IMPLEMENTACION DE UN SISTEMA DE CONTROL DE TEMPERATURA EN TIEMPO REAL UTILIZANDO LA HERRAMIENTA *xPC-Target* DE *MATLAB*®

MANUEL LEONARDO OLEJUA ORTIZ
OSCAR JAVIER OSPINA VASQUEZ

UNIVERSIDAD PONTIFICIA BOLIVARIANA
ESCUELA DE INGENIERÍA Y ADMINISTRACIÓN
FACULTAD DE INGENIERÍA ELECTRÓNICA
BUCARAMANGA
2008

IMPLEMENTACION DE UN SISTEMADE CONTROL DE TEMPERATURA EN TIRMPO REAL UTILIZANDO LA HERRAMIENTA *xPC-Target* DE *MATLAB*®

MANUEL LEONARDO OLEJUA ORTIZ
OSCAR JAVIER OSPINA VASQUEZ

Trabajo de Grado para optar al título de Ingeniero Electrónico

Director
OMAR PINZÓN ARDILA
Ph D.

UNIVERSIDAD PONTIFICIA BOLIVARIANA
ESCUELA DE INGENIERÍA Y ADMINISTRACIÓN
FACULTAD DE INGENIERÍA ELECTRÓNICA
BUCARAMANGA
2008

No hay forma de poder pagar a mis padres todo lo que me han dado en la vida, ellos son los únicos responsables de que yo alcance cada logro y cada meta, por eso les dedico con mucho cariño este trabajo. A mi hermanita Lina Maria que siempre me apoya en la distancia. A mi hermano Andrés también va mi dedicatoria pues siempre me apoyaba y me aconsejaba para que las cosas salieran bien. A mi abuelita que rezó todo el tiempo para que las cosas me salieran bien y por ultimo quiero dedicarle también este trabajo a mi novia Jennifer Rojas porque siempre estuvo a mi lado cuando salían las cosas bien o cuando no. Te agradezco mucho por tenerme la paciencia que solo tú me puedes tener.

Oscar Javier Ospina Vásquez

A mis padres ya que a ellos debo todo lo que soy y quienes con su inmenso amor y ejemplo han hecho posible la realización de mis metas.

A mi tía Alicia y mis primos (Henry, Fabian, Irma, Paty, Mery Martin y Rene), por su apoyo incondicional, sus enseñanzas, colaboración y esmero fueron decisivos para culminar con éxito esta meta tan anhelada.

A todos mis amigos y compañero de proyecto gracias por todos esos momentos que compartimos juntos

Manuel Leonardo Olejua Ortiz

AGRADECIMIENTOS

En todos los procesos de aprendizaje tanto laboral como académico, se presentan personas llenas de aptitudes y actitudes, dispuestas a compartir esos dones que DIOS les ha regalado, sin esperar nada a cambio. Además de exigir esperando obtener profesionales competitivos.

Damos inmensas gracias a nuestro director de proyecto, el Dr. Omar Pinzón Ardila por su gran ayuda, su comprensión y su enorme espíritu colaborativo. A todos los profesores que tuvimos durante la carrera, de los cuales se tuvo la oportunidad de aprender no solo conceptos teóricos sino también enseñanzas para toda la vida.

CONTENIDO

LISTA DE FIGURAS

RESUMEN GENERAL DE TRABAJO DE GRADO

TITULO: IMPLEMENTACION DE UN SISTEMA DE CONTROL DE TEMPERATURA EN TIEMPO REAL UTILIZANDO LA HERRAMIENTA *xPC-Target* DE *MATLAB*

AUTORES: MANUEL LEONARDO OLEJUA ORTIZ
OSCAR JAVIER OSPINA VASQUEZ

FACULTAD: FACULTAD DE INGENIERIA ELECTRONICA

DIRECTOR: OMAR PINZON ARDILA

RESUMEN

En este proyecto se ha implementado un sistema de control sencillo donde el regulador se ejecuta en tiempo real utilizando la herramienta xPC-Target de Matlab. Adicionalmente, la plataforma soporta entradas y salidas de corriente de 4 a 20 mA que permiten controlar procesos de un lazo de control, tal como la temperatura, presión o el nivel de un líquido en un tanque. El xPC-Target es una herramienta que utiliza modelos diseñados en Simulink permitiendo interactuar y generar código para un sistema empotrado de tiempo real.

El prototipo del sistema de tiempo real adquiere una señal medida por el sensor (temperatura, presión, nivel, etc.) utilizando una tarjeta de adquisición de datos para procesarse en el sistema empotrado de tiempo real, el cual genera la acción de control utilizando una de las salidas de la tarjeta de adquisición de datos. Para validar los resultados se muestran las simulaciones y pruebas experimentales del funcionamiento del sistema.

PALABRAS CLAVES: Control de procesos, tiempo real, sistemas empotrados, xPC-Target, Simulink, Matlab.

RESUMEN GENERAL DE TRABAJO DE GRADO

TITULO: IMPLEMENTACION DE UN SISTEMA DE CONTROL DE TEMPERATURA EN TIEMPO REAL UTILIZANDO LA HERRAMIENTA *xPC-Target* DE *MATLAB*

AUTORES: MANUEL LEONARDO OLEJUA ORTIZ
OSCAR JAVIER OSPINA VASQUEZ

FACULTAD: FACULTAD DE INGENIERIA ELECTRONICA

DIRECTOR: OMAR PINZON ARDILA

RESUMEN

In this project we have implemented a simple system where the regulator is running in real time using the tool XPC-Target of Matlab. Additionally, the platform supports input and output current of 4 to 20mA to control a process control loop, such as temperature, pressure or level of a liquid in a tank. The xPC-Target is a tool that it uses models designed in Simulink allowing to interact and generate code for an embedded system in real time.

The prototype system acquires real-time signal measured by a sensor (temperature, pressure, level, etc.) Using a data acquisition board to be processed in real-time embedded system, which generates the control action using one of the output of the data acquisition board. To validate the results there appear the simulations and experimental tests of system performance.

PALABRAS CLAVES: Process control, real-time embedded systems, XPC-Target, Simulink, Matlab.

INTRODUCCIÓN

El control de procesos ha desempeñado un papel esencial en el desarrollo de la industria. Debido a su gran importancia en sistemas tales como vehículos espaciales, guiado de misiles y robots, el control automático se ha convertido en un componente importante e integral de los actuales procesos industriales y de manufactura. Por ejemplo, el control automático es esencial en el control numérico de las maquinas-herramienta en la industria de manufactura, en el diseño de sistemas de piloto automático, en la industria aeroespacial y en diseño de automóviles en la industria automotriz. También es fundamental en las operaciones industriales tales como el control de presión, temperatura, humedad, viscosidad y flujo.[1]

Actualmente, existen sistemas operativos de tiempo real (RTOS) que se usan en los sistemas empotrados para el control de procesos industriales, los robots industriales y en la industria aeronáutica. Los sistemas operativos de tiempo real se requieren en procesos críticos cuando las tareas se deben realizar en un tiempo definido, porque de lo contrario el funcionamiento errático de una maquina puede causar lesiones o muerte en un ser humano o daños en la infraestructura de la empresa.[2]

El matemático y científico de la computación Donald Gillies define un sistema de tiempo real como: “Un sistema donde las operaciones computacionales se ejecutan adecuadamente, si las operaciones lógicas se realizan correctamente en el tiempo que fue asignado para realizarlas. Si las restricciones de tiempo se violan se dice que el sistema ha fallado.” [2] El concepto de tiempo real no significa rapidez, es decir, un sistema no es de tiempo real porque se cumplen las tareas en tiempos muy cortos, sino porque es capaz de ejecutar las tareas programadas en el tiempo asignado.

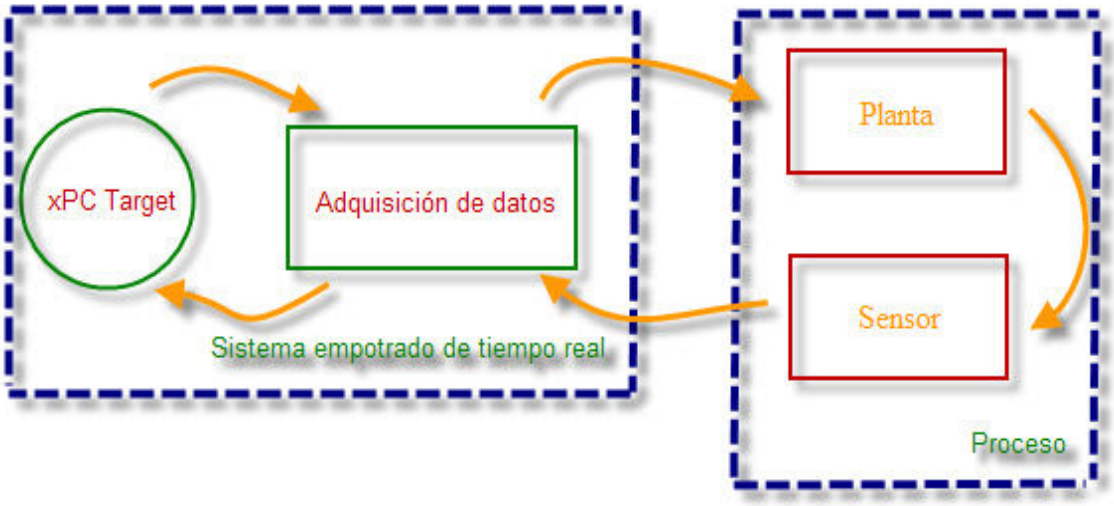
---

[1] OGATA, Katsuhiko. Ingeniería de control moderna 3ª edición. México: Prentice Hall, 1998. p. 1
[2] Furr, Steve. What is real time and why do I need it? .QNX Software Systems Ltd.

En este proyecto se plantea implementar un sistema de tiempo real usando la herramienta xPC-Target de *MATLAB*®.[3] Este sistema puede controlar un proceso sencillo, tal como la temperatura, la presión o el nivel de un tanque.

En la figura 1 se muestra el esquema del prototipo del sistema de tiempo real que se ha diseñado en este trabajo. Básicamente el sistema adquiere una señal medida por el sensor (temperatura, presión, nivel, etc.) utilizando una tarjeta de adquisición de datos. Esta señal se procesa en el sistema empotrado de tiempo real que a su vez genera la acción de control utilizando una de las salidas de la tarjeta de adquisición de datos.

Figura 1. Esquema del sistema de control en tiempo real.



Fuente: Los autores.

[3] MathWork, xPC Target Guetting Started. The MathWorks, Inc.Version 2. Octubre, 2004.

1. OBJETIVOS

1.1. OBJETIVO GENERAL

Implementar un sistema de control de temperatura en tiempo real utilizando la herramienta xPC-Target de *MATLAB*®.

1.2. OBJETIVOS ESPECÍFICOS

- Diseñar e implementar una modulación por ancho de pulso (PWM) en *Simulink*®, para controlar la potencia de la fuente de calor.

- Programar los drivers de la tarjeta de adquisición de datos para xPC-Target.

- Diseñar e implementar un sistema de control PI realimentado para controlar la temperatura de un bombillo.

- Diseñar e implementar un circuito transmisor y receptor de corriente, así como un acondicionador de señal para adquirir una temperatura y generar la señal de control utilizando *Simulink*®.

- Configurar la herramienta xPC-Target para ejecutar en tiempo real el modelo de control de temperatura creado en *Simulink*®.

- Diseñar los circuitos impresos de la planta, transmisor y receptor de corriente, acondicionador de señal y fuente de alimentación.

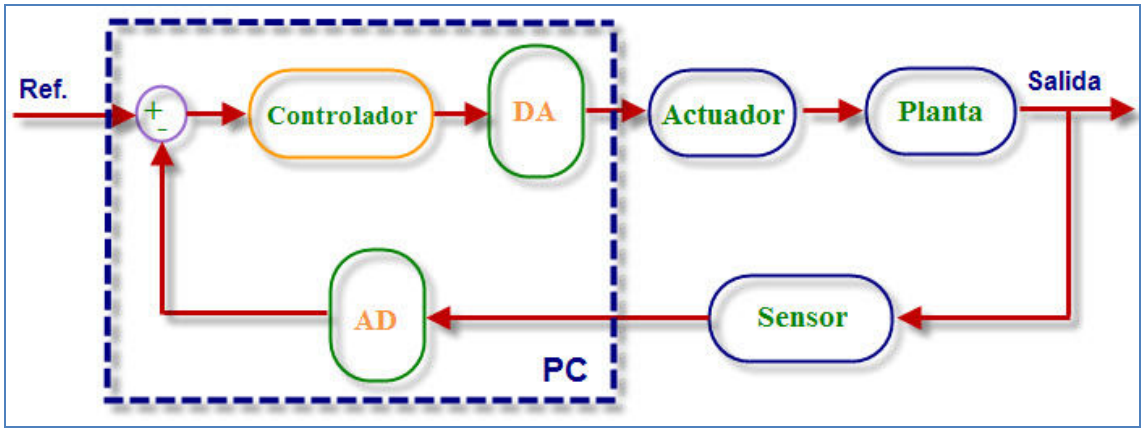
2. GENERALIDADES

2.1. SISTEMAS DE CONTROL

Un sistema de control es un dispositivo o un conjunto de dispositivos que dirigen o regulan el comportamiento de una variable. En este proyecto se diseña un control de temperatura mediante un computador.

En la figura 2 se muestra un diagrama de bloques de los componentes fundamentales de un sistema de control digital.

Figura 2. Sistema de control digital



Fuente. Astrom, Karl J, Computer-Controlled Systems Theory and Desing, Tercera Edicion, Editorial Prentice Hall, Pag 2.

- **Convertidor analógico-digital (A/D)**: Este dispositivo es el encargado de convertir una señal analógica en una señal digital. Este convertidor hace las veces de interfaz entre un componente analógico y otro digital. La conversión de analógico a digital es en

realidad una aproximación y este proceso de aproximación se conoce con el nombre de cuantificación.[4]

- **Convertidor digital- analógico (D/A)**: Algunas veces denominado decodificador y es el dispositivo encargado de convertir una señal digital en una señal analógica. [4]

- **Planta o proceso:** Una planta es cualquier elemento físico susceptible de controlarse. Se puede llamar proceso a una operación controlable. Algunos ejemplos son los procesos químicos, económicos, biológicos. [4]

- **Transductor o sensor:** Es un dispositivo que transforma una señal física de entrada en una señal de salida de naturaleza diferente, como por ejemplo una señal de temperatura en una señal de voltaje.[4]

2.1.1. Acciones básicas de control

Una acción de control es la forma como un controlador produce la señal de control. Los controladores industriales se clasifican dependiendo de la acción de control que realizan. A continuación se mencionan algunos de los principales tipos de controladores:

- De dos posiciones o de encendido y apagado (On/Off)
- Proporcionales
- Integrales
- Proporcionales-Integrales
- Proporcionales-Derivativos

[4] OGATA, Katsuhiko. Sistemas de control en tiempo discreto 2ª Edición. Prentice Hall, 1996. p. 7

- Proporcionales-Integrales-Derivativos
- Avanzados

En este proyecto se diseña un controlador Proporcional-Integral para controlar la temperatura de una fuente de calor (Bombilla o resistencia). Este controlador combina las características de un controlador proporcional y un controlador integral. El primero hace las veces de un amplificador con ganancia variable y la salida del controlador integral es la integral de la señal de error del sistema. La acción de control Proporcional-Integral se define mediante la siguiente ecuación:

$$u(t) = K_p e(t) + \frac{K_p}{T_i}\int_0^t e(t)dt \qquad (1)$$

Donde, K_p es la constante de proporcionalidad y T_i es el tiempo integral.

El propósito principal del controlador integral es disminuir el error en estado estacionario y mejorar la respuesta transitoria del sistema.[5]

2.2. SISTEMAS EMPOTRADOS

2.2.1. Definición de sistema empotrado

Un sistema empotrado o embebido (*embedded* system) es un sistema basado en microprocesador que se construye para controlar una función o un rango de funciones. as, al contrario de un computador de sobremesa o portátil. Es correcto afirmar que el usuario final puede escoger o seleccionar funciones del sistema pero

[5] OGATA, Katsuhiko. Ingeniería de control moderna 3ª edición. México: Prentice Hall, 1998. p.212-218

no puede modificar la funcionalidad del sistema agregando o sustituyendo el *software*.[6]

Si bien siempre se relaciona la palabra microprocesador con un computador de sobremesa o portátil, estos no son los únicos dispositivos donde se pueden encontrar. Muchos dispositivos electrónicos de uso domestico e industrial son sistemas empotrados que tienen un microprocesador que realiza una tarea previamente asignada.

Un automóvil puede tener cerca de 50 microprocesadores controlando diversas funciones, tales como, el sistema del motor, el antibloqueo electrónico de frenos, la caja de cambios controlada electrónicamente, el sistema de *airbag*, vidrios eléctricos, entre otros. Cada una de estas funciones se controla mediante un sistema empotrado. Una lavadora automática de ropa, los teléfonos celulares, algunos juguetes, los hornos microondas, son otros ejemplos de sistemas empotrados.

La teoría de control está muy ligada a los sistemas empotrados, ya que el objetivo principal de un sistema empotrado es controlar una o varias variables de un sistema físico, tales como la temperatura, el movimiento, el peso, la presión, entre otras. A pesar de las diferencias entre un sistema empotrado y un computador de propósito general, estos últimos han proporcionado una gran variedad de *hardware* y *software*, que son la base para diseñar los sistemas empotrados.[7]

---

[6] HEATH, Steve. Embedded Systems Design, Second Edition. Butterworth-Heinemann, 2003. p.1
[7] HEATH, Steve. Embedded Systems Design, Second Edition. Butterworth-Heinemann, 2003. p.1-7

2.2.2. Componentes de un sistema empotrado

Un sistema empotrado se compone de: un microprocesador, una placa base, una memoria, los periféricos, el software y los algoritmos. Estas partes se describen a continuación.

2.2.2.1. El microprocesador

El microprocesador es el elemento más importante de un sistema de cómputo quien se encarga de interpretar las instrucciones y procesar los datos contenidos en la ejecución del programa del computador.

Todos los sistemas de computadora tienen modelos definidos que indican la forma de seleccionar e interconectar los componentes de *hardware*. Esto se conoce como arquitectura de computadores.

Los microprocesadores también tienen sus propias arquitecturas. Algunas de las más importantes son: [8]

- Intel (4556, 4040, 8970, 8085, Zilog Z80, Itanium, I860, I515)
- X86 (Intel 8086-8088-80186-80188, Intel 80286, IA32, x86-84)
- MIPS (Microprocessor without Interlocked Pipeline Stages)
- Motorola (L6, 6809, C115, Corelduo 15485, 88000)
- IBM POWER (Power PC, G3, G4, G5)

[8] HEATH, Steve. Embedded Systems Design, Second Edition. Butterworth-Heinemann, 2003. p.8

- ARM y AVR

2.2.2.2. La placa base

La placa base es la tarjeta de circuitos impresos de un computador que sirve como medio de conexión entre el microprocesador, los circuitos electrónicos de soporte, las ranuras para conectar parte o toda la RAM del sistema, la ROM y las ranuras especiales (slots) que permiten la conexión de tarjetas adaptadoras adicionales.

Entre los diferentes tipos de placas base que se encuentran en el mercado se tiene: XT, AT, Baby-AT, ATX, EATX, Mini-ATX, Micro-ATX, ETX y PC104 (Utilizadas en sistemas empotrados)[8]

2.2.2.3. La memoria

La memoria es una parte importante de un sistema empotrado y la selección del tipo y tamaño de ésta, depende del software que se diseñe para el sistema empotrado. La memoria realiza esencialmente dos funciones en los sistemas empotrados:

- Provee almacenamiento para el software que se ejecuta y funciona como una memoria no-volatil que almacena el contenido cuando la alimentación esta desconectada. Ésta memoria puede ser una memoria de solo lectura (ROM) fabricada en un chip o puede ser una memoria programable externa del tipo

EPROM. El software que contiene puede ser el programa completo o una rutina de inicialización que obtiene el programa completo de otra parte del sistema.

- Provee almacenamiento para datos, como variables de programa, información de estado y cualquier otro dato que sea creado en el momento que el software se ejecute. La cantidad de memoria que se necesita para almacenar variables es frecuentemente menor que la requerida por el programa ejecutado. Como la memoria RAM es más costosa que la memoria ROM, muchos sistemas empotrados y en particular los microcontroladores, tienen pequeñas cantidades de RAM comparadas con las cantidades de memoria ROM que hay disponibles para el programa. Por lo tanto, el software que es escrito para los sistemas empotrados debe ser diseñado para minimizar el uso de memoria RAM.[9]

2.2.2.4. Los periféricos

Los periféricos son los dispositivos que le permiten al sistema empotrado comunicarse con el mundo exterior. Los periféricos de entrada están usualmente asociados con los sensores que miden las señales de las variables físicas, que son las que se procesan en el sistema empotrado y posteriormente la información procesada es generada por los periféricos de salida. En este proyecto la

[9] HEATH, Steve. Embedded Systems Design, Second Edition. Butterworth-Heinemann, 2003. p.8-9

tarjeta de adquisición de datos es tanto el periférico de entrada como el de salida. Entre los principales tipos de periféricos se encuentran:

- **Salidas binarias:** Son simplemente puertos en los cuales su estado lógico puede controlarse mediante el procesador, tomando dos únicos valores, cero lógico o uno lógico. Estos puertos también pueden ser configurados como entradas, usados para leer datos binarios que provienen de un acondicionador de señal o de un sensor.

- **Puerto Serie:** Es una interfaz que envía o recibe datos usando una o dos líneas de transmisión. Tienen una complejidad menor para conectar pero son mucho más complicados para programar.

- **Valores analógicos:** A diferencia del mundo real, los procesadores trabajan con datos en forma digital, por ello es necesario utilizar una interfaz entre el sistema y el mundo externo, las cuales permiten hacer conversiones de analógico a digital y viceversa.

- **Pantallas:** Las pantallas se han convertido en partes importantes de los sistemas empotrados y comúnmente son pantallas de siete segmentos o paneles alfa-numéricos de LCD. Las pantallas son usadas para interactuar con el usuario final,

permitiéndole visualizar los menús y mostrar señales o gráficos.[10]

2.2.2.5. *Software*

A menudo, el *software* que forma parte de un sistema empotrado, es la tecnología que define la forma del funcionamiento del sistema. El *software* puede estar compuesto por varios elementos, entre ellos se encuentran: [10]

- Inicialización y configuración.
- Sistema operativo.
- Las aplicaciones de usuario final.
- Gestión de errores.
- Soporte para la depuración y mantenimiento.

2.2.2.6. Algoritmos

Los algoritmos son una parte clave del *software* y son quienes establecen el comportamiento del sistema empotrado. Los algoritmos pueden manejar procesamiento matemático o modelos de variables físicas. [10]

En este proyecto los algoritmos son los que procesan las señales adquiridas y generan las señales de control dependiendo de los valores establecidos en el sistema.

[10] HEATH, Steve. Embedded Systems Design, Second Edition. Butterworth-Heinemann, 2003. p.9-10

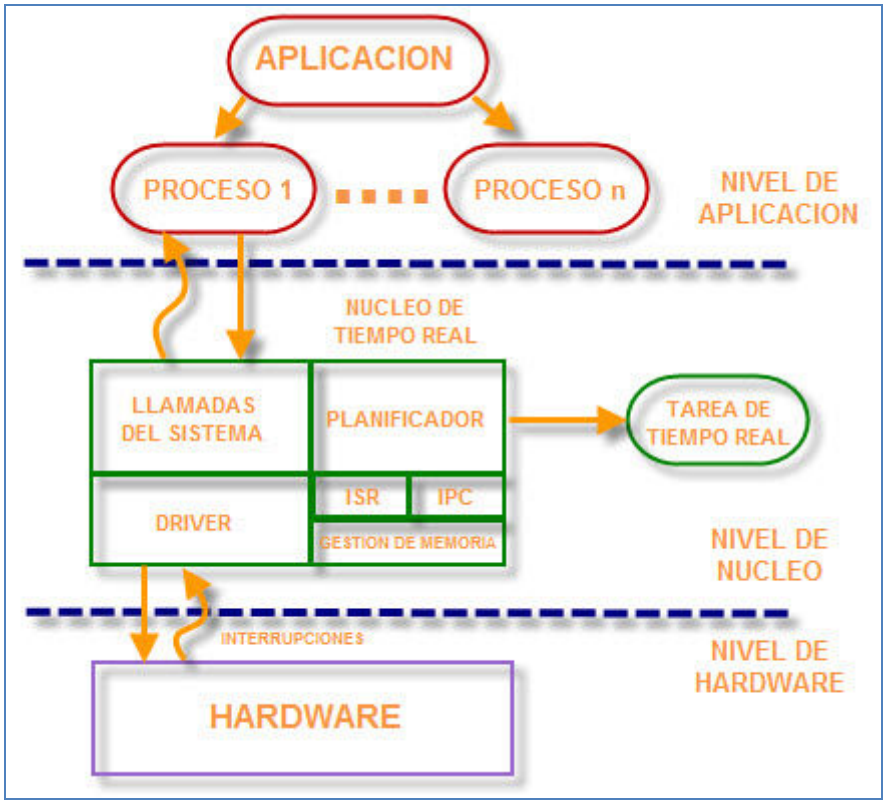
2.3. SISTEMAS DE TIEMPO REAL

Un sistema operativo es el componente de *software* que permite administrar, gestionar o coordinar la manera como se comparten los recursos en un equipo de cómputo. Un sistema operativo de tiempo real *RTOS* (de sus siglas en ingles *Real-Time Operating Systems)*, se encarga de administrar los recursos en un computador, pero se diferencia de los sistemas operativos de propósito general porque en ellos se respetan unas restricciones de tiempo previamente establecidas.

Para lograr que las tareas del sistema de tiempo real se ejecuten en el tiempo asignado, dicho sistema debe ser predecible, es decir, trabajar de forma determinista. Según el diccionario de la real academia Española, “El determinismo hace referencia a la teoría que supone que la evolución de los fenómenos naturales está completamente determinada por las condiciones iniciales”.

Los sistemas de tiempo real se fundamentan en los sistemas operativos de propósito general, por esta razón, tienen similitudes en sus estructuras y en su funcionamiento. En la figura 3 se muestra un esquema de un sistema operativo de tiempo real.

Figura 3. Esquema de un RTOS.



Fuente: Los autores

El *kernel* o núcleo es la parte más importante del sistema operativo, el cual se ocupa de ejecutar las diferentes funciones dentro del sistema operativo, entre las cuales están: planificar, comunicación entre tareas y procesos, atender las interrupciones, gestionar la memoria, administrar el *hardware* y atender las llamadas del sistema.

A continuación se explica de forma detallada las funciones que realiza el *kernel* en un sistema operativo de tiempo real.

2.3.1. Gestión y planificación de tareas computacionales

Una tarea computacional es un conjunto de instrucciones de programa que se ejecutan en un tiempo previamente asignado. Las tareas inician y terminan en tiempos específicos, el planificador de tareas se encarga de asignar los tiempos de ejecución o prioridades a cada una de ellas.

La planificación del ingles (*"Scheduling"*) de tareas o procesos es una función primordial en un sistema operativo. Las tareas deben ser creadas y eliminadas mientras el sistema se está ejecutando, además las tareas pueden cambiar sus niveles de prioridad, sus restricciones de tiempo, la cantidad de memoria requerida, etcétera.

La planificación de tareas en un sistema de tiempo real es más compleja que en un sistema operativo de propósito general. Esto implica que una tarea de tiempo real necesita obtener sin retardo la cantidad de memoria que requiere, adicionalmente, esa cantidad de memoria que se reserva debe ser bloqueada en la memoria principal y no puede ser usada por otra tarea, esto con el fin de evitar latencias debido al intercambio de datos.

El planificador de tareas gestiona los tiempos de ejecución de las tareas basándose en la prioridad de cada una de ellas. El cambio de las prioridades influye en el comportamiento de ejecución del sistema y por lo tanto también en la predicibilidad del mismo. Esta última es de suma importancia en los sistemas de tiempo real, por lo tanto, el buen funcionamiento de estos depende en gran parte de la buena planificación de las tareas.

Los sistemas operativos planifican las tareas basados en algoritmos de programación, los cuales asignan los diferentes recursos del sistema para las diferentes tareas que se ejecutan. Los sistemas operativos de tiempo real que utilizan los sistemas empotrados usan algoritmos simples y eficientes para planificar las tareas. Debido a que dichos sistemas requieren tiempos cortos de procesamiento, sus tareas se ejecutan de forma determinista, además usan pequeñas cantidades de memoria para su código.

Los sistemas de propósito general y los de tiempo real difieren considerablemente en sus algoritmos de planificación de tareas. Ambos tipos de sistemas operativos tienen los mismos principios básicos de funcionamiento, pero se aplican de una manera diferente en cada uno de ellos porque tienen que satisfacer diferentes criterios de rendimiento. Por ejemplo el objetivo de un PC (sistema de propósito general) es obtener el máximo rendimiento de la maquina, sin embargo, en un sistema empotrado de tiempo real, el objetivo es conseguir un comportamiento determinista, disminuir la cantidad de memoria y mantener un bajo consumo de potencia.

Para cumplir estas características especiales en los sistemas de tiempo real existe una gran variedad de algoritmos de programación utilizados para la planificación de tareas. Entre los algoritmos más comunes de planificación se encuentran los siguientes:

- **Planificador estático de prioridades (*Static Priority Scheduling):*** Es el algoritmo más sencillo debido a que asigna prioridades estáticas a cada tarea. La prioridad de cada tarea se asigna en el momento en que ésta se crea.

- **Prioridad a la tarea más urgente *EDF* (**de sus siglas en ingles **E*arliest Deadline First)*:** En este tipo de algoritmo, se asigna la prioridad más alta a la tarea cuyo tiempo de entrega esté más próxima a terminar. Es decir, si hay dos tareas que tienen un tiempo de entrega de 1 y 2 milisegundos respectivamente, el planificador le asigna a la tarea con el tiempo de entrega de 1 milisegundo una prioridad más alta. Para lograr esto, el planificador debe tener conocimiento del tiempo límite de entrega y la duración de cada tarea.

- ***Planificador de tasa mono-tónica (Rate-monotonic scheduling)***: Utilizando este algoritmo, el planificador asigna la prioridad más alta a la tarea que se ejecuta con más frecuencia. Este algoritmo es muy eficiente cuando todas las tareas son periódicas y el planificador debe conocer el periodo de cada tarea.[11]

Las tareas que se planifican necesitan comunicarse y sincronizarse entre sí, puesto que no pueden utilizar un recurso al mismo tiempo. Por esto otra de las funciones de los RTOS es comunicar y sincronizar eficientemente las tareas que se ejecutan.

2.3.2. Comunicación y sincronización de tareas computacionales

La comunicación entre procesos IPC (de sus siglas en ingles *Inter-Process Communication)* es la forma como el sistema operativo comunica y sincroniza las tareas entre sí. En términos generales, el IPC reúne un gran número de instrucciones de programación que el sistema operativo pone a disposición de las tareas que necesitan

[11] BRUYNINCKX, Herman. Real-Time and Embedded Guide. 2002. p. 1

intercambiar información con otras tareas, o para sincronizar sus acciones. Es importante destacar que un sistema operativo de tiempo real tiene que asegurarse que esta comunicación y sincronización se haga de una manera determinista.[12]

Además de planificar las tareas, sincronizarlas y comunicarlas entre sí, el sistema operativo de tiempo real debe compartir los recursos disponibles entre los diferentes dispositivos periféricos. Estos dispositivos de *hardware* solicitan un recurso del sistema mediante interrupciones. Una interrupción es la señal que recibe el procesador y que le indica que debe interrumpir su ejecución actual para atender un requerimiento de un componente de *hardware*.

2.3.3. Servicios de interrupciones

Como se describe anteriormente, un sistema operativo de tiempo real no sólo debe programar tareas de acuerdo a un algoritmo determinista, adicionalmente, debe controlar periféricos como: contadores, motores, sensores, dispositivos de comunicación, discos duros, pantallas, etc. Todos ellos pueden solicitar la atención del sistema operativo de forma asíncrona, por ejemplo: al mismo tiempo que un periférico utiliza un servicio del sistema operativo, este tiene que asegurar que está preparado para realizar las peticiones de servicios.

El procesador del sistema operativo puede atender dos clases de interrupciones. Estas son las interrupciones por *hardware* y las interrupciones por *software*. A continuación se explica cada una de ellas.

---

[12] BRUYNINCKX, Herman. Real-Time and Embedded Guide. 2002. p. 2-3

- **Interrupción por hardware:** Es el de tipo de interrupción en el cual un periférico le indica al procesador que requiere algún tipo de servicio, mediante una señal eléctrica (cambio de estado). Cuando el procesador recibe esta señal, detiene la ejecución de las tareas que estaba realizando y guarda el estado de las mismas con el fin de ejecutarlas después de atender la interrupción. Así mismo, el procesador deshabilita las interrupciones, ya que no se pueden ejecutar dos interrupciones al mismo tiempo. Posteriormente ejecuta la rutina de la interrupción, restaura las tareas a su estado previo y habilita nuevamente las interrupciones.

- **Interrupción de software:** Muchos procesadores tienen instrucciones de software programadas de fábrica, con las cuales se interrumpe la ejecución del programa en el procesador al igual que en una interrupción por *hardware*. La diferencia consiste en que la interrupción por software no le indica al procesador que requiere un servicio mediante una señal eléctrica, sino por medio de una instrucción. El uso de interrupciones por software hace que los programas sean más cortos, mejor escritos y de mejor desempeño.[13]

Entre los diferentes recursos que comparten los sistemas operativos, uno de los más importantes es la memoria. Para aprovechar al máximo la cantidad de memoria disponible en un sistema computacional es necesario llevar a cabo un proceso de gestión de memoria.

[13] BRUYNINCKX, Herman. Real-Time and Embedded Guide. 2002. p. 2

2.3.4. Gestión de memoria

El sistema operativo de tiempo real también es responsable de asegurar una gestión eficiente de las cantidades de memoria disponibles para ejecutar las tareas.

Las funciones que debe realizar un sistema operativo para la gestión de memoria son:

- Asignar a cada tarea la cantidad de memoria requerida (*memory allocation)*.

- "Mapear" la memoria real en los rangos de dirección usados en las diferentes tareas (*memory mapping*).

- Tomar la decisión adecuada cuando una tarea utiliza memoria no asignada (*memory protection)*.[13]

Como se ha explicado anteriormente los sistemas operativos llevan a cabo funciones importantes como gestión y planificación de tareas, procesos y recursos. En los sistemas operativos de tiempo real estas funciones se realizan bajo unas condiciones de tiempo especiales, debido a esto, es importante conocer algunos conceptos referentes a la gestión del tiempo en dichos sistemas, tales como: Tiempo real, Latencia, restricción de tiempo y estructuras de datos para el tiempo.

2.3.5. Tiempo real

Probablemente hay muchas definiciones que intenta explicar el significado de tiempo real, una de ellas es la que define tiempo real como una cualidad que tienen algunos sistemas operativos para poder ejecutar todas las tareas que han sido programadas, sin violar las restricciones especificas de tiempo. Por otro lado, otra definición que nos permite conocer la importancia del concepto de tiempo real es que el tiempo en el cual las tareas se realizan, puede ser estimado de forma determinista con base en el conocimiento previo del *hardware* y *software* del sistema. Esto significa que si el hardware puede hacer el trabajo, el sistema operativo de tiempo real lo hace de forma determinista.[14]

2.3.6. Latencia

La latencia de una tarea corresponde a la diferencia entre el instante de tiempo en el cual la tarea tendría que haber iniciado (o terminado) y el instante de tiempo en el cual lo hizo realmente. En otras palabras, es el tiempo entre la generación de un evento y la percepción del mismo.

Las latencias en los sistemas se deben a muchos factores, tales como:

- La frecuencia del procesador, del bus, de la memoria (*Cache, RAM, ROM*) y de los dispositivos periféricos.

- La programación de tareas del sistema operativo.

[14] BRUYNINCKX, Herman. Real-Time and Embedded Guide. 2002. p. 5-6

- Las características del *Kernel* del sistema.
- La carga computacional en el sistema, lo cual hace referencia al número de tareas que se programan consecutivamente.

- El tiempo que el procesador requiere para almacenar los datos de la tarea que está ejecutando en un momento determinado y sustituirlos con los datos de las nuevas tareas programadas.

Algunas de las funciones del *kernel* que introducen comportamientos no deterministas del tiempo de ejecución de los sistemas operativos de propósito general son:

- Acceso al disco duro: Debido a la alineación de los sectores, la distancia entre pistas, una tarea no sabe con seguridad cuanto tiempo le tomará en acceder a cierto dato que necesita. Además los discos duros son dispositivos mecánicos, en los cuales los tiempos de acceso a los datos son mucho más largos que en dispositivos netamente electrónicos (como las memorias RAM).

- Acceso a una red: Cuando se tiene una comunicación en red entre dos PCs, a menudo, los sistemas de detección y corrección de errores de dichas redes, reenvían paquetes en caso de detectar errores de transmisión. Esto genera una latencia en el intercambio de información.

- Otro retardo relacionado con el tiempo es el hecho que al programar el temporizador del chip del procesador, a menudo genera retardos impredecibles. Estos retardos son del orden

de microsegundos, por lo tanto solo importan si se tratan de sistemas donde la gestión del tiempo debe ser de alta precisión.

- Los "*Drivers"* de dispositivos que no fueron programados para su ejecución en tiempo real a menudo descuidan la administración del tiempo y en lugar de usar las interrupciones, hacen que el dispositivo se ponga en modo ocupado por periodos de tiempo impredecibles.

- Gestión y asignación de la memoria: El tiempo que se ocupa en la asignación de memoria cuando una tarea ha pedido una cantidad de ella es impredecible. Especialmente cuando la memoria está fragmentada y no se encuentra en bloques continuos.[15]

2.3.7. Restricciones de tiempo

Cada aplicación tiene diferentes restricciones de tiempo, las cuales, deben ser idealmente satisfechas por los sistemas operativos de tiempo real. Sin embargo aun no hay un algoritmo de programación de tareas perfecto que pueda garantizar el cumplimiento de las siguientes clases de restricciones:

- Plazo de entrega (*Deadline*): Cuando una tarea tiene que ejecutarse antes de un instante de tiempo dado. Por ejemplo, el dato obtenido de la adquisición de una señal proveniente de un sensor, debe estar disponible antes que otro periférico intente leer ese valor.

[15] BRUYNINCKX, Herman. Real-Time and Embedded Guide. 2002. p. 6-7

- Tiempo de ejecución cero (*Zero execution time*): La tarea debe realizarse en un periodo de tiempo igual a cero en el caso ideal. Un ejemplo lo explica la teoría de control digital cuando asume que al tomar una medida, calcular la acción de control y aplicar la salida del dispositivo periférico, debe hacerse instantáneamente.

- Calidad del servicio (*Quality of service – QoS*): La tarea debe obtener una cantidad fija de servicios por unidad de tiempo (En este caso servicio hace referencia a tiempo de procesamiento). Esto es importante para aplicaciones multimedia (cuando se transmite audio o video a los dispositivos de multimedia), o en servidores de red.
  El concepto de calidad de servicio también se puede explicar mediante el significado de estos parámetros: "s" segundos de servicio en cada trama de tiempo de "t" segundos. Una especificación de 5 micro-segundos por 20 micro-segundos se acerca mucho más a la calidad de servicio en tiempo real que una especificación de 5 segundos por 20 segundos.

  El mayor de los problemas en el diseño de sistemas de tiempo real es que el programador de tareas necesita tener completo conocimiento de cuánto tiempo tarda una tarea en completarse y cuando está preparada para ejecutarse. Esta información es prácticamente imposible de obtener, incluso si estuviera disponible, calcular un plan óptimo de programación de tareas es una operación de alta complejidad y por lo tanto gastaría bastante tiempo para llevarla a cabo.[16]

[16] BRUYNINCKX, Herman. Real-Time and Embedded Guide. 2002. p. 7-8

2.3.8. Estructuras de datos para el tiempo

La fracción de tiempo más pequeña usada en los sistemas operativos de propósito general es mayor que 1 ms. No porque los procesadores no sean lo suficientemente rápidos para hacer grandes cantidades de trabajos en esa parte de tiempo, sino porque las maquinas de 32 bits tienen solo 2^{32} fracciones de tiempo antes que sus contadores se rebosen. A 1000 ciclos por segundo, esto corresponde a menos de 50 días, lo cual es ciertamente insuficiente para los servidores y los sistemas empotrados. Linux utiliza fracciones de tiempo para la programación de tareas de 10 ms en la mayoría de los procesadores.

Las restricciones de tiempo de las tareas de tiempo real son normalmente expresadas con resoluciones mucho mayores que las de propósito general, es decir se expresan en micro-segundos en lugar de mili-segundos. Por lo tanto, las estructuras de datos en las cuales el tiempo se guarda deben adaptarse a estas velocidades más altas, con el fin de evitar desbordamientos en el procesador.[17]

Los anteriores conceptos de tiempo real son la base para desarrollar los sistemas de tiempo real que actualmente se encuentran disponibles comercialmente. Además, cada sistema de tiempo real se programa de acuerdo a ciertos estándares.

[17] BRUYNINCKX, Herman. Real-Time and Embedded Guide. 2002. p. 8-9

2.3.9. RTOS estándares

Como en cualquier otro software, en los sistemas de tiempo real, la disponibilidad de las normas facilita enormemente el trabajo de los programadores ya que hace que sea más fácil, más económico y más rápido desarrollar nuevas aplicaciones.

A continuación se mencionan algunos sistemas operativos en los que se han hecho esfuerzos por estandarizar el mundo de los sistemas operativos de tiempo real.

- **POSIX (*Portable Operating System Interface*):**

  El estándar POSIX define un conjunto de llamadas al sistema (interrupciones por *software*) que debe proporcionar todo sistema UNIX compatible con este estándar[18]. En los últimos años este estándar se ha convertido en una parte muy importante para el desarrollo de los sistemas operativos incluyendo los RTOS. La importancia radica en que actualmente es difícil encontrar *software* comercial que no esté basado en el estándar POSIX. POSIX está dividido en 4 grupos: POSIX1, POSIX2, POSIX3 Y POSIX4.

  POSIX4 (también llamado POSIX-RT) es el estándar que explica los requerimientos que debe tener un sistema de tiempo real para cumplir con el estándar POSIX. Los principales requerimientos son:

[18] TANENBAUM, Andrew s. Sistemas operativos modernos. Segunda edición, 2003. Prentice Hall. p.44

Planificador de ejecución *(Execution Scheduling)*: Un RTOS para ser compatible con POSIX-RT debe tener soporte para prioridades estáticas de tiempo real.

Requerimientos de rendimiento en las llamadas al sistema *(Performance Requirements on system calls)*: Esto especifica los tiempos mínimos de ejecución requeridos en la mayoría de servicios de tiempo real.

Niveles de prioridad *(Priority Levels)*: Un sistema de tiempo real debe tener al menos 32 niveles de prioridad.

Temporizadores *(Timers)*: Un RTOS debe soportar temporizadores periódicos y los temporizadores de tipo *watch dog timer*. El reloj del sistema es llamado Reloj de Tiempo Real cuando el sistema soporta POSIX-RT.

Archivos de tiempo real *(Real-Time Files)*: Un sistema de archivos de tiempo real debe estar implementado. Un archivo de tiempo real es aquel que se guarda en bloques continuos en un disco duro, con el fin de tener retardos predecibles a la hora de acceder al archivo en sistemas.

Bloqueo de memoria *(Memory locking)*: POSIX-RT define los servicios mlockall(), mlock(), mlockpage() para el bloqueo de memoria y los servicios munlockall(), munlock(), y munlockpage() para desbloquear la memoria. Los servicios de bloqueo y

desbloqueo de la memoria se usan para lograr un acceso a la memoria de forma determinista.

Soporte multi-hilo *(Multithreading support)*: El soporte de hilos de ejecución de tiempo real es necesario. Un hilo de ejecución es una característica que permite a una aplicación realizar varias tareas concurrentemente. Los distintos hilos de ejecución comparten una serie de recursos tales como el espacio de memoria, los archivos abiertos, situación de autenticación, etc. Muchas de las extensiones en tiempo real de POSIX ya se han aplicado en RTLinux y RTAI. [19]

- **Unix98:** Es el estándar para los sistemas operativos UNIX manejado por "*The Open Group*". Este estándar incorpora muchos de los estándares de POSIX.

- **EL/IX:** EL/IX es una API para sistemas empotrados que busca seguir los estándares POSIX y ANSI C.

- **µITRON:** Es un estándar japonés para sistemas empotrados. "TRON" hace referencia al nucleo del sistema operativo en tiempo real (*The Real-Time Operating System Nucleus*), la letra "I" significa Industrial y la letra µ hace referencia a micro.

- **OSEK:** Es un estándar alemán para una arquitectura abierta de unidades de control de vehículos. La arquitectura es abierta pero no hay software gratuito disponible para ella.

[19] Embedded System from ECE Version II Kharagpur, India.

- **ADA 95:** El sistema operativo MaRTE (*Minimal Real-Time Operating System for Embedded Applications)* es un ejemplo de un *kernel* de tiempo real gratuito usado en aplicaciones de sistemas empotrados, que sigue las especificaciones del estándar de tiempo real POSIX13. La mayor parte de su código está escrito en ADA con algunas partes de C y lenguaje ensamblador.[20]

2.4. MODULACIÓN DE ANCHO DE PULSO (PWM)

En este trabajo se utiliza una modulación de ancho de pulso (PWM) para controlar la potencia entregada a la fuente de calor del sistema de control de temperatura. La modulación de ancho de pulso PWM (*Pulse-Width Modulation*) es un tipo de modulación donde la señal moduladora modifica el ciclo de trabajo de una señal periódica (que puede ser sinusoidal o cuadrada).

El ciclo de trabajo de una señal periódica es la relación que hay entre el tiempo en que la señal tiene un valor positivo y el periodo total de la señal. Es decir:

$$D = (\tau / T) \qquad (2)$$

Donde: D es el ciclo de trabajo, τ es el tiempo en que la señal es positiva y T es el periodo de la señal.

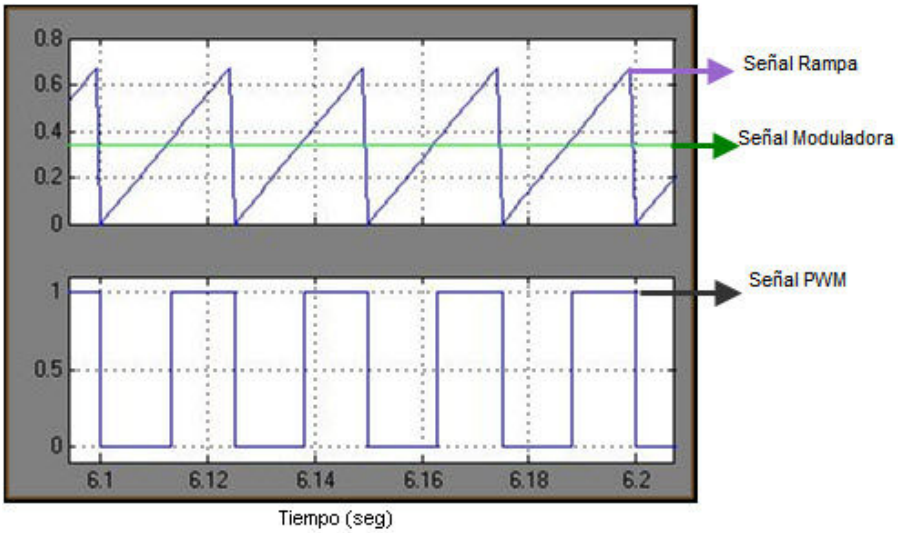
Hay tres partes esenciales en una modulación por ancho de pulso:

[20] BRUYNINCKX, Herman. Real-Time and Embedded Guide. 2002. p. 11-14

1. Un generador de ondas en rampa, que generalmente tiene una frecuencia constante.
2. Un comparador, para detectar cuando el voltaje de rampa ha superado el voltaje de la señal de control.
3. Un dispositivo electrónico que conecta la corriente a la carga, en el momento en que el comparador detecta el punto crítico en la onda rampa.[21]

Una señal PWM se construye comparando una señal tipo rampa con otra señal de referencia (moduladora). En la figura 4 muestra una señal PWM.

Figura 4. Señal PWM.



Fuente. Los autores.

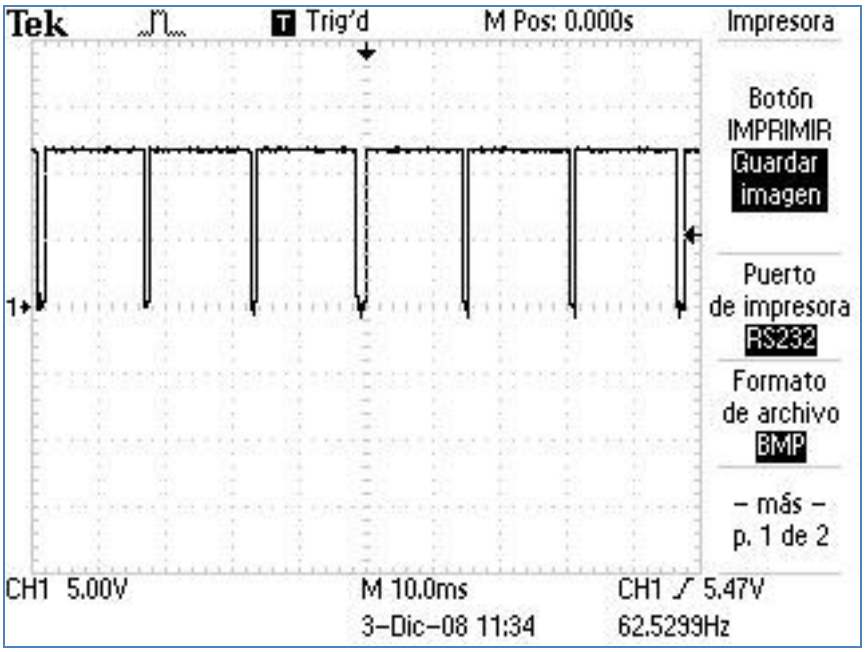
Las dos señales mostradas en la parte superior de la figura (señal rampa y señal de referencia) se restan y se hace una comparación mayor o igual que cero. Si el resultado de esta comparación es igual o mayor que cero el valor en la salida del comparador es 1 lógico. De lo contrario el valor en la salida del comparador es cero.

[21] MALONEY, Timothy. Electrónica Industrial Moderna, Quinta Edición. Prentice Hall, 2006. p.753-755

El ciclo de trabajo de la señal PWM varía de acuerdo al valor de la señal de referencia, como se observa en la figura cuanto más se acerque el valor de la señal de referencia al valor máximo de la señal diente sierra, es menor el tiempo en que la salida del comparador se mantendrá en 1. Por lo tanto el ciclo de trabajo disminuye.

En este proyecto se genera una onda PWM por una salida digital de la tarjeta de adquisición de datos. En la figura 5 se muestra una señal PWM generada por la tarjeta de adquisición de datos.

Figura 5. Señal PWM mostrada en el osciloscopio.



Fuente: Los autores.

3. DESCRIPCION DE LA PLATAFORMA EXPERIMENTAL

Para el proyecto de control de procesos en tiempo real, se diseña un prototipo conformado por componentes de software y hardware.

3.1 COMPONENTES DE *HARDWARE*.

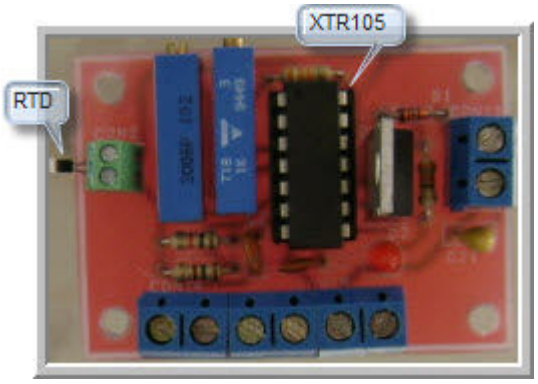
Los componentes de *hardware* del prototipo son: el sensor, la tarjeta de adquisición de datos, la planta, el actuador, el acondicionador de señal, el transmisor y el receptor de corriente y los equipos *Host* y *Target*. Estos componentes se describen a continuación.

- **Sensor:** Es el encargado de tomar la variable física y transformarla en una señal eléctrica. Esta señal eléctrica se transmite desde el transmisor de corriente hacia el receptor de corriente. El sensor que se utiliza en este proyecto para el control de temperatura es una RTD (de sus siglas en ingles *Resistance Temperature Detector*). Una RTD es un detector de temperatura que varía su resistencia a medida que varía la temperatura. Para mayor información sobre la RTD y sus características, refiérase al ANEXO A.

- **Transmisor de corriente:** Es el dispositivo encargado de tomar la señal del sensor y convertirla en una señal de 4-20mA. Esta señal se transmite mediante un cable de par trenzado hacia un receptor de corriente. En este proyecto el dispositivo transmisor de corriente se diseña con el circuito integrado XTR105 (ver figura 7). El XTR105 es un transmisor que puede generar una señal de corriente de 4-20mA a partir de la señal de una RTD o de un sensor de presión. En el ANEXO B se encuentra información sobre transmisores de corriente y sobre el

XTR105. En la figura 6 se muestra el transmisor de temperatura XTR105.

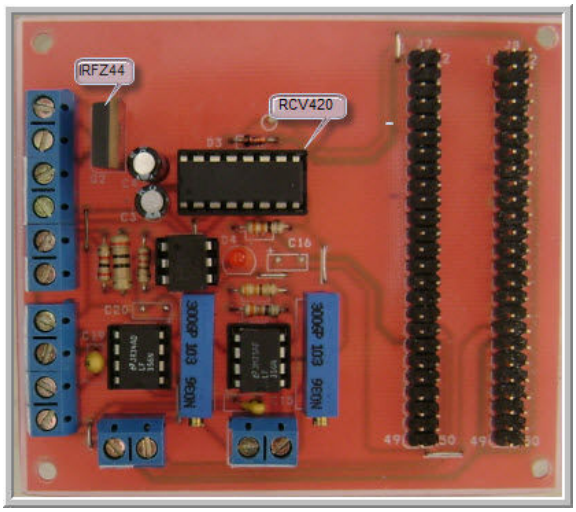
Figura 6. Transmisor de corriente XTR105.



Fuente. Los autores.

- **Receptor de corriente:** Toma la señal de corriente proveniente del transmisor y la convierte en una señal de voltaje. Esta señal de voltaje entra a la tarjeta de adquisición de datos, para procesarse en el equipo *target*. Entre los receptores de corriente comerciales se ha seleccionado el RCV420, el cual se muestra en la figura 7. El RCV420 es un receptor de precisión de corriente que convierte la señal de 4 a 20mA en una señal de salida de 0 a 5 voltios. Ver ANEXO B donde se profundiza en las características del RCV420.

Figura 7. MOSFET y Receptor de corriente.



Fuente. Los autores.

- **Tarjeta de adquisición de datos:** La tarjeta de adquisición de datos captura la señal a controlar, la acondiciona y la convierte a datos en forma digital. Estos datos digitales se procesan en un computador usando *MATLAB*® y Simulink®. Posteriormente se ejecutan en tiempo real usando la herramienta xPC-Target. Entre las tarjetas que se encuentran comercialmente se decidió utilizar la tarjeta PCI-DAS6036 de la empresa *Measurement Computing.* Para más información acerca de las características de la tarjeta de adquisición de datos referirse al ANEXO C.

- **PC *host:*** El equipo *host* es el encargado de descargar la aplicación de tiempo real en el equipo *target,* además, permite visualizar señales y modificar parámetros de la aplicación en tiempo real. Para más información sobre el equipo host, ver el ANEXO D.

- **PC *target:*** El equipo *target* es el cerebro de todo el sistema de control en tiempo real. Es un sistema empotrado que se diseña específicamente para controlar la temperatura de una fuente de calor (bombillo), pero haciendo unos pequeños cambios el sistema se puede

adecuar para controlar cualquier otra planta. En el equipo *target* se llevan a cabo las tareas de procesamiento en tiempo real, mediante xPC-Target. Por medio de la tarjeta de adquisición de datos, se adquieren las señales que ingresan por las entradas analógicas y se generan otras señales por las salidas analógicas y digitales. Para profundizar sobre el funcionamiento del equipo target y como configurar xPC-Target refiérase al ANEXO D.

- **Planta:** En este proyecto la planta es básicamente un transistor MOSFET IRF-z44, el cual es el encargado de dar corriente a la carga, dependiendo de la señal de control PWM generada en el equipo *target*. La carga en este caso es un bombillo que funciona como fuente de calor que mide el sensor para completar el lazo de control. Ver ANEXO E (Etapa de potencia).

- **Acondicionador de señal:** Se encarga de convertir la señal de voltaje que se genera por la salida analógica de la tarjeta de adquisición de datos, en una señal de 4-20mA. Esto con el fin de tener una señal de control estándar permitiendo alimentar cualquier tipo de actuador industrial que funcione con este tipo de señal. Ver ANEXO F (Acondicionadores de señal).

- **Actuador:** Realiza la acción de control generada por el sistema empotrado de tiempo real. Algunos ejemplos de actuador son: las resistencias calefactoras, los motores, las válvulas, las bombas, los ventiladores, etc.

3.2 COMPONENTES DE *SOFTWARE*.

Los programas de *software* que se usan en este proyecto son: *Matlab*, *xPC-Target* y *Simulink.*

- **Matlab:** *MATLAB*® es un software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Sus prestaciones básicas son: manipulación de matrices, representación de datos y funciones, implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. El paquete *MATLAB*® posee una herramienta adicional que expande sus prestaciones, *Simulink*®. Además, se pueden ampliar las capacidades de *MATLAB*® con las *cajas de herramientas* (*toolboxes*); y las de *Simulink*® con los *paquetes de bloques* (*blocksets*).

  Para el desarrollo del proyecto es necesario instalar la *toolbox* de *MATLAB*® xPC-Target la cual se describe más adelante.

- **Simulink:** *Simulink*® es un *software* con el que se puede hacer modelos, simular y analizar sistemas dinámicos. Con *Simulink*®, es posible construir fácilmente modelos empezando desde cero, o modificar modelos existentes para que se ajusten a una necesidad específica. Los sistemas también pueden ser *multirate*, es decir, que pueden tener diferentes partes que son muestreadas o actualizadas a diferentes tiempos de muestreo.

  En *Simulink*® se construye un modelo de bloques el cual consta de entradas y salidas análogas y digitales de la tarjeta de adquisición de datos, que interactúan con otros bloques (control PID, generador de señal, constantes, scopes, etc) para lograr realizar el control de

temperatura. El modelo en *Simulink*® se compila y se envía al equipo *target* mediante la herramienta xPC-Target. Ver anexo G (Simulink).

- **xPC-Target:** xPC-Target es una herramienta de la compañía *Mathworks*, que se usa para el diseño de prototipos, prueba de sistemas y desarrollo de sistemas en tiempo real, usando un computador estándar. El xPC-Target es adecuado para la simulación o el control en tiempo real de procesos, ya que permite al PC comunicarse con el exterior usando tarjetas de adquisición de datos.

  xPC-Target utiliza el código creado en *Simulink*® para crear la aplicación en tiempo real para lograr el control del prototipo. Ver anexo D.

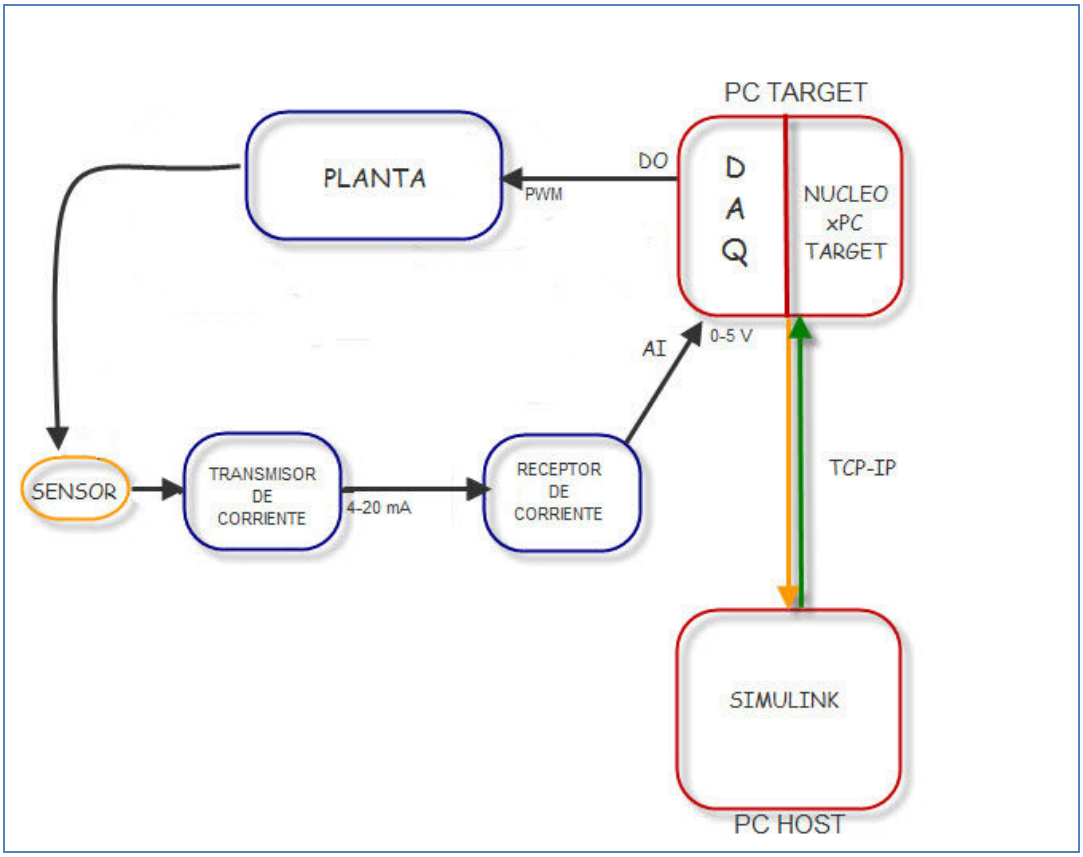
3.3 MODOS DE FUNCIONAMIENTO DEL PROTOTIPO.

El proyecto de control de procesos en tiempo real tiene dos modos de funcionamiento, uno de ellos controla la planta que se diseñó para este trabajo y el otro modo controla una planta externa.

3.3.1. Modo 1 – Control de una bombilla mediante PWM.

El primer modo genera una señal de control mediante PWM por una salida digital de la tarjeta de adquisición de datos. En la figura 8 se muestra un diagrama de este primer modo.

Figura 8. Control mediante PWM.



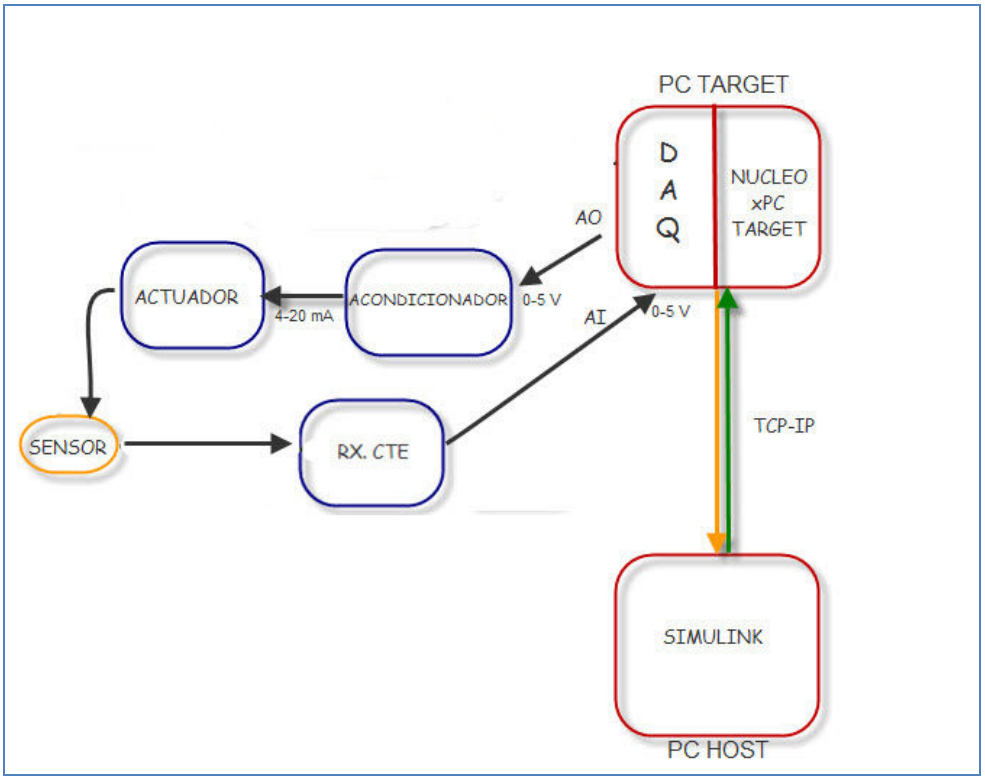
Fuente: Los autores.

El funcionamiento de este primer modo de control de procesos es el siguiente: Se transmite la señal del sensor de temperatura (RTD) con el transmisor de corriente XTR105 hacia el receptor de corriente RCV420. La señal que llega al receptor se adquiere por la entrada analógica (AI) de la tarjeta de adquisición de datos (DAQ), posteriormente esta se procesa en tiempo real en el equipo *target*. En la salida digital (DO), se genera la señal de control (PWM) la cual se aplica a la etapa de potencia, en este caso el MOSFET IRF-Z44.El MOSFET suministra la corriente a la bombilla.

3.3.2 Modo 2 – Control mediante salida y entrada de corriente de 4-20mA.

En el segundo modo se genera una señal de corriente (4-20mA), la cual se usa para manipular cualquier actuador que funcione con esta señal estándar de corriente. En la figura 9 se muestra un esquema del modo de funcionamiento mediante corriente 4-20mA.

Figura 9. Control mediante corriente 4-20mA.



Fuente: Los autores.

La señal de la planta externa (señal de corriente 4-20mA) se transmite al receptor de corriente RCV420, el cual a su vez la convierte en una señal de voltaje de 0-5V. Esta señal se adquiere por la entrada analógica (AI) de la tarjeta de adquisición de datos y posteriormente se procesa en el sistema empotrado de

tiempo real. Posteriormente, se genera la acción de control por medio de la salida analógica (AO) de la tarjeta de adquisición de datos. Esta señal de 0-5V se lleva a un acondicionador de señal que la convierte en una señal de corriente de 4-20mA, la cual es la señal que se aplica al actuador de la planta externa.

4. IDENTIFICACION DEL MODELO MATEMATICO DE LA PLANTA Y DISEÑO DEL CONTROLADOR PI

En este apartado se obtiene el modelo matemático de la planta. La planta se compone de un transistor de potencia que alimenta una bombilla mediante una modulación de ancho de pulso. El modelo matemático de la planta se representa mediante una función de transferencia y con la función de transferencia de la planta se puede diseñar un controlador PI. En este trabajo se busca alcanzar dos objetivos de diseño que son: un sobrepaso (*overshoot*) menor al 15% y un tiempo de establecimiento del sistema menor al tiempo de establecimiento la planta en lazo abierto.

4.1. MODELO MATEMÁTICO DE LA PLANTA.

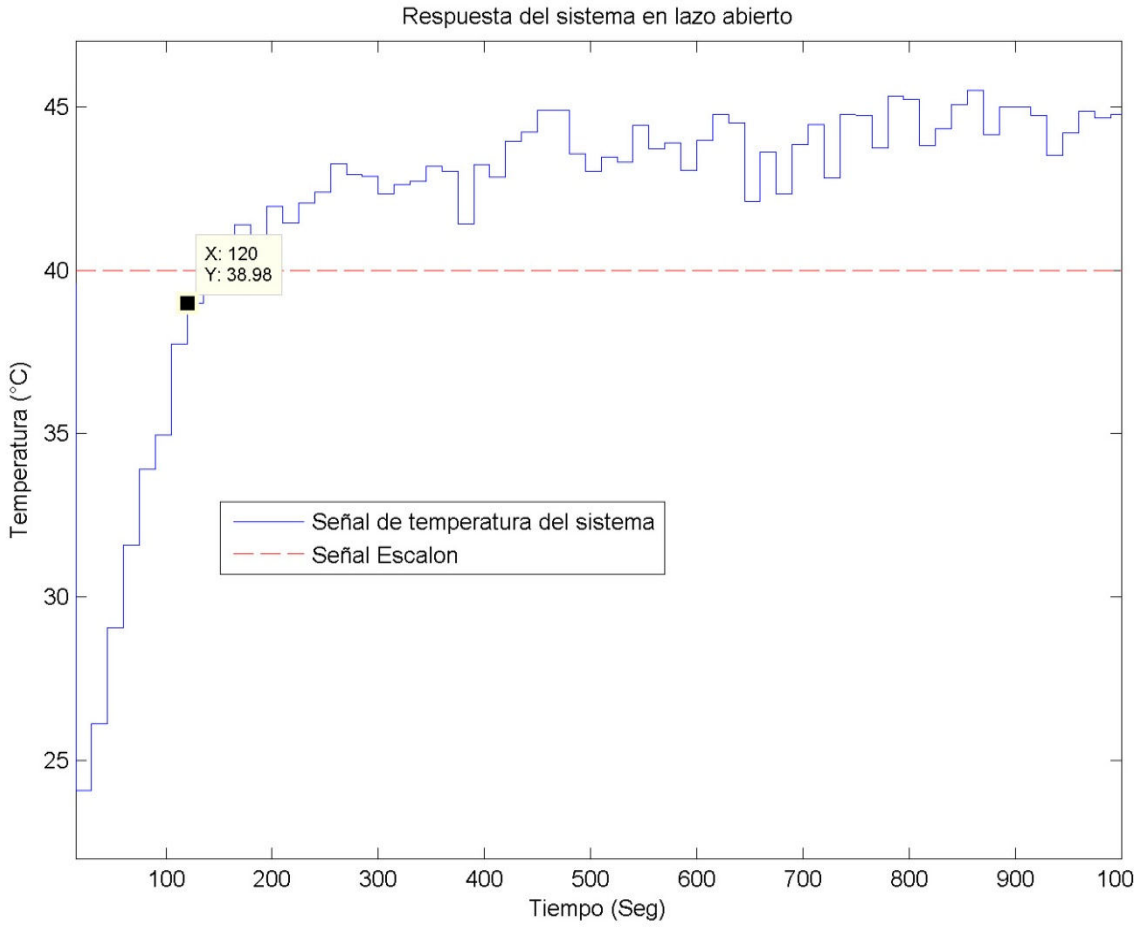
Por simplicidad del procedimiento la planta se puede aproximar a una función de transferencia de primer orden:

$$\frac{\Delta T_{(s)}}{U_{(s)}} = \frac{K_o}{\tau S + 1} \qquad (3)$$

Donde ΔT es el incremento de temperatura cuando se aplica una modulación de ancho de pulso, U es el porcentaje de la modulación de ancho de pulso aplicado a la bombilla, K_o es la ganancia estática del sistema en lazo abierto y τ es la constante de tiempo que representa la dinámica de la planta.

Para obtener las constantes τ y K_o de (3) se analiza la respuesta de la planta ante una entrada de tipo escalón. En la figura 10, se muestra la respuesta de la planta en lazo abierto ante una entrada escalón de 40°C.[22] Fundamentándose en la teoría clásica de control se puede determinar la constante de tiempo τ como el tiempo requerido para alcanzar el 63.2% de la temperatura final.

Figura 10. Respuesta del sistema en lazo abierto.



Fuente. Los autores.

[22] OGATA, Katsuhiko. Sistemas de control en tiempo discreto 2ª Edición. Prentice Hall, 1996. Pags. 136-137.

En la figura 10, también se muestra que la temperatura final se establece aproximadamente en 45°C y asumiendo que la temperatura inicial es la temperatura ambiente es aproximadamente de 22°C, la constante de tiempo obtenida es de 129.95s.

Por otro lado, la ganancia estática de la planta en lazo abierto se puede calcular utilizando la expresión en (4). [23]

$$K_p = \lim_{s \to 0} \frac{\Delta T_{(s)}}{U_{(s)}} \qquad (4)$$

De (4) y la figura 10 se tiene que $K_p = 28.75°C$.

Teniendo τ y K_p, la función de transferencia resultante de la planta es:

$$\frac{\Delta T_{(s)}}{U_{(s)}} = \frac{28.75}{129.95S+1} \qquad (5)$$

El controlador PI se diseña a partir de la ecuación 5 sujeto a las especificaciones de diseño mencionados anteriormente.

Es importante aclarar que la función de transferencia de la planta es una aproximación del sistema real y no se han contemplado dinámicas de mayor orden (capacitancia térmica, flujo de calor en estado estable y resistencia térmica entre otras). Por lo tanto aumentar el orden y complejidad del diseño, aumentaría el orden de la función de transferencia de la planta.

[23] OGATA, Katsuhiko. Sistemas de control en tiempo discreto 2ª Edición. Prentice Hall, 1996. Pags. 136-137.

4.2. DISEÑO DEL CONTROLADOR PI

En este proyecto se utiliza un controlador proporcional e integral en tiempo discreto cuya función de transferencia es la siguiente:

$$C_p(z) = K_p + \frac{Ki*Ts}{z-1} \qquad (6)$$

Siendo K_p la ganancia proporcional y K_i es la ganancia integral.

Para calcular las constantes de este controlador se utiliza la herramienta *sisotool* de *MATLAB*®. Esta herramienta permite diseñar compensadores de manera sencilla a través de su interfaz grafica de usuario. Con la herramienta *sisotool* se pueden ubicar de forma conveniente los polos y ceros en lazo cerrado, de tal manera que el sistema cumpla con los requerimientos de sobrepaso y tiempo de establecimiento.

Para calcular las constantes del controlador es necesario discretizar la planta. En este proyecto la planta se ha discretizado usando la función *c2d* de MATLAB utilizando un tiempo de muestreo de 5s. La función obtenida de la planta en tiempo discreto es:

$$G_p(z) = \frac{1.085}{z-0.9623} \qquad (7)$$

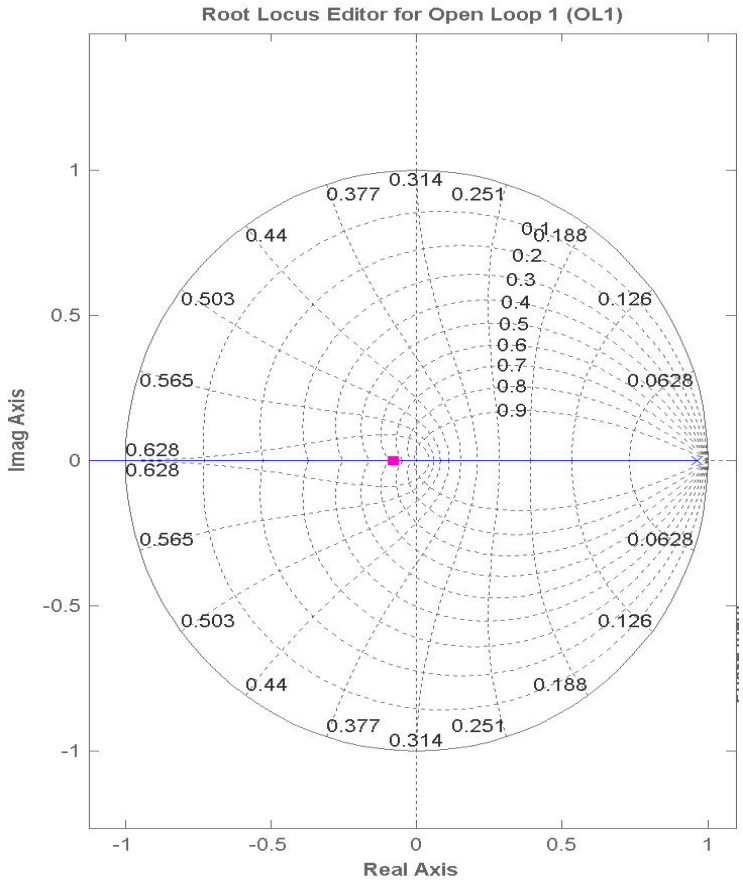
En la figura 11 muestra la ubicación de los polos y ceros en lazo abierto utilizando la herramienta *sisotool*.

En la figura 11, también se muestra un polo en el eje real que se localiza en 0.9623 rad/s, que corresponde al polo de la función de transferencia en tiempo discreto de la planta.

Al introducir un controlador PI se agrega un polo y un cero al sistema en lazo abierto y se localizan en los puntos de tal forma que el sistema cumpla con los requerimientos de diseño establecidos.

$$G_c(z)=K_c\ \frac{z+z_1}{z+p_1} \qquad (8)$$

Figura 11. Ubicación de los polos y ceros de la planta en lazo abierto.



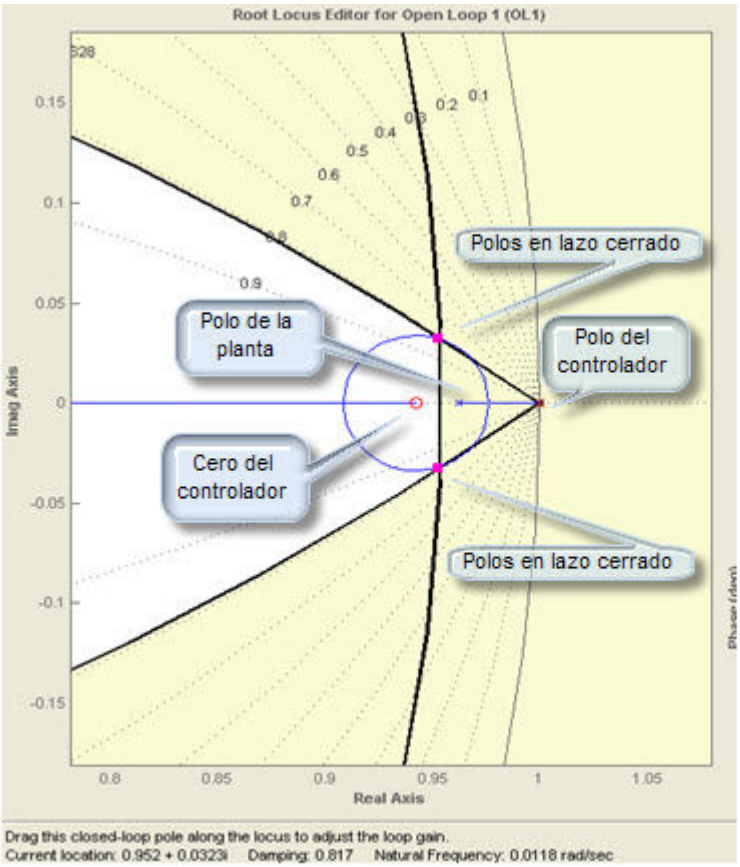
Fuente. Los autores.

La función de transferencia que resulta del sistema en lazo abierto es:

$$G_p(z) * G_c(z) = k_c * \frac{(z-z_1)*1.085}{(z+p_1)(z-0.9623)} \qquad (9)$$

En la figura 12 se muestra la ubicación final de los polos en lazo cerrado del controlador PI.

Figura 12. Ubicación de los polos y ceros del sistema.



Fuente. Los autores.

En la figura 12 también se muestra que para una ganancia de K_c de 0.0030847 los polos de lazo cerrado se localizan en 0.952+/-0.0332i rad/s. Con los valores obtenidos de K_c, z_1 y p_1 se obtienen las constantes K_p y K_i igualando (6) y (8).

$$K_p + \frac{Ki*Ts}{z-1} = K_c \frac{z+z_1}{z+p_1} \qquad (9)$$

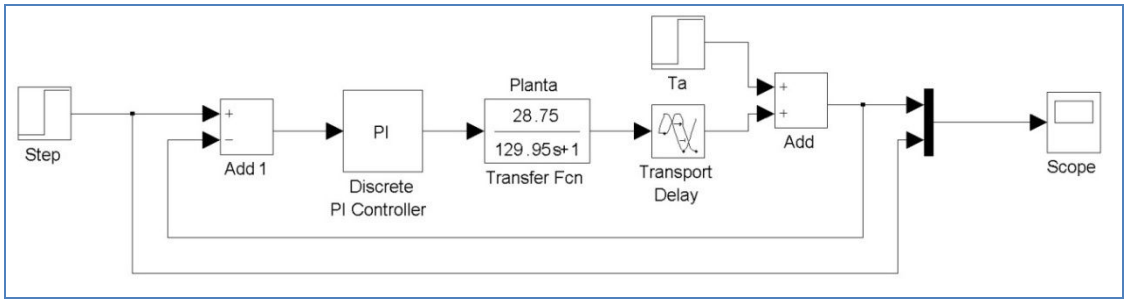
Las constantes obtenidas son K_p = 0.1405132 y K_i = 1.5788e-3, donde K_p y K_i son las constantes que se utilizaran en el controlador PI.

5. RESULTADOS DE SIMULACION Y EXPERIMENTALES

Con el modelo de la planta y el controlador PI se realizan las pruebas para verificar el cumplimiento de los requerimientos de sobrepaso menor al 15% y que el tiempo de establecimiento en lazo cerrado sea menor al obtenido en lazo abierto.

5.1. PRUEBAS DE SIMULACION

Figura 13. Diagrama de bloques del modelo de la Planta.



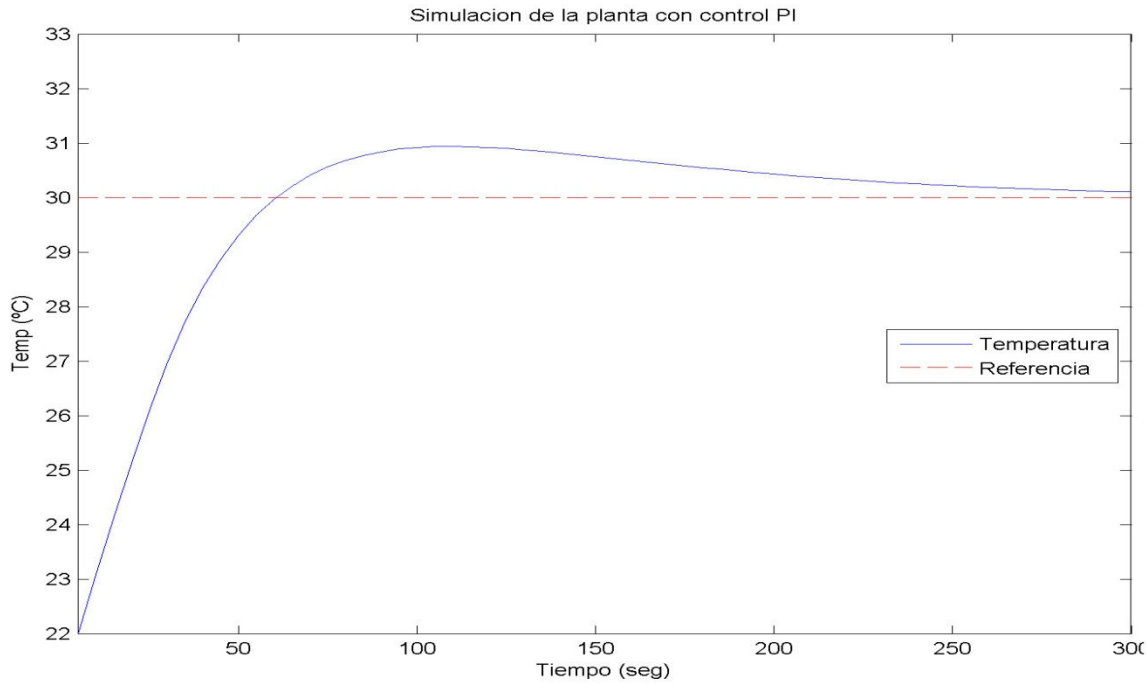
Fuente. Los autores.

En la figura 13 se muestran los bloques de *Simulink* necesarios para realizar la simulación. En este modelo se destacan, el controlador PI, la función de transferencia de la planta, dos bloques de señal escalón y un *scope*.

Para realizar la prueba de la respuesta transitoria del sistema, en la entrada del modelo se ubica una señal tipo escalón de 30°C, en la figura 14 se muestra la respuesta transitoria del sistema.

La temperatura ambiente (Ta) de 22°C entra al sistema como una perturbación. Además tiene un retardo (*Transport Delay*) que simula el intervalo de tiempo que introduce la planta.

Figura 14. Respuesta de la planta ante una entrada escalón.



Fuente. Los autores

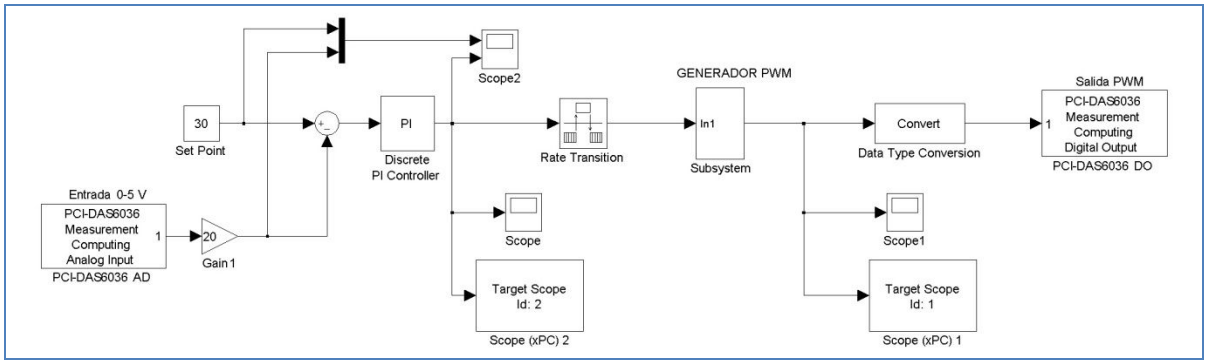
En figura 14 se muestra la respuesta del sistema en lazo cerrado ante una señal de entrada tipo escalón de 30°C.

El valor máximo de la salida del sistema se encuentra a una temperatura de 30.9ºC lo cual corresponde a un sobrepaso de 11.25% y se encuentra dentro de las condiciones de diseño establecidas. Se observa que el tiempo de establecimiento es menor al de lazo abierto ya que se alcanza en un tiempo de establecimiento de 250 segundos.

5.2. PRUEBAS EXPERIMENTALES

En la figura 15 se muestra el diagrama de bloques del sistema en lazo cerrado, utilizando un controlador PI. En este diagrama se utilizan los bloques *Digital Output* y *Analog Input* diseñados para que la tarjeta de adquisición de datos interactue físicamente con la planta en tiempo real.

Figura 15. Esquema utilizado para realizar el control de la planta.

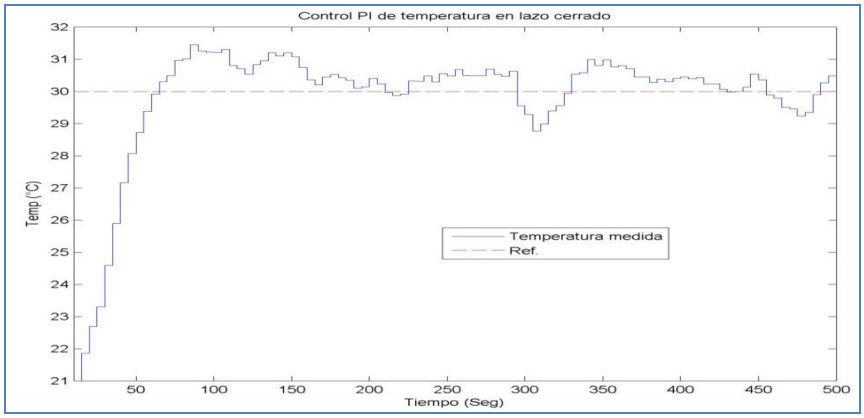


Fuente. Los autores.

Esta prueba se realiza con una temperatura de referencia de 30ºC que se resta con la señal de realimentación (Señal del sensor) y produciendo como resultado el error del sistema. Este error alimenta la entrada del controlador PI el cual se encarga de generar una acción de control determinada. Conforme a una acción de control dada, el bloque encargado de generar el PWM varía el ciclo de trabajo de la señal que se aplica al transistor de potencia que suministra potencia a la bombilla.

En la figura 16 se muestra el resultado experimental del control de temperatura en lazo cerrado cuando la temperatura de referencia es 30 ºC.

Figura 16. Control PI de temperatura en lazo cerrado.



Fuente. Los autores.

Con base en la figura 16 se puede concluir que el sistema cumple con las condiciones de diseño del controlador, puesto que el sobrepaso corresponde a un 14,4% y que el tiempo de establecimiento es de 150s, menor al obtenido en lazo abierto.

CONCLUSIONES

- En este trabajo se logró realizar el control de la variable de temperatura a través de la herramienta xPC-TARGET de Matlab.

- Se ha diseñado un sistema de control empotrado el cual cumple con los requerimientos necesarios para ejecutarse en tiempo real.

- Se ha diseñado un controlador PI utilizando un PC compatible y un software de arquitectura abierta como lo es *MATLAB* junto con su herramienta *SIMULINK* que permite diseñar un diagrama en bloques para el prototipo diseñado.

- Se ha diseñado un controlador PI para lograr que el sistema cumpla con los requerimientos de diseño.

- Se ha implementado una modulación de ancho de pulso (PWM) en Simulink para controlar la potencia suministrada a la bombilla.

- Se han programado los drivers de la tarjeta de adquisición de datos PCI-DAS6036 de la empresa *Measurement Computing* para trabajar en tiempo real añadidos a la *toolbox xPC-Target* de Matlab.

- Se ha demostrado la forma de utilizar la toolbox de Matlab *xPC-Target* para diseñar sistemas de tiempo real.

- Se han diseñado y construido 3 módulos (Transmisor, Receptor y fuentes) para realizar el control de proceso en tiempo real.

BIBLIOGRAFIA

ASTROM, KARL J, Computer-Controlled Systems Theory and Desing, Tercera Edicion, Editorial Prentice Hall, Pag 2.

COUGHLIN, ROBERT F. y DRISCOLL, FREDERICK F. Amplificadores operacionales y circuitos integrados lineales. Editorial Prentice Hall, 1999. Pags. 73-77, 371-373.

HEATH, STEVE. Embedded Systems Design, Second Edition. Butterworth-Heinemann, 2003. p.1

HRISTU-VARSAKELIS, DIMITRIOS, Handbook of networked and embedded control systems, Editorial Board. Pags 419-477.

KATSUHIKO OGATA, Ingeniería de Control Moderna, Pearson Education, S.A., Madrid, 2003, p. 1.

MALONEY, TIMOTHY J. Electrónica industrial moderna. 5a.Edición Pearson Educacion. Pags. 753,755.

RAMON PALLAS, ARNEY, Sensores y Acondicionadores de señal, Tercera Edicion, Editorial PRENTICE-HALL, Pag.68.

TANENBAUM, ANDREW S. Sistemas operativos modernos. Segunda edición, 2003. Prentice Hall. p.44

BRUYNINCKX, Herman. Real-Time and Embedded Guide. K.U. Leuven,Belgium. 2002. p. 2-3

Embedded System from ECE Version II Kharagpur, India.

FURR, STEVE. What is real time and why do I need it? .QNX Software Systems Ltd.

The Mathworks. *Getting Started with xPC Target*, Version 2 ed., The Mathworks, Inc., Natick, MA, July 2002.

MEASUREMEN COMPUTING CORPORATION, STC Register Map for the PCI-DAS6000 Series.

TEXAS INSTRUMENTS, Burr-Brown Products. XTR105, 4-20mA Current Transmitter with Sensor Excitation and Linearization.

TEXAS INSTRUMENTS, Burr-Brown Products. RCV420, Precision 4mA to 20mA Current Loop Receiver.

The Mathworks. xPC-TARGET 3, Perform real-time rapid prototyping and hardware-in-the-loop simulation using PC hardware, © 2006 by The Mathworks, Inc.

The Mathworks. *xPC-TARGET, User´s Guide*, Version 2 ed., The Mathworks, Inc., Natick, MA, November 2003.

LISTA DE ANEXOS

ANEXO A

1. TRANSDUCTORES Y SENSORES

Se denomina transductor, a todo dispositivo que convierte una señal de una forma física en una señal correspondiente pero de otra forma física distinta. Es, por tanto, un dispositivo que convierte un tipo de energía en otro.

Por ejemplo, al medir una fuerza, se supone que el desplazamiento del transductor es despreciable, es decir, que no carga al sistema, ya que de lo contrario podría suceder que este fuera incapaz de aportar la energía necesaria para el desplazamiento. Pero en la transducción siempre se extrae cierta energía del sistema que se mide, por lo tanto es importante garantizar que esto no lo perturba.

Un sensor es un dispositivo que, a partir de la energía del medio bajo medición, da una señal de salida transducible que es función de la variable medida. El sensor y transductor se emplean a veces como sinónimos, pero sensor sugiere un significado más extenso: la ampliación de los sentidos para adquirir un conocimiento de cantidades físicas que, por su naturaleza o tamaño, no pueden ser percibidas directamente por los sentidos. Por su parte el transductor, en cambio, sugiere que la señal de entrada y la de salida no deben ser homogéneas[24].

[24] RAMON PALLAS, ARNEY, Sensores y Acondicionadores de señal, Tercera Edicion, Editorial PRENTICE-HALL, Pag.3.

1.1 TIPOS DE SENSORES.

El número de sensores disponibles para las distintas magnitudes físicas es tan elevado que no se puede proceder racionalmente a su estudio sin clasificarlos previamente de acuerdo con algún criterio.

Según el aporte de energía, los sensores se pueden dividir en moduladores y generadores. En los sensores moduladores o activos, la energía de la señal de salida procede, en su mayor parte, de una fuente de energía auxiliar. La entrada solo controla la salida. En los generadores o pasivos, en cambio, la energía de salida es suministrada por la entrada.

Los sensores requieren en general más hilos que los generadores, ya que la energía de alimentación suele suministrarse mediante hilos distintos a los empleados para la señal.

Según la señal de salida, los sensores se clasifican en analógicos o digitales. En los analógicos la salida varia, a nivel macroscópico, de forma continúa. La información está en la amplitud, si bien se suelen incluir en este grupo los sensores con salida en el dominio temporal. Si es en esta forma de frecuencia, se denominan, a veces, casi digitales, por la facilidad con la que se puede convertir en una salida digital.

En los sensores digitales, la salida varía en forma de saltos o pasos discretos. No requieren conversión A/D y la transmisión de su salida es más fácil. Tienen también mayor fidelidad y mayor fiabilidad, y muchas veces mayor exactitud, pero lamentablemente

no hay modelos digitales para muchas magnitudes físicas de mayor interés.

Atendiendo a modo de funcionamiento, los sensores pueden ser de deflexión o de comparación. En los sensores que funcionan por deflexión, la magnitud medida produce algún efecto físico, que engendra algún efecto similar, pero opuesto, en alguna parte del instrumento y que está relacionado con alguna variable útil. Un dinamómetro para la medida de fuerzas es un sensor de este tipo en el que la fuerza aplicada deforma un muelle hasta que la fuerza de recuperación de este, proporcional a su longitud, iguala la fuerza aplicada.

Los sensores que funcionan por comparación, se intenta mantener nula la deflexión mediante la aplicación de un efecto bien conocido, opuesto al generado por la magnitud a medir. Hay un detector del desequilibrio y un medio para restablecerlo. En una balanza manual, por ejemplo, la colocación de una masa en un platillo provoca un desequilibrio, indicado por una aguja sobre una escala. En la tabla 1 se recogen todos estos criterios de clasificación.

Tabla 1. Clasificación de los sensores

Criterio	Clases	Ejemplos
Aporte de energía	Moduladores	Termistor
	Generadores	Termopar
Señal de salida	Analógicos	Potenciómetro
	Digitales	Codificador de posición
Modo de operación	De deflexión	Acelerómetro de deflexión
	De comparación	servo acelerómetro

Fuente. RAMON PALLAS, ARNEY, Sensores y acondicionadores de señal, Tercera Edición, Editorial PRENTICE-HALL, Pag. 54.

Desde el punto de vista de la Ingeniería Electrónica, es más atractiva la clasificación de los sensores de acuerdo con el parámetro de la variable: resistencia, capacidad, inductancia, presión, etc. En la tabla 2 se recogen los sensores y métodos de detección ordinarios para las magnitudes más frecuentes.

Tabla 2. Sensores y métodos de detección ordinarios para las magnitudes más frecuentes.

Sensores	Magnitudes Posición Distancia desplazamiento	Velocidad	Aceleración Vibración	Temperatura	Presión	Caudal Flujo	Nivel	Fuerza	humedad
Resistivos	Potenciómetros Galgas Magneto resistencias		Galgas + masa- resorte	RTD Termistores	Potenciómetros +tubo Bourdon	Anemómetros de hilo caliente Termistores	Potenciómetro + flotador Termistores LDR	Galgas	Humistor
Capacitivos	Condensador diferencial				Condensador variable + Diafragma		Condensador variable	Galgas capacitivas	Dieléctrico variable
Inductivos y electromagnéticos	LVDT Corrientes Foucault Efecto Hall	Ley Faraday LVT Efecto Hall Corrientes Foucault	LVDT + masa- resorte		LVDT + Diafragma Reductancia variable + Diafragma	LVDT + Rotámetro Ley de Faraday	LVDT + Flotador Corrientes de Foucault	Magneto – elástico LVDT + célula de carga	
Generadores			Piezoeléctricos + masa- resorte	Termopares Piroelectricos	Piezoeléctricos				
Digitales	Codificadores incrementales y absolutos	Codificadores incrementales		Osciladores de cuarzo	Codificadores +tubo Bourdon	Vórtices		Piezoeléctricos	SAW
Uniones p-n	Fotoeléctricos			Diodo Transistor Convertidores T/I			fotoeléctricos		
ultrasonidos	Reflexión	Efecto Doppler				Efecto Doopler	Reflexión Absorción		

Fuente. RAMON PALLAS, ARNEY, Sensores y acondicionadores de señal, Tercera Edición, Editorial PRENTICE-HALL, Pag. 56.

1.2 SENSORES RESISTIVOS.

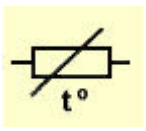
Los sensores basados en la variación de la resistencia eléctrica de un dispositivo son probablemente los más abundantes. Ello se debe a que son muchas las magnitudes físicas que afectan al valor de la resistencia de algún material. En consecuencia, ofrecen una solución válida para numerosos problemas de medida.

1.2.1 DETECTORES DE TEMPERATURA RESISTIVOS (RTD)

Los detectores de temperatura basados en la variación de una resistencia eléctrica se suelen designa con sus siglas inglesas *RTD* (*Resistance Temperature Detector*). Dado que el material empleado con mayor frecuencia para esta finalidad es el platino, se habla a veces de *PRT* (*Platinum Resistance Thermometer*).

El símbolo general para estos dispositivos es el de la figura 1. La línea recta en diagonal sobre el resistor indica que varía en forma intrínseca lineal y la anotación junto a dicha línea denota que la variación es debida a la temperatura y tiene coeficiente positivo.

Figura 1. Símbolo para una RTD.



Fuente. ALBERT D. HELFRICK, WILLIAM D. COOPER, Instrumentacion Electronica Moderna y Tecnicas de Medicion, Primera Edicion, Editorial PRENTICE-HALL.

El fundamento de las RTD es la variación de la resistencia de un conductor con la temperatura. En un conductor, el número de electrones disponibles para la conducción no cambia apreciablemente con la temperatura. Pero si ésta aumenta, las vibraciones de los átomos alrededor de sus posiciones de equilibrio son mayores y así dispersan más eficazmente los electrones, reduciendo su velocidad media. Esto implica un coeficiente de temperatura positivo, es decir, un aumento de la resistencia con la temperatura.

Los sensores RTD suelen ir asociados a montajes eléctricos tipo Puente de Wheatstone, que responden a la variación de la resistencia eléctrica por efecto de la temperatura para originar una señal analógica de 4-20 mA que es la que se utiliza en el sistema de control correspondiente como señal de medida.

Un tipo de RTD son las Pt100 o Pt1000. Estos sensores deben su nombre al hecho de estar fabricados de platino (Pt) y presentar una resistencia de 100ohms o 1000ohms respectivamente a 0ºC. Son dispositivos muy lineales en un gran rango de temperaturas, por lo que suele expresarse su variación como[25]:

$$Rt = R_{ref}(1 + \propto \Delta t) \qquad (1)$$

Donde: Rt = resistencia del conductor a la temperatura t (°C)

[25] ALBERT D. HELFRICK, WILLIAM D. COOPER, Instrumentacion Electronica Moderna y Tecnicas de Medicion, Primera Edicion, Editorial PRENTICE-HALL.

R_{ref}= resistencia a la temperatura de referencia normalmente a 0°C.

α= coeficiente de temperatura de resistencia.

Δt= diferencia entre la temperatura de referencia y la de operación.

ANEXO B

1. SISTEMAS DE COMUNICACIÓN PARA SENSORES

Las señales obtenidas por los sensores, una vez acondicionadas, hay que comunicarlas a un receptor o un dispositivo de presentación, cercano o remoto. Cuando el emisor y el receptor no están muy lejos, se suele emplear transmisión por hilos (par trenzado, cable coaxial, línea telefónica). También se emplea transmisión por hilos en instalaciones extensas que incluyan una infraestructura adecuada (redes de distribución eléctrica, oleoductos). Sus limitaciones son que tanto el ancho de banda como la velocidad de transmisión permitida son pequeños.

Para distancias muy grandes, o cuando el emisor o receptor son inaccesibles o interesa que se puedan mover libremente uno respecto al otro se emplea telemedida vía radio. Su ancho de banda y velocidad son mucho mayores. Cualquiera que sea el medio de comunicación empleado es necesario acondicionar las señales de los sensores para adaptarlas a las características de aquel[26].

1.1 Telemedida por corriente: bucle 4-20mA.

En la telemedida por corriente, la magnitud medida se convierte en una corriente continua proporcional, que se envía por la línea y es detectada en el extremo receptor midiendo la caída de tensión en una resistencia conocida. Los valores de corriente normalizados son: 4-20mA, 0-5mA, 0-20mA, 10-50mA, 1-5mA, y 2-10mA. Para evitar acoplamientos inductivos, que harían circular corrientes que interfieran entre si, se emplea un par de hilos trenzados. Las

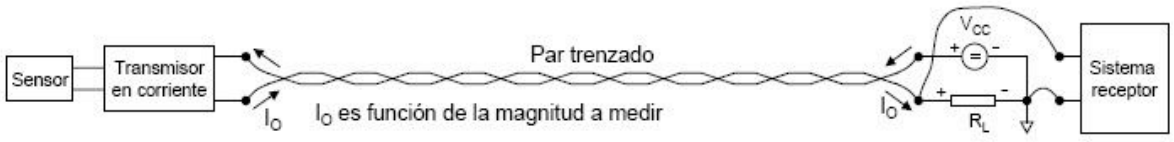
[26] RAMON PALLAS, ARNEY, Sensores y Acondicionadores de señal, Tercera Edición, Editorial PRENTICE-HALL, Pag.68.

interferencias capacitivas son pequeñas porque la impedancia equivalente del circuito es pequeña (la resistencia de entrada del receptor).

El valor de corriente habitual es 4-20mA. Que fue adoptado como norma en 1975 y existen numerosos transmisores comerciales que se ajustan a esta norma. La ventaja de emplear 4mA para el valor de cero es que permite distinguirlo de un circuito abierto (0mA).

Otra ventaja de la telemedida por corriente es que, si el transmisor es flotante, a veces es posible realizar el enlace con solo dos hilos compartidos por la alimentación y la señal. En la figura 1 se conecta la fuente de alimentación en serie con el dispositivo o dispositivos de lectura.

Figura1. Conexión típica para un Transmisor de Corriente.



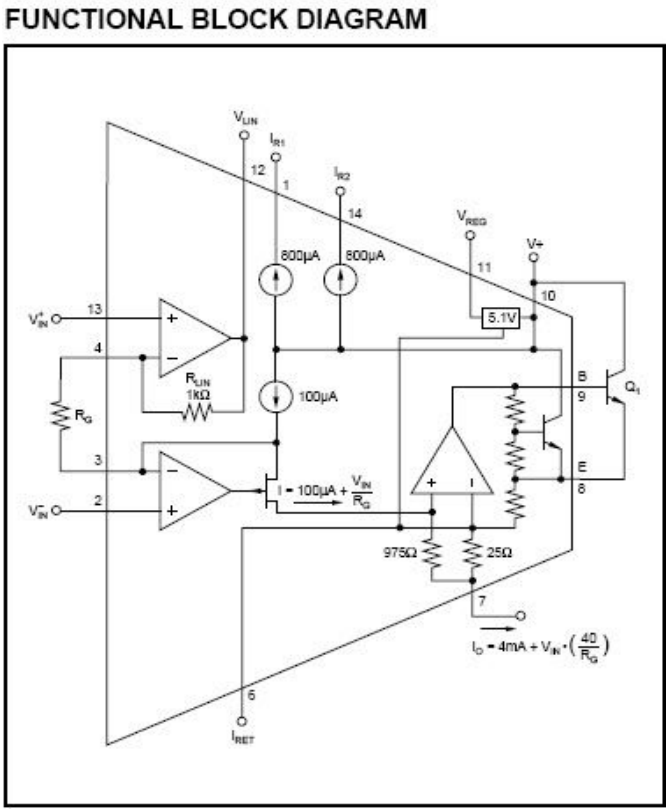
Fuente. RAMON PALLAS, ARNEY, Sensores y Acondicionadores de señal, Tercera Edicion, Editorial PRENTICE-HALL.

La posibilidad de emplear un sistema con solo dos hilos, con el consiguiente ahorro, depende de cómo sea el transmisor.

2. TRANSMISOR DE CORRIENTE

El XTR105 es un transmisor de corriente de 4-20mA con fuentes de precisión de corriente. Se trata de un circuito integrado de 14 pines que funcionalmente se puede representar según se muestra en la figura.

Figura 2. Diagrama funcional del XTR105.



Fuente. Burr-Brown Products from Texas Instruments. XTR105, 4-20mA Current Transmitter with Sensor Excitation and Linearization.

La ganancia del amplificador instrumentación puede configurarse para un amplio rango de medidas de temperatura o presión.

Algunas características del circuito integrado son:

- La etapa de entrada está formada por un amplificador de instrumentación con ganancia ajustable.

- Bajo desajuste de error para permitir su utilización en muchas aplicaciones.

- El circuito está dotado de dos fuentes de corriente (IR1, IR2) de 800μA cada una.

- La alimentación es unipolar en un amplio margen (de 7.5V a 36V).

- Dispone de ajuste de offset opcional con un potenciómetro externo.

- Operación con RTDs de dos y tres hilos.

- Bajo ruido en la corriente de salida 30nApp.

- El transistor externo opcional, cuando se usa, queda conectado en paralelo con un transistor interno. De esta forma, gran parte de la corriente de salida procede directamente de Vcc a través de Q1, reduciendo el auto calentamiento del XTR105 y aumentando su precisión al disminuir las derivas térmicas.

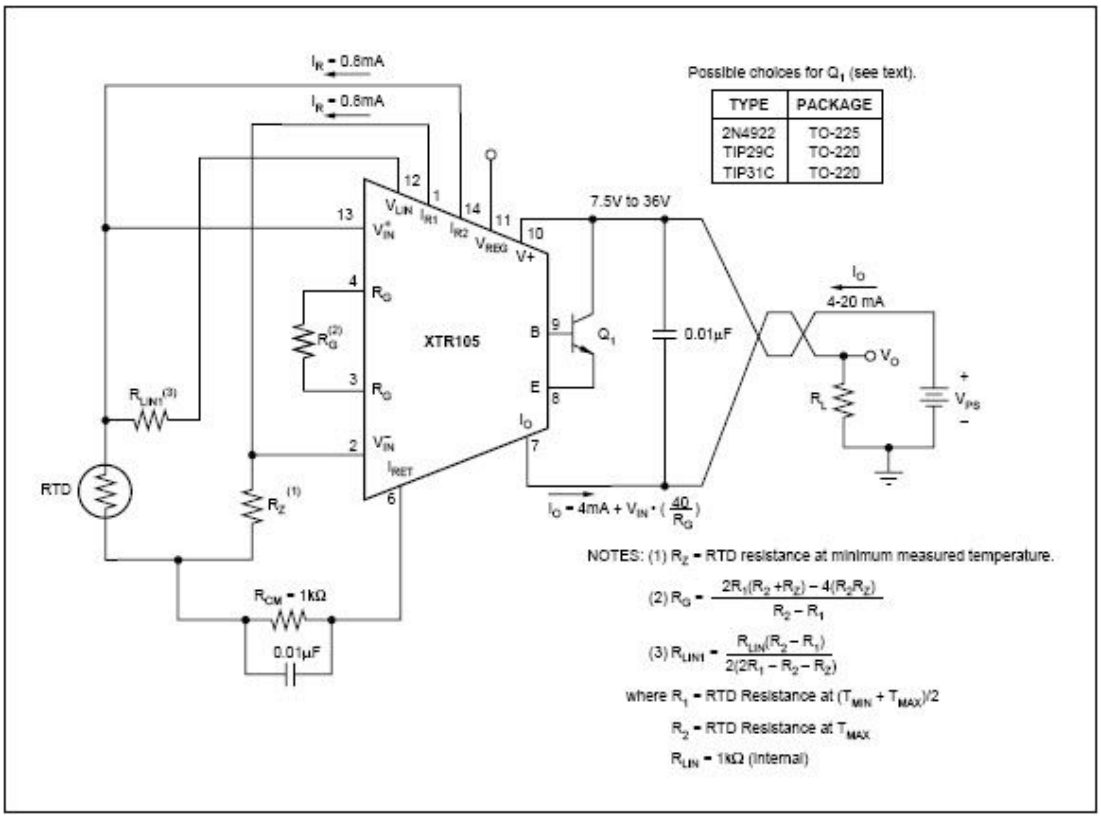
- Alto rechazo en modo común (CMR). Mínimo 86dB.

Dentro de sus aplicaciones están:

Algunas aplicaciones del XTR105 están en el control de procesos industriales, automatización industrial, SCADA, control remoto de temperatura y presión.

La figura 3 muestra el diagrama básico de conexión para el XTR 105 con una RTD de dos hilos.

Figura 3. Esquema de conexión del XTR105



Fuente. Burr-Brown Products from Texas Instruments. XTR105, 4-20mA Current Transmitter with Sensor Excitation and Linearization.

En la figura 3 se muestra el esquema de conecion del XTR105 el cual posee una fuente de alimentación Vps que proporciona la alimentación a todo el circuito. La corriente de salida puede ser medida en serie con la resistencia de carga RL.

Las dos fuentes de corriente drenan la RTD y en Rz se realiza el ajuste de cero, Rz debe ser del mismo valor de la RTD a la temperatura mínima que se pretenda medir para que a esta temperatura la corriente sea 4mA. El

amplificador de instrumentación mide la diferencia de voltaje entre la RTD y Rz y los condensadores se utilizan para filtrar el ruido de alta frecuencia.

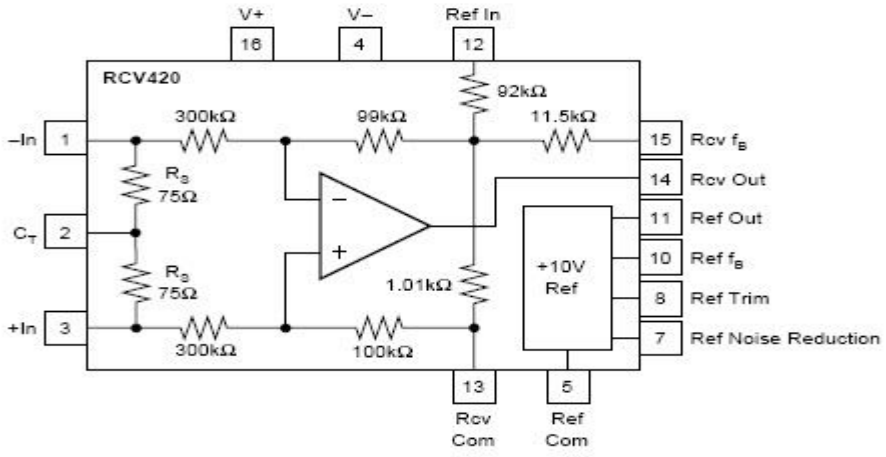
Los sensores para medir temperatura son fiables pero no lineales y con la adición de una o dos resistencias externas, RLIN1 y RLIN2, es posible compensar la mayor parte de esta no linealidad causando 40:1 que mejora la linealidad sobre la salida no compensada[27].

3. RECEPTOR DE CORRIENTE

EL RCV420 es un receptor de precisión de corriente que convierte la señal de entrada de 4 a 20mA en una señal de salida de 0 a 5 voltios.

Este circuito integrado contiene en su interior un amplificador operacional, una red de resistencias de precisión y una referencia de 10 voltios de precisión. Todo esto integrado en un chip que ofrece una alta rentabilidad a un bajo costo[28].

Figura 4. Esquema del RCV420.



Fuente. Burr-Brown Products from Texas Instruments. XTR105, Precision 4mA to 20mA Current Loop Receiver.

[27] Burr-Brown Products from Texas Instruments. XTR105, 4-20mA Current Transmitter with Sensor Excitation and Linearization.

[28] Burr-Brown Products from Texas Instruments. XTR105, Precision 4mA to 20mA Current Loop Receiver.

Algunas características de este circuito integrado son:

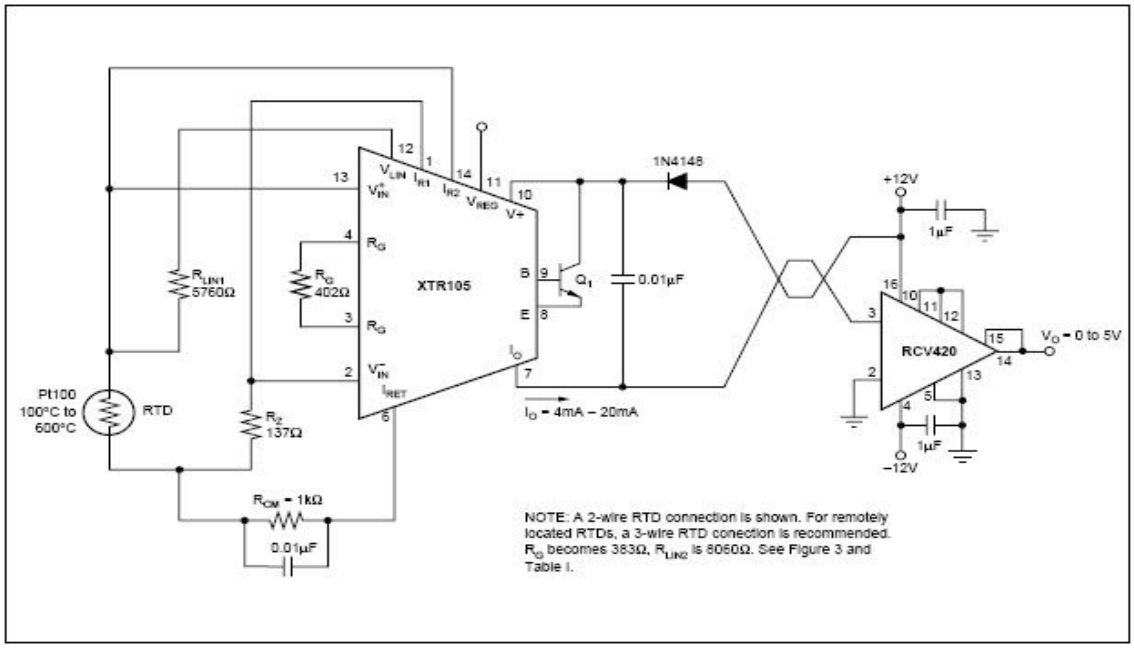
- Conversión de 4-20mA a 0-5V.
- Referencia de precisión de 10V.
- Rango de entrada en modo común ±40V.
- Alta inmunidad al ruido: 86dB.

APLICACIONES:

Dentro de las aplicaciones de este circuito integrado están el control de procesos, control Industrial, automatización industrial, adquisición de datos, SCADA, RTU.

El transmisor de corriente puede acoplarse perfectamente con el receptor, como se muestra en la figura 5.

Figura 5. Acople entre el transmisor y el Receptor de corriente



Fuente. Burr-Brown Products from Texas Instruments. XTR105, Precision 4mA to 20mA Current Loop Receiver.

La temperatura se mide a través de una RTD de dos hilos, esta señal llega a la entrada del XTR105 la cual convierte la señal de entrada en una señal de salida de 4-20mA, esta señal viaja a través de par trenzado la cual llegara a la entrada del RCV420 la cual convertirá esta señal de corriente en una señal de salida de voltaje de 0-5V, la cual ira a la entrada de la tarjeta de adquisición de datos PCI-DAS6036, esta señal se comparara con la referencia en el modelo de Simulink donde se realiza todo el control del proceso para mantener la temperatura constante.

ANEXO C
TARJETA DE ADQUISICIÓN DE DATOS PCI-DAS6036

La tarjeta de adquisición de datos PCI-DAS6036 es un potente instrumento de medición que se conecta a cualquier ranura PCI de un computador. Esta tarjeta puede ser utilizada para automatizar experimentos, construcción de productos, para la supervisión y control de procesos o puede ser incorporada en productos tales como sistemas de seguridad de los aeropuertos y aviones de combate de simuladores de prueba.

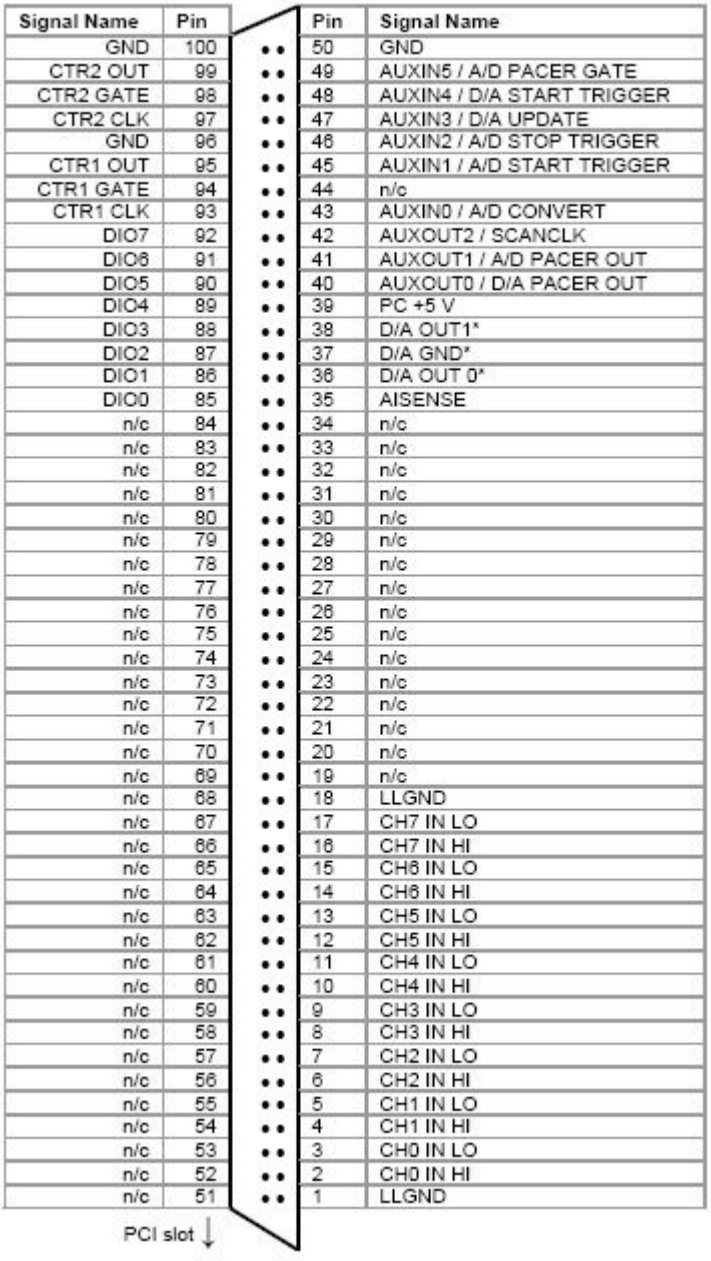
Algunas de las características de estas tarjetas son:

- 16 canales con una resolución A/D de 16 *bits*.
- Tiempo de muestreo de hasta 200.000 muestras por segundo.
- 2 salidas análogas de 16 *bits* de resolución.
- 8 líneas de E/S para las entradas y salidas digitales.
- 2 contadores de 16 bits.

A continuación se muestra la disposición de pines de la tarjeta de adquisición de datos.[29]

[29] Measurement Computing, User's Guide - PCI-DAS6036 Analog and Digital I/O Boards. Document Revision 3, September 2006.

Figura 1. Descripción de pines de la DAQ.



Signal Name	Pin	Pin	Signal Name
GND	100	50	GND
CTR2 OUT	99	49	AUXIN5 / A/D PACER GATE
CTR2 GATE	98	48	AUXIN4 / D/A START TRIGGER
CTR2 CLK	97	47	AUXIN3 / D/A UPDATE
GND	96	46	AUXIN2 / A/D STOP TRIGGER
CTR1 OUT	95	45	AUXIN1 / A/D START TRIGGER
CTR1 GATE	94	44	n/c
CTR1 CLK	93	43	AUXIN0 / A/D CONVERT
DIO7	92	42	AUXOUT2 / SCANCLK
DIO6	91	41	AUXOUT1 / A/D PACER OUT
DIO5	90	40	AUXOUT0 / D/A PACER OUT
DIO4	89	39	PC +5 V
DIO3	88	38	D/A OUT1*
DIO2	87	37	D/A GND*
DIO1	86	36	D/A OUT 0*
DIO0	85	35	AISENSE
n/c	84	34	n/c
n/c	83	33	n/c
n/c	82	32	n/c
n/c	81	31	n/c
n/c	80	30	n/c
n/c	79	29	n/c
n/c	78	28	n/c
n/c	77	27	n/c
n/c	76	26	n/c
n/c	75	25	n/c
n/c	74	24	n/c
n/c	73	23	n/c
n/c	72	22	n/c
n/c	71	21	n/c
n/c	70	20	n/c
n/c	69	19	n/c
n/c	68	18	LLGND
n/c	67	17	CH7 IN LO
n/c	66	16	CH7 IN HI
n/c	65	15	CH6 IN LO
n/c	64	14	CH6 IN HI
n/c	63	13	CH5 IN LO
n/c	62	12	CH5 IN HI
n/c	61	11	CH4 IN LO
n/c	60	10	CH4 IN HI
n/c	59	9	CH3 IN LO
n/c	58	8	CH3 IN HI
n/c	57	7	CH2 IN LO
n/c	56	6	CH2 IN HI
n/c	55	5	CH1 IN LO
n/c	54	4	CH1 IN HI
n/c	53	3	CH0 IN LO
n/c	52	2	CH0 IN HI
n/c	51	1	LLGND

PCI slot ↓

Fuente: Measurement Computing, User's Guide - PCI-DAS6036 Analog and Digital I/O Boards. Document Revision 3, September 2006. Pág: 13.

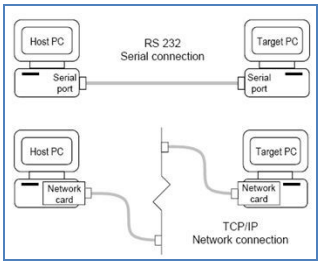
ANEXO D

1. *xPC Target*®

xPC Target® es una herramienta de la compañía *Mathworks*, que se usa para el diseño de prototipos, prueba de sistemas y desarrollo de sistemas en tiempo real, usando un computador estándar. El xPC Target se fundamenta en un ambiente que consta de un *PC Target* y un *PC Host*. El *PC Host* es un computador de escritorio con Matlab®, *Simulink*® y un compilador de C, que es usado para crear modelos usando bloques de *Simulink*® y a partir de estos generar el código en tiempo real utilizando la herramienta *Real Time Workshop*® de *Matlab*®. Posteriormente, este código se ejecuta en tiempo real en el *PC Target.*

El *PC Target* se arranca usando un disquete que contiene el núcleo en tiempo real. También se puede utilizar una memoria USB como dispositivo de arranque. Este disquete se crea con *xPC Target*®. Una vez cargado el sistema operativo con el disquete, se envía el código generado en *Simulink*® desde el *PC Host* al *PC Target* por medio de una conexión TCP/IP o una conexión serial vía RS-232. En la figura 1 se muestran estos dos tipos de conexión.[30]

Figura 1. Tipos de conexión en *xPC Target*®.



Fuente: The MathWork, xPC Target Guetting Started. The MathWorks, Inc.Version 3.4. Marzo, 2008. Pag. 1-13,1-14.

[30] The MathWork, xPC Target Guetting Started. The MathWorks, Inc.Version 3.4. Marzo, 2008. Pag. 1-2.

1.1. INSTALACIÓN Y CONFIGURACIÓN DE *xPC Target*®

En esta parte del documento, primero se describe y explica el *software* y *hardware* requeridos para diseñar un sistema de tiempo real con *xPC Target*. Finalmente, se muestra como se configura el explorador de xPC Target, se explican los tipos de conexiones disponibles para la comunicación entre el *PC Target* y el *PC Host* y como se crea el disquete de arranque.

1.1.1. Productos requeridos

xPC Target requiere 3 productos de *Mathworks* y un compilador de lenguaje C para ejecutar aplicaciones de tiempo real. Estos productos son: Matlab®, *Simulink*® y *Real Time Workshop*®. A continuación se describe cada uno de ellos.

1.1.1.1. *Matlab*®

Matlab® proporciona una interfaz de línea de comandos con el xPC Target®. Con el xPC Target®, se tiene control total del computador *target* y de las aplicaciones ejecutadas en éste, por medio de las funciones de xPC TARGET y de la línea de comandos de MATLAB. El xPC Target puede realizar las siguientes acciones:

- **Control de las aplicaciones de tiempo real:** Descargar, iniciar y detener las aplicaciones en el computador *target*.
- **Adquisición y análisis de señales:** Es posible guardar datos de señales mientras una aplicación está corriendo

en el equipo *target* y analizar estos datos posteriormente. También se puede visualizar señales mientras la aplicación se está ejecutando en tiempo real.

- **Sintonización de parámetros:** Esta característica indica que es permitido cambiar parámetros mientras se está ejecutando la aplicación en tiempo real.

1.1.1.2. *Simulink*®

Simulink® proporciona un ambiente donde se puede modelar un sistema físico y controlar en forma de diagramas de bloques. Se utiliza el *xPC Target*® con la mayoría de bloques de *Simulink*®, incluyendo los de tiempo discreto y los de tiempo continuo.

1.1.1.3. *Real Time Workshop*®

Real-Time Workshop® permite convertir modelos de *Simulink*® a código C y luego, con un compilador C/C++, compilar el código en un ejecutable de tiempo real.

1.1.1.4. El compilador de C

El compilador de C crea archivos ejecutables a partir del código en C generado en *Real-Time Workshop*® y del código de las *S-functions* que el usuario crea. El xPC Target utiliza este código ejecutable para crear una

imagen ejecutable (aplicación *target*), que corre con el *kernel* de xPC target en el computador *target.*[31]

1.1.2. Requerimientos del sistema

Los requerimientos de *hardware* y *software* son distintos para el computador *host* y para el computador *target*.

1.1.2.1. Requerimientos del equipo *host*.

El computador o equipo *host* es usualmente el equipo donde se instala *Matlab®, Simulink®, Real-Time Workshop®, xPC Target®.*

- **Requerimientos de *software***

  La tabla 1 muestra los requerimientos de *software* para el computador *host*.

Tabla 1. Requerimientos de *software* para el *PC host.*

Software	Descripción
Sistema operativo de 32 bits	Plataformas de Windows que soporten los productos de *The MathWorks.*
MATLAB	MATLAB Versión 7.6
Simulink	Simulink Versión 7.1
Compilador de lenguaje C	Microsoft® Visual Studio® C/C++

[31] The MathWork, xPC Target Guetting Started. The MathWorks, Inc.Version 3.4. Marzo, 2008. Cap. 2,Pag. 3-5.

	Professional Edition Version 6.0, 7.1, or 8.0 Open Watcom C/C++ Version 1.3
Real-Time Workshop	Real-Time Workshop Versión 7.1
xPC Target	xPC Target Versión 3.4

Fuente: The MathWork, xPC Target Guetting Started. The MathWorks, Inc.Version 3.4. Marzo, 2008. Cap. 2, Pag. 8-9.

- **Requerimientos de *hardware***

  La tabla 2 muestra los requerimientos mínimos de *hardware* para el computador *host*.

Tabla 2. Requerimientos de *hardware* para el *PC host.*

Hardware	Descripción
Comunicación	Un puerto serial libre (COM1 o COM2), o una tarjeta Ethernet conectada a una red.
CPU	Pentium, Athlon, o mejor.
Periféricos	Disco duro con 60 MB de espacio libre. Una unidad de disquete.
RAM	128 MB o más.

Fuente: The MathWork, xPC Target Guetting Started. The MathWorks, Inc.Version 3.4. Marzo, 2008. Cap. 2, Pag. 9.

1.1.2.2. Requerimientos del equipo *target*

El equipo *target* puede ser un computador de escritorio o un sistema de uso industrial, tal y como un *PC/104* o un *CompactPCI*.

- **Requerimientos de *software***

  En la tabla 3 se muestran los requerimientos de *software* para el computador *target*.

Tabla 3. Requerimientos de *software* para el *PC target*.

Software	Descripción
Sistema operativo	Ninguno.
BIOS	Compatible con PC.

Fuente: The MathWork, xPC Target Guetting Started. The MathWorks, Inc.Version 3.4. Marzo, 2008. Cap. 2, Pag. 10.

- **Requerimientos de *hardware***

  En la tabla 4 se muestran los requerimientos mínimos de *hardware* para el computador *target*.

Tabla 4. Requerimientos de *hardware* para el *PC target.*

Hardware	Descripción
Chipset	PC compatible con UART, controlador programable de interrupciones, controlador de teclado, y contador.
Comunicación	Un puerto serial libre (COM1 o COM2), o una tarjeta Ethernet conectada a una red.
CPU	Intel® 386/486/Pentium o AMD® K5/K6/Athlon con o sin coprocesador de punto flotante.
Teclado	Se necesita para controlar el computador *target* cuando se crean aplicaciones que corren en modo *stand-alone*.
Pantalla	Se recomienda usar una pantalla, pero no es necesario. Se puede obtener toda la información del equipo *target* usando las funciones de xPC Target en el equipo *host*.
RAM	8 MB o más.
Periféricos	Una unidad de disquete.

Fuente: The MathWork, xPC Target Guetting Started. The MathWorks, Inc.Version 3.4. Marzo, 2008. Cap. 2, Pag. 10-11.

1.1.3. El explorador de *xPC Target*®

xPC Target® tiene una interfaz gráfica de usuario (GUI) que permite configurar e interactuar con el xPC Target® de una manera sencilla. Con esta interfaz se pueden realizar operaciones básicas, tales como:

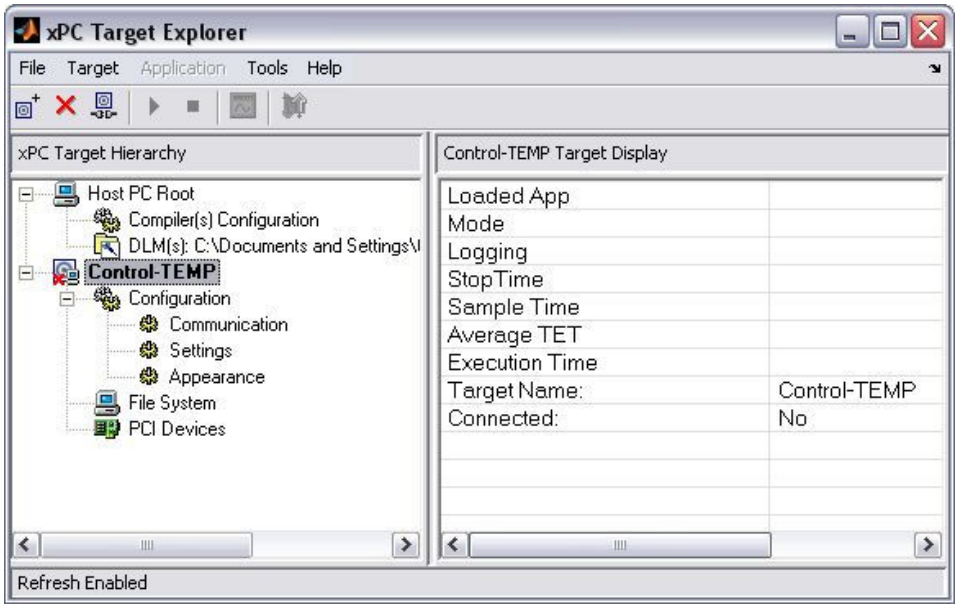
- Configurar el equipo *host* para trabajar con el software de xPC Target®.
- Agregar y configurar los equipos *target* para trabajar con el software de xPC Target®. Es posible configurar hasta 64 equipos *target*.
- Crear discos de arranque para cada equipo *target*.
- Conectar los equipos *target* a el equipo host.
- Descargar al equipo *target* un aplicación de tiempo real de xPC Target®. Estas aplicaciones son archivos que tienen extensión .DLM.
- Iniciar y parar la aplicación que se descargó en el equipo *target*.
- Agregar *scopes* de tipo *host*, *target* y *file* a la aplicación descargada en el equipo *target*.
- Seguimiento de señales.
- Agregar señales a los *scopes* del computador *target* y removerlas.
- Iniciar y parar los *scopes*.
- Ajustar los valores de los parámetros para las señales mientras la aplicación en el equipo target se ejecuta.[32]

[32] The MathWork, xPC Target Guetting Started. The MathWorks, Inc.Version 3.4. Marzo, 2008. Cap. 2, Pag. 20.

Para acceder al explorador de *xPC Target*® se debe ejecutar el comando *xpcexplr* en la ventana de comandos de *Matlab*®.

En la figura 2 se muestra el explorador de *xPC Target*®.

Figura 2. Explorador de *xPC Target*®.



Fuente: Los autores.

En esta ventana se pueden observar todos los elementos que contiene el sistema *xPC Target*®. Conforme se agreguen objetos al sistema, el explorador de *xPC Target*® agrega los nodos correspondientes en la jerarquía de *xPC Target*®. El nodo más importante es *Host PC Root*. Este nodo representa el equipo host. El panel derecho muestra información del *ítem* seleccionado en el panel de la izquierda.

1.1.4. Comunicación serial.

Antes de poder crear y ejecutar una aplicación en el equipo *target*, se necesita configurar la conexión entre los equipos *host* y *target*. En este documento no se profundizará en la comunicación tipo serial, pues el sistema se configuró con conexión tipo *Ethernet*. Si es necesario configurar una conexión serial para comunicar los equipos, refiérase al manual de usuario de *xPC Target*®.[33]

1.1.5. Comunicación por red

Es posible conectar los equipos *host* y *target* mediante una red de computadores.

1.1.5.1. Ventajas de una comunicación por red

La conexión entre el *host* y el *target* usando redes TCP/IP tiene ventajas sobre la comunicación serial RS-232:

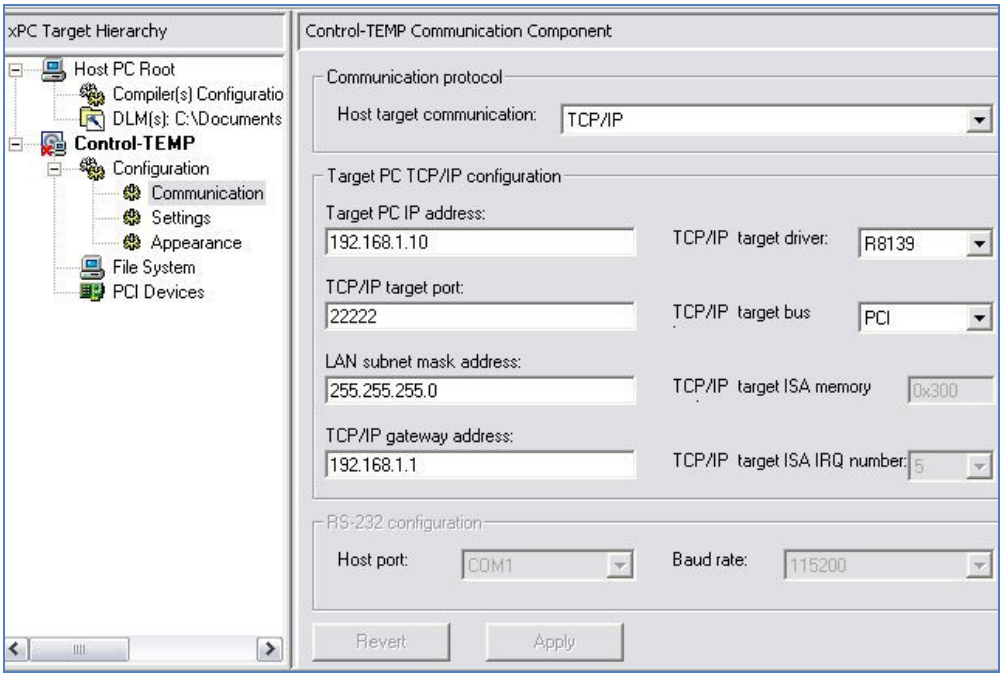
- Tasa de transferencia de datos más alta: La conexión por red usando *ethernet* puede transferir datos en una velocidad de hasta 100 Mbit/segundos, en vez de la tasa máxima de transferencia de 115 kBaudios de la comunicación serial.
- Distancias más largas entre el *host* y el *target*: Usando repetidores y *routers* no se restringe la distancia entre el equipo *host* y el *target*. Ademas, también es posible la comunicación por *Internet*.

[33] The MathWork, xPC Target Guetting Started. The MathWorks, Inc.Version 3.4. Marzo, 2008. Cap. 2, Pag. 24.

1.1.5.2. Configuración de una conexión por red

En la figura 3 se muestra la manera como se ha configurado la conexión por red para el sistema de control en tiempo real, usando xPC TARGET.

Figura 3. Configuración de la conexión de red.



Fuente: Los autores.

A continuación se describen los pasos para configurar la conexión por red, conectando los dos equipos directamente por medio de un cable cruzado:

1. Seleccionar *communication* en el panel izquierdo del explorador de xPC TARGET.

2. Escoger TCP/IP en el menú desplegable *Host target communication.*
3. La dirección IP para el equipo *target* debe encontrarse en el rango de direcciones en que se encuentra la puerta de enlace. En este caso se asigna 192.168.1.10.
4. El puerto por donde se comunican los equipos es 22222 por defecto.
5. La dirección de la máscara de red es 255.255.255.0
6. Se asigna la dirección 192.168.1.1 a la puerta de enlace, que en este caso es la dirección IP del equipo *host.* Si la red se hace por medio de un repetidor o *router,* la dirección de la puerta de enlace debe ser la IP del *router* o del repetidor.
7. La tarjeta de red usada en este proyecto es una Realtek 8139, por lo tanto el tipo de driver escogido en la opción *TCP/IP target driver* debe ser R8139. Si se tiene otro tipo de tarjeta de red, se debe verificar en el manual de usuario de *xPC Target*®, que la tarjeta es compatible con *xPC Target*®.
8. El tipo de *bus* de la tarjeta de red es PCI.[34]

1.1.6. Disco de arranque del equipo *target*

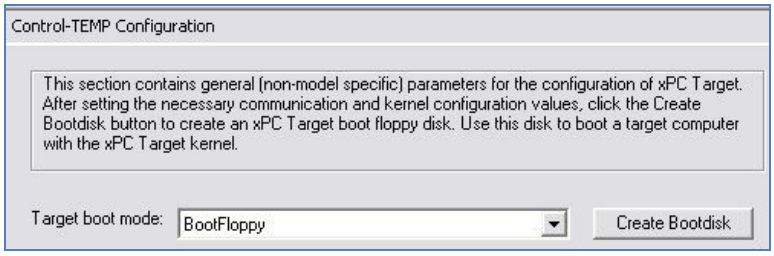
El disco de arranque incluye el *kernel* de *xPC Target*® y los archivos de configuración de comunicación serial o comunicación por red.

[34] The MathWork, xPC Target Guetting Started. The MathWorks, Inc.Version 3.4. Marzo, 2008. Cap. 2, Pag. 31-45.

A continuación se mencionan los pasos que se deben seguir para crear un disquete de arranque de *xPC target*®:

2. Seleccionar la opción *configuration* en el panel izquierdo del explorador de xPC TARGET.
3. Seleccionar *BootFloppy* en la opción *Target boot mode tal y* como se muestra en la figura 4.

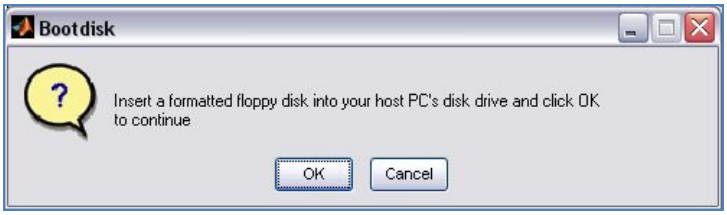
Figura 4. Modo de arranque del *PC target.*



Fuente: Los autores.

4. Hacer *click* en el botón *Create Bootdisk*. Aparecerá una venta como la que se muestra la figura 5 y en donde se debe hacer un click en *OK* para que comience el proceso de creación del disco.

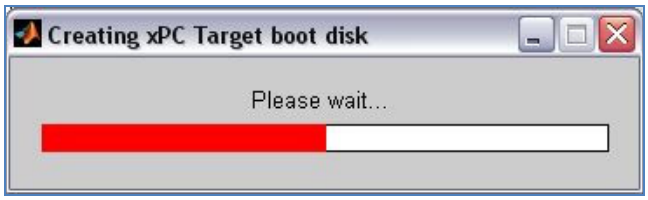
Figura 5. Creación del disco de arranque.



Fuente: Los autores.

5. El *xPC Target*® muestra el progreso del proceso de creación tal y como se puede observar en la figura 6. Este proceso toma entre 1 y 2 minutos para completarse.

Figura 6. Proceso de creación del disco de arranque.



Fuente: Los autores.

6. Finalmente, inserte el disco de arranque en el equipo target y reinícielo para que el equipo arranque con el sistema en tiempo real.[35]

Se ha descrito la forma como se debe configurar la herramienta *xPC Target*® para crear sistemas que trabajen en tiempo real. En el ANEXO D se dan a conocer las características de Simulink y el procedimiento para crear modelos, que posteriormente se cargarán en el equipo *target,* para ejecutarlos en tiempo real.

[35] The MathWork, xPC Target Guetting Started. The MathWorks, Inc.Version 3.4. Marzo, 2008. Cap. 2, Pag. 46-54.

ANEXO E

MOSFET IRF-Z44

En la planta de control de temperatura se usa un MOSFET IRF-Z44 para suministrar potencia a la bombilla. Este integrado es un MOSFET de enriquecimiento de canal N. Las principales características de este MOSFET se muestran en la figura 1.

Figura 1. Características del MOSFET IRF-Z44

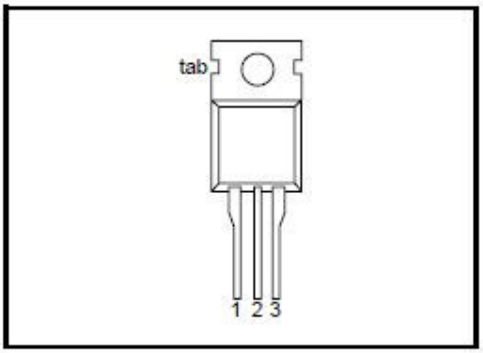
SYMBOL	PARAMETER	MAX.	UNIT
V_{DS}	Drain-source voltage	55	V
I_D	Drain current (DC)	49	A
P_{tot}	Total power dissipation	110	W
T_j	Junction temperature	175	˚C
$R_{DS(ON)}$	Drain-source on-state resistance V_{GS} = 10 V	22	mΩ

Fuente. Philips Semiconductors, Product Specification. N-channel enhancement mode TrenchMOSTM transistor - IRFZ44N. Pag. 1

La figura 2 muestra la distribución de pines del IRF-Z44.

Figura 2. Distribución de pines del integrado.

PIN	DESCRIPTION
1	gate
2	drain
3	source
tab	drain



Fuente. Philips Semiconductors, Product Specification. N-channel enhancement mode TrenchMOSTM transistor - IRFZ44N. Pag. 1

ANEXO F

1. ACONDICIONADOR DE SEÑAL

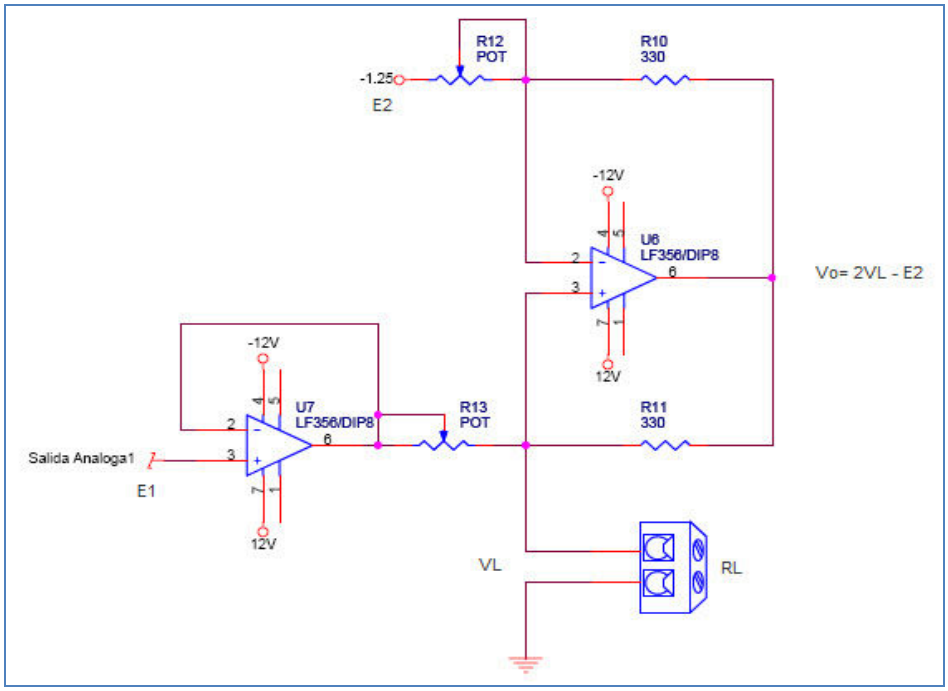
Los acondicionadores de señal, adaptadores o amplificadores en sentido amplio, son elementos del sistema de medida que ofrecen a partir de la señal de salida de un sensor electrónico, una señal apta para ser presentada o registrada o que simplemente permita un procesamiento posterior mediante un equipo o instrumento estándar. Consisten normalmente en circuitos electrónicos que ofrecen, en otras funciones, las siguientes: amplificación, filtrado, adaptación de impedancias y modulación o demodulación.

Por ejemplo, en el caso en que una de las etapas de tratamiento de la señal de medida es digital, si la salida del sensor es analógica, que es lo más frecuente, hará falta un convertidor A/D. Estos tienen una impedancia de entrada limitada, exigen que la señal aplicada sea continua o de frecuencia de variación lenta y que su amplitud este entre unos límites determinados que no suelen exceder de 10V. Todas estas exigencias obligan a interponer un acondicionador de señal entre el sensor (que muchas veces ofrece señales de apenas unos milivoltios) y el convertidor A/D.

1.1 Convertidor de voltaje diferencial a corriente

En la figura 1 se muestra el circuito convertidor de voltaje a corriente que se diseña para el proyecto.

Figura1. Acondicionador de señal de Voltaje a Corriente.



Fuente. Los Autores.

El circuito de la figura 1 es un convertidor de voltaje diferencial a corriente debido a que la corriente de carga IL depende de la diferencia entre los voltajes de entrada E1 y E2 y la resistencia R. La corriente IL no depende de la resistencia de carga RL, por tanto, si E1 y E2 son constantes, la carga conectada a tierra está alimentada por una corriente constante. La corriente de carga puede fluir en cualquier dirección, de modo que este circuito podría ser fuente o consumidor de corriente[36].

La corriente de carga IL se determina por:

$$I_L = \frac{(E_1 - E_2)}{R} \qquad (1)$$

[36] ROBERT F. COUGHLIN, Amplificadores Operacionales y Circuitos Integrados Lineales, Cuarta Edicion, Editorial ALFAOMEGA, Pag. 120

Un valor positivo de IL significa que fluye hacia abajo y VL es positivo con respecto a tierra. Un valor negativo de IL significa que VL es negativo con respecto a tierra y la corriente fluye hacia arriba.

El voltaje de carga VL depende de la resistencia de carga RL:

$$V_L = I_L * R_L \quad (2)$$

Para asegurar que el amplificador operacional no se sature, V_o debe conocerse y es posible calcularlo mediante:

$$V_o = 2V_L - E_2 \quad (3)$$

ANEXO G

1. *SIMULINK®*

Simulink es un *software* con el que se puede crear modelos, simular y analizar sistemas dinámicos. Con *Simulink*, es posible construir fácilmente modelos empezando desde cero, o modificar modelos existentes para que se ajusten a una necesidad específica. *Simulink* soporta sistemas lineales y no lineales, modelos en tiempo continuo o en tiempo discreto, o una combinación de los dos. Los sistemas también pueden ser *multirate*, es decir, que pueden tener diferentes partes del modelo con diferentes periodos de muestreo.

Simulink puede utilizarse para modelar y resolver problemas reales, en una gran variedad de industrias, incluyendo:

- Aeroespacial y Defensa.
- Automotriz.
- Comunicaciones.
- Electrónica y Procesamiento de señal.
- Instrumentación Médica.[37]

1.1. Creación de modelos en *Simulink*

La creación de aplicaciones en tiempo real con *xPC Target* consta de 3 etapas importantes: La primera de ella es crear un modelo de *Simulink* a partir del cual se genera un código en lenguaje C mediante la

[37] The MathWork, Simulink Guetting Started guide. The MathWorks, Inc.Tercera Impresión. Marzo, 2008. Pag. 1-2.

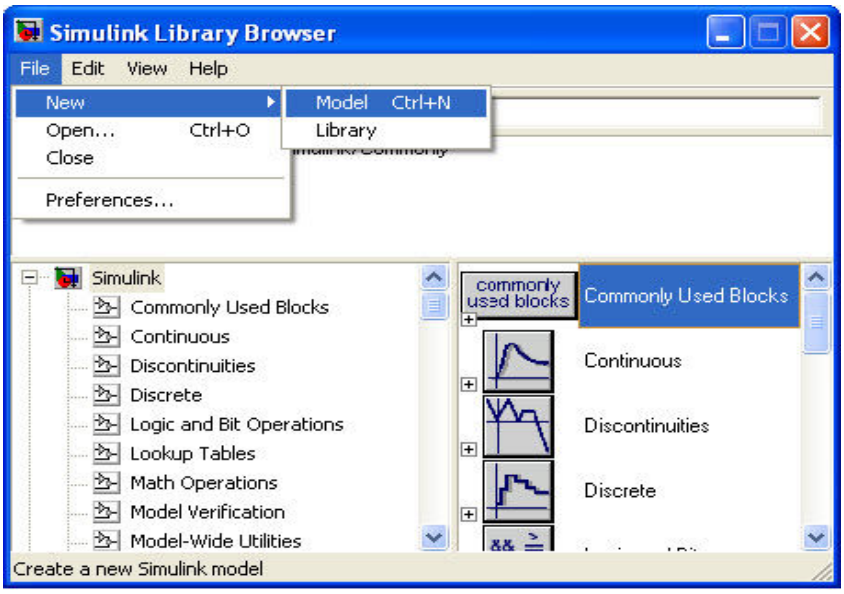
herramienta *Real-Time Workshop* y por ultimo este código se compila para crear la aplicación que se cargará posteriormente al equipo *target*.
A continuación se muestra un ejemplo sencillo utilizando *Simulink*, donde se explican los diferentes pasos que se deben seguir para crear una aplicación que funcione en tiempo real, usando xPC Target.

El modelo está conformado por un bloque generador de pulso y un bloque de salida digital de la tarjeta de adquisición de datos PCI-DAS 6036. También se han agregado dos osciloscopios (*scopes)* para visualizar la señal del generador de pulso tanto en el PC Host como en el PC Target.

1. En la ventana de comandos de MATLAB, teclear: *simulink.*

2. En el menú *file*, seleccionar la opción *new,* y hacer un *click* en *model*.

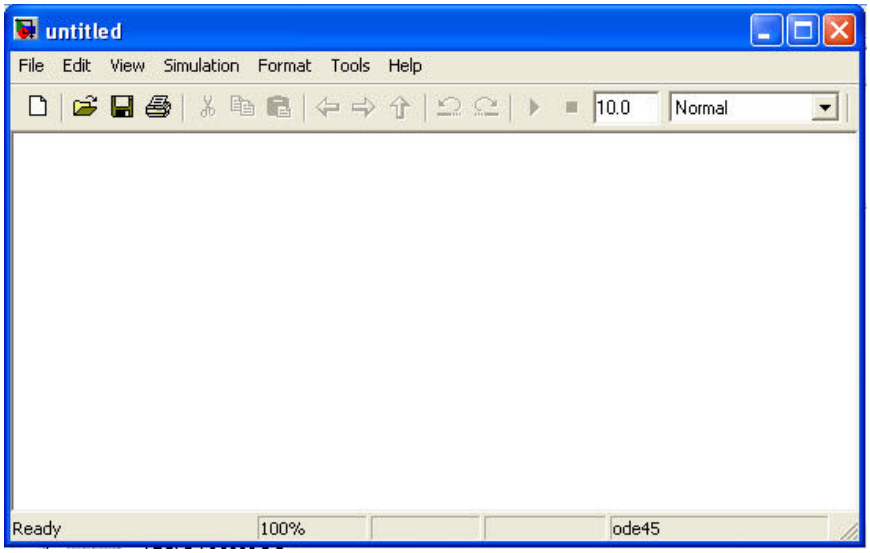
Figura 1. Creación de un nuevo modelo.



Fuente: Los autores.

Con el anterior procedimiento se abre una ventana con un modelo de *simulink* en blanco como se muestra en la figura 2.

Figura 2. Modelo en blanco.



Fuente: Los autores.

3. En el panel izquierdo de la ventana librería, haga doble click en el menú *simulink*.

4. En ***sources*** busque y arrastre al modelo en blanco el bloque ***Pulse Generator.***

5. En ***sinks*** busque y arrastre al modelo en blanco el bloque ***scope.***

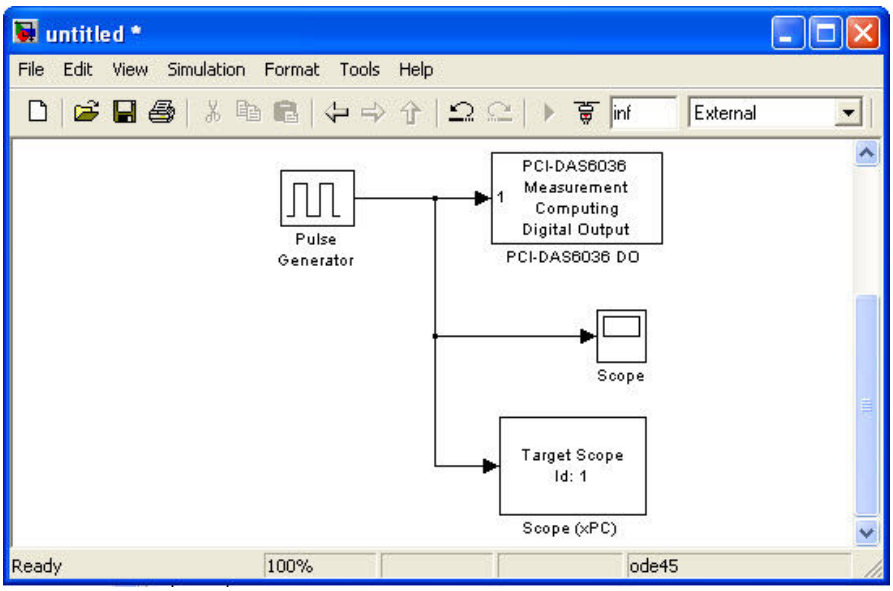
6. Nuevamente en el panel izquierdo de la ventana librería haga doble *click* en *xPC Target*.

7. Busque el bloque ***scope (xPC)*** en el menú ***Misc***.

8. En digital output se debe seleccionar el bloque para la salida digital, dependiendo de la marca y del modelo de la tarjeta de adquisición de datos. En este caso es una tarjeta PCI-DAS6036, de la empresa *Measurement Computing*.

   Después de seleccionar y poner en el modelo los bloques necesarios, se deben unir según sea necesario. La figura 3 muestra el modelo con sus bloques unidos entre sí.

Figura 3. Bloques de *Simulink.*



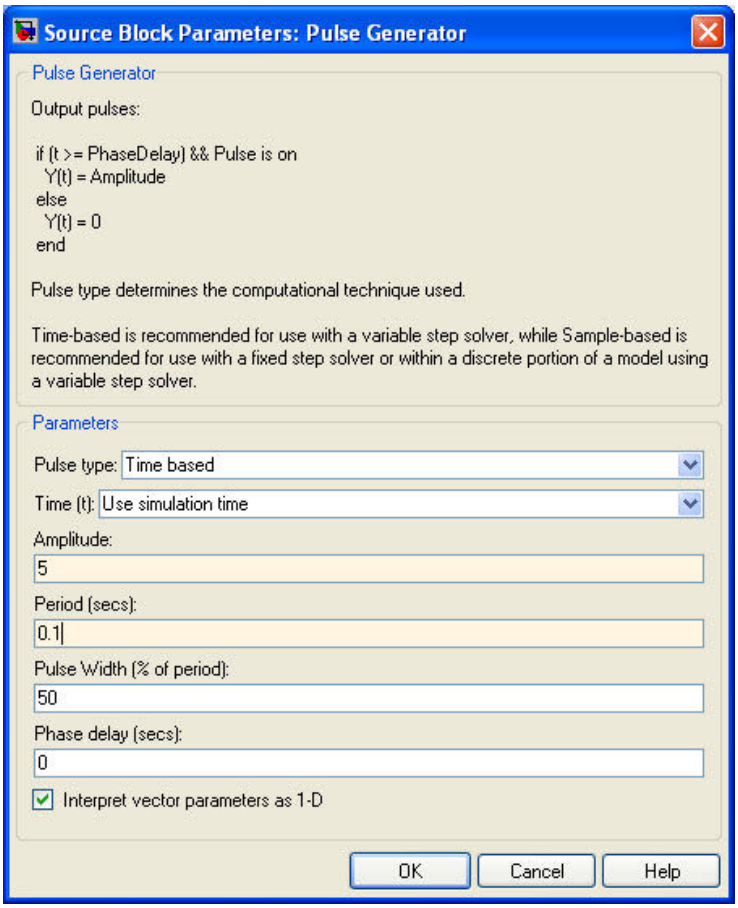
Fuente: Los autores.

Con los anteriores pasos se describe como crear un modelo sencillo de *simulink*. A continuación se muestra el procedimiento para hacer las configuraciones necesarias en cada bloque, así como las configuraciones necesarias para que el modelo funcione con xPC Target.

1. Para editar los parámetros de un bloque se hace doble *click* sobre éste.

2. Hacer doble *click* en el bloque pulse *generator*. La figura 4 muestra los parámetros para este bloque.

Figura 4. Parámetros del bloque *generator.*

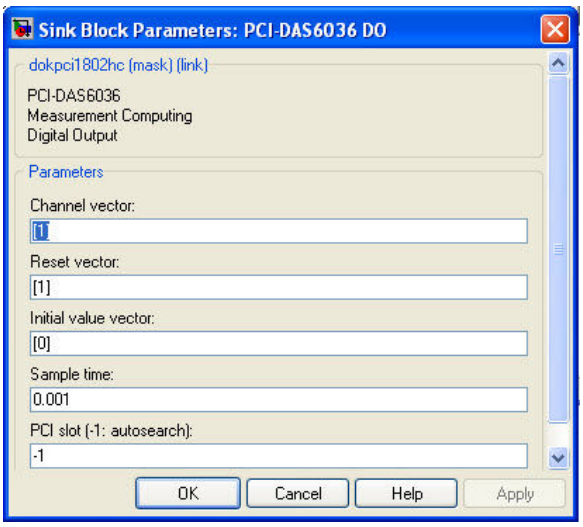


Fuente: Los autores.

Solo se cambia el parámetro de amplitud en 5 voltios y el periodo en 0.1 segundos. Con esto se genera una onda cuadrada que varía entre 0 y 5 voltios y tiene una frecuencia de 10Hz.

3. Para editar los parámetros del bloque *digital ouput*, se hace doble *click* en éste. En la figura 5 se muestran estos parámetros.
   Solamente se modifica el tiempo de muestreo, es decir en el parámetro *Sample time* se asigna el valor 0.001.

Figura 5. Parámetros del bloque *digital ouput*.



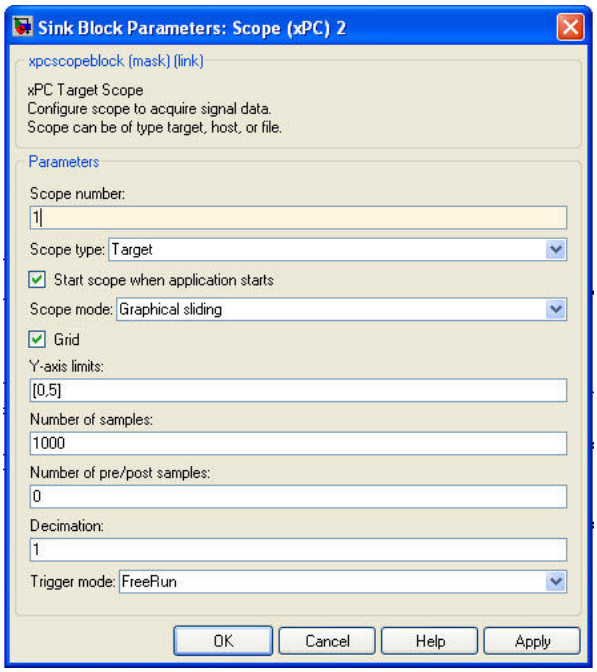
Fuente: Los autores.

4. Hacer doble *click* en el bloque *scope* (xPC) para modificar sus parámetros. Estos parámetros se muestran en la figura 6.

   Se modifica el numero de muestras a 1000 muestras y se cambian los limites para el eje Y por [0,5].

5. Por último configuramos los parámetros del modelo de Similink. Esta opción se encuentra en el menú Simulation/Configuration Parameters de *simulink*.

Figura 6. Parametros del bloque *scope* (xPC).



Fuente: Los autores.

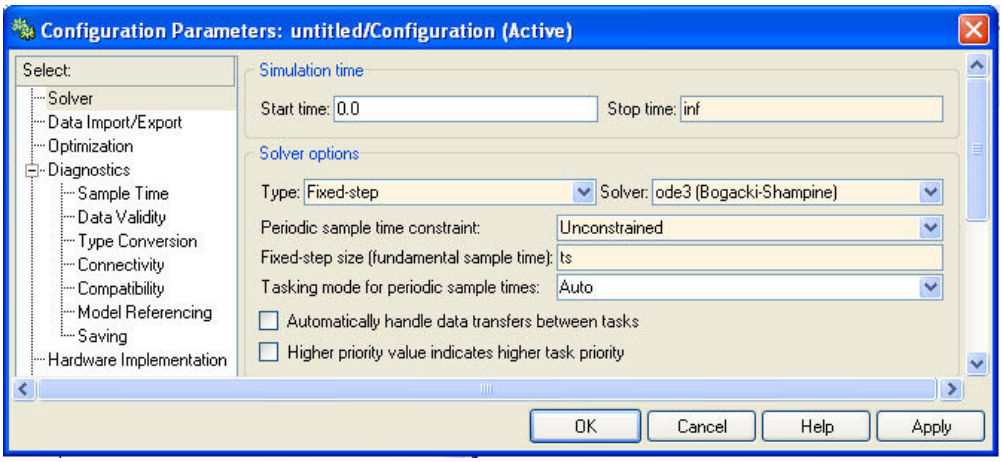
En esta ventana se configuran el tiempo de ejecución de la simulación. Cuando se configura *stop time* como *inf*, la simulación se ejecutara cíclicamente hasta que el usuario la detenga.

En el campo *Type* seleccionamos la opción Fixed-step para poder configurar posteriormente el tiempo de muestreo.
El resto de opciones se dejan como se muestra en la figura 7.

Posteriormente se configura la opción *Real-Time Workshop*

6. En el campo *System Target File* se selecciona xpctarget.tlc que es el target correspondiente a xPC Target. El resto de parámetros se dejan como vienen por defecto.

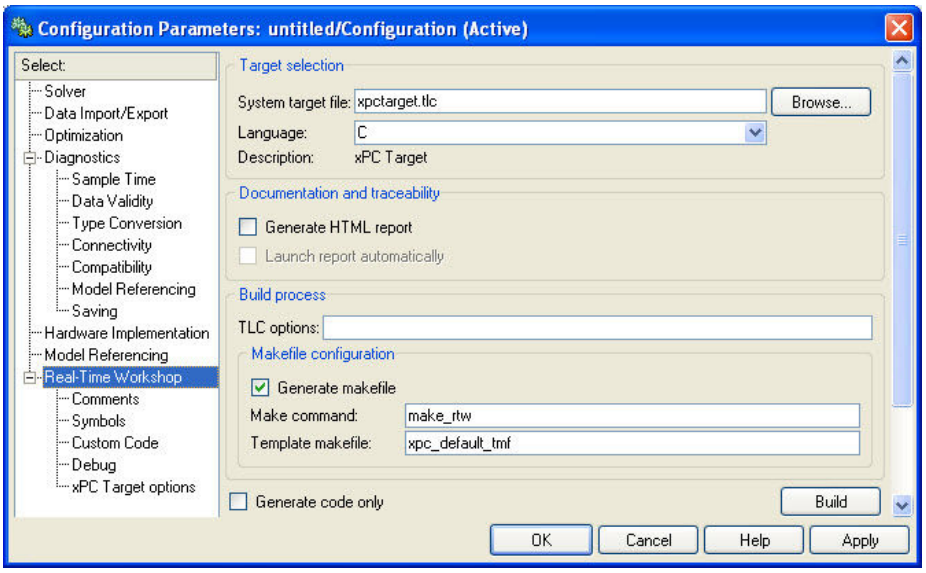
Figura 7. Parametros del modelo de *Simulink*.



Fuente: Los autores.

En la figura 8 se muestra como queda la ventana de parámetros con las modificaciones hechas en el paso 6.

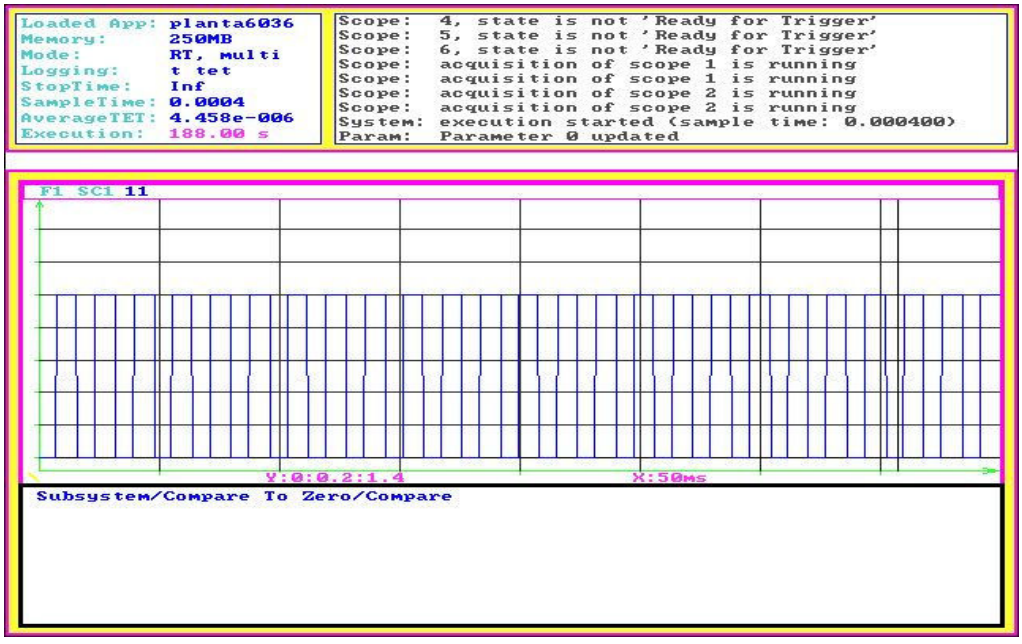
Figura 8. Parámetros de *Real-Time Workshop en Simulink*.



Fuente: Los autores.

En la figura 9 se muestra la señal cuadrada que se genera en el ejemplo vista en el osciloscopio (*scope*) de *xPC Target*.

Figura 9. Señal cuadrada en el *scope (xPC Target).*



Fuente: Los autores.

El modelo se puede compilar y ejecutar cuando haya sido configurado como se describió anteriormente.

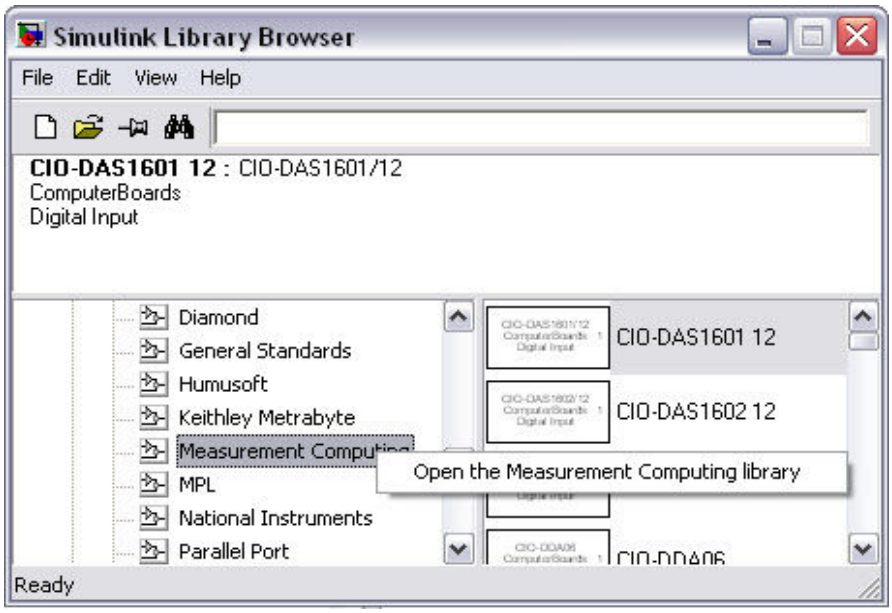
ANEXO H

A continuación se describen los pasos necesarios para realizar los drivers de la tarjeta de adquisición de datos PCI-DAS6036.

1. ENTRADA DIGITAL (DIGITAL INPUT)

Para diseñar el driver de la entrada digital de la tarjeta de adquisición de datos PCI-DAS6036 inicialmente se abre *Matlab* y *Simulink*, luego hace un *click* en la librería de xPC-TARGET y se identifican los enlaces donde están los *drivers* de entradas digitales de la empresa Measurement Computing. (Ver figura 1).

Figura 1. Librería de Simulink

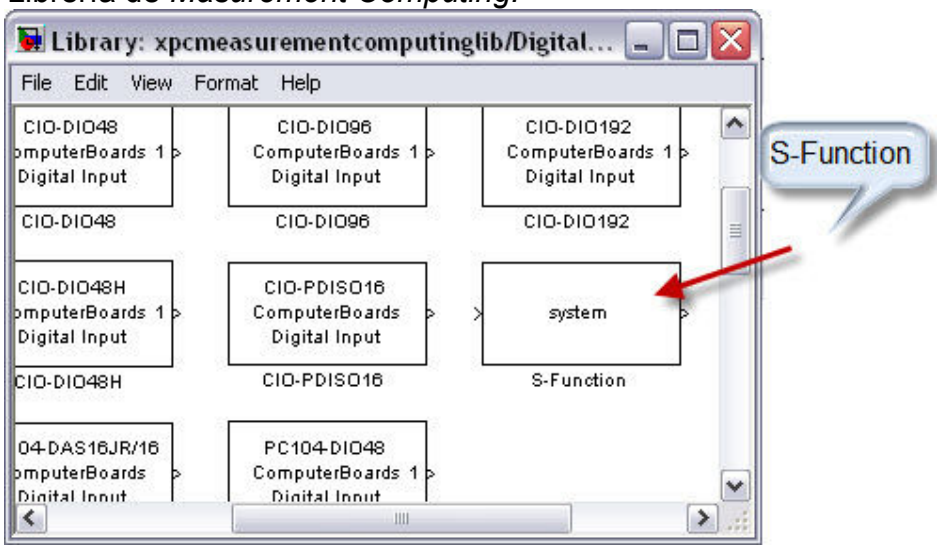


Fuente. Los Autores

En la librería de *Measurement Computing* (ver figura 2) se encuentran varios drivers de entradas digitales producidas por esta empresa.

Posteriormente se agrega una *S-Function* que se encuentra en la librería Simulink/User-Defined Functions.

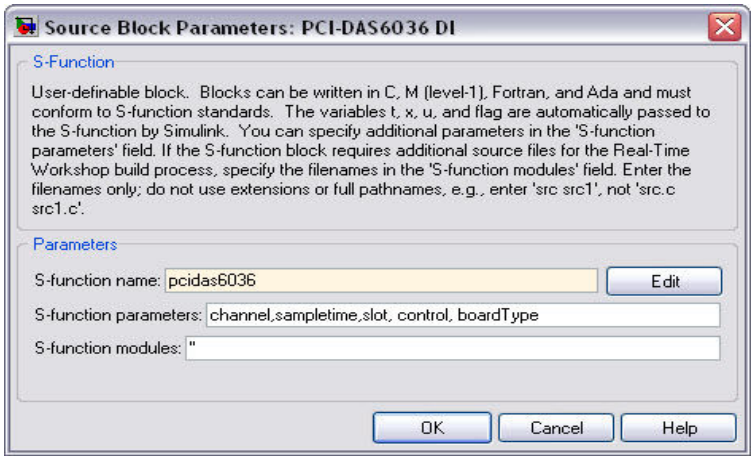
Figura2. Librería de *Masurement Computing.*



Fuente. Los Autores

Luego de crear la *S-Function* hace doble *click* sobre la *S-Function* y se escribe el nombre y los parámetros requeridos por el driver (ver figura 3).

Figura 3. Parámetros de la S-Function.

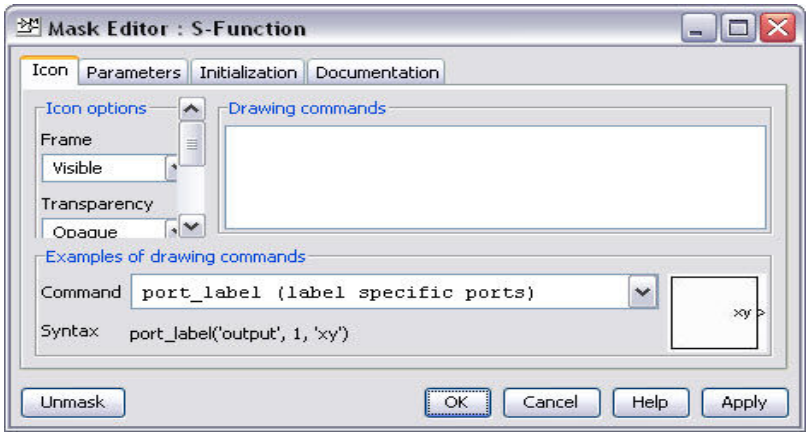


Fuente. Los Autores

El archivo *pcidas6036* es el nombre del archivo dado para el driver.

Luego de agregarse los parámetros del driver (*channel, sampletime, slot, control, boardType*) se hace un *click* derecho en la *S-Function* y se escoge la opción **Mask *S-Function*** donde aparece el cuadro de dialogo que se muestra en la figura 4.

Figura 4. Edición de la máscara de la S-Function.

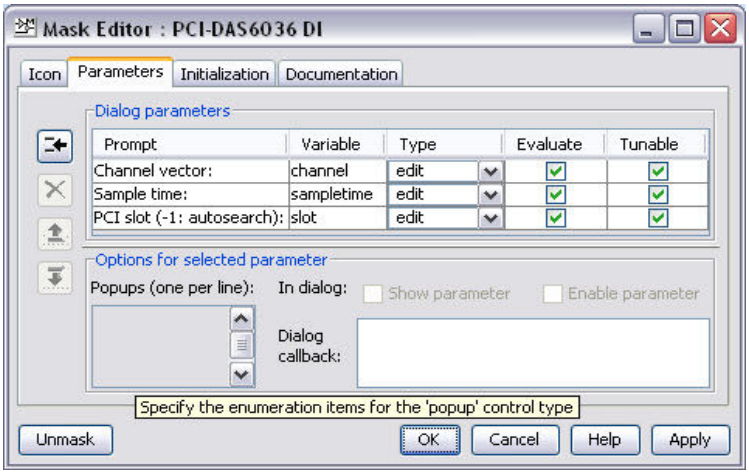


Fuente. Los Autores

En el área de *Drawing Commands* de la figura 4 se escribe el siguiente código:

```
disp('PCI-DAS6036\nMeasurement\n                Computing\nDigital Input');port_label('output',1,'1');
```

Figura 5. Parámetros de la Mascara

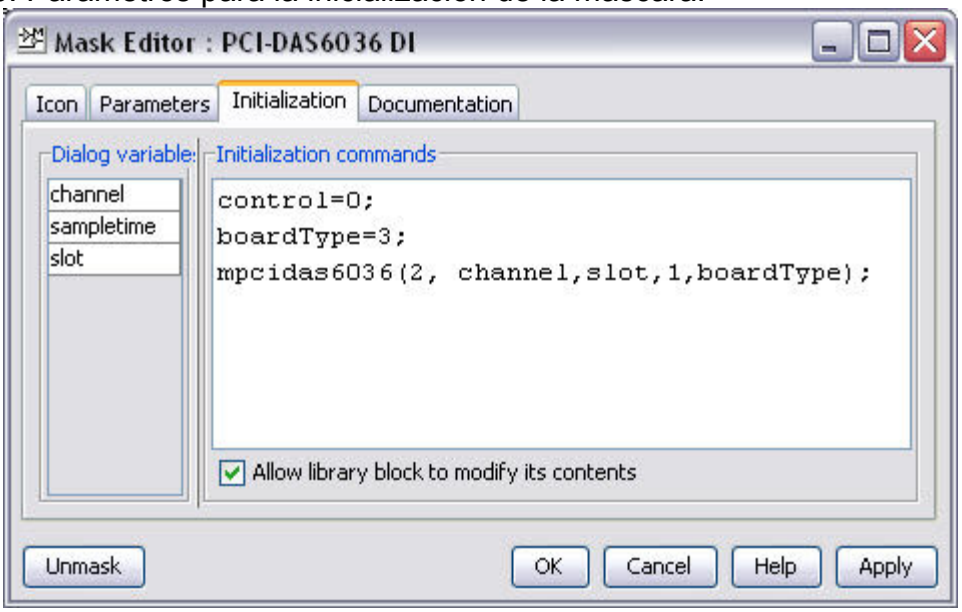


Fuente. Los Autores

En el área donde se establecen los parámetros de la máscara (ver figura 5) se agregan los *Prompts* que corresponden a las deferentes variables que se pueden sintonizar cuando se realiza la simulación.

En el enlace de inicialización (ver figura 6) se anexa el siguiente archivo *mpcidas6036* el cual debe guardarse en la misma carpeta que se guarda el archivo *pcidas6036*. El enlace de inicialización se debe configurar como se muestra en la figura 6 ya que son datos necesarios para la inicialización de algunas variables del driver.

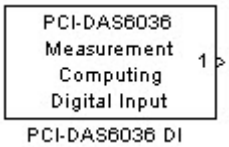
Figura 6. Parámetros para la inicialización de la máscara.



Fuente. Los Autores

Luego de esto se modifica el nombre de la *S-Function* para asignarle el nombre de la tarjeta que se está utilizando (ver figura 7) se hace un *click* en la parte inferior de la *S-Function* y se cambia el nombre.

Figura 7. *S-Function* PCI-DAS6036



Fuente. Los Autores

Al finalizar el anterior procedimiento se ejecuta la orden (rehash toolbox) que actualiza la librería de *Measurement Computing.*

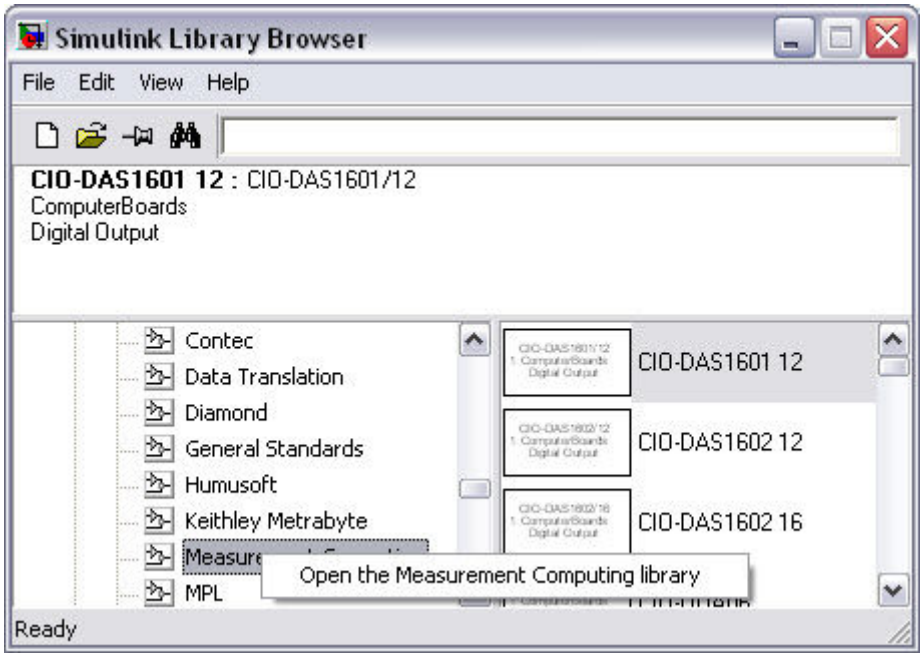
El driver para la entrada digital se puede encontrar en: Matlab/Simulink/Xpctarget/Digital Input/Measurement Computing. Por último se modifica el tiempo de muestreo que se requiere para la simulación, el campo de PCI slot por defecto se dejara -1 ya que el *driver* automáticamente reconoce el slot del PC donde se encuentra conectada la tarjeta de adquisición de datos.

2. SALIDA DIGITAL (DIGITAL OUTPUT)

La salida digital es muy parecida a la entrada digital, ya que solo se modifican algunos parámetros similares a los que se configuraron para la entrada digital.

Para diseñar el driver para la salida digital de la tarjeta de adquisición de datos PCI-DAS6036 inicialmente se abre *Matlab* y *Simulink*, luego hace un *click* en la librería de xPC-TARGET y se identifican los enlaces donde se encuentran los drivers de salidas digitales de la empresa *Measurement Computing* (ver figura 8).

Figura 8. Librería de Simulink

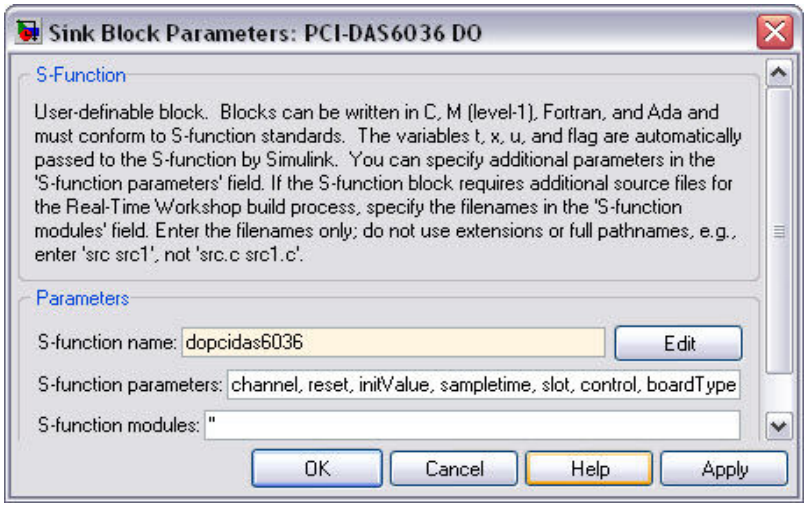


Fuente. Los Autores

En la librería de *Measurement Computing* se encuentran los drivers de salidas digitales producidos por esta empresa. Posteriormente se agrega una *S-Function* que se encuentra en Simulink/User-Defined Functions .

Luego de crear la *S-Function* hace doble *click* y se escribe lo siguiente (ver figura 9).

Figura 9. Parámetros de la S-Function



Fuente. Los Autores

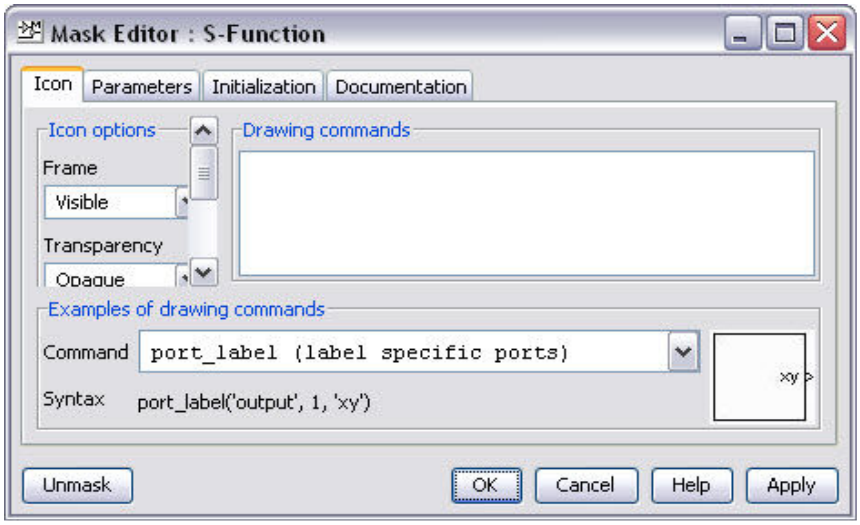
El archivo ***dopcidas6036*** es el nombre del archivo dado para el driver.

Luego de agregados los parámetros se hace un *click* derecho en la *S-Function* y se escoge la opción **Mask *S-Function*** donde aparece el cuadro de dialogo que se muestra en la figura 10.

En el área de *Drawing Commands* se escribe el siguiente código:

```
disp('PCI-DAS6036\nMeasurement\n Computing\nDigital Output');port_label('input',1,'1');
```

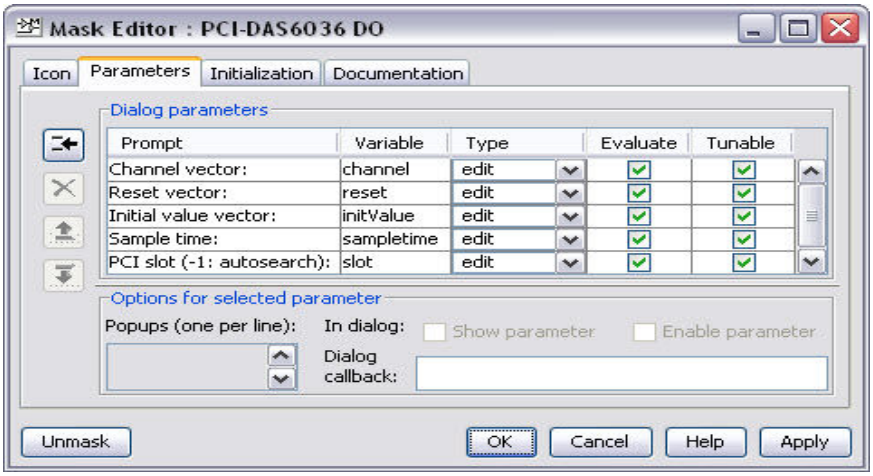
Figura 10. Edición de la máscara de la S-Function



Fuente. Los Autores

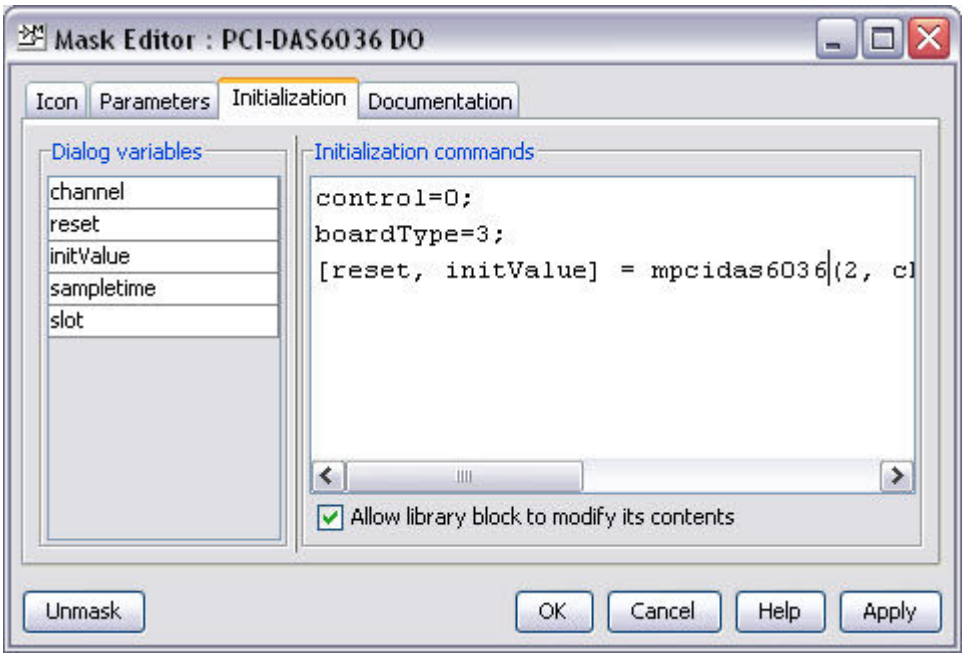
En el enlace de parámetros se escriben las variables encontradas en la figura 11 para poder configurar la salida digital.

Figura 11. Parámetros de la máscara.



Fuente. Los Autores

Figura 12. Inicialización de la mascara



Fuente. Los Autores

Los parámetros de inicialización (ver figura 12) se requieren por la *S-Function* cada vez que se utilice por lo cual se deben configurar tal y como se muestra a continuación.

control=0;

boardType=3;

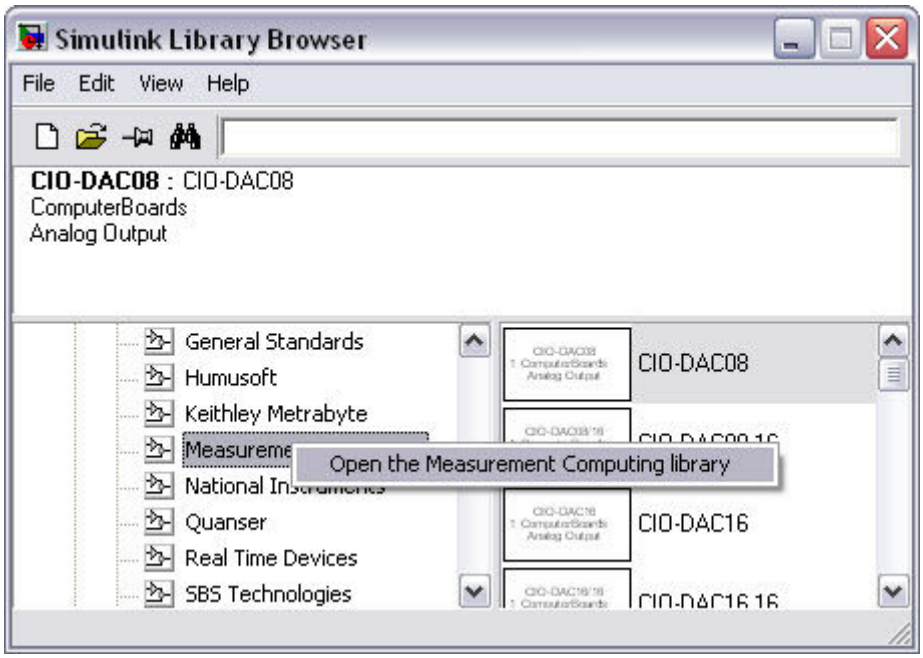
[reset, initValue] = mpcidas6036(2, channel, slot, 2, boardType, reset, initValue);

Por último se ejecuta el comando *Rehash toolbox* en el área de escritura de Matlab, quedando configurada la salida digital de la tarjeta de adquisición de datos.

3. SALIDA ANALOGA (Conversión D/A)

Para diseñar el driver de la salida análoga de la tarjeta de adquisición de datos PCI-DAS6036 inicialmente se abre *Matlab* y *Simulink*, luego hace un *click* en la librería de xPC-TARGET y se identifican los enlaces donde se encuentran los *drivers* de salidas analógicas de *Measurement Computing* (ver figura 13).

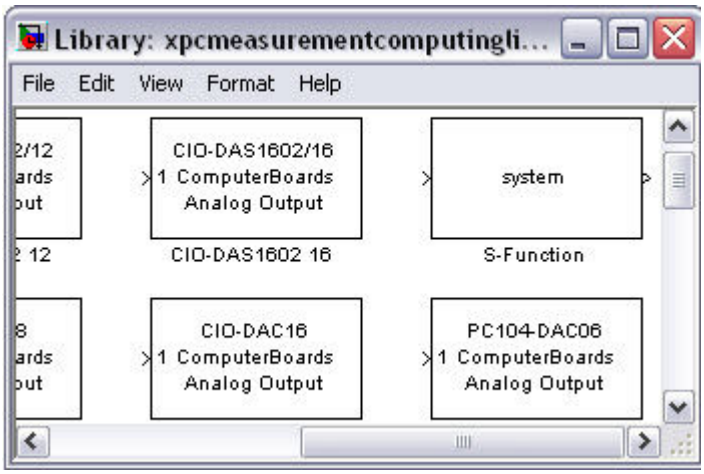
Figura 13. Librería de Simulink.



Fuente. Los Autores

En esta librería *Measurement Computing* se encuentran varios *drivers* de salidas analógicas producidas por esta empresa. Posteriormente se agrega una *S-Function* que se encuentra en la librería Simulink/User-Defined Functions (ver figura 14).

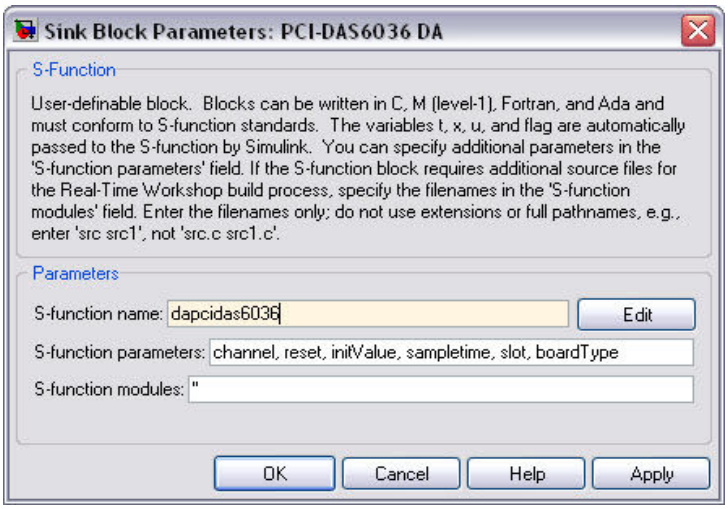
Figura 14. Librería de *Measurement Computing*.



Fuente. Los Autores

Luego de crear la *S-Function* hace doble *click* sobre esta y se escriben el nombre y los parámetros (ver figura 15) requeridos por el *driver*.

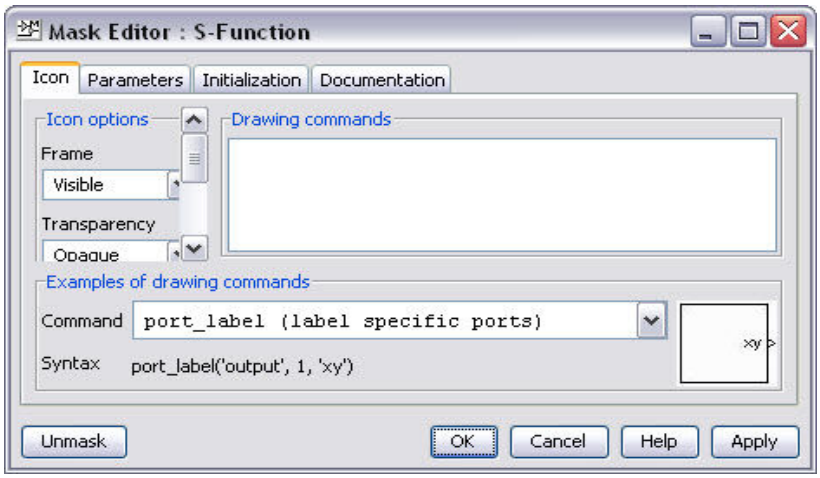
Figura 15. Parámetros de la *S-Function*.



Fuente. Los Autores

El archivo *dapcidas6036* es el nombre del archivo dado para el *driver*.

Figura 16. Configuración de la máscara.



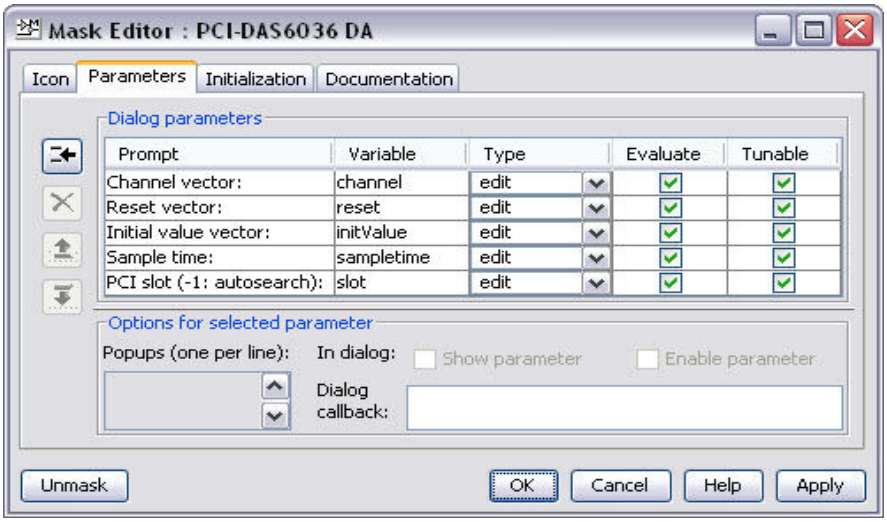
Fuente. Los Autores

Luego de agregarse los parámetros del driver se hace un *click* derecho en la *S-Function* y se escoge la opción ***Mask S-Function*** donde aparece el cuadro de dialogo que se muestra en la figura 16.

En el área de *Drawing Commands* se escribe el siguiente código:

```
disp('PCI-DAS6036\nMeasurement\nComputing\nAnalog Output');port_label('input',1,'1');
```

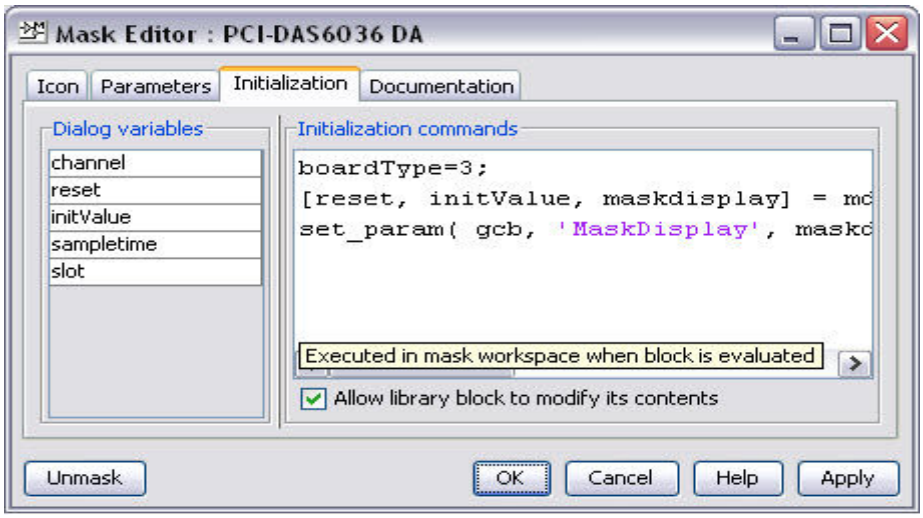
Figura 17. Parámetros de la máscara.



Fuente. Los Autores

En el área donde se establecen los parámetros de la máscara, se agregan las variables que se muestran en la figura 17, que serán variables y se pueden sintonizar cuando se realiza la simulación.

Figura 18. Inicialización de la máscara.



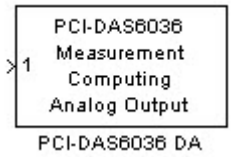
Fuente. Los Autores

Para el enlace de inicialización (ver figura 18) se anexa el siguiente archivo *mdapcidas6036* el cual debe quedar guardado guardarse en la misma carpeta que se guarda el archivo *dapcidas6036*. El enlace de inicialización debe configurarse de la siguiente forma.

```
boardType=3;
[reset, initValue, maskdisplay] = mdapcidas6036(2, channel, reset, initValue,
slot, boardType);
set_param( gcb, 'MaskDisplay', maskdisplay );
```

Por último se modifica el nombre de la *S-Function* para asignarle el nombre de la tarjeta que se está utilizando, se hace *click* en la parte inferior de la *S-Function* y se cambia el nombre que trae por defecto por el nombre de la tarjeta de adquisición de datos que se disponga tal y como se muestra en la figura 19.

Figura 19. *S-Function* PCI-DAS6036.



Fuente. Los Autores

Al finalizar el anterior procedimiento se ejecuta la orden (rehash tollbox) que actualiza la librería de *Measurement Computing*.

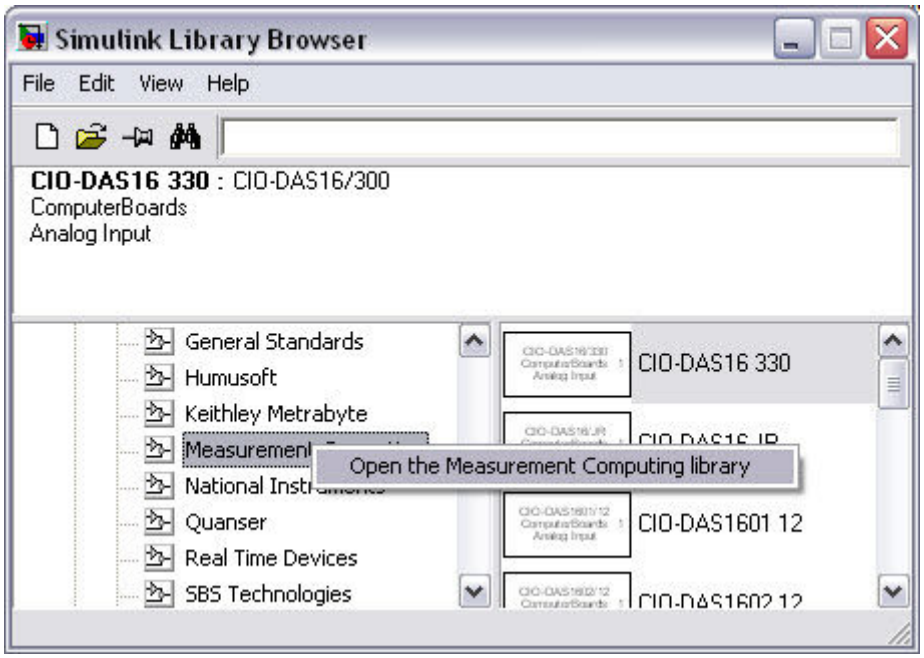
El driver para la salida análoga de la tarjeta PCI-DAS6036 se encuentra en: Matlab/Simulink/Xpctarget/D/A/*Measurement Computing*. Por último se modifica el tiempo de muestreo que se requiere para la simulación, el canal que se desee utilizar y el campo de PCI slot que por defecto se dejara en -1

ya que el driver automáticamente reconoce el slot del PC donde se encuentra conectada la tarjeta de adquisición de datos.

4. ENTRADA ANALOGA conversión(A/D)

Para diseñar el driver de la entrada análoga de la tarjeta de adquisición de datos PCI-DAS6036 inicialmente se abre Matlab y Simulink, luego hace *click* en la librería de *XPC-TARGET* y se identifica el enlace donde se encuentran los *drivers* de entradas análogas de *Measurement Computing* (ver figura 20).

Figura 20. Librería de Simulink.

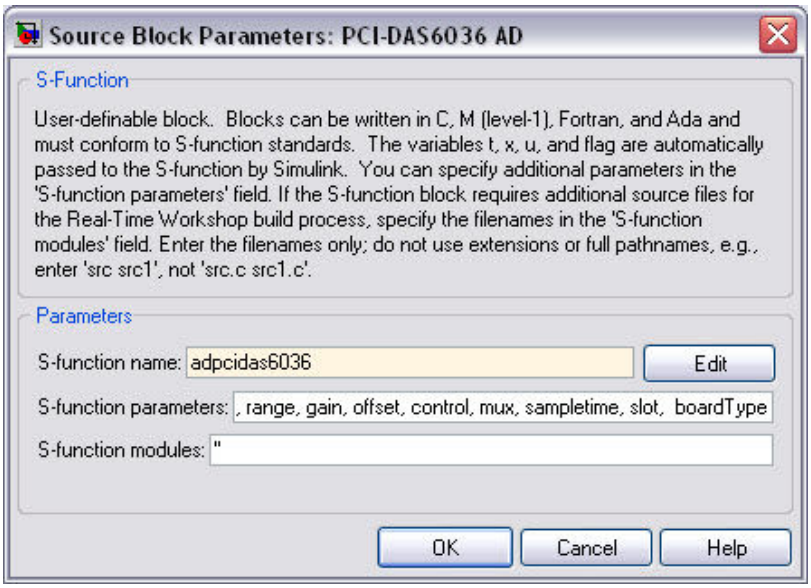


Fuente. Los Autores.

En esta librería *Measurement Computing* se encuentran varios *drivers* de entradas analógicas producidas por esta empresa. Posterirmente se agrega una *S-Function* que se encuentra en la librería Simulink/User-Defined Functions (ver figura 21).

El archivo ***adpcidas6036*** es el nombre del archivo dado para el driver.

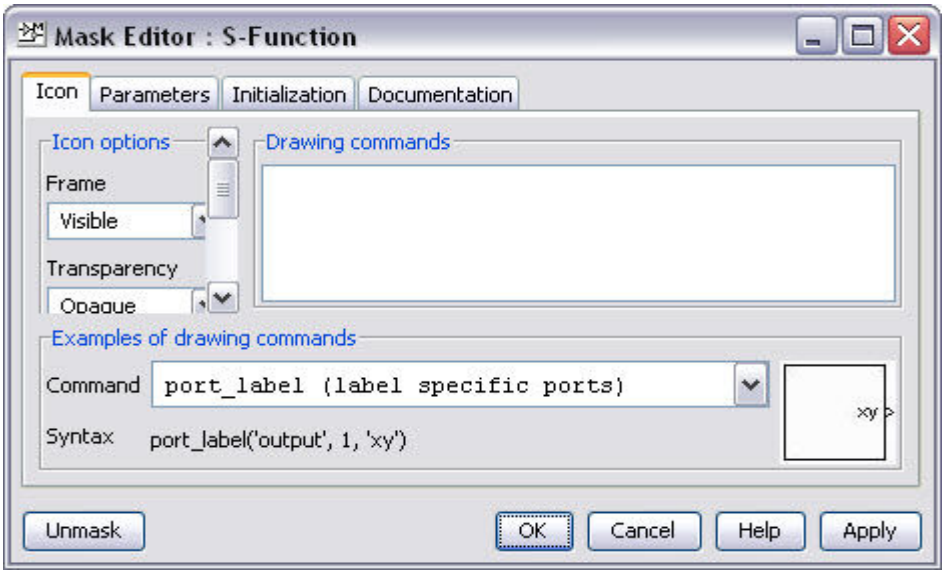
Figura 21. Configuración de la *S-Function*



Fuente. Los Autores

Se hace un *click* derecho en la *S-Function* y se escoge la opción **Mask *S-Function*** donde aparece el cuadro de dialogo que se muestra en la figura 22.

Figura 22. Configuración de la máscara.



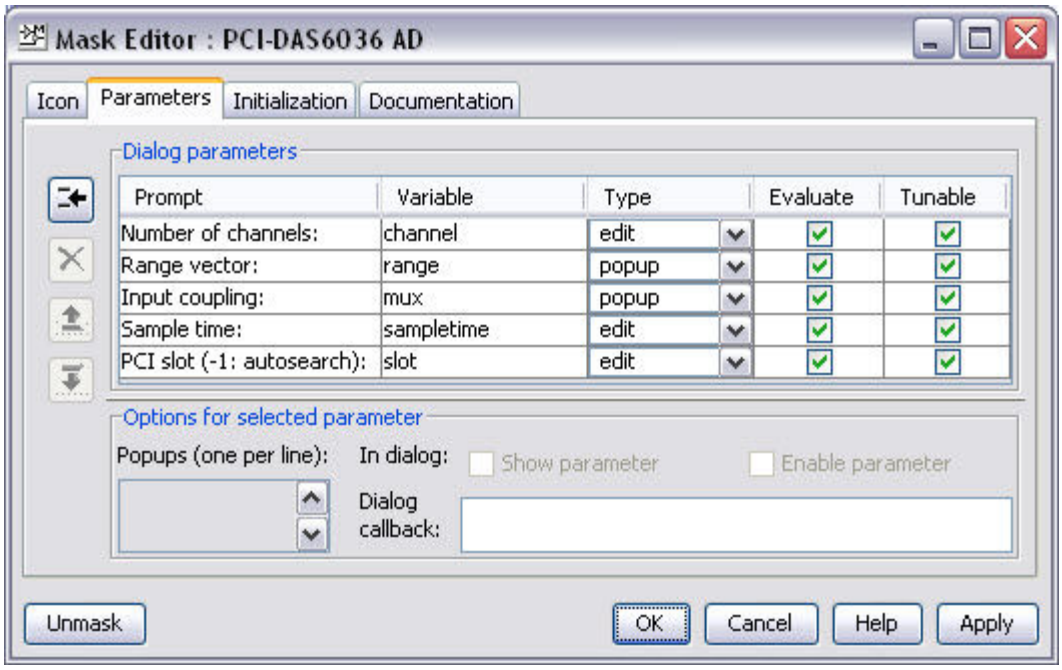
Fuente. Los Autores.

En el área de *Drawing Commands* se escribe el siguiente código:

```
disp('PCI-DAS6036\nMeasurement\nComputing\nAnalog Input');port_label('output',1,'1');
```

En el área donde se establecen los parámetros de la mascara (ver figra 22) corresponden a las diferentes variables que se pueden sintonizar cuando se realiza la simulación.

Figura 23. Configuración de parámetros de la máscara.



Fuente. Los Autores.

Los parámetros de inicialización son requeridos por la *S-Function* cada vez que se utilice por lo cual se deben configurar tal y como se muestra a continuación.

```
boardType=2;
[gain, offset, control, maskdisplay]=madpcidas6036(2, channel, range, mux, slot, boardType);
```

```
set_param( gcb, 'MaskDisplay', maskdisplay );
```

Por último se ejecuta el comando Rehash toolbox en la ventana de MATLAB , quedando configurada la entrada análoga.