SISTEMA DIDÁCTICO PARA LA IMPLEMENTACIÓN DE CONTROLADORES DIGITALES

Yamir Hernando Bolaños Muñoz
Luisa Fernanda Pineda Calvache

Director
Mag. Víctor Hugo Mosquera

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Electrónica, Instrumentación y Control
Ingeniería en Automática Industrial
Popayán, Agosto de 2011

SISTEMA DIDÁCTICO PARA LA IMPLEMENTACIÓN DE CONTROLADORES DIGITALES

Yamir Hernando Bolaños Muñoz
Luisa Fernanda Pineda Calvache

Monografía presentada como requisito parcial para optar por el título de Ingenieros en Automática Industrial

Director
Mag. Víctor Hugo Mosquera

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Electrónica, Instrumentación y Control
Ingeniería en Automática Industrial
Popayán, Agosto de 2011

Nota de aceptación: _____
_____
_____
_____
_____
_____

_____
Carlos Felipe Rengifo Rodas

_____
Carlos Felipe Rengifo Rodas

_____
Vladimir Trujillo Arias

Popayán, Agosto de 2011

AGRADECIMIENTOS

A Dios por darnos sabiduría y las capacidades para desarrollar este trabajo.

A nuestros padres por su apoyo y entrega incondicional.

A nuestro director Víctor Hugo Mosquera por su colaboración y tiempo dedicado a este proyecto.

Al Magíster Juan Fernando Flórez y al Doctor Carlos Felipe Rengifo, cuyos consejos e interés fueron un aporte significativo para llevar a cabo este proyecto.

A los miembros del Departamento de Electrónica, Instrumentación y Control por su profesionalismo y cumplimiento en la labor de docentes.

A nuestros compañeros de estudio por sus consejos, apoyo y amistad.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ABREVIACIONES

A Amperio
AC Alternating Current, Corriente Alterna
AD Análogo Digital
DIP Dual in line package, Empaquetado de doble línea
DLL Dynamic Link Library, biblioteca de enlace dinámico
EP End Point
Gbps Giga bits por segundo
GNU General Public License, Licencia Pública General
HID Human Interface Device, dispositivo de Interfaz Humana
I2C Intger Integrated Circuits
ICSP In Circuit Serial Programming, Programación Serial en Circuito
IDE Integrated Development Environment, Entorno de desarrollo integrado
KB Kilo Byte
KB/s Kilo Byte por segundo
Khz Kilohercio
LCD Liquid Crystal Display, Pantalla de cristal liquido
LED Light Emitting Diode, Diodo emisor de luz
mA Miliamperio
Mbps Mega bits por segundo
pA Pico Amperio
PC Personal Computer, Computador personal
PCB Printed Circuit Board, Tarjeta de circuito Impreso
PIC Peripheral Interface Controller, circuito integrado programable, hace referencia solo a dispositivos de la empresa Microchip.
PID En terminología de control, Proporcional Integra Derivativo
PWM Pulse Width Modulation, Modulacion por ancho de pulso
RAM Random Access Memory, Memoria de acceso aleatorio
RC Resistencia Capacitor
RS - 232 Recommended Standard, es un protocolo de comunicación serial
SPI Serial Peripheral Interface, periférico de interfaz serial
Tohm Tera ohmio
USB Universal Serial Bus, Bus serial universal
VCA Voltaje corriente alterna
VCC Voltaje colector colector
VDC Voltaje corriente Directa
VDD Voltaje Drenador Drenador
VID & PID Vendor Identification, identificador de vendedor & Product Identification, Identificación del producto.
VPP Voltaje de Programación
VSS Voltaje surtidor surtidor
W vatios

INTRODUCCIÓN

Los sistemas didácticos son una herramienta potente que facilita los procesos de aprendizaje en los estudiantes, del mismo modo que despiertan su interés en las temáticas relacionadas al control, ya que mediante las experiencias prácticas, se aplica y refuerzan los conocimientos teóricos adquiridos.

Mediante la realización de este proyecto se busca construir un sistema que permita la implementación de controladores digitales y que a su vez se adapte fácilmente a las necesidades de diferentes áreas del Departamento de Instrumentación y Control de la Facultad de Ingeniería Electrónica y Telecomunicaciones de la Universidad Del Cauca.

Para construir el sistema didáctico para la implementación de controladores digitales se plantearon las siguientes actividades:

- Diseñar y elaborar el hardware de la tarjeta principal para implementación de controladores digitales.
- Diseñar y elaborar plantas experimentales a controlar.
- Desarrollar del software para la intercomunicación del sistema didáctico experimental.
- Modelar plantas experimentales a controlar.
- Validar el sistema.

Las anteriores actividades serán presentadas en cada capítulo del presente proyecto.

Antecedentes de algunos sistemas didácticos de control

Diseño, construcción y control de un sistema de levitación neumática.

Proyecto en el cual se construyó un prototipo (sistema de levitación neumática), donde se emplea aire sobre un material con el fin de contrarrestar la fuerza gravitatoria, de modo que el objeto se mantiene estable sin necesidad de contacto físico con ninguna parte del contenedor que lo aloja.

En este proyecto el objeto se encuentra en un ambiente cerrado y en suspensión estable; de modo que la posición del mismo es controlable con lo cual se provee de una herramienta académica, para el programa de Automática Industrial, donde se evalúan estrategias de control en malla abierta y en malla cerrada por medio de la implementación de un PID serie, el cual es sintonizado mediante ajuste manual.

Figura 1. Sistema de levitación neumática en funcionamiento.

Sistema de prototipado rápido de control para el módulo de prácticas MIC955 de la empresa Feedback

Proyecto en el cual se desarrolló de un Sistema de Prototipado Rápido de Control (RCP) para el módulo MIC955 de la empresa Feedback, el sistema de RCP permite que desarrollar diversas prácticas de control de temperatura en tiempo real, mediante el uso de diferentes de tipos de controladores, bajo el sistema operativo Linux (GALÍNDEZ & JARAMILLO, 2011).

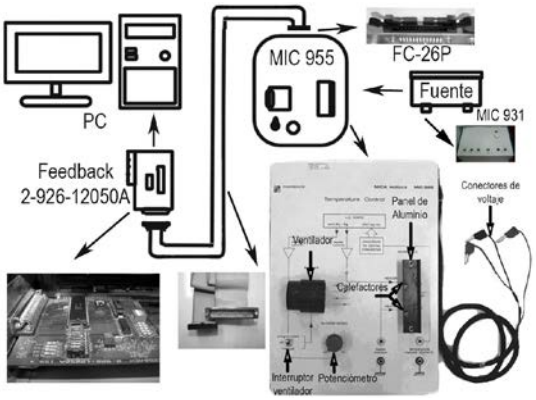


Figura 2. Módulo MIC 955

Sistema de desarrollo QIC

Es un conjunto integrado de componentes de desarrollo. Su parte hardware está constituida por QIC placa base y QIC modulo esclavo. QIC tarjeta madre, aloja el microcontrolador y los componentes asociados para la comunicación y la programación. QIC modulo esclavo alberga las interfaces y opciones de acondicionamiento de señales, que se adaptan a diferentes aplicaciones, tales como: suministro de energía y controladores de motores.

Por otro lado en su parte software cuenta con QIC comandante y QIC programador. QIC comandante es un paquete de software con su propio protocolo de comunicación para el monitoreo de la QIC, los datos recibidos desde el conversor A / D se trazan para el análisis visual. Los datos trazados se pueden guardar para el análisis utilizando MATLAB. QIC programador es la parte que

permite descargar programas a la placa base (QIC), a través del puerto serial, el cual también es empleado en la comunicación.

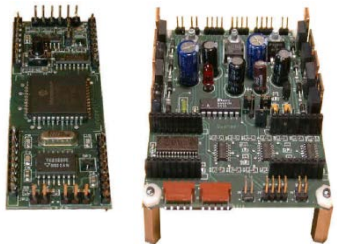


Figura 3. QIC Sistema de desarrollo.

Equipo didáctico en control de procesos de Lab-Volt, modelo 3521

Es un banco de capacitación portátil, el parámetro físico que se controla con el equipo didáctico en control de procesos es la temperatura de un radiador. Se emplea un calefactor y un ventilador para calentar y enfriar el radiador, respectivamente. La temperatura del radiador se mide con un transmisor de temperatura que se vale de un termopar para convertir dicha temperatura en una señal eléctrica proporcional. El equipo didáctico incluye fuentes de corriente continua, un controlador P.I.D., un detector de alarma y un generador de ruido.
Emplea como software de control y simulación LVPROSIM.

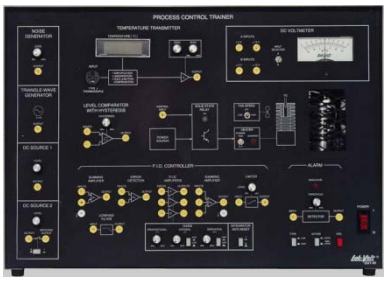


Figura 4. Equipo didáctico en control de procesos modelo 3521.

Estructura de la monografía.

El trabajo se divide en los siguientes capítulos teniendo en cuenta las actividades mencionadas anteriormente. En el capítulo 1 se expone el proceso de diseño e implementación a nivel de hardware del sistema. El capítulo 2 aborda los aspectos de diseño del software que complementa la funcionalidad del sistema. Posteriormente en el capítulo 3 se presenta el diseño de una interfaz de usuario básica para el monitoreo de las variables del sistema. Luego en el capítulo 4 se obtienen los modelos de las plantas para proceder con la implementación de controladores que permitirán realizar la validación del sistema. Finalmente se presentan las conclusiones del proyecto.

RESUMEN

En la realización del proyecto se llevaron a cabo actividades de diseño tanto a nivel hardware como software para construir un sistema didáctico que sirve de apoyo en procesos de enseñanza-aprendizaje en diferentes áreas de interés en el programa de Ingeniería en Automática Industrial del departamento de Electrónica y Telecomunicaciones de la Universidad del Cauca.

El sistema desarrollado se compone de tres módulos hardware principales, una tarjeta planta de temperatura, una tarjeta planta de circuitos RC y por ultimo una tarjeta principal que se conecta a las plantas y al computador por medio del bus USB, de modo que se pueden desarrollar acciones como: adquisición y monitoreo de variables y control sobre las variables controladas en las plantas. Este desarrollo hardware se complementa con componentes software que hacen al sistema flexible, intuitivo y fácil de usar.

En los siguientes capítulos se presenta el proceso de diseño, implementación y validación del sistema, así como los resultados obtenidos durante todo el proceso de desarrollo. Se inicia con un capítulo dedicado al hardware, seguido por dos capítulos referentes al diseño e implementación software y finalmente un capítulo dedicado a la validación del sistema mediante el desarrollo de ciertas prácticas que demuestran la versatilidad del sistema.

DESCRIPCIÓN DEL SISTEMA DIDÁCTICO PARA LA IMPLEMENTACIÓN DE CONTROLADORES DIGITALES

El presente proyecto presenta una herramienta didáctica orientada al apoyo en los procesos de enseñanza-aprendizaje que se desarrollan en el programa de Ingeniería en Automática Industrial de la universidad del Cauca, este es un desarrollo con componentes tanto a nivel de hardware como software que permiten al usuario interactuar y elaborar prácticas relacionadas con áreas como: control digital, instrumentación, modelado e identificación de procesos.

El diseño del sistema se basa en una estructura modular y flexible, de modo que el usuario puede intercambiar módulos para realizar diferentes tipos de prácticas, además se agregan características como portabilidad y bajo costo en relación a sistemas comerciales, por esta razón se diseñan tres tarjetas tipo hardware las cuales son:

Tarjeta principal de control: será la encargada de establecer la comunicación con los diferentes módulos del sistema del mismo modo debe permitir la comunicación con un computador desde el cual se puedan realizar diferentes acciones como: adquisición de datos, monitoreo de variables, análisis de datos mediante métodos gráficos. Además la tarjeta principal ofrece la posibilidad de ser programada mediante código generado en lenguaje C para adaptarla según diferentes necesidades, esta tarjeta permite realizar prácticas de control por PC donde el controlador está en el computador y a través de la tarjeta se envía la señal de control a la planta objeto de control, también permite implementar esquemas de control embebido, en este método el controlador se lleva a cabo en la tarjeta principal y es donde se realizan los diferentes cálculos para generar una señal que permita controlar un sistema específico (planta).

Planta de temperatura: sistema electrónico con componentes como: un elemento calefactor, etapa de potencia (actuador) que permite controlar la carga eléctrica aplicada al dispositivo calefactor, sensor de temperatura para medir la variable a controlar, sistema para generación de disturbios en la planta. Esta planta permite al usuario apreciar diferentes aspectos en el comportamiento de un sistema de temperatura como tiempos de estabilización, inercia del sistema y disipación de potencia entre otros.

Planta de Circuitos RC: es también un sistema electrónico muy práctico basado en mallas de resistencias y condensadores, en el cual la variable a controlar es la tensión o voltaje a la salida del circuito aplicado a una carga resistiva, el sistema es altamente flexible ya que permite ser configurado como sistemas de 1er, 2do y 3er orden.

En referencia al componente software se puede decir que hay dos partes importantes, en primera instancia está el software que permite la programación de

la tarjeta principal de modo que la tarjeta principal pueda operar con las plantas y realizar acciones sobre las diferentes señales que se generan. Luego está un software que permite el monitoreo del sistema desde un computador facilitando al usuario el análisis del comportamiento de los procesos de forma natural y de forma controlada según las técnicas de control que se implementen en las practicas.

El software para monitoreo y control de la tarjeta desde el PC está desarrollado en Matlab, dado que esta es una herramienta altamente utilizada en el programa de Ingeniería en Automática Industrial de la universidad del Cauca, razón por la cual a los usuarios del sistema les resultara cómodo y rápido familiarizarse con la utilización del sistema.

1 DISEÑO E IMPLEMENTACIÓN DEL HARDWARE DEL SISTEMA

Equation Section 1

En el presente capitulo se expone el proceso de diseño e implementación a nivel de hardware del sistema para implementación de controladores digitales, el cual se compone de:

- Una tarjeta principal para adquisición y control.
- Una tarjeta correspondiente a la planta de temperatura.
- Una tarjeta correspondiente a la planta de circuitos RC.

Los temas relacionados con implementación y desarrollo de software se trataran en detalle en el siguiente capítulo.

1.1 Diseño y Elaboración de Prototipo Hardware de Tarjeta Electrónica

En esta sección se describe el proceso de diseño y elaboración de la tarjeta principal del sistema para implementación de controladores digitales, el proceso de diseño de esta tarjeta es muy importante, debido a que dicha tarjeta debe ser altamente flexible y con varias prestaciones, de modo que permita conectar diferentes plantas con cambios mínimos en hardware y software.

Se establecen las siguientes características como base para iniciar el diseño y desarrollo de la tarjeta principal:

- Comunicación directa a un PC: Se requiere que la tarjeta se pueda comunicar con un PC para realizar diferentes acciones como: monitoreo de variables (*set point, variable controlada, variable manipulada, esfuerzo de control*), y configuración del sistema (selección de tipo planta, configuración de entradas-salidas).
- El software que se escogió para el manejo desde el PC es Matlab, se elige Matlab ya que es altamente versátil e integra muchas herramientas útiles para el diseño de controladores, análisis y modelado de los sistemas entre otras, además este es un software con el que tanto docentes como estudiantes están ampliamente familiarizados por su uso en múltiples asignaturas del programa.
- Múltiples recursos para manejo de periféricos: Se requieren recursos tales como entradas-salidas digitales, entradas- salidas analógicas, salidas PWM.
- Programación rápida: Una de las funciones de la tarjeta principal es la adquisición de las variables de las plantas, y otra muy importante es contener y ejecutar código correspondiente al control diseñado por el usuario (función en caso de control embebido), así que se requiere de un mecanismo que permita una programación y reprogramación versátil, que en lo posible no involucre el uso de herramientas adicionales como un

programador externo o acciones como retirar y reemplazar componentes de la tarjeta.

- Capacidad de procesamiento acorde con las tareas a desarrollar: La tarjeta principal del sistema deberá ejecutar múltiples tareas de procesamiento, tales como: operaciones de adquisición (comunicación con PC), procesamiento en el manejo de periféricos y procesamiento de acciones de control de las plantas en el caso de control embebido, por lo tanto se requiere que la velocidad y capacidad de procesamiento sean eficientes ante los diferentes requerimientos del sistema.

1.1.1 Selección Microcontrolador

La propuesta es desarrollar una tarjeta con base en dispositivos programables, específicamente con microcontroladores PIC de Microchip, se propone esta casa fabricante ya que es ampliamente utilizada en todo el mundo en desarrollo de proyectos con sistemas embebidos, además los estudiantes están familiarizados con su manejo.

Algunos factores relevantes para trabajar con microcontroladores de Microchip son: la amplia documentación que se encuentra sobre el manejo de estos dispositivos, además existen muchas herramientas de desarrollo como programadores, compiladores en diferentes lenguajes (C, Basic, C++), librerías para manejo de periféricos comunes como sensores, pantallas LCD, teclados, las referencias más comunes de estos microcontroladores están equipados con memoria tipo flash lo que permite reprogramar los dispositivos muchas veces (100.000 ciclos de lectura/escritura).

Se inicia el diseño con la selección del microcontrolador, para lo cual se analizan características como, numero de pines de I/O (entrada-salida), protocolos de comunicación (I2C, RS-232, RS-485, USB), módulos internos (PWM, Timers, interrupciones), capacidad de memoria de programa, entre las más importantes.

En la selección no se incluyen microcontroladores de tipo montaje superficial.

Tabla 1.1. Selección microcontrolador

Características	**Referencia microcontrolador**		
	PIC 18F452	**PIC 18F2550**	**PIC 18F4550**
Empaquetado	DIP-40	DIP-28	DIP-40
Pines I/O	33	24	35
Comunicación	RS-232,RS-485, I2C, PSP, 3Wire SPI	USB2.0, RS-232,RS-485, I2C, PSP, 3Wire SPI	USB2.0, RS-232,RS-485, I2C, PSP, 3Wire SPI
Salidas PWM	2 (CCP1,CCP2) 156Khz a 8Bits 38Khz a 10Bits	1, 2 o 4 según el modo.	1, 2 o 4 según el modo.

Frecuencia Max. Operación	40Mhz	48Mhz	48Mhz
Entradas Analógicas	8	10	13
Memoria de Programa(Flash)	32 KBytes 100.000 ciclos W/R	32 KBytes 100.000 ciclos W/R	32 KBytes 100.000 ciclos W/R
Memoria de datos (RAM)	1536 Bytes	2048 Bytes	2048 Bytes
Costo (Pesos)	14.500	16.800	17.980

Analizando la tabla anterior se opta por utilizar un microcontrolador con protocolo USB, los otros posibles protocolos (específicamente RS-232 y el RS-485) se descartan por las siguientes razones:

El protocolo USB puede ser más rápido en transmisión que un puerto serie común (hasta 20 veces más rápido), el protocolo USB puede operar en dos modos de velocidad LowSpeed (1.5Mbps) y HigSpeed (12Mbps).
Tanto el bus USB como su respectivo Puerto USB son estándar (Cables genéricos y de fácil consecución) y actualmente se encuentran integrados en la gran mayoría de computadores, en cambio puertos como el paralelo y el serial se están descontinuando paulatinamente.

El puerto USB tiene alimentación propia, dos de sus 4 hilos proveen un voltaje de 5VDC con una corriente máxima de 500mA, en cambio para operar con el puerto serial se requiere de fuente externa. El protocolo USB permite transmisión de datos por paquetes de forma más eficiente que el protocolo serial en el que se hace mediante elaboradas rutinas por software. Se cuenta con conocimientos y experiencias previas que permiten afirmar la utilidad y versatilidad del uso del protocolo de comunicación USB.

Siguiendo con el análisis por características se descarta el microcontrolador 18F452 por el tipo de comunicación, queda entonces elegir entre los dos restantes, dado que tanto el PIC 18F2550 como el PIC18F4550, cumplen con las características necesarias para el proyecto y la única diferencia relevante radica en el número de pines de I/O (entrada-salida), se plantea la selección según esta característica, haciendo una estimación de la cantidad de pines, necesarios para la implementación del proyecto, así:

Pines de I/O estimados

Tabla 1.2. Pines de I/O estimados

Uso	Tipo	Cantidad
Salidas PWM	Salida	2
Salidas discretas propósito general	Salida	8
Entradas discretas propósito general	Entrada	6

Entradas analógicas	Entrada	4
Detección planta Conectada	Entrada	3
Señal de reset	Entrada	1
Conexión cristal externo	Entrada-Salida	2
TOTAL		26

Según la tabla anterior el candidato elegido es el microcontrolador PIC 18F4550, ya que cuenta con los recursos suficientes para iniciar el diseño de la tarjeta principal.

En principio se hace un diseño con lo básico que debe tener la tarjeta, conexión USB, cristal externo, pulsador de *reset* y un indicador tipo dual LED para el estado de la conexión USB, la Figura 1.1muestra el diagrama esquemático del diseño propuesto:

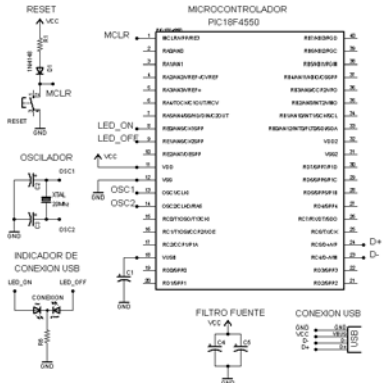


Figura 1.1. Configuración básica de microcontrolador 18F4550 para el proyecto.

1.1.2 Visualización local de variables mediante pantalla LCD 2x16

Se opta por agregar una pantalla LCD de 2 filas × 16 columnas, como medio de visualización local, este periférico es de mucha utilidad ya que:

Permite visualizar variables y estados que ayuden en el proceso de diseño y depuración de código, esto representa una ventaja ya que la visualización de variables extensas como flotantes y enteros 16 bits se puede hacer de forma directa en la pantalla LCD mientras que para transmitir este tipo de variables por el bus USB estas deben ser segmentadas en bytes (elementos de 8 bits).
Permite verificar los valores enviados al PC por medio del bus USB.

En caso de operar sin PC, la pantalla LCD es una herramienta muy útil para monitorear el estado de las variables.
Permite desplegar mensajes para facilitar el manejo del sistema tales como: estados del sistema, estado de las plantas, conexiones con PC y plantas.

Cabe recordar que lo referente a tratamiento de la información (variables, método de transmisión PC-Tarjeta principal, software en general) será abordado en detalle en el capítulo 2 intercomunicación a nivel software del sistema.

Para utilizar la menor cantidad de pines de I/O del microcontrolador se utiliza una conexión a 6 hilos, 4 hilos (4 bits) para datos y 2 hilos (2 bits) para control de la pantalla.

La Figura 1.2, representa el diagrama esquemático de la conexión de la pantalla LCD al microcontrolador.

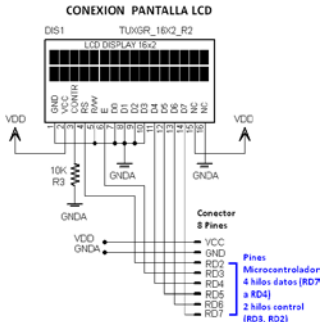


Figura 1.2. Diagrama esquemático pantalla.

Para conectar la pantalla LCD a la tarjeta principal se requiere de ocho (8) hilos en total, seis (6) de I/O y dos (2) para la alimentación de la pantalla, debido a que el tipo de pantalla que se desea utilizar trae 16 pines (estándar) se diseña un PCB donde se conecta la pantalla y tiene como salida los 8 hilos necesarios para la conexión con la tarjeta principal, el PCB se muestra más adelante en la sección “Diagrama Esquemático y PCB tarjeta principal”.

1.1.3 Programación ICSP y Bootloader USB de Microchip

Uno de los requerimientos para el diseño de la tarjeta principal es que tenga versatilidad en la programación y reprogramación, para cumplir con este requerimiento y luego de investigar sobre métodos de programación (embeber código en el microcontrolador) de los microcontroladores PIC de Microchip se decide implementar los siguientes dos métodos:

- Implementar un conector para programación en circuito ICSP (Programación Serial En Circuito), en modo alto voltaje.
- Implementar un bootloader o cargador de arranque.

A continuación se describe en detalle cada uno de los anteriores métodos que facilitaran la programación de la tarjeta principal.

1.1.3.1 Programación en modo ICSP (In Circuit Serial Programming)

Para embeber un *firmware* (código maquina) en un microcontrolador generalmente se hace uso de herramientas tipo hardware especialmente creadas para esta tarea, a estas herramientas se les conoce como programadores hardware, la función que estos cumplen es establecer comunicación con el PC donde se ha desarrollado y depurado el *firmware*, al mismo tiempo se comunica con el microcontrolador y hace que este ingrese en modo programación para así poder transferir el *firmware* desde el PC hacia el microcontrolador y finalmente grabarlo en su memoria de programa (memoria flash), la Figura 1.3 muestra el proceso.

Figura 1.3. Método de programación hardware de microcontrolador.

Una utilidad de los programadores hardware es la programación en circuito ICSP (*programación serial en circuito*) este es un método que permite programar el microcontrolador directamente en la tarjeta de aplicación sin necesidad de retirarlo de la misma, esto representa una gran ventaja ya que ahorra tiempo en el proceso de programación y además evita posibles daños al microcontrolador por manipulación indebida (quebrar pines o colocación incorrecta del circuito integrado). Para programar en circuito se necesita tener acceso a 5 pines indispensables, las señales de estos pines son:

- VPP: Señal de voltaje que permite ingresar en modo programación en el microcontrolador.
- VDD: Señal de alimentación +5VDC
- GND: Tierra
- PGD: Señal para transmisión de datos en el proceso de programación.
- PGC: Señal de reloj para sincronismo en el proceso de programación.

La Figura 1.4 muestra un ejemplo comparativo entre la programación en circuito y la programación por zócalo.

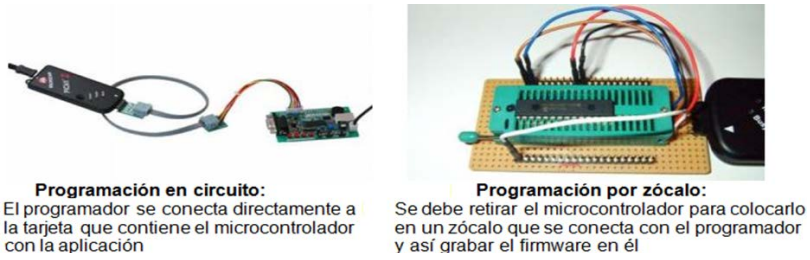


Figura 1.4. Programación en circuito Vs Programación por zócalo.

Para implementar el método de programación en modo ICSP es indispensable agregar un conector que permita el acceso a los cinco pines mencionados anteriormente, el diagrama de conexión entre microcontrolador-conector y programador se muestra en la Figura 1.5.

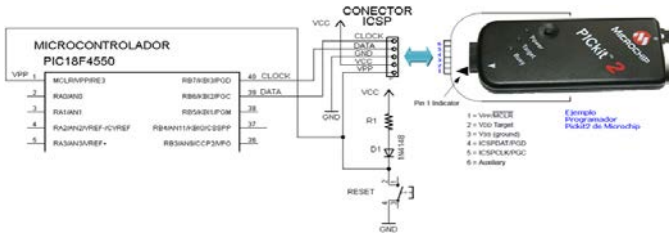


Figura 1.5. Conexión para programación en modo ICSP.

1.1.3.2 Bootloader USB de Microchip

El segundo sistema consiste en la implementación de un bootloader USB o cargador de arranque. Por ahora diremos que un bootloader o cargador de arranque es un sistema software que permite al microcontrolador programarse a sí mismo, sin necesidad de utilizar un programador externo. El hardware que se requiere es un pulsador normalmente en uno (1) lógico, conectado a un pin de I/O del microcontrolador, el pulsador de reset y dos indicadores tipo LED. El siguiente es un diseño adaptado de la tarjeta PICDEM FS USB de Microchip, la cual es una tarjeta basada en el microcontrolador PIC 18F4550 y es una tarjeta de entrenamiento para programación del microcontrolador en general y aplicaciones con comunicación USB.

Todo lo referente a cómo funciona un bootloader y la elección del mismo (ya que existen varios) se explicara más adelante en el capítulo 3, intercomunicación a nivel software del sistema.

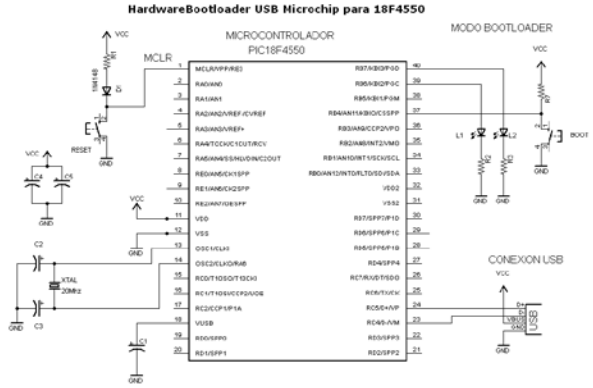


Figura 1.6. Hardware auxiliar para bootloader.

1.1.4 Sistema para detección de plantas

En el diseño se integra un método sencillo que permite detectar que planta se ha conectado a la tarjeta principal, este método consiste en tres entradas digitales

conectadas por medio de resistencias a tierra, de modo que si por cualquier pin ingresa un uno (1) o un (0) lógico este podrá ser leído, las resistencias eliminan falsas lecturas debidas a la alta impedancia de entrada. Luego si no hay alguna planta conectada las tres entradas generan el código binario 000, o cero (0) en decimal, cada planta que se conecte deberá tener salidas que generen un código fijo entre 001 y 111 en binario, o de uno (1) a siete (7) en decimal, con este método se pueden conectar hasta 7 plantas diferentes, en la Figura 1.7 se muestra el método explicado.

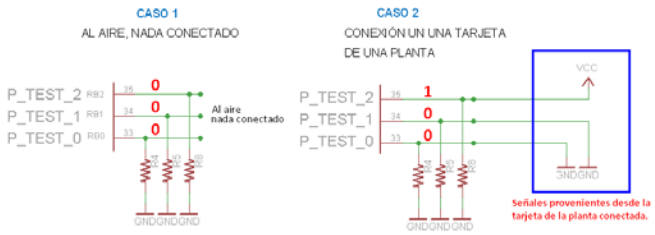


Figura 1.7. Sistema para detección de plantas.

1.1.5 Conexión Entre Tarjeta Principal y Plantas.

Finalmente para completar el diseño se debe implementar un sistema que permita una fácil conexión entre la tarjeta principal y las plantas experimentales, después de analizar diferentes clases de conectores existentes en el mercado se decide utilizar un conector para cable ribbon o cinta plana de 14 pines; en la Figura 1.8 se muestra la conexión física entre el conector y el microcontrolador, la descripción detallada de la señal de cada pin se presenta en la Tabla 1.3.

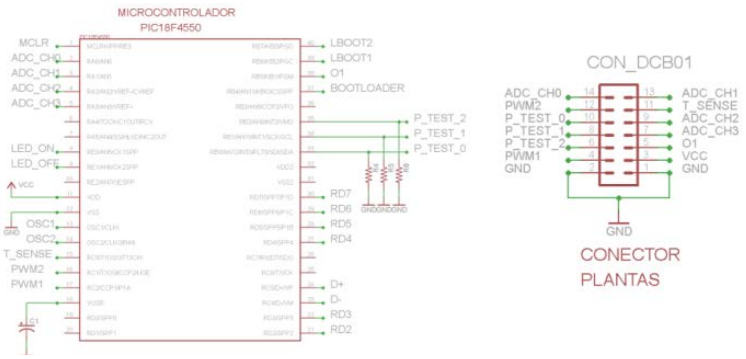


Figura 1.8. Conexión y señales conector-tarjeta principal.

Tabla 1.3. Descripción señales conector-tarjeta principal

Pin Microcontrolador		Descripción	PIN conector	Nombre ID
12	Vss	Conexión a tierra	1	GND
12	Vss	Conexión a tierra	2	GND
11	Vdd	Conexión a VCC +5VDC obtenidos del bus USB, Imax=500mA	3	VCC
17	RC2/CCP1	Salida PWM No. 1, salida del módulo CCP1.	4	PWM1

38	RB5	Salida digital 1, (configurada por defecto como salida pero, se ser modificada).	5	O1
35	RB2/INT2	Pin de entrada, corresponde al 3er bit de tres dedicados para reconocimiento de planta conectada.	6	P_TEST_2
5	RA3/ANA3	Entrada Analógica, canal 3 del microcontrolador.	7	ADC_CH3
34	RB1/INT1	Pin de entrada, corresponde al 2do bit de tres dedicados para reconocimiento de planta conectada.	8	P_TEST_1
4	RA2/ANA2	Entrada Analógica, canal 2 del microcontrolador.	9	ADC_CH2
33	RB0/INT0	Pin de entrada, corresponde al 1er bit de tres dedicados para reconocimiento de planta conectada.	10	P_TEST_0
15	RC0	Pin Entrada-Salida, configurado para el manejo del sensor DS18S20 (Protocolo 1Wire), Se puede cambiar la configuración por SW.	11	T_SENSE
16	RC1/CCP2	Salida PWM No. 2, salida del módulo CCP2	12	PWM2
3	RA1/ANA1	Entrada analógica, canal 1 del microcontrolador.	13	ADC_CH1
2	RA0/ANA0	Entrada analógica, canal 0 del microcontrolador.	14	ADC_CH0

1.1.6 Diagramas esquemáticos y fotos reales de tarjeta principal y pantalla LCD

Como resultado del proceso de diseño e implementación de la tarjeta principal, a continuación se muestran los diagramas esquemáticos y PCB's.

Tanto para el diseño de los esquemáticos como de los circuitos impresos PCB, se utiliza la herramienta CadSoft Eagle 5.9.0 versión de evaluación.

Diagramas esquemáticos

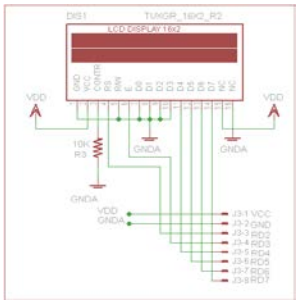


Figura 1.9. Diagrama esquemático conexión pantalla LCD.

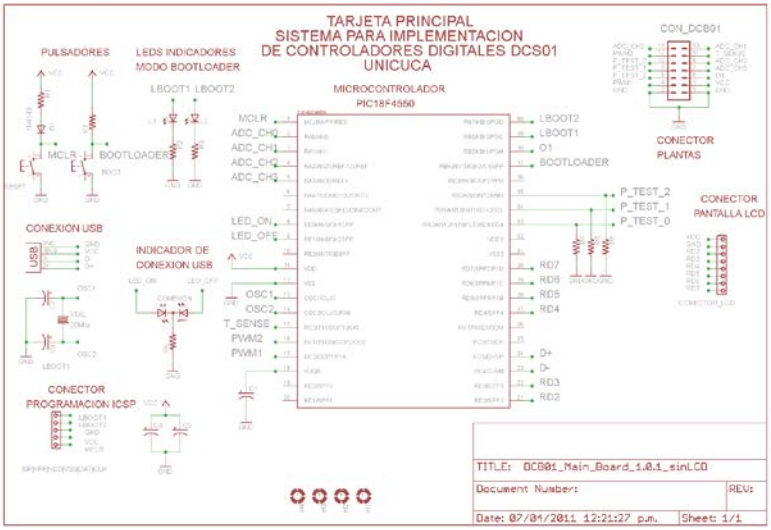


Figura 1.10. Diagrama esquemático tarjeta principal.

Partes Tarjeta principal

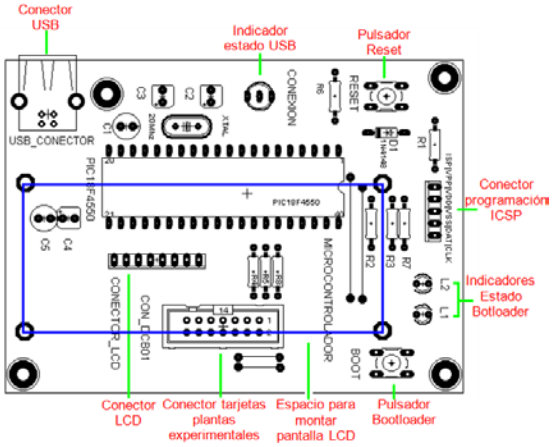


Figura 1.11. Partes tarjeta principal.

A continuación se muestran imágenes de la tarjeta principal diseñada, al lado derecho se encuentra la tarjeta principal con la pantalla LCD.

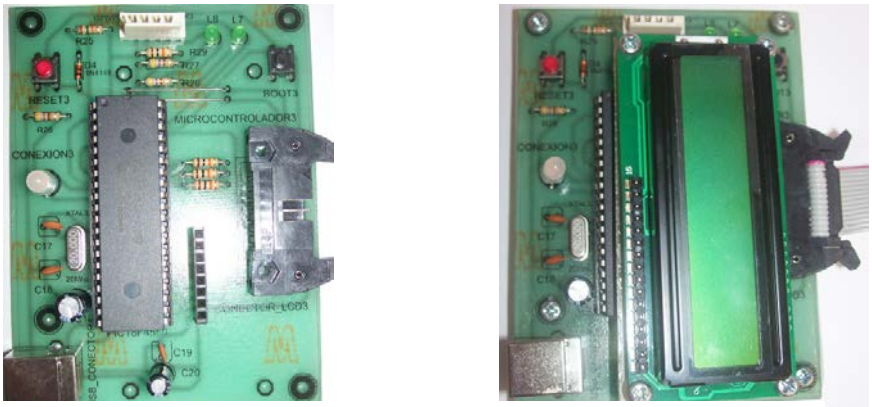


Figura 1.12. Imágenes reales de la tarjeta principal

Las características de la tarjeta principal diseñada son:

Tabla 1.4. Características tarjeta principal

Parámetro	Detalle
Frecuencia máxima de operación	48Mhz, con multiplicador de frecuencia, (cristal de 20Mhz externo).
Conexión con PC	USB 2.0.
Alimentación	5 VDC obtenidos del bus USB directamente. Consumo máximo 500mA.
Entorno de trabajo	Sistema operativo: Windows XP, comunicación con Matlab, programación en lenguaje C, IDE de desarrollo Mplab integrado con compilador CCS
Métodos de programación	Programación in-circuit ICSP, bootloader USB de Microchip.
Entradas analógicas	4, con resolución de 8 o 10 bits por hardware. Voltaje de entrada máximo 5 voltios sin acondicionamiento
Salidas PWM (Generar esfuerzo de control)	2 por hardware (módulos internos del microcontrolador). Resolución 8 o 10 bits configurable por software. Frecuencia configurable por software, configurada a 3Khz por defecto para el proyecto.
Entradas digitales fijas	3 para detección de planta conectada. 1 pulsador para entrar a modo de programación por bootloader 1 para pulsador de reset
Salidas digitales fijas	4 para indicadores LED’s
Canales bidireccionales digitales	2 configurables como entrada o salida por software, uno es utilizado para manejo del sensor de temperatura, el otro está libre.
Sistema de visualización local	Pantalla LCD de 2×16, utiliza 4 pines del microcontrolador para transmitir datos y 2 para control.

1.2 Diseño y Elaboración de Prototipo Hardware de Plantas Didácticas

La siguiente fase del proyecto consiste en diseñar e implementar dos plantas experimentales, la primera de ellas una planta de temperatura y la segunda una planta de circuitos RC, en primer lugar se establecen algunos criterios que serán la base para el diseño de dichas plantas, criterios como:

- Definición clara de los tipos de variables en cada planta: variable controlada, variable manipulada, esfuerzo de control.
- Tiempo de respuesta: se debe considerar que la capacidad de procesamiento de la tarjeta principal permita realizar una buena adquisición de las variables en cada planta, teniendo en cuenta que la tarjeta principal puede muestrear señales de hasta 100Khz para señales analógicas. Este requerimiento no es crítico en el caso de la planta de temperatura, ya que

los procesos térmicos se caracterizan por ser procesos lentos, por el contrario en el caso de la planta de circuitos RC es un aspecto a tener en cuenta en el diseño.

- Disturbios: la generación de disturbios es un aspecto importante en la experimentación práctica en la implementación de controladores, por lo que se deben implementar mecanismos que permitan generar disturbios en las plantas.
- Sensado de variable(s): tener en cuenta el tipo de sensor a utilizar, precisión del sensor, rango de variables.

1.2.1 Diseño e implementación de una planta de temperatura experimental de bajo costo

Según los criterios anteriormente expresados, se inicia el proceso en primer lugar para el diseño e implementación de la planta de temperatura, esta debe ser una planta con un tiempo de respuesta lento y con un rango amplio en la temperatura a controlar (20 a 50 grados, por ejemplo temperatura mínima 30 ^{o}C y temperatura máxima 70 ^{o}C), esto para permitir que el usuario pueda variar en múltiples valores la consigna o *set point* de la planta, además que los elementos con los que se fabrique sean de fácil consecución en el mercado.

Se propone un diseño a base de un elemento resistivo de bajo valor óhmico y alta capacidad de disipación de potencia, la idea es que al hacer circular corriente eléctrica a través de dicho elemento se produzca una disipación de potencia en forma de calor con lo que se tendrá un aumento en la temperatura alrededor del dispositivo, este es el principio de funcionamiento que se propone para la construcción de la planta experimental de temperatura, teniendo una idea base para el diseño, surgen los siguientes interrogantes:

- ¿Qué tipo de resistencia utilizar?
- ¿Qué tipo de corriente es más apropiada, AC o DC, según las características del proyecto?
- ¿Qué valor óhmico y potencia nominal debe tener la resistencia a utilizar?, ¿Se consigue este dispositivo fácilmente en el mercado?
- ¿Qué elementos se deben utilizar en la etapa de potencia?
- ¿Qué sensor utilizar?

1.2.1.1 Selección de la resistencia calefactora

Para la elección del tipo de resistencia, haciendo referencia al dispositivo, más no a la magnitud debida a la propiedad eléctrica de los materiales, se tienen las siguientes opciones:

Una resistencia a base de filamento similar a las utilizadas en artefactos como: secadores de pelo, estufas, diferentes tipos de planchas, calentadores de agua, entre otros.

Figura 1.13. Resistencias a base de filamentos.

Una resistencia de alta disipación de potencia a base de carbón y cerámica.

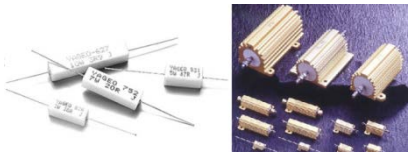


Figura 1.14. Resistencias de potencia a base de carbón.

El tipo de resistencia calefactora que se utilice definirá ciertas características de la planta como la etapa de potencia a utilizar, tiempo de respuesta, rango de operación, fuente de alimentación entre otros, por lo tanto es muy importante analizar las características de cada uno de los resistores propuestos y determinar los factores favorables y desfavorables que implican su uso.

Para definir si utilizar una resistencia de filamento o una resistencia de potencia inicialmente se debe analizar el tipo de corriente con que típicamente operan (AC o DC), aunque un resistor al ser un dispositivo pasivo puede operar con cualquiera de los dos tipos de corriente, se debe tener en cuenta que a nivel comercial hay una diferencia establecida por las aplicaciones comerciales. Las resistencias a base de filamentos son comúnmente utilizadas con corriente de tipo AC, operando con el voltaje nominal de las redes domiciliarias de 110 VAC, y con potencias que pueden fluctuar entre 5W hasta unos 3000W, esto implica corrientes de aproximadamente 0.5 Amperios hasta unos 15 Amperios, (en dispositivos como secadores de pelo, soldadores de estaño, calentadores de agua, planchas y estufas), mientras que las resistencias de potencia a base de carbón y cerámica son generalmente utilizadas en etapas de potencia en dispositivos electrónicos y se encuentran en el comercio con capacidades de disipación de potencia de 1, 2,5 y 10W, con valores fijos de resistividad y una tolerancia baja (inferior al 10%) en estos tipos de resistores la corriente dependerá del voltaje que circule a través del dispositivo mas no de alguna aplicación específica, típicamente se pueden tener corrientes desde los mA hasta unos pocos amperios.

Otro factor a analizar es la etapa de potencia que se utiliza según el tipo de resistencia. La etapa de potencia (actuador) es una interfaz entre la tarjeta principal y la resistencia, el objetivo es tener un actuador que permita una transferencia variable de la energía disponible a la resistencia, de modo que la señal de control pueda tener efecto en la disipación de potencia en la resistencia y

por consiguiente en la temperatura generada de este proceso. En las resistencias a base de filamento por su uso típico con corriente de tipo AC se utiliza habitualmente una etapa de potencia con algún tiristor, generalmente un TRIAC, dado que la señal de corriente alterna es de forma senoidal, con semiciclos positivos y negativos, el control de potencia se debe hacer en cada semiciclo activando durante cierto tiempo el TRIAC para entregar tensión a la carga, esto se debe hacer tanto para el semiciclo positivo como para el negativo, dicha operación requiere detección de cruce por cero ya que este indica el inicio de un nuevo semiciclo. El control de potencia en corriente alterna se puede apreciar en la Figura 1.15.

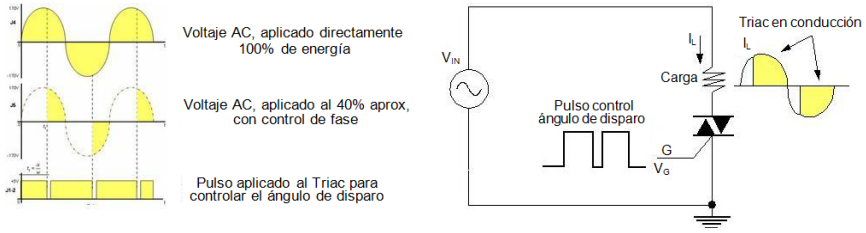


Figura 1.15. Control de potencia en corriente alterna.

En las resistencias a base de carbón utilizadas con corriente directa se puede utilizar como etapa de potencia, arreglos a base de transistores, los transistores funcionan como llaves que regulan el suministro de potencia a la carga. Como señal de entrada a dicho arreglo se suele utilizar como señal de control una señal de tipo PWM (modulación por ancho de pulso), la cual consiste en una señal cuadrada de frecuencia constante, pero con un ciclo de trabajo (tiempo en alto) variable de 0 al 100% lo que permite controlar la transferencia de potencia variando este parámetro (ciclo de trabajo o *duty cicle*). La Figura 1.16 muestra ejemplos típicos de etapas de potencia con transistores.

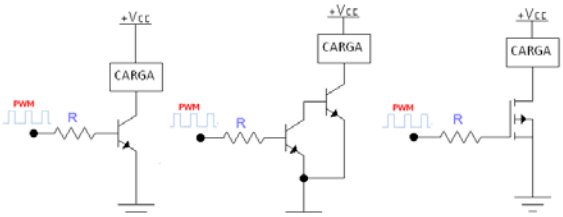


Figura 1.16. Etapa de potencia con transistores para cargas que operan con DC.

El resumen se presenta en la Tabla 1.5, en la que se comparan las características de los resistores propuestos.

Tabla 1.5. Características tipos de resistencias

Característica	Resistor de filamento	Resistor a base de carbón
Tipo de voltaje con el que opera típicamente	Corriente Alterna AC	Corriente Alterna AC y Corriente Directa DC
Consumo (Amperios)	Mayor a 0.5 A	Variable mA - A
Etapa de potencia	Control de fase en AC con TRIAC	Transistores FET, MOSFET

Forma y tamaño	Varía según la aplicación, se consigue como un alambre que puede estar enrolladlo como un espiral o no, según la aplicación	Cilíndrica, forma de cubo rectangular, hasta 5cm largo.
Precisión	No definida, varia con el uso y el tamaño	Dada por el fabricante ±10%

Se opta por elegir el resistor a base de carbón y recubrimiento con cerámica como elemento calefactor para la planta de temperatura, ya que presenta las siguientes ventajas:

- Fácil consecución en el mercado.
- Existen múltiples valores de resistencias y con capacidades diferentes de disipación de potencia, lo que permite amplias posibilidades de experimentación, además su valor óhmico presenta una tolerancia relativamente baja en comparación al otro tipo de resistencia propuesta.
- Se puede operar con corrientes bajas y a la vez disipar potencia en forma de calor suficiente para generar temperaturas útiles para el proyecto.
- La anterior característica de manejo de corriente permite utilizar una fuente de alimentación de tipo DC, lo cual es menos peligroso que utilizar AC y se puede utilizar una fuente de baja capacidad de corriente.
- Su tamaño reducido permitirá diseñar un PCB pequeño que contenga la etapa de potencia (actuador), sensor y elementos necesarios para la comunicación con la tarjeta principal.
- La transmisión de potencia no implica adiciones a nivel hardware o software especialmente dedicados para la detección de cruce por cero, que si se requerirían en el caso de trabajar con un dispositivo de corriente AC.

Se inicia un diseño con resistencias de potencia de diferentes valores óhmicos, teniendo como referencia el uso de una fuente que pueda suministrar 12VDC continuos y 1A como corriente máxima, se determina un rango de valores de resistencias para un consumo de corriente entre 100 mA y 800mA así:

$$\mathrm{VDC} = 12\mathrm{V},\ \ \mathrm{Imin} = 100\mathrm{mA},\ \ \mathrm{Imax} = 800\mathrm{mA}$$

$$\mathrm{Rmin} = \frac{\mathrm{VDC}}{\mathrm{Imax}} = \frac{12\mathrm{V}}{800\mathrm{mA}} = 15\Omega \qquad (1.1)$$

$$\mathrm{Rmax} = \frac{\mathrm{VDC}}{\mathrm{Imin}} = \frac{12\mathrm{V}}{100\mathrm{mA}} = 120\Omega \qquad (1.2)$$

$$\mathrm{Pmin} = \mathrm{VDC} \times \mathrm{Imin} = 12\mathrm{V} \times 100\mathrm{mA} = 1.2\ \mathrm{Watts} \qquad (1.3)$$

$$\mathrm{Pmax} = \mathrm{VDC} \times \mathrm{Imax} = 12\mathrm{V} \times 800\mathrm{mA} = 9.6\ \mathrm{Watts} \qquad (1.4)$$

El rango de las resistencias obtenidas es de 15Ω a 120Ω, con una potencia de 1.2 Watts a 9.6 Watts.

1.2.1.2 Etapa de potencia para la planta de temperatura (Actuador)

Luego de haber seleccionado el rango de valores para la resistencia calefactora, el siguiente paso es elegir una etapa de potencia robusta que permita una transferencia de potencia en forma eficiente. El objetivo principal en esta fase de diseño es tener una etapa de potencia que permita realizar pruebas con resistencias de distintos valores y así medir la temperatura generada por la disipación de potencia, con este proceso se determinara el rango de temperatura que se puede obtener con cada valor de resistencia y el tiempo que toma el sistema en ir desde la temperatura ambiente hasta la máxima temperatura según la corriente consumida, finalmente se escogerá la resistencia que presente mejores características para la implementación de la planta.

En el proceso de diseño y selección de la etapa de potencia se analizó el módulo MIC 955 de empresa Feedback, un sistema didáctico para el estudio de control de un sistema de temperatura, este módulo se encuentra disponible en el departamento instrumentación y control; analizando el manual de usuario del módulo MIC955 se encontró el diagrama esquemático de la electrónica que corresponde a la etapa de potencia (actuador), para el sistema de calefacción del mismo, además de un diagrama de potencia para el uso de un ventilador, utilizado para la introducción de disturbios en el sistema, se decide poner a prueba este modelo y con el realizar prácticas para la selección de la resistencia calefactora. La Figura 1.17 muestra la etapa de potencia.

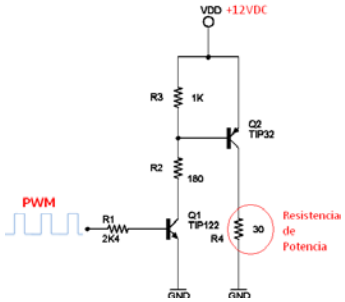


Figura 1.17. Etapa de potencia adaptada del módulo MIC 955.

La resistencia R4 es la resistencia de potencia que funcionara como dispositivo calefactor. Para seleccionar un valor de resistencia adecuado se eligen cinco (5) valores dentro del rango Rmin - Rmax calculado anteriormente, con ellas se prepara una práctica en la cual se pone a prueba la etapa de potencia y se toman los datos necesarios para seleccionar el valor de resistencia apropiado para la planta. A continuación se describe en breve la práctica realizada y los resultados obtenidos.

Realizar el montaje de la etapa de potencia (iniciar con la resistencia R4 de menor valor), ver Figura 1.18.

Ubicar la resistencia calefactora (R4) y la termocupla de tal modo que queden aproximadamente a 3 – 4 mm de separación entre ellas.

Anotar la temperatura inicial (temperatura ambiente).

Cambiar el estado del interruptor normalmente conectado a tierra para ingresar 5V a la base del transistor a través de la resistencia R1; con este proceso se introduce un cambio de 0 al 100% en la energía aplicada al transistor.

Inmediatamente se cambia el estado del interruptor se debe iniciar el cronometro.

Medir la corriente atreves de la resistencia calefactora R4, ver conexión de amperímetro en la Figura 1.18.

Parar el cronometro si la temperatura alcanza un valor de 100ºC o si la temperatura se estabiliza en un valor inferior.

Anotar la temperatura final.

Realizar el mismo procedimiento para cada una de las resistencias (18, 22, 30, 36, 51).

La Figura 1.18 representa el esquema de la práctica y los resultados obtenidos se consignan en la Tabla 1.6.

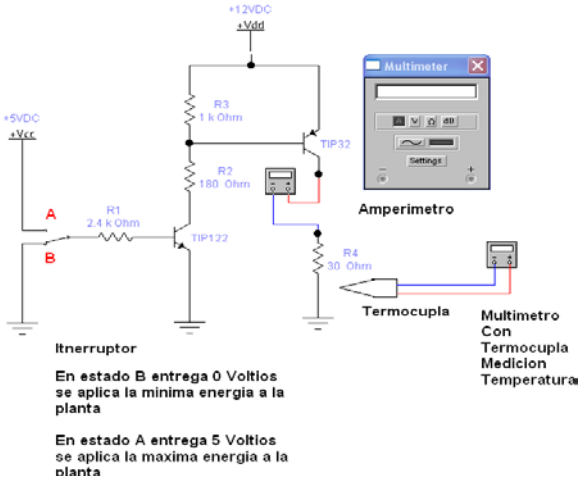


Figura 1.18. Esquema practica selección de resistencia.

Tabla 1.6. Resultados practica selección de resistencia

Practica	Valor resistencia [Ohms]	PWM aplicado [%]	Temperatura inicial (T. ambiente) [ºC]	Temperatura Final [ºC]	Tiempo en Alcanzar T. Final [min]	Corriente Consumida [mA]
1	18	100	21	100	1,7	610
2	22	100	25	100	1,17	520
3	30	100	26	100	3,2	370
4	36	100	23	83	4,15	330
5	51	100	26	67	5	230

Según los resultados obtenidos de la práctica se descartan las resistencias de las practicas 1 y 2 por el elevado consumo de corriente, ya que con estos valores de resistencia se observó una alta disipación de potencia en la resistencia R2 de la etapa de potencia. Se descarta la resistencia de la practica 5 porque la temperatura máxima es baja, lo cual reduce considerablemente el rango de trabajo. Quedan entonces como opciones las resistencias de las prácticas 3 y 4 de las cuales se elige la número 3 ya que presenta un mejor rango en la variación de temperatura, respecto al número 4, con un consumo moderado de corriente.

Caracterización del actuador de la planta de temperatura

Es importante analizar el comportamiento del actuador de la planta de temperatura por lo que se llevó a cabo una práctica que permite observar el efecto de la variación de la señal PWM de entrada frente a la caída de voltaje en la resistencia calefactora y por ende la potencia disipada en la misma.

La práctica consiste en realizar una variación en el ciclo útil de la señal PWM de 0 a 100% en pasos de 5, para cada valor diferente en el rango se midió con un voltímetro la caída de voltaje en la resistencia calefactora, con este valor y considerando que la resistencia es de valor constante (30Ω), se puede calcular la corriente consumida por la resistencia y la potencia disipada.

Tabla 1.7. Toma de datos caracterización actuador planta temperatura.

Ciclo útil PWM		**Voltaje en la resistencia calefactora (30 Ω) [Voltios]**	**Corriente a través de la resistencia calefactora [mA] I = V/R**	**Potencia disipada por la resistencia calefactora [mW] $P = V^2 / R$**
Porcentaje [0-100]	**Valores de registro [0-1000]**			
0	0	0	0,00	0,00
5	50	0,99	33,00	32,67
10	100	1,6	53,33	85,33
15	150	2,19	73,00	159,87
20	200	2,78	92,67	257,61
25	250	3,37	112,33	378,56
30	300	3,96	132,00	522,72
35	350	4,56	152,00	693,12
40	400	5,14	171,33	880,65
45	450	5,71	190,33	1086,80
50	500	6,33	211,00	1335,63
55	550	6,92	230,67	1596,21
60	600	7,52	250,67	1885,01
65	650	8,11	270,33	2192,40
70	700	8,7	290,00	2523,00
75	750	9,26	308,67	2858,25
80	800	9,87	329,00	3247,23

85	850	10,46	348,67	3647,05
90	900	11,06	368,67	4077,45
95	950	11,64	388,00	4516,32
100	1000	11,76	392,00	4609,92

La siguiente imagen muestra la relación, entre el ciclo útil de la señal PWM y el voltaje en la resistencia calefactora, se puede apreciar que existe una relación lineal entre estas dos variables.

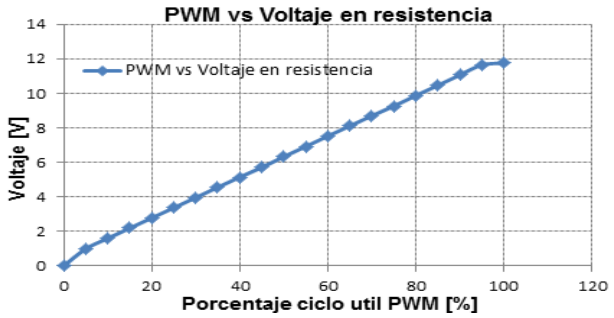


Figura 1.19. Relación entre ciclo útil señal PWM y caída de voltaje en la resistencia calefactora.

También se debe tener en cuenta que la variable manipulada es la potencia disipada por la resistencia, por lo que se debe analizar la relación entre la señal PWM y dicha variable.

En la siguiente figura se muestra la relación entre el ciclo útil de la señal PWM contra la potencia disipada por la resistencia, se puede observar que no existe una relación lineal, la relación es cuadrática.

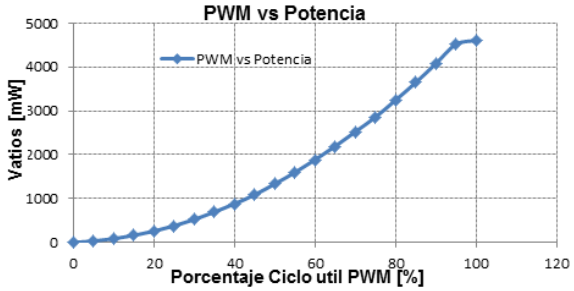


Figura 1.20. Relación entre ciclo útil señal PWM y potencia disipada por la resistencia calefactora.

Según los resultados de los análisis obtenidos se resalta la importancia de tener en cuenta la no linealidad entre la señal PWM y la potencia disipada (variable manipulada) para generar el esfuerzo de control, el cual debe indicar al actuador la cantidad de energía que se debe aplicar para realizar las acciones de control en la planta de temperatura.

La siguiente imagen presenta en una sola grafica la relación entre las señales PWM, la caída de voltaje y la potencia disipada por la resistencia, los datos son

normalizados en valores de porcentaje de 0 a 100 para un mejor análisis, la gráfica muestra que al fijar un porcentaje en el ciclo útil de la señal de PWM se obtiene un valor equivalente en el voltaje aplicado entre los terminales de la resistencia pero el porcentaje de la potencia difiere dada la no linealidad, por ejemplo:
Si porcentaje PWM = 60% el voltaje es aproximadamente el 60% del valor máximo (11,76v según la tabla), pero la potencia es solo el 40%.

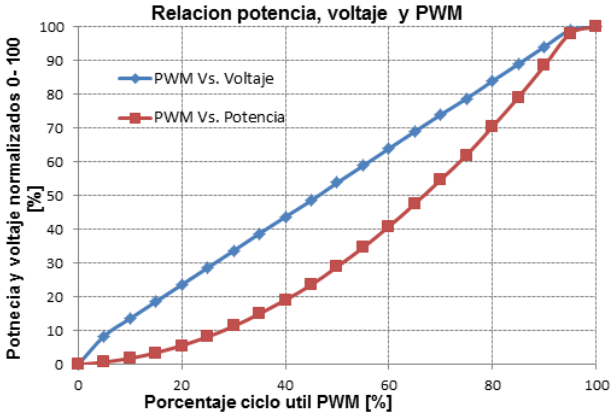


Figura 1.21. Relación normalizada entre ciclo útil PWM, voltaje en la resistencia y potencia disipada.

1.2.1.3 Selección del sensor de temperatura

Una fase importante en el diseño de la planta de temperatura es la selección del sensor, esta selección se hace teniendo en cuenta características como: precisión, rango de medición, costo y manejo entre otras. Para la selección del sensor se opta por dejar de lado algunos tipos de sensores como termocuplas y sensores tipo PT100, y elegir un sensor de tipo circuito integrado, ya que estos sensores requieren menos elementos para adecuar la señal, operan en temperaturas de hasta 120 ºC, son de fácil consecución en el mercado, de bajo costo y fácil manejo, luego de consultar referencias sobre este tipo de sensores se eligen los siguientes cuatro (4) candidatos.

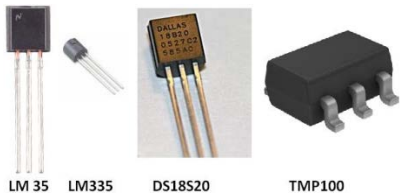


Figura 1.22. Circuitos integrados para medir temperatura.

Los anteriores sensores se consiguen fácilmente en el mercado y son ampliamente utilizados en aplicaciones donde se requiere mediciones de temperaturas en promedio de hasta unos 120 ºC. La selección se hace comparando las principales características de cada sensor, en la Tabla 1.8 se mencionan las más relevantes para el proyecto.

Tabla 1.8. Características sensores de temperatura

Características	Referencia Sensor			
	LM 35 (National Instruments)	**LM 335 (National Instruments)**	**DS18S20 (Dallas Semiconductor)**	**TMP100 (BB-Texas Instruments)**
Unidad de medición	°C	Kelvin, °K	°C o °F Configurable	°C
Rango de medición	−55° a +150°C	−55° a +150°C	–55°C a +125°C	–55°C a +125°C
Precisión	±1°C	±1°C	±0.5°C, ±0.2°C Configurable	±2°C
Salida	Voltaje, 10mV/°C	Voltaje, 10 mV/°K	Digital, protocolo 1wire	Digital, protocolo I2C
Voltaje Alimentación	4 a 20 VDC	5 a 20 VDC	3 a 5.5 VDC	2.7 a 5.5 VDC
Requiere adecuación externa	Si	Si	No	No
Empaquetado	TO-92 tres pines	TO-92 tres pines	TO-92 tres pines	SOT-23, Montaje superficial
Costo (Pesos)	3.190	2.088	6.380	9.280

Analizando las características de la tabla anterior, se opta por trabajar con el sensor DS18S20 de Dallas semiconductor ya que es un sensor de bajo costo y presenta múltiples características las cuales son ideales para el desarrollo del presente proyecto tales como: como su precisión, montaje simple y rango de medición. El esquema de conexión del sensor se muestra en la Figura 1.23, la operación a nivel software del sensor se explica en el capítulo 3, más detalles del sensor ver datasheet (MAXIM, 2010).

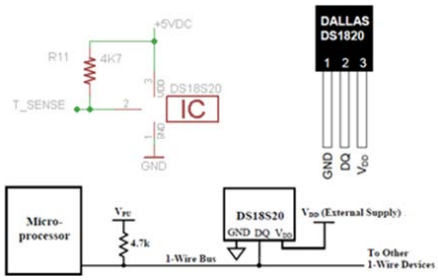


Figura 1.23. Conexión sensor de temperatura.

1.2.1.4 Introducción de disturbios en la planta de temperatura

Dada la importancia de la experimentación en los métodos de control, se agrega un mecanismo para introducir disturbios en la planta de temperatura, este consiste en un ventilador. El diagrama electrónico para su manejo se adapta del diseño del módulo MIC 955 de la empresa FeedBack, el sistema tiene dos modos, el primero de ellos es un método manual para activar el ventilador mediante un

potenciómetro que permite graduar la velocidad, el segundo método es automático y permite el control del ventilador (velocidad) con la segunda salida PWM mediante software, la selección del modo se hace mediante un interruptor, en la Figura 1.24 se detalla el diagrama esquemático del sistema para introducir el disturbio en la planta.

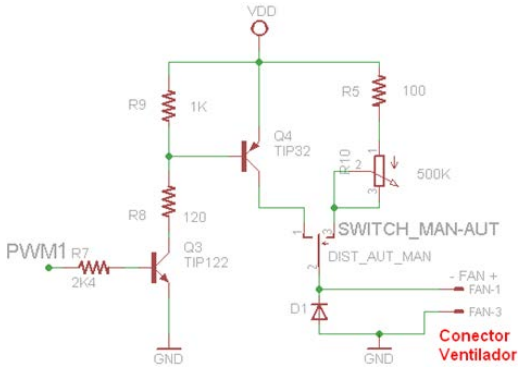


Figura 1.24. Etapa de potencia ventilador.

1.2.1.5 Reconocimiento de la planta de temperatura

Acorde con lo explicado en la sección 1.1.4 Sistema para detección de plantas, para que la planta sea detectada esta debe generar un código binario de tres (3) bits, se genera el código 010 colocando los pines P_TES_0 = 0, P_TES_1 = 1, P_TES_2 = 0, teniendo como referencia que 0 = conexión a tierra y 1 = conexión a +5VDC, ver Figura 1.25.

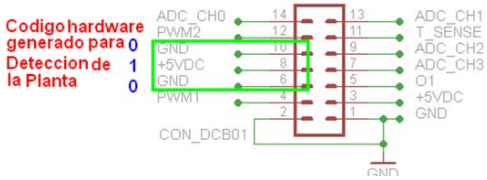


Figura 1.25. Código hardware para detección de planta de temperatura.

1.2.1.6 Diagrama esquemático y fotos de la planta de temperatura

El diagrama esquemático completo de la electrónica de la planta de temperatura se encuentra en la Figura 1.26, en ella se muestran los bloques principales, etapa de potencia (actuador), resistencia calefactora, la etapa de potencia para controlar el ventilador (generador de disturbios) y conexión a la tarjeta principal.

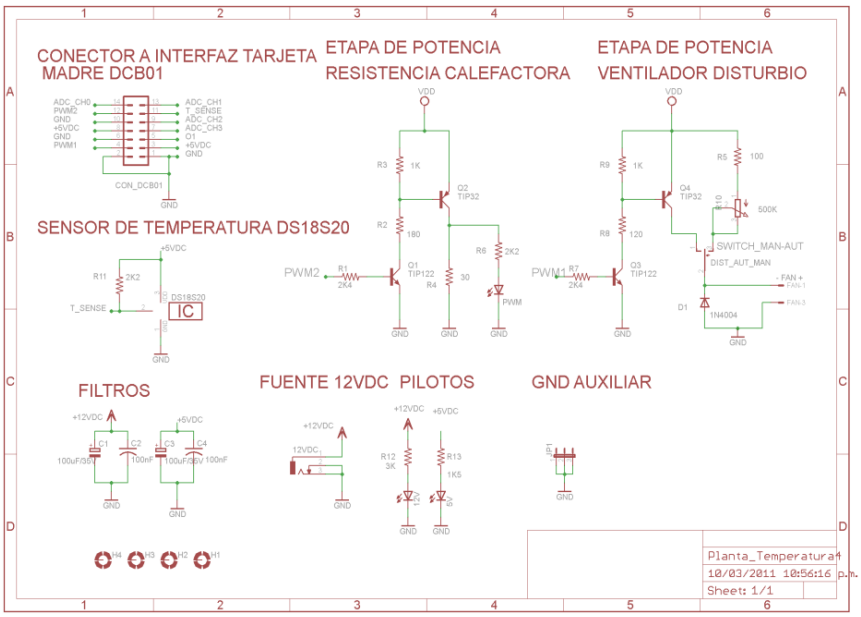


Figura 1.26. Diagrama esquemático planta de temperatura.

Posterior a la prueba de funcionamiento del sistema electrónico en protoboard y teniendo definido el esquema electrónico se procede al diseño del circuito impreso PCB según diagrama esquemático, para ello se utiliza del software CadSoft Eagle 5.9.0 versión de prueba, (ver anexo 2).

La Figura 1.27 muestra las partes de la tarjeta de la planta de temperatura.

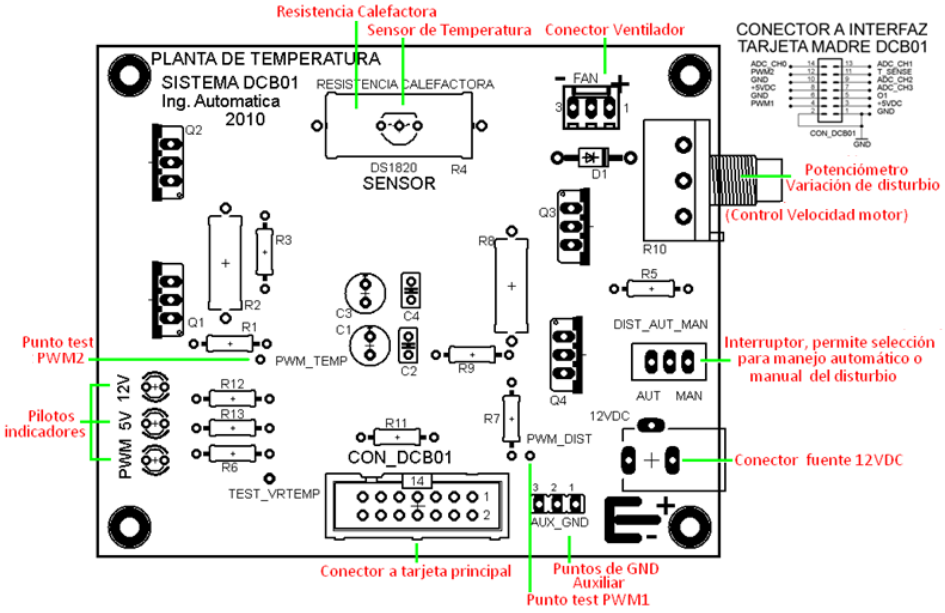


Figura 1.27. Partes planta de temperatura.

A continuación se muestran dos imágenes de la planta de temperatura diseñada, a la izquierda se encuentra la tarjeta que corresponde a la planta y a la derecha la planta con su respectiva estructura.

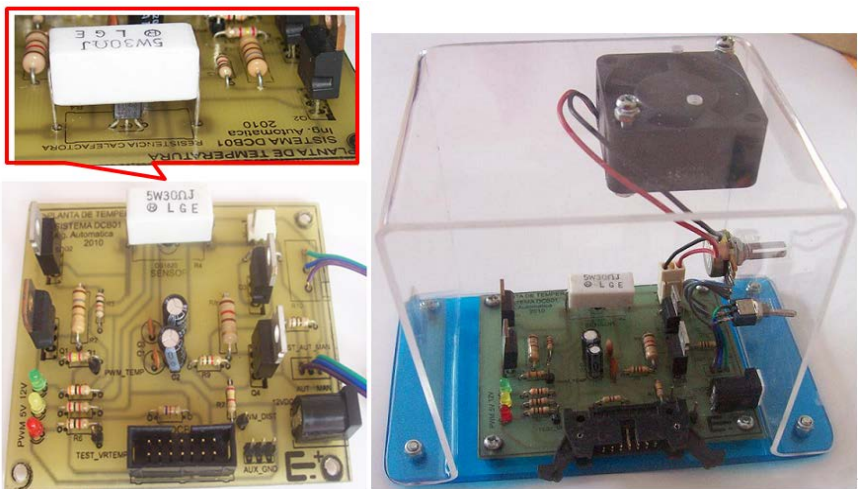


Figura 1.28. Imágenes reales de la planta de temperatura

Las características de la planta de temperatura son:

Tabla 1.9. Características planta temperatura

Parámetro	Detalle
Rango de temperatura de operación (Variable controlada).	25 – 95 ºC, se dejara como rango de operación 25 – 85 ºC.
Spam	60 ºC
Tiempo que toma alcanzar la temperatura máxima.	6 min, temperatura medida con sensor DS18S20.
Consumo máximo de corriente.	420 mA
Sistema Sensor - Transmisor.	Circuito integrado DS18S20 (Sensor y trasmisor empaquetado)
Voltaje de operación	12VDC
Señal de control	PWM 3KHZ
Disturbio experimental	Ventilador controlado por PWM
Puntos de prueba	Dos en los que se puede medir mediante un osciloscopio las señales de PWM (PWM1 para disturbio, PWM2 para resistencia calefactora).
Puntos de tierra auxiliar	Tres para realizar mediciones.

1.2.2 Diseño y construcción de una planta eléctrica de circuitos (mallas) RC.

Para esta planta se propone un circuito eléctrico conformado por tres mallas RC, configurables, de modo que se pueda obtener sistemas de primer, segundo y tercer orden, esto permitirá poner en práctica diferentes controladores según el orden del sistema.

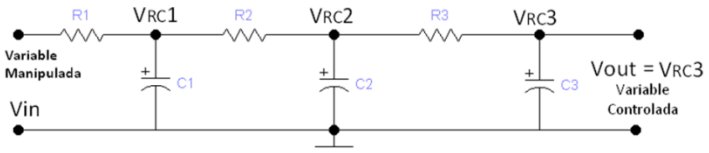


Figura 1.29. Diagrama para planta circuitos RC.

1.2.2.1 Etapa de potencia de la planta de circuitos eléctricos RC (Actuador)

Como variable de entrada (variable manipulada) para esta planta se requiere una señal de voltaje variable a la entrada del circuito, después de analizar los recursos con los que cuenta la tarjeta principal de control, se proponen dos soluciones para obtener dicha señal.

Agregar un conversor Digital Analógico (DAC), y controlarlo de forma serial desde la tarjeta de control.

Utilizar una de las dos salidas PWM con que cuenta la tarjeta principal y utilizar un filtro pasa-bajo para realizar una conversión de PWM a voltaje DC.

La primera solución involucra el uso de dos circuitos integrados, un registro para convertir información serie a paralelo, ya que los DAC más comunes y de bajo costo como el DAC0800 o el DAC0808 tienen una entrada digital en paralelo de 8 bits y salida analógica de 0 a 5v DC, la resolución es igual al número de entradas, es decir 8 Bits así que para generar un voltaje de salida entre 0 y 5 voltios se tiene como entradas 255 valores diferentes.

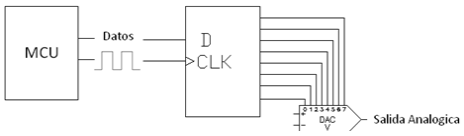


Figura 1.30. Modelo actuador con DAC externo.

La segunda opción es conectar un filtro pasa-bajo al pin de salida de una señal de PWM, con esta opción la implementación hardware es menor que en la opción anterior, además la resolución del voltaje DC generado será igual que resolución de la señal PWM, en este caso 10 bits, hasta 1024 valores diferentes de entrada para una salida de voltaje de 0 a 5 voltios, algo importante a tener en cuenta es la selección del conjunto resistencia-capacitor, ya que dicho conjunto creara un retardo en el actuador, que está dado por Tao = R×C, entonces al elegir este método la resistencia y el capacitor que se utilicen deberán generar un Tao mucho menor que el mínimo generado por la planta, el actuador debe ser más rápido que la planta.

Figura 1.31. Modelo actuador con filtro RC.

Se elige la opción del filtro pasa-bajo para generar la variable manipulada (voltaje de entrada a la planta), adicional a este filtro se debe tener en cuenta una etapa de potencia que permita entregar esta señal a la planta. Un dispositivo ampliamente usado es el amplificador operacional, con características de alta impedancia de entrada, esta característica evita la atenuación en la señal PWM generada por el microcontrolador. Se plantean las siguientes opciones:
Utilizar un amplificador en modo seguidor de tensión con lo que se tendrá una señal de control variable en un rango de 0 a 5 voltios.

Utilizar un amplificador operacional en modo no inversor con una ganancia de dos (2), con este método se amplía el rango de trabajo de 0 - 5 voltios por un voltaje variable en un rango de 0 - 9 o 0 - 10 voltios como máximo, esto teniendo en cuenta factores como: la linealidad, la saturación del amplificador y el hecho que se cuenta con una fuente de 12VDVC.

Ya que la segunda opción presenta la ventaja adicional de ampliar el rango de la variable manipulada, manteniendo la característica de alta impedancia de entrada y además el diseño e implementación del amplificador en modo no inversor solo involucra adicionar 4 resistencias; se elige esta opción para su implementación.

En la Figura 1.32 se presenta el esquema propuesto:

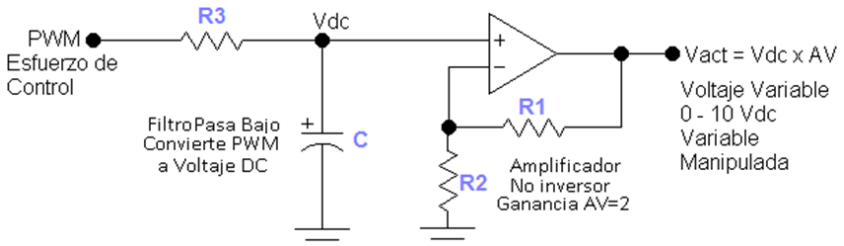


Figura 1.32. Actuador planta de circuitos RC.

Se procede con el diseño del amplificador en modo no inversor con ganancia 2.

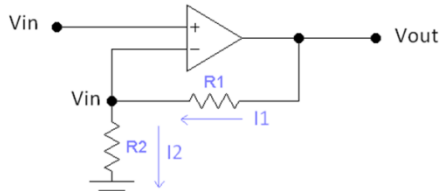


Figura 1.33. Amplificador no inversor.

Para un amplificador en modo no inversor se tiene que:

$$Vout = Vin \times \left(1 + \frac{R1}{R2}\right) \qquad (1.5)$$

Luego la ganancia está dada por la salida sobre la entrada, así

$$AV = \frac{Vout}{Vin} \tag{1.6}$$

Por lo tanto, este circuito tiene una ganancia en tensión $AV = 1 + \frac{R1}{R2}$ (Electrónica Unicrom).

Este circuito tiene como característica que la impedancia de entrada Zin es muy elevada, mientras que la impedancia de salida Zout es muy baja.
Para una ganancia de dos (2) se fijan los valores de R1 y R2 de igual valor, así:
$R1 = R2 = 1K\Omega$.

Para el diseño del filtro se debe tener en cuenta la frecuencia de la señal de PWM; como se mencionó anteriormente, la dinámica del actuador debe ser mucho más rápida que la dinámica de la planta, de modo que para generar una señal analógica con una señal PWM, la constante de tiempo (τ) de la pareja resistencia-capacitor (RC) debe ser 100 veces mayor que el periodo de la señal de PWM, entonces se tiene que:
(EMCELETTRONICA, 2008)

$$\mathrm{Fpwm} = 3\mathrm{Khz}$$

$$\mathrm{Tpwm} = \frac{1}{\mathrm{Fpwm}} = 0.333\mathrm{ms} \tag{1.7}$$

$$\tau_{\mathrm{RC}} = 100 \times \mathrm{Tpwm} = 33.3\mathrm{ms} \tag{1.8}$$

El periodo del PWM (Tpwm), indica que la constante de tiempo (τ) del arreglo resistencia-capacitor (RC) debe ser de 33.3ms entonces se eligen los siguientes valores comerciales tanto para resistencia como para capacitor:

$$\mathrm{R} = 3.3\mathrm{K}\Omega$$

$$\mathrm{C} = 10\mu\mathrm{F}$$

$$\tau = \mathrm{R} \times \mathrm{C} = 3.3\mathrm{K}\Omega \times 10\mu\mathrm{F} = 33\mathrm{ms} \tag{1.9}$$

Si el capacitor es de menor valor este posiblemente no sea capaz de filtrar las oscilaciones y por el contrario si el capacitor es de mayor valor, el actuador se vuelve lento ante cambios en el ciclo útil de la señal de PWM. En la Figura 1.34, el primer recuadro corresponde al circuito implementado para realizar las simulaciones de la respuesta del filtro, en los siguientes tres recuadros de la misma figura, se muestra el comportamiento con los valores de resistencia y capacitor elegidos a través del diseño, ante cambios en el ciclo útil de la señal PWM, la Figura 1.35 corresponde a la simulación con un capacitor de 1µF, allí se puede observar que se genera un rizado a la salida del filtro.

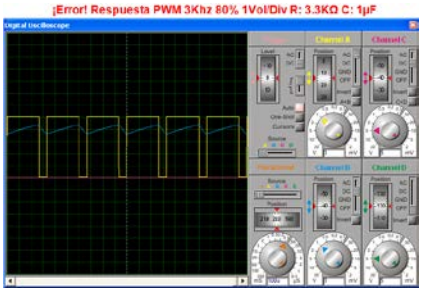


Figura 1.34. Simulación filtro planta circuitos RC

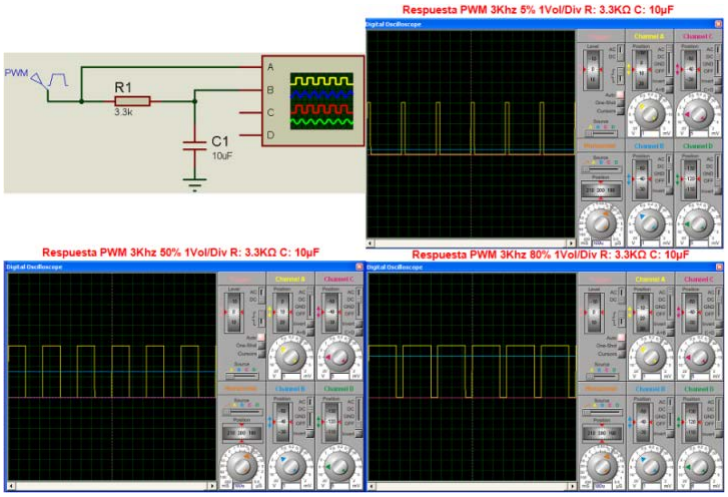


Figura 1.35. Simulación filtro con capacitor 1µF.

Se realizan pruebas con el amplificador operacional CA3140, se elige por conocimientos adquiridos en prácticas anteriores, en las que se comprobó su excelente comportamiento, además este amplificador tiene la característica que al usarlo como amplificador no inversor presenta un voltaje de 0V ante una entrada de 0V sin necesidad de componentes adicionales.

Características importantes del amplificador operacional

- Voltaje Alimentación 4 - 36 V
- Alta Impedancia de entrada 1.5Tohm típicamente
- Bajo consumo de corriente de entrada 10pA a +-15V

Se realiza el montaje de la Figura 1.32 con los valores de resistencia, capacitor obtenidos y el amplificador operacional CA3140. Se genera una señal de PWM con el microcontrolador 18F4550, la cual se varía de 0% al 100% en pasos incrementales de 5%, el objetivo de esta práctica es analizar el comportamiento y linealidad del actuador diseñado.

Tabla de caracterización del actuador

Tabla 1.10. Caracterización del actuador planta circuitos RC

Valor registro PWM	Ciclo útil PWM [%]	Voltaje Salida Filtro	Voltaje Salida Actuador (VM)	Ganancia Amplificador Operacional	Voltaje Salida Actuador (VM)Medido con la Tarjeta principal
0	0	0,02	0,04	2	0,03
50	5	0,26	0,53	2,04	0,528
100	10	0,51	1,02	2	1
150	15	0,75	1,52	2,03	1,49
200	20	1	2,01	2,01	1,97
250	25	1,24	2,51	2,02	2,46
300	30	1,49	3,01	2,02	2,95
350	35	1,73	3,5	2,02	3,43
400	40	1,98	4	2,02	3,92
450	45	2,24	4,5	2,01	4,41
500	50	2,47	4,99	2,02	4,89
550	55	2,72	5,49	2,02	5,38
600	60	2,97	5,98	2,01	5,87
650	65	3,2	6,48	2,03	6,36
700	70	3,46	6,98	2,02	6,84
750	75	3,7	7,47	2,02	7,34
800	80	3,95	7,97	2,02	7,81
850	85	4,2	8,47	2,02	8,3
900	90	4,44	8,96	2,02	8,79
950	95	4,68	9,46	2,02	9,27
1000	100	4,93	9,89	2,01	9,69

A partir de los datos se genera la siguiente gráfica, en la cual se muestra la linealidad del actuador para la planta de circuitos RC.

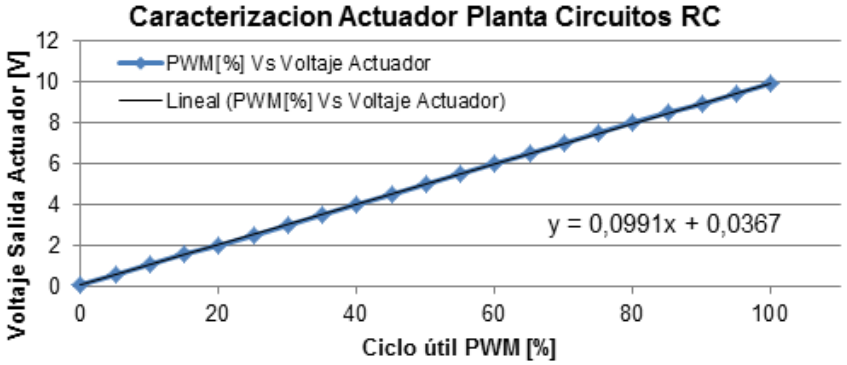


Figura 1.36. Caracterización del actuador planta circuitos RC.

De la Figura 1.36 se puede concluir que el arreglo propuesto para el actuador, tiene una alta linealidad en el rango de trabajo, por lo que se considera muy apropiado para integrarlo a la planta de circuitos RC.

1.2.2.2 Diseño de las mallas de la planta RC

Ahora se procede con la selección de los valores para las tres (3) resistencias y los tres (3) capacitores que conforman la planta, (Figura 1.29), en este punto el criterio de diseño es el τ de cada malla, se eligen capacitores con valores altos de capacitancia de modo que el menor de ellos genere un τ mayor (mínimo seis veces mayor), que el τ del actuador el cual es 33.3ms. Los valores elegidos son:

Malla 1:

$$R = 2K\Omega, \qquad C = 330\mu F$$
$$\tau = R \times C = 2K\Omega \times 330\mu F = 660ms \tag{1.10}$$

Malla 2

$$R = 2K\Omega, \qquad C = 220\mu F$$
$$\tau = R \times C = 2K\Omega \times 220\mu F = 440ms \tag{1.11}$$

Malla 3

$$R = 2K\Omega, \qquad C = 100\mu F$$
$$\tau = R \times C = 2K\Omega \times 100\mu F = 200ms \tag{1.12}$$

Para tener un sistema en el cual se pueda configurar el orden de la planta, se realiza un diseño hardware, basado en *jumpers*, los cuales permiten habilitar o deshabilitar los diferentes elementos (resistencias y capacitores) que componen la planta.

El sistema funciona de la siguiente manera, si se desea deshabilitar una resistencia esta se debe poner en corto, en el caso de un capacitor para deshabilitarlo este se debe poner en circuito abierto, en la Figura 1.37 se muestra el mecanismo de configuración.

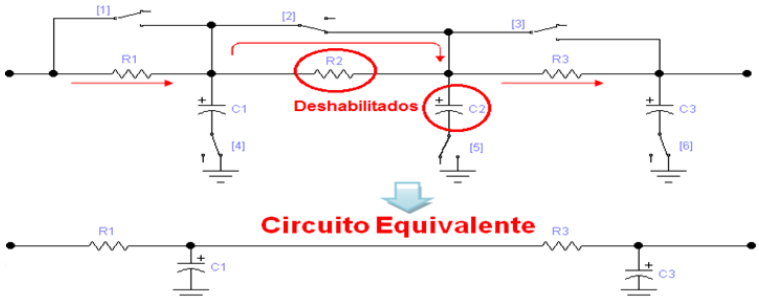


Figura 1.37. Mecanismo de configuración de orden planta circuitos RC.

1.2.2.3 Disturbio hardware para planta de circuitos RC.

Es importante contar con la posibilidad de introducir disturbios en la planta de modo que se pueda poner a prueba y experimentar la robustez de los controladores que se implementen para ella, hay dos métodos para introducir un disturbio, el primero es generar un disturbio controlado mediante software, el proceso puede ser restringiendo los límites del esfuerzo de control (generado por el ciclo útil señal PWM) durante cierto tiempo, el segundo método es mediante

hardware, este consiste en modificar la carga a la salida del circuito RC, para dicho efecto se instaló un sistema de interruptor conectado en serie a la resistencia de carga, de modo que en el estado cerrado habilita la carga al circuito y en estado abierto la deshabilita, la siguiente figura muestra el esquema del circuito.

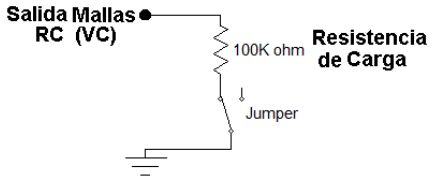


Figura 1.38. Sistema generación disturbio hardware para la planta RC

1.2.2.4 Sensor transmisor de voltaje.

Las variables que se tienen en la planta de circuitos RC son voltajes que pueden ser medidos mediante las entradas analógicas de la tarjeta principal, para lo cual se debe tener en cuenta dos aspectos importantes, el primero es el manejo de impedancias; se debe garantizar que al conectar las señales de la planta a la tarjeta principal no se genere atenuación en el nivel de la señal, por tanto no es recomendable conectar directamente las variables a las entradas analógicas, para prevenir este inconveniente se plantea el uso de amplificadores operacionales en modo seguidor de tensión o adaptador de impedancia (uno para cada variable a medir). El segundo aspecto es el rango de medición de las entradas analógicas en la tarjeta principal, puesto que estas soportan hasta 5V DC como voltaje máximo de entrada, y en la planta se tiene hasta 10 V DC, entonces se opta por colocar un divisor de tensión a la salida del seguidor de tensión, el divisor estará conformado por dos resistencias de igual valor, con lo que se divide el voltaje entregado por el seguidor de tensión a la mitad, luego por software se compensa esta división multiplicando el voltaje medido por dos (2). El esquema del sensor-transmisor se muestra en la Figura 1.39.

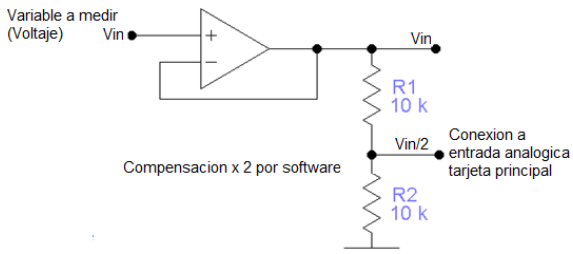


Figura 1.39. Esquema sensor-transmisor planta circuitos RC.

Cabe recordar que el amplificador en modo seguidor es básicamente un amplificador no inversor cuya resistencia de conexión a tierra tiene un valor infinito, la resistencia de realimentación es cero y la ganancia es la unidad (1), es decir el voltaje a la entrada es igual al de salida.

La característica principal de este amplificador es que tiene una impedancia de entrada Zin muy elevada, y una impedancia de salida Zout muy pequeña. Por este motivo se utiliza principalmente para aislar dos circuitos, de manera que el

segundo no resulte una carga para el primero, pues la impedancia vista será la altísima Zin del operacional, a este proceso se le conoce como “adaptación de impedancias”.

La referencia de amplificador operacional, que se elige es el LM358, ya que trae dos (2) amplificadores operacionales en un empaquetado tipo DIP de ocho (8) pines, las características más relevantes se muestran a continuación, así como el diagrama del pines del mismo.

Características importantes del amplificador LM358:

- Voltaje alimentación 4 - 32 VDC
- Contiene dos amplificadores empaquetados en uno
- Ganancia unitaria compensada por efectos de temperatura
- Alta impedancia de entrada
- Bajo consumo de corriente de entrada < 500uA

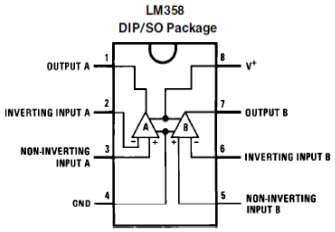


Figura 1.40. Diagrama de pines LM358.

1.2.2.5 Reconocimiento de la planta de Circuitos RC

Acorde con lo explicado en la sección 1.1.4 Sistema para detección de plantas, para que la planta sea detectada esta debe generar un código binario de tres (3) bits, se genera el código 001 colocando los pines P_TES_0 = 0, P_TES_1 = 0, P_TES_2 = 1, teniendo como referencia que 0 = conexión a tierra y 1 = conexión a +5VDC, ver Figura 1.41.

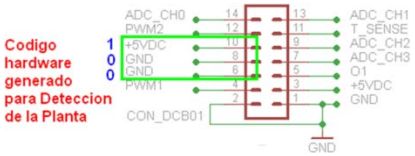


Figura 1.41. Código hardware para detección de planta de circuíos RC.

1.2.2.6 Diagrama esquemático y foto real de la planta de circuitos RC

El diagrama esquemático completo de la electrónica de la planta de circuitos RC se encuentra en la Figura 1.42, en ella se muestran los bloques principales, etapa de potencia (actuador), mallas RC, sensores, jumpers para configuración del sistema y conector a la tarjeta principal.

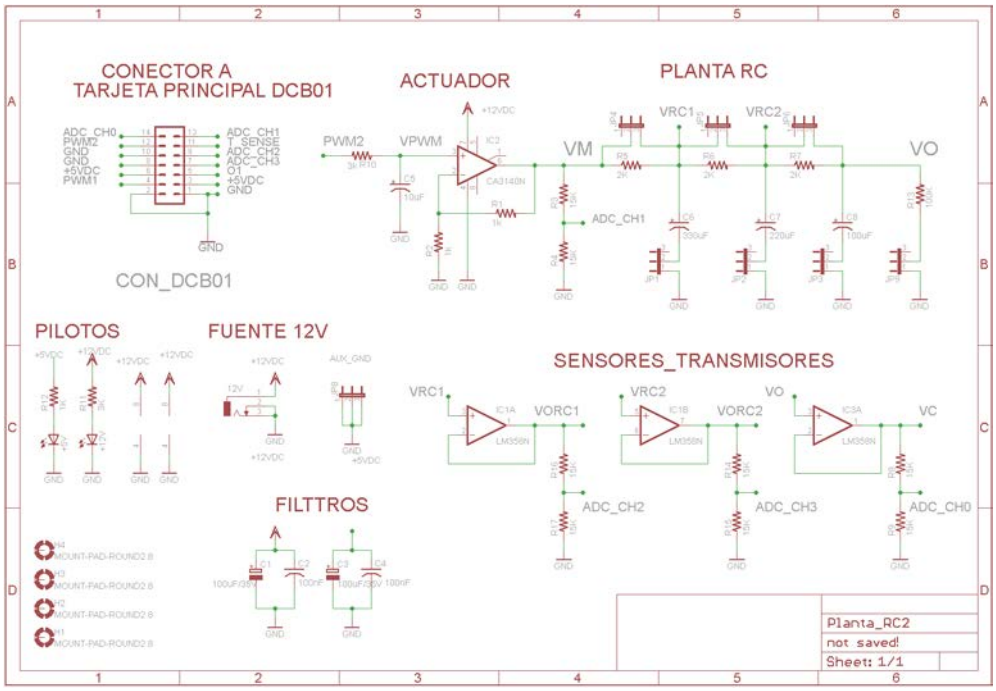


Figura 1.42. Diagrama esquemático planta de circuitos RC.

La Figura 1.43 muestra las partes de la tarjeta de la planta de circuitos RC.

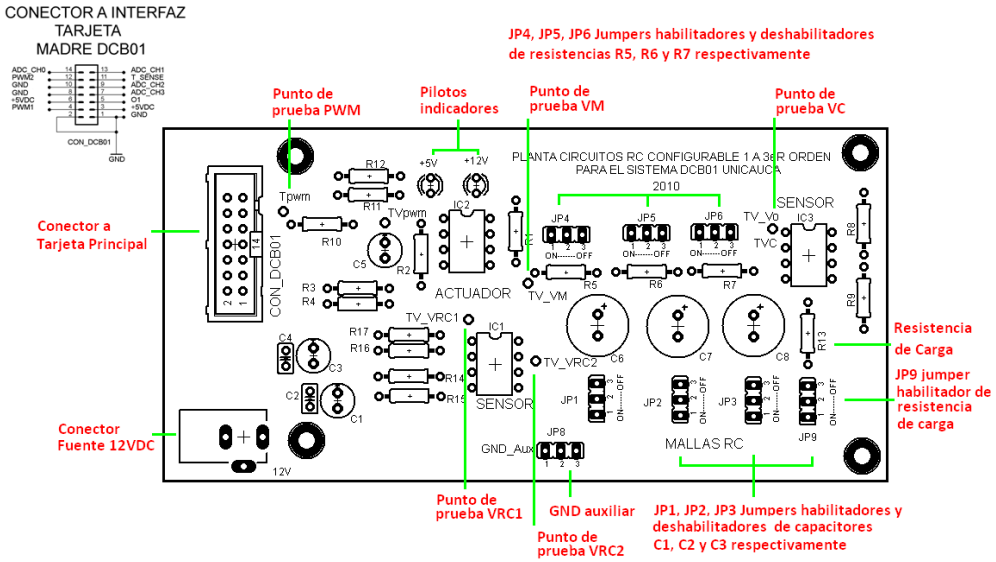


Figura 1.43. Partes planta de circuitos RC.

La Figura 1.44 muestra una imagen de la planta de circuitos diseñada

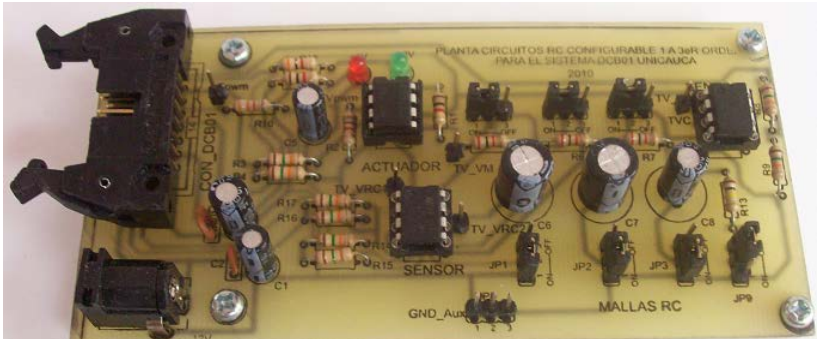


Figura 1.44. Imagen real de la planta de circuitos RC

Las características de la planta de circuitos RC son:

Tabla 1.11. Características planta de circuitos RC

PARÁMETRO	DETALLE
Alimentación	12 VDC
Orden del sistema	1er, 2do y 3er Orden Configurable por Hardware mediante jumpers
Rango Voltaje de operación(Variable controlada)	0 a 10 VDC
Spam	10 Voltios
Sistema Sensor-Transmisor	Amplificador Operacional, como seguidor de tensión o adaptador de impedancias más divisor de tensión.
Señal para generar esfuerzo de control	PWM 3KHZ
Variables Medibles	Voltaje de entrada (variable manipulada). Voltaje malla 1, malla2 y malla3, (variable controlada) Señal PWM que genera el esfuerzo de control, se puede medir la frecuencia mediante osciloscopio.
Disturbio experimental	Disturbio por cambios en la carga a la salida de la planta de forma manual mediante jumper. Posibilidad de implementación de disturbio por software limitando el esfuerzo de control.

Equation Section 2

2 INTERCOMUNICACIÓN A NIVEL SOFTWARE DEL SISTEMA

Como se mencionó anteriormente, el *sistema para la implementación de controladores digitales* se compone de una parte hardware y una parte software, en el capítulo 1 Diseño e implementación del hardware del sistema, se trató lo referente a diseño hardware del sistema (tarjeta principal, tarjeta de planta de temperatura y tarjeta de planta de circuitos RC), en este capítulo se tratara aspectos de diseño del software que complementa la funcionalidad del sistema.

Para presentar la información de modo organizado este capítulo se divide en los siguientes temas principales:

- Estructura *firmware* y lenguaje de programación de la tarjeta principal.
- Configuración de recursos del microcontrolador de la tarjeta principal.
- Comunicación USB entre tarjeta principal y Matlab bajo Windows XP.
- Comunicación entre tarjeta principal, plantas experimentales y pantalla LCD.
- Integración de *Bootloader* USB para programación de la tarjeta principal.

2.1 Estructura firmware y lenguaje de programación de la tarjeta principal.

Un planteamiento para el diseño del sistema es que sea modular, de modo que permita realizar diferentes actividades experimentales como complemento a los procesos educativos, es por esto que el sistema debe ser flexible tanto a nivel hardware como software. Para tener flexibilidad en el software se debe permitir que el usuario pueda crear sus propios programas y embeberlos en la tarjeta principal, por esta razón se desarrolló una plantilla del *firmware* para el microcontrolador de la tarjeta principal. La plantilla que se creó tiene todas las herramientas necesarias para establecer comunicación y control con las plantas, así como la comunicación con un PC, esta se diseñó de manera que el usuario pueda hacer uso de todos los recursos del sistema con la menor cantidad de código posible, por esta razón se crean librerías con funciones específicas (adquisición de variables, manejo de pantalla LCD, generación de esfuerzo de control, envió y recepción de datos por USB entre otras), que el usuario puede utilizar según los requerimientos de la práctica, la programación del microcontrolador de la tarjeta principal se realiza en lenguaje C, con el estándar ANSI C reducido característico en los compiladores de C para microcontroladores.

En la programación (generación del código o *firmware*) del microcontrolador 18F4550 de la tarjeta principal, se utilizan las herramientas Mplab y el compilador de CCS. Mplab de Microchip es una interfaz de usuario gratuita que permite crear, editar, compilar y depurar código para todos los microcontroladores PIC; Mplab IDE por si sola permite la programación en lenguaje ensamblador, pero también

permite agregar alternativas de programación mediante la instalación y asociación de compiladores como MC18, picbasic, compiladores de CCS, PIC C Hitech; Para el presente proyecto se integra Mplab con el compilador PCWHD de CCS para permitir la programación en lenguaje C. Pese a que el compilador de CCS tiene su propia IDE, Mplab presenta mayores ventajas en aspectos como enlace directo con programadores hardware (picstarplus, pickit2, pickit3, ICD2, ICD3), manejo de archivos, simulación, emulación y depuración, por otro lado el compilador de CCS es una herramienta muy versátil en la programación de microcontroladores en lenguaje C, entonces al unir estos dos componentes se obtiene una potente herramienta.

La integración y uso básico de estas herramientas se trata en detalle en el anexo manual de usuario, por ahora es importante resaltar que:

- El método de trabajo es por proyectos.
- La plantilla es un proyecto que está compuesto por múltiples archivos cada uno con una función específica.

La ventana típica y la estructura del proyecto (plantilla), en Mplab se muestran en la Figura 2.1.

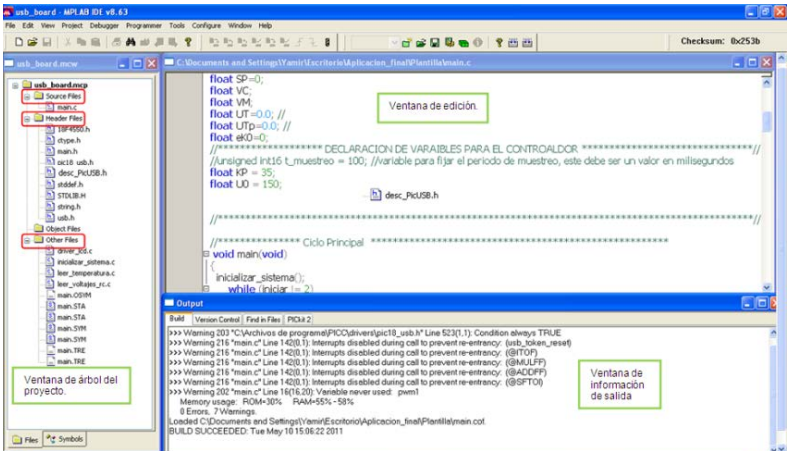


Figura 2.1. Estructura de un proyecto en Mplab de la plantilla del *firmware* de la tarjeta principal.

De la Figura 2.1 se destaca la ventana del árbol del proyecto, es la ventana vertical del lado izquierdo, en ella se muestran los diferentes archivos que componen el proyecto (plantilla), hay tres partes importantes, la primera es la sección *suorce files* donde está el archivo principal del proyecto (*main.c*) este es el archivo que se encarga de llamar a los otros según se requiera y donde el usuario creara su código, luego está la sección *header files* o archivos de cabecera, allí se incluyen archivos con extensión *.h,* como librerías de comunicación USB, declaración de constantes y definiciones para trabajar con el microcontrolador PIC 18F4550 y por ultimo *other files* en esta sección están las librerías que contienen funciones específicas que realizan tereas tales como: adquisición de variables de

las plantas (temperatura, voltajes), manejo pantalla LCD, configuración del sistema. En la Figura 2.2, se detalla el árbol del proyecto.

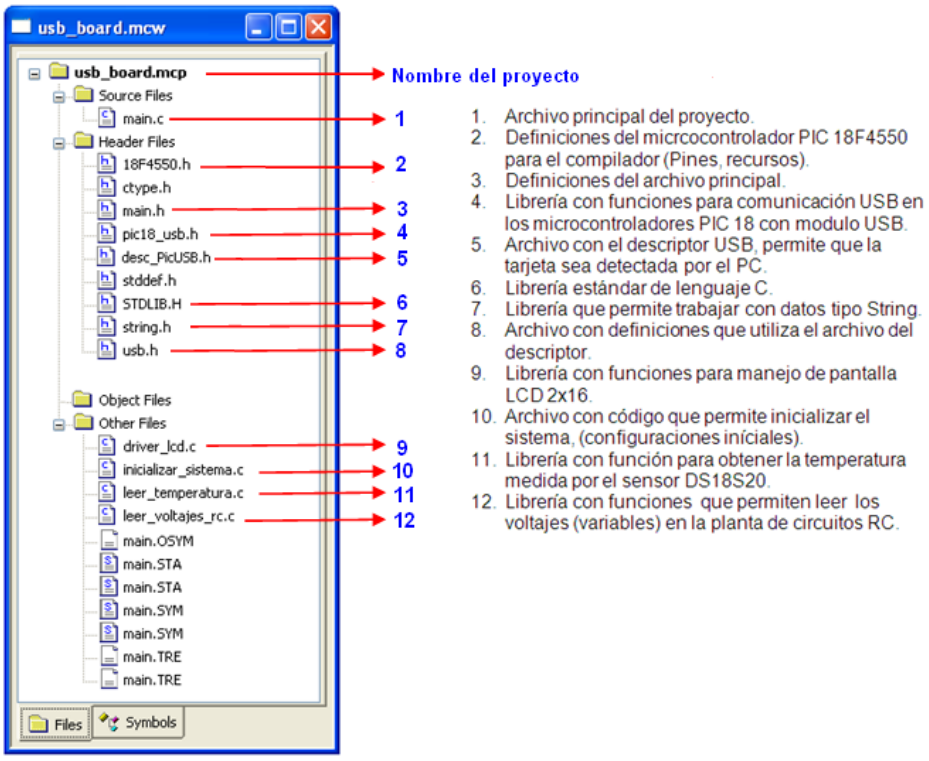


Figura 2.2. Árbol de archivos de la plantilla para generar código de la tarjeta principal.

Es importante tener en cuenta la información mencionada sobre este código (plantilla de la tarjeta principal), de aquí en adelante se explicara sobre el diseño y funcionalidad del contenido de los archivos relevantes para el funcionamiento del sistema, archivos que el usuario puede utilizar para realizar sus prácticas.

2.2 Configuración de recursos del microcontrolador de la tarjeta principal

Para que la tarjeta principal tenga las funcionalidades especificadas (ver capítulo 1 Tabla 1.4. Características tarjeta principal), lo primero que se debe hacer es configurar el microcontrolador y sus recursos, la tarea es colocar líneas de código y definiciones para que el compilador entienda como se quiere operar con él, que recursos se utilizaran y sus respectivas configuraciones. Los aspectos a configurar son:

- Definir frecuencia de trabajo.
- Resolución del conversor AD.
- Definición de constantes para el compilador.
- Configuración de entradas y salidas digitales.

- Configuración de entradas analógicas.
- Habilitar módulos PWM y configuración de frecuencia de operación de los mismos.
- Configuración de comunicación USB.
- Inicializar periféricos (pantalla LCD, sensor de temperatura DS1820)

Gran parte de la configuración de los ítems mencionados anteriormente para el microcontrolador se realiza mediante la herramienta *PIC Wizard*, esta es una aplicación muy completa que permite realizar una configuración asistida de los recursos del Microcontrolador, se puede acceder a la aplicación desde la IDE del compilador PCWHD CCS, a continuación una figura de la aplicación PIC Wizard.

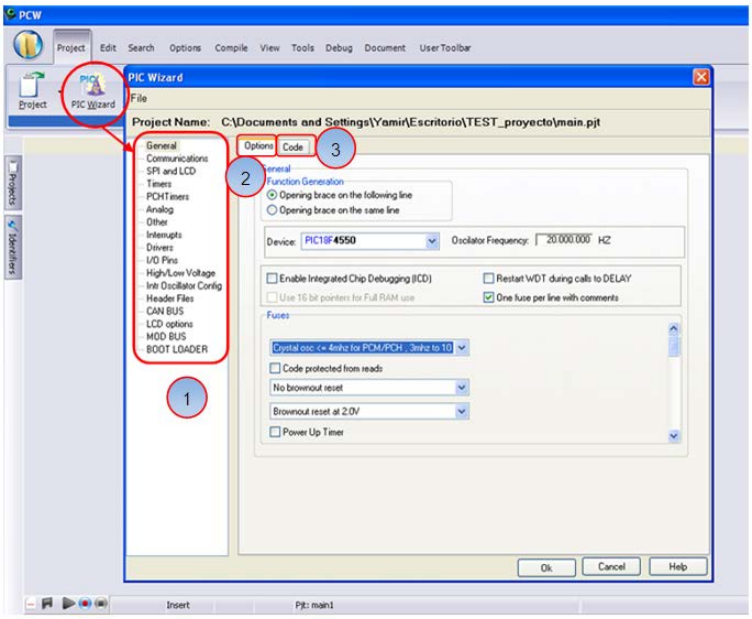


Figura 2.3. Interfaz aplicación PIC Wizard del compilador PCWHD.

Al dar clic sobre el botón *PIC Wizard* en la ventana principal del IDE del compilador PCWHD se abre la ventana de la aplicación, en ella se puede destacar tres partes importantes:

- Barra de módulos (número 1): en esta sección están listados los recursos que se pueden configurar en el microcontrolador.
- Pestaña de Opciones (número 2): en esta sección se muestran las opciones de configuración, las opciones cambian según el ítem seleccionado en la barra de módulos.
- Pestaña de pre-visualización de código generado (número 3): en esta pestaña se muestra el código que automáticamente va generando la aplicación, según las configuraciones que se vayan realizando.

La siguiente figura muestra un ejemplo de la opción de configuración del conversor AD del microcontrolador.

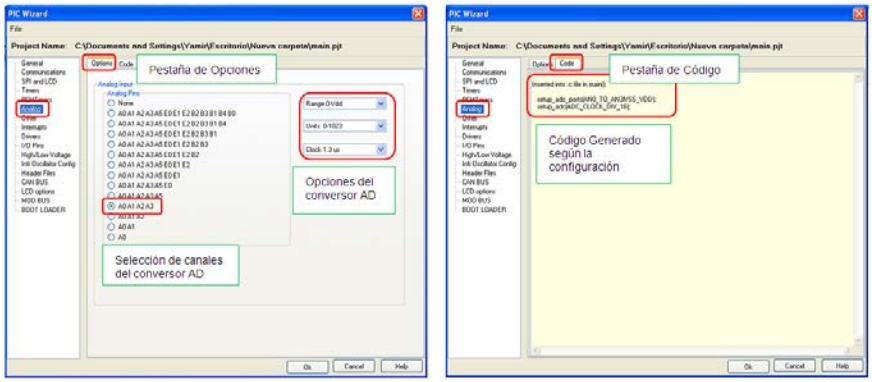


Figura 2.4. Generación de código para configuración de conversor AD mediante PIC Wizard.

Como resultado del uso de la herramienta *PIC Wizard* al finalizar se generan dos archivos, main.h y main.c, en el primero están todas las definiciones generadas según la configuración realizada para el microcontrolador, el segundo es el archivo principal, este tiene la estructura básica de un código en C, en este último también aparece el código de configuración de los recursos del microcontrolador y el espacio para que el usuario cree su código, dentro de un bucle infinito *while true*.

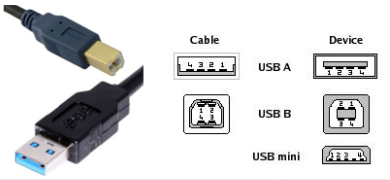
2.3 Comunicación USB entre tarjeta principal y Matlab bajo Windows XP

El proceso de integración de la comunicación USB al proyecto se desarrolló en dos fases importantes, la primera consistió en el estudio y recopilación de información sobre USB y luego la implementación de la comunicación.

2.3.1 Conceptos básicos sobre comunicación USB

Antes de comentar sobre el proceso realizado para diseñar y establecer la comunicación USB entre la tarjeta principal y Matlab, es indispensable tener en cuenta ciertos aspectos básicos sobre comunicación USB, a continuación se presentan términos y definiciones básicas sobre el bus USB orientado a comunicaciones con microcontroladores.

Interfaz física del bus USB: la interfaz física está formada por cuatro hilos dos para la alimentación y dos para datos (transmisión tipo diferencial), el bus provee su propia alimentación, esta es de 5VDC con una capacidad de corriente máxima de 500mA, a los conectores se les denominan de tipo A y de tipo B y mini USB, que se muestran en la siguiente figura.



Pin	Signal	Color	Description
1	VCC		+5V
2	D-		Data -
3	D+		Data +
4	GND		Ground

Figura 2.5. Conectores USB y señales

Host: es el dispositivo maestro que inicia y controla la comunicación (generalmente es el PC), se pueden conectar máximo 127 dispositivos al bus USB incluyendo al host.

Velocidades del bus USB: los datos siguientes son solo teóricos y de referencia ya que la velocidad real dependerá de la cantidad de dispositivos conectados al bus simultáneamente.

- **Low speed:** 1,5 Mbps. Soportado por las especificaciones 1.1, 2.0 y 3.0, es la velocidad utilizada por dispositivos como teclados, ratones, joystick, etc.
- **Full speed:** 12 Mbps. Soportado por las especificaciones 1.1, 2.0 y 3.0.
- **High speed:** 480 Mbps. Soportado por las especificaciones 2.0 y 3.0.
- **Super speed:** 5Gbps solo es soportado en dispositivos con especificación USB 3.0.

Enumeración: proceso mediante el cual el host obtiene información que permite detectar el dispositivo, dicha información se encuentra en el dispositivo en los llamados descriptores.

Descriptores: son datos que se guardan en la memoria no volátil del PIC y contienen datos tales como: VID, PID, consumo de corriente, tipo de transferencia que se va a utilizar, *endpoints* utilizados, versión de USB, clase utilizada por el dispositivo entre otros. El propósito de un descriptor es comunicar la identidad del periférico (microcontrolador), permitiendo que el host (PC) lo reconozca para así poder clasificarlo e identificarlo.

VID&PID (Vendor Identification & Producto Identification): son números en hexadecimal que permiten identificar el vendedor y el dispositivo respectivamente, por ejemplo VID&PID = 04D8h 00Bh identifica a la familia de los PIC 18FX5XX de microchip con módulo de comunicación USB.

Es importante definir valores de VID y PID diferentes a valores de dispositivos comerciales, ya que en caso de tener dos dispositivos diferentes con los mismos indicadores se creara un conflicto hardware con lo que ambos dispositivos dejan de funcionar.

***Endpoints* o Puntos terminales:** son buffers de almacenamiento de datos (bytes), alojados en la memoria de datos (RAM), según las especificaciones del bus USB un dispositivo puede soportar hasta 16 *endpoints*, numerados de 0 a 15, donde el *endpoint* 0 es obligatorio y está reservado para comandos de control en el proceso de enumeración, en el microcontrolador 18F4550 se configura un máximo de 3 *endpoints* (EP0, EP1 y EP2) ya que este tiene una memoria RAM (2KB) bastante limitada.

Pipes o tuberías: son una conexión lógica entre un *endpoint* y el software del controlador del host (enlace virtual o puente entre el host y el dispositivo, el microcontrolador en este caso), se producen tras el proceso de enumeración, las pipes se deben configurar con indicadores diferentes según su función y son asociadas a un *endpoint*, las pipes también definen el tamaño del paquete de bytes a transmitir y el tipo de comunicación.

En la siguiente figura se muestra el flujo de comunicación USB

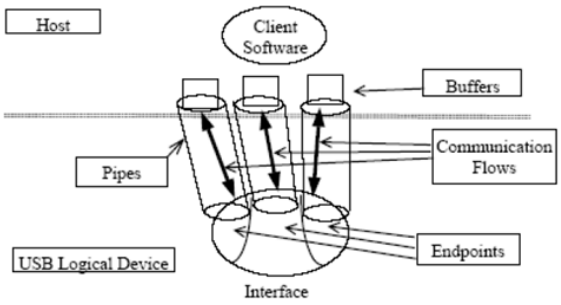


Figura 2.6. Flujo de comunicación USB.

Tipos de transferencias por USB: una transferencia se puede definir como el conjunto global de los datos que forman una comunicación USB, una transferencia está formada a su vez por una o varias transacciones que a su vez están formadas por diferentes paquetes de datos que contienen las tramas de una comunicación USB.
No existe un formato único de transferencia, la especificación USB permite cuatro tipos diferentes:

- Transferencias de control
- Transferencias masivas (Bulk Transfer)
- Transferencias isócronas
- Transferencias interruptivas

Los dispositivos usan uno o más tipos de transferencia, la de control es utilizada siempre por todos los dispositivos en el proceso de enumeración. Si se utiliza otra transferencia habrá que hacerlo en función del tipo y cantidad de datos a transmitir.

Clase de dispositivos USB: una clase es un grupo de dispositivos (o interfaces dentro de un dispositivo), con ciertas características en común. Típicamente, dos dispositivos pertenecen a la misma clase si ambos utilizan formatos similares en los datos que reciben o transmiten, o si ambos utilizan una misma forma de comunicarse con el sistema (por ejemplo, un teclado y un ratón por sus características pertenecerán a la misma clase la llamada *Human Interface Device*).

La especificación USB tiene muchas clases para facilitar la vida al desarrollador de dispositivos, las clases más utilizadas con microcontroladores son:

- **HID** (Human Interface Device).
- **MSD** (Mass Storage Device Class.
- **CDC** (Communications Device Class.
- **Custom Class**(clase personalizada).

Drivers y librerías USB para microcontroladores PIC: los drivers permiten que el sistema operativo pueda detectar clasificar y operar con un dispositivo USB, las clases HID, MSD y CDC tienen la ventaja de que no necesitan drivers, generalmente ya se encuentran preinstalados en el sistema operativo, si el dispositivo no encaja en ninguna de estas tres clases quiere decir que pertenece a la clase Custom Class, para esta clase se requiere de un driver personalizado. Diseñar un driver desde cero es algo complejo, sin embargo existen drivers personalizados que se pueden utilizar en los proyectos de diseño, algunos de ellos son los siguientes:

- **Mchpusb:** proporcionado por la empresa fabricante Microchip, está conformado por dos archivos, **mchpusb.sys** que es el driver en sí y el segundo archivo **mchpusb.inf** el cual contiene información del driver necesaria para el proceso de enumeración.
- **WinUSB :** driver de Microsoft al igual que Microchip, Microsoft proporciona los archivos **Winusb.sys** y **Winusb.ini**

Las librerías o bibliotecas facilitan por medio de funciones la comunicación entre la aplicación de escritorio y el dispositivo, las más utilizadas con microcontroladores son:

- **Mpusbapi:** librería que ofrece Microchip, está creada y compilada en el lenguaje Borland C. Microchip facilita su código fuente, por lo que hay varias formas de utilizarla. Si se utiliza el mismo IDE de Borland para crear la aplicación de escritorio, simplemente hay que añadir la librería (el archivo mpusbapi.lib) al proyecto como una librería más, si se utiliza otro compilador diferente se debe compilar de nuevo los archivos fuente para obtener una nueva versión que sea compatible con el compilador utilizado, otra opción es añadirla al proyecto de forma dinámica, para ello solo se necesita el archivo con extensión .dll (mpusbapi.dll) proporcionado también

por Microchip, de esta forma si se utiliza plataformas diferentes como Visual, .NET, LabVIEW, Matlab, RealBAsic, etc. No hay que modificar el código fuente de la DLL, simplemente importan las funciones públicas de la DLL desde la aplicación de escritorio.

- **libUSB:** librería de código abierto con licencia GNU, se puede instalar en múltiples sistemas operativos como Linux, MAC y otros, incluyendo Windows a través de su versión libusb-win32.

Con los anteriores conceptos claros, se prosigue con el diseño de la comunicación, lo primero es elegir el tipo de comunicación (bulk, isócrona o interruptiva; hay que recordar que la comunicación de control es obligatoria), se opta por una comunicación del tipo bulk, dado que presenta la ventaja de poder obtener altas velocidades de transferencia sin pérdida de información, además el tamaño del paquete puede ser de hasta 64KBytes, la clase seleccionada es una tipo Custom Class, esto debido a que Microchip y el compilador CCS ofrecen varios ejemplos de comunicación con transferencia tipo bulk transfer y esta clase.

Luego de la selección del tipo de comunicación se procede con la implementación de la comunicación, en este proceso se tienen en cuenta los siguientes aspectos:

- Diseño y configuración de la conexión USB desde el microcontrolador.
- Diseño y configuración de la comunicación USB desde el PC (Host).
- Estandarización de los datos a comunicar entre el host (PC) y la tarjeta principal (Microcontrolador).

2.4 Diseño y configuración de la conexión USB desde el microcontrolador

Se hace una revisión de las herramientas que ofrece el compilador PCWHD de CCS para establecer una comunicación USB con los PIC 18Fx5xx con módulo USB. En el microcontrolador se debe configurar: el tipo de comunicación, la clase del dispositivo, el tamaño del paquete a transmitir (*endpoints*), información del descriptor entre otros.

En la plantilla del microcontrolador los archivos relacionados con la comunicación USB son: *desc_PicUSB.h, usb.h, pic18_usb.h*, estos archivos se pueden encontrar en el árbol del proyecto en la sección *header files* (Figura 2.2. Árbol de archivos de la plantilla para generar código de la tarjeta principal.), la función de cada uno de ellos se describe a continuación.

desc_PicUSB.h: es el archivo descriptor, el cual está basado en uno de los ejemplos que trae el compilador, el archivo original usb_desc_scope.h es propiedad del compilador de CCS y se puede encontrar en la carpeta de ejemplos que trae el compilador, este archivo se puede personalizar según la aplicación que se quiera desarrollar, en el proyecto se utilizó la aplicación *enumer_picusb v3.1* que permite la configuración automática del descriptor, además se puede generar el archivo *mcphusb.inf* que permite la detección del dispositivo desde el PC, la

siguiente figura muestra la aplicación mencionada y los campos que el usuario puede modificar para crear un descriptor personalizado.

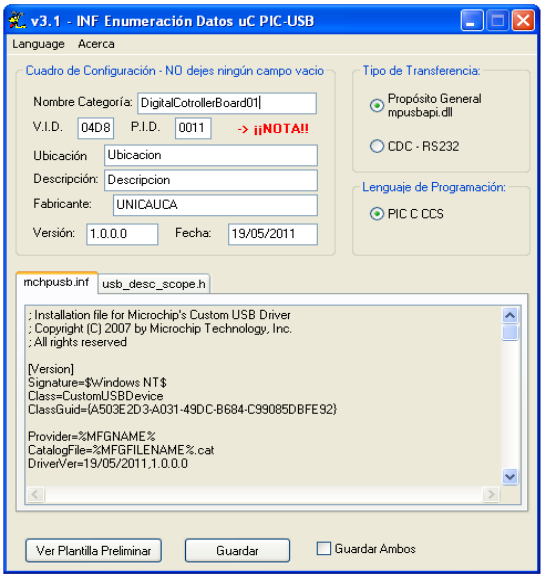


Figura 2.7. Ventana aplicación para configuración de enumerador para PIC 18F4550 y generación de *mcphusb.inf* para detección desde el PC.

usb.h: este es un archivo que contiene solo definiciones que son requeridas por el archivo del descriptor, y se encuentra en la carpeta drivers del compilador PCWHD de CCS.

pic18_usb.h: es uno de los archivos más importantes, contiene las funciones que permiten al usuario establecer comunicación USB entre el microcontrolador de la tarjeta principal y el host (PC), la siguiente tabla indica las funciones así como una descripción de las mismas.

Tabla 2.1. Funciones del compilador CCS para comunicación USB entre el microcontrolador PIC 18F4550 y el host.

Función	Declaración requerida	Descripción
void usb_init();	Ninguna	Inicializa comunicación USB, conecta el modulo al bus, permite interrupciones. Es la primera función USB que se debe usar
Int1 usb_put_packet(int8 endpoint,int*buffer,int16 buffersize, bit_toogle);	Declarar *endpoint* Tamaño buffer. Nombre buffer	Envía un paquete al host, el paquete de datos es un vector de bytes de tamaño buffersize.
void usb_kbhit();	Ninguna	Permite detectar si hay información en el *endpoint* de

		entrada (recepción de información), si es el caso se debe leer dicha información.
int16 usb_rx_packet_size(int 8 endpoint);	Declarar endpoint	Retorna el tamaño del paquete de entrada.
int16 usb_get_packet(int8 endpoint, int8 * buffer, int16 buffersize,);	Declarar endpoint Tamaño buffer. Nombre buffer	Recoge la información de entrada enviada por el host, la función *usb_kbhit()*; debe ser llamada antes para detectar si hay información que leer.
void usb_detach();	Ninguna	Desconecta el modulo USB del microcontrolador, libera los pines D+ y D-.
void usb_attached();	Ninguna	Permite detectar si el microcontrolador está conectado a un cable USB.
void usb_task();	Ninguna	Permite sensar continuamente la conexión USB llamando a las funciones *USB_detach(); y usb_attached();*
usb_wait_for_enumeration();	Ninguna	Espera a que el host (PC) reconozca el dispositivo y retorna true.

Para garantizar una comunicación constante entre el host y el microcontrolador, se opta por utilizar una interrupción por desborde de un temporizador interno del microcontrolador, este es un recurso que permite generar una interrupción cada determinado tiempo en el microcontrolador, específicamente se configuró la interrupción para que se genere cada 10ms, la tarea asignada a dicho evento es la de recepción y transmisión de datos a través de las pipes de comunicación.

El código correspondiente a la función de atención de la interrupción se encuentra en el archivo *main.c* al final.

2.5 Diseño y configuración de la comunicación USB desde el PC (host)

Para la comunicación USB desde el PC se necesitan el driver y la librería de comunicación, que a continuación se describen:

mchpusb.sys: es el driver que permite que el sistema operativo reconozca al microcontrolador.

mchpusb.inf: (creado con la aplicación *enumer_picusb v3.1*) este es un archivo que contiene información sobre el dispositivo, esta información es necesaria para

que el driver *mchpusb.sys* pueda configurar correctamente el dispositivo, los parámetros más importantes que contiene este archivo son: nombre del dispositivo, VID, PID, fabricante y descripción del dispositivo.

mpusbapi.dll: es la librería de intercambio dinámico de datos, permite establecer comunicación USB entre el PC y el microcontrolador utilizando diferentes lenguajes de programación, como en el caso de este proyecto en el que se hace comunicación a través de Matlab.

La integración de la librería *mpusbapi.dll* se hace gracias a la opción *loadlibrary* de Matlab, que permite integrar librerías generadas con lenguajes diferentes al de Matlab y hacer uso de ellas.

La *mpusbapi.dll* contiene diferentes funciones que permiten establecer la comunicación USB, a continuación se presenta una tabla con dichas funciones y una breve descripción de cada una, para profundizar en el manejo de la *mpusbapi.dll* remítase al anexo manual de usuario.

Tabla 2.2. Funciones de la librería *mpusbapi.dll*

Función	Descripción
MPUSBGetDLLVersion(void)	Devuelve la versión de la dll en un numero de 32 bits, ej (1.0.0.0.0)
MPUSBGetDeviceCount(vid_pid)	Devuelve el número del dispositivo según el vid_pid que se le pase.
MPUSBOpen(instance, pvid_pid, pep, dwdir, dwreserved)	Abre los *endpoints* y las pipes para establecer comunicación con el dispositivo (microcontrolador), los parámetros son: instance: debe ser el número del dispositivo. pvid_pid: PID&VID del dispositivo objetivo. pep: número del *endpoint* que se va a abrir. dwdir: dirección del *endpoint*. dwerserved: actualmente no tiene una función específica.
MPUSBRead(handle, pdata, dwlen, plength, dwmilliseconds)	Permite leer datos enviados por el dispositivo (microcontrolador). handle: identifica el pipe del *endpoint* que se va a leer. pdata: puntero al buffer que contiene los datos que se van a leer. dwlen: número de bytes que se quieren leer. plenght: indica el número de bytes leídos. dwmilliseconds: intervalo de time-out, tiempo que toma en leer los datos, configurable según el tamaño del paquete.
MPUSBWrite(handle, pdata, dwlen, plength, dwmilliseconds)	Permite enviar datos hacia el dispositivo (microcontrolador). handle: identifica el pipe del endpoint que se va a escribir. pdata: puntero al buffer que contiene los datos que se van a escribir. dwlen: número de bytes que se quieren escribir en la

	pipe. plenght: indica el número de bytes que se escriben al llamar la función. dwmilliseconds: intervalo de time-out, tiempo que toma en escribir los datos, configurable según el tamaño del paquete.
MPUSBReadInt(handle, pdata, dwlen, plength, dwmilliseconds)	Similar a MPUSBRead, pero utilizada para comunicaciones por interrupciones.
MPUSBClose(handle)	Cierra las pipes de comunicación.

Para implementar la comunicación USB entre Matlab y el microcontrolador PIC 18F4550, se debe crear un archivo para edición de código de Matlab, es decir un archivo-m, en la misma carpeta donde se tiene este archivo deben estar los archivos *mpusbapi.dll* y *_mpusbapi.h*, el primero como ya se mencionó contiene las funciones que permiten establecer la comunicación y el segundo es un archivo con definiciones requeridas por la dll.

A continuación se describe de manera sencilla el proceso para configurar y establecer la comunicación USB en Matlab.

El primer paso es realizar el enlace o cargar la librería a Matlab:

```
clear all; close all;
loadlibrary mpusbapi _mpusbapi.h alias libreria
```

Luego se definen los vectores para transmisión y recepción de datos, estos deben ser de tipo entero de 8 bits (bytes).

```
data_in = eye(1,16,'uint8'); % Vector para datos enviados desde el microcontrolador
data_out = eye(1,16,'uint8'); % Vector para datos enviados hacia él microcontrolador.
```

En el ejemplo anterior se definen dos vectores de 16 bytes de tamaño, uno para transmisión y uno para recepción de datos.

Posteriormente se debe configurar la comunicación, definiendo los valores de VID&PID y creando las pipes de comunicación y también definiendo el endpoint.

```
vid_pid_norm = libpointer('int8Ptr',[uint8('vid_04d8&pid_000b') 0]);
out_pipe = libpointer('int8Ptr',[uint8('\MCHP_EP1') 0]);
in_pipe = libpointer('int8Ptr',[uint8('\MCHP_EP1') 0]);
```

A este punto ya se tiene configurada la comunicación; el proceso continúa con la detección del dispositivo y posteriormente la apertura de los canales de comunicación (pipes) para su posterior uso (transmitir y recibir datos).

Para que Matlab pueda detectar si el microcontrolador está conectado al bus, este debe estar conectado y previamente haber instalado los drivers correspondientes para que el sistema operativo lo reconozca, los drivers son el *mchpusb.sys* y el *mchpusb.inf*, el proceso de instalación de la tarjeta principal y los drivers del microcontrolador se detalla en el anexo manual de usuario.
La detección se realiza mediante la siguiente línea de código:

```
[conectado] = calllib('libreria','MPUSBGetDeviceCount',vid_pid_norm)
```

La función MPUSBGetDeviceCount retorna el valor 1 en caso que el dispositivo (microcontrolador de tarjeta principal) sea detectado.

Ahora solo queda preguntar si el dispositivo está conectado, entonces se deben abrir los canales de comunicación y dejarlos listos para la transferencia de datos.

```
if conectado == 1
[my_out_pipe] = calllib('libreria', 'MPUSBOpen',uint8 (0), vid_pid_norm, out_pipe, uint8(0), uint8 (0)); % Se abre el túnel de %envio de datos hacia el microcontrolador.
[my_in_pipe] = calllib('libreria', 'MPUSBOpen',uint8 (0), vid_pid_norm, in_pipe, uint8 (1), uint8 (0)); % Se abre el túnel de %recepción de datos provenientes del microcontrolador.
```

Con los canales de comunicación se puede leer y escribir en el bus, para ello se utilizan las funciones de lectura y escritura que provee la *mpusbapil.dll*

```
calllib('libreria', 'MPUSBWrite',my_out_pipe, data_out, uint8(16), uint8(16), uint8(2)); % Se envía el paquete de datos hacia el PIC
[aa,bb,data_in,dd] = calllib('libreria', 'MPUSBRead',my_in_pipe, data_in, uint8(16), uint8(16), uint8(2)); % Se recibe el paquete de % datos desde el PIC
```

El proceso de lectura y escritura de datos se puede repetir cuantas veces se requiera, pero es indispensable que para finalizar la comunicación se deben cerrar los canales de comunicación (pipes) y descargar la librería, ya que de lo contrario el sistema operativo mantendrá los canales abiertos agotando el ancho de banda del bus, para ello se debe hacer uso de las siguientes líneas de código.

```
calllib('libreria', 'MPUSBClose', my_in_pipe);  % Se cierra la pipe de recepción.
calllib('libreria', 'MPUSBClose', my_out_pipe); % Se cierra la pipe de transmisión.
end
unloadlibrary libreria % se descarga la librería mpusbapi.dll
```

2.6 Estandarización de los datos a comunicar entre el host (PC) y la tarjeta principal (Microcontrolador)

Como se menciona en la descripción anterior, la comunicación USB por el método *bulktransfer* se basa en la transmisión de paquetes de bytes (vectores), esto tiene dos aspectos a tener en cuenta, el primero es que hay inconveniente para transmitir algunos tipos de variable como flotantes y enteros de valor mayor a 255, el segundo aspecto es que se debe estandarizar la información a transmitir.

Ya que variables flotantes del sistema como la temperatura y el voltaje no superan el valor de 100,00 (85 ^{o}C es el valor máximo en el caso de la planta de temperatura, y 10.00 Voltios en la planta de circuitos RC), la solución propuesta para transmitir variables de este tipo es tomar la parte entera más dos decimales y enviarlos en dos bytes, asumiendo un valor máximo de 99.99, se envía parte entera y parte decimal. Se debe realizar un proceso tanto en transmisión como en recepción para adecuar los datos, los siguientes ejemplos muestran el proceso matemático que se realiza según los diferentes casos:

Ejemplo: enviar variable flotante desde el microcontrolador a Matlab

En transmisión (microcontrolador):

Temperatura = 57.3867; // 57,3867 ^{o}C
t_entera = (int8) (Temperatura); //t_entera = 57
*t_decimal = (int8) ((Temperatura – t_entera) *100); //t_decimal = 38*

En recepción (Matlab):
*Temperatura = double (t_entera) + double (t_decimal * 0.01)*

Ejemplo: enviar variable double desde el Matlab al microcontrolador

En transmisión (Matlab):
Set_Point = 42.540; % 42.540^{o}C
Sp_entero = uint8 (Set_point – mod (Set_point,1)); % Sp_entero = 42

Para generar la parte entera se resta el modulo (parte decimal) del valor doble en el caso de las variables de Matlab.

Sp_decimal = uint8 (set_point – Sp_entero * 100); % Sp_decimal = 54

La parte decimal se envía como el entero generado de la resta entre el valor double y el valor entero, esto multiplicado por 100.

En recepción (microcontrolador):

*SP = float (Sp_entero) + float (Sp_decimal *0.01)*

Luego la estandarización de los datos a comunicar es muy importante, permite al usuario tener un orden para enviar y recibir datos tanto desde el microcontrolador de la tarjeta principal como desde Matlab.

Hay que recordar que los datos van empaquetados en vectores de bytes por lo que hay que tener en cuenta que en lenguaje C (lenguaje de programación tarjeta principal) los elementos de un vector (bytes en este caso) se enumeran desde 0 hasta n-1, por otro lado en Matlab los vectores se enumeran desde 1 hasta n, donde n es la cantidad de elementos del vector.

En la siguiente figura se muestra la organización de la información entre los vectores de recepción en Matlab y el de transmisión en la tarjeta principal:

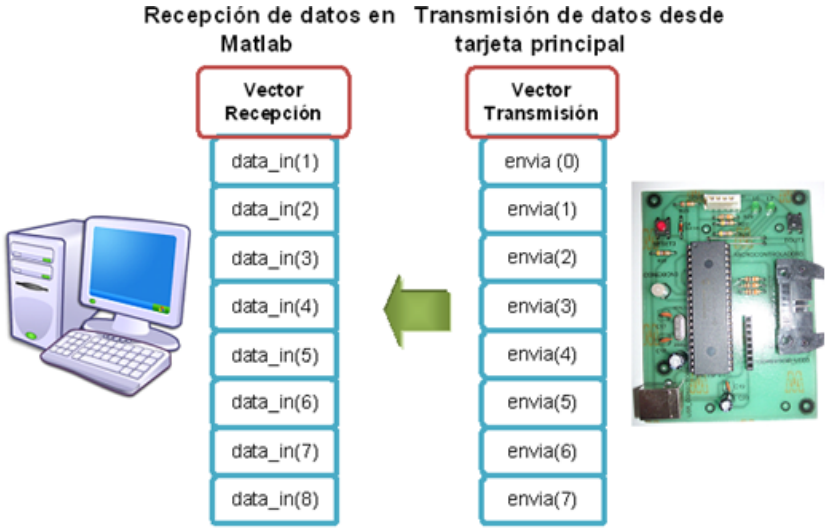


Figura 2.8. Transmisión desde tarjeta principal y recepción en Matlab.

Tabla 2.3. Descripción contenido información elementos de vectores de comunicación tarjeta principal-Matlab.

Elemento de Vector	**Descripción de la información**
envía(0) → data_in(1)	Tipo de planta conectada, variable *Tplanta* en el microcontrolador, (2=Planta temperatura, 4= planta circuitos RC).
envía(1) → data_in(2)	Parte entera de la variable controlada, VC en el microcontrolador.
envía(2) → data_in(3)	Parte decimal de la variable controlada VC en el microcontrolador.
envía(3) → data_in(4)	Parte entera de la variable manipulada, VM en el microcontrolador.
envía(4) → data_in(5)	Parte decimal de la variable manipulada, VM en el microcontrolador.

envía(5) → data_in(6)	Parte entera de variación de PWM, UTp en el microcontrolador.
envía(6) → data_in(7)	Parte decimal de variación de PWM, UTp en el microcontrolador.
envía(7) → data_in(8)	Libre

Nota: Tanto la parte entera como la parte decimal son dos dígitos.

En la siguiente figura se muestra la organización de la información entre los vectores de transmisión en Matlab y el de recepción en la tarjeta principal:

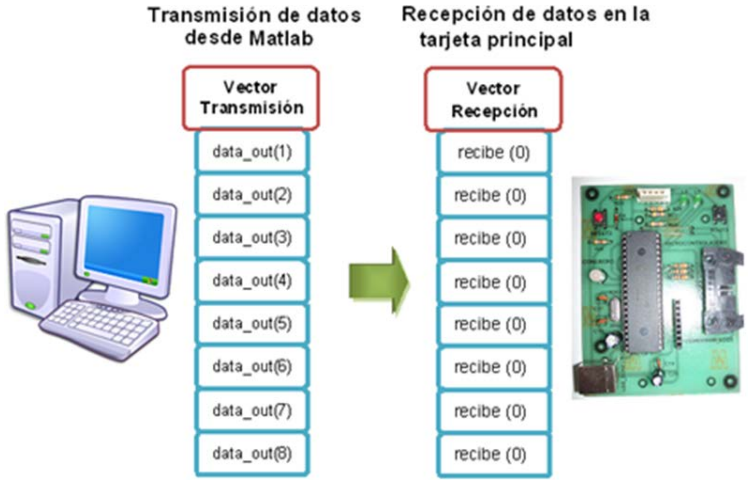


Figura 2.9. Transmisión Matlab recepción en tarjeta principal.

Tabla 2.4. Descripción contenido información elementos de vectores de comunicación Matlab-tarjeta principal.

Elemento de Vector	Descripción de la información
data_out(1) → recibe(0)	Valor que permite iniciar el sistema en la tarjeta principal, este valor se pasa a la variable *iniciar* en el microcontrolador, se debe enviar el valor 2 para que la tarjeta inicie.
data_out(2) → recibe(1)	Parte entera de la variable set point, en el microcontrolador se une con la parte decimal en la variable flotante SP.
data_out(3) → recibe(2)	Parte decimal de la variable set point, en el microcontrolador se une con la parte entera en la variable flotante SP.
data_out(4) → recibe(3)	Libre
data_out(5) → recibe(4)	Libre
data_out(6) → recibe(5)	Libre

data_out(7) → recibe(6)	Libre
data_out(8) → recibe(7)	Libre

2.7 Comunicación entre tarjeta principal, plantas experimentales y pantalla LCD.

En esta sección se describe el diseño software para la comunicación entre la tarjeta principal y las plantas de temperatura y circuitos RC. A continuación se describen las funciones (rutinas software) que permiten realizar acciones sobre las plantas, tales como: sensado de variables, generación del esfuerzo de control y detección de la planta, en la siguiente tabla se encuentran detalladas las acciones que se implementaron en el proyecto.

Tabla 2.5. Acciones que realiza la tarjeta principal sobre las plantas.

Acción	Dispositivo	Tipo
Detección de planta	Plantas de temperatura y circuitos RC	Lectura
Generar esfuerzo de control mediante señal PWM	Plantas de temperatura y circuitos RC	Salida
Leer temperatura medida por el sensor DS18S20	Planta de temperatura	Lectura
Leer voltaje salida (VC) en planta RC	Planta de circuitos RC	Lectura
Leer voltaje entrada (VM) en planta RC	Planta circuitos RC	Lectura
Imprimir mensajes en pantalla LCD	Pantalla LCD	Escritura

A continuación se presenta una descripción del código implementado que permite al usuario realizar las acciones mencionadas en la tabla anterior.

2.8 Detección de plantas

La acción de detectar las plantas se realiza por medio de la lectura del código generado por los pines P_TEST_0 a P_TEST_2 (ver Tabla 1.3. Descripción señales conector-tarjeta principal), estos pines son entradas digitales que pueden tomar valores lógicos 1 o 0, los tres pines en conjunto forman un numero de 3 bits (valores de 000 a 111 en binario ó 0 a 7 en decimal), cuando no hay nada conectado a la tarjeta principal estos pines entregan el código 000 y cuando se conecta una planta este código cambia por uno fijado según el hardware de la planta, la planta de temperatura genera el código 010 o 2 en decimal y la planta de circuitos el código 100 o 4 en decimal.
La detección de la planta se hace leyendo el número generado por los bits así:

Archivo: main.h

#define PTemp 2 // constante para la planta de temperatura,
#define PRc 4 // constante para la planta del circuito RC,
int8 Tplanta=0; // variable declarada para leer código de detección de plantas
Archivo: main.c
Tplanta = 7 & input_b(); // lectura del Puerto B y operación and con 7

La línea de código anterior, permite la lectura del código de tres bits que posibilita la identificación de la planta, se hace la operación *and* para obtener la información de los tres bits menos significativos de 8 que se leen del puerto.

	b7	b6	b5	b4	b3	b2	b1	b0
Lectura puerto B								
Numero 7	0	0	0	0	0	1	1	1
Resultado operación *and*	0	0	0	0	0	b2	b1	b0

Con la información de los tres bits de entrada ya se puede reconocer si se ha conectado una planta, según las siguientes condiciones:

- Si Tplanta es igual a 0 entonces no hay planta conectada.
- Si Tplanta es igual a 2 entonces planta de temperatura conectada.
- Si Tplanta es igual a 4 entonces planta de circuitos RC conectada.
- Si Tplanta es diferente de los anteriores valores entonces hay un error o puede ser otra planta en caso de diseños futuros, si se llega a agregar una nueva planta (diferente a temperatura y circuitos RC), esta debe generar su propio código y debe ser diferente de 2 y 4.

2.9 Generación de esfuerzo de control

La señal de control o esfuerzo de control se genera mediante una señal de PWM, el microcontrolador de la tarjeta principal (PIC 18F4550) tiene bloques internos que permiten generar hasta 4 señales de este tipo, se decide utilizar solo dos (PWM1 y PWM2), en la configuración inicial de recursos se fija la frecuencia de las dos salidas, esta se establece en un valor de 3Khz, se escoge una frecuencia alta para evitar inconvenientes con el periodo de muestreo de las variables de las plantas, dado que se debe cumplir que la frecuencia de PWM sea 5 veces mayor que el periodo de muestreo.

La resolución de la señal PWM está dada por los registros internos CCP1 y CCP2 (registros asociados a los módulos CCPX) del microcontrolador, estos son de 10 bits, entonces se tiene valores desde 0 hasta 1024 para un porcentaje en el ciclo útil de la señal del 0% al 100%, por razones de linealización y reducir carga en las operaciones de procesamiento del microcontrolador, se opta por dejar un rango de

0 a 1000 equivalente al porcentaje de 0% a 100%, las instrucciones para modificar el esfuerzo de control (cambiar el ciclo útil de la señal PWM) son:

Archivo: main.c

Declaraciones: *unsigned uint16 pwmX* (x = 1 o 2 según la señal de PWM)
float UT = 0;

Para asignar la señal de esfuerzo de control que se obtiene como resultado de un controlador (salida del controlador), se debe tener en cuenta las unidades del esfuerzo de control, por ejemplo:
Si la salida del controlador (esfuerzo de control), está directamente en valores de 0 a 1000, la asignación se hace de forma directa así:

set_pwmX_duty(pwmX);

Si por el contrario la salida del controlador está en porcentaje este se debe normalizar a valores del registro del módulo PWM, así:

*pwmX = (unsigned int16)(UT*10);*

Con la anterior línea se hace una conversión de porcentaje de 0% a 100% a un valor de 0 a 1000 que es el valor que luego se asigna al registro de módulo PWM así:

set_pwmX_duty(pwmX);

En caso que la salida del controlador este en otra unidad se debe realizar la conversión de dicha unidad a valores del registro de 0 a 1000.

2.10 Lectura de temperatura medida por el sensor DS1820

Para el manejo del sensor de temperatura se creó el archivo *leer_temperatura.c,* este archivo contiene una rutina que hace uso del pin T_SENSE (ver Tabla 1.3. Descripción señales conector-tarjeta principal) para establecer una comunicación por protocolo 1wire que es con el que opera el sensor y de este modo solicitar el valor de la temperatura sensada y digitalizada por el mismo.

El código es una adaptación de la librería para manejo del sensor DS18S20 de la página web del compilador PCWHD CCS, de manera que para el usuario el proceso interno de comunicación entre sensor y tarjeta principal sea transparente. Se diseñó una función que permite obtener la temperatura medida por el sensor, esta función tiene la siguiente estructura:

Archivo: leer_temperatura.c
Función: float var = leer_temperatura();

La anterior función solo retorna un elemento y no se le debe pasar nada, en la variable de retorno *var* se entrega al usuario el valor de la temperatura, previamente se debe declarar esta variable y tiene como condición estricta que debe ser de tipo flotante, ya que el sensor entrega valores de temperatura en un rango de 0 a 125 ºC con una precisión de ±0.5 ºC, así que es posible tener valores con decimales, como por ejemplo 43.15ºC.

2.11 Medición de voltajes de la planta de circuitos RC

Las variables de la planta de circuitos RC son voltajes, en esta planta se pueden medir cuatro voltajes diferentes:

- Voltaje de entrada al circuito RC (Vin, variable manipulada).
- Voltaje de salida malla 1(Vrc1, variable intermedia).
- Voltaje de salida malla 2(Vrc2, variable intermedia).
- Voltaje de salida malla 3 (Vrc3, variable controlada).

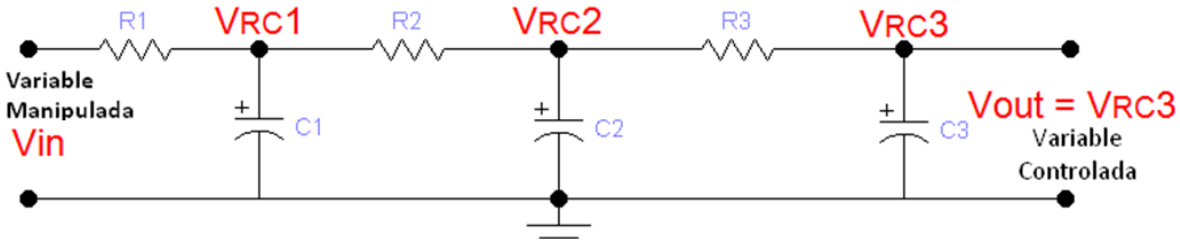


Figura 2.10. Puntos medición variables de la planta de circuitos RC.

Para la lectura de cada variable se asigna una de entrada analógica específica así:

Tabla 2.6. Variable de la planta de circuitos RC y su respectivo canal de medición en el conversor AD del microcontrolador.

Variable a medir	Entrada analógica PIC	Canal
Voltaje de salida(variable manipulada)	ADC_CH0 (RA0/AN0)	0
Voltaje malla 1	ADC_CH2 (RA2/AN2)	2
Voltaje malla 2	ADC_CH3 (RA3/AN3)	3
Voltaje malla 3 (variable controlada)	ADC_CH1 (RA0/AN1)	1

Cabe recordar que los voltajes de la planta pueden tomar valores en un rango de 0 a 10VDC, y la entrada normal del conversor AD del microcontrolador por si sola soporta máximo 5VDC, por lo que en el sistema sensor transmisor se implementó un divisor de tensión (ver capítulo 2 sección 2.3.4. sensor transmisor de voltaje), que cambia el rango del voltaje medible de 0 a 10VDC a un voltaje de 0 a 5VDC,

entonces para medir cualquiera de los voltajes de la planta RC se debe realizar una multiplicación por dos mediante software compensando así la salida del divisor de tensión. Para medir un voltaje se utiliza el siguiente código:

```
set_adc_channel (canal);
variable = 0.009764* read_adc();
```

En la primera línea se selecciona el canal de entrada al conversor, en la segunda se efectúa la medición y el escalamiento de la variable (convertir valor de registro digital de 10 bits a voltaje). El escalamiento consiste en multiplicar el valor del registro digital del conversor por la constante 0.009764, el valor de esta constante se obtiene del siguiente cálculo:

Dado que el conversor es de 10bits, normalmente para medir un voltaje máximo de 5VDC la resolución del conversor está dada por:

$$\text{Resolución} = \frac{\text{Vmax}}{2^n - 1} \tag{2.1}$$

Dónde:
Vmax: Voltaje máximo que soporta la entrada del conversor por hardware, esta es de 5VDC.

n: Tamaño del registro en bits del conversor (10 bits para el conversor del PIC18F4550)

$$\text{Resolución} = \frac{5\text{VDC}}{2^{10}} = \frac{5\text{VDC}}{1023} = 0.00488 = 4.88\text{mV / bit} \tag{2.2}$$

Para medir un voltaje se tiene que:

$$\text{Variable} = \text{Resolución} \times \text{Valor Registro AD} \tag{2.3}$$

Luego para compensar la salida del divisor de tensión el voltaje real debe ser el doble del medido por lo tanto:

$$Variable = (Resolución \times Valor\ Registro\ AD) \times 2 \tag{2.4}$$

Entonces:

$$Variable = 0.09764 \times Valor\ Registro\ AD \tag{2.5}$$

En el proyecto con el fin de optimizar recursos en la programación y reducir código se creó el archivo *leer_voltajes.c*, en este archivo se implementan cuatro (4) funciones, el usuario solo debe llamar la que requiera para obtener la medición en voltios de la variable deseada.

Tabla 2.7. Funciones para lectura de variables de planta de circuitos RC.

Función y uso	Acción que realiza	Declaraciones requeridas
var1 = leer_vout ();	Mide voltaje salida (VC)	Declarar la variable de retorno varX(X =1, 2, 3, 4) como flotante.
var2 = leer_vin();	Mide voltaje entrada (VM)	
var3 = leer_vrc1();	Mide voltaje salida Malla 1	
var4 = leer_vrc2();	Mide voltaje salida Malla 2	

2.12 Imprimir mensajes en pantalla LCD

La pantalla LCD es un periférico auxiliar importante en el desarrollo de prácticas, las ventajas más relevantes se mencionaron en el capítulo 1 sección 1.1.2 Visualización local de variables mediante pantalla LCD 2x16 funcionalidades de la pantalla LCD son:

- Permite imprimir mensajes de texto.
- Permite imprimir tipos de variables como enteros (con y sin signo) y flotantes.

Para el manejo de la pantalla LCD se tiene el archivo *driver_lcd.c*, esta librería internamente permite configurar los pines del microcontrolador destinados al manejo de la pantalla, estos son cuatro pines de datos y dos de control (ver Tabla 1.3. Descripción señales conector-tarjeta principal), también en este archivo se encuentra la función de inicialización y las funciones para el manejo de la pantalla LCD.
El método de escritura en la pantalla varía según el tipo de información que el usuario quiera visualizar, en la siguiente tabla se muestra en detalle las funciones que contiene el archivo *driver_lcd.c*.

Tabla 2.8. Funciones para manejo de pantalla LCD 2x16.

Prototipo función	Declaración requerida	Ejemplo de uso	Descripción
lcd_init();	Ninguna	lcd_init();	Función que inicializa la pantalla LCD, se llama al inicio en el archivo *inicializar_sistema.c*
lcd_gotoxy (int8 x, int8 y);	Opcional: declarar las variables enteras de 8 bits *x* y *y*, de lo contrario colocar directamente los valores	lcd_gotoxy (1,2);	Ubica el cursor de la pantalla en las coordenadas x, y, en el ejemplo se ubica en la columna 1, fila 2.
lcd_putc (char c) ;	Opcional, declarar la variable C de tipo Char, de lo contrario	Lcd_putc ("conexio_ ok");	Permite escribir el mensaje en la pantalla, suele ser usada en conjunto con la

	se puede escribir directamente para mensajes fijos.		función estándar de C *printf*.
lcd_getc(int8 x, int8 y);	Declarar la variable de retorno (ver ejemplo). Opcional: declarar las variables enteras de 8 bits *x* y *y*, de lo contrario colocar directamente los valores	char var; var = lcd_getc(1, 2);	Es una función que sirve para leer información que contenga la pantalla de escrituras anteriores, retorna la información de un carácter en la pantalla.

Nota: Las filas y las columnas se enumeran a partir de 1.

Para imprimir variables de tipo numérico en la pantalla LCD se hace uso de la función *printf* combinada con la función *lcd_putc*, de este modo se puede imprimir distintos tipos de variables, un ejemplo de cómo imprimir una variable tipo flotante se muestra a continuación:

```
lcd_gotoxy(1,1);
printf(lcd_putc, "Voltaje1:%2.3f", V1);
```

En las líneas del ejemplo anterior primero se ubica el cursor de la pantalla LCD en la fila1 columna 1, con la sentencia *lcg_gotoxy*, luego a partir de esa posición se escribe el valor de la variable flotante *V1*, antecedida del mensaje *Voltaje1* con la segunda sentencia *printf(lcd_putc, "Voltaje1:%2.3f", V1);*

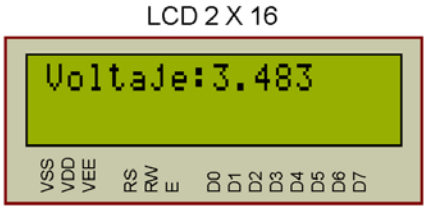


Figura 2.11. Ejemplo despliegue de información en pantalla LCD 2x16.

En la segunda línea de código los caracteres *%2.3f* indican el tipo de variable y el formato para presentarla, en este caso con *f* se indica que la variable es de tipo flotante y con 2.3 que se mostrara 2 dígitos enteros y 3 decimales, se pueden imprimir varios tipos de variables, en la siguiente tabla se presentan los indicadores y una descripción según el tipo de variable.

Tabla 2.9. Indicadores según tipos de variables que se pueden imprimir en la pantalla LCD.

Indicador	Tipo Variable	Descripción
c	char	Carácter
s	string, char	Cadena o carácter

u	unsigned int	Entero sin singo
d	signed int	Entero con signo
Lu	long unsigned int	Entero largo sin signo
Ld	long signed int	Entero largo con signo
f	float	Flotante con decimal
g	float	Flotante con decimal redondeado

2.13 Integración de Bootloader USB para programación de la tarjeta principal

Como se menciona en el capítulo 1 un bootloader es un sistema software que permite programar un microcontrolador sin necesidad de utilizar un programador externo.
Un bootloader o también llamado cargador de arranque es un código que se graba en la memoria de programa del microcontrolador, este presenta la funcionalidad de poder establecer comunicación con el PC y realizar una transferencia directa del firmware desde el PC y grabarlo en la memoria de programa del microcontrolador, los tipos más comunes de dicha comunicación son: comunicación serial RS-232 o por USB, esto depende del tipo de bootloader. La siguiente figura muestra el proceso de programación de un microcontrolador utilizando el método por bootloader en contraste con el método tradicional con programador externo.

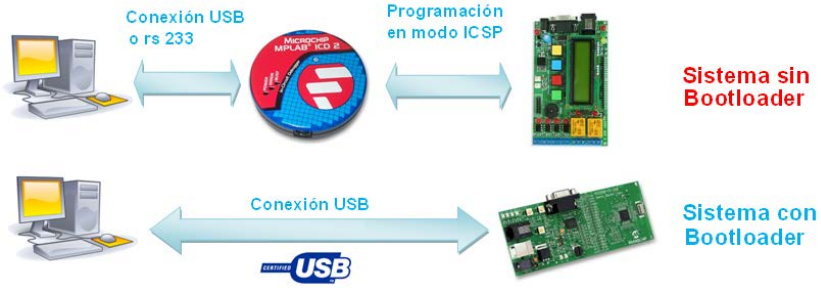


Figura 2.12. Programación con programador hardware externo vs. Programación con bootloader integrado.

Existen varias ventajas por las cuales se decidió implementar un bootloader en la tarjeta principal del proyecto, a continuación se mencionan las más relevantes:

- El método de programación por bootloader permite una rápida programación de la tarjeta principal.
- El uso de un bootloader no requiere del uso constante de un programador hardware externo para grabar el firmware en el PIC (solo se requiere una vez para grabar el código del bootloader).
- La programación del microcontrolador se puede hacer directamente en la tarjeta principal sin necesidad de extraer el microcontrolador.
- El bootloader solo utiliza una conexión USB física (cable USB y conector USB).

- Presenta una ventaja económica ya que se evita el uso constante de un programador externo.

Existen varios tipos bootloader para los microcontroladores PIC de Microchip, muchos de ellos de uso libre, estos están diseñados para operar con comunicación USB o serial, debido a que la tarjeta principal integra una conexión por USB, se opta por utilizar un bootloader USB, investigando sobre *bootloaders* USB se encontró que Microchip ofrece uno especialmente diseñado para los microcontroladores PIC de la familia 18Fx5xx con módulo USB dentro de la cual se encuentra el microcontrolador PIC 18F4550, el código del bootloader mencionado es un código implementado en C con el compilador MC18 de Microchip, el código es de libre uso y el usuario puede modificarlo según sus necesidades, este se encuentra en el paquete *Microchip Application Libraries v2010-08-04 Installer.exe* y puede ser descargado de: *www.Microchip.com/usb*, al instalar el paquete se generan varias carpetas y archivos, el siguiente es un ejemplo de la ubicación del proyecto correspondiente al código del bootloader utilizado en el proyecto:

C:\Microchip Solutions v2010-08-04\USB Device - Bootloaders\Vendor Class - MCHPUSB Bootloader\Bootloader - Firmware for PIC18F4550 Family Devices

El código del bootloader se encuentra en la carpeta *Bootloader - Firmware for PIC18F4550 Family Devices*, originalmente el bootloader está diseñado para operar con la tarjeta PICDEM FS USB que comercializa Microchip, dado que algunas de las funcionalidades no son utilizadas por la tarjeta principal del proyecto, se realizó una revisión del código y se modificó para adecuarlo mejor según las características del proyecto, el código que resulto de dicha revisión se entrega como anexo digital y lo puede encontrar en el CD de usuario del sistema DCB01.

Por otra parte hay que recordar que un elemento importante es la interfaz de usuario que permite operar con el bootloader desde el PC, para ello Microchip igualmente ofrece una interfaz de usuario, esta también viene en el paquete *Microchip Application Libraries v2010-08-04 Installer.exe* y se puede acceder a ella luego de la instalación, el siguiente es un ejemplo de la dirección dentro de los directorios de instalación:

C:\Microchip Solutions v2010-08-04\USB Tools\Pdfsusb

Dentro de la carpeta *Pdfsusb*, se encuentra la aplicación *PDFSUSB.exe*, la siguiente figura muestra la interfaz de usuario para programar la tarjeta principal mediante el *bootloader* USB de Microchip y una descripción de sus partes.

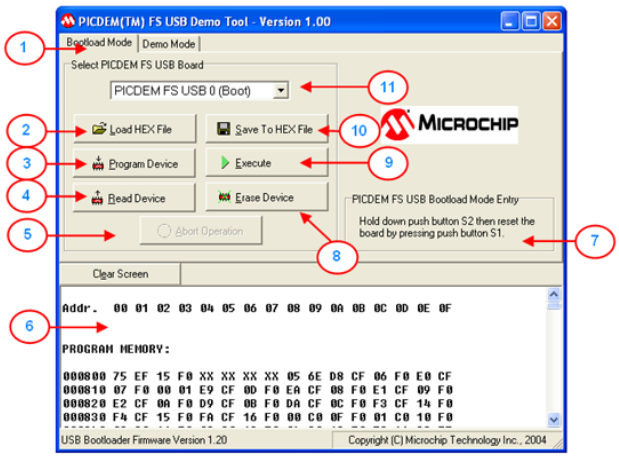


Figura 2.13. Interfaz PICDEM FS USB Tool para programación con bootloader.

1. Pestaña selección modo
2. Cargar firmware a grabar en el PIC
3. Opcion para iniciar la programacion del PIC
4. Opcion para leer memoria flash del PIC
5. Opcion cancelar proceso (grabación, lectura o borrado) una vez iniciado.
6. Sección desplegar mensajes y firmware (codigo .hex) del PIC
7. Mensaje con instrucciones para activar el modo bootloader en la tarjeta (S1=botón Reset y S2= botón Boot)
8. Opción de borrar memoria de programa del PIC
9. Opción para ejecutar el firmware una vez grabado o salir del modo bootloader
10. Opción para grabar el código leído de la memoria flash del PIC
11. Pestaña para seleccionar la tajeta una vez ha sido detectada en modo bootloader

La implementación de un bootloader presenta dos desventajas, la primera está dada por la naturaleza del mismo, dado que el bootloader es un programa que reside en la memoria de programa del microcontrolador, el cual ocupa una parte de la memoria de programa que el usuario podría llegar a necesitar en caso de tener códigos extensos. En el caso del presente proyecto el microcontrolador PIC 18F4550 tiene 32KB de memoria de programa y el bootloader seleccionado ocupa 2KB, por lo que quedan 30KB (15Kpalabras, una palabra o código de instrucción es conformada por 2 Bytes) lo cual es suficiente para las aplicaciones de adquisición y control a las que está destinada la tarjeta principal. La segunda desventaja depende de la ubicación del bootloader en la memoria de programa, este puede ser ubicado al inicio o al final de la memoria de programa, si se ubica al inicio el problema está en que se debe re-mapear vectores preestablecidos como son las direcciones de reset y de atención a interrupciones, por el contrario si el bootloader se ubica al final no se debe re-mapear, pero se corre el riesgo de dañar el código del bootloader, por lo que se opta por colocar el bootloader al inicio y re-mapear las direcciones, la siguiente figura muestra la organización normal de la memoria de programa del microcontrolador y como queda luego de la instalación del bootloader.

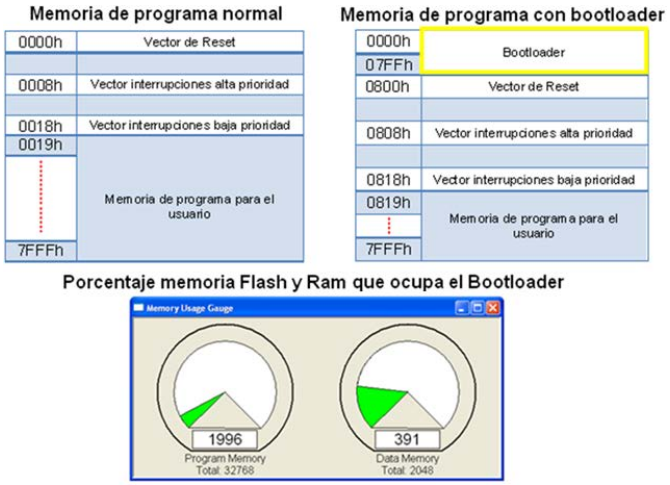


Figura 2.14. Distribución memoria de programa microcontrolador normal y con bootloader

En la plantilla en el archivo main.h se re-mapea así:

Archivo main.h

#BUILD (reset=0x800, interrupt=0x808)
#ORG 0x0000, 0x07ff {}

Las dos sentencias anteriores reubican los vectores de reset y atención a interrupciones, se debe tener especial cuidado de no borrar o modificar estas líneas de código en el código plantilla, ya que si se hace se puede dañar el código del bootloader, por ejemplo si estas se borran el bootloader solo funcionara una vez ya que el código que se grabe sobrescribirá el del bootloader.

La instalación y modo práctico de manejo del bootloader en la programación de la tarjeta principal se explica en de talle en el anexo *manual de usuario sección programación de la tarjeta principal mediante bootloader USB*.

Equation Section 3

3 DISEÑO DE UNA INTERFAZ DE USUARIO BÁSICA PARA INTERACCIÓN CON EL SISTEMA DESDE EL PC

Después de haber establecido un método de comunicación entre la tarjeta principal, las tarjetas a controlar y Matlab sobre el sistema operativo Windows XP, como se describió en el capítulo 3 *Intercomunicación a nivel software del sistema*, se procede con el diseño de una aplicación que sirva como herramienta básica para el monitoreo de las variables de proceso para el control de las plantas del sistema.

3.1 Interfaz de usuario en el PC

En el diseño de la interfaz de usuario se deben tener en cuenta criterios que permitan al usuario una fácil interacción con el sistema, y a la vez debe satisfacer la necesidad de conocer el comportamiento de las variables de la planta (variable manipulada, variable controlada), así como las variables derivadas del control como el error, el esfuerzo de control y el set point, de modo que el usuario pueda analizar rápidamente el comportamiento de una planta al implementar un controlador.

Para lograr que la interfaz facilite la interacción y análisis del comportamiento de las plantas ante los controladores implementados por el usuario, se plantea implementar los siguientes requerimientos:

- Ingresar y enviar información desde el PC a la tarjeta principal: tal como el set point o el esfuerzo de control según el tipo de planta conectada.
- Capacidad de Monitorear variables: visualizar de forma numérica las variables que intervienen en el proceso de modo que se tengan valores precisos.
- Graficar variables: graficar las variables de proceso de modo que se pueda analizar de forma rápida el comportamiento de las variables del proceso en el tiempo.
- Controlar de forma manual las plantas: permitir que el usuario realice cambios en el esfuerzo de control, de modo que obtenga datos que sirvan para la caracterización del sistema.

Teniendo en cuenta que el correcto análisis de un proceso está dado por el conocimiento que se tiene del mismo, el cual se adquiere a través de las variables que describen el comportamiento del sistema, entonces se pone a disposición del usuario las variables necesarias para analizar el comportamiento de las plantas al implementar un controlador, como son: el set point, la variable controlada, el esfuerzo de control, la variable manipulada y el error de estado estacionario.

La implementación de la interfaz se desarrolló en GUIDE, entorno de programación visual disponible en Matlab, el cual proporciona un conjunto de herramientas para crear interfaces gráficas de usuario (GUI). GUIDE tiene las características básicas de los programas visuales tales como Visual Basic o Visual C++ (Help gui Matlab, 2011).

3.2 Diseño de la interfaz gráfica

El diseño de la interfaz de usuario inicio con la búsqueda de un método para graficar las variables del proceso, lo cual en principio se realizó a través de archivos *.m, graficando en un bucle secuencial, es decir se leían datos y se graficaba inmediatamente al recibirlos, lo que causaba mucha congestión en las operaciones realizadas por Matlab, además el tiempo estaba muy desfasado del comportamiento real de la planta.

Por todo lo anterior se buscó un método que permitiera mejorar el proceso de graficar las variables, luego de estudiar varias alternativas se optó por el uso de timers. A continuación se da una breve explicación sobre estos objetos de Matlab.

Un timer es un objeto que se puede utilizar para programar la ejecución de funciones en Matlab. Se dice que esta encendido cuando el tiempo señalado en el timer ha transcurrido y ejecuta las funciones que se le han especificado.

Para usar un timer en Matlab se debe crear un objeto *timer*, el cual soporta distintas propiedades y funciones que controlan su comportamiento. Para crear un timer se debe usar la función ‘‘*timer’’* la cual crea un objeto válido con los valores por defecto para la mayoría de las propiedades del mismo (Mathworks, 2011).

En el diseño de la interfaz se crearon dos timers como se muestra a continuación:

```
t1 = timer('TimerFcn',@tiempo, 'ExecutionMode', 'fixedSpacing' ,'BusyMode', 'queue' ,'Period', tm);
t2 = timer('TimerFcn',@graficar, 'ExecutionMode', 'fixedSpacing' ,'BusyMode', 'queue' ,'Period', tg);
start (t1);
start (t2);
```

Donde t1 y t2 son objetos de tipo ‘‘*timer*’’ a los cuales se les definió la función que deben ejecutar al ser activados, el modo de ejecución y el tiempo de ejecución, para posteriormente iniciarlos con la función *start;* una vez inician se ejecutan en segundo plano y el único control que se puede hacer sobre ellos es detenerlos y borrarlos.

Los timer creados se ejecutan en tiempos diferentes de modo que el t1 ejecuta la función *tiempo*, la cual se encarga de realizar la adquisición y almacenamiento de datos de las variables; Su ejecución es más rápida que la del t2 que es el encargado de ejecutar la función *graficar* la cual se encarga de actualizar las gráficas con los vectores donde se han almacenado los nuevos datos de las diferentes variables del proceso. Esto último es una gran ventaja ya que realiza una actualización cada cierto número de muestras, y no de forma constante como se hacía en el método inicial que sobrecargaba el sistema.

Luego de tener el método para graficar establecido, se decide conservar la gráfica en el *axes* que proporciona Matlab, en vez de incorporarlo en la interfaz de usuario diseñada, esto con el fin de conservar las herramientas de edición y visualización, que facilitan el análisis de la respuesta de cada planta mostrada de forma gráfica en el *axes,* así que la gráfica se despliega en una ventana independiente a la aplicación. De modo que el usuario tiene a su disposición una interfaz dividida en dos partes, una para realizar análisis mediante la gráfica mostrada en el *axes* y otra para escoger el tipo de planta a trabajar, ingresar el set point, monitorear los valores numéricos de las variables del proceso y guardar los datos obtenidos de la práctica realizada, todo lo anterior a través de la GUI desarrollada con el GUIDE de Matlab. El diseño realizado es el de una interfaz básica de tamaño reducido para que el usuario pueda ponerla sobre el *axes* sin que interfiera en la visualización del comportamiento de las variables graficadas.

Una aplicación desarrollada con GUIDE consta de dos archivos, un archivo .*m (ejecutable) que controla el funcionamiento de la interfaz gráfica de usuario; y otro archivo *.fig que corresponde a la parte gráfica. Las dos partes están unidas a través de subrutinas llamadas callback. Usando el editor del archivo *.m, el diseñador de la interfaz puede agregar código en cada callback, para lograr el funcionamiento deseado de los componentes utilizados.

3.3 Atributos de elementos en una GUI de Matlab.

Antes de iniciar con la descripción del diseño, es necesario tener claros algunos conceptos esenciales en el desarrollo de una interfaz de usuario con el entorno GUIDE de Matlab, a continuación se explican algunos de los más usados:

Callback o función de llamada: es una función que se crea al colocar un objeto con el GUIDE, el código que se ponga en el callback especifico se ejecuta cada vez que el sistema o el usuario interactúa con el objeto. El código en estas funciones es definido por el diseñador con el fin de obtener la respuesta deseada en cada componente.

En la siguiente imagen se tiene un ejemplo del código generado cuando se crea un *push button* con el GUIDE de Matlab, la última función que aparece en el código corresponde a la función *callback*.

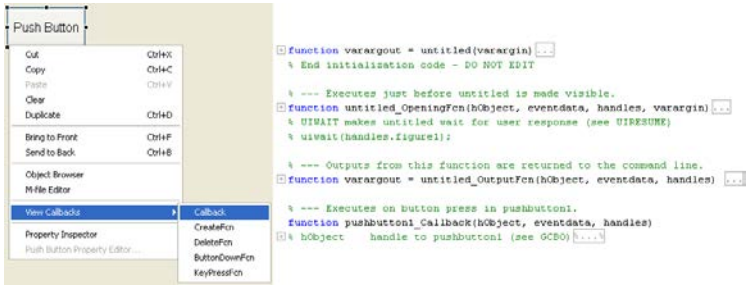


Figura 3.1. Ejemplo objeto GUIDE y código generado en Matlab

En la Figura 3.1, al lado izquierdo se encuentra una imagen de las opciones que se despliegan al hacer clic derecho sobre un componente en el GUIDE de Matlab, al lado derecho se encuentra el código generado automáticamente por el GUIDE.

Handles: Todos los valores de las propiedades de los componentes (color, valor, posición, string, entre otros) y los valores de las variables transitorias del programa se almacenan en una estructura, los cuales son accedidos mediante un único identificador para cualquier componente. Este identificador se llama *handles*.

Get: Accede al valor del componente a través del identificador *handles*, en el proyecto se usa para obtener el valor numérico del set point ingresado a través de un componente de tipo *Edit Text* o el estado de un botón como es el caso del botón de inicio o los *Radio Button* empleados en la selección del tipo de planta a trabajar, entre otros*.*

Set: Se encarga de asignar valores ya sean caracteres o numéricos, cuando las propiedades del componente de destino lo permite, esto se hace a través del identificador *handles.* En el proyecto se usa para actualizar el valor de las variables monitoreadas al igual que sus unidades, entre otras cosas.

3.4 Componentes de GUIDE empleados en el diseño de la interfaz de usurario.

A continuación se encuentran los componentes (objetos de GUIDE), empleados en el diseño de la interfaz de usurario, se describe tanto su uso, como el código que ejecuta internamente al activarlo, todo esto acorde con los requerimientos planteados anteriormente para el diseño de la interfaz de usuario.

El mecanismo para escoger el tipo de planta a trabajar, está conformado por dos tipos de componentes, los *radio button* y el *button group.* Un *radio button* funciona como una casilla de verificación, al tener varios de estos componentes se desea que su comportamiento sea mutuamente exclusivo, es decir si un botón esta encendido, todos los demás botones del mismo tipo se apagan, para lograr esto se emplea el componente *button group,* el cual permite exclusividad de selección entre los *radio button*. En el archivo *.m se modifican las funciones *callback* de

cada *radio button,* en ella se declaran variables globales asociadas al estado de cada componente, con el fin que esta información se pueda compartir en las demás funciones que hacen parte del archivo *.m de la GUI.

En la verificación del funcionamiento se encontró el inconveniente que a pesar de que los *radio button* son mutuamente exclusivos al estar agrupados en el componente *button group,* el usuario podía deshabilitar las dos opciones, tanto la planta de circuitos RC como la de temperatura, y aun así introducir un SP, el inconveniente radica en que el rango de selección del SP está dado según el tipo de planta que se desee trabajar, de modo que en la función callback, de cada tipo de planta, se implementó un código que deshabilita la casilla de ingreso del SP si el usuario no ha escogido una de las dos plantas, del mismo modo se deshabilitan los botones de *Iniciar* y *Guardar* y mediante un mensaje de alerta se le pide al usuario que escoja una de las plantas disponibles. Por otro lado, dado que en la planta de temperatura no se mide la variable manipulada, puesto que no hay implementado un sistema hardware que permita la adquisición de dicha variable en esta planta, entonces la casilla de la variable manipulada, en la sección de variables de proceso, también aparece deshabilitada al escoger la planta de temperatura.

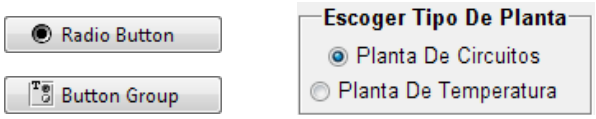


Figura 3.2. Radio button y button group de GUIDE e implementación.

En la Figura 3.2, al lado izquierdo se encuentran los componentes empleados como aparecen en la paleta de componentes del GUIDE de Matlab, al lado derecho se tiene una imagen de la sección escoger tipo de planta como se encuentra en la GUI diseñada,

El ingreso del SP al sistema se hace a través de un componente de tipo *edit text,* que permite al usuario ingresar datos en forma de cadena de texto, la cual se convierte a valores numéricos con el comando *str2num* de Matlab. En el callback asociado a este componente se implementa un código que convierte la cadena de entrada a valores numéricos. Si el usuario ha escogido una planta y el set point introducido está dentro del rango establecido para la planta seleccionada, se habilitan los botones *Iniciar* y *Guardar.* Si el valor numérico esta fuera del rango, se despliega un mensaje para recordarle al usuario cual es el rango de operación de cada planta, por otro lado si el valor introducido es un carácter o no se introduce dato alguno, se abre un mensaje de error, pidiéndole al usuario que ingrese un valor numérico.

Cuando el usuario escoge trabajar con la planta de circuitos RC, el rango de trabajo está entre 0 y 10 voltios por otro lado el rango de trabajo de la planta de temperatura está entre 25 y 85 °C.

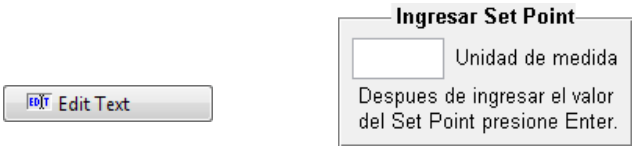


Figura 3.3. Edit text de GUIDE e implementación.

En la Figura 3.3, al lado izquierdo se encuentra el componente empleado como aparece en la paleta de componentes del GUIDE de Matlab, al lado derecho se tiene una imagen de la sección Ingresar Set Point como se encuentra en la GUI diseñada.

En el archivo *.m asociado a la interfaz gráfica de usuario se encuentra una función llamada *graficar* en la cual se actualizan los valores de las variables graficadas en el *axis,* y el valor numérico desplegado en la interfaz de monitoreo. *graficar* es la función asociada con el *timer t2*, mencionado al inicio de la sección Diseño de la interfaz gráfica de usurario, la implementación se lleva a cabo en esta función dado que el tiempo que tarda en desbordarse este *timer* permite visualizar el cambio en el valor de las variables monitoreadas. A continuación se describe el funcionamiento de la sección Variables Del Proceso.

En la GUI diseñada se deben desplegar mensajes estáticos, como en el caso del mensaje que aparece en la sección Ingresar Set Point, que le recuerda al usuario que debe presionar *enter* después de ingresar el valor de set point deseado. También se despliegan valores que se deben actualizar como en el caso de las variables de proceso o sus unidades que se modifican dependiendo de la planta que se escoja.

El componente del GUIDE utilizado para implementar el despliegue de los valores mencionados anteriormente, es el *static text,* este componente tiene la capacidad de desplegar símbolos, mensajes o valores numéricos en una GUI. Este tipo de componente no tiene un *callback* asociado. Cuando se emplea este componente y se desea actualizar su valor, se accede a él haciendo uso de las funciones *handles y set*. En el archivo *.m, el valor de las variables de proceso se asocia a cada *static text* en la función *graficar* de uno de los timer implementados, dado que el tiempo que este tarda en desbordarse es suficiente para permitirle al usuario una correcta visualización del valor de cada variable.

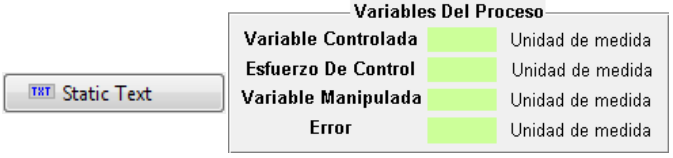


Figura 3.4. Static text de GUIDE e implementación.

En la Figura 3.4, al lado izquierdo se encuentra el componente empleado como aparece en la paleta de componentes del GUIDE de Matlab, al lado derecho se tiene una imagen de la sección Variables Del Proceso como se encuentra en la GUI diseñada.

Cuando el usuario escoge trabajar con la planta de temperatura, la casilla asociada al valor de la variable manipulada cambia a color gris dando a entender que esta se encuentra inactiva, esto se debe a que no hay un sistema hardware que permita la adquisición de dicha variable.

El botón de Inicio/Detener esta implementado a través de un componente llamado *toggle button*, el cual genera una acción que indica un estado binario, on/off.

Cuando el usuario activa el botón *Iniciar* en el archivo *.m se ejecutan las siguientes acciones:

- El texto mostrado en el botón, pasa de *Iniciar* a *Detener.*
- Se obtiene el valor del estado del botón.
- Deshabilita las opciones de escoger planta.
- Se configura y carga la librería para iniciar la comunicación entre la tarjeta principal y el PC.
- En caso de no detectar la tarjeta principal, se despliega un mensaje que permite al usuario buscarla de nuevo o cancelar la búsqueda.
- Prepara el *axis,* dado que el valor y nombre de los ejes, así como el titulo varía según el tipo de planta escogida, también se definen las características de color, ancho y tipo de línea que representa las variables de proceso graficadas.
- Si la tarjeta principal ha sido detectada entonces se le envía el set point.

Por otro lado cuando el usuario pulsa el botón *Detener,* en el archivo *.m se ejecutan las siguientes acciones:

- Se despliega un mensaje para preguntarle al usuario si realmente desea detener el proceso, en caso afirmativo se consulta si desea guardar las variables de proceso, de ser así, se guardan de lo contrario se borran los datos.
- Se cambia el nombre mostrado en el botón, pasa de *Detener* a *Iniciar*
- Deshabilita los botones de *Iniciar* y *Guardar.*
- Se descarga la librería de comunicación.
- Se elimina lo que hay hasta ese momento en el Workspace de Matlab.
- Se despliega un mensaje pidiéndole al usuario que reinicie la tarjeta principal de modo que el sistema queda listo para comenzar nuevamente.
- En caso que el usuario oprima por error el botón detener, puede continuar el proceso normalmente si al preguntarle si desea detener el proceso oprime no.

Figura 3.5. Toggle button de GUIDE e implementación.

En la Figura 3.5, al lado izquierdo se encuentra el componente empleado como aparece en la paleta de componentes del GUIDE de Matlab, al lado derecho se tiene una imagen del botón *Iniciar* como se encuentra en la GUI diseñada.

En el proceso realizado es muy importante que el usuario pueda conservar los datos obtenidos en la práctica, de modo que se implementó un botón que le permite guardar todas las variables del proceso en cualquier momento. El componente del GUIDE utilizado para cumplir con esta función es el *push button*, el cual ejecuta la acción que se encuentra en su callback. El código del archivo *.m asociado a este botón se encarga de guardar todas las variables del proceso en un archivo *.mat, del cual el usuario puede escoger su localización.

Figura 3.6. Push button de GUIDE e implementación.

En la Figura 3.6, al lado izquierdo se encuentra el componente empleado como aparece en la paleta de componentes del GUIDE de Matlab, al lado derecho se tiene una imagen del botón *Guardar Datos* como se encuentra en la GUI diseñada.

En la Figura 3.7, se muestran dos imágenes de la GUI diseñada para el monitoreo de las variables, al lado izquierdo se tiene una muestra de la interfaz cuando se trabaja con la planta de circuitos RC y al lado derecho esta la muestra cuando se trabaja con la planta de temperatura.

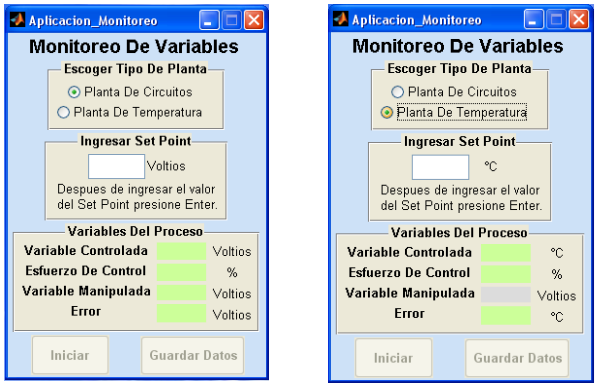


Figura 3.7. Interfaz de usuario para el monitoreo de variables del sistema

Cabe resaltar que teniendo la comunicación establecida se pueden crear muchas más aplicaciones según las necesidades del usuario, por ejemplo para realizar control manual se diseñó otra interfaz similar a la descrita anteriormente, la cual permite al usuario modificar el ciclo útil de la señal PWM que genera el esfuerzo de control. Esta función no se integró en la GUI de monitoreo de variables dado que se considera que es de uso ocasional. En el desarrollo del sistema se emplea esta interfaz para realizar la caracterización del actuador en la planta de circuitos RC.

Los componentes empleados en desarrollo de la GUI control manual son los mismos usados en la interfaz monitoreo de variables, por esta razón no se hace una descripción detallada del diseño. En la siguiente figura se muestra una imagen correspondiente a la interfaz diseñada.

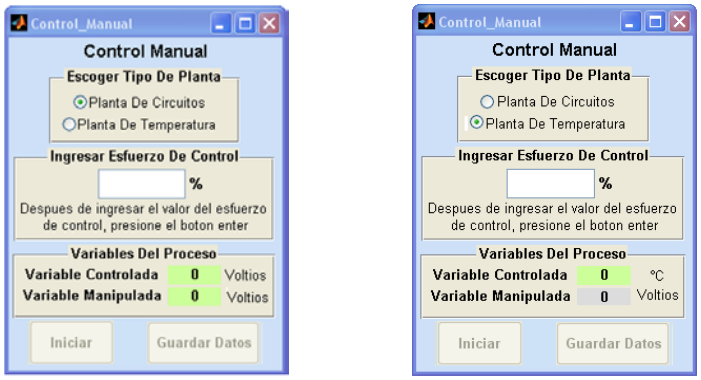


Figura 3.8. Interfaz de usuario para el control manual.

En la Figura 3.8 a la izquierda se encuentra la apariencia de la interfaz al seleccionar la planta de circuitos. Al lado derecho se tiene la apariencia de la interfaz al seleccionar la planta de temperatura.

En el manual de usuario (ver anexo 1), se detallan los procedimientos para el uso del sistema.

Equation Section 4

4 MODELADO DE PLANTAS EXPERIMENTALES Y DISEÑO DE CONTROLADORES

En este capítulo se obtienen los modelos de las plantas para proceder con la implementación de los controladores que permitirán realizar la validación del sistema.

4.1 Modelado de plantas experimentales

En esta sección se mostraran los procedimientos realizados para obtener los modelos tanto de la planta de temperatura como de la planta de circuitos RC.

Para realizar el control de las plantas diseñadas se debe conocer la naturaleza del proceso, ya que son sus características las que definen los parámetros adecuados para su control.

La comprensión del proceso se puede obtener mediante el desarrollo de un modelo teórico utilizando los balances de energía, balances de masa o leyes químicas y físicas, sin embargo, frecuentemente esto conlleva esfuerzos y consumos de tiempo que ocasionalmente se pueden evitar. Una buena aproximación de la dinámica del proceso se puede obtener mediante el uso de métodos simplificados de cálculo, como los modelos de primer y segundo orden más tiempo muerto, entre otros (LIPTÁK, 2006).

4.1.1 Modelado planta de temperatura

Para realizar el modelado de la planta de temperatura, se realizó un experimento práctico con el fin de obtener datos reales que describan su comportamiento, estos datos están alrededor de un punto de operación, que se encuentra entre 50 y 60°C, con los cuales se desea obtener un modelo de primer orden más tiempo muerto.

En la siguiente figura se muestra una imagen de los datos obtenidos durante 3500 segundos.

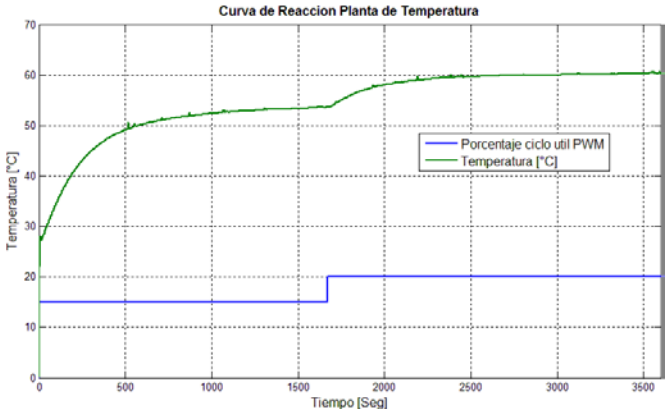


Figura 4.1. Curva de reacción planta de temperatura

En el desarrollo de este experimento se utilizó la tarjeta principal, la planta de temperatura y la interfaz diseñada para controlar manualmente las plantas. En resumen el procedimiento que se desarrolla en la práctica es el siguiente:

- Manualmente introducir un cambio en el valor de la señal PWM.
- Esperar a que la variable controlada llegue a un valor de estado estable.
- Manualmente introducir un cambio de entre 5% y 10% del valor de la señal PWM.
- Esperar a que la variable controlada llegue a un valor de estado estable.
- Guardar los datos obtenidos del procedimiento anterior.
- Después de aplicar el procedimiento descrito anteriormente se obtuvieron los datos que describe la anterior figura, en donde se puede apreciar el cambio en la variable controlada (temperatura), al variar el esfuerzo de control en un 5%.

Con los datos obtenidos en el procedimiento descrito anteriormente, se busca obtener un modelo de primer orden más tiempo muerto, el cual tiene la siguiente forma y sus parámetros se describen a continuación:

$$G_{(s)} = \frac{K_P}{\tau s + 1} e^{-t_0 s} \qquad (4.1)$$

Dónde:
K_P: Ganancia del proceso
τ: Constante de tiempo del proceso
t_0: Tiempo muerto

En el proceso del modelado de la planta de temperatura se aplicaron dos métodos de identificación, con ambos se obtuvieron modelos de primer orden más tiempo muerto y posteriormente se realizó una comparación entre los modelos, para luego seleccionar uno de ellos destinado al diseño de controladores.

El primer método empleado para hallar los parámetros del modelo de primer orden más tiempo muerto, fue el método de los dos puntos propuesto por Smith, para el cual se requiere la curva de reacción.

Los instantes seleccionados por Smith fueron los tiempos requeridos para que la respuesta alcance el 28.3% ($t_{0.2838}$) y el 63.2% ($t_{0.632}$) de su valor final en estado estable. En la siguiente figura se muestra la manera de obtener los dos puntos.

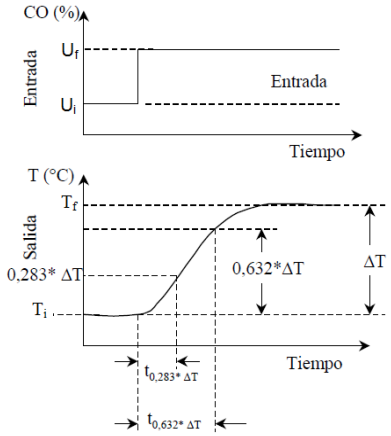


Figura 4.2. Método de los dos puntos para obtener un modelo POMTM

Teniendo $t_{0.283}$ y $t_{0.632}$ como datos, la constante de tiempo (τ) y el tiempo muerto (t_o) son determinados por las siguientes ecuaciones

$$\tau = 1.5(t0.632 - t0.283) \quad (4.2)$$

$$t_0 = t\big|_{0.632\Delta_0} - \tau \quad (4.3)$$

La ganancia K se halla con la ecuación que se muestra a continuación

$$K_P = \frac{T_f - T_i}{U_f - U_i} \quad (4.4)$$

Dónde:

τ: Constante de tiempo del proceso.

$t|_{0.632}$: Tiempo en que la salida alcanza el 63.2% de su valor final en estado estable.

$t|_{0.283}$: Tiempo en que la salida alcanza el 28.3% de su valor final en estado estable.

t_0: Tiempo muerto del proceso.

K_P Ganancia del proceso.

T_f: Valor final de la variable controlada (temperatura).

T_i: Valor inicial de la variable controlada (temperatura).

U_f: Valor final del esfuerzo de control

U_i: Valor inicial del esfuerzo de control

Para aplicar el método descrito anteriormente, se toman los datos a partir del momento en el que se introduce el segundo cambio manual en el esfuerzo de control, el tiempo en el que se genera el cambio se toma como tiempo inicial (tiempo igual a cero) y se obtienen los datos descritos por la siguiente figura, en ella se aplica el método de los dos puntos.

Teniendo:

$$\Delta T = 6.75$$
$$t|_{0.283} \times \Delta T = 1.91$$
$$t|_{0.632} \times \Delta T = 4.26$$

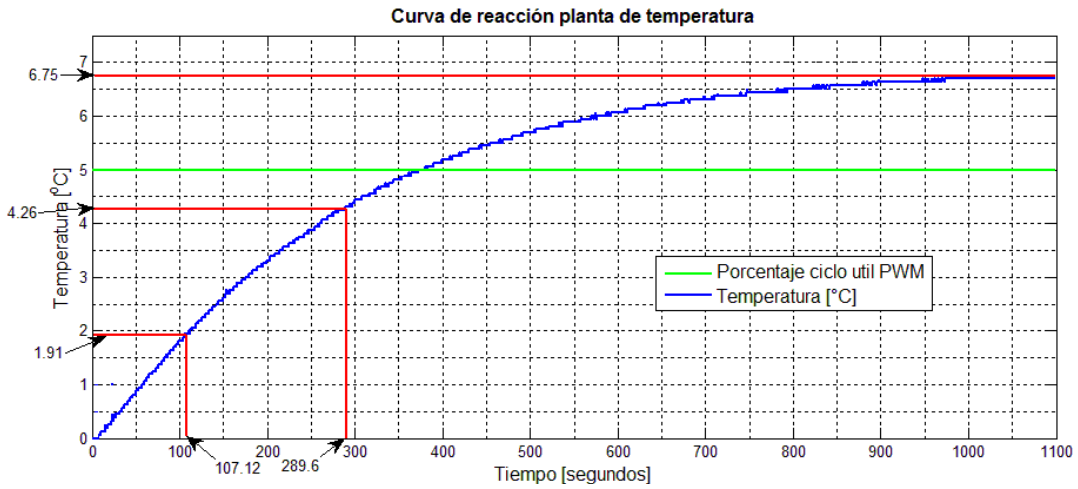


Figura 4.3. Curva de reacción planta de temperatura al aplicar el método de los dos puntos

De la Figura 4.3. Curva de reacción planta de temperatura al aplicar el método de los dos puntos se obtiene:

$$t|_{0.283} = 107.12$$
$$t|_{0.632} = 289.6$$

Aplicando las fórmulas para obtener la constante de tiempo (τ), el tiempo muerto ($\boldsymbol{t_0}$) y la ganancia K, se tiene:

$$\tau = 1.5\left(289.6 - 107.12\right) = 273.72seg \quad (4.5)$$

$$t_0 = 289.6 - 273.72 = 15.88seg \quad (4.6)$$

$$K_P = \frac{6.75 - 0}{5 - 0} = 1.35\left[\frac{℃}{\%}\right] \quad (4.7)$$

Luego con el método de los dos puntos propuesto por Smith, se obtiene el primer modelo POMTM para la planta de temperatura como se muestra en la ecuación 4.8:

$$G_{(s)} = \frac{1.35}{273.72s + 1} e^{-15.88s} \quad (4.8)$$

El segundo método de modelado fue la estimación de parámetros, por medio del criterio de la integral del error cuadrado, a través de archivos *.m de Matlab, para obtener el modelo se adecuaron los datos obtenidos en el experimento descrito anteriormente, de modo que se separan los datos a partir del momento en el que se introduce el segundo cambio manual en el ciclo útil de la señal PWM, el tiempo en el que se genera el cambio se toma como tiempo inicial (tiempo igual a cero), estos datos se pasan al archivo *estimacion_del_modelo.m*, el cual entrega los parámetros del POMTM (RENGIFO, 2011). El resultado de aplicar este método fue el siguiente modelo de POMTM:

$$G_{(s)} = \frac{1.3853}{284.3269s + 1} e^{-13.5189s} \quad (4.9)$$

La Figura 4.4, corresponde a la comparación entre el comportamiento de la planta real ante el cambio en el esfuerzo de control y la simulación del mismo cambio en el modelo obtenido mediante el criterio de la integral del error cuadrado, como se puede apreciar el modelo es bastante aproximado al comportamiento real de la planta de temperatura.

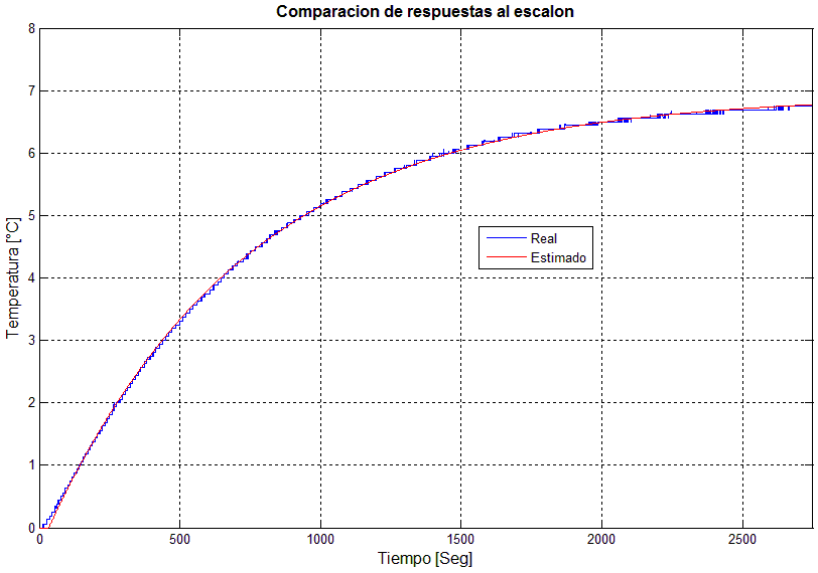


Figura 4.4. Planta real Vs simulación modelo POMTM obtenido por el criterio de la integral del error cuadrado

Comparación de los dos modelos obtenidos según los métodos aplicados

En la siguiente tabla, se muestran los parámetros obtenidos mediante el método de los dos puntos de Smith y el criterio de la integral del error cuadrado, como se puede apreciar los parámetros son muy similares.

Tabla 4.1. Parámetros de modelos POMTM

Parámetro	Método	
	De los dos puntos	Integral del error cuadrado
τ	273.72	284.3269
t_0	15.88	13.5189
K_P	1.35	1.3853

Se pone a prueba los modelos obtenidos mediante la respuesta al escalón en lazo abierto a través de simulación en Simulink, donde se obtienen los resultados mostrados en la Figura 4.5.

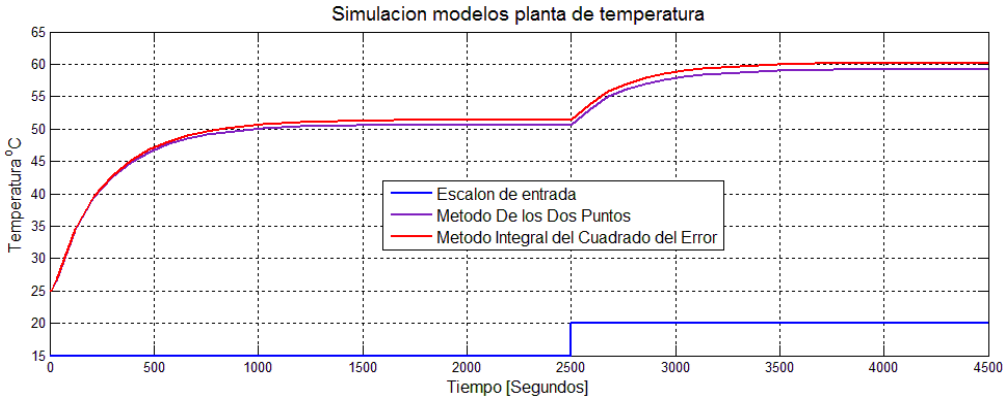


Figura 4.5. Simulación de modelos POMTM

En la Figura 4.5 en color morado se tiene la simulación de la respuesta al escalón del modelo obtenido mediante el método de los dos puntos, en color rojo se tiene la simulación correspondiente al modelo hallado mediante el método de la integral del error cuadrado. Como se puede apreciar en la gráfica, a pesar de las mínimas diferencias en los modelos en cuanto al tao, tiempo muerto y ganancia, las respuestas al escalón son muy similares, sin embargo el modelo obtenido por el método de la integral, se estabilizo alrededor de los 60°C al igual que el sistema real, de modo que se escoge el modelo obtenido por este método para realizar el controlador que validara el sistema con la planta de temperatura

por esta razón se decide trabajar con el modelo obtenido mediante el método de la integral del cuadrado del error.

4.1.2 Modelado planta de circuitos RC

La obtención de un modelo de la planta de circuitos RC se realizó teóricamente a través de modelos matemáticos de los componentes, la aplicación de la ley de voltajes de Kirchhoff y la definición de la reactancia capacitiva en función de la transformada de Laplace. Para recordar estos dos conceptos se puede decir que la ley de voltajes de Kirchhoff establece que: en toda malla la suma de todas las

caídas de tensión es igual a la tensión total suministrada. De forma equivalente, en toda malla la suma algebraica de las diferencias de potencial eléctrico es igual a 0, matemáticamente esto es:

$$\sum_{k=1}^{n} V_k = V_1 + V_2 + V_3 \ldots + V_n = 0 \qquad (4.10)$$

Por otro lado la reactancia capacitiva (Xc) en función de la transformada de Laplace está dada por:

$$Xc_x = \frac{1}{C_x \times s} \qquad (4.11)$$

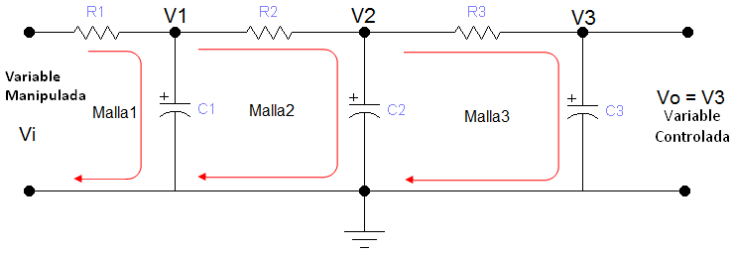


Figura 4. 6. Diseño para la planta de circuitos RC

Recordando que el diseño para la planta de circuitos está representado por la Figura 4. 6, y teniendo en cuenta que el circuito es configurable de modo que se puede obtener sistemas de primer, segundo y tercer orden, entonces se procede a modelar el sistema para cada orden posible en la planta, para ello se aplica la ley de voltajes de Kirchhoff en cada nodo, como se muestra a continuación.

4.1.2.1 Planta de circuitos RC configurada como sistema de primer orden

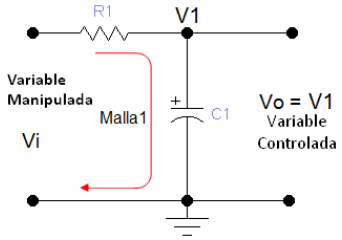


Figura 4.7. Diseño de la planta de circuitos RC configurada como sistema de primer orden

Un sistema de primer orden está representado por el circuito de la figura anterior.

Aplicando la ley de voltajes de Kirchhoff para cada nodo y la definición de la reactancia capacitiva en función de la transformada de Laplace, en la malla1 mostrada en la figura anterior, se tiene:

En el nodo V1

$$\frac{V_i - V_o}{R_1} = \frac{V_o}{Xc_1} \tag{4.12}$$

Luego,

$$\frac{V_i}{R_1} - \frac{V_o}{R_1} = \frac{V_o}{Xc_1} \rightarrow \frac{V_i}{R_1} = \frac{V_o}{R_1} + \frac{V_o}{Xc_1} \rightarrow \frac{V_i}{R_1} = V_o\left(\frac{1}{R_1} + \frac{1}{Xc_1}\right) \tag{4.13}$$

$$\frac{V_i}{V_o} = 1 + \frac{R_1}{Xc_1} \rightarrow \frac{V_i}{V_o} = \frac{1}{\frac{R_1}{Xc_1} + 1} \tag{4.14}$$

Por último reemplazando la definición de la reactancia capacitiva se obtiene la función de transferencia de primer orden:

$$G_{(s)} = \frac{1}{C_1 \times R_1 \times s + 1} \tag{4.15}$$

4.1.2.2 Planta de circuitos RC configurada como sistema de segundo orden

Un sistema de segundo orden está representado por el circuito de la siguiente figura, (malla1 y malla2).

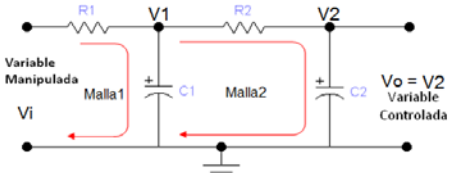


Figura 4.8. Diseño de la planta de circuitos RC configurada como sistema de segundo orden

Aplicando la ley de voltajes de Kirchhoff para cada nodo y la definición de la reactancia capacitiva en función de la transformada de Laplace, en el circuito que se muestra en la figura anterior, se tiene:

En el nodo V1

$$\frac{V_i - V_1}{R_1} = \frac{V_1}{Xc_1} + \frac{V_1 - V_o}{R_2} \tag{4.16}$$

En el nodo V2

$$\frac{V_1 - V_o}{R_2} = \frac{V_o}{Xc_2} \tag{4.17}$$

De 4.14

$$\frac{V_i}{R_1} = V_1\left(\frac{1}{R_1} + \frac{1}{Xc_1} + \frac{1}{R_2}\right) - \frac{V_o}{R_2} \tag{4.18}$$

De 4.15

$$V_1 - V_o = \frac{R_2 \times V_o}{Xc_2} \quad \rightarrow \quad V_1 = V_o\left(1 + \frac{R_2}{Xc_2}\right) \tag{4.19}$$

De 4.14 y 4.15

$$\frac{V_i}{R_1} = \left(V_o\left(1 + \frac{R_2}{Xc_2}\right)\right) \times \left(\left(\frac{1}{R_1} + \frac{1}{Xc_1} + \frac{1}{R_2}\right) - \frac{V_o}{R_2}\right) \tag{4.20}$$

$$\frac{V_i}{R_1} = V_o\left[\left(1 + \frac{R_2}{Xc_2}\right) \times \left(\left(\frac{1}{R_1} + \frac{1}{Xc_1} + \frac{1}{R_2}\right) - \frac{1}{R_2}\right)\right] \tag{4.21}$$

$$\frac{V_i}{V_o} = R_1\left[\left(1 + \frac{R_2}{Xc_2}\right) \times \left(\left(\frac{1}{R_1} + \frac{1}{Xc_1} + \frac{1}{R_2}\right) - \frac{1}{R_2}\right)\right] \tag{4.22}$$

$$\frac{V_o}{V_i} = \frac{1}{R_1\left[\frac{1}{R_1} + \frac{1}{Xc_1} + \frac{1}{R_2} + \frac{R_2}{Xc_2 \times R_1} + \frac{R_2}{Xc_1 \times Xc_2} + \frac{1}{Xc_2} - \frac{1}{R_2}\right]} \tag{4.23}$$

$$\frac{V_o}{V_i} = \frac{1}{1 + \frac{R_1}{Xc_1} + \frac{R_2}{Xc_2} + \frac{R_1 \times R_2}{Xc_1 \times Xc_2} + \frac{R_1}{Xc_2}} = \frac{1}{\frac{R_1 \times R_2}{Xc_1 \times Xc_2} + \frac{R_1}{Xc_1} + \frac{1}{Xc_2} \times (R_1 + R_2) + 1} \tag{4.24}$$

$$\frac{V_o}{V_i} = \frac{1}{c_1 \times c_2 \times R_1 \times R_2 \times s^2 + \left[\left(R_1 + R_2\right)c_2 + c_1 \times R_1\right]s + 1} \tag{4.25}$$

Por último reemplazando la definición de la reactancia capacitiva se obtiene la función de transferencia de segundo orden:

$$G_{(s)} = \frac{1}{c_1 \times c_2 \times R_1 \times R_2 \times s^2 + (R_1 \times c_2 + R_2 \times c_2 + c_1 \times R_1)\ s + 1} \qquad (4.26)$$

4.1.2.3 Planta de circuitos RC configurada como sistema de tercer orden

Un sistema de tercer orden está representado por el circuito de la siguiente figura, (malla1, malla2 y malla3).

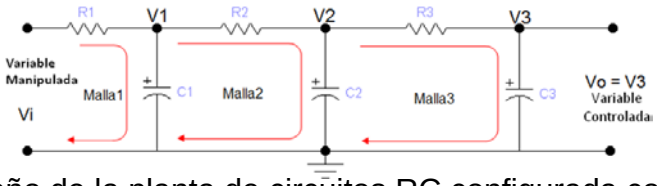


Figura 4.9. Diseño de la planta de circuitos RC configurada como sistema de tercer orden

Aplicando la ley de voltajes de Kirchhoff para cada nodo y la definición de la reactancia capacitiva en función de la transformada de Laplace, en el circuito que se muestra en la figura anterior, se tiene:

En el nodo V1:

$$\frac{V_i - V_1}{R_1} = \frac{V_1}{Xc_1} + \frac{V_1 - V_2}{R_2} \qquad (4.27)$$

De donde 4.25:

$$\frac{V_i}{R_1} = \frac{V_1}{R_1} + \frac{V_1}{Xc_1} + \frac{V_1}{R_2} - \frac{V_2}{R_2} \quad \rightarrow \quad \frac{V_i}{R_1} = V_1\left(\frac{1}{R_1} + \frac{1}{Xc_1} + \frac{1}{R_2}\right) - \frac{V_2}{R_2} \qquad (4.28)$$

En el nodo V2

$$\frac{V_1 - V_2}{R_2} = \frac{V_2 - V_o}{R_3} + \frac{V_2}{Xc_2} \qquad (4.29)$$

De donde 4.27:

$$V_1 - V_2 = \frac{V_2 \times R_2}{R_3} - \frac{V_o \times R_2}{R_3} + \frac{V_2 \times R_2}{Xc_2} \quad \rightarrow \quad V_1 = V_2\left(1 + \frac{R_2}{R_3} + \frac{R_2}{Xc_2}\right) - \frac{V_o \times R_2}{R_3} \qquad (4.30)$$

En el nodo V3

$$\frac{V_2 - V_o}{R_3} = \frac{V_o}{Xc_3} \qquad (4.31)$$

De donde 4.29:

$$V_2 - V_o = \frac{R_3 \times V_o}{Xc_3} \quad \rightarrow \quad V_2 = V_o\left(1 + \frac{R_3}{Xc_3}\right) \tag{4.32}$$

De 4.28 y 4.30:

$$V_1 = V_o\left(1 + \frac{R_3}{Xc_3}\right) \times \left(1 + \frac{R_2}{R_3} + \frac{R_2}{Xc_2}\right) - \frac{V_o \times R_2}{R_3} = V_o\left[1 + \frac{R_2}{R_3} + \frac{R_2}{Xc_2} + \frac{R_3}{Xc_3} + \frac{R_2}{Xc_3} + \frac{R_2 \times R_3}{Xc_2 \times Xc_3} - \frac{R_2}{R_3}\right] \tag{4.33}$$

De 4.26, 4.30 y 4.31

$$\frac{V_i}{R_1} = V_o\left[1 + \frac{R_2}{Xc_2} + \frac{R_3}{Xc_3} + \frac{R_2}{Xc_3} + \frac{R_2 \times R_3}{Xc_2 \times Xc_3}\right] \times \left(\frac{1}{R_1} + \frac{1}{Xc_1} + \frac{1}{R_2}\right) - \frac{V_o\left(1 + \frac{R_3}{Xc_3}\right)}{R_2}$$

$$\frac{V_i}{R_1} = V_o\left[\begin{array}{l} \frac{1}{R_1} + \frac{1}{Xc_1} + \frac{1}{R_2} + \frac{R_2}{R_1 \times Xc_2} + \frac{R_2}{Xc_1 \times Xc_2} + \frac{1}{Xc_2} + \frac{R_3}{R_1 \times Xc_3} + \frac{R_2}{Xc_1 \times Xc_3} + \frac{1}{Xc_3} + \\ \frac{R_2 \times R_3}{Xc_2 \times Xc_3 \times R_1} + \frac{R_2 \times R_3}{Xc_1 \times Xc_2 \times Xc_3} \\ + \frac{R_3}{Xc_2 \times Xc_3} - \frac{1}{R_2} - \frac{R_3}{R_2 \times Xc_3} \end{array}\right] \tag{4.34}$$

$$\begin{aligned} \frac{V_i}{V_o} = 1 + + & \frac{R_1}{Xc_1} + \frac{R_2}{Xc_2} + \frac{R_1 \times R_2}{Xc_1 \times Xc_2} + \frac{R_1}{Xc_2} + \frac{R_3}{Xc_3} + \frac{R_1 \times R_2}{Xc_1 \times Xc_3} \\ & + \frac{R_1}{Xc_3} + \frac{R_2 \times R_3}{Xc_2 \times Xc_3} + \frac{R_1 \times R_2 \times R_3}{Xc_1 \times Xc_2 \times Xc_3} + \frac{R_1 \times R_3}{Xc_2 \times Xc_3} - \frac{R_1 \times R_3}{R_2 \times Xc_3} \end{aligned} \tag{4.35}$$

$$\begin{aligned} \frac{V_i}{V_o} = & 1 + c_1 \times R_1 \times s + c_2 \times R_2 \times s + c_1 \times c_2 \times R_1 \times R_2 \times s^2 + \\ & c_2 \times R_1 \times s + c_3 \times R_3 \times s + c_1 \times c_3 \times R_1 \times R_2 \times s^2 + \\ & c_3 \times R_1 \times s + c_2 \times c_3 \times R_2 \times R_3 \times s^2 \\ & + c_1 \times c_2 \times c_3 \times R_1 \times R_2 \times R_3 \times s^3 + \\ & c_2 \times c_3 \times R_1 \times R_3 \times s^2 - \frac{R_1 \times R_3 \times c_3 \times s}{R_2} \end{aligned} \tag{4.36}$$

Por último reemplazando la definición de la reactancia capacitiva se obtiene la función de transferencia de tercer orden:

$$A = c1 \times c2 \times c3 \times R1 \times R2 \times R3$$
$$B = c1 \times c2 \times R1 \times R2 + c1 \times c3 \times R1 \times R2 + c2 \times c3 \times R2 \times R3 + c2 \times c3 \times R1 \times R3$$
$$D = c1 \times R1 + c2 \times R2 + c3 \times R3 + c2 \times R1 \times + c3 \times R1 - R1 \times R3 \times c3R2 - \frac{R1 \times R3 \times c3}{R2}$$

$$G_{(s)} = \frac{1}{As^3 + B\,s^2 + Ds + 1} \qquad (4.37)$$

Respuesta real y simulación de la planta de circuitos RC configurada como sistema de tercer orden

En la Figura 4.10 al lado derecho se puede apreciar una imagen de datos reales de la planta de circuitos RC, al introducir un escalón a la entrada de 10 voltios, (que corresponde a la línea azul), en la imagen se ve como la variable controlada (línea roja), no se estabiliza en el valor final del escalón aplicado, esto se debe a las caídas de tensión que se presentan las resistencias que componen cada malla.

Por otro lado en la misma figura al lado derecho se encuentra la simulación en Simulink del modelo de tercer orden de la planta de circuitos RC, en ella se puede apreciar el comportamiento ideal de la planta de circuitos RC, configurada como un sistema de tercer orden.

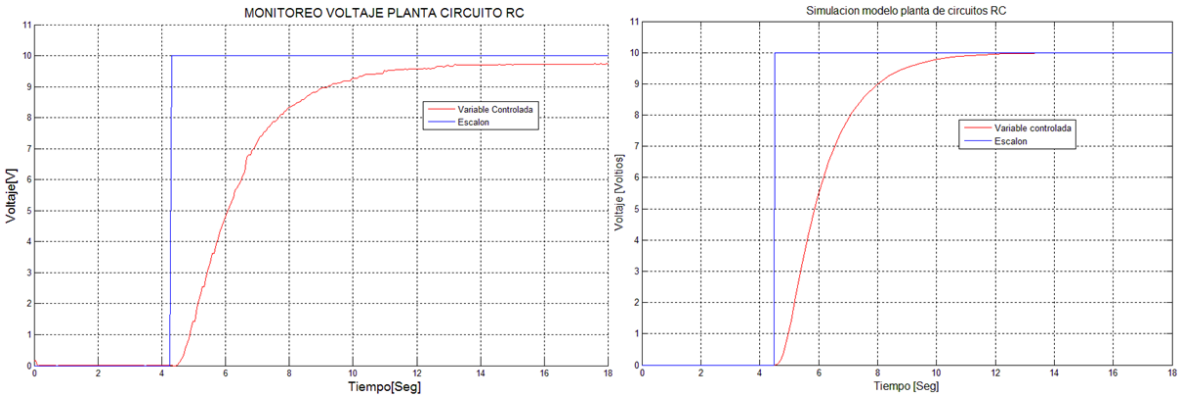


Figura 4.10. Respuesta real vs simulación de la planta de circuitos RC configurada como sistema de tercer orden

El error de estado estacionario de la planta de circuitos RC fue de 0.2 voltios.

4.2 Implementación de controladores para validar el sistema

En esta sección se encuentran descritos los diferentes controladores implementados para llevar a cabo la validación del sistema, los cuales son: un on/off y un proporcional para la planta de temperatura y un PID para la planta de circuitos RC. El esfuerzo de control de estos controladores está asociada al módulo PWM que se maneja a través de un registro de salida del

microcontrolador, el cual puede tomar como valor máximo 1000 y como valor mínimo 0, que corresponden al 100% y 0% de la energía entregada al sistema respectivamente; estos valores se asignan al esfuerzo de control dependiendo de la señal de error.
Para que el proceso de control sea satisfactorio, es necesario que los datos utilizados para tal fin contengan información significativa sobre el sistema, lo cual está directamente relacionado con el periodo de muestreo, puesto que un periodo de muestreo muy pequeño puede llevar a la obtención de datos redundantes, que no aportan información sobre él, mientras que un periodo de muestreo demasiado grande provoca grandes dificultades a la hora de identificar la dinámica del sistema.

Una regla comúnmente usada para seleccionar el valor máximo del periodo de muestreo, consiste en tomar una décima parte de la constante de tiempo del sistema.

$$\mathrm{h}_{\max} = \frac{\tau}{10} \tag{4.38}$$

4.2.1 Implementación de controlador on/off para la planta de temperatura

Un controlador de tipo on/off, actúa sobre la variable manipulada solo cuando la variable controlada sobre pasa el set point, ya sea con valores por encima o por debajo. La salida únicamente tiene dos estados, completamente apagada o completamente encendida. La principal característica del control on/off es que la variable controlada siempre estará oscilando alrededor del set point. La tasa a la cual la variable controlada oscila y la desviación de ésta respecto al set point dependen de las características dinámicas del sistema (CONSIDINE & McMILLAN, 1999).
La ley de control esta expresada por la siguiente ecuación:

$$U_{(t)} = \begin{cases} U_{max} \,\forall\, e_{(t)} > 0 \\ U_{min} \,\forall\, e_{(t)} < 0 \end{cases}$$

Dónde:
$U_{(t)}$: Es el esfuerzo de control.
U_{max} : Es el valor máximo que puede tomar el esfuerzo de control.
U_{min} : Es el valor mínimo que puede tomar el esfuerzo de control.

Para hallar el periodo de muestreo se toma el tiempo que la planta tarda en estabilizarse para hallar el τ del sistema, para lo cual se debe introducir un escalón al sistema para obtener el tiempo que tarda en llegar a una estabilización de alrededor del 70% para aplicar el criterio de estabilización de 5τ.

En el experimento se introdujo un cambio en la señal de PWM de 0% a 80%, con el fin de obtener una respuesta que describa el comportamiento de la planta en el rango de operación de la misma, la respuesta obtenida se muestra en la siguiente figura.

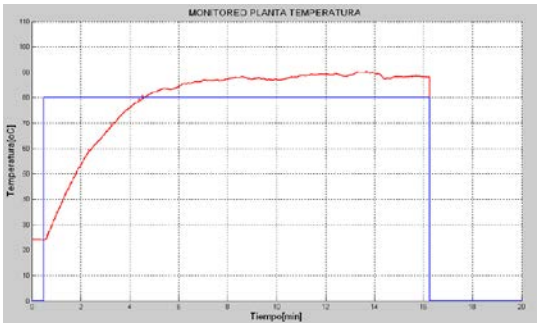


Figura 4.11.Respuesta de la planta de temperatura ante un escalón del 80%

De la imagen anterior se deduce el tiempo que tarda en llegar al 70%, desde que se introduce el escalón, lo cual corresponde aproximadamente a 390 segundos. Aplicando el criterio de estabilización se tiene:

$$5\tau = 390\ segundos \quad \rightarrow \quad \tau = \frac{390}{5} = 78\ \text{segundos}$$

$\tau = 78 segundos$ es la constante de tiempo del sistema, entonces el máximo valor que puede tomar el periodo de muestreo está dado por:

$$h_{max} = \frac{78}{10} = 7.8\text{segundos} \qquad (4.39)$$

Entonces el periodo de muestreo máximo que puede tener la planta de temperatura es de 7.8 segundos.

A continuación se describen los pasos para la implementación del controlador on/off.

- Se calcula el error.
- Si el error es mayor a cero se asigna al esfuerzo de control el valor máximo de la potencia, en este caso el 100% que corresponde a fijar el valor del registro asociado al PWM en 1000.
- Si el error es menor a cero se asigna al esfuerzo de control el valor mínimo de la potencia, en este caso el 0% que corresponde a fijar el valor del registro asociado al PWM en 1.
- Si el error es igual a cero se conserva el valor del esfuerzo de control.
- Posteriormente se asigna el valor del esfuerzo de control al registro asociado al módulo de PWM.

A continuación se muestra el código correspondiente a la implementación del controlador de tipo on/off en la planta de temperatura.

- Algoritmo de control:
  eK0 = SP - VC; // Calculo del error
  if (eK0>0) UT = 1000; // asigna el 100% de la potencia, 1000 es el valor máximo para el registro de la salida PWM
  if (eK0<0) UT = 1; // asigna el un mínimo de energía el valor mínimo es 0.
  if (eK0==0) UT = UT;

- Asignación del esfuerzo de control:
  pwm2 = (unsigned int16)(UT); // se asigna el valor del esfuerzo de control al registro de PWM
  set_pwm2_duty(pwm2);
  delay_ms(h); // retardo de tiempo h, periodo de muestreo

Dónde:
eK0: es el error actual del sistema.
SP: set point ingresado por el usuario desde la interfaz.
VC: variable controlada del proceso, en este caso temperatura.
UT: esfuerzo de control de la planta, es la salida del controlador.
pwm2: variable que contiene el valor que se le asigna al registro, debe tener valores enteros entre 0 y 1000, ya que la resolución del módulo PWM es de 10 bits.
h: periodo de muestreo.

En la Figura 4.12, se muestra la respuesta obtenida al implementar el controlador de tipo on/off, en principio con un set point de 45°C que luego cambia a 70°C. En la figura también se puede apreciar la característica principal de un control de tipo on/off, respecto a las oscilaciones sostenidas alrededor del set point.

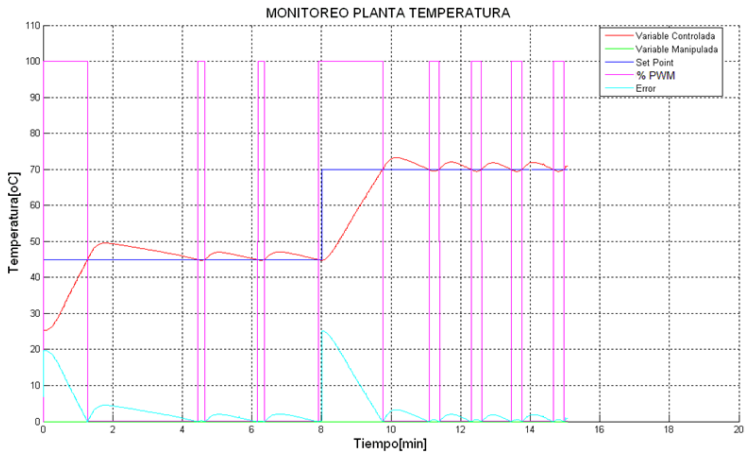


Figura 4.12. Respuesta de la planta de temperatura ante un controlador on/off

4.2.2 Implementación de un controlador proporcional para la planta de temperatura

La corrección generada por este tipo de controladores es proporcional a la señal de error. La siguiente ecuación describe el funcionamiento del controlador proporcional:
(LIPTÁK, 2006)

$$U_{(t)} = K_c e_{(t)} + b \tag{4.40}$$

Dónde:
$U_{(t)}$: es el esfuerzo de control
$e_{(t)}$: es la desviación de la variable controlada respecto al set point o señal de error
K_c: es la ganancia proporcional del controlador
BP: es la banda proporcional
b: es el bias, salida del controlador cuando el error es cero

El término K_c establece la sensibilidad que tendrá el controlador respecto a la señal de error, es decir cuánto cambia la salida del controlador por unidad de cambio en el error. El control proporcional disminuye el error, sin embargo no lo elimina completamente dejando una desviación llamada error de estado estacionario.

En la mayoría de procesos existe un valor límite para la ganancia proporcional $\boldsymbol{K_c}$, antes de que el proceso se vuelva inestable, esta ganancia última $\boldsymbol{K_{cu}}$, es el valor máximo que puede ajustarse en el controlador proporcional para mantener la estabilidad del sistema, por lo tanto el error en estado estacionario no puede ser eliminado completamente en un controlador proporcional (SMITH & CORRIPIO, Principles and Practice of Automatic Process Control, 1997).
A continuación se describe el proceso realizado para llevar a cabo la implementación del control proporcional en la planta de temperatura.

Se emplea el modelo de POMTM, que se obtuvo anteriormente por medio del criterio de la integral del cuadrado del error para aplicar la fórmula propuesta por Ziegler y Nichols en la sintonización de un controlador tipo proporcional.

$$G_{(s)} = \frac{K_P}{\tau s + 1} e^{-t_0 s} = \frac{1.3853}{284.3269 s + 1} e^{-13.5189 s} \tag{4.41}$$

La ecuación para hallar la ganancia proporcional, es la siguiente:

$$K_c = \frac{\tau}{K_P t_0} \tag{4.42}$$

Donde reemplazando los valores de los parámetros obtenidos en el modelo POMTM, se tiene:

$$K_c = \frac{284.3269}{(1.3853)(13.5189)} = 15.18 \qquad (4.43)$$

Luego de obtener el valor de la ganancia proporcional K_c, se impelmenta la siguiente secuencia en el código del controlador:

- Se calcula el error.
- Se implementa la ley de control que describe el funcionamiento del controlador proporcional.
- Se implementa un saturador que se encarga de limitar los valores máximo y mínimo del esfuerzo de control.
- Posteriormente se asigna el valor del esfuerzo de control al registro asociado al módulo de PWM.

A continuación se tiene el código implementado del controlador proporcional:

- Algoritmo de control:

  eK0 = SP - VC; // Calculo del error
  *UT = eK0*KP+ U0; // Ley de control Acción proporcional*
  // Las siguientes dos líneas son un saturador para el controlador, limitan el valor máximo y mínimo del esfuerzo de control
  if (UT>1000) UT = 1000;
  if (UT<0) UT = 0;

- Asignación del esfuerzo de control:

  pwm2 = (unsigned int16)(UT); // se asigna el valor del esfuerzo de control al registro de PWM
  set_pwm2_duty(pwm2);
  delay_ms(h); // retardo de tiempo h, periodo de muestreo

Dónde:
eK0: es el error actual del sistema.
SP: set point ingresado por el usuario desde la interfaz.
VC: variable controlada del proceso, en este caso temperatura.
UT: esfuerzo de control de la planta, es la salida del controlador.
pwm2: variable que contiene el valor que se le asigna al registro, debe tener valores enteros entre 0 y 1000, ya que la resolución del módulo PWM es de 10 bits.
h: periodo de muestreo

A continuación se muestran las respuestas obtenidas al implementar el controlador proporcional.

En la Figura 4.13 se muestra la respuesta de la planta al implementar el controlador proporcional con la ganancia hallada anteriormente, en ella se puede apreciar un pequeño error de estado estacionario al introducir un set point de 55ºC.

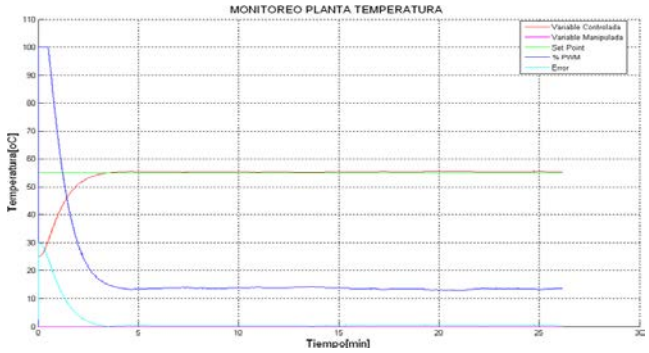


Figura 4.13. Respuesta de la planta de temperatura ante un controlador proporcional

La Figura 4.14 corresponde a una imagen de la respuesta de la planta al implementar el controlador proporcional con una ganancia de 50, lo cual hace que la variable controlada oscile alrededor del set point de 58ºC. Con esto se confirma que *ganancias grandes hacen la respuesta más oscilatorias* (SMITH, Automated Continous Process Control, 2002).

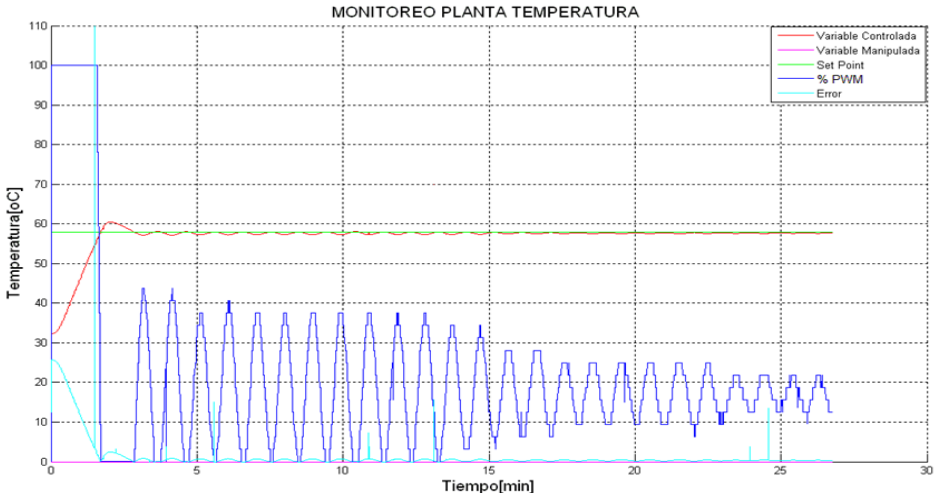


Figura 4.14. Respuesta planta de temperatura ante controlador proporcional con ganancia de 50

En la Figura 4.15 se muestra la respuesta obtenida al introducir un disturbio en el sistema, lo cual se logra teniendo la planta en modo manual y variando la velocidad del ventilador ubicado en la parte superior de la planta, esto último se hace a través de un potenciómetro. En la Figura 4.15 se puede apreciar como el

controlador proporcional compensa el cambio en la variable controlada rápidamente y llega a error de estado estacionario inferior a 1°C.

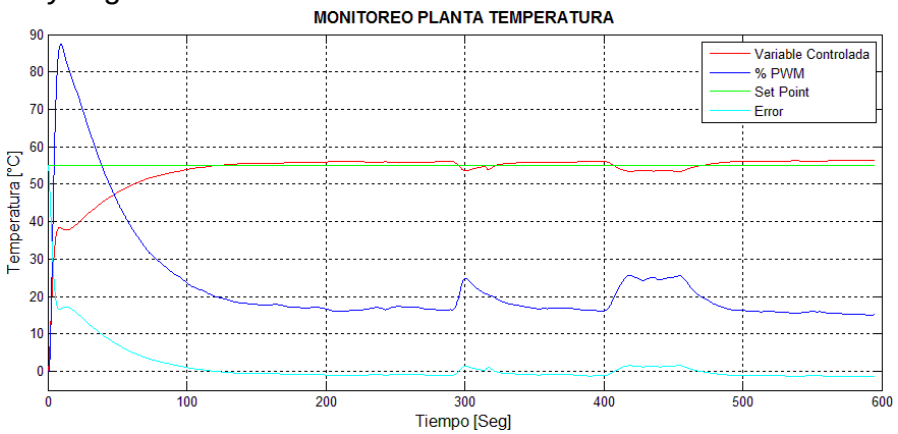


Figura 4.15. Corrección de disturbios en la planta de temperatura

4.2.3 Implementación de un controlador proporcional integral derivativo PID para la planta de circuitos RC

El controlador PID se implementa en la planta de circuitos RC configurada como sistema de tercer orden. Para hallar el periodo de muestreo se debe hallar la constante de tiempo del sistema τ, lo cual se hace de forma matemática sumando las constantes de tiempo de cada malla, que se obtienen operando los valores de los componentes del sistema, dado que en un sistema RC el tao esta dato por:

$$\tau = \mathrm{R} \times \mathrm{C} \qquad (4.44)$$

Dónde:
R: resistencia de malla
C: capacitor de malla

$$\tau = \underset{Malla\,1}{(330\mu f \times 2k\Omega)} + \underset{Malla\,2}{(220\mu f \times 2k\Omega)} + \underset{Malla\,3}{(100\mu f \times 2k\Omega)} = 1.3 segundos \qquad (4.45)$$

Luego $\tau = 1.3\, segundos$ es la constante de tiempo del sistema, entonces el máximo valor que puede tomar el periodo de muestreo está dado por:

$$\mathrm{h}_{\max} = \frac{1.3}{10} = 0.13 \mathrm{segundos} \qquad (4.46)$$

Entonces el periodo de muestreo máximo que puede tener la planta de circuitos RC es de 130 milisegundos.

A continuación se describe el proceso para la implementación del controlador PID.

El controlador PID implementado en la planta de circuitos RC es un controlador tipo ideal también es conocido como controlador PID no interactivo, debido a que el componente integral no tiene influencia sobre el componente derivativo y viceversa, es decir estos componentes no interactúan entre sí en el dominio del tiempo, aunque la ganancia proporcional si influye en los tres modos. Este algoritmo admite ceros complejos lo cual resulta útil cuando se controla sistemas con polos oscilatorios (ASTRÖM & HÄGGLUND, 2006).

La función de transferencia de un controlador PID ideal se muestra en la siguiente ecuación.

$$G_{PID(s)} = K_p\left(1+\frac{1}{T_i s}+T_d s\right) \tag{4.47}$$

El diagrama en bloques del controlador PID ideal se muestra en la siguiente figura.

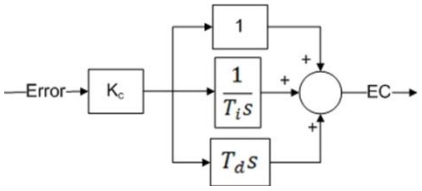


Figura 4. 16. Diagrama en bloques del controlador PID ideal

Teniendo la ecuación 4.45 de la función de transferencia del PID ideal expresada como:

$$\frac{U_{T(s)}}{E_{k(s)}} = K_p\left(1+\frac{1}{T_i s}+T_d s\right) \tag{4.48}$$

Dónde:
$U_{t(s)}$: es el esfuerzo de control.
$E_{k(s)}$: es el error.
K_p: es la constante proporcional.
T_i: es la constante integral.
T_d: es la constante derivativa.

Se aplica el método de la derivada para discretizar la función anterior, este método consiste en aproximar la derivada por la pendiente de la recta que pasa por dos muestras consecutivas, con lo cual se obtiene:

$$\frac{dy_{(t)}}{dt} \rightarrow \frac{y_k - y_{k-1}}{T} \tag{4.49}$$

Que visto en sus correspondientes transformadas se convierte en:

$$sY_{(s)} \ \underline{z} \ \frac{1-z^{-1}}{T} \ Y_{(z)} \tag{4.50}$$

Donde T corresponde al periodo de muestreo.

Por lo tanto para obtener el PID discreto basta con sustituir en la función de transferencia las s por $\frac{1-z^{-1}}{T}$ (CUADRADO, 2009).

Discretizando la ecuación 4.46 que corresponde a la función de transferencia del PID ideal, se tiene:

$$\frac{U_{T(z)}}{E_{k(z)}} = K_p\left(1 + \frac{T}{T_i}\frac{1}{1-z^{-1}} + (1-z^{-1})\frac{T_d}{T}\right)$$

Teniendo en cuenta que los valores $\frac{T}{T_i}$ y $\frac{T_d}{T}$ son constantes, entonces se hace el siguiente cambio:

$$\frac{U_{T(z)}}{E_{k(z)}} = K_p\left(1 + a\frac{1}{1-z^{-1}} + \left(1-z^{-1}\right)b\right) \tag{4.51}$$

Dónde:

$$a = \frac{T}{T_i}$$

$$b = \frac{T_d}{T}$$

Sacando factor común se tiene:

$$\frac{U_{T(z)}}{E_{k(z)}} = K_p\left(\frac{1-z^{-1}+a+\left(1-z^{-1}\right)^2 b}{1-z^{-1}}\right) \tag{4.52}$$

Desarrollando el polinomio:

$$\frac{U_{T(z)}}{E_{k(z)}} = K_p\left(\frac{1-z^{-1}+a+(1-2z^{-1}+z^{-2})b}{1-z^{-1}}\right) \tag{4.53}$$

Luego:

$$U_{T(z)}\left(1-z^{-1}\right) = E_{k(z)}K_p(1-z^{-1}+a+(1-2z^{-1}+z^{-2})b) \tag{4.54}$$

De donde:

$$U_{T(z)}\left(1-z^{-1}\right) = E_{k(z)}K_p(1-z^{-1}+a+b-b2z^{-1}+bz^{-2}) \tag{4.55}$$

Factorizando:

$$U_{T(z)}\left(1-z^{-1}\right)=E_{k(z)}K_p(1+a+b+(1+2b)z^{-1}+bz^{-2}) \tag{4.56}$$

Luego:

$$U_{T(z)}\left(1-z^{-1}\right)=K_1E_{k(z)}+K_2E_{k(z)}z^{-1}+K_3E_{k(z)}z^{-2} \tag{4.57}$$

Dónde:

$$K_1=K_p\left(1+a+b\right)\rightarrow K_1=K_p\left(1+\frac{T}{T_i}+\frac{T_d}{T}\right) \tag{4.58}$$

$$K_2=-K_p\left(1+2b\right)\rightarrow K_2=-K_p\left(1+2\frac{T_d}{T}\right) \tag{4.59}$$

$$K_3=K_pb\rightarrow K_3=K_p\frac{T_d}{T} \tag{4.60}$$

Despejando $U_{t(z)}$:

$$U_{T(z)}=K_1E_{k(z)}+K_2E_{k(z)}z^{-1}+K_3E_{k(z)}z^{-2}+U_{T(z)}z^{-1} \tag{4.61}$$

Al pasar la ecuación 4.59, a ecuaciones en diferencias se obtiene:

$$U_{T(h)}=K_1E_{k(h)}+K_2E_{k(h-1)}+K_3E_{k(h-2)}+U_{T(h-1)} \tag{4.62}$$

La ecuación 4.60, corresponde a la ley de control implementada en la tarjeta principal, en la cual se encuentra con las siguientes variables, (IBRAHIM, Microcontroller Base Applied Digital Control, 2006):

$$U_{T0}=K_1E_{k0}+K_2E_{k1}+K_3E_{k2}+U_{T1} \tag{4.63}$$

Dónde:
U_{T0}: es la salida actual del controlador.
E_{k0}: es el error actual del sistema.
E_{k1}: es el error anterior del sistema.
E_{k2}: es el error anterior al error E_{k1} del sistema.
U_{T1}: es la salida anterior del controlador.

Sintonización del controlador PID por el método de la ganancia ultima de Ziegler y Nichols

Empleando la herramienta rltool de Matlab, se obtiene el lugar geométrico de las raíces, correspondiente a la respuesta de la planta en lazo cerrado del sistema de

3er orden, esto se hace con el fin de hallar la ganancia última K_u y la frecuencia natural del sistema w_n. En la siguiente figura se muestra la imagen obtenida.

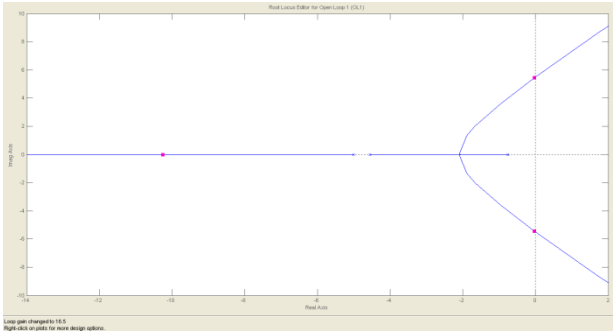


Figura 4.17. Lugar geométrico de las raíces de la planta de circuitos RC

Del procedimiento anterior se tiene que:

$$K_u = 16.5$$
$$w_n = 5.44\, rad/seg$$

Teniendo la ganancia ultima y la frecuencia natural del sistema se implementa el método de sintonización de Ziegler y Nichols, para obtener los parámetros K_p, T_i y T_d.
Las ecuaciones de sintonización del controlador PID son:

$$K_p = 0.6K_u \; a \; 1.0K_u$$
$$T_i = 0.5T_u$$
$$T_d = 0.125T_u$$

Donde T_u es el periodo ultimo y se halla asi:

$$T_u = \frac{2\pi}{w_n} \qquad (4.64)$$

Los valores obtenidos para los parámetros del controlador PID son:

$$K_p = 13.1534$$
$$T_i = 0.6400$$
$$T_d = 0.1629$$

Teniendo los valores K_p, T_i y T_d, se pueden hallar los valores de las constantes K_1, K_2 y K_3, presentes en la ley de control expresada en ecuaciones en diferencias hallada anteriormente. Luego:

$$K_1 = 61.1228$$
$$K_2 = -106.7691$$
$$K_3 = 46.7678$$

- Algoritmo de control:

eK0 = SP - VC; // Calculo del error
*UT = UT1 + K1*eK0+K2*eK1+K3*eK2;*
if (UT>1000) UT = 1000;
if (UT<1) UT = 1;

- Asignación del esfuerzo de control

pwm2 = (unsigned int16)(UT);
set_pwm2_duty(pwm2);
Actualización de variables del controlador.
eK2 = eK1;
eK1 = eK0;
UT1 = UT;
delay_ms(h); // retardo, periodo de muestreo

Dónde:
eK0: es el error actual del sistema.
SP: set point ingresado por el usuario desde la interfaz.
VC: variable controlada del proceso, en este caso temperatura.
UT: esfuerzo de control de la planta, es la salida del controlador.
pwm2: variable que contiene el valor que se le asigna al registro, debe tener valores enteros entre 0 y 1000, ya que la resolución del módulo PWM es de 10 bits.
h: periodo de muestreo.

En la Figura 4.18 se muestra la respuesta obtenida al implementar el controlador PID. En ella se puede apreciar como el controlador estabiliza la planta alrededor de los 6 segundos, no presenta oscilaciones y tiene un error de estado estable igual a cero.

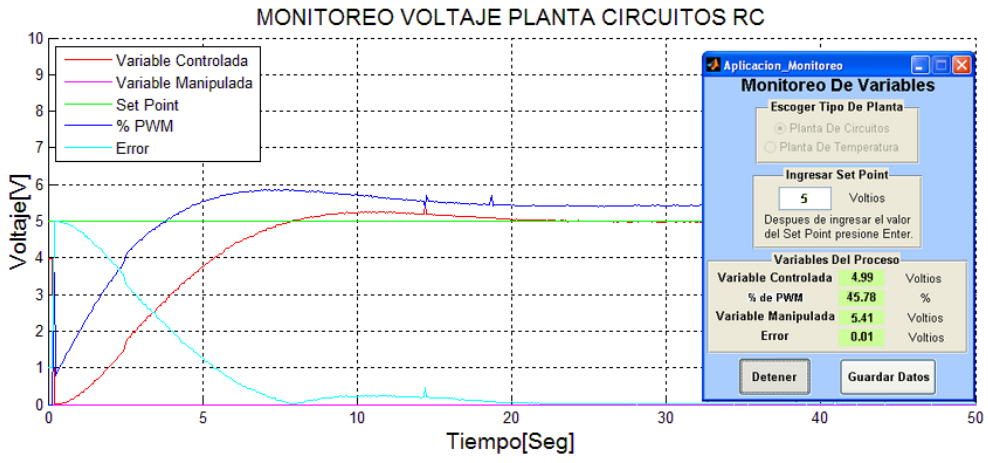


Figura 4.18. Respuesta de la planta de circuitos RC ante un controlador PID

En la Figura 4.19 se muestra la respuesta obtenida al introducir un disturbio en el sistema, lo cual se logra deshabilitando y luego habilitando nuevamente la resistencia de carga del sistema. En ella se puede apreciar como el controlador compensa el cambio en la variable controlada rápidamente y llega a error de estado estacionario igual a cero.

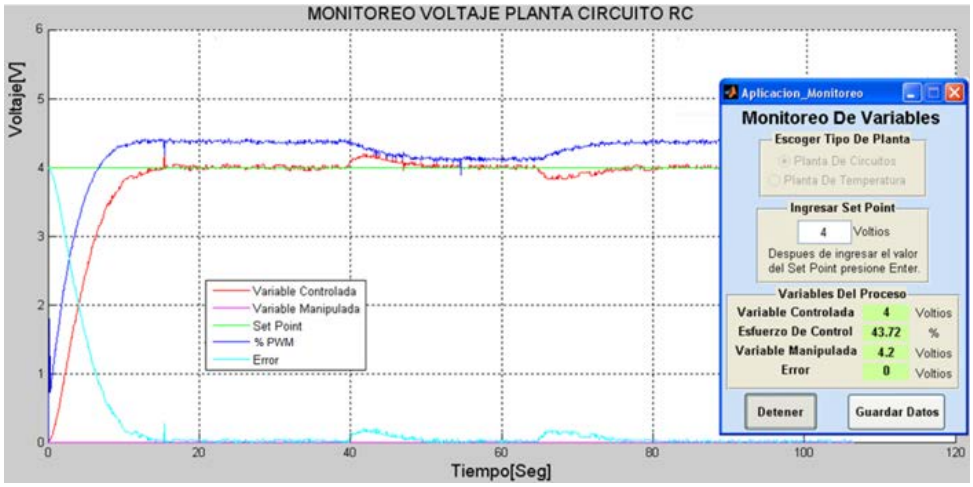


Figura 4.19. Corrección de disturbios en la planta de circuitos RC

CONCLUSIONES

Como resultado del desarrollo de este proyecto se creó un sistema que permite aplicar de forma práctica diferentes conocimientos teóricos de áreas, tales como: sistemas continuos, sistemas discretos, caracterización por curvas de reacción, técnicas de modelado e implementación de controladores, entre otros.

Mediante técnicas de prototipado electrónico se logró desarrollar la parte hardware del sistema, la cual está compuesta por una tarjeta principal con comunicación USB y dos tarjetas correspondientes a las plantas experimentales. Las tarjetas desarrolladas cuentan con ventajas tales como tamaño reducido, lo cual permite una alta portabilidad, bajo consumo y bajo costo en comparación a sistemas comerciales similares. Lo anteriormente mencionado facilita tanto el uso como la reproducción del sistema.

Se realizaron procesos de diseño, implementación y adecuación de software para crear un entorno de trabajo mediante el cual el usuario pueda hacer uso del sistema para el desarrollo de sus prácticas de forma rápida. En el proceso se tuvieron en cuenta ciertos criterios de diseño como: crear interfaces amenas intuitivas y simples para facilitar el manejo por parte del usuario, permitir flexibilidad en cuanto a las posibilidades para realización de prácticas, entre otros.

Se desarrollaron diferentes técnicas teórico – prácticas para el modelado de sistemas, con el fin de obtener un modelo matemático que permita la caracterización de las plantas didácticas. Dichos modelos fueron aplicados en el diseño e implementación de controladores.

Se validó el sistema mediante la realización y desarrollo de ejemplos prácticos donde se muestra al usuario las opciones y capacidades que brinda el sistema específicamente en temas como modelado y análisis de procesos e implementación de esquemas de control.
Durante la validación se muestra que el sistema para la implementación de controladores digitales es versátil, y se convierte en una herramienta importante para afirmar y aplicar conocimientos prácticos de áreas como control digital, instrumentación e identificación de sistemas.

BIBLIOGRAFÍA

ASTRÖM, K., & HÄGGLUND, T. (2006). *PID Controllers: Theory, Design, and Tuning. Advanced PID Control.* ISA.

AXELSON, J. (2009). *USB Complete The Developer's Guide.* Lakeview Research.

BARRAGAN, D. (5 de Mayo de 2008). *Manual de interfaz gráfica de usuario en Matlab.* Recuperado el 3 de Mayo de 2011, de Repositorio de la Escuela Politécnica Nacional: http://www.dspace.espol.edu.ec/bitstream/123456789/10740/11/MATLAB_GUIDE.pdf

BIBLIOMAN. (2010). *Comunicación USB con el PIC 18F4550*. Recuperado el 3 de Mayo de 2011, de Aqui Hay Apuntes : http://aquihayapuntes.com/indice-practicas-pic-en-c/comunicacion-usb-pic18f4550-utilizando-la-clase-cdc.html

CONSIDINE, D., & McMILLAN, G. (1999). *Process/Industrial Instruments and Controls Handbook* (5 ed.). McGraw-Hill.

CUADRADO, A. A. (18 de Mayo de 2009). *Control Digital.* Recuperado el 3 de Mayo de 2011, de Universidad de Oviedo, Área de Ingenieria de Sistemas y Automática: http://isa.uniovi.es/~cuadrado/archivos/cdigitalbase.pdf

Electrónica Unicrom. (s.f.). *Amplificador Operacional no inversor*. Recuperado el 3 de Mayo de 2011, de Electrónica Unicrom: http://www.unicrom.com/Tut_OpAmpNoInversor.asp

EMCELETTRONICA. (2008). *How-to use PWM to Generate Analog (or Analogue) Voltage in Digital*. Recuperado el 3 de Mayo de 2011, de Your Electronics Open Source: http://dev.emcelettronica.com/how-to-use-pwm-to-generate-analog-or-analogue-voltage-digital-circuits-part-2

FUJITSU. (2000). *Manual de formación de USB.* Recuperado el 3 de Mayo de 2011, de Fujitsu: http://www.fujitsu.com/downloads/EU/es/soporte/escaneres/cursousb.pdf

GALÍNDEZ, J. R., & JARAMILLO, O. (2011). Sistema de prototipado rápido de control para el módulo de prácticas mic955 de la empresa Feedback. *Tesis de pregrado en Ingenieria en Automatica Industrial, Universidad del Cauca*. Popayán.

GARCÍA Breijo, E. (2008). *Compilador C CCS y simulador Proteus para microcontroladores PIC.* Marcombo Ediciones Técnicas.

GARCIA, C. (2010). *El USB desencadenado.* Recuperado el 3 de Mayo de 2011, de PicMania: http://picmania.garcia-cuervo.net/usb_0_desencadenado.php

GONZÁLEZ, J. (2003). *Grabación de microcontroladores PIC.* Recuperado el 3 de Mayo de 2011, de IEA ROBOTICS: http://www.iearobotics.com/proyectos/cuadernos/ct4/ct4.html

HERRERA, V. (s.f.). *Funciones de la librería USB (mpusbapi.dll).* Recuperado el 3 de Mayo de 2011, de Scribd: http://es.scribd.com/doc/41805431/Funciones-de-La-Libreria-Usb-110

IBRAHIM, D. (2006). *Microcontroller Base Applied Digital Control.* Cyprus: Department of Computer Engineering Nerar Eat University.

LIPTÁK, B. (2006). *Process Control and Optimization* (Vol. II). CRC Press - ISA Press.

LÓPEZ, E. (9 de Agosto de 2005). *Protocolo USB (universal serial bus).* Recuperado el 3 de Mayo de 2011, de Ingeniería en microcontroladores: http://www.i-micro.com/pdf/articulos/usb.pdf

MARCHAND, P., & HOLLAND, T. (2003). *Graphics and GUIs with MATLAB* (3 ed.). Chapman & Hall/Crc.

Mathworks. (2011). *Using a MATLAB Timer Object*. Recuperado el 3 de Mayo de 2011, de Mathworks: http://www.mathworks.com/help/techdoc/matlab_prog/f9-38055.html

MAXIM. (2010). *DS18S20 High-Precision 1-Wire Digital Thermometer.* Recuperado el 3 de Mayo de 2011, de Maxim Innovation Delivered: http://datasheets.maxim-ic.com/en/ds/DS18S20.pdf

Microchip Technology Incorporated. (2004). *Picdem Fs usb demonstration board users guide, sección 3.5 Bootload Mode.* Recuperado el 3 de Mayo de 2011, de Microchip: http://ww1.microchip.com/downloads/en/devicedoc/51526a.pdf

Microchip Technology Incorporated. (2007). *PIC18F2455/2550/4455/4550 Data Sheet 28/40/44-Pin, High Performance, Enhanced Flash, USB Microcontrollers with nanoWatt Technology, sección 21.3.* Recuperado el 3 de Mayo de 2011, de http://ww1.microchip.com/downloads/en/devicedoc/39632d.pdf

PALAZZESI, A. (2008). *CCS - Uso de LCD's alfanuméricos.* Recuperado el 3 de Mayo de 2011, de uControl: http://www.ucontrol.com.ar/wiki/index.php?title=CCS_-_LCD#Funciones_en_LCD.C

POOL, G. J. (4 de Julio de 2009). Transferencia y procesamiento de datos a alta velocidad, mediante el uso de Matlab, el puerto USB 2.0 y PIC18F2455 de Micochip. *Universidad Modelo, Escuela de Ingeniería, Maestría en Mecatrónica*. Mérida, Yucatán, México.

RENGIFO, C. F. (2011). Electiva Implementacion de Controladores PID. *Apuntes de clase. Universidad del Cauca*. Popayán.

SMITH, C. (2002). *Automated Continous Process Control.* New York: Jhon Wiley & Sons, Inc.

SMITH, C., & CORRIPIO, A. (1997). *Principles and Practice of Automatic Process Control* (2 ed.). John Wiley & Sons.

TACCONI, E., MANTZ, R., & SOLSONA, J. (2005). Controladores Basados en Estrategias PID. LEICI, Facultad de Ingeniería, UNLP. La Plata, Argentina.

Un Poco de Electronica. (2009). *Generador automático para los drivers USB de Microchip.* Recuperado el 3 de Mayo de 2011, de Un Poco de Electronica: http://www.unpocodelectronica.netau.net/generador-de-inf-para-los-drivers-usb-de-microchip

ZIEGLER, J., & NICHOLS, N. (1942). *Optimum settings for automatic controllers* (Vol. 64). Trans. ASME.