SISTEMA DIDÁCTICO PARA LA IMPLEMENTACIÓN DE CONTROLADORES DIGITALES

Anexo 1: Manual de usuario

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

TABLA DE CONTENIDO

1. Introducción y descripción general del sistema

La teoría de control es un aspecto fundamental en diferentes campos de la ingeniería especialmente en la Automatización Industrial, en donde es importante conocer sobre el comportamiento de diferentes procesos físicos de modo que el ingeniero pueda diseñar esquemas de control adecuados para realizar una tarea o trabajo según se requiera, por esta razón entre otras importantes es necesario para el futuro ingeniero que el conocimiento en las diferentes áreas de control sea soportado por bases firmes tanto a nivel teórico como practico.

El sistema para implementación de controladores digitales surge como una herramienta orientada y diseñada como apoyo en la realización de diferentes prácticas de control, de modo que contribuya en el aprendizaje y generación de conocimiento teórico-práctico sobre los sistemas de control.

Se presenta al usuario (docentes y estudiantes) un sistema que permite diseñar e implementar diferentes prácticas sobre sistemas de control, el cual está conformado por elementos tanto software como hardware, en cuanto al hardware se tienen tres tarjetas:

- Tarjeta principal de control
- Planta de temperatura.
- Planta de Circuitos RC

El sistema está diseñado con una estructura modular de modo que se puedan intercambiar las diferentes tarjetas, las plantas específicamente, de modo que el usuario pueda utilizar la misma tarjeta principal con cualquiera de las plantas sin tener que realizar cambios drásticos en cuanto a software y hardware. En referencia al 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 esta un software que permite un monitoreo del sistema desde un computador facilitando que el usuario pueda analizar el comportamiento de los procesos de forma natural y de forma controlada según las técnicas de control que se implementen en las practicas.

2. Hardware del sistema

En esta sección se presenta una descripción detallada de cada tarjeta que compone el hardware del sistema

En esta sección se presenta una descripción de cada tarjeta que compone el hardware del sistema.

2.1 Tarjeta Principal

En esta sección se describe la parte hardware de la tarjeta principal del sistema, en la siguiente figura encontrara las partes que la conforman.

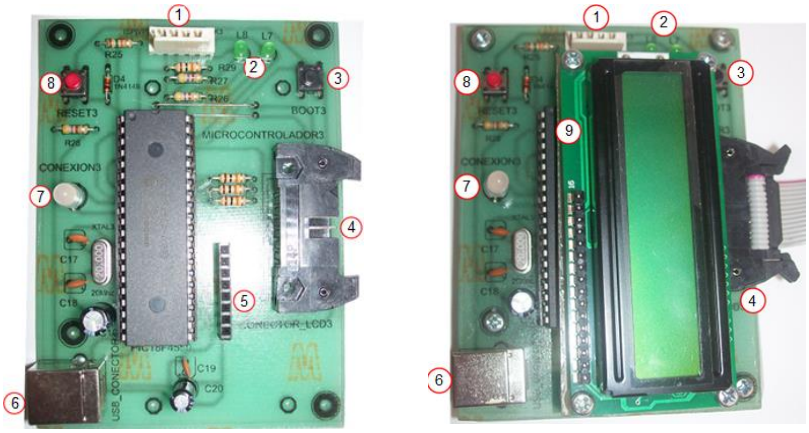


Figura 1

1. **Conector programación ICSP:** conector para programación in-circuit de la tarjeta principal.
2. **Indicadores de estado bootloader:** led´s que cuando están encendidos de manera intermitente indican que la tarjeta se encuentra en modo bootloader.
3. **Pulsador bootloader:** pulsador que al oprimirse junto con el pulsador de *reset* permite que la tarjeta principal entre en modo de programación por bootloader.
4. **Conector para plantas experimentales:** conector IDE para cable ribbon o cinta plana de 14 pines; en la siguiente figura se muestra la conexión física entre el conector y el microcontrolador, la descripción detallada de la señal de cada pin se presenta en la siguiente tabla.

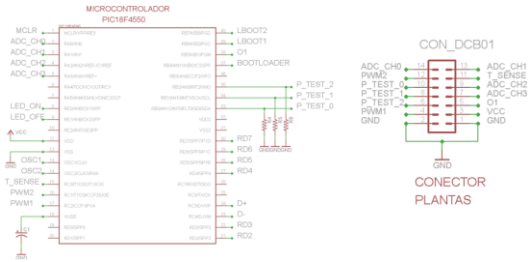


Figura 2

Tabla 1 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, puede 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

5. **Conector LCD:** conector de 8hilos necesarios para la conexión entre la LCD y la tarjeta principal.
6. **Conector USB:** puerto USB estándar para la comunicación entre la tarjeta principal y el PC. A través de este conector la tarjeta recibe la alimentación +5VDC con una corriente máxima de 500mA. El cable que conecta entre la tarjeta principal y el PC debe tener un conector tipo A en el extremo que se conecta al PC y uno tipo B en el extremo correspondiente a la tarjeta.

7. **Indicador de estado de conexión USB:** indicador tipo dual LED para verificar el estado de la conexión USB, cuando esta de color rojo indica que la tarjeta no ha sido reconocida, por el contrario si su color es verde la tarjeta esta lista para ser usada.
8. **Pulsador reset:** este pulsador tiene dos utilidades, una es reiniciar la tarjeta principal, la otra es que al ser oprimido junto con el pulsador bootloader, se logra que la tarjeta entre en modo bootloader.
9. **Pantalla LCD:** la pantalla LCD permite visualizar variables y estados que ayudan en el proceso de diseño y depuración de código.

Nota: Recordar que la secuencia para ingresar en modo bootloader es: primero oprimir el pulsador bootloader, luego el pulsador reset, posteriormente se debe liberar el pulsador reset y por último el pulsador bootloader.

En la Tabla 2 se resumen las principales características de la tarjeta principal.

Tabla 2 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 (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.

a. Planta de temperatura

En esta sección se describe la parte hardware de la planta de temperatura del sistema, en la siguiente figura encontrara las partes que la conforman.

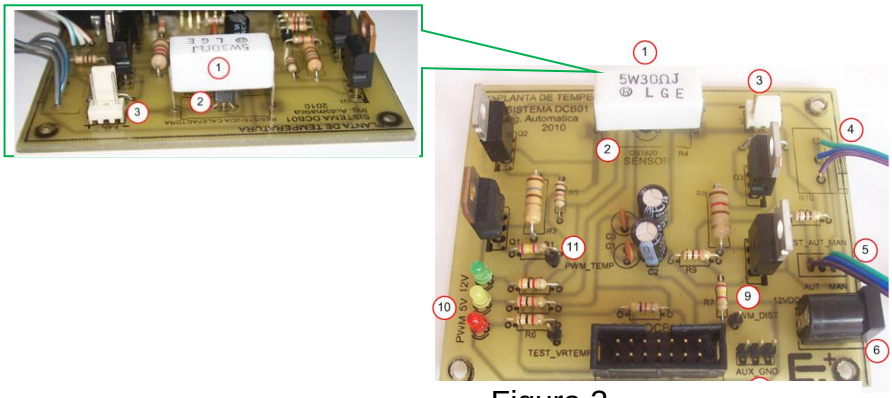


Figura 3

1. **Resistencia calefactora:** resistencia a base de carbón con recubrimiento de cerámica, con alimentación de tipo DC.
2. **Sensor de temperatura:** circuito integrado DS18S20, es el sistema sensor – trasmisor de la planta.
3. **Conector ventilador:** conector de dos pines en el cual se conecta el ventilador usado para introducir disturbios en la planta.
4. **Potenciómetro, variación del disturbio:** regula la intensidad del disturbio a través del control de la velocidad del motor del ventilador.
5. **Interruptor:** permite la selección para manejo automático o manual del disturbio, actualmente el disturbio solo se puede introducir de forma manual al sistema.
6. **Conector fuente 12VDC:** conector a través del cual se alimenta la planta.
7. **Puntos de GND auxiliar:** puntos de tierra auxiliar para realizar mediciones.
8. **Punto de prueba PWM1:** punto de prueba en el cual se puede medir mediante un osciloscopio la señal de PWM1 correspondiente al disturbio.
9. **Conector a tarjeta principal:** conector a través del cual se envían y reciben señales de la tarjeta principal. Por este medio envía el código binario de tres (3) bits de reconocimiento de la planta, que en este caso corresponde al código 010.
10. **Pilotos indicadores:** se trata de un conjunto de tres led´s que están asociados al estado de las señales. El led amarillo corresponde a la señal de +5VDC, el led verde corresponde a la señal +12VDC y el led rojo corresponde a la señal la señal de control PWM aplicada a la etapa de potencia de la resistencia calefactora.
11. **Punto de prueba PWM2:** punto de prueba en el cual se puede medir mediante un osciloscopio la señal de PWM2 correspondiente a la señal de control PWM aplicada a la etapa de potencia de la resistencia calefactora.

En la Tabla 3 se resumen las principales características de la planta de temperatura.

Tabla 3 Características de la planta de 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 TX empaquetado)
Voltaje de operación	12VDC
Señal de control	PWM 3KHZ
Disturbio experimental	Ventilador controlado por PWM

En la siguiente figura se muestra una imagen de la planta de temperatura, en ella se puede observar en la parte superior el ventilador que genera el disturbio y en la parte lateral derecha: el potenciómetro para regular de forma manual el disturbio y el interruptor que permite la selección para manejo automático o manual del disturbio.

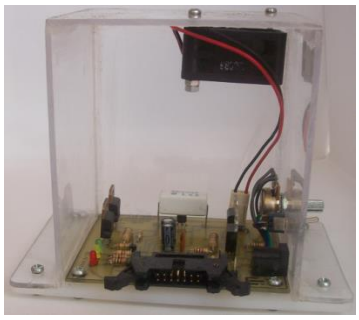


Figura 4

2.2 Planta de circuitos RC

En esta sección se describe la parte hardware de la planta de circuitos RC del sistema, en la siguiente figura encontrara las partes que la conforman.

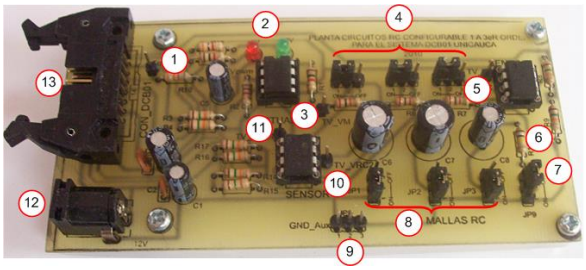


Figura 5

1. **Punto de prueba PWM:** punto de prueba en el cual se puede medir mediante un osciloscopio la señal de PWM aplicada al actuador de la planta.
2. **Pilotos indicadores:** se trata de un conjunto de dos led´s que están asociados al estado de las señales. El led rojo corresponde a la señal de +5VDC y el led verde corresponde a la señal +12VDC.
3. **Punto de prueba VM:** punto en el cual se puede medir la variable manipulada.
4. **Jumpers para configuración de resistencias:** jumpers habilitadores y deshabilitadores de las resistencias de cada malla.
5. **Punto de prueba VRC1:** punto en el cual se puede medir el voltaje de salida de la malla 1, variable controlada en malla 1.
6. **Resistencia de carga:** resistencia de 100KΩ, que al habilitarla o deshabilitarla introduce un error en el sistema.
7. **Jumper para configuración de resistencia de carga:** jumper habilitador y deshabilitador de la resistencia de carga.
8. **Jumpers para configuración de capacitores:** jumpers habilitadores y deshabilitadores de los capacitores de cada malla.
9. **Puntos de GND auxiliar:** puntos de tierra auxiliar para realizar mediciones.
10. **Punto de prueba VRC2:** punto en el cual se puede medir el voltaje de salida de la malla 3, variable controlada en malla 3.
11. **Punto de prueba VRC1:** punto en el cual se puede medir el voltaje de salida de la malla 2, variable controlada en malla 2.
12. **Conector fuente 12 VDC:** conector a través del cual se alimenta la planta.
13. **Conector a tarjeta principal:** conector a través del cual se envían y reciben señales a la tarjeta principal. Por este medio envía el código binario de tres (3) bits de reconocimiento de la planta, que en este caso corresponde al código 001.

En la Tabla 4se resumen las principales características de la planta de circuitos RC.

Tabla 4 Características planta de circuitos RC

Parámetro	**Detalle**
Alimentación	12 VDC
Orden del sistema	1er, 2do y 3er orden configurable por hardware mediante jumpers
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 de control	PWM
Variables Medibles	Voltaje de entrada (Variable manipulada) Voltaje malla 1, malla2 y malla3, (Variable controlada) Señal PWM (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.

2.3 Sistema de alimentación

La tarjeta principal se alimenta directamente de las señales de voltaje que genera el bus USB del PC, este provee la corriente necesaria para que pueda funcionar correctamente, y a su vez hace que el sistema sea más económico, dado que no se necesitan componentes adicionales para adecuar las señales, ni fuentes de alimentación externa.

El sistema está equipado con una fuente externa de 12 voltios con una corriente de hasta 1.5 amperios, la cual se encarga de alimentar tanto a la planta de temperatura como a la de circuitos RC.

Es importante que la fuente tenga estas características ya que la planta de temperatura consume alrededor de medio amperio, además es de fácil consecución en el mercado, lo cual es importante dado que en caso de avería o falla se puede reemplazar sin ningún inconveniente.

En la siguiente figura se muestra una imagen de la fuente externa del sistema.

Figura 6

3. Software del sistema

Figura 7

Como se menciona en la introducción de este manual de usuario el sistema se compone de componentes tipo software, específicamente se puede hablar de software para operación, monitoreo y programación de la tarjeta principal desde el computador y software para desarrollo de código para la tarjeta principal a

continuación se hace una lista con una breve descripción de cada componente software que requiere el sistema.

Software para desarrollo de código (programación) de la tarjeta principal:

Para permitir al usuario desarrollar e implementar sus propias prácticas hace uso de las herramientas **Mplab IDE** y el compilador **PCWHD** de **CCS** integradas, juntas permiten crear y depurar código para que sea ejecutado en el microcontrolador PIC 18F4550 de la tarjeta principal. Existe una plantilla que se pone a disposición del usuario en la cual se integran librerías y funciones pre-diseñadas con las cuales se puede tener acceso a los recursos del sistema (adquisición de variables, comunicación).

Sistema para programación de la tarjeta principal: en este caso la palabra programación hace referencia a transferir código desarrollado para la tarjeta principal desde el PC y que este sea grabado en la memoria de programa del microcontrolador de la tarjeta. Para realizar la acción de programación se utiliza un bootloader, este es un conjunto software/firmware, el software para manejo desde el PC es la aplicación ***PICDEM FS USB Tool***, ***driver del bootloader*** y ***firmware*** del mismo los cuales se encuentran en el CD de usuario.

Interfaz de usuario: es una aplicación desarrollada en **Matlab** con la herramienta para creación de interfaces de usuario **GUIDE**, tiene como función establecer comunicación a través del bus USB permitiendo el intercambio de información con la tarjeta principal, básicamente se busca que el usuario pueda recibir datos en el computador como el valor de diferentes variables del proceso (variable controlada, variable manipulada, esfuerzo de control) y realizar acciones como graficar dichas variables en el tiempo, almacenar los datos para realizar análisis y cálculos. La aplicación del computador también brinda la posibilidad de controlar el inicio y finalización de un experimento, también permite fijar valores de la variable de referencia o set-point o enviar un valor manual para el esfuerzo de control.

Para operar con la interfaz de usuario se requiere:

- **Librería de comunicación USB (mpusbapi.dll)**, permite enviar y recibir información desde la interfaz de usuario en Matlab.
- **Driver personalizado** diseñado especialmente para que el sistema operativo detecte y reconozca la tarjeta principal.

En los siguientes puntos se muestra el proceso de instalación, configuración, prueba y descripción de los diferentes componentes software requeridos para operar con el ***sistema para implementación de controladores digitales***.

3.1 Instalación de software para programación de la tarjeta principal

La tarjeta principal del sistema requiere de dos paquetes para su programación, el primero es el compilador PCWHD de CCS y el Mplab IDE de Microchip, a continuación se muestra el proceso detallado para la respectiva instalación de cada uno de ellos.

3.1.1 Instalación Mplab IDE de Microchip

Mplab IDE es la interfaz de usuario para programación de la tarjeta principal, en ella se carga la plantilla de programación para que el usuario diseñe sus programas para la tarjeta para las prácticas con los diferentes módulos (plantas experimentales). El instalador de Mplab se pone a disposición del usuario en el CD de usuario dentro de la carpeta ***SW_Programacion_Tarjeta_Principal*** como se muestra en la siguiente figura.

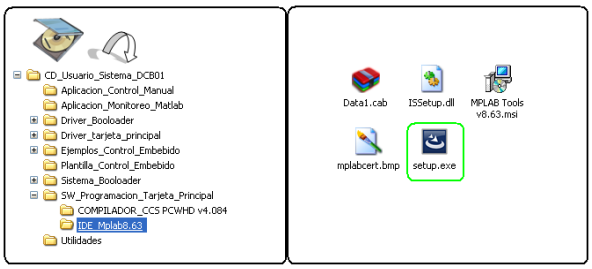


Figura 8

El proceso de instalación es también sencillo al igual que el del compilador, se inicia dando doble clic sobre el icono ***setup.exe*** que se encuentra dentro de la carpeta del programa.

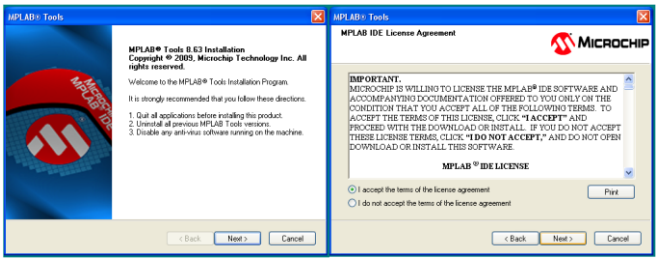


Figura 9

En el tercer paso de instalación (***Setup Type***), se recomienda configurar la instalación de modo personalizado, para ello seleccionar la opción ***Custom***.

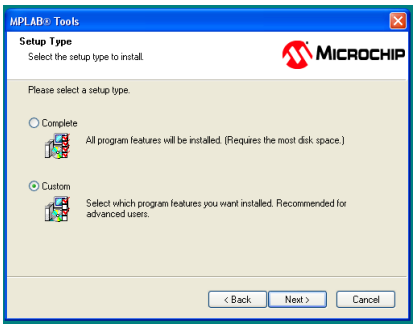


Figura 10

Al seleccionar el modo personalizado se pretende instalar solamente lo necesario para poder programar la tarjeta principal, al seleccionar este modo en la siguiente ventana aparece otra con un recuadro donde se pueden tildar o no algunas opciones, en la siguiente imagen se muestra dicha ventana al lado izquierdo, al lado derecho se muestra cómo deben quedar marcadas las opciones que se presentan.

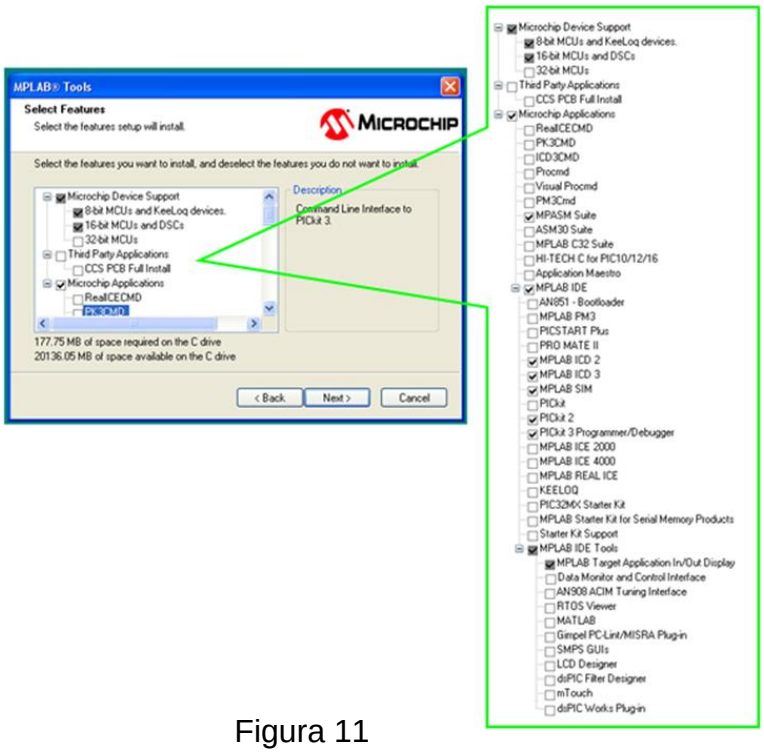


Figura 11

Luego de seleccionar las opciones personalizadas de instalación se procede con la instalación, el siguiente paso es elegir un directorio dentro del disco duro para la instalación de Mplab y sus herramientas, el directorio por defecto es: ***c:\Archivos de Programa\Microchip***, se deja a elección del usuario si cambia la ruta de instalación.

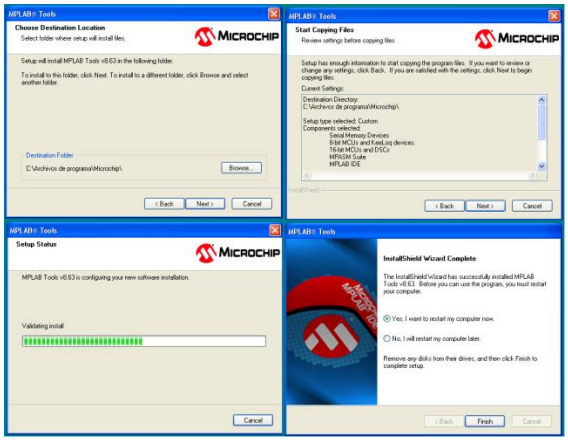


Figura 12

Al finalizar la instalación, se debe reiniciar el computador, con esto ya queda instalado el Mplab IDE de Microchip para programación de la tarjeta principal.

3.1.2 Instalación compilador PCWHD de CCS inc.

El instalador del compilador se encuentra en el CD de usuario dentro de la carpeta ***SW_programacion_Tarjeta_Principal*** como se muestra en la siguiente imagen:

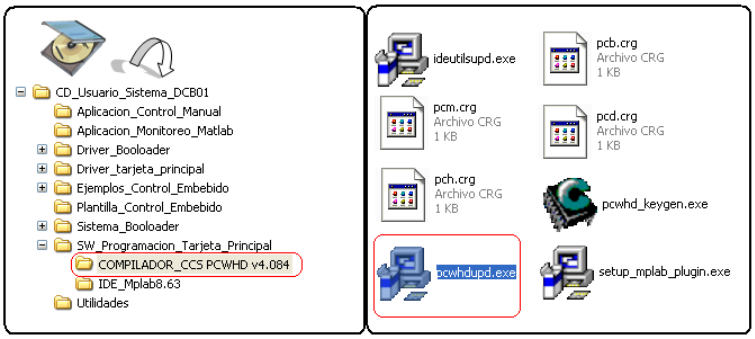


Figura 13

El instalador es ***pcwhdupd.exe*** al dar doble clic sobre el icono iniciara el proceso de instalación:

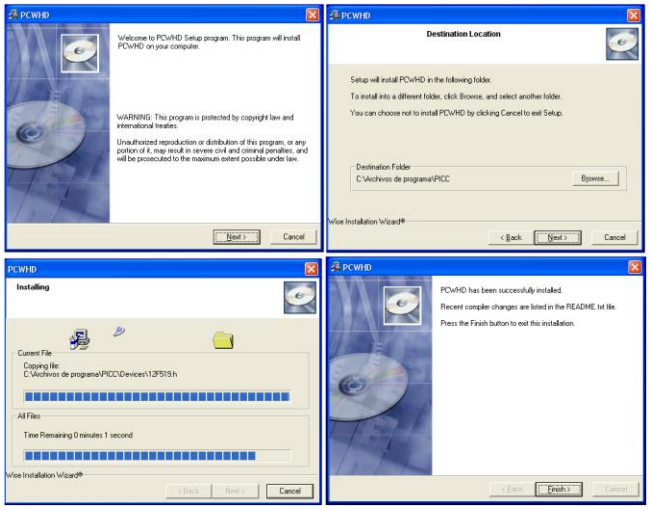


Figura 14

El proceso de instalación es sencillo solo dar clic en el botón next, escoger un directorio de instalación, ***c:\Archivos de programa\ PICC*** es el directorio por defecto, se puede cambiar si el usuario lo desea.

En la parte final de la instalación si se solicita algún archivo de licencia, en la carpeta del compilador ***COMPILADOR_CCS PCWHD v4.084*** están las licencias del compilador, estas tienen extensión ****.crg***, elegir cualquiera de ellas.

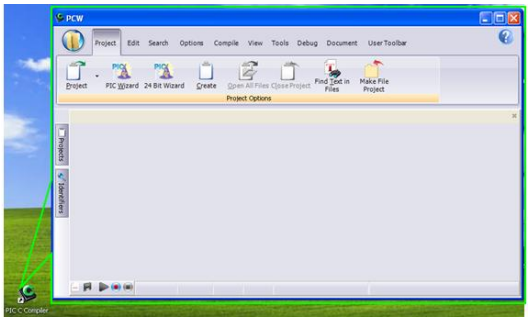


Figura 15

3.1.3 Integración compilador PCWHD y Mplab

Luego de la instalación del compilador **PCWHD** y **Mplab** se debe realizar un último paso y es integrar estos dos programas, el objetivo de este paso es que el usuario pueda crear y compilar código en C con el compilador de CCS a través de Mplab, además aprovechar las ventajas de este último software. Para realizar este proceso existe un ***plugin*** que permite enlazar el compilador PCWHD con el entorno de Mplab, el ***plugin*** se encuentra en la carpeta del compilador en el CD de usuario, como se muestra en la siguiente imagen.

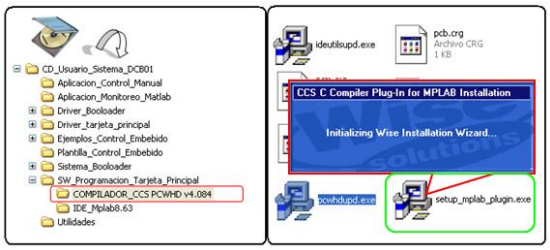


Figura 16

La instalación de este plugin se inicia dando doble clic sobre el icono que se resalta en la imagen anterior, **se debe tener en cuenta que el directorio de instalación del este plugin corresponda con la ruta de instalación del compilador**.

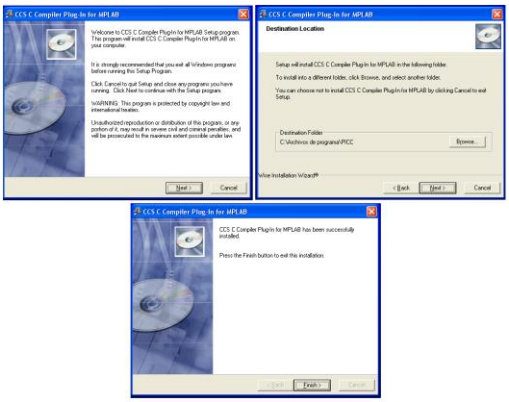


Figura 17

Luego de instalar el plugin, se debe configurar en Mplab la ubicación del compilador PCWHD de CCS, para ello se debe abrir Mplab y en la opción ***Project*** al final de la lista ubicar la opción ***Set Language Tool Locations***.

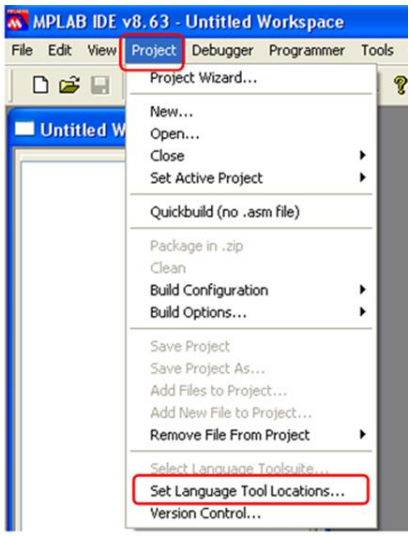


Figura 18

Al seleccionar la opción anterior, se abre una ventana con una lista de varias herramientas de programación que se pueden utilizar con Mplab, de la lista se debe seleccionar la opción ***CCS C compiler for PIC10/12/14/16/18/24/dsPIC30/dsPIC33*** y seguir los siguientes pasos:

Paso 1: ubicar y marcar la opción CCS C Compiles (ccs.exe)

Paso 2: ubicar el botón ***Browser***, al dar clic sobre él aparece una ventana de exploración que permite ir a la ruta donde está instalado el compilador PCWHD y seleccionar su ejecutable, generalmente por defecto la dirección es ***c:\Archivos de programa\ PICC\Ccsc.exe***.

Paso 3 y 4: seleccionar el ejecutable del compilador llamado ***Ccsc.exe***, dar clic en el botón **Abrir.**

Paso 5: Por ultimo verificar que en la casilla llamada ***Location*** en la ventana Set **Language Tool Locations** contenga la ruta del ejecutable del compilador, si es correcto oprimir el botón ***OK***.

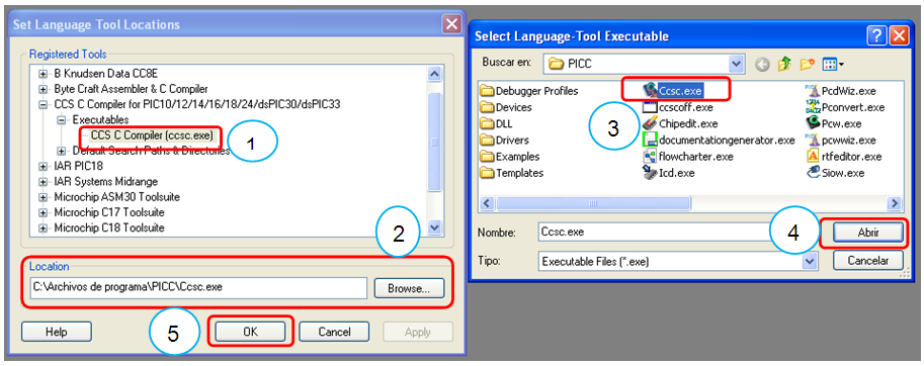


Figura 19

Para verificar que el proceso de instalación e integración de las dos herramientas (compilador PCWHD y Mplab) se han realizado correctamente se debe realizar una prueba con un último procedimiento, para ello utilizara el proyecto de la plantilla de programación ubicado en la carpeta ***Plantilla_Control_Embebido***, que se encuentra dentro del CD de usuario, se debe copiar la carpeta al PC y luego abrir el proyecto dando doble clic sobre el archivo llamado ***usb_board.mcp***.

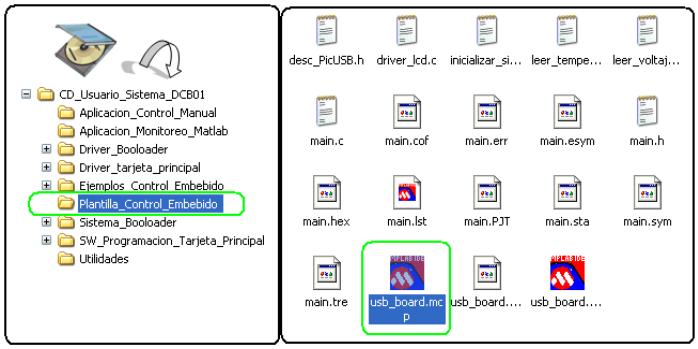


Figura 20

Una vez abierto ir a la opción ***Project*** y ubicar al final de la lista la opción ***Select Language Toolsuite***.

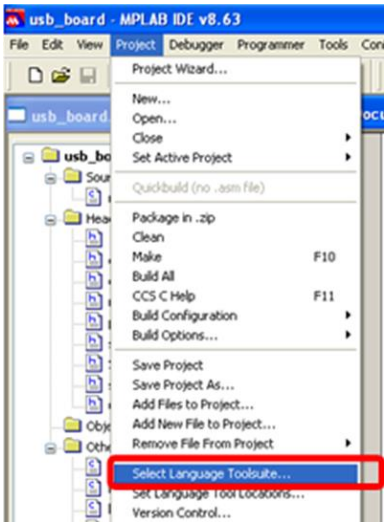


Figura 21

Al seleccionar la opción anterior, se abre una ventana donde se debe configurar la dirección de instalación del compilador PCWHD donde debe estar el ejecutable del compilador, se debe seguir los siguientes pasos:

Paso 1: verificar que en la casilla ***Active Toolsuite*** este seleccionada la opción ***CCS C compiler for PIC10/12/14/16/18/24/dsPIC30/dsPIC33***.
Paso 2: ubicar el botón ***Browser***, al dar clic sobre el aparece una ventana de exploración que permite ir a la ruta donde está instalado el compilador PCWHD y seleccionar su ejecutable, generalmente por defecto la dirección es ***c:\Archivos de programa\ PICC\Ccsc.exe*.**

Paso 3 ,4 y 5: seleccionar el ejecutable del compilador llamado ***Ccsc.exe***, dar clic en el botón **Abrir.**

Paso 6: Por ultimo verificar que en la barra llamada ***Location*** en la ventana Set **Select Language Toolsuite** contenga la ruta del ejecutable del compilador, si es correcto oprimir el botón ***OK***.

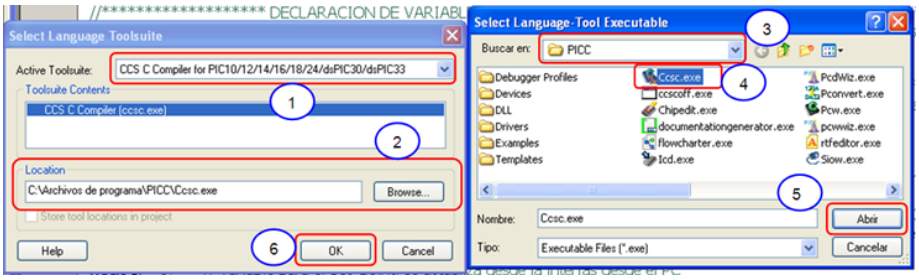


Figura 22

Si todos los pasos anteriores se han realizado con éxito ahora se prueba la correcta integración de los dos programas, para ello se realiza el proceso de compilación, se puede hacer con la tecla ***F10*** o mediante el icono que aparece en la siguiente imagen marcado con el número ***1***, el código debe ser compilado se puede ver en la ventana ***output*** el estado de la compilación, en la imagen se muestra con el número **2** que la compilación ha sido exitosa (***BUILD SUCCEEDED***).

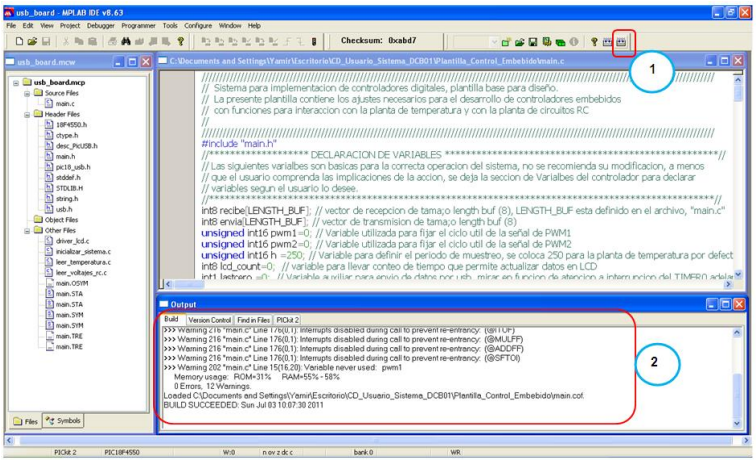


Figura 23

3.2 Instalación de drivers para tarjeta principal para Windows XP

La tarjeta principal del sistema requiere la instalación de dos drivers para que el computador pueda reconocerla y operar con ella en los diferentes modos (modo bootloader y modo desarrollo), el primero es necesario para utilizar el sistema de bootloader y el segundo es un driver personalizado para operar normalmente en modo desarrollo, de modo que la tarjeta ejecute los códigos de usuario y que el software desarrollado en Matlab pueda reconocer la tarjeta.

Antes de iniciar se aclara que los procedimientos descritos a continuación tanto para el driver del bootloader como para el driver personalizado de la tarjeta principal están orientados al sistema operativo **Windows XP**, además se aclara que los puertos USB del computador deben soportar la especificación **USB 2.0**

que es la que maneja el modulo USB del microcontrolador PIC 18F4550 de la tarjeta principal.

3.2.1 Instalación driver para el Bootloader

Para la instalación del driver para el sistema bootloader es necesario que el firmware del bootloader este previamente grabado en el microcontrolador (ver sección 7 de este manual de usuario), luego realizar el siguiente procedimiento:

Conectar la tarjeta principal al PC mediante su respectivo cable USB, y hacer que la tarjeta ingrese en modo bootloader, para ingresar en modo bootloader se hace manualmente mediante los pulsadores **BOOT** y **RESET** en la tarjeta principal, para ello se debe mantener presionado el botón **BOOT**, posteriormente presionar y soltar el botón **RESET**, por último soltar el botón **BOOT**, es importante recordar que “el botón **BOOT** es el primero que se presiona y el ultimo que se suelta”, si esta operación se realiza adecuadamente los dos diodos LED verdes en la tarjeta principal empezaran a parpadear ambos al mismo tiempo de forma consecutiva.

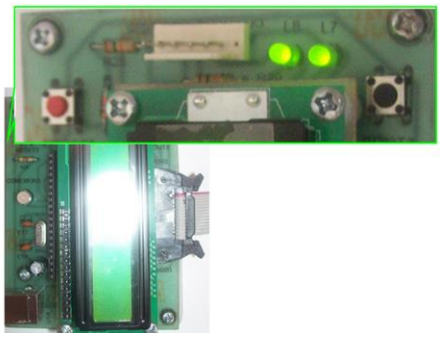


Figura 24

En el computador aparecerá un mensaje informando que se ha detectado un nuevo dispositivo USB.

Figura 25

Posterior al mensaje aparecerá la ventana del asistente para agregar nuevo hardware de Windows XP, en esta ventana hay tres opciones, las dos primeras hacen referencia a utilizar la opción automática de Windows para que instale el driver apropiado para el nuevo dispositivo desde internet, la última da la opción de continuar con el proceso de forma local, seleccionar la última opción “***No por el momento***” y dar clic en siguiente.

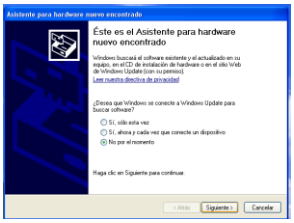


Figura 26

En seguida aparece una ventana con dos opciones, la primera es para instalar el driver automáticamente, esta opción busca un driver apropiado en los drivers que tiene Windows incorporados, la segunda opción permite una instalación manual del driver, **seleccionar la segunda opción** y dar clic en siguiente.

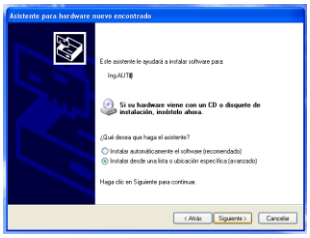


Figura 27

En seguida aparece una nueva ventana con dos opciones principales, la primera permite buscar manualmente el controlador y la ruta, la segunda es una opción más avanzada que permite instalar un driver de forma específica conociendo cual es el archivo del driver y su nombre, se debe **seleccionar la primera opción**.

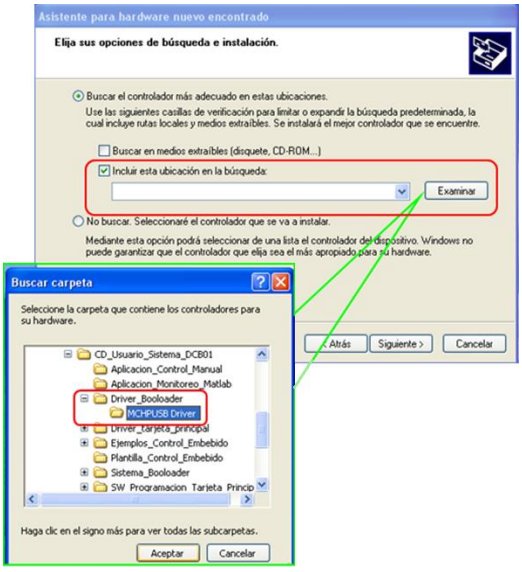


Figura 28

Al seleccionar la opción "***Buscar el controlador más adecuado en estas ubicaciones***" aparece una ventana que permite buscar la carpeta contenedora del driver, en este caso la carpeta con el driver se encuentra en el CD de usuario, la cual tiene por nombre ***Driver Bootloader***, dentro de ella se debe seleccionar la

carpeta ***MCHPUSB Driver*** y dar clic en aceptar, tal como se muestra en la imagen anterior.

Al presionar en el botón siguiente de la ventana del asistente se inicia el proceso de instalación del driver como se muestra en la siguiente imagen.

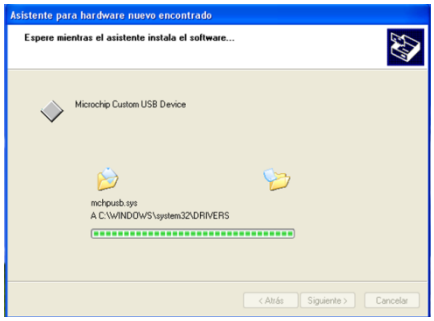


Figura 29

En algunos casos (según la versión de Windows XP instalado) el asistente podrá arrojar el siguiente mensaje, si es el caso dar clic en la opción continuar.

Figura 30

Otra situación que se puede dar es que la instalación pida la ruta y el nombre especifico del driver que en este caso es el ***mchpusb.sys*** y se encuentra en la carpeta del driver del bootloader en el CD de usuario según se muestra en la siguiente imagen:

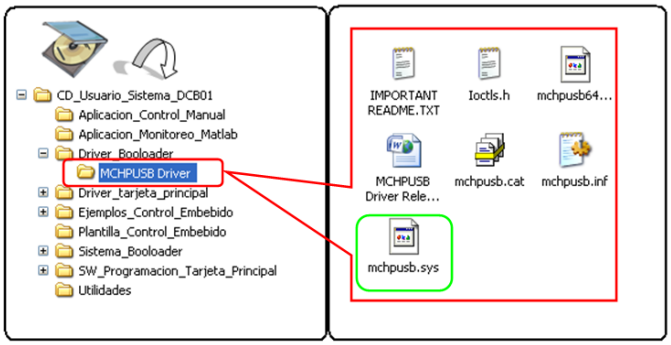


Figura 31

La ventana que solicita el driver ***mchpusb.sys*** es la siguiente:

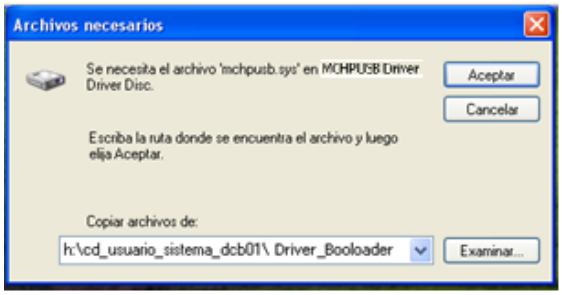


Figura 32

Si es el caso y aparece se debe ir a la carpeta ***MCHPUSB Driver*** y seleccionar el archivo ***mchpusb.sys*** mediante el botón ***Examinar*** y oprimir ***Aceptar***, como se muestra en la siguiente imagen:

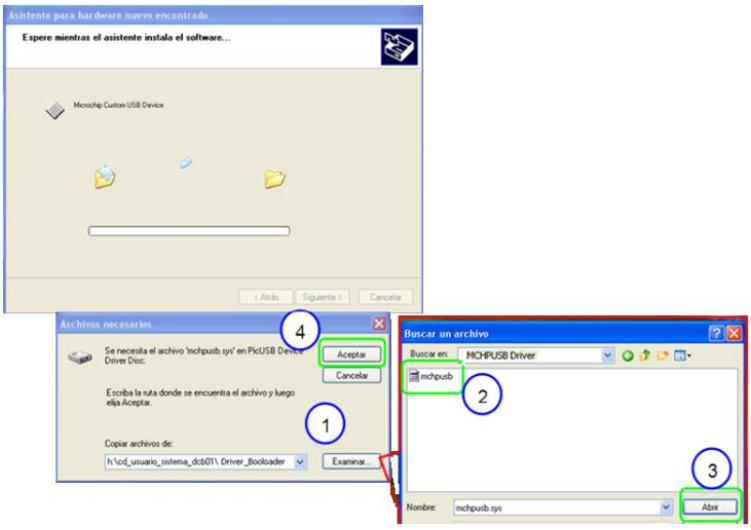


Figura 33

Al final de la instalación debe aparecer una ventana del asistente con el mensaje informando que ya termino el proceso de instalación

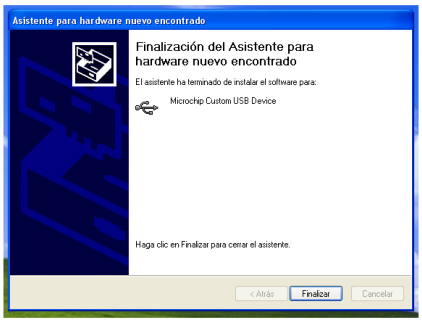


Figura 34

Y por último aparece el siguiente mensaje:

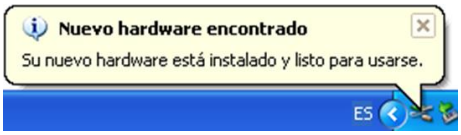


Figura 35

Al finalizar la instalación en la tarjeta principal los dos LED deben parpadear pero ya en forma intercalada uno luego del otro de forma consecutiva.

Nota: algunas veces se requiere realizar el mismo proceso por cada puerto USB diferente al que se conecte la tarjeta, si se utiliza el mismo puerto el sistema siempre debe reconocer automáticamente la tarjeta cuando se ingrese en modo bootloader.

Se puede verificar en el panel de control con la tarjeta conectada al PC en modo bootloader la correcta instalación del dispositivo, debe aparecer dentro de la opción ***Custon USB Devices*** el dispositivo nombrado como ***Microchip USB Custom USB Device***, en la siguiente imagen se muestra en detalle.

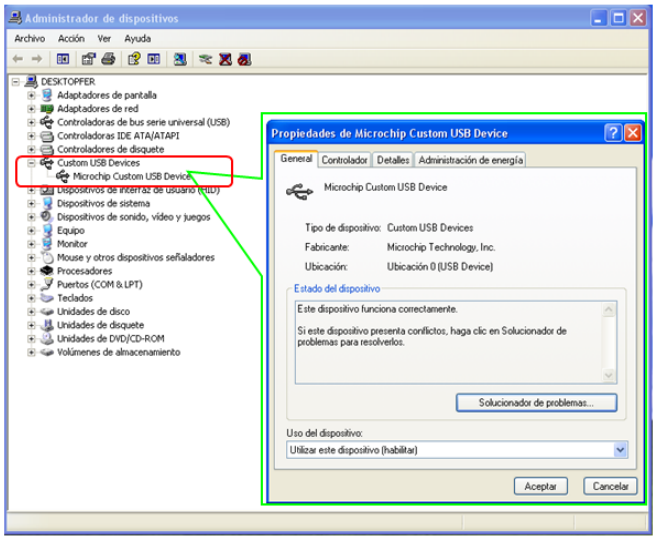


Figura 36

3.2.2 Instalación driver personalizado de la tarjeta principal

El proceso de instalación del driver personalizado para la tarjeta principal es similar al anteriormente descrito para el modo bootloader, pero antes de iniciar con el proceso de instalación es necesario que el microcontrolador tenga grabado un firmware, puede ser la ***Plantilla_Control_Embebido*** (ver sección 7 de este manual de usuario) que trae el CD de usuario, esto es muy importante ya que en el código esta la configuración e implementación para establecer una conexión USB.

Conectar la tarjeta principal al PC mediante su respectivo cable USB, en este caso el computador detecta el dispositivo, se debe notar que ahora el nombre ya está personalizado.

Figura 37

En la tarjeta principal se enciente el LED de ***CONEXIÓN*** en color **rojo**.

Figura 38

Posterior al mensaje aparecerá la ventana del asistente para agregar nuevo hardware de Windows XP, en esta ventana hay tres opciones, las dos primeras hacen referencia a utilizar la opción automática de Windows para que instale el driver apropiado para el nuevo dispositivo desde internet, la última da la opción de continuar con el proceso de forma local, seleccionar la última opción "***No por el momento***" y dar clic en siguiente.

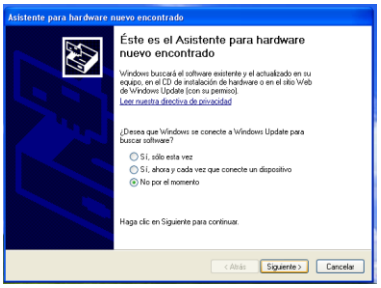


Figura 39

En seguida aparece una ventana con dos opciones, la primera es para instalar el driver automáticamente, esta opción busca un driver apropiado buscando en los

drivers que tiene Windows incorporados, la segunda opción permite una instalación manual del driver, **seleccionar la segunda opción** y dar clic en siguiente.

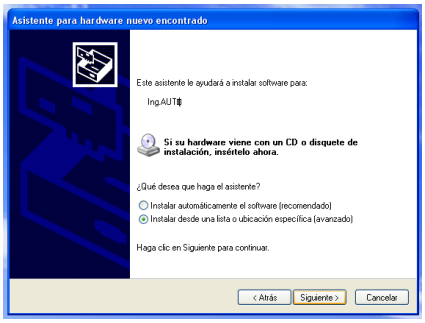


Figura 40

En seguida aparece una nueva ventana con dos opciones principales, la primera permite buscar manualmente el controlador y la ruta, la segunda es una opción más avanzada que permite instalar un driver de forma específica conociendo cual es el archivo del driver y su nombre, se debe seleccionar la primera opción.

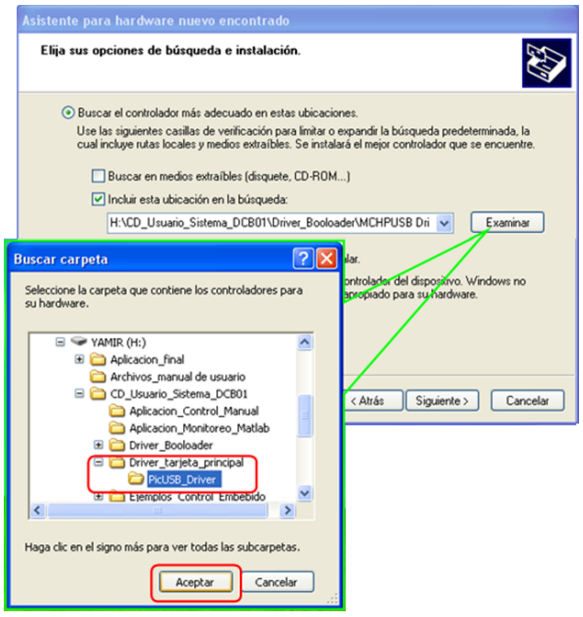


Figura 41

Al seleccionar la opción "***Buscar el controlador más adecuado en estas ubicaciones***" aparece una ventana que permite buscar la carpeta contenedora del driver, en este caso la carpeta con el driver se encuentra en el CD de usuario, la cual tiene por nombre ***Driver_tarjeta_principal***, dentro de ella se debe seleccionar la carpeta ***PicUSB_Driver*** y dar clic en aceptar, tal como se muestra en la imagen anterior.

Al presionar en el botón siguiente de la ventana del asistente se inicia el proceso de instalación del driver como se muestra en la siguiente imagen

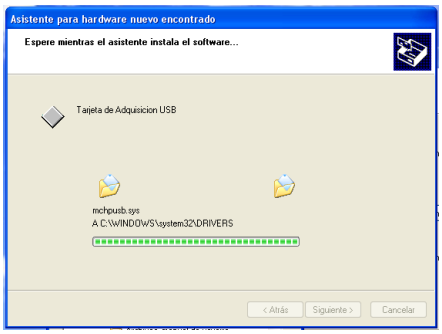


Figura 42

En algunos casos (según la versión de Windows XP instalado) el asistente podrá arrojar el siguiente mensaje, si es el caso dar clic en la opción continuar.

Figura 43

Otra situación que se puede dar es que la instalación pida la ruta y el nombre especifico del driver que en este caso es el ***mchpusb.sys*** y se encuentra en la carpeta del driver del bootloader en el CD de usuario según se muestra en la siguiente imagen:

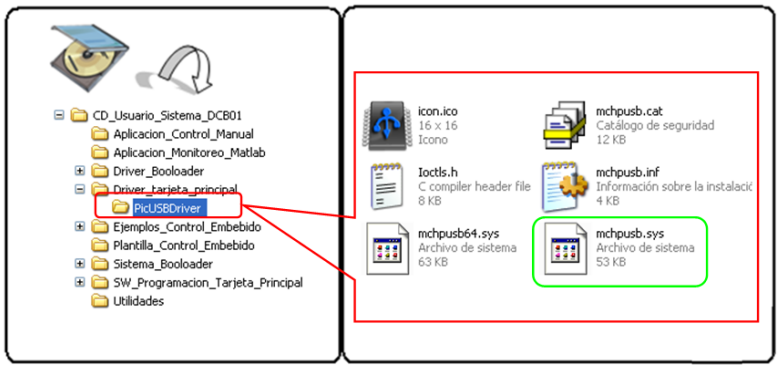


Figura 44

La ventana que solicita el driver *mchpusb.sys* es la siguiente:

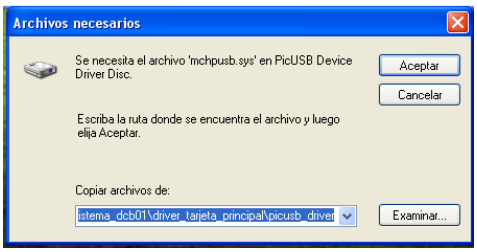


Figura 45

Si es el caso y aparece se debe ir a la carpeta ***Driver_tarjeta_principal*** y seleccionar el archivo ***mchpusb.sys*** mediante el botón ***Examinar*** y oprimir ***Aceptar***, como se muestra en la siguiente imagen:

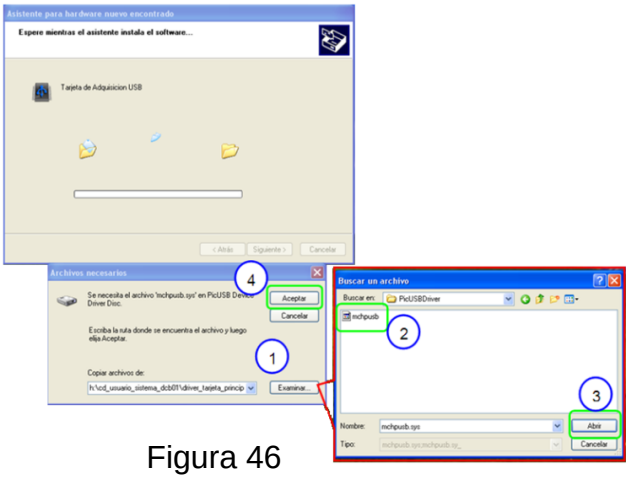


Figura 46

Al final de la instalación debe aparecer una ventana del asistente con el mensaje informando que ya termino el proceso de instalación

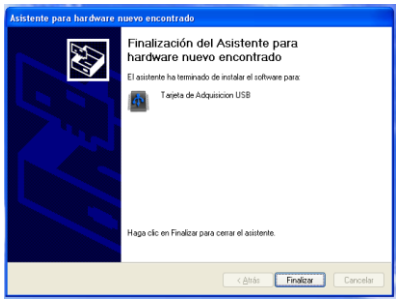


Figura 47

Y por último aparece el siguiente mensaje:

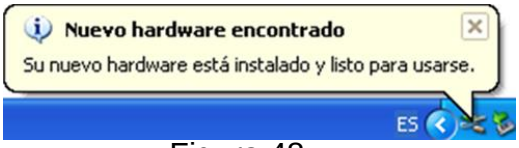


Figura 48

Al finalizar la instalación en la tarjeta el diodo LED de conexión debe cambiar de color rojo a verde esto indica que la tarjeta ha sido detectada y reconocida por el sistema operativo.

Figura 49

Nota: algunas veces se requiere realizar el mismo proceso por cada puerto USB diferente al que se conecte la tarjeta, si se utiliza el mismo puerto el sistema siempre debe reconocer automáticamente la tarjeta cuando.

Se puede verificar en el panel de control con la tarjeta conectada al PC en modo bootloader la correcta instalación del dispositivo, debe aparecer dentro de la opción ***Tarjeta de Adquisición USB*** el dispositivo nombrado como ***Tarjeta de Adquisición USB***, en la siguiente imagen se muestra en detalle.

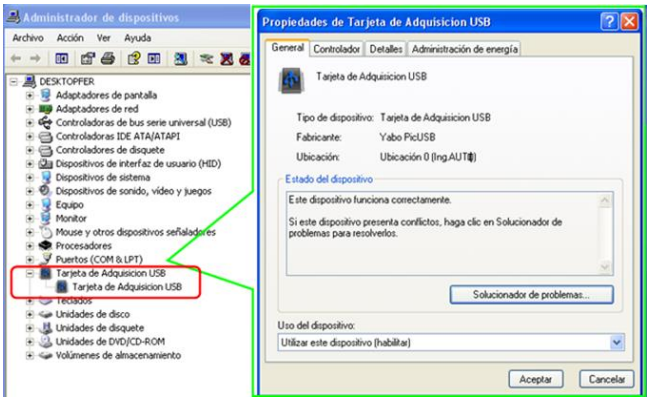


Figura 50

3.3 Modo de uso de plantilla para desarrollo de programas para la tarjeta principal.

La tarjeta principal tiene posibilidad de ser programada por el usuario para desarrollar diferentes actividades según la práctica a implementar. El medio de programación es por medio de lenguaje C, atreves de la integración de las herramientas Mplab y el compilador PCWHD de CCS, mediante estas herramientas se desarrolla código para que sea ejecutado en el microcontrolador PIC 18F4550.

Las posibilidades en la programación en C son bastante amplias y en usuarios con poco manejo y bajo conocimiento de la arquitectura del microcontrolador resultaría complejo el diseño de programas para la tarjeta principal del sistema, pensando en esta circunstancia, se pone a disposición del usuario una plantilla en la cual previamente se han realizado las principales configuraciones del microcontrolador PIC 18F4550 para que opere en óptimas condiciones según las características del sistema, por otra parte en dicha plantilla también se ha integrado librerías o bibliotecas con funciones específicas que permiten al usuario realizar fácilmente por medio de una línea de código realizar acciones como:

- Adquisición de la variable controlada en cualquiera de las plantas (temperatura o circuitos RC).
- Adquisición de la variable manipulada (solo disponible en la planta de circuitos RC).
- Adquisición de variables intermedias en la planta de circuitos RC.
- Generar la señal del esfuerzo de control.
- Enviar información por el bus USB.
- Recibir información por el bus USB.
- Imprimir mensajes y el valor de diferentes variables en la pantalla LCD de forma local.

El uso de la plantilla de programación permite al usuario hacer uso de todos los recursos del sistema y para ello solo se requiere nociones básicas de programación en lenguaje C.

La plantilla en mención se encuentra en el CD de usuario y tiene por nombre ***Plantilla_Control_Embebido***, está conformada por un proyecto (***usb_board.mcp***) que se abre en **Mplab** y contiene varios archivos de los cuales se explicara únicamente su función en forma general ya que el usuario no tendrá que modificarlos, solamente utilizara el principal de ellos para crear su propio código. La siguiente imagen muestra la ubicación de la plantilla.

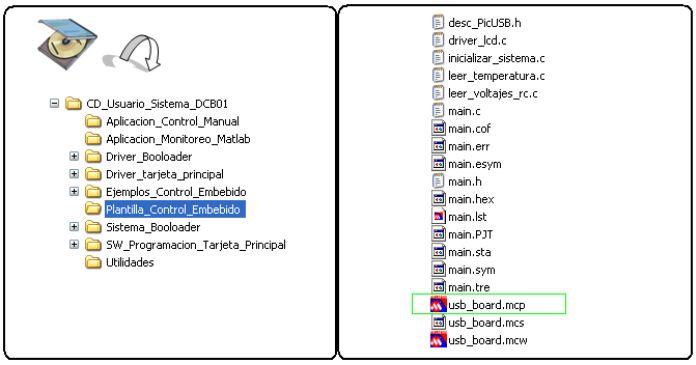


Figura 51

A continuación se hará una descripción de la estructura de la plantilla y como el usuario puede hacer uso de ella.

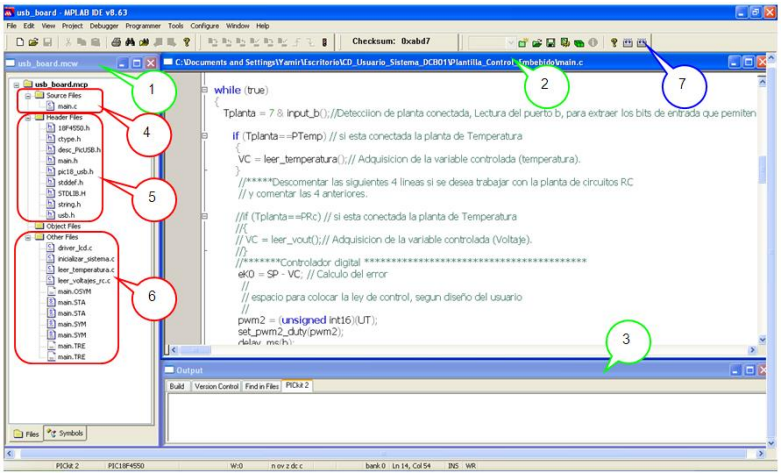


Figura 52

Las partes indicadas en la anterior figura son:

1. **Ventana del proyecto:** Muestra un árbol con los diferentes archivos del proyecto clasificados en tres grupos (*Source Files, Header Files y Other Files*), desde esta ventana se puede abrir los diferentes archivos que componen el proyecto simplemente dando doble clic sobre encima del archivo, también se pueden agregar nuevos archivos dando clic derecho sobre la capeta de grupo y seleccionar la opción *Add Files*, también se puede remover los archivos presentes en el árbol dando clic derecho sobre el archivo y seleccionar la opción *Remove*.
2. **Ventana de trabajo:** En esta sección se abre el archivo activo del proyecto, en la imagen anterior se muestra el contenido del archivo *main.c*, este es el archivo donde el usuario debe crear el código para realizar sus prácticas.
3. **Ventana de salida de mensajes:** esta es una ventana auxiliar por medio de la cual el software se comunica con el usuario atreves de diferentes mensajes, por ejemplo el resultado de la compilación de código, mensajes de error y una descripción de los mismos, estado de conexión de dispositivos como programadores externos entre otros.
4. **Sección source files:** en esta sección se encuentra el archivo *main.c*, que es el archivo principal del proyecto, este se encarga de llamara a los otros archivos según sea necesario y es donde el usuario debe crear su propio código.
5. **Sección header files:** sección donde se anexa al proyecto archivos de cabecera con extensión .h, dentro de ellos se encuentran archivos que realizan configuraciones del microcontrolador, archivos con funciones que son propios del compilador y son cargados desde la carpeta de instalación del mismo.
6. **Sección other files:** es una sección donde se anexan archivos creados por el usuario, estos son de extensión .c y dentro de ellos se encuentran las

librerías con las funciones creadas especialmente para manejar los módulos (plantas y periféricos) del sistema.

7. **Botón de compilar:** Mediante este botón se puede compilar el proyecto, el proceso de compilación permite generar el archivo con el código que se graba en el microcontrolador de la tarjeta principal.

Modo de uso de la plantilla

La plantilla está diseñada para que el usuario utilice la menor cantidad de código posible y además que pueda crear programas para la tarjeta principal teniendo nociones básicas de programación.

NOTA: Antes de iniciar a experimentar con el desarrollo de código utilizando la plantilla asegúrese de tener instalado en su computador las siguientes herramientas:

- Mplab IDE (ver sección **3.1.1**)
- Compilador PCWHD CCD Integrado con Mplab IDE, (ver secciones **3.1.2** y **3.1.3**)
- Driver de bootloader y driver personalizado para la tarjeta principal,
- Aplicación PIDCEM FS USB Tool (ver sección **3.5**)

Para hacer uso de la plantilla se recomienda seguir los siguientes pasos:

Paso 1: realizar una copia de la plantilla en el disco duro del PC.

Realizar una copia de la carpeta ***"plantilla_control_embebido"*** que contiene los archivos que componen la plantilla al computador en un lugar de preferencia para el usuario, recordar que la carpeta se encuentra en el CD de usuario, se recomienda renombrar la copia con un nombre acorde con la función del programa que se desea desarrollar, por ejemplo "***control_PIC_temperatura***".

Paso 2: verificar la correcta compilación del código.

Abrir el proyecto en Mplab, abrir el archivo que tiene por nombre "***usb_board.mcp***", para verificar que el código base compila sin errores presionar el botón de compilar (*Make project*) o también a través de la tecla *F10*, mirar en la ventana de salida que aparezca el mensaje "*BUILD SUCEEDEED*".

Paso 3: elaborar el código en lenguaje c:

Abrir el archivo *main.c*, dentro ubicar el ciclo continuo ***while(true)***, en este archivo se encuentran predefinidas algunas variables y definiciones, al igual que se ha señalado espacios donde el usuario debe realizar acciones como declarar e inicializar variables y un espacio para digitar el código, en la siguiente imagen se muestra en detalle las partes principales de este archivo.

3.4 Firmware para control manual de las plantas de Temperatura y Circuitos RC.

Una de las opciones que brinda el sistema es la posibilidad de hacer control manual sobre las plantas y almacenar información del experimento, esta opción es útil para analizar y modelar el comportamiento de los sistemas.

Para realizar control manual de las plantas del sistema, se ha desarrollado un sistema mediante el cual el usuario puede ingresar manualmente un valor en porcentaje de la señal pwm para el esfuerzo de control a aplicar a cualquiera de las plantas (temperatura o circuitos RC) , este sistema está conformado por una aplicación para el computador que se ejecuta en matlab y se describe en la sección 3.6 y un código para la tarjeta principal diseñado especialmente para recibir el valor del control manual ingresado desde el PC y transferirlo hacia las plantas, de igual manera hace lectura de la variable controlada y la envía al PC, este código se encuentra en la carpeta *Control_Manual_Desde_PC_Temp_RC* en la capeta de ejemplos que trae el CD de usuario.

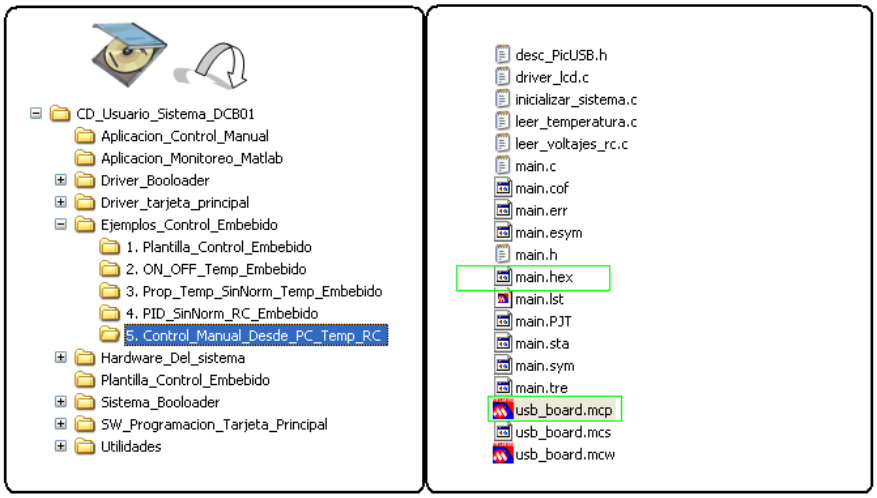


Figura 53

Para su uso solo se debe grabar el archivo main.hex en la tarjeta principal por medio del bootloader (ver sección 3.5 de este manual).

3.5 Modo de uso de Bootloader para programación de la tarjeta principal.

La tarjeta principal del sistema para implementación de controladores digitales, está equipada con un Bootloader USB, este es un programa que se aloja en la parte inicial de la memoria de programa (flash) en el microcontrolador, el cual permite al usuario transferir el firmware (código para la tarjeta principal) de forma directa usando solamente el cable de conexión USB, en esta sección se describe el procedimiento para grabar un firmware en la tarjeta principal a través del sistema Bootloader.

Para operar con el bootloader es necesario lo siguiente:

- Tener la tarjeta principal con el firmware del Bootloader grabado en su microcontrolador PIC 18F4550, las tarjeta se entrega con el bootloader previamente grabado en el microcontrolador y listo para ser utilizado, en caso de no ser así o si por algún motivo la tarjeta se desprograma, el código se encuentra en el CD de usuario, en la siguiente dirección:

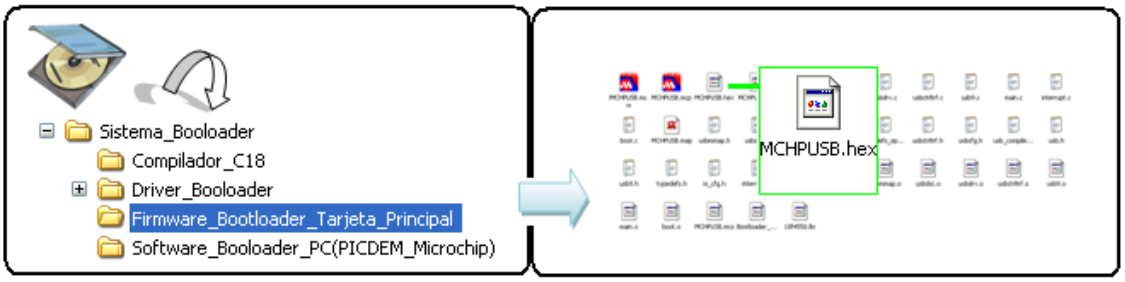


Figura 54

Según se muestra en la imagen dentro del directorio del CD de usuario hay una carpeta llamada *Sistema_Bootloader*, en ella se encuentra la carpeta *Firmware_Bootloader_Tarjeta_principal*, dentro de esta carpeta está el proyecto con el código del firmware del Bootloader y el archivo ***MCHPUSB.hex*** el cual es el código del bootloader y es el archivo que se debe grabar en el microcontrolador, si es necesario se debe hacer uso de un programador externo (Picstar plus, ICD2, ICD3 o algún programador genérico) con soporte para el microcontrolador PIC 18F4550, con este borrar toda la memoria del microcontrolador y luego grabar el archivo ***MCHPUSB.hex***.

- Tener instalado el driver del Bootloader (ver sección 3.2.1Instalación driver para el Bootloader de este manual de usuario)
- Software para comunicación con el Bootloader USB desde el PC, es necesario contar con un aplicativo en el PC que permita al usuario seleccionar el firmware que se desea transferir al microcontrolador de la tarjeta principal a través del Bootloader, se ha elegido la aplicación *PICDEM FS USB tool* (***PDFSUSB.exe***) la cual es diseñada especialmente para trabajar con el bootloader USB de Microchip, la aplicación se encuentra en el CD de usuario en la carpeta *Software_Booloader_PC(PICDEM_Microchip)* dentro de la carpeta *Sistema_Bootloader* , se recomienda a los usuarios copiar la carpeta de la aplicación junto con su contenido en el disco duro del PC y crear un acceso directo al escritorio para su fácil acceso.

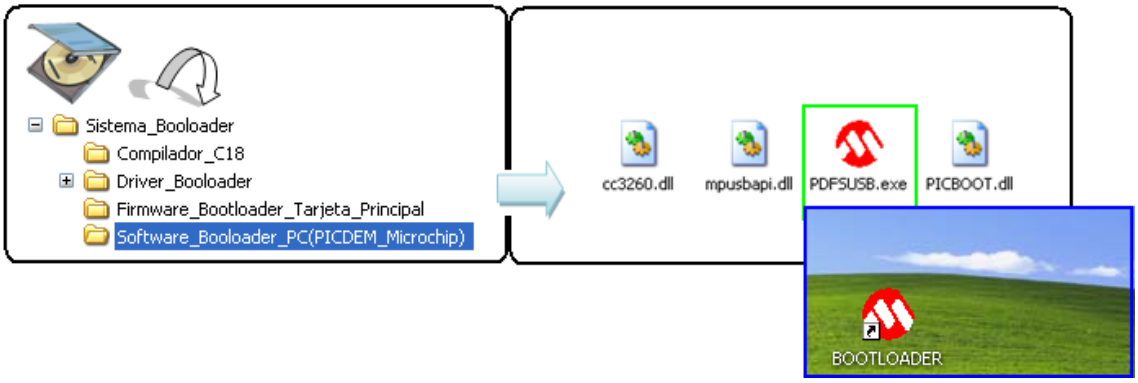


Figura 55

Antes de iniciar con una demostración del modo de funcionamiento del sistema bootloader, es importante comentar sobre la aplicación *PICDEM FS USB tool*, La siguiente figura muestra la ventaja de la aplicación así como sus partes y funcionalidades de la misma.

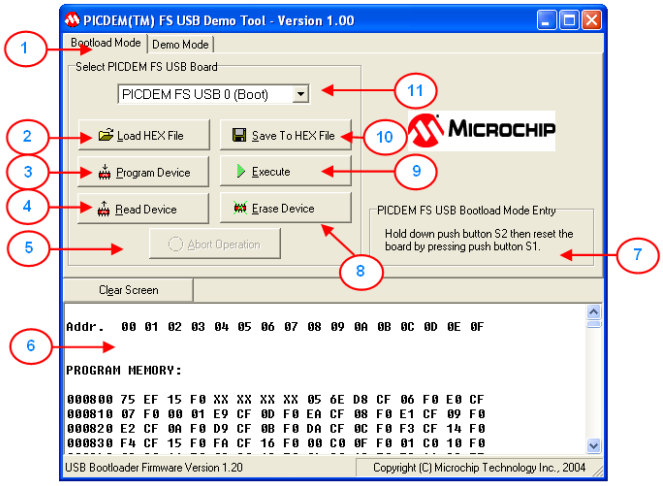


Figura 56

La aplicación es un ejecutable, se abre directamente al dar doble clic sobre el icono ***PDFSUSB.exe*** 
La interfaz que presenta esta aplicación está diseñada para que el usuario se familiarice rápidamente con ella, las partes enumeradas en la figura anterior se describen a continuación:

Tabla 5 partes de la Interfaz PICDEM FS USB Tool

ITEM	Descripción
1	Pestañas de **selección del modo** para la aplicación, hay dos Bootload mode y Demo mode, para utilizar el sistema bootloader siempre se debe seleccionar la primera pestaña.
2	Botón que permite explorar en **busca** de un firmware para grabar en el microcontrolador de la tarjeta principal (los códigos tienen

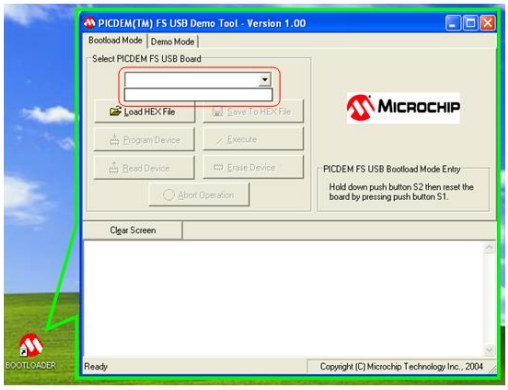
	extensión **.hex*).
3	Botón de **programación**, permite programar el dispositivo microcontrolador, este botón se habilita solamente cuando la tarjeta ha ingresado en modo bootloader.
4	Botón de **lectura**, permite leer la memoria de programa (flash) del microcontrolador, este botón se habilita solamente cuando la tarjeta ha ingresado en modo bootloader.
5	Botón **abortar operación**, detiene un proceso en curso, puede ser una lectura o escritura del microcontrolador, este botón se habilita solamente cuando hay un proceso activo.
6	**Ventana de mensajes**, en esta ventana se despliega información según como mensajes de estado de programación, contenido de la memoria de programa del microcontrolador, entre otros mensajes.
7	**Mensaje para operar en modo bootloader**, el mensaje indica que se debe mantener presionado el botón s2, luego presionar el botón s1, los botones **s1** y **s2** son nombrados como *reset* y *boot* respectivamente en la tarjeta principal.
8	Botón de **borrar**, permite borrar el contenido de la memoria de programa (flash) del microcontrolador, este botón se habilita solamente cuando la tarjeta ha ingresado en modo bootloader.
9	Botón de **ejecución**, este botón provoca un reset en el microcontrolador, sirve para que el microcontrolador salga del modo bootloader y ejecute el código de usuario.
10	Botón de **guardar**, permite guardar el código leído de la memoria de programa (flash) del microcontrolador y almacenarlo en el computador como un archivo **.hex*, este botón se habilita solamente cuando la tarjeta ha ingresado en modo bootloader.
11	**Pestaña de selección** de la tarjeta en modo bootloader, se habilita una vez la tarjeta haya ingresado en modo bootloader realizando el proceso mencionado en el ítem 7 de esta tabla.

Teniendo en cuenta los anteriores aspectos ahora se procede a detallar los pasos para hacer uso del sistema bootloader para grabar un programa o código en la tarjeta principal, se muestra el proceso de modo de uso paso a paso mediante un ejemplo.

Requerimiento: se desea grabar mediante el bootloader el código del controlador ON/OFF para la planta de temperatura que trae el CD de usuario.

Paso 1: conectar la tarjeta principal al PC por medio de su respectivo cable USB.

Paso 2: Abrir la aplicación **PICDEM FS USB tool**



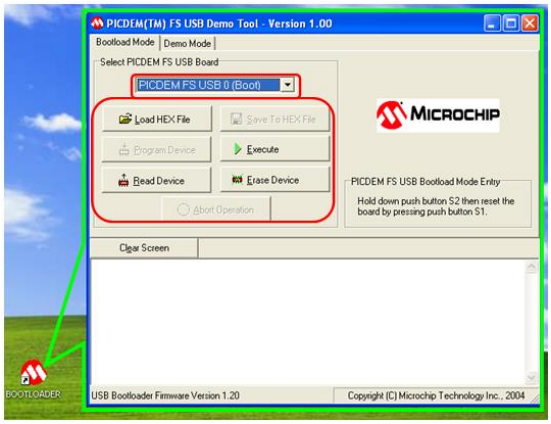
Nótese que al dar clic sobre la pestaña de selección se despliega un cuadro vacio, esto se debe a que la tarjeta aun no ha ingresado manualmente en modo bootloader, por lo tanto no ha sido detectada por el PC.

Figura 57

Paso 3: Ingresar la tarjeta en modo bootloader manualmente mediante los pulsadores BOOT y RESET en la tarjeta principal, para ello se debe mantener presionado el botón **BOOT**, posteriormente presionar y soltar el botón **RESET**, por último soltar el botón **BOOT**, es importante recordar que “el botón **BOOT** es el primero que se presiona y el último que se suelta”, si esta operación se realiza adecuadamente los dos diodos LED verdes en la tarjeta principal empezaran a parpadear intercaladamente uno luego del otro de forma consecutiva.

Figura 58

Paso 4: Seleccionar la tarjeta en modo Bootloader



Con el paso anterior el computador debe detectar que la tarjeta ha ingresado en modo bootloader, con lo que aparecerá el mensaje **PICDEM FS USB 0 (Boot)**. En la imagen se muestra que algunos botones se habilitan y otros no, esto es consecuente ya que los botones **programar** y **grabar** solo se deben habilitar cuando se haya seleccionado y cargado un código o cuando se haya realizado un proceso de lectura del microcontrolador.

Figura 59

Paso 5: Seleccionar el firmware o código a grabar en el microcontrolador de la tarjeta principal

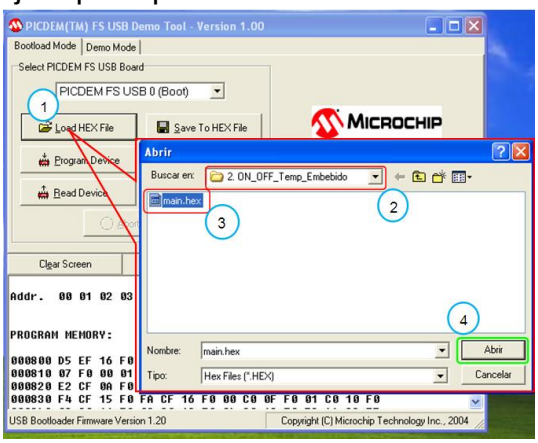


Figura 60

En este caso se selecciona el código para el control ON/OFF para la planta de temperatura que viene en el CD de usuario, el archivo a grabar está en la carpeta:

2. ON_OFF_Tmep_Embebido, dentro de la carpeta ***Ejemplos_control_Embebido***

Al darle clic en el botón abrir o doble clic sobre el archivo main.hex, puede que aparezca una pequeña ventana, si es el caso y aparece se debe dar clic en la opción **Si**, esto se debe a que algunos de los fusibles de configuración son diferentes entre el código del bootloader y el código diseñado para el control ON/OFF, al oprimir el botón si de esta pantalla inmediatamente se carga el código y queda listo para ser transferido al microcontrolador.

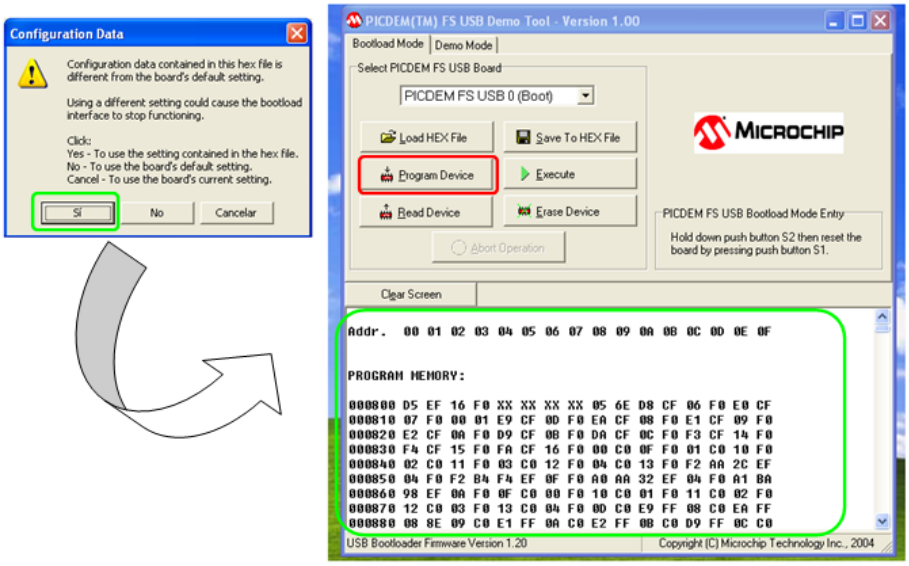


Figura 61

En la figura se muestra que al dar clic en la opción **Si** el código se carga en la ventana de mensajes, hay que notar que el código se carga desde la dirección 000800 en adelante, esto está diseñado así para que el código no sobrescriba el del bootloader en el microcontrolador, también se puede notar en el recuadro rojo que el botón de **Programar** también queda habilitado.

Paso 6: Grabar el código o firmware en el microcontrolador

Se oprime el botón **Program Device**, inmediatamente inicia el proceso, lo primero que hace la aplicación es borrar la memoria de programa para grabar el nuevo firmware, al terminar aparecen algunos mensajes en la ventana de mensajes, como se muestra en la siguiente figura.

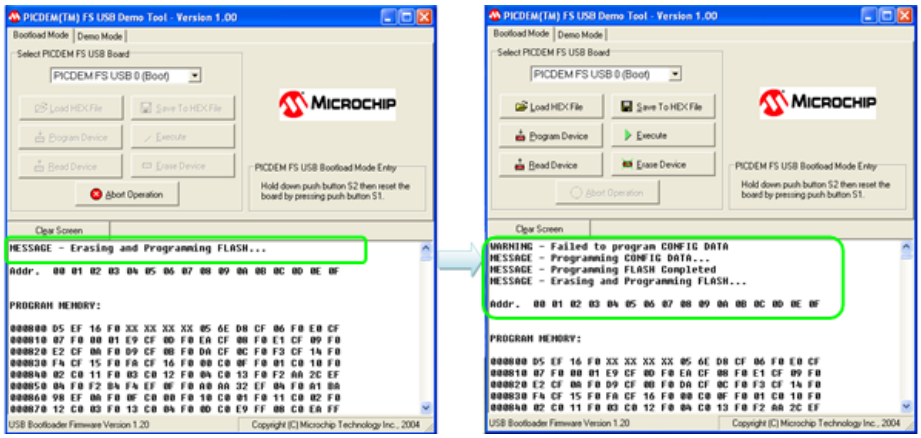


Figura 62

Paso 7: por ultimo solo queda ejecutar el nuevo código grabado en el microcontrolador de la tarjeta principal saliendo del modo bootloader, esto se puede hacer de dos maneras, la primera es manualmente oprimiendo el botón de RESET en la tarjeta principal o también desde el computador oprimiendo el botón **Execute**, al salir del modo bootloader la tarjeta principal queda lista con el código del control ON/OFF para la planta de temperatura para verificar su funcionamiento se debe abrir la aplicación de monitoreo en Matlab ingresar un set-point e iniciar el sistema.

3.6 Aplicaciones, interfaz de usuario para monitoreo e interfaz para control manual del sistema en Matlab.

En esta sección se presenta una descripción de uso de las interfaces de usuario del sistema, las cuales son: una para el monitoreo de las variables de las plantas y otra para realizar control manual.

Estas aplicaciones permiten establecer comunicación con la tarjeta principal a través del bus USB, además se pueden realizar diferentes actividades, como son: escoger tipo de planta a trabajar, ingresar el valor del set point o el esfuerzo de control, según la aplicación en la cual se esté trabajando, iniciar el proceso y guardar datos.

Las aplicaciones están desarrolladas con la herramienta GUIDE de Matlab, la cual 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, se personalizaron las funciones de modo que los componentes tengan el comportamiento que se explicara más adelante en esta sección.

3.6.1 Interfaz de usuario para el monitoreo de variables del sistema

Teniendo instalado Matlab en el equipo puede abrir la interfaz de usuario dirigiéndose a la siguiente dirección:

X:\CD_Usuario_Sistema_DCB01\Aplicacion_Monitoreo_Matlab

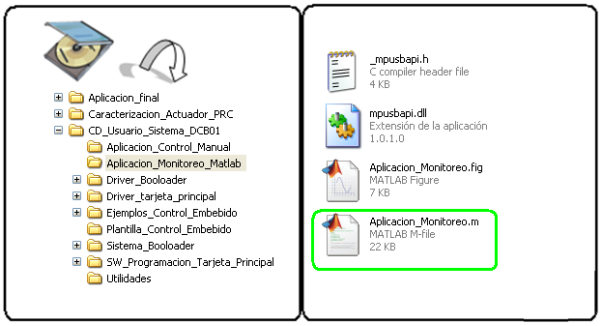


Figura 63

Allí encontrara cuatro archivos:

mpusbapi.dll es la librería de intercambio dinámico de datos que permite establecer la comunicación entre el PC y la tarjeta principal.

_mpusbapi.h son definiciones que necesita la librería mpusbapi.dll

Aplicación_Monitoreo.fig archivo de Matlab que contiene los componentes de la interfaz de usuario.

Aplicación_Monitoreo.m archivo de Matlab que contiene las funciones asociadas a los componentes de la interfaz de usuario.

Nota: es muy importante que el usuario NO modifique los archivos anteriormente mencionados. Especialmente la librería de intercambio dinámico de datos y las definiciones.

Para abrir la interfaz de usuario para el monitoreo de variables debe ejecutar el archivo Aplicación_Monitoreo.m

Descripción de la aplicación para el monitoreo de variables.

En las siguientes imágenes se muestran las partes que componen la interfaz para el monitoreo de variables, que se explicaran posteriormente:

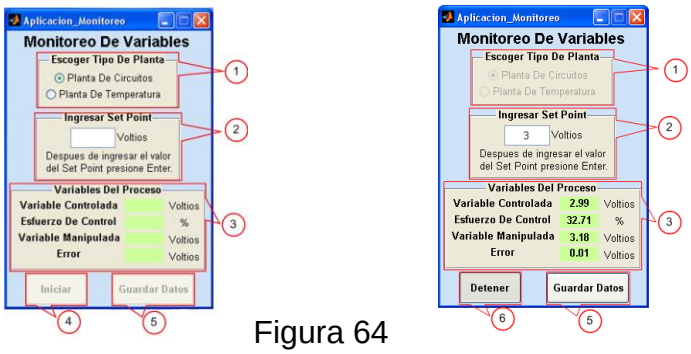


Figura 64

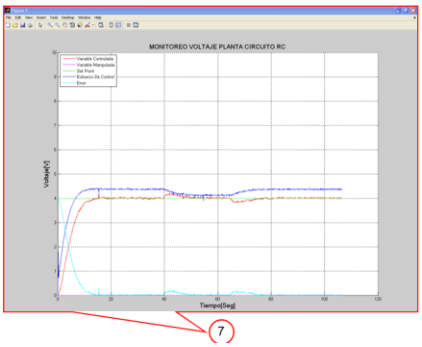


Figura 65

1. **Escoger tipo de planta:** puede escoger entre la planta de circuitos RC y la planta de temperatura, seleccionando una al hacer clic sobre los botones dispuestos al lado del nombre de cada planta. Esto es muy importante, dado que es lo que define el rango de trabajo, que influye en los valores que puede tomar el set point y los rangos de los ejes en la gráfica de las variables de proceso.
2. **Escoger set point:** el valor que se ingresa aquí depende del tipo de planta escogida, como se mencionó anteriormente, de modo que si se trabaja con la planta de circuitos RC el rango está entre 0 y 10 voltios, por otro lado si se experimenta con la planta de temperatura el rango está entre 25 y 85ºC.
3. **Variables del proceso:** en esta sección se actualizan los valores de las variables del proceso, recibidos desde la tarjeta principal.
4. **Iniciar el proceso:** al oprimir este botón se le envía una orden a la tarjeta principal para que inicie el proceso, se despliega una ventana donde se graficaran los datos del proceso y se actualizarán constantemente los valores de las variables del proceso.
5. **Guardar datos:** el usuario puede guardar las variables del proceso en cualquier momento sin tener que detener el experimento, al hacer clic sobre el botón *Guardar Datos,* lo cual le permite digitar del archivo y la ubicación en la que desea guardarlo.
6. **Detener el proceso:** puede detener el proceso pulsando el botón *Detener*, o cerrando la aplicación haciendo clic en el botón *Cerrar*, de la ventana de la interfaz de usuario, esta última acción elimina todas las variables del proceso.
7. **Grafica de variables:** es donde se grafican las variables del proceso, el usuario tiene a su disposición diversas herramientas de análisis, que brinda Matlab, como lo son: el zoom, la selección de datos y la inserción de mensajes, entre otros.

Funcionamiento de la interfaz *Monitoreo de Variables*

Para mostrar el funcionamiento de la interfaz se analizara el comportamiento del controlador PID diseñado para la planta de circuitos RC.

Antes de empezar debe asegurarse que la tarjeta principal tenga cargado el firmware del controlador PID para la planta de circuitos RC, de no ser así, puede referirse a la sección: *Modo de uso de bootloader para programación de la tarjeta principal.*
Una vez la tarjeta tenga grabado el firmware del controlador, se puede proceder del siguiente modo:

Paso 1: abrir el archivo **Aplicación_Monitoreo.m** que se encuentra ubicado como lo muestra la siguiente imagen:

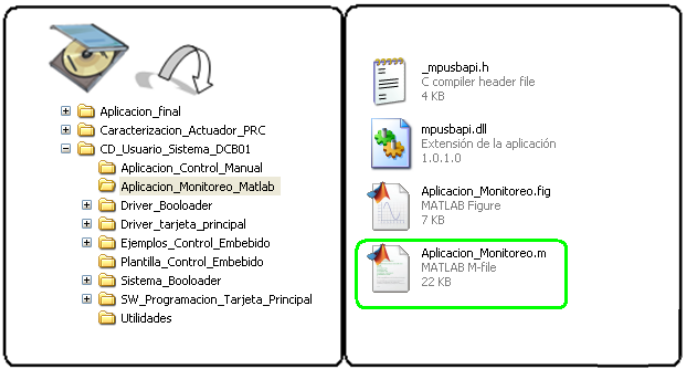


Figura 66

Paso 2: conectar la planta de circuitos RC a la tarjeta principal.
Paso 3: conectar la tarjeta principal al PC.
Paso 4: ejecutar el archivo **Aplicación_Monitoreo.m** lo cual abrirá una ventana como la mostrada en la siguiente imagen.

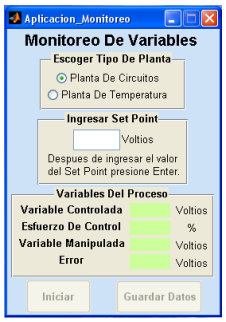


Figura 67

Paso 5: escoger el tipo de planta, en este caso se escoge la planta de circuitos RC, como se ve en la imagen anterior.
Si el usuario no selecciona alguna planta le aparecerá un mensaje pidiéndole que seleccione una, de igual forma las opciones de *Ingresar Set Point* y los botones *Iniciar* y *Guardar Datos* se deshabilitan, como se muestra en la siguiente imagen.

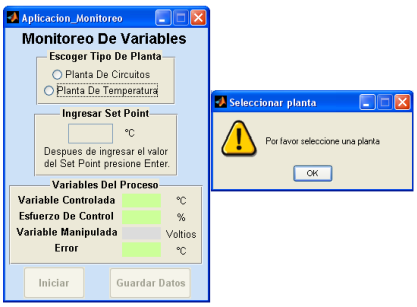


Figura 68

Paso 6: ingresar el valor del set point, este debe estar entre 0 y 10 voltios.

Si ingresa un valor fuera del rango de trabajo de la planta, se desplegara un mensaje de error que le recordara el rango de trabajo de la planta seleccionada. A continuación se muestra una imagen del mensaje.

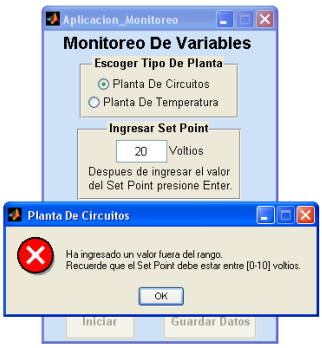


Figura 69

Paso 7: iniciar el proceso haciendo clic en el botón *Iniciar.*

En caso que la tarjeta no se encuentre conectada o no haya sido detectada se desplegara un mensaje indicándole al usuario que la tarjeta no se encuentra y se le da la opción de volver a buscarla o cancelar el proceso. A continuación se muestra una imagen del mensaje desplegado.

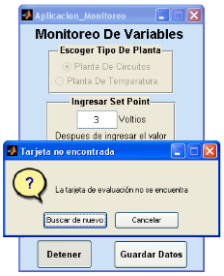


Figura 70

Si las conexiones se han realizado correctamente, es decir la tarjeta principal tiene una planta conectada con alimentación, y además está conectada al PC, entonces inicia el proceso de adquisición y grafica de las variables.

Cuando el proceso inicia en la interfaz se actualizan constantemente los valores de las variables del proceso, además se despliega una ventana donde se grafican las variables, como la mostrada a continuación:

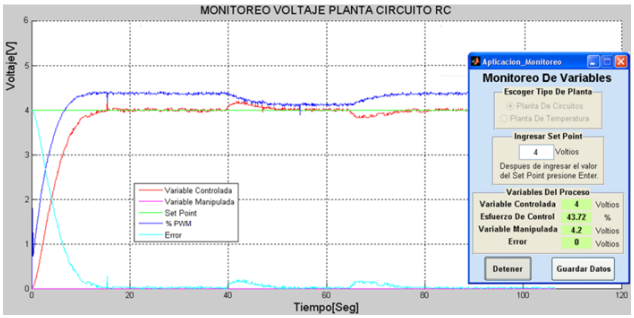


Figura 71

Paso 8: guardar los datos obtenidos en el proceso, haciendo clic sobre el botón *Guardar Datos,* al hacerlo le aparecerá una ventana en la que debe escribir el nombre y seleccionar la ubicación del archivo, este se almacenara con una extensión .mat, que luego puede ser cargado en Matlab para realizar los respectivos análisis.

A continuación se muestra una imagen de la ventana que aparece al pulsar el botón *Guardar Datos.*

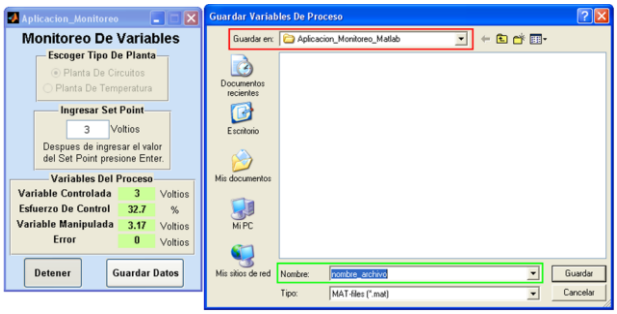


Figura 72

Paso 9: detener el proceso si ha terminado con el experimento, para ello pulse el botón *Detener,* lo cual desplegara una ventana que le preguntara si realmente desea detener el proceso, como se muestra en la siguiente imagen.

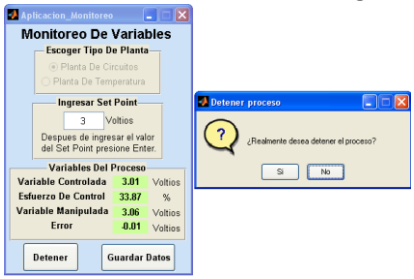


Figura 73

Si realmente desea detener el proceso y da clic en *Si,* entonces se le preguntara si desea guardar los datos del proceso, como se muestra en la siguiente imagen.

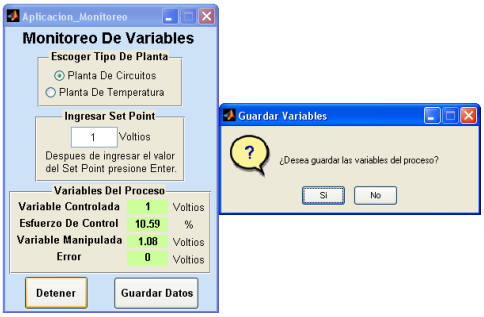


Figura 74

Posteriormente aparecerá un mensaje que le pide que reinicie la tarjeta principal, como se muestra a continuación.

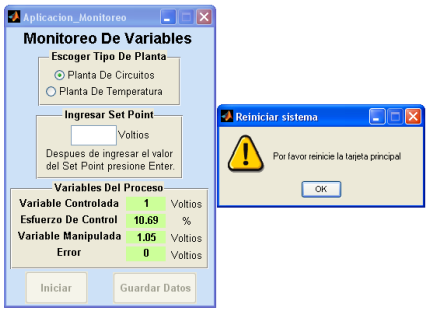


Figura 75

En caso de haber pulsado el botón *Detener* accidentalmente puede volver al proceso al hacer clic en *No*, cuando se le pregunta si desea detener el proceso.

Otra forma de detener el proceso es haciendo clic en el botón *Cerrar* de la ventana de la interfaz, esta es la forma más recomendable de reiniciar el sistema, dado que con esto se garantiza que todas las variables y funciones del proceso son eliminadas.

Al hacer clic sobre el botón *Cerrar,* aparece una ventana para confirmar si el usuario desea cerrar la aplicación, si pulso el botón por accidente, puede regresar al proceso haciendo clic en el botón *NO,* de lo contrario al pulsar el botón *SI* la aplicación se cerrara.

A continuación se muestra una imagen de la ventana de confirmación de cierre de la aplicación.

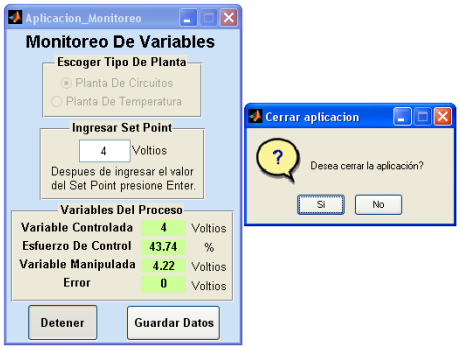


Figura 76

3.6.2 Interfaz de usuario para realizar control manual sobre las plantas

Teniendo instalado Matlab en el equipo puede abrir la interfaz de usuario dirigiéndose a la siguiente dirección:

X:\CD_Usuario_Sistema_DCB01\Aplicacion_Control_Manual

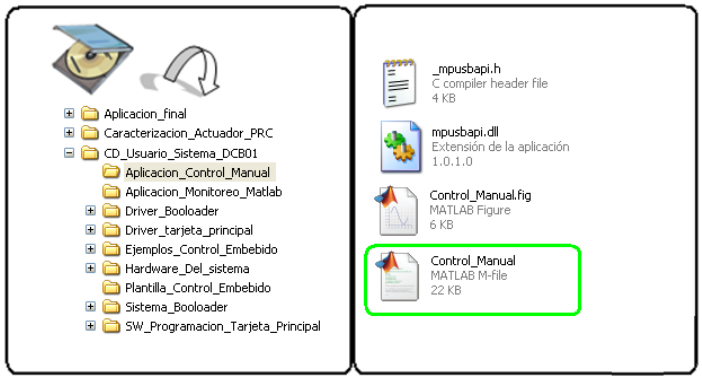


Figura 77

Allí encontrara cuatro archivos:
mpusbapi.dll es la librería de intercambio dinámico de datos que permite establecer la comunicación entre el PC y la tarjeta principal.
_mpusbapi.h son definiciones que necesita la librería mpusbapi.dll
Control_Manual.fig archivo de Matlab que contiene los componentes de la interfaz de usuario.
Control_Manual.m archivo de Matlab que contiene las funciones asociadas a los componentes de la interfaz de usuario.

Nota: es muy importante que el usuario NO modifique los archivos anteriormente mencionados. Especialmente la librería de intercambio dinámico de datos y las definiciones.

Para abrir la interfaz de usuario para el control manual de las plantas debe ejecutar el archivo **Control_Manual.m**, lo cual le abrirá una ventana como la mostrada en la siguiente imagen.

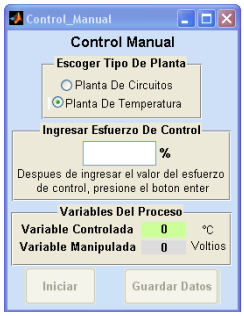


Figura 78

Descripción de la aplicación para el control manual de las plantas.

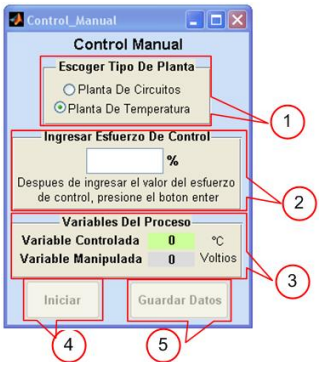


Figura 79

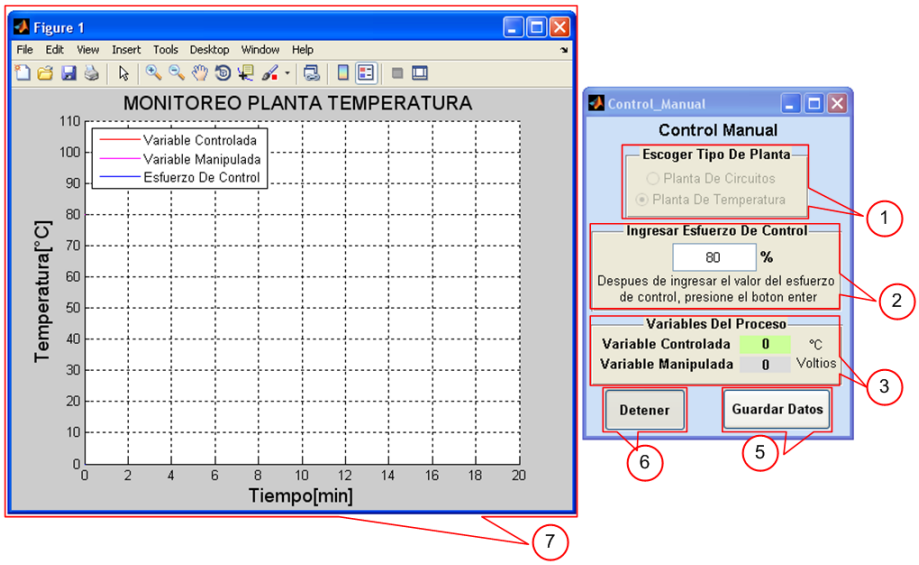


Figura 80

1. **Escoger tipo de planta:** puede escoger entre la planta de circuitos RC y la planta de temperatura, seleccionando una al hacer clic sobre los botones dispuestos al lado del nombre de cada planta.
2. **Ingresar esfuerzo de control:** el valor ingresado aquí está entre 0 y 100 en unidades de porcentaje de energía.
3. **Variables del proceso:** en esta sección se actualizan los valores de las variables del proceso, recibidos desde la tarjeta principal.
4. **Iniciar el proceso:** al oprimir este botón se le envía una orden a la tarjeta principal para que inicie el proceso, se despliega una ventana donde se grafican los datos del proceso y se actualizan constantemente los valores de las variables del proceso.
5. **Guardar datos:** el usuario puede guardar las variables del proceso en cualquier momento sin tener que detener el experimento, al hacer clic sobre el botón *Guardar Datos,* lo cual le permite digitar del archivo y la ubicación en la que desea guardarlo.
6. **Detener el proceso:** puede detener el proceso pulsando el botón *Detener*, o cerrando la aplicación haciendo clic en el botón *Cerrar*,  de la ventana de la interfaz de usuario, esta última acción elimina todas las variables del proceso.
7. **Grafica de variables:** es donde se grafican las variables del proceso, el usuario tiene a su disposición diversas herramientas de análisis, que brinda Matlab, como lo son: el zoom, la selección de datos y la inserción de mensajes, entre otros.

Funcionamiento de la interfaz *Control Manual*

Para mostrar el funcionamiento de la interfaz se desarrollara una práctica con el fin de obtener una curva de reacción de la planta de temperatura.
El proceso no se seguirá en detalle puesto que los mensajes desplegados por esta aplicación son los mismos mencionados en la interfaz *Monitoreo de Variables.*

Antes de empezar debe asegurarse que la tarjeta principal tenga cargado el firmware del control manual desde el PC, de no ser así, recuerde que el archivo se encuentra ubicado en el siguiente directorio:

X:\CD_Usuario_Sistema_DCB01\Ejemplos_Control_Embebido\5. Control_manual_desde_PC_Temp_RC

Si necesita más información sobre como grabar el firmware en la tarjeta principal, puede referirse a la sección: *Modo de uso de bootloader para programación de la tarjeta principal.*

Una vez la tarjeta tenga grabado el firmware del controlador, se puede proceder del siguiente modo:

Paso 1: abrir el archivo **Control_Manual.m** que se encuentra en:

X:\CD_Usuario_Sistema_DCB01\Aplicacion_Control_Manual\Control_Manual

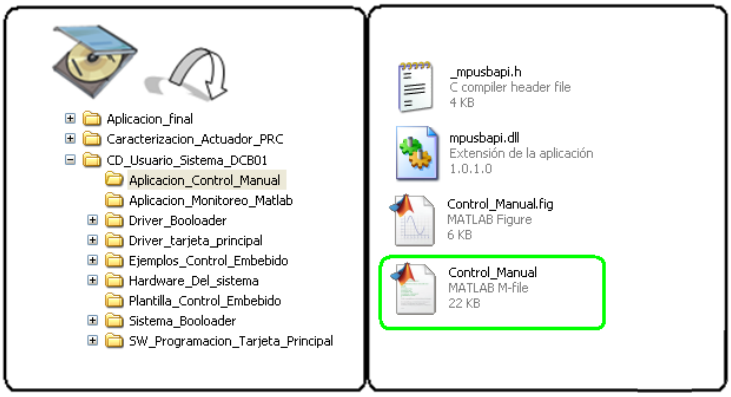


Figura 81

Paso 2: conectar la planta de temperatura a la tarjeta principal.
Paso 3: conectar la tarjeta principal al PC.
Paso 4: ejecutar el archivo **Control_Manual.m**.
Paso 5: escoger el tipo de planta, en este caso se escoge la planta de temperatura, como se muestra en la imagen anterior.
Paso 6: ingresar el valor del esfuerzo de control, este debe estar entre 0 y 100%, en este caso se introdujo un esfuerzo de control del 15%.
Paso 7: iniciar el proceso haciendo clic en el botón *Iniciar.*
Paso 8: cuando la variable controlada se haya estabilizado, introduzca nuevamente un cambio en el esfuerzo de control, en este caso se introdujo un cambio en el esfuerzo de control del 20%.
Los datos obtenidos al transcurrir 1400 segundos se muestra en la siguiente imagen.

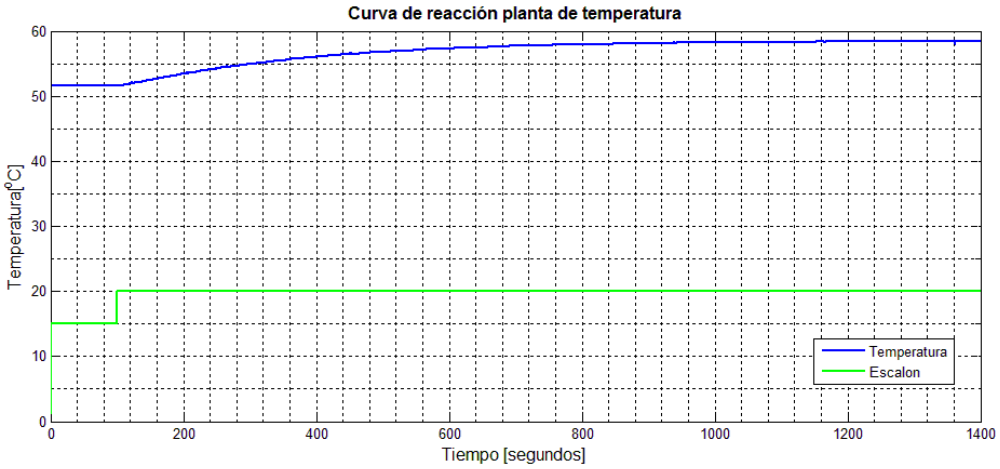


Figura 82

Paso 8: guardar los datos obtenidos en el proceso, haciendo clic sobre el botón *Guardar Datos,* al hacerlo le aparecerá una ventana en la que debe escribir el

nombre y seleccionar la ubicación del archivo, este se almacenara con una extensión .mat, que luego puede ser cargado en Matlab para realizar los respectivos análisis.

Paso 9: detener el proceso si ha terminado con el experimento, para ello pulse el botón *Detener,* lo cual desplegara una ventana que le preguntara si realmente desea detener el proceso. En caso de querer detener el proceso y oprimir el botón *Si,* entonces se le preguntara si desea guardar los datos del proceso.
Posteriormente aparecerá un mensaje que le pide que reinicie la tarjeta principal.

En caso de haber pulsado el botón *Detener* accidentalmente puede volver al proceso al hacer clic en *No*, cuando se le pregunta si desea detener el proceso.

Otra forma de detener el proceso es haciendo clic en el botón *Cerrar* de la ventana de la interfaz, esta es la forma más recomendable de reiniciar el sistema, dado que con esto se garantiza que todas las variables y funciones del proceso son eliminadas.
Al hacer clic sobre el botón *Cerrar,* aparece una ventana para confirmar si desea cerrar la aplicación, si pulso el botón por accidente, puede regresar al proceso haciendo clic en el botón *NO,* de lo contrario al pulsar el botón *SI* la aplicación se cerrara.

4. Ejemplos de uso e implementación de controladores en el sistema.

En esta sección se muestra paso a paso como reproducir los ejemplos que trae el CD de usuario en su carpeta ***Ejemplos control Embebido***, esto con el fin que el usuario se familiarice con manejo del sistema.
Se aclara que todos los ejemplos de implementación de controladores a desarrollar y que están presentes en el CD de usuario están basados en estrategias de control **embebido**, es decir todos los esquemas de control se implementan y se programan en el microcontrolador de la tarjeta principal. La aplicación de monitoreo en Matlab que se ejecuta en el computador solo le permite al usuario la selección del tipo de planta, ingresar un valor de referencia o set-point al inicio y durante la ejecución de la práctica, iniciar y detener el proceso y guardar datos del mismo.

4.1 Implementación un controlador ON/OFF embebido para planta de temperatura.

Un control tipo ON/OFF es un tipo de control lógico y la forma más simple de controlador, este control se basa en aplicar toda o nada de la energía disponible para llevar el sistema a un punto deseado.
A continuación se muestran los pasos para implementar un control de tipo ON/OFF embebido para la planta de temperatura y analizar su comportamiento en la aplicación de monitoreo desde Matlab en un PC.

Paso 1: realizar una copia de la carpeta ***Plantilla_Control_Embebido*** en algún lugar de preferencia en el PC, se recomienda renombrar la copia de la carpeta según la práctica a desarrollar.

Paso 2: abrir el proyecto en Mplab, dentro de la carpeta dar doble clic sobre el icono ***USB_board.mcpb.***

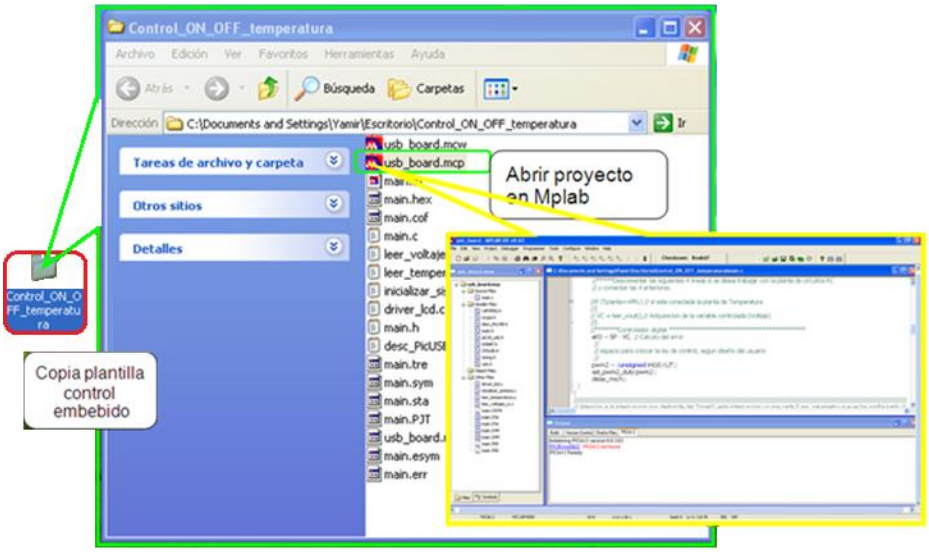


Figura 83

Paso 3: En la ventana de Matlab ubicar el archivo ***main.c***, en este archivo ubicar el ciclo ***while(true)***, el cual está destinado para que el usuario cree su código.

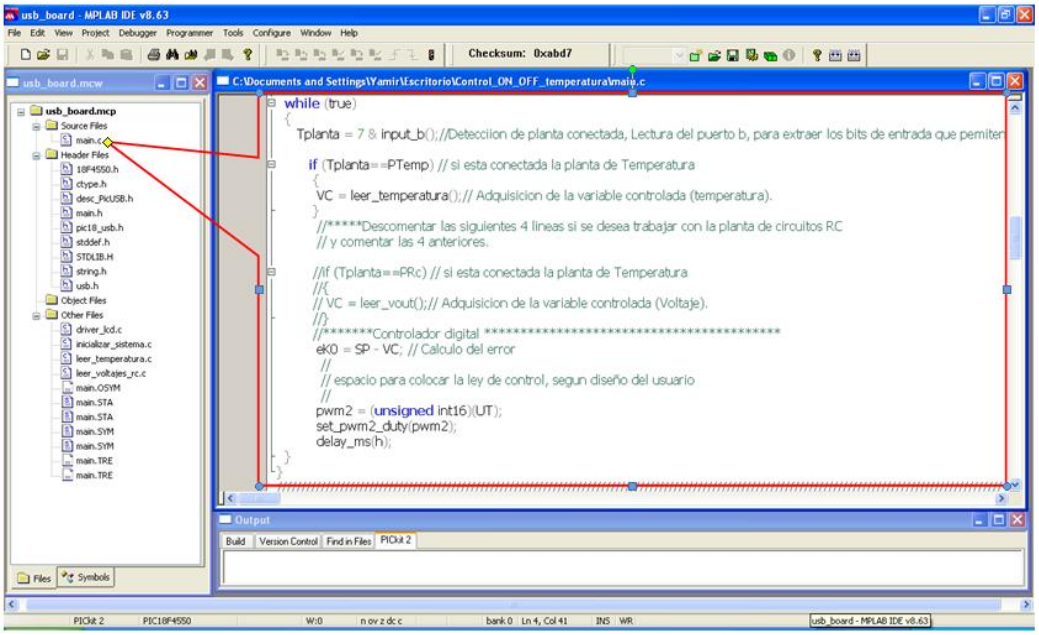


Figura 84

Paso 4: Diseñar y programar la ley de control en la plantilla

Debido a lo simple de la estructura del control ON/OFF y que la plantilla ya tiene definidas algunas variables comunes y necesarias para la implementación de cualquier tipo de controlador como son:

- VC = Variable controlada.
- SP = Set point
- eK0 = error

Solo basta con digitar la ley de control según la siguiente lógica:

- **Si error < 0**; quiere decir que la variable controlada (temperatura) está por debajo del valor del set point, entonces se debe **aplicar toda la energía disponible** (valor de registro para PWM en 1000) para aumentar la temperatura y disminuir el error.
- **Si error > 0**; esto significa que la variable controlada está por arriba del valor del set point, por lo que **la energía aplicada debe ser mínima** (valor de registro para PWM en 0).
- **Si error = 0;** esta condición no es muy probable debido a la conmutación de la energía de máxima a mínima, pero si es el caso se puede decir que se deja el sistema tal como esa sin hacer ningún cambio.

Dado que la energía aplicada se controla mediante el valor asignado al registro del módulo PWM en el microcontrolador, se aplica los rangos para máxima y mínima energía directamente en el algoritmo, por tanto no se debe escalar la salida del controlador y el valor se asigna directamente. Según lo anterior el código en C para el control ON/OFF queda de la siguiente forma:

```
while (true)
{
  Tplanta = 7 & input_b();//Deteccion de planta conectada, Lectura del puerto b, para extraer los bits de entrada que pemiten idenficar la planta

    if (Tplanta==PTemp) // si esta conectada la planta de Temperatura
     {
      VC = leer_temperatura();// Adquisicion de la variable controlada (temperatura).
     }
      //*****Descomentar las siguientes 4 lineas si se desea trabajar con la planta de circuitos RC
      // y comentar las 4 anteriores.

    //if (Tplanta==PRc) // si esta conectada la planta de Temperatura
    //{
    // VC = leer_vout();// Adquisicion de la variable controlada (Voltaje).
    //}
    //*******Controlador digital ****************************************
     eK0 = SP - VC; // Calculo del error

     if (eK0<0) UT = 1000; // aplica maxima energia a la planta de temperatura
     if (eK0>0) UT = 1;    // aplica minima energia a la planta de temperatura
     if (eK0==0) UT = UT; // no se realiza cambioa en la salida del controlador

     pwm2 = (unsigned int16)(UT);
     set_pwm2_duty(pwm2);
     delay_ms(h);
}
}
```

Figura 85

Paso 5: compilar el código, se puede hacer mediante la tecla ***F10***, o mediante el icono que se muestra en la siguiente imagen en la ventana de Mplab.

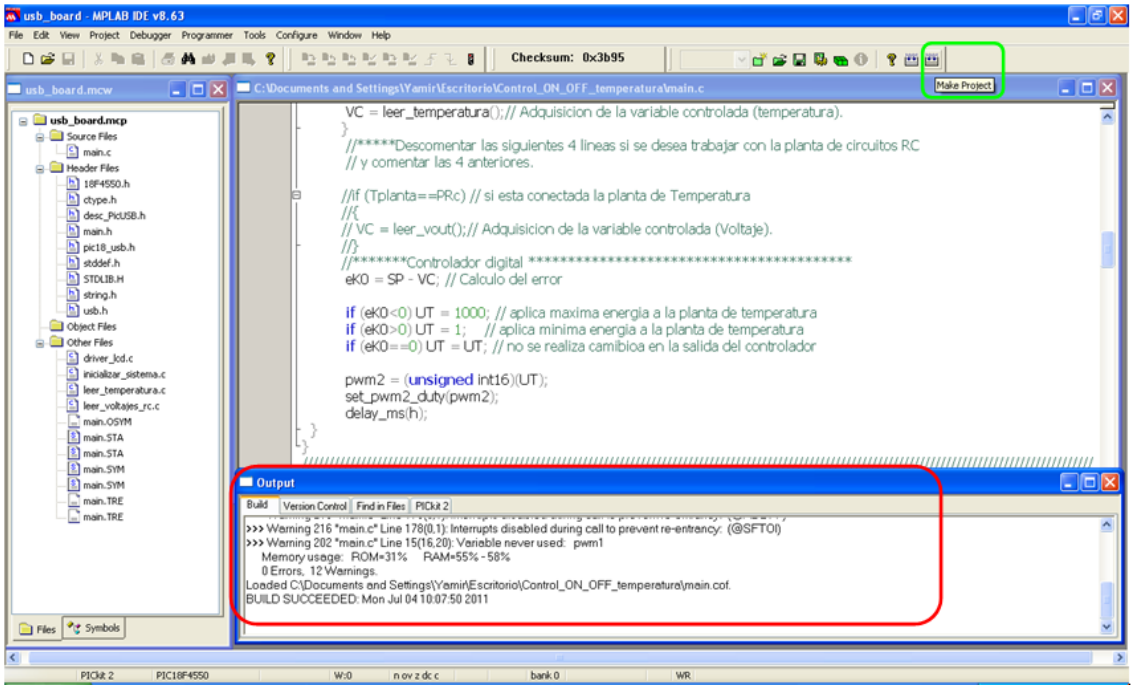


Figura 86

Verificar que en la ventana ***Output*** aparezca el mensaje "**BUILD SUCCEEDED**", si aparece "**BUIL FAILED**" significa que hay algún error, en la misma ventana se enumerara el error y se indicara la fuente del error para su corrección.

Paso 6: grabar el código para el microcontrolador de la tarjeta principal mediante el uso del bootloader (ver sección 3.5 de este manual de usuario).

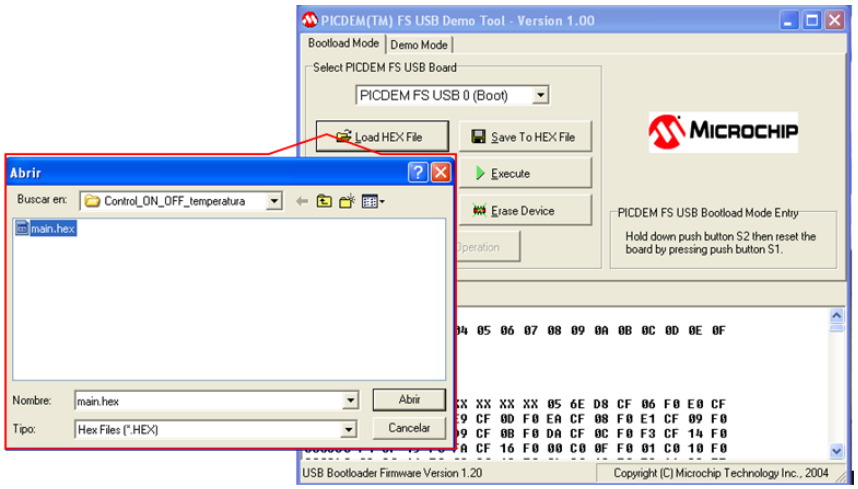


Figura 87

Paso 7: realizar la prueba de funcionamiento del controlador.

Para ello se procede a conectar la planta con la tarjeta principal y la fuente de alimentación, se abre la aplicación de monitoreo en Matlab y se ejecuta.

Las siguientes imágenes muestran el proceso y los resultados obtenidos:
Abrir la aplicación, como se muestra en la siguiente imagen

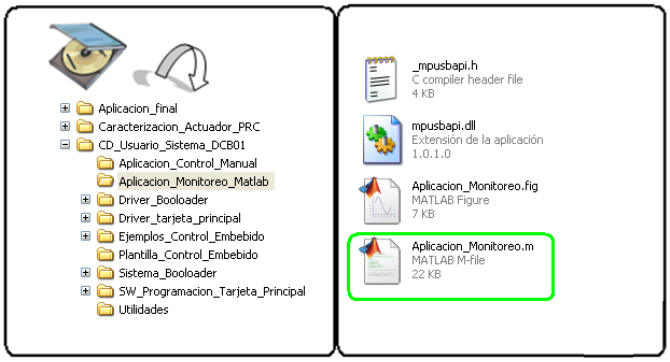


Figura 88

Ejecución de aplicación para monitoreo en Matlab, al ejecutar la aplicación se despliega en pantalla la ventana donde se debe seleccionar la planta de temperatura e ingresar un set-point valido (entre 25 y 85 ºC).

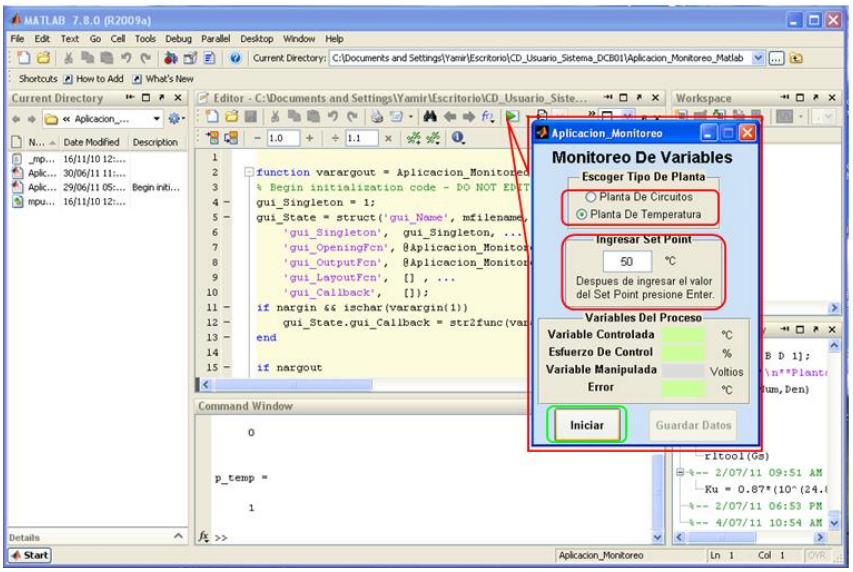


Figura 89

Luego de seleccionar el tipo de planta y haber ingresado el set-point o valor de referencia se oprime el botón iniciar, inmediatamente el sistema debe iniciar, desde la aplicación se envía la orden a la tarjeta principal para que ejecute el código dentro del bucle ***while(true)*** que en este caso es el correspondiente al control ON/OFF para la planta de temperatura, la siguiente es una imagen tomada después de cierto tiempo de ejecución, donde se ve el comportamiento del control implementado.

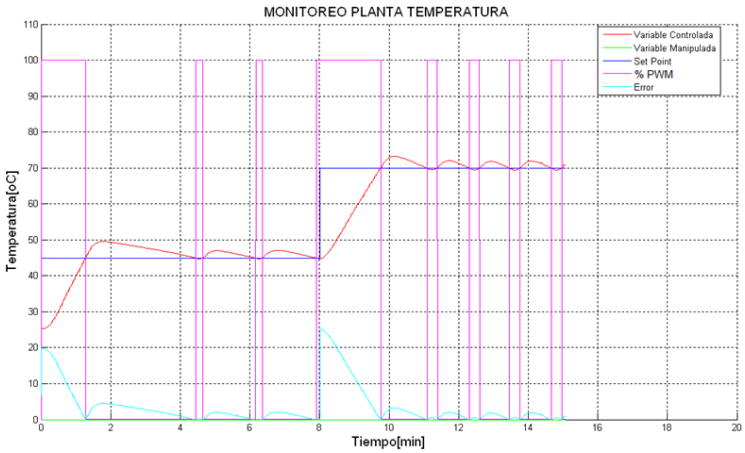


Figura 90

En la gráfica se puede analizar el comportamiento en forma de pulso del esfuerzo de control, el cual conmuta entre máximo y mínimo para tratar de mantener la variable controlada (temperatura) alrededor del valor de referencia o set-point, en la gráfica también se aprecia el comportamiento del error, se aclara que en la aplicación se grafica el valor absoluto del mismo, se hace así para mejorar la presentación de la gráfica. Por otro lado a los 8 minutos se puede observar un incremento en el valor del set point, lo cual hace que el porcentaje de PWM este en el 100% durante más tiempo, posteriormente la variable controlada continua oscilando alrededor del set point.

4.2 Obtener modelo alrededor de un punto de operación para la planta de temperatura.

A continuación se describe la práctica mediante la cual se pueden obtener datos de las plantas en lazo abierto, a través del ingreso del esfuerzo de control por parte del usuario.

Para la toma de datos se emplea la aplicación de control manual diseñada en Matlab, para una descripción de ella refiérase a la sección 3.6 de este manual.

Para tomar los datos se requiere tener instalado el firmware la carpeta *Control_Manual_Desde_PC_Temp_RC*, (ver sección 3.4 de este manual), previamente cargado en la tarjeta principal; (ver sección 3.5 de este manual).

Una vez haya cargado el firmware en la tarjeta principal debe seguir los siguientes pasos:

Paso 1: abrir el archivo **Control_Manual.m** que se encuentra en:

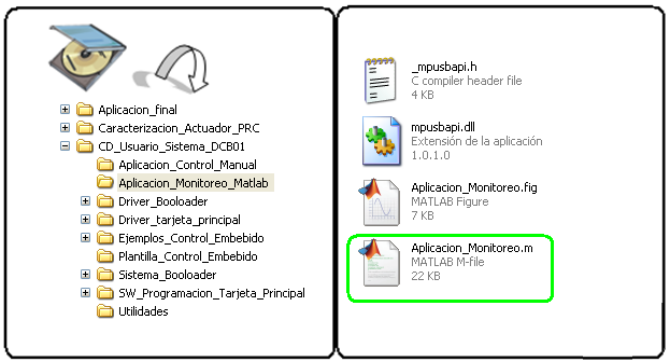


Figura 91

X:\CD_Usuario_Sistema_DCB01\Aplicacion_Control_Manual

Paso 2: conectar la planta de temperatura a la tarjeta principal.
Paso 3: conectar la tarjeta principal al PC.
Paso 4: ejecutar el archivo **Control_Manual.m** lo cual abrirá una ventana como la mostrada en la siguiente imagen.

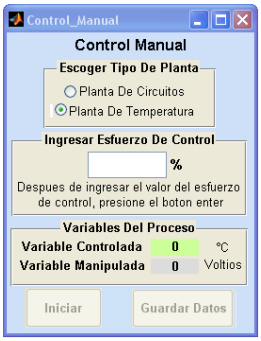


Figura 92

Paso 5: escoger el tipo de planta, en este caso se escoge la planta de temperatura, como se muestra en la imagen anterior.
Paso 6: ingresar el valor del esfuerzo de control, este debe estar entre 0 y 100%, en este caso se introdujo un esfuerzo de control del 15%.
Paso 7: iniciar el proceso haciendo clic en el botón *Iniciar.*
Paso 8: cuando la variable controlada se haya estabilizado, introduzca nuevamente un cambio en el esfuerzo de control, en este caso se introdujo un cambio en el esfuerzo de control del 20%.
Los datos obtenidos al transcurrir 1400 segundos se muestra en la siguiente imagen.

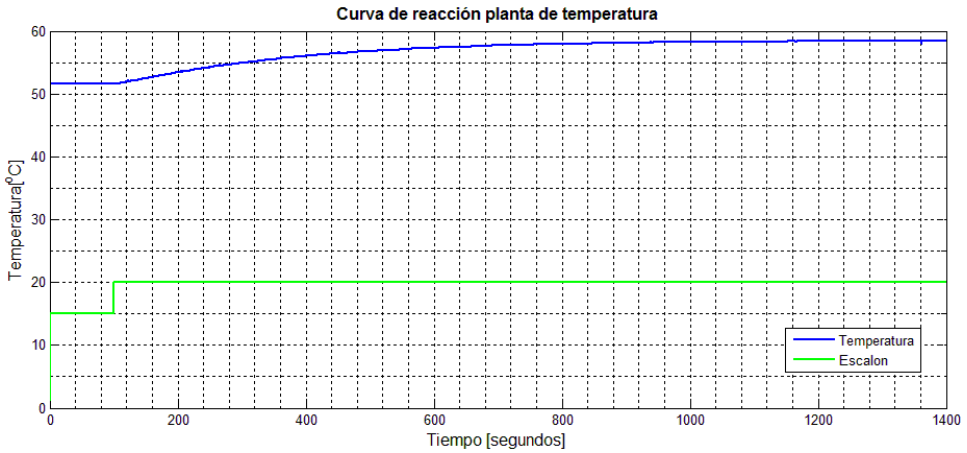


Figura 93

Paso 8: guardar los datos obtenidos en el proceso, haciendo clic sobre el botón *Guardar Datos,* al hacerlo le aparecerá una ventana en la que debe escribir el nombre y seleccionar la ubicación del archivo, este se almacenara con una extensión .mat, que luego puede ser cargado en Matlab para realizar los respectivos análisis.

Paso 9: cargar los datos guardados en Matlab a través del comando *load,* le aparecerán las variables almacenadas en el *Workspace* de Matlab, allí encontrara la variable *esf_c_porcent*, en ella podrá ver el momento exacto en el cual introdujo el escalón, al hacer doble clic sobre ella, de modo que se abra el editor de variables de Matlab, de allí debe obtener la posición a partir de la cual se realiza el cambio en el valor del esfuerzo de control, como se muestra a continuación:

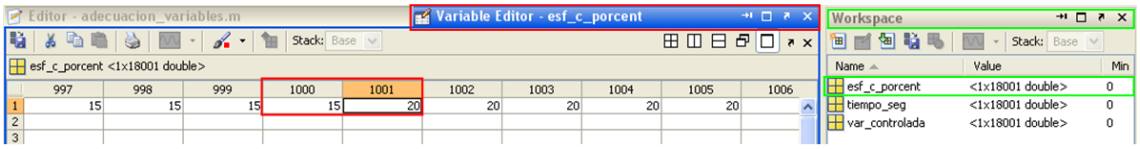


Figura 94

Paso 10: hacer una copia de la carpeta *Modelo_POMTM* ubicada en

Figura 95

Allí encontrara el archivo adecuacion_variables.m, este archivo se encargara de adecuar los datos para obtener la curva de reacción, para hallar el modelo de primer orden más tiempo muerto. A este archivo le debe ingresar la posición que obtuvo en el paso anterior, al ejecutarlo obtendrá un archivo *.mat con los valores de la curva de reacción.

Paso 11: cargar el archivo estimación_del_modelo.m, (ubicado en la misma carpeta copiada en el paso anterior), al ejecutarlo se cargara el archivo *.mat que obtuvo en el paso anterior, por lo tanto es importante que se asegure que este se encuentre en la misma carpeta de trabajo. Posteriormente se generaran los parámetros del modelo de primer orden más tiempo muerto según el método de la integral del error cuadrado. Los parámetros aparecerán en el *Comand Window*, de Matlab. Como se muestra en la siguiente imagen.

Figura 96

Además le proporcionara una imagen que compara la respuesta al escalón de los datos obtenidos en la práctica con la planta y la aproximación que el modelo estimado realiza de los mismos, como se muestra en seguida:

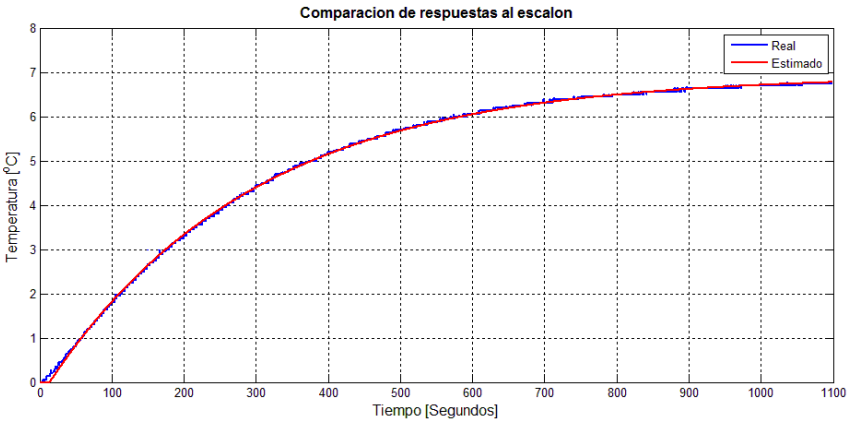


Figura 97

Como resultado los parámetros obtenidos son:
K = 1.3853
L = 13.5189
T = 284.3269

Donde
K: ganancia de la planta
L: tiempo muerto de la planta
T: tao de la planta

De modo que el modelo de primer orden más tiempo muerto queda de la siguiente forma:

4.3 Implementación Control proporcional embebido en 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:
(B. G. Lipták, 2006)

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

Dónde:
$U_{(t)}$: es el esfuerzo de control
$e_{(t)}$: es la desviación de la VC respecto al SP 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

A continuación se muestran los pasos para implementar un control de tipo proporcional embebido para la planta de temperatura y analizar su comportamiento en la aplicación de monitoreo desde Matlab en un PC.

Paso 1: realizar una copia de la carpeta ***Plantilla_Control_Embebido*** en algún lugar de preferencia en el PC, se recomienda renombrar la copia de la carpeta según la práctica a desarrollar.

Paso 2: abrir el proyecto en Mplab, dentro de la carpeta dar doble clic sobre el icono ***USB_board.mcpb.***

Paso 3: En la ventana de Matlab ubicar el archivo ***main.c***, en este archivo ubicar el ciclo ***while(true)***, el cual está destinado para que el usuario cree su código.

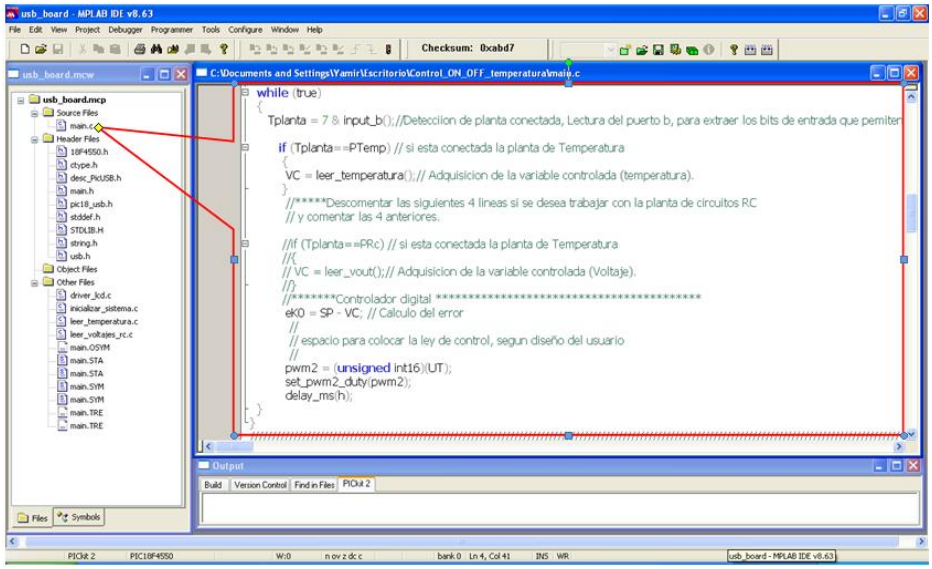


Figura 98

Paso 4: Diseñar y programar la ley de control en la plantilla

Teniendo en cuenta que ley de control que describe el funcionamiento del controlador proporcional está dada por:

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

Entonces se procede con la sintonización del controlador, lo cual en este caso se hace por el método propuesto por Ziegler y Nichols, para hallar el valor de la ganancia K_c.
Para implementar el método de Ziegler y Nichols, se emplea el modelo de POMTM de la planta de temperatura, el cual está dado por

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

La ecuación para hallar la ganancia proporcional según Ziegler y Nichols, es la siguiente:

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

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

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

Luego la ley de control que describe el funcionamiento del controlador proporcional para la planta de temperatura está dada por:

$$U_{(t)} = 15.18 e_{(t)} + b$$

Dado que la energía aplicada se controla mediante el valor asignado al registro del módulo PWM en el microcontrolador, se debe fijar el valor del esfuerzo de control a este registro. Por último se debe digitar la ley de control y la asignación del esfuerzo de control, en la sección controlador digital, de la plantilla.
Cabe resaltar que es importante para mejorar el desempeño del controlador saturar sus valores de salida, restringiendo los valores máximos y mínimos a los que puede llegar.

A continuación encontrara el código resultado de los pasos anteriores:

- 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 modulo PWM es de 10 bits.
h: periodo de muestreo
Como resultado del proceso anterior el código en C para el control PID queda de la siguiente forma en la plantilla:

```
while (true)
{
  Tplanta = 7 & input_b();//Deteccion de planta conectada, Lectura del puerto b, para extraer los bits de entrada que permiten identficar la planta
    if (Tplanta==PTemp) // si esta conectada la planta de Temperatura
    {
     VC = leer_temperatura();// Adquisicion de la variable controlada (temperatura).
    }
    //*******Controlador digital ****************************************
     eK0 = SP - VC; // Calculo del error
     UT = eK0*KP+ U0; // Ley de control Accion proporcional
     // Las siguientes dos lineas son un saturador para el controlador, limitan el valor maximo y minimo del esfuerzo de control
     if (UT>1000) UT = 1000;
     if (UT<0) UT = 0;
     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
}
}
```

Figura 99

Paso 5: compilar el código, se puede hacer mediante la tecla ***F10***, o mediante el icono que se muestra en la imagen del paso 5 de *Implementación un controlador ON/OFF embebido para planta de temperatura*.

Verificar que en la ventana ***Output*** aparezca el mensaje "**BUILD SUCCEEDED**", si aparece "**BUIL FAILED**" significa que hay algún error, en la misma ventana se enumerara el error y se indicara la fuente del error para su corrección.

Paso 6: grabar el archivo main.hex para el microcontrolador de la tarjeta principal, que se generó a través del proceso de compilación anterior, mediante el uso del bootloader (ver sección 3.5 Modo de uso de Bootloader para programación de la tarjeta principal. de este manual de usuario).

Paso 7: realizar la prueba de funcionamiento del controlador.

Para ello se procede a conectar la planta de temperatura con la tarjeta principal y la fuente de alimentación, se abre la aplicación de monitoreo en Matlab y se ejecuta.

Las siguientes imágenes muestran el proceso y los resultados obtenidos:

Figura 100

Ejecución de aplicación para monitoreo en Matlab, al ejecutar la aplicación se despliega en pantalla la ventana donde se debe seleccionar la planta de temperatura e ingresar un set-point valido (entre 25 y 85 ^{o}C).

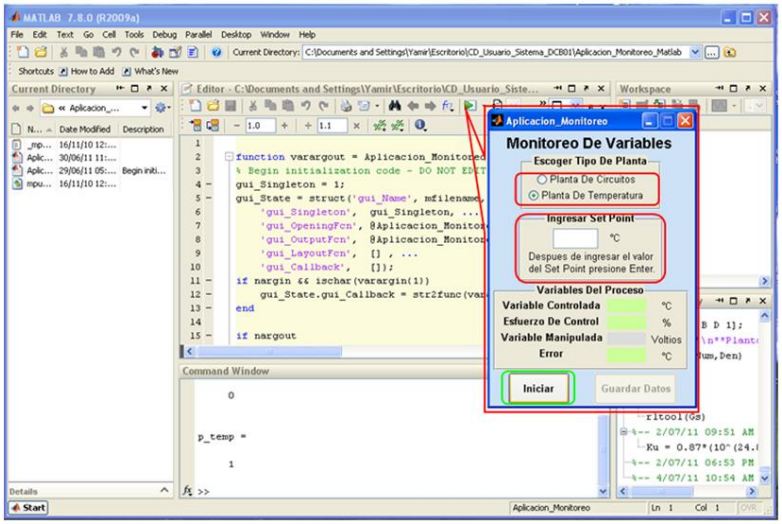


Figura 101

Luego de seleccionar el tipo de planta y haber ingresado el set-point o valor de referencia se oprime el botón iniciar, inmediatamente el sistema debe iniciar, desde la aplicación se envía la orden a la tarjeta principal para que ejecute el código dentro del bucle ***while(true)*** que en este caso es el correspondiente al control proporcional para la planta de temperatura, la siguiente es una imagen tomada después de cierto tiempo de ejecución, donde se ve el comportamiento del control implementado.

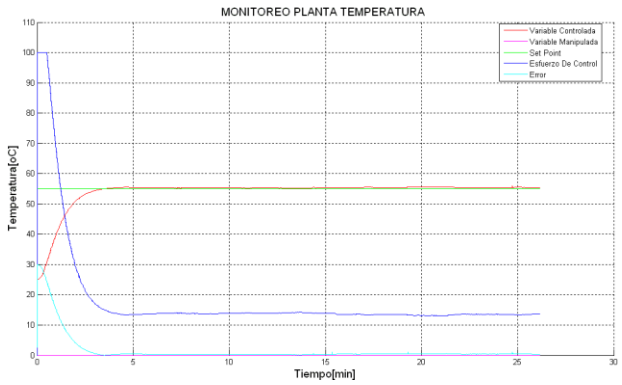


Figura 102

En la gráfica se puede apreciar un pequeño error de estado estacionario al introducir un set point de 55ºC, lo cual es característico de este tipo de control.

4.4 Implementación de controlador PID embebido en planta circuitos RC

El controlador PID ideal implementado en la planta de circuitos RC, genera una señal de control u(t) compuesta por tres términos aditivos que, como su nombre lo indica son proporcionales a la señal de error, a su integral y a su derivada.
La función de transferencia de un controlador PID ideal se muestra en la siguiente ecuación.

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

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

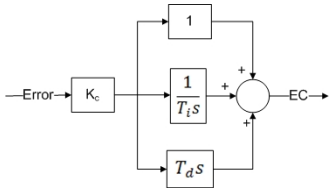


Figura 103

A continuación se muestran los pasos para implementar un controlador PID embebido para la planta de circuitos RC y analizar su comportamiento en la aplicación de monitoreo desde Matlab en un PC.

Paso 1: realizar una copia de la carpeta ***Plantilla_Control_Embebido*** en algún lugar de preferencia en el PC, se recomienda renombrar la copia de la carpeta según la práctica a desarrollar.

Paso 2: abrir el proyecto en Mplab, dentro de la carpeta dar doble clic sobre el icono ***USB_board.mcpb.***

Paso 3: En la ventana de Mplab ubicar el archivo ***main.c***, en este archivo ubicar el ciclo ***while(true)***, el cual está destinado para que el usuario cree su código.

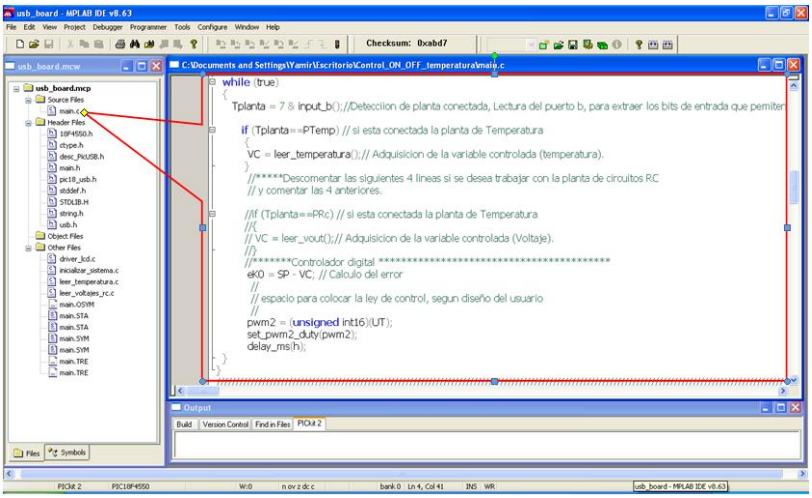


Figura 104

Paso 4: Diseñar y programar la ley de control en la plantilla

La ley de control de un controlador tipo PID debe implementarse en ecuaciones en diferencias, de modo que se debe iniciar por discretizar la función de transferencia del mismo, a continuación se muestra el procedimiento a través del método de la derivada, el cual consiste en aproximar la derivada por la pendiente de la recta que pasa por dos muestras consecutivas, con lo cual se obtiene

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

Que visto en sus correspondientes transformadas se convierte en:

$$sY_{(s)} \rightarrow \frac{1 - z^{-1}}{T} Y_{(z)}$$

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 $\frac{1-z^{-1}}{T}$.
Teniendo la función de transferencia del PID ideal, expresada como:

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

Discretizando a través del método de la derivada, se obtiene:

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

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

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

Dónde:

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

Sacando factor común se tiene:

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

Desarrollando el polinomio:

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

Luego:

$$U_{T(z)}(1 - z^{-1}) = E_{k(z)}K_p(1 - z^{-1} + a + (1 - 2z^{-1} + z^{-2})b)$$

De donde:

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

Factorizando:

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

Luego:

$$U_{T(z)}(1 - z^{-1}) = K_1 E_{k(z)} + K_2 E_{k(z)}z^{-1} + K_3 E_{k(z)}z^{-2}$$

Dónde:

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

$$K_2 = -K_p(1+2b) \rightarrow K_2 = -K_p\left(1+2\frac{T_d}{T}\right)$$
$$K_3 = K_p b \rightarrow K_3 = K_p\frac{T_d}{T}$$

Despejando $U_{t(z)}$:

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

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

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

La ecuación anterior, corresponde a la ley de control implementada en la tarjeta principal, la ecuación se implementa con las variables que se muestran a continuación.

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

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.

La sintonización del controlador se hace por el método de la ganancia última de Ziegler y Nichols. Para ello se emplea la herramienta rltool de Matlab, para obtener 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.

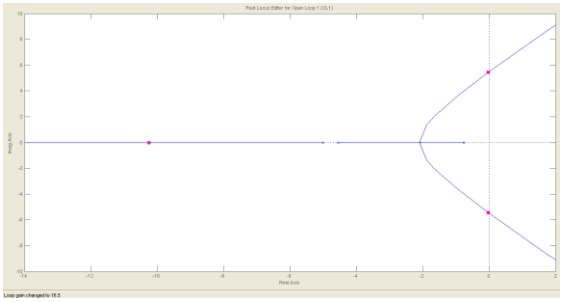


Figura 105

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:

$$T_u = \frac{2\pi}{w_n}$$

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 reemplazando se obtiene:

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

Dado que la plantilla ya tiene definidas algunas variables comunes y necesarias para la implementación de cualquier tipo de controlador como son:

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.

Dado que la energía aplicada se controla mediante el valor asignado al registro del módulo PWM en el microcontrolador, se debe asignar el valor del esfuerzo de control a este registro. Por último se debe digitar la ley de control, la asignación del esfuerzo de control y la actualización de las variables, en la sección controlador digital, de la plantilla. A continuación encontrara el código resultado de los pasos anteriores:

- 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
```

Según lo anterior el código en C para el control PID queda de la siguiente forma en la plantilla:

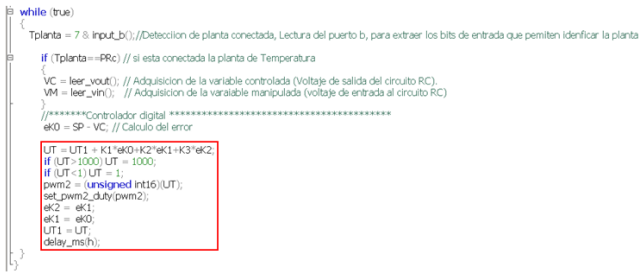


Figura 106

Paso 5: compilar el código, se puede hacer mediante la tecla ***F10***, o mediante el icono que se muestra en la imagen del paso 5 de *Implementación un controlador ON/OFF embebido para planta de temperatura*.
Verificar que en la ventana ***Output*** aparezca el mensaje "**BUILD SUCCEEDED**", si aparece "**BUIL FAILED**" significa que hay algún error, en la misma ventana se enumerara el error y se indica la fuente del error para su corrección.

Paso 6: grabar el archivo main.hex para el microcontrolador de la tarjeta principal, que se generó a través del proceso de compilación anterior, mediante el uso del bootloader (ver sección 3.5 Modo de uso de Bootloader para programación de la tarjeta principal.).

Paso 7: realizar la prueba de funcionamiento del controlador.

Para ello se procede a conectar la planta de circuitos RC con la tarjeta principal y la fuente de alimentación, se abre la aplicación de monitoreo en Matlab y se ejecuta.

Las siguientes imágenes muestran el proceso y los resultados obtenidos:

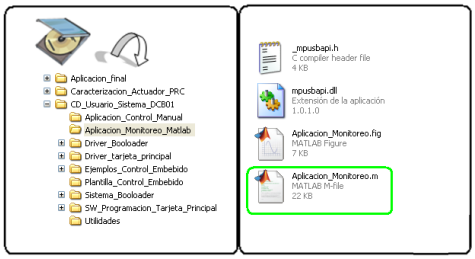


Figura 107

Ejecución de aplicación para monitoreo en Matlab, al ejecutar la aplicación se despliega en pantalla la ventana donde se debe seleccionar la planta de circuitos RC e ingresar un set point valido (entre 0 y 10 voltios)

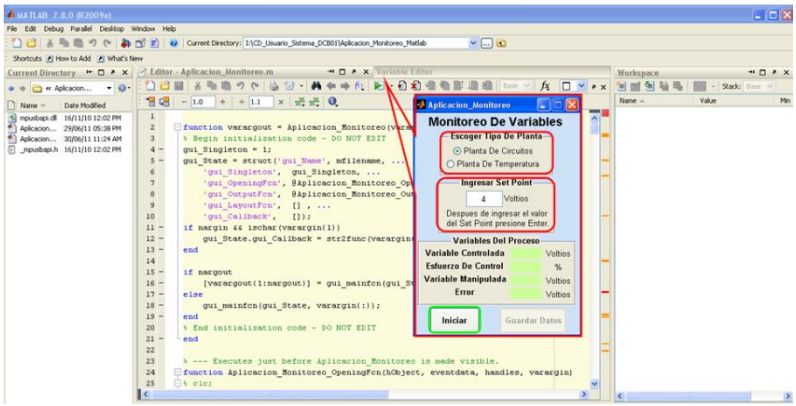


Figura 108

Luego de seleccionar el tipo de planta y haber ingresado el set point o valor de referencia se oprime el botón iniciar, inmediatamente el sistema debe iniciar. Desde la aplicación se envía la orden a la tarjeta principal para que ejecute el código dentro del bucle ***while(true)*** que en este caso es el correspondiente al control PID para la planta de circuitos RC , la siguiente es una imagen tomada después de cierto tiempo de ejecución, donde se ve el comportamiento del control implementado.

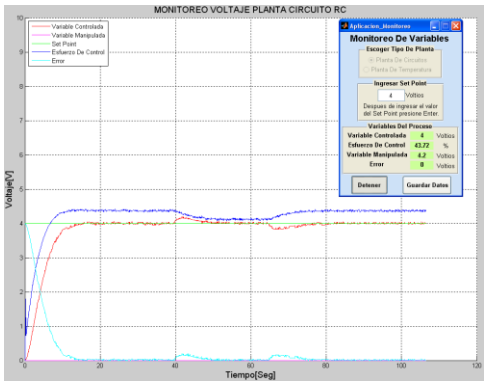


Figura 109

En la gráfica se puede analizar el comportamiento del controlador, como rápidamente compensa el cambio en la variable controlada y llega a error de estado estacionario igual a cero. Además se puede apreciar la reacción ante disturbios, como los introducidos cerca de los 40 segundos y los 70 segundos, estos disturbios se crearon deshabilitando y habilitando nuevamente la resistencia de carga del circuito, respectivamente. También se aprecia el comportamiento del error, cabe aclarar que en la aplicación se grafica el valor absoluto del mismo, se hace así para mejorar la presentación de la gráfica.

5. Listado de definiciones, variables, funciones y sentencias principales para programación en lenguaje C de la tarjeta principal.

En esta sección del manual de usuario encontrara material de ayuda para realizar la programación de la tarjeta principal en lenguaje C.

5.1 Tipos de Variables que maneja el compilador PCWHD de CCS

Tipos básicos:

Tabla 5.1

Tipo	Tamaño	Rango		Dígitos
		Con signo	Sin signo	
int1	1 bit	0 a 1	N/A	1/2
int8	8 bit	0 a 255	-128 a 127	2-3
int16	16 bit	0 to 65535	-32768 a 32767	4-5
int32	32 bit	0 a 4294967295	-2147483648 to 2147483647	9-10
float32	32 bit	-1.5×10^{45} a 3.4×10^{38}		7-8

Tipos Estándar en C:

Tabla 5.2

short	int1
char	unsigned int8

int	int8
long	int16
long long	int32
float	float32

5.2 Definiciones y variables incluidas en la plantilla del proyecto

Las siguientes definiciones y variables, son incluidas en el código de la plantilla del proyecto que trae el CD de usuario.

Definiciones propias del sistema

Se usan en sentencias *if* para determinar que planta está conectada a la tarjeta principal.

Tabla 5. 3

Definición	Descripción
#define PTemp 2	// numero para la planta de temperatura
#define PRc 4	// numero para la planta del circuito RC
Ejemplo de uso	
if (Tplanta == PTemp**){** **Printf('planta de temperatura conectada');** **VC = leer_temperatura();}**	

Variables pre-declaradas del sistema: las variables que se muestran a continuación

Tabla 5. 4

Nombre de la variable	*Tplanta*
Tipo	*Int 8*
Archivo donde está declarada	*main.h*
Descripción	*En esta variable se leen tres bits de los tres pines menos significativos del puerto B del microcontrolador, se utiliza para detectar si hay alguna planta conectada a la tarjeta principal.*

Tabla 5. 5

Nombre de la variable	*iniciar*
Tipo	*Int 8*
Archivo donde está declarada	*main.c*
Descripción	*En esta se carga un valor que permite ejecutar el programa en el microcontrolador una vez se dé la orden de iniciar desde la aplicación en el computador.*

Tabla 5. 6

Nombre de la variable	*pwm1*
Tipo	*Int16*
Archivo donde está	*main.c*

declarada	
Descripción	*Esta es una variable utilizada para fijar el ciclo útil del módulo pwm1, el cual es encargado de generar la señal para el esfuerzo de control.*

Tabla 5. 7

Nombre de la variable	*pwm2*
Tipo	*Int16*
Archivo donde está declarada	*main.c*
Descripción	*Esta es una variable utilizada para fijar el ciclo útil del módulo pwm2, el cual se puede usar para controlar la velocidad del ventilador en la planta de temperatura para generar disturbios.*

Tabla 5. 8

Nombre de la variable	*SP*
Tipo	*float*
Archivo donde está declarada	*main.c*
Descripción	*Esta es una variable destinada a almacenar el valor del Setpoint o valor de referencia, se utiliza en algoritmos de control para determinar el error.*

Tabla 5. 9

Nombre de la variable	*VC*
Tipo	*float*
Archivo donde está declarada	*main.c*
Descripción	*Esta es una variable destinada a almacenar el valor la variable controlada, en esta variable se recibe la información procedente de los sistemas sensor transmisor de cada planta.*

Tabla 5. 10

Nombre de la variable	*VM*
Tipo	*float*
Archivo donde está declarada	*main.c*
Descripción	*Esta es una variable destinada a almacenar el valor la variable manipulada, esta variable solo es útil cuando se opera con la planta de circuitos RC, ya que es la única de las dos plantas que tiene implementado un sistema sensor transmisor para esta variable de proceso.*

Tabla 5. 11

Nombre de la variable	*UT*
Tipo	*float*
Archivo donde está	*main.c*

declarada	
Descripción	*Esta es una variable destinada a almacenar el valor del generado a la salida del controlador (esfuerzo de control), su valor debe ser escalado si es necesario para convertirlo a valores enteros de 0 a 1000, que son los aceptados por el registro que controla el ciclo útil del módulo PWM.*

Tabla 5. 12

Nombre de la variable	*UTp*
Tipo	*float*
Archivo donde está declarada	*main.c*
Descripción	*En esta variable se almacena el valor del asignado al registro del módulo pwm1, en unidades de porcentaje, para luego enviarlo a la interfaz de monitoreo en el computador.*

Tabla 5. 13

Nombre de la variable	*eK0*
Tipo	*float*
Archivo donde está declarada	*main.c*
Descripción	*En esta variable se almacena el resultado del error, el cual se determina por la resta entre el valor del Set-Point y la variable controlada*

Tabla 5. 14

Nombre de la variable	*h*
Tipo	*Unsigned int16*
Archivo donde está declarada	*main.c*
Descripción	*En esta variable se almacena el valor en milisegundos que permite generar un retardo para realizar el periodo de muestreo, por defecto en la plantilla es fija en 250ms, periodo de muestreo por defecto para la planta de temperatura, el usuario puede modificarlo según el tipo de planta y el diseño de la práctica a implementar.*

5.3 Variables para operación con las plantas de temperatura y circuitos RC.

Tabla 5. 15

Función	*leer_temperatura();*
Descripción	*Permite obtener el valor de la temperatura medida por el sensor DS18S20 de la planta de temperatura.*
Sintaxis	*float var = leer_temperatura(void);*
Parámetros que recibe	*Ninguno*
Retornos	*var, es el valor de la temperatura en ^{o}C*
Requerimientos	*Declarar la variable de retono como flotante, en la plantilla se usa VC.*
Ejemplo de uso	*VC = leer_temperatura();*

Archivo donde se encuentra	*leer_temperatura.c*

Tabla 5. 16

Función	*leer_vout();*
Descripción	*Permite obtener el valor del voltaje a la salida del circuito RC, es decir la variable controlada (VC)*
Sintaxis	*float var = leer_vout (void);*
Parámetros que recibe	*Ninguno*
Retornos	*var, es el valor del voltaje de salida en la planta de circuitos RC.*
Requerimientos	*Declarar la variable de retorno como flotante, en la plantilla se usa VC.*
Ejemplo de uso	*VC = leer_vout();*
Archivo donde se encuentra	*leer_voltajes_rc.c*

Tabla 5. 17

Función	*leer_vin();*
Descripción	*Permite obtener el valor del voltaje a la entrada del circuito RC, es decir la variable manipulada (VM)*
Sintaxis	*float var = leer_vin (void);*
Parámetros que recibe	*Ninguno*
Retornos	*var, es el valor del voltaje de entrada a la planta de circuitos RC.*
Requerimientos	*Declarar la variable de retorno como flotante, en la plantilla se usa VC.*
Ejemplo de uso	*VM = leer_vin();*
Archivo donde se encuentra	*leer_voltajes_rc.c*

Tabla 5. 18

Función	*leer_vrc1();*
Descripción	*Permite obtener el valor del voltaje a la salida de la malla 1 de tres que componen la planta RC.*
Sintaxis	*float var = leer_vrc1 (void);*
Parámetros que recibe	*Ninguno*
Retornos	*var, es el valor del voltaje de salida de la malla 1*
Requerimientos	*Declarar la variable de retorno como flotante*
Ejemplo de uso	*V_malla1 = leer_vrc1();*
Archivo donde se encuentra	*leer_voltajes_rc.c*
Función	***leer_vrc2();***
Descripción	*Permite obtener el valor del voltaje a la salida de la malla 2 de tres que componen la planta RC.*
Sintaxis	*float var = leer_vrc2 (void);*
Parámetros que recibe	*Ninguno*
Retornos	*var, es el valor del voltaje de salida de la malla 2*
Requerimientos	*Declarar la variable de retorno como flotante*
Ejemplo de uso	*V_malla1 = leer_vrc2();*

Archivo donde se encuentra	*leer_voltajes_rc.c*

5.4 Funciones para comunicación USB

Tabla 5. 19

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 endoint,int*buffer,int16buffersize, bit_toogle);*	Endpoint, Tamaño buffer y 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(int8 endpoint);	Endpoint	Retorna el tamaño del paquete de entrada.
*int16 usb_get_packet(int8 endpoint, int8 * buffer, int16 buffersize,);*	Endpoint, Tamaño buffer y 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.

5.5 Manejo de pantalla LCD

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 se puede escribir directamente para mensajes fijos.	*Lcd_putc ("conexio_ok");*	Permite escribir el mensaje en la pantalla, suele ser usada en conjunto con la 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);*

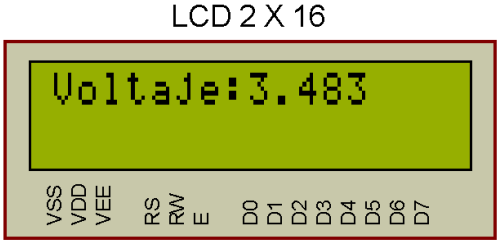


Figura 110

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.

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

Tabla 5.20

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

NOTA: Si el usuario requiere más información sobre lenguaje C para programación del microcontrolador de la tarjeta principal, en el CD de usuario en la carpeta utilidades se encuentra el manual de usuario del compilador CCS.

6. Notas para usuarios desarrolladores.

En esta sección, se dan indicaciones para usuarios avanzados que desean realizar modificaciones al sistema o desean reproducirlo.

6.1 Como reproducir el hardware del sistema:

El diseño de las placas de circuitos impreso (PCB) fue desarrollado en el software CadSoftEagle, en el CD de usuario encontrara una carpeta llamada *hardware del sistema*, dentro de esta carpeta se encuentran archivos en carpetas separadas que corresponden al diseño electrónicos de cada tarjeta, en la carpeta correspondiente se encuentran el archivo del esquemático (extensión *.sch*) y un archivo con el diseño PCB (extensión *.brd*) que pueden abrirse en el software CadSoftEagle, del este último se entrega el instalador del software, este se encuentra en la carpeta *utilidades* en el CD de usuario.

6.2 Que se debe tener en cuenta si se desea agregar una nueva planta al sistema:

Es sistema para implementación de controladores digitales ha sido diseñado con base a una arquitectura modular, de modo que este permite agregar nuevos módulos, si el usuario desea puede diseñar y construir nuevas plantas experimentales y agregarlas al sistema, para ello debe tener en cuenta aspectos como:

- Tener en cuenta el tipo de conector que utiliza la tarjeta principal y las señales de cada pin (ver sección 2 de este manual).
- Analizar el consumo del nuevo módulo, ya que la fuente externa que utiliza el sistema puede proveer 12V tipo DC y un máximo de 1.5 Amperios de corriente.
- Diseñar y crear archivos (librerías) independientes con funciones que permitan la adquisición de las variables de forma que el usuario solo tenga que llamar una función para realizar la tarea.
- Agregar mediante hardware, un código binario para identificación de la planta, diferente a 010 y 100, (ver sección 2 de este manual en la descripción de cada planta existente, las señales del conector).

6.3 Como modificar aspectos de la comunicación USB:

En la comunicación USB un aspecto importante es saber qué tipo de datos se comunican y de qué forma, el sistema ya tiene estandarizado el sistema de comunicación de modo que se tiene definido cierto tipo de datos, si el usuario desea realizar modificaciones debe tener lo siguiente:

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:

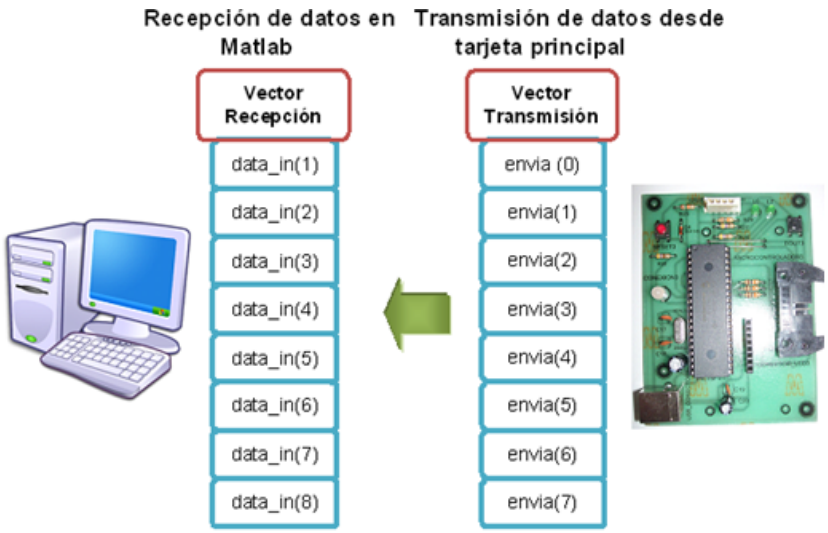


Figura 111

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

Tabla 6. 1

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 del esfuerzo de control, UTp en el microcontrolador.
envía(6) → data_in(7)	Parte decimal de la variable controlada, 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:

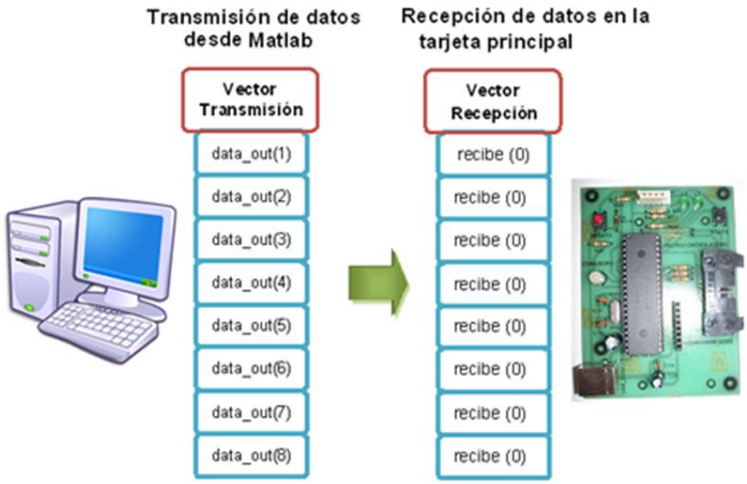


Figura112

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

Tabla 6. 2

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

Adicionalmente en la carpeta utilidades del CD de usuario se ha agregado la carpeta ***Basico_sobre_USBy Matlab*** cuyo contenido son dos ejemplos básicos y documentados sobe como implementar una comunicación USB entre Matlab y el PIC 18F4550.