_c(s) = K_1.s}$
PROPORCIONAL INTEGRADOR	PROPORCIONAL DIFERENCIADOR	PROPORCIONAL INTEGRAL DIFERENCIADOR
$\mathbf{G_c(s) = K_1 + \frac{K_2}{s}}$	$\mathbf{G_c(s) = K_1 + K_2.s}$	$\mathbf{G_c(s) = K_1 + \frac{K_2}{s} + K_3.s}$

TABLA 3.3: ALGORITMOS DE CONTROL

Por otra parte, al tener un sistema de control podemos establecer que cuando R(s) y C(s) están en las mismas unidades el Error del Sistema (E(s)) es:

$$\mathbf{E(s) = R(s) - C(s)}$$

Además sabemos que:

$$\frac{C(s)}{R(s)} = \frac{F.G_c.G(s)}{1 + G_c.G.H(s)}$$

Entonces:

$$E(s) = R(s)\left[1 - \frac{F.G_c.G(s)}{1 + G_c.G.H(s)}\right]$$

Si además, F(s)=H(s) nos queda:

$$E(s) = \frac{R(s)}{1 + G_c.G.H(s)}$$

Si el sistema es estable, entonces el error de estado estacionario (Ess) es:

$$e_{ss} = \lim_{s \to 0} s.E(s)$$

Por otra parte, si revisamos la Ecuación12 podemos ver que los polos de la función de transferencia están dados por la ecuación:

$$1 + G_c.G.H(s) = 0$$

A la que se conoce como Ecuación Característica del sistema y nos permite trazar el Lugar Geométrico de las Raíces del sistema. La herramienta "Sisotool" de Matlab nos permite ajustar el controlador cuando lo expresamos en la forma:

$$G_C(s) = K_P + \frac{K_I}{s} + sK_D = \frac{s^2 K_D + sK_P + K_I}{s} = K_D \frac{(s + z_1)(s + z_2)}{s}$$

$$K_P = K_D(z_1 + z_2)$$

$$K_I = K_D(z_1 * z_2)$$

En este caso, el modo Proporcional-Integral (PI) contribuye con un polo en el origen y un cero y en combinación con el modo diferencial (PID) contribuye con otro cero; en este caso, generalmente se fija primero la ubicación de los ceros y luego se varía el parámetro KD.

3.2.4.4 PRÁCTICA

Se requiere un controlador el cual me permita que mi sistema responda a los siguientes parámetros

- Ess(error de estado estacionario)=0.

- Sobre nivel Porcentual=15%.
- Tiempo de estabilización=0.7segundos.

Nota: los parámetros se ajustan de acuerdo a las condiciones físicas y pruebas con la planta.

- **Paso 1:** Para el diseño del controlador lo haremos con la herramienta sisotool en la cual ingresamos como parámetro la planta a la cual queremos controlar como lo vemos mi planta se encuentra en la variable G.

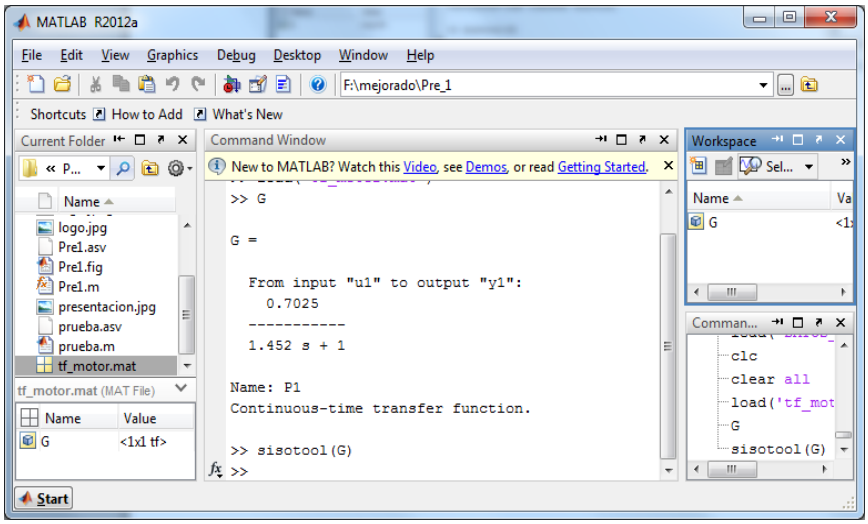

FIGURA 3.51: INGRESO DE LA PLANTA IDENTIFICADA

- **Paso 2:** Aquí podemos ver la trayectoria de las raíces y el grafico de bode de mi sistema en lazo cerrado, podemos dar cuenta que el sistema es estable en las trayectoria de las raíces, ahora no necesitaremos trabajar con el diagrama de bode así que lo cerraremos como veremos a continuación.

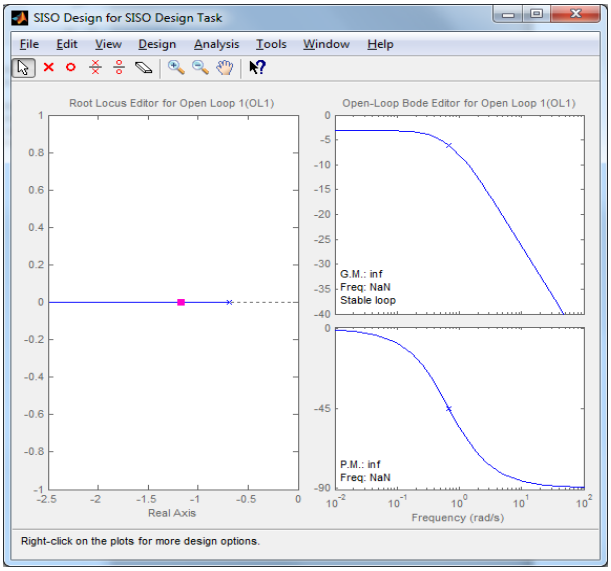

FIGURA 3.52: TRAYECTORIA DE LAS RAÍCES Y GRÁFICA DE BODE DEL SISTEMA

- **Paso 3:** En esta ventana (*Control and Estimation Tools Manager*) se encuentra algunas pestañas las cuales usaremos dos (*Graphical Tunning y Compesator Editor*) como lo mencionamos anteriormente no vamos a trabajar con el grafico de bode así que procederemos a cerrarlo**.**

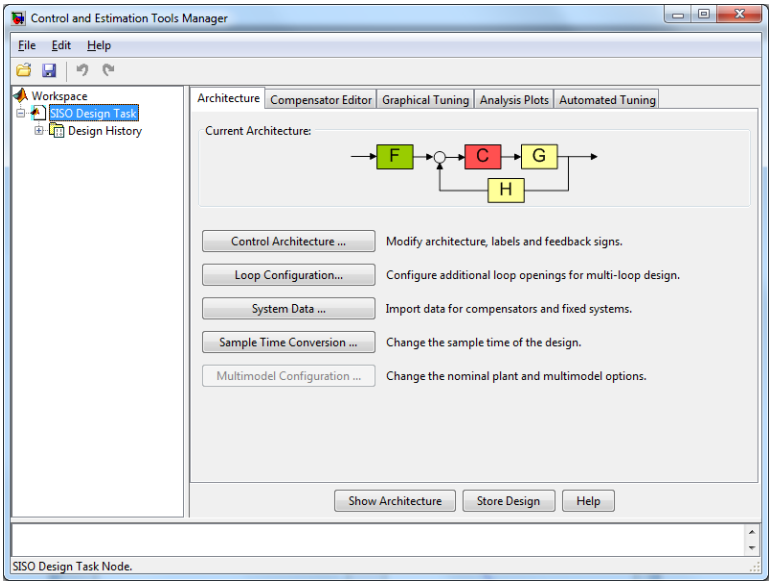

FIGURA 3.53: HERRAMIENTAS DE ESTIMACIÓN Y CONTROL

- **Paso 4:** Damos clic en la pestaña que dice *Graphical Tunning* y en la fila donde dice Plot2 escogemos la opción (*None*).

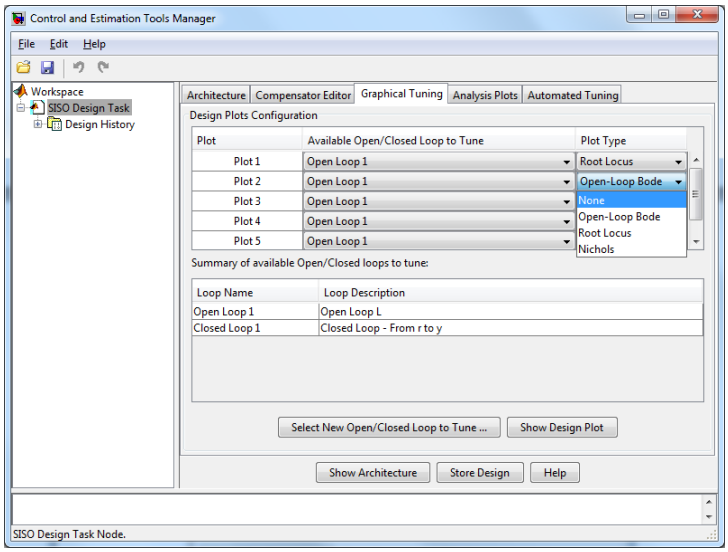

FIGURA 3.54: HERRAMIENTA GRAPHICAL TUNNING

- **Paso 5:** Como podemos ver solo nos queda la gráfica de la trayectoria de las raíces de mi sistema. La línea azul es la trayectoria de mis polos de lazo cerrado.

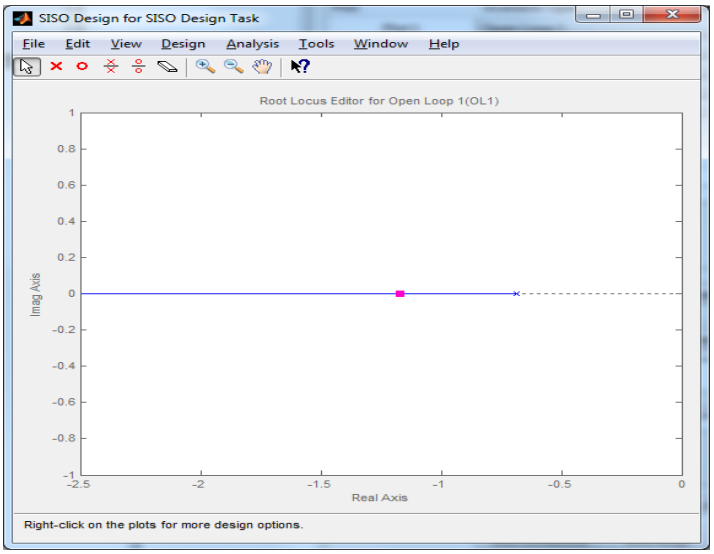

FIGURA 3.55: TRAYECTORIA DE LAS RAÍCES

- **Paso 6:** Para poder ver como variar ciertos parámetros de mi sistema y ver cómo responde, vamos a excitar al sistema con una entrada paso como lo vemos en la figura. Damos clic en Analysis y escogemos la opción (Response to Step Command).

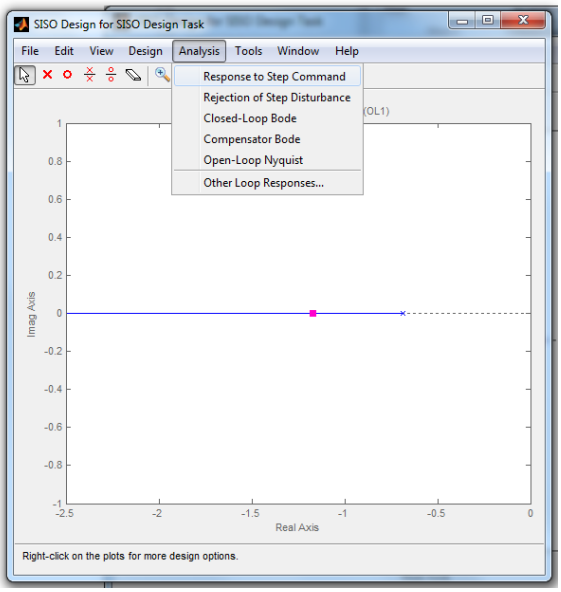

FIGURA 3.56: SELECCIÓN DE RESPUESTA ANTE UNA ENTRADA PASO

- **Paso 7:** Como vemos se nos forman dos respuestas deseleccionamos con el cual no vamos a trabajar entonces escogemos el sistema de color verde para quitarle el visto entonces damos clic derecho – System—closed loop r to u(green).

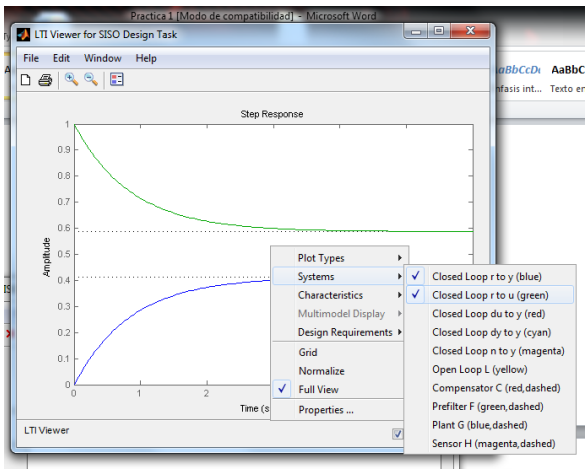

FIGURA 3.57: RESPUESTA ANTE UNA ENTRADA PASO

- **Paso 8:** Una vez tengamos listo la gráfica de nuestro sistema procederemos a ver los datos de interés como lo son settling time(tiempo de estabilizacion), Overshoot(sobre nivel porcentual), stady Ready(Aquí podemos ver el valor final que tendrá mi sistema).

  Para ver estos datos tenemos que dar clic derecho y poner características y seleccionamos los datos de interés ya mencionados.

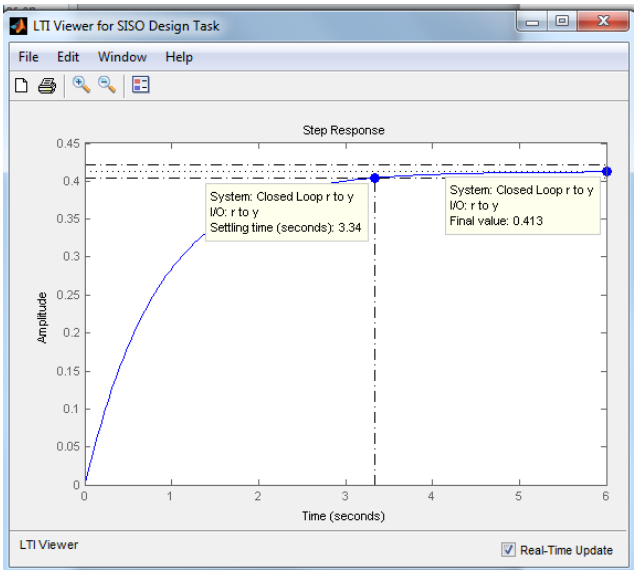

FIGURA 3.58: INFORMACIÓN DE LA RESPUESTA DEL SISTEMA ANTE UNA ENTRADA PASO

- **Paso 9:** Ahora vamos a trabajar con las trayectoria de las raíces, lo primero que vamos hacer es poner todos los requerimientos pedidos como lo son el tiempo de estabilización deseado y el sobre nivel porcentual deseado. Lo hacemos dando clic derecho—Diseño de Requerimientos –New.

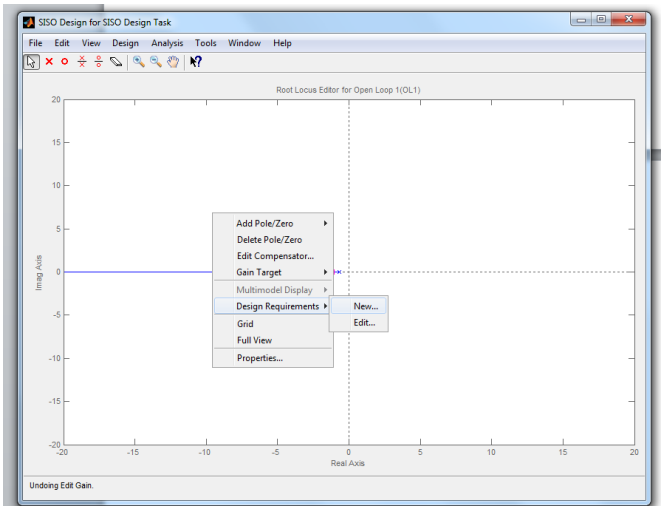

FIGURA 3.59: REQUERIMIENTOS EN LA GRÁFICA DE POLOS Y CEROS

- **Paso 10:** Se nos abrirá una ventanita en la cual tenemos que elegir el requerimiento, como se ve en la imagen se eligió el requerimiento Percent Overshoot y luego ponemos el valor en nuestro caso es 15%(solo se pone el 15) y luego agregamos de la misma manera el tiempo de estabilización deseado.

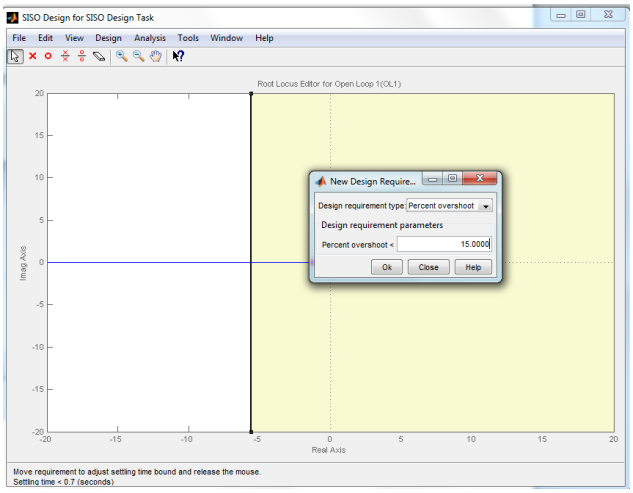

FIGURA 3.60: REQUERIMIENTOS DE SOBRE NIVEL PORCENTUAL

- **Paso 11:** Una vez elegidos los requerimientos vemos que se forman una línea recta de color negro, la cual significa que a lo largo de esa línea tendremos un tiempo de estabilización deseado, de la misma manera vemos que se formó dos líneas negras que forman una especie de triángulo, si recorremos el triángulo en cada punto tendremos el sobre nivel porcentual deseado.

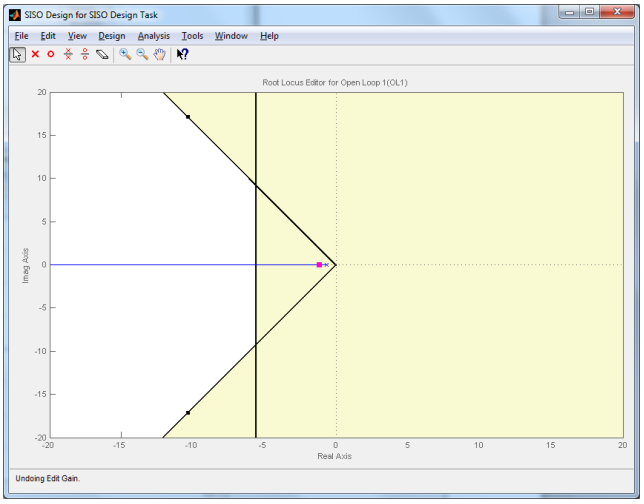

FIGURA 3.61: REQUERIMIENTO DE TIEMPO DE ESTABLECIMIENTO

- **Paso 12:** Nos falta un parámetro y ese es el error de estado estacionario, Para eso tenemos que agregar un integrador, porque se desea que el sistema tenga error 0 y además la planta es tipo 0.

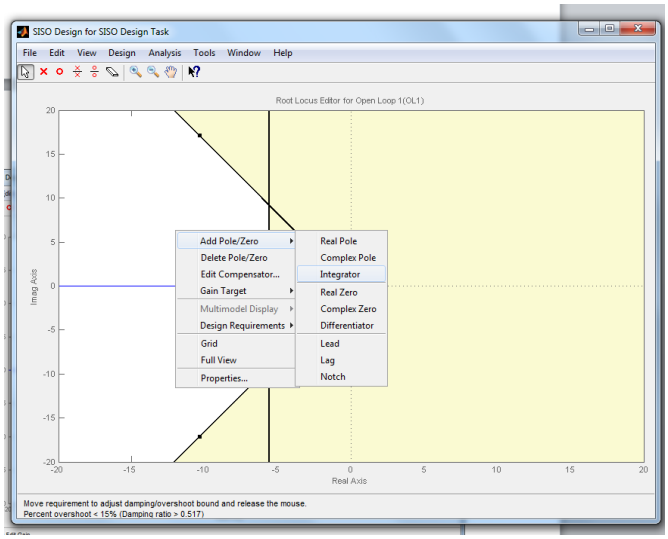

FIGURA 3.62: INTEGRADOR PARA ERROR ESTADO ESTACIONARIO CERO

- **Paso13:** El objetivo es hacer que la posición de mis polos de lazo cerrado en este caso los cuadritos rojos estén en la intersección de la las líneas negras porque justo en esa posición tenemos los parámetros deseados. Podemos ver que nuestros que la trayectoria de las raíces (la línea azul) nunca pasara por la intersección de las líneas negras, entonces tenemos que modificar la trayectoria de las raíces.

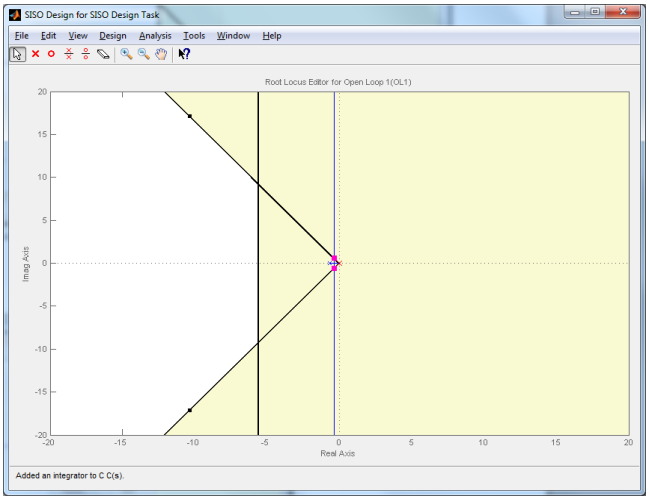

FIGURA 3.63: MODIFICACIÓN DE LA TRAYECTORIA DE LAS RAÍCES

- **Paso 14:** Para modificar la trayectoria de las raíces tenemos que agregar un zero y lo hacemos de la siguiente forma, en la barra de herramientas vemos unos dibujos ahí seleccionamos la imagen que representa un zero como se puede ver en la figura lo seleccionamos y lo ponemos en la línea entre cortada.

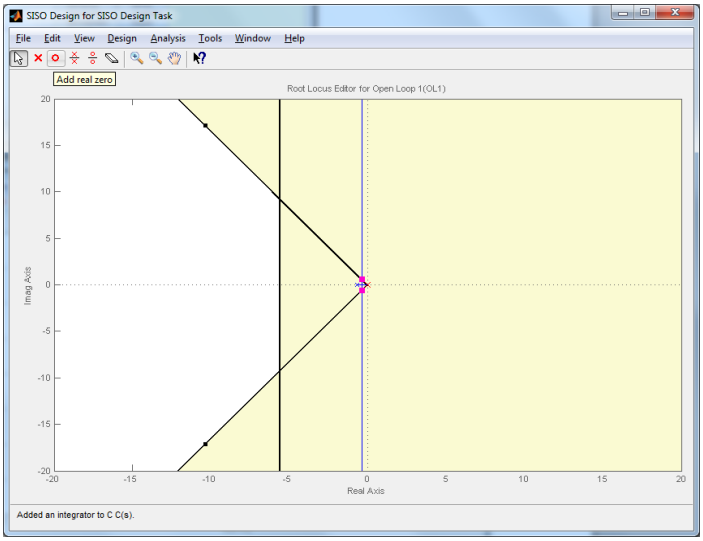

FIGURA 3.64: MODIFICACIÓN DE LA TRAYECTORIA DE LAS RAÍCES

- **Paso 15:** Aquí podemos ver cómo cambio nuestra trayectoria de las raíces, pero aun mi trayectoria (línea azul) no pasa por la intersección de las líneas negras. Tenemos que hacer que la trayectoria pase por la intersección de mis requerimientos.

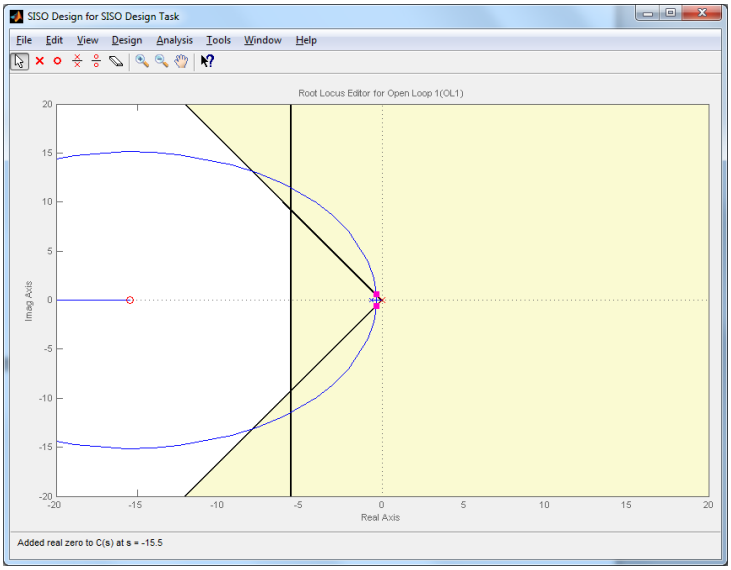

FIGURA 3.65: AJUSTES DEL CONTROLADOR

- **Paso 16:** Ahora procedemos a mover el zero que agregamos, podemos moverlo izquierda o derecha eso va a depender de cómo se comporte nuestra trayectoria de las raíces y hacer que pase por la intersección de las líneas negras como vemos en nuestro ejemplo.

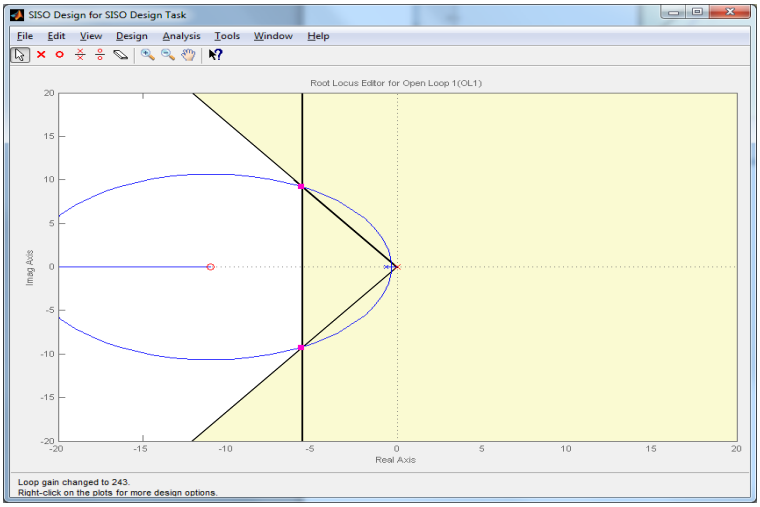

FIGURA 3.66: AJUSTES DEL CONTROLADOR

- **Paso 17:** Una vez mi trayectoria pase por la intersección de los requerimientos tenemos que hacer que los puntos rojos los cuales representan donde van a estar nuestros polos de lazo cerrado estén en la intersección, entonces los seleccionamos y los arrastramos siguiendo la línea azul y los colocamos en la intersección como vemos en la gráfica.

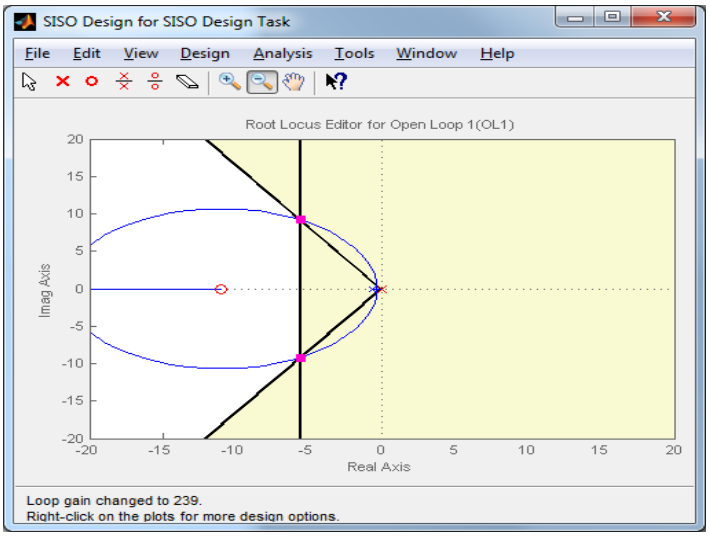

FIGURA 3.67: AJUSTES DEL CONTROLADOR

- **Paso 18:** Ahora podemos verificar en la ventana donde está la respuesta de nuestro sistema a una entrada paso, como está respondiendo el sistema y ver los datos de interés y tenemos error cero, tiempo de estabilización de 0.69 y el sobre nivel porcentual de 26.9%. Cumplimos con los requisitos menos con el sobre nivel porcentual, eso es debido al efecto del zero que agregamos más adelante veremos cómo mejorarlo.

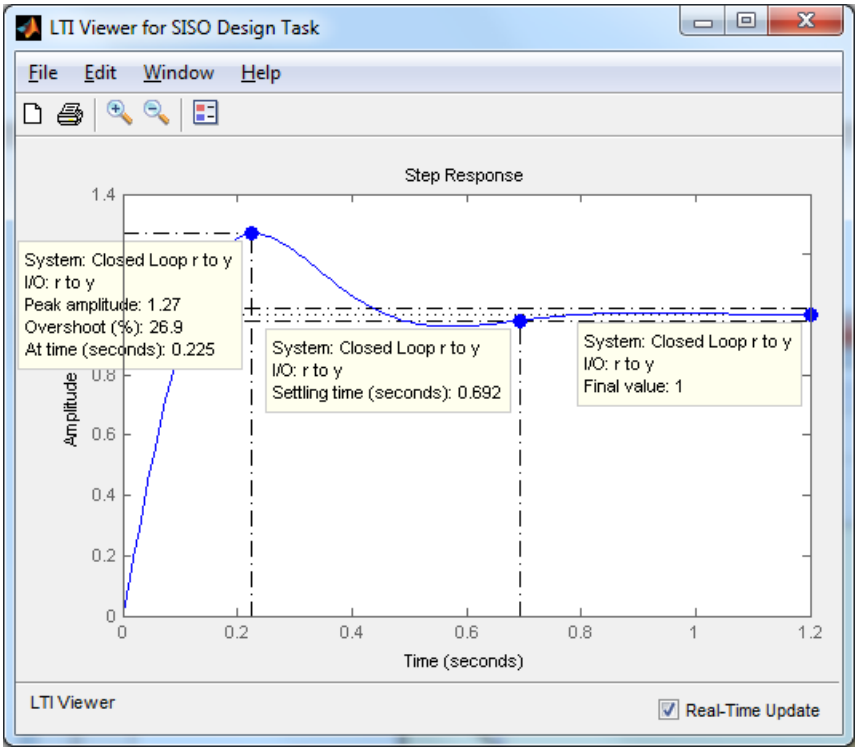

FIGURA 3.68: RESPUESTA DEL SISTEMA EN LAZO CERRADO

- **Paso 19:** Ahora procederemos a revisar el controlador que hemos creado en la ventana *(Control and Estimation Tools Manager)* y seleccionamos la pestaña *Compensator Editor*. Podemos ver que tenemos un zero y un integrador este

controlador es un PI (*Proporcional integral*). Para eliminar el efecto del zero de mi controlador usaremos un pre filtro. Lo primero que tenemos que hacer es copiar la localización del zero de mi controlador, podemos ver que está localizado en -11.022.

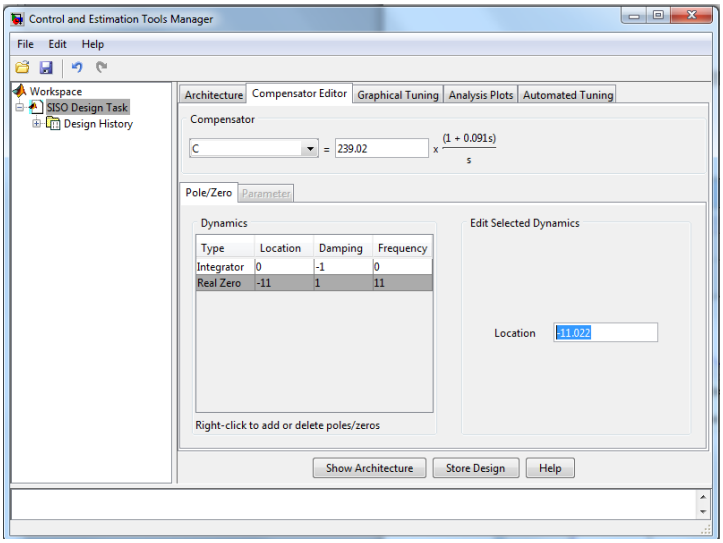

FIGURA 3.69: DISEÑO DEL CONTROLADOR

- **Paso 20:** Una vez tengamos la localización del Zero del controlador tenemos que proceder a ir la lista y seleccionar la letra F eso significa que vamos a crear nuestro pre filtro, como vemos en la figura.

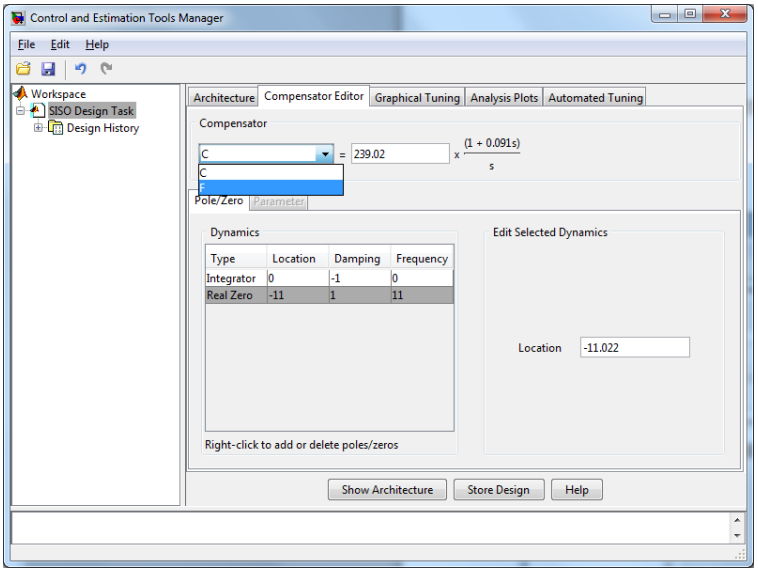

FIGURA 3.70: DISEÑO DEL PRE FILTRO

- **Paso 21:** En la parte que dice Dynamics damos clic derecho y agregamos un Polo (y así anulamos el efecto del Zero del controlador) y en Location ponemos la localización que habíamos copiado anteriormente.

-

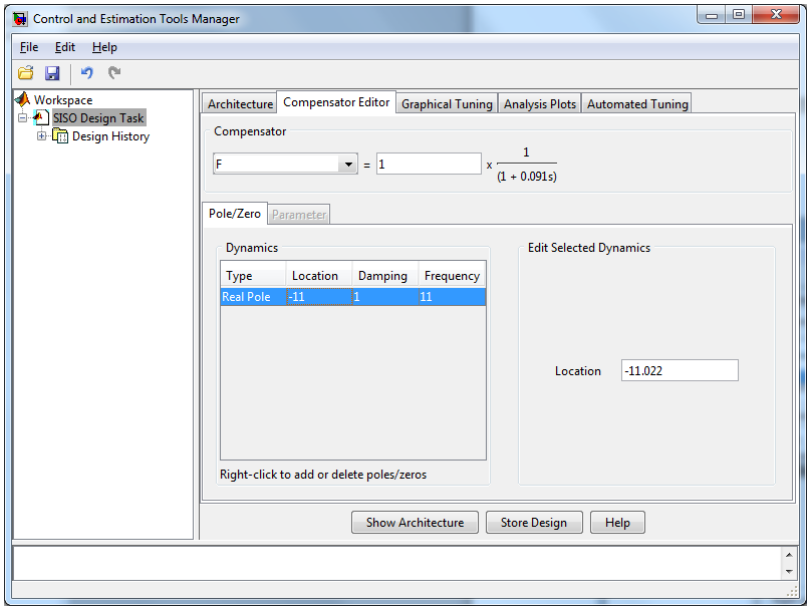

FIGURA 3.71: DISEÑO DEL PRE FILTRO

- **Paso 22:** Aquí podemos ver como nuestro sistema cumple con todas las condiciones propuestas.

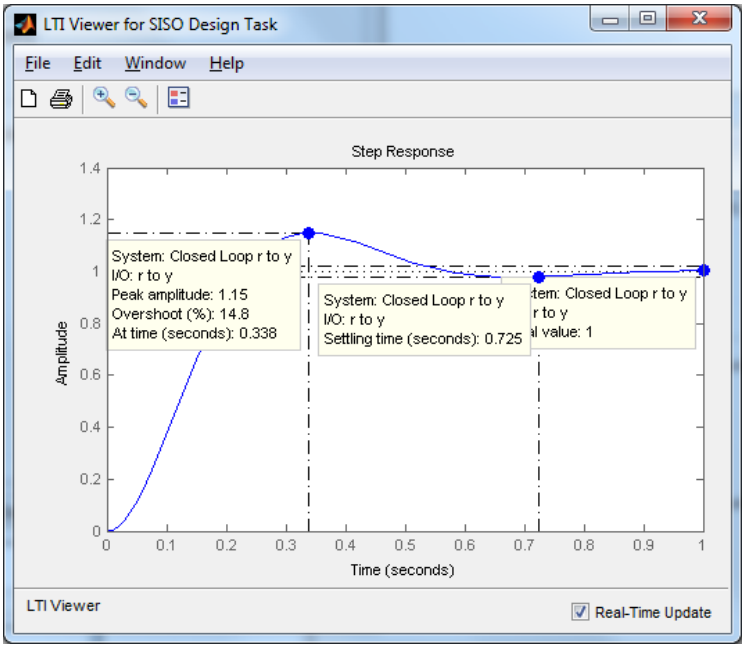

FIGURA 3.72: RESPUESTA DEL SISTEMA CONTROLADO

- **Paso 23:** Para poder tener mi controlador en el workspace tenemos que exportarlo y lo hacemos de la siguiente manera, clic en file y seleccionamos Export y se nos abrirá una nueva ventana.

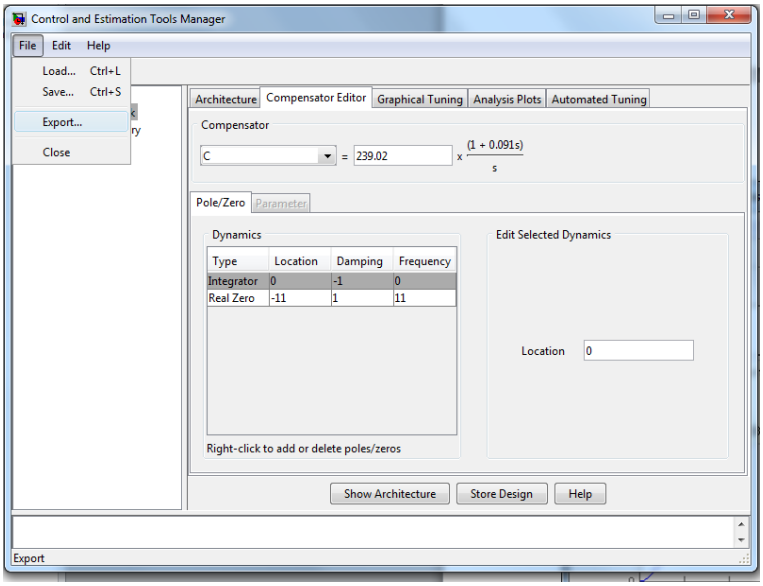

FIGURA 3.73: CONTROLADOR LISTO PARA EXPORTAR

- **Paso 24:** En esta ventana seleccionamos Compensator C y damos clic en el botón (Export to Workspace) si queremos exportar el prefiltro hacemos lo mismo pero tendremos que seleccionar Prefiltre F.

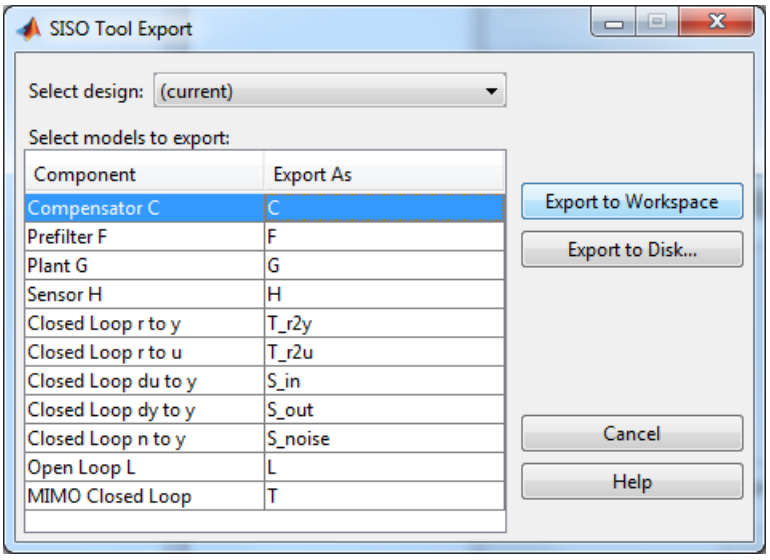

FIGURA 3.74: EXPORTANDO EL CONTROLADOR

- **Paso 25:** Para ver mi controlador ejecutamos la siguiente instrucción tf(C) y podemos ver nuestro controlador PI donde el valor de Kp=21.69 y el Ki = 239.

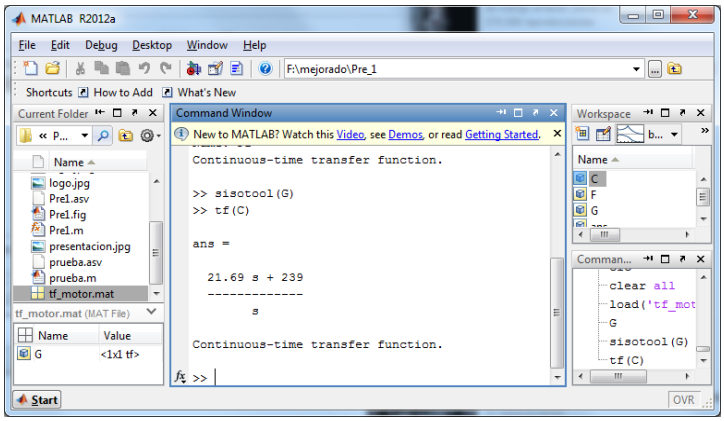

FIGURA 3.75: FUNCIÓN DE TRANSFERENCIA DEL CONTROLADOR

- **Paso 26:** También podemos ver nuestro pre filtro ejecutando la siguiente instrucción tf(F).

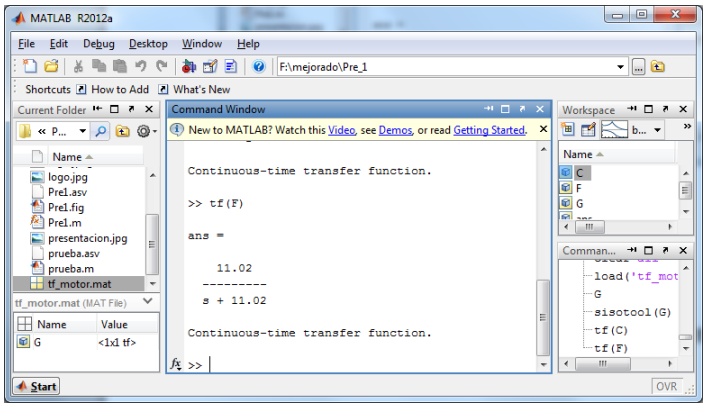

FIGURA 3.76: TRANSFER FUNCTION DEL PRE FILTRO

Nota: Finalmente hemos obtenido el controlador para mi planta Motor DC recordar que el controlador que hallamos funcionara de una manera correcta para valores alrededor de mi punto de operación, el cual es aproximadamente 33 - 34 RPS valor el cual se estabilizo la planta.

CAPÍTULO 4

PRUEBAS

4.1 ANÁLISIS DE RESULTADOS

En la adquisición de datos existen pequeños ruidos producto de interferencias de comunicación, tipo de cable usado para conectar los sensores, vibraciones de la plataforma cuando el rotor del motor gira a su máxima velocidad, procesamiento del computador, etc.

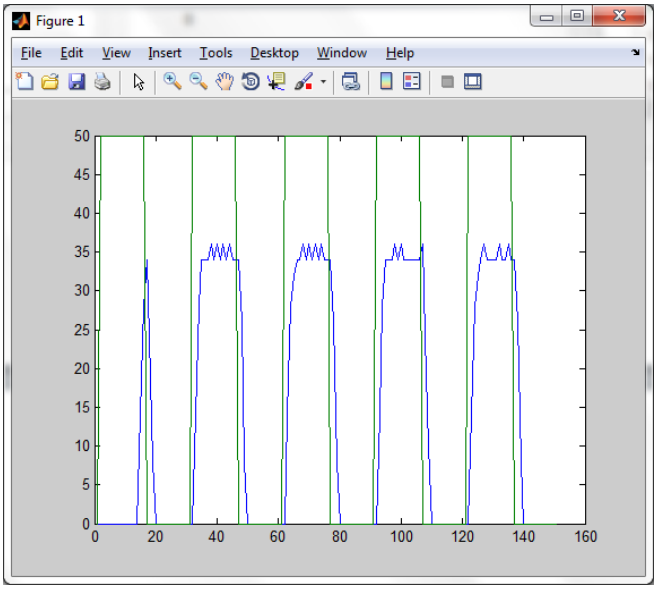

FIGURA 4.1: PRESENCIA DE RUIDO EN LA ADQUISICIÓN

Esta presencia de ruido nos permite trabajar sin mayor problema en la identificación del sistema gracias a las herramientas de pre-filtrado de la señal que nos ofrece la herramienta de identificación de sistemas de Matlab.

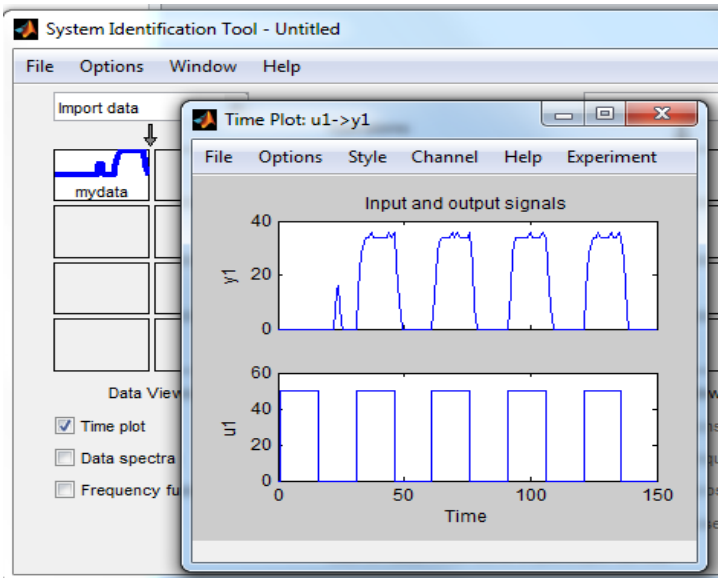

FIGURA 4.2: VISUALIZACIÓN DE LOS DATOS DE TRABAJO

Indiferentemente que exista o no ruido en los procesos en los que estemos midiendo o adquiriendo datos es recomen hacerlo durante mínimo 5 ciclos de trabajo del proceso. Por lo que como notamos estaos cumpliendo con el grupo de datos adquiridos.

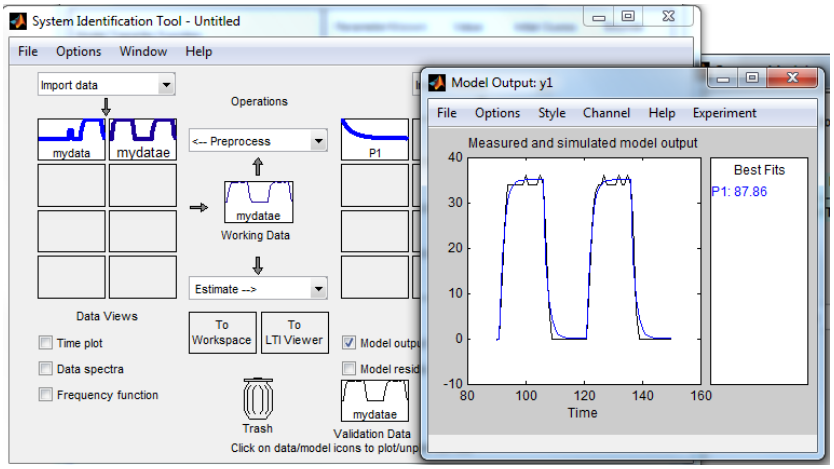

FIGURA 4.3: PORCENTAJE DE APROXIMACIÓN DE MODELOS DE ESTIMACIÓN

Con el porcentaje de aproximación mayor a un 70% podemos tener una buena identificación del sistema para su posterior diseño de controlador.

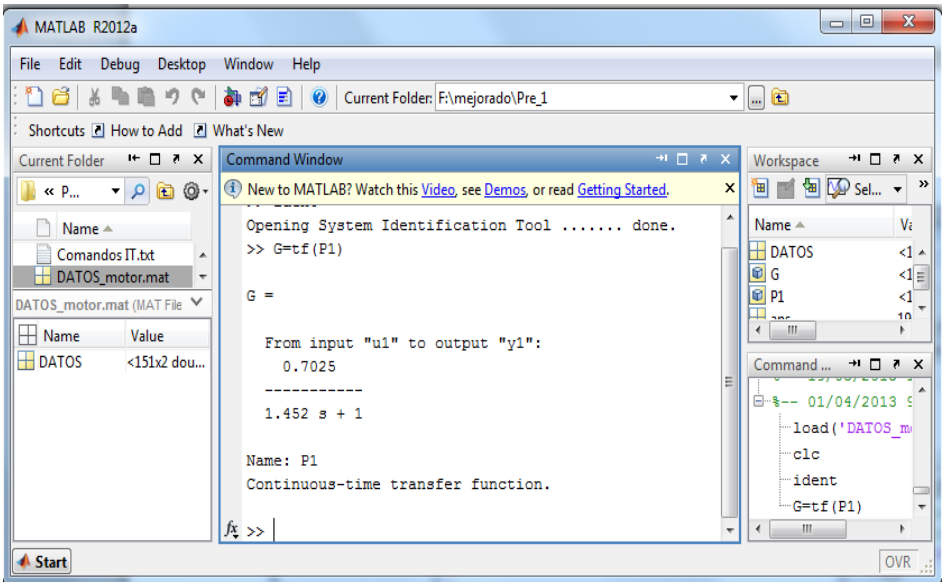

FIGURA 4.4: FUNCIÓN DE TRANSFERENCIA DEL MODELO DE ESTIMACIÓN

Con la ecuación característica identificada procedemos por medio del sisotool a crear el controlador adecuado cuya respuesta en lazo cerrado ya con el control nos da una respuesta satisfactoria.

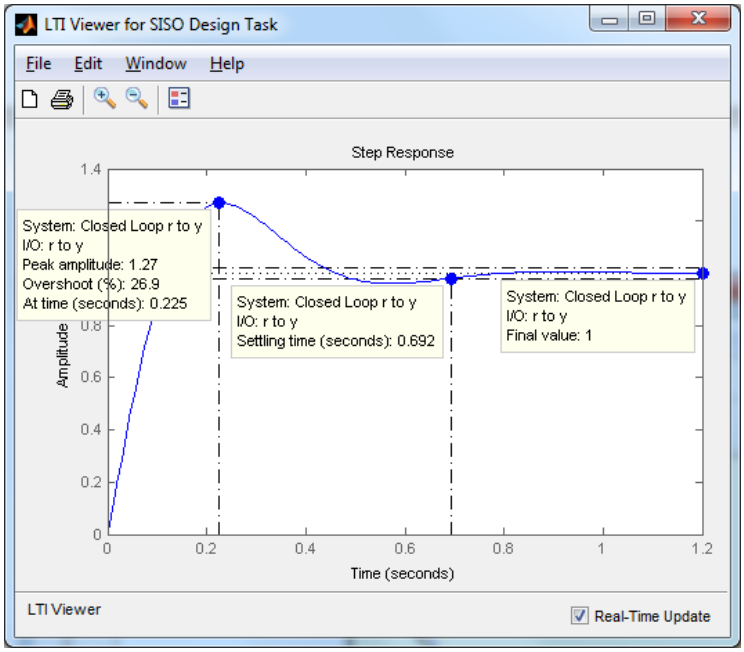

FIGURA 4.5: RESPUESTA DEL SISTEMA EN LAZO CERRADO CON EL CONTROLADOR

4.2 VALIDACIÓN DEL SISTEMA Y DE LAS PRUEBAS

La comprobación la realizaremos con la herramienta de Matlab Simulink, simularemos el proceso completo en lazo cerrado incluido el controlador encontrado.

Los parámetros que necesitamos para realizar la correcta simulación son:

- Pulse Generator: Duty cicle 50%, periodo 2s, amplitud 1.
- PID: Controlador que hallamos Kp=21.69, Ki=239.
- Stop time 5 segundos.

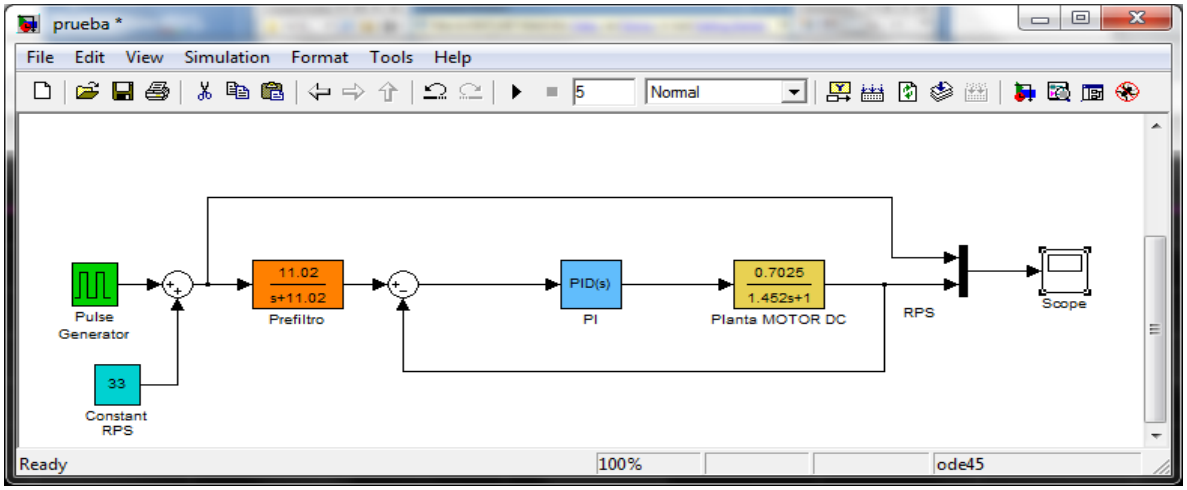

FIGURA 4.6: SISTEMA COMPLETO EN SIMULINK

Para la validación completa se ha incluido la ecuación de la planta identificada, el algoritmo de control encontrado con sisotool y el prefiltro, todo esto en lazo cerrado.

La respuesta del sistema frente a una señal de referencia actúa tal y como se buscó que actuara bajo la acción del controlador. De esta forma queda demostrado que el sistema identificado ha sido controlado con el algoritmo de control.

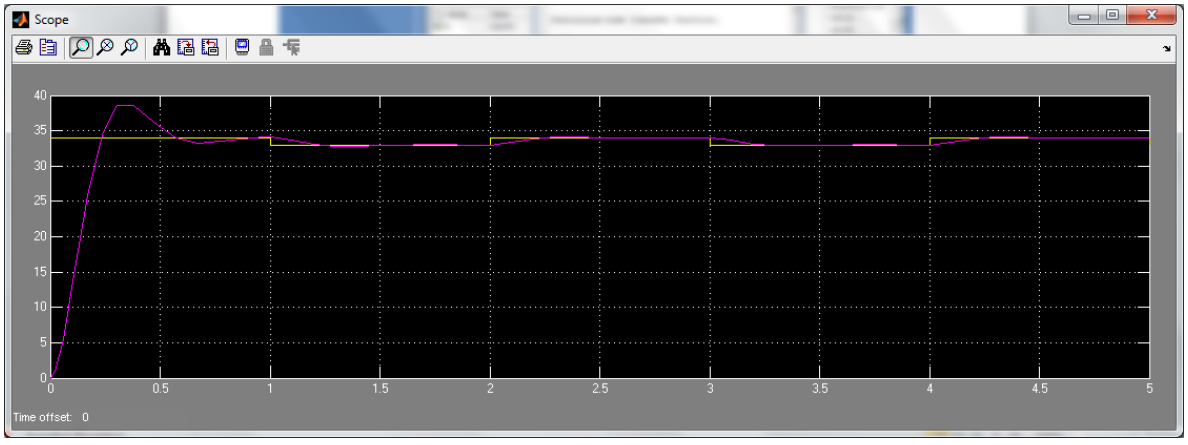

FIGURA 4.7: RESPUESTA DEL SISTEMA COMPLETO

CONCLUSIONES

- Esta herramienta de hardware nos permitió reforzar conocimientos teóricos haciendo pruebas con plantas en hardware académicas reales, interactuando con Matlab.

- Es necesario entender la importancia de la correcta selección y adecuación de los sensores en cualquier proceso que se desee medir y controlar.

- Es de gran ayuda utilizar equipos modulares o bloques básicos en hardware para realizar grandes aplicaciones con fines académicos o prácticos.
- Proyectos que nos permiten desarrollar herramientas para su posterior uso en fines académicos es de utilidad para estudiantes y profesores, en el desarrollo de automatización a bajo costo.

- Las guías de práctica son de suma importancia en el desarrollo de cualquier herramienta ya que nos permiten reproducir los experimentos para su mayor provecho académico.

- Es importante que en las prácticas de adquisición de datos se deban considerar los tiempos de muestreo y tiempo de adquisición en función de la planta con la cual se esté trabajando.

- Para tener éxito en las prácticas de identificación, diseño de controlador y verificación del controlador, se debe contar con excelentes datos adquiridos con la planta de hardware por lo que si es necesario se deberán repetir las adquisiciones hasta tener datos legibles. Esta libertad de reproducir la adquisición o medición de datos es muy complicado con procesos industriales reales por los costos y materia prima que implica dicho experimento en lazo abierto.

RECOMENDACIONES

- Utilizar medios de comunicación inalámbrica para que la maqueta realice un broadcast para muchos estudiantes y así todos adquieran datos al mismo tiempo para que trabajen en sus prácticas de adquisición.

- Realizar adquisiciones de datos de la planta generadora de calor con periodos de muestreo mayores a 4 horas.

- En caso de querer integrar plantas con tiempos de respuesta más rápidos de del motor DC, utilizar comunicación serial síncronas con SPI o I2C.

- Si se desea incrementar el número de plantas se recomienda utilizar hardware de mejores características de procesamiento como FPGA.

- Minimizar la longitud de cable entre el sensor y el módulo ADC para evitar ruido.

- Usar un cable USB apantallado o sistemas de radiocomunicación Xbee.

- Preferiblemente utilizar sensores que nos den señales de corriente de (4 a 20 mA o 0 a 20 mA) que son más inmunes al ruido.

- Es importante que el encoder óptico no roce con el disco acoplado en el rotor del motor para evitar mediciones incorrectas.

- Utilizar módulos de hardware desarrollados por empresas nacionales que permita tener soporte y contingencia inmediata.

ANEXOS

ANEXO I – CÓDIGO FUENTE DEL MÚDULO MEI&T04

```
program PLANTA_INDUSTRIAL_V2

SYMBOL CRUCE_CERO=PORTB.0

SYMBOL DISPARO=PORTC.3

SYMBOL RELAY=PORTC.5

SYMBOL TEMPERATURA=PORTC.4

SYMBOL ENCODER=PORTC.0

SYMBOL DIR2=PORTA.3

SYMBOL NDIR2=PORTA.5

SYMBOL PWM1=PORTC.2

SYMBOL LDR=PORTA.2

SYMBOL ANLG0=PORTA.0

SYMBOL ANLG1=PORTA.1


DIM INTENSIDAD,DUTY,RECEIVE,INDICE,ORDEN AS BYTE

DIM LUZ,VELOCIDAD,RETARDO AS BYTE[4]

DIM CONTROL_L,CONTROL_M,i,j,PULSOS,LDR_ANALOG as word

DIM TEMP_LOW,TEMP_HIGH ,tiempo AS WORD

DIM TEMP_ENTERA, TEMP_DECIMAL, TEMP_TOTAL, TEMP_INICIAL,

TEMP_FINAL    AS WORD
```

```
DIM MENOS,SENSOR ,ACTUADOR,  OPCION_ENCODER,  OPCION_TEMP,
OPCION_LDR AS BYTE
DIM OPCION_MOTOR,OPCION_LUZ AS BYTE
DIM TXT AS CHAR[5]

' Declarations section
SUB PROCEDURE interrupt()
    IF(PIR1.RCIF=1)then
      IF(UART1_Data_Ready = 1) then
       RECEIVE = Uart1_Read
       SELECT CASE INDICE
       CASE 0
        IF(receive="I")then
          INDICE=1
        else
          INDICE=0
        END IF
       CASE 1
        IF(receive="T")then
          INDICE=2
        ELSE
          INDICE=0
```

```
  END IF
CASE 2
  IF(receive="+")then
    INDICE=3
  ELSE
     INDICE=0
  END IF
CASE 3
   IF(receive="S")then
    INDICE=4
   ELSE
   IF(receive="A")THEN
     INDICE=5
   ELSE
   IF(receive="D")THEN
     INDICE=11
   ELSE
   IF(receive="L")THEN
     INDICE=8
     CONTROL_L=1
   ELSE
   IF(receive="M")THEN
```

```
    INDICE=8
    CONTROL_M=1
    END IF
   END IF
  END IF
 END IF
END IF
CASE 4
  SENSOR=RECEIVE
  SENSOR=SENSOR-0X30
  INDICE=6
CASE 5
  ACTUADOR=RECEIVE
  ACTUADOR=ACTUADOR-0X30
  INDICE=6
CASE 6
 IF(receive="=")then
  INDICE=7
 ELSE
  INDICE=0
 END IF
CASE 7
```

```
   ORDEN=RECEIVE
   ORDEN=ORDEN-0X30
   INDICE=12
CASE 8
 IF(receive="")then
   IF(CONTROL_L=1)THEN
   INDICE=9
   CONTROL_L=0
   END IF
   IF(CONTROL_M=1)THEN
   INDICE=10
   CONTROL_M=0
   END IF
   i=0
 ELSE
  INDICE=0
 END IF
CASE 9
 LUZ[i]=RECEIVE
 inc(i)
 IF(i=3)then
  LUZ[0] =( LUZ[0]-0X30)
```

```
 LUZ[1] =( LUZ[1]-0X30)
 LUZ[2] =( LUZ[2]-0X30)
 INTENSIDAD = LUZ[0]*100 + LUZ[1]*10 + LUZ[2]
 INDICE=0
 i=0
END IF
CASE 10
 VELOCIDAD[i]=RECEIVE
 inc(i)
 IF(i=3)then
  VELOCIDAD[0] =( VELOCIDAD[0]-0X30)
  VELOCIDAD[1] =( VELOCIDAD[1]-0X30)
  VELOCIDAD[2] =( VELOCIDAD[2]-0X30)
  DUTY = VELOCIDAD[0]*100 + VELOCIDAD[1]*10 + VELOCIDAD[2]
  INDICE=0
  i=0
 END IF
CASE 11
  IF(receive="=")then
   INDICE=12
    j=0
  ELSE
```

```
        INDICE=0
        END IF
      CASE 12
        RETARDO[J]=RECEIVE
        INC(J)
        IF(J=3)THEN
        RETARDO[0] =( RETARDO[0]-0X30)
        RETARDO[1] =( RETARDO[1]-0X30)
        RETARDO[2] =( RETARDO[2]-0X30)
        TIEMPO = RETARDO[0]*100 + RETARDO[1]*10 + RETARDO[2]
        INDICE=0
        J=0
        END IF
      END SELECT
    END IF
      PIR1.RCIF=0   ' SI EL DATO HA LLEGADO LIMPIO LA BANDERA DE
RECEPCIÓN
      PIE1.RCIE=1   ' HABILITAR NUEVAMENTE LA INTERRUPCIÓN POR
USART
      INTCON.TMR0IF=0
      INTCON.INTF=0
      PIR1.TMR1IF=0
```

```
 END IF


IF (PIR1.TMR1IF=1) THEN
    INC(PULSOS)
    TMR1H = $FF        ' INITIALIZAR TIMER1 REGISTER
    TMR1L = $FF        ' INITIALIZER TIMER1 REGISTER
    PIR1.TMR1IF=0
  ' PIE1.TMR1IE=1
END IF


IF ((INTCON.TMR0IF=1)AND ( OPCION_ENCODER=0)) THEN
    INTCON.TMR0IF=0
    DISPARO = 1
    INTCON.TMR0IE = 0
    INTCON.TMR0IF=0
    INTCON.INTE = 1
END IF
IF (INTCON.INTF=1) THEN
    INTCON.INTF=0
    TMR0 = INTENSIDAD
    DISPARO = 0
    INTCON.TMR0IF=0
```

```
        INTCON.TMR0IE = 1
        INTCON.INTE = 0
    END IF
END SUB

SUB PROCEDURE VALIDAR_ORDEN()
  IF(INDICE=12)THEN
   IF(SENSOR<>0)THEN
    SELECT CASE SENSOR
     CASE 1
      IF(ORDEN=1)THEN
       OPCION_ENCODER=1
      ELSE
       OPCION_ENCODER=0
       T1CON.TMR1ON=0
       PIR1.TMR1IF=0
       PIE1.TMR1IE=0
       INTCON.INTE = 0
       INTCON.TMR0IE = 0
       INTCON.TMR0IF=0
       INTCON.INTF=0
       PIR1.RCIF=0
```

```
 DISPARO=0
 END IF
CASE 2
 IF(ORDEN=1)THEN
 OPCION_TEMP=1
 ELSE
 OPCION_TEMP=0
 END IF
CASE 3
 IF(ORDEN=1)THEN
 OPCION_LDR=1
 ELSE
 OPCION_LDR=0
 END IF
END SELECT
INDICE=0
SENSOR=0
ORDEN=0
PIR1.RCIF=0
PIE1.RCIE=1
END IF
```

```
IF(ACTUADOR<>0)THEN
 SELECT CASE ACTUADOR
 CASE 1
  IF(ORDEN=1)THEN
   OPCION_MOTOR=1
   PIR1.TMR1IF=0
   INTCON.TMR0IF=0
   INTCON.INTF=0
   DISPARO=0
   PIE1=%00100001
   PIR1.RCIF=0
  ELSE
   OPCION_MOTOR=0
   PWM1_SET_DUTY(0)
  END IF
 CASE 2
  IF(ORDEN=1)THEN
    OPCION_LUZ=1
    OPTION_REG=%11000100
    INTCON=%11110000
    DISPARO=0
    INTENSIDAD=0
```

```
      ELSE
       OPCION_LUZ=0
       OPTION_REG=%10000100
       INTCON=%11000000
       PIE1=%00100001
       T1CON = %00000010
       PIR1=0
       PIR2=0
       DISPARO=0
      END IF
    END SELECT

     INDICE=0
     ACTUADOR=0
     ORDEN=0
     PIR1.RCIF=0
     PIE1.RCIE=1
   END IF
  END IF
 END SUB

MAIN:
```

OSCCON = 0X65

OPTION_REG=%10000100

ANSEL = %0000111 ' CONFIGURE AN0,AN1 ENTRADA ANALÓGICA

ANSELH = 0

TRISA=%00000111

TRISB=%00000001

TRISC=%10010001

PORTA=0X00

PORTB=0X00

PORTC=0X00

INTCON=%11000000 'TMER0 DISABLE , INT ESABLE , GIE SABLE PEIE

SABLE

PIR1=$00

PIR2=0

T1CON = %00000010 'BIT 2 PARA TMR1CS = 1 = DO NOT SYNCHRONIZE

EXTERNAL CLOCK ´

' INPUT / 0 = SYNCHRONIZE EXTERNAL CLOCK INPUT

PIE1=%00100001 'DISABLE TIMER1 OVERFLOW,ENABLE RX INT

TMR0=0

INTENSIDAD=0

TEMP_FINAL=0

TEMP_INICIAL=0

SENSOR=0

ACTUADOR=0

INDICE=0

DISPARO=0

DIR2=0

NDIR2=0

DUTY=0 'VELOCIDAD INICIAL 0

PULSOS=0

OPCION_ENCODER=0

OPCION_TEMP=0

OPCION_LDR=0

OPCION_MOTOR=0

OPCION_LUZ=0

TIEMPO=500

UART1_INIT(9600)

DELAY_MS(100)

PWM1_INIT(1000)

PWM1_START()

```
PWM1_SET_DUTY(0)
DELAY_MS(100)

WHILE(1)
 VALIDAR_ORDEN()
 IF(OPCION_MOTOR=1)THEN
  DIR2=1
  NDIR2=0
  PWM1_SET_DUTY(DUTY)
 END IF
 IF(OPCION_ENCODER=1)THEN
  PULSOS=0
  TMR1H = $FF        ' INITIALIZE TIMER1 REGISTER
  TMR1L = $FF        ' INITIALIZE TIMER1 REGISTER
  T1CON.TMR1CS=1
  PIR1.TMR1IF=0
  PIE1.TMR1IE=1
  T1CON.TMR1ON=1
  INTCON=%11000000
  PIR1.RCIF=0
  DISPARO=0
  DELAY_MS(1000)   'RETARDO DE 1 SEG CADA MUESTREO
```

```
  WORDTOSTRWITHZEROS(PULSOS*2,TXT)
  UART1_WRITE("P")
  IF(PULSOS > 99)THEN
   UART1_WRITE(TXT[2])
   UART1_WRITE(TXT[3])
   UART1_WRITE(TXT[4])
  ELSE
   UART1_WRITE(48)
   UART1_WRITE(TXT[3])
   UART1_WRITE(TXT[4])
  END IF
   PULSOS=0
END IF
IF(OPCION_TEMP=1)THEN
 OW_RESET(PORTC,4)
 OW_WRITE (PORTC,4,$CC)
 OW_WRITE (PORTC,4,$44)
 DELAY_US(500)
 OW_RESET (PORTC,4)
 OW_WRITE(PORTC,4,$CC)
 OW_WRITE (PORTC,4,$BE)
```

```
TEMP_LOW=OW_READ (PORTC,4)
TEMP_HIGH=OW_READ(PORTC,4)
MENOS= TEMP_HIGH >> 3
TEMP_TOTAL= (TEMP_HIGH << 8) OR TEMP_LOW
IF(MENOS=%000011111)THEN
  TEMP_TOTAL= NOT TEMP_TOTAL
  TEMP_TOTAL =TEMP_TOTAL + 1
END IF

TEMP_ENTERA  = (TEMP_TOTAL AND $0FF0) >> 4
TEMP_DECIMAL = (TEMP_LOW  AND $000F)*625

IF((TEMP_ENTERA>0))THEN
   TEMP_INICIAL=TEMP_ENTERA
ELSE
   TEMP_ENTERA=TEMP_INICIAL
END IF
 WORDTOSTR(TEMP_ENTERA,TXT)
 DELAY_MS(1000)   'RETARDO DE 1 SEG CADA MUESTREO
 UART1_WRITE("T")
 UART1_WRITE("0")
 UART1_WRITE(TXT[3])
```

```
    UART1_WRITE(TXT[4])
  END IF
  IF(OPCION_LDR=1)THEN
     LDR_ANALOG=ADC_READ(2)>>2
     LDR_ANALOG=255-LDR_ANALOG
     WORDTOSTRWITHZEROS(LDR_ANALOG,TXT)
     UART1_WRITE("L")
     IF(LDR_ANALOG > 99)THEN
      UART1_WRITE(TXT[2])
      UART1_WRITE(TXT[3])
      UART1_WRITE(TXT[4])
     ELSE
       UART1_WRITE(48)
       UART1_WRITE(TXT[3])
       UART1_WRITE(TXT[4])
     END IF
  END IF
     VDELAY_MS(0)
WEND
END.
```

BIBLIOGRAFÍA

[1] Ogata K., Sistemas de Control en Tiempo Discreto, Prentice Hall, 1996

[2] Maxim, DS18B20 Programmable Resolution 1-Wire Digital Thermometer, goo.gl/1E74p, fecha de consulta diciembre 2012.

[3] Santos M., Ingeniería de Sistemas y Automática Facultad de Informática, goo.gl/g12uF, fecha de consulta octubre 2012.

[4] National Semiconductor, LM340/LM78XX Series 3-Terminal Positive Regulators, goo.gl/xnDUz, fecha de consulta diciembre 2012.

[5] Microchip, PIC16F882/883/884/886/887 Data Sheet, goo.gl/I7CmX, fecha de consulta Enero 2013.

[6] Motorola Smiconductor, MOC70, goo.gl/hZ4yR, fecha de consulta enero 2013.

[7] Lennart Ljung, System Identification Toolbox For Use with MATLAB, The MathWorks, Inc., agosto de 1995.

[8] Idetec Cia. Ltda., Módulo de desarrollo MEI&T04, goo.gl/WVv1d, fecha de consulta Diciembre 2012.

[9] Idetec Cia. Ltda., Módulo de control de carga AC I&T, goo.gl/fuKgi, fecha de consulta Diciembre 2012.

[10] Idetec Cia. Ltda., Módulo disparador de relé I&T, goo.gl/w84Ge, fecha de consulta Diciembre 2012.