UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA
DEPARTAMENTO DE SISTEMAS Y AUTOMÁTICA
PROYECTO DE TRABAJO ESPECIAL DE GRADO

IMPLEMENTACION DE UN SISTEMA DE MONITOREO DE TEMPERATURA PARA OBTENER MEDICIONES DEL EQUIPO EVAPORADOR-CONDENSADOR VERTICAL DEL LABORATORIO DE INGENIERIA QUÍMICA HACIENDO USO DEL MICROCONTROLADOR PIC18F2550

Autores:
José G. Salas J.
Simón A. Vitriago R.
Tutor:
Ing. Luis Llave

Bárbula, Noviembre de 2011

UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA
DEPARTAMENTO DE SISTEMAS Y AUTOMÁTICA
PROYECTO DE TRABAJO ESPECIAL DE GRADO

IMPLEMENTACION DE UN SISTEMA DE MONITOREO DE TEMPERATURA PARA OBTENER MEDICIONES DEL EQUIPO EVAPORADOR-CONDENSADOR VERTICAL DEL LABORATORIO DE INGENIERIA QUÍMICA HACIENDO USO DEL MICROCONTROLADOR PIC18F2550

TRABAJO ESPECIAL DE GRADO PRESENTADO ANTE LA ILUSTRE UNIVERSIDAD DE CARABOBO PARA OPTAR AL TITULO DE INGENIERO ELECTRICISTA

Autores:
José G. Salas J.
Simón A. Vitriago R.
Tutor:
Ing. Luis Llave

Bárbula, Noviembre de 2011

UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA
DEPARTAMENTO DE SISTEMAS Y AUTOMÁTICA
PROYECTO DE TRABAJO ESPECIAL DE GRADO

CERTIFICADO DE APROBACIÓN

Los abajo firmantes, miembros del jurado asignado para evaluar el trabajo especial de grado titulado "**IMPLEMENTACION DE UN SISTEMA DE MONITOREO DE TEMPERATURA PARA OBTENER MEDICIONES DEL EQUIPO EVAPORADOR-CONDENSADOR VERTICAL DEL LABORATORIO DE INGENIERIA QUÍMICA HACIENDO USO DEL MICROCONTROLADOR PIC18F2550**", realizado por los bachilleres: Salas Jaimes José Gustavo, C.I: 18.082.869, y Vitriago Roman Simón Alejandro, C.I: 18.086.356, hacemos constar que hemos revisado y aprobado dicho trabajo.

_____

Prof. Luis Llave
TUTOR

_____

Prof. Demetrio Reylago
JURADO

_____

Prof. Roberto Muñoz
JURADO

DEDICATORIA

- *Dedicamos éste trabajo principalmente a Dios nuestro señor creador del cielo y de la tierra por habernos dado la vida.*

- *A nuestros padres, hermanos y demás familiares por ser la guía y el apoyo a lo largo de nuestra vida, por ayudarnos a culminar nuestra carrera universitaria y por contribuir a ser quienes somos.*

- *A nuestros amigos que nos han apoyado a lo largo de la carrera, que han compartido alegrías y tristezas.*

- *Finalmente, a todas aquellas personas que nos han ayudado a la culminación de nuestro trabajo especial de grado.*

AGRADECIMIENTO

- *Le agradecemos a Dios primero que todo por el don de la vida, por estar en cada paso que damos y por fortalecer nuestros corazones e iluminar nuestras mentes, para así culminar con éxito éste trabajo de grado.*

- *Gracias a nuestros padres, por siempre ser una de las fuentes principales para conseguir todas nuestras metas, además le agradecemos por su incondicional apoyo durante nuestra vida y especialmente durante nuestra carrera.*

- *Le agradecemos a nuestros hermanos y familiares por estar siempre a nuestro lado apoyándonos en todo momento con sus sabios consejos.*

- *Gracias a nuestros compañeros y amigos con los cuales hemos compartido mucho tiempo durante todo el curso de nuestra carrera y por estar siempre presente en todo momento.*

- *Le agradecemos al Prof. Luis Llave el cual ha sido nuestro guía y tutor para la realización de este trabajo de grado y que siempre nos brindo su ayuda para solventar cualquier inconveniente que se iba presentando a lo largo de éste trabajo.*

- *Además agradecemos a los profesores que siempre nos ofrecieron su ayuda cuando teníamos cualquier inquietud o duda respecto a alguna materia de la escuela, como son: Ing. Oriana Barrios, Ing. Efraín Roca, Ing. Wilmer Sanz, Ing. Aída Pérez, Ing. Andrés Simone, Ing. Angel Villegas, Ing. Roberto Muñoz (Específicamente Cuando era preparador de Lógica Digital), Ing. Teddy Rojas, Ing. Francisco Arteaga.*

Gracias a todos y que Dios les dé mucha vida y salud.

IMPLEMENTACION DE UN SISTEMA DE MONITOREO DE TEMPERATURA PARA OBTENER MEDICIONES DEL EQUIPO EVAPORADOR-CONDENSADOR VERTICAL DEL LABORATORIO DE INGENIERIA QUÍMICA HACIENDO USO DEL MICROCONTROLADOR PIC18F2550

(Realizado por los bachilleres: Salas J. José G. y Vitriago R. Simón A. con el tutor Luis Llave)

RESUMEN

El presente trabajo de grado tiene como objetivo principal implementar un sistema de monitoreo de temperatura en el equipo evaporador-condensador vertical del laboratorio de ingeniería química haciendo uso del microcontrolador PIC18F2550. La problemática existente en la planta se refiere al procedimiento de medición que se poseen en el laboratorio de ingeniería química (LIQ), ya que se utiliza sólo una termocupla tipo k, lo que hace que el proceso sea totalmente manual y por lo tanto lento. Como solución a ésta problemática se creó el sistema SMT-Cy-Gus, el cual está formado principalmente por una Tarjeta de Adquisición de Datos (TAD) la cual permite medir la temperatura por medio de sensores de temperatura 1-Wire® (DS1820) hasta en 15 puntos alrededor del equipo evaporador vertical, y visualizar el valor de cada uno de ellos en tiempo real en una pantalla LCD de 16X2. Además de la TAD el sistema consta de un software realizado en el entorno de programación orientado a objetos (POO) Microsoft Visual C# 2008, en el cual es posible visualizar tanto los valores de temperatura medidos como el serial único de cada sensor 1-Wire®, el software tiene la opción de calcular temperatura promedio, almacenar valores en un archivo de texto y mostrar gráfica de perfil de temperatura.

Se realizaron pruebas al sistema SMT-Cy-Gus en el equipo evaporador-condensador vertical, obteniéndose resultados muy cercanos a la medición con la termocupla tipo k, considerando así la posibilidad de sustituir el uso de la termocupla por el sistema creado.

Palabras claves: Microcontrolador, PIC18F2550, POO, Sensor, TAD, USB, 1-Wire®.

INTRODUCCIÓN

El presente trabajo de grado tiene como finalidad describir el proceso llevado a cabo para lograr la implementar un sistema de monitoreo de temperatura en el equipo evaporador-condensador vertical del laboratorio de ingeniería química haciendo uso del microcontrolador PIC18F2550. Está dividido en 5 capítulos con el siguiente contenido:

El Capítulo I titulado “Planteamiento del problema”, en donde se plantea la problemática en la cual se enfoca el presente trabajo de grado, así como su debida justificación. Además, se fijan los objetivos que se esperan alcanzar al culminar la investigación.

El Capítulo II que lleva por nombre “Marco Teórico, reúne las bases teóricas necesarias para el entendimiento y realización del trabajo, así como una recopilación de antecedentes a la investigación.

El Capítulo III, “Marco Metodológico”, ubica el trabajo de grado en los tipos de investigación propuestos por el “Manual de Trabajos de Grado” de la UPEL, y expone los pasos a seguir para llevar a cabo el presente proyecto.

El Capítulo IV titulado “Análisis de los resultados” describe en detalle las condiciones de diseño, funcionamiento y manejo de los dispositivos, el análisis de los códigos de programación y la presentación de resultados obtenidos al cumplir con los objetivos planteados en el proyecto.

El Capitulo V, “Conclusiones y Recomendaciones” finaliza el trabajo de grado aportando una serie de conclusiones y recomendaciones producto de la realización del mismo, para que de esta manera sea posible un análisis retrospectivo del proyecto y se abra la posibilidad de continuar y mejorar éste.

ÍNDICE GENERAL

Pág.

ÍNDICE DE CONTENIDO

Pág.

ÍNDICE DE FIGURAS

CAPITULO I **Pág.**

CAPITULO II **Pág.**

CAPITULO IV **Pág.**

ÍNDICE DE TABLAS

CAPITULO II **Pág.**

CAPITULO IV **Pág.**

CAPÍTULO V:

CONCLUSIONES Y RECOMENDACIONES

CAPÍTULO I

PLANTEAMIENTO DEL PROBLEMA

1.1 Formulación del problema

Actualmente en la Escuela de Ingeniería Química de la Universidad de Carabobo, se imparten las materias Fenómenos de transporte I, Fenómenos de transporte II y Operaciones Unitarias, éstas disponen de un laboratorio, “Laboratorio de Ingeniería Química (LIQ)”, que cumple con la función de proveer el espacio físico necesario para desarrollar en forma práctica, experiencias dirigidas a impartir conocimientos a los estudiantes de dicha área.

El principal objetivo de este laboratorio es poner en práctica los conocimientos teóricos adquiridos por los estudiantes en las diversas asignaturas relacionadas con: estudio de calderas, evaporadores, condensadores, bombas centrífugas, medidores de flujo, pérdidas de energía, aletas y aislantes. Además se manejan criterios de selección de dispositivos tales como: válvulas, trampas de vapor, indicadores de temperatura, indicadores de presión, indicadores de nivel, termocuplas, rotámetros, entre otros.

Para poder realizar lo expuesto anteriormente el laboratorio cuenta con diversos equipos, entre los que se encuentran: equipo de aletas y aislantes, equipo de bombas (bomba centrífuga y bomba móvil), equipo evaporador-condensador vertical, equipo de pérdidas de energía, equipo medidor de flujo y torre de relleno.

Según opinión del personal técnico del laboratorio de ingeniería química: “El equipo evaporador-condensador vertical presenta un proceso de medición de temperatura lento y totalmente manual, éste se realiza en tres puntos obteniendo así un error, sabiendo que al tener un mayor número de lecturas, el cálculo del promedio tendrá menor desviación del valor exacto”.

Actualmente el proceso se lleva a cabo mediante solo una termocupla tipo K que poseen en el laboratorio, por lo que no es factible realizar la cantidad de lecturas necesarias para obtener un valor cercano al real, ya que representaría un alto gasto físico y pérdida de tiempo, para el estudiante, técnico o profesor.

Por otro lado, en el laboratorio no cuentan con la cantidad de termocuplas necesarias para colocar cada una en puntos fijos, debido a que las mismas son de alto costo, por encima de los 2000Bs/cu. Se puede notar que si el dispositivo medidor de temperatura, en este caso, termocupla, estuviese siempre fijo en los puntos donde se requieren medir la temperatura, se podría apreciar el valor de temperatura en cualquier momento.

En virtud a todo lo expuesto anteriormente, se desea automatizar y mejorar el proceso de medición de temperatura de pared en el evaporador vertical, así como también, la obtención de información acerca de los valores de temperatura en el equipo y realizar un perfil de temperatura eficiente.

Es por ello que se decidió crear un sistema de monitoreo de temperatura, capaz de leer la temperatura de pared hasta en 100 puntos del evaporador vertical, considerando que cada punto de medición corresponde a un sensor digital de temperatura 1-Wire®, el cual posee un costo de 50Bs/cu, según la empresa PLUS ELECTRONICS, C.A. (http:\\electrónica.com.ve). En la figura 1 se presenta un bosquejo de la propuesta.

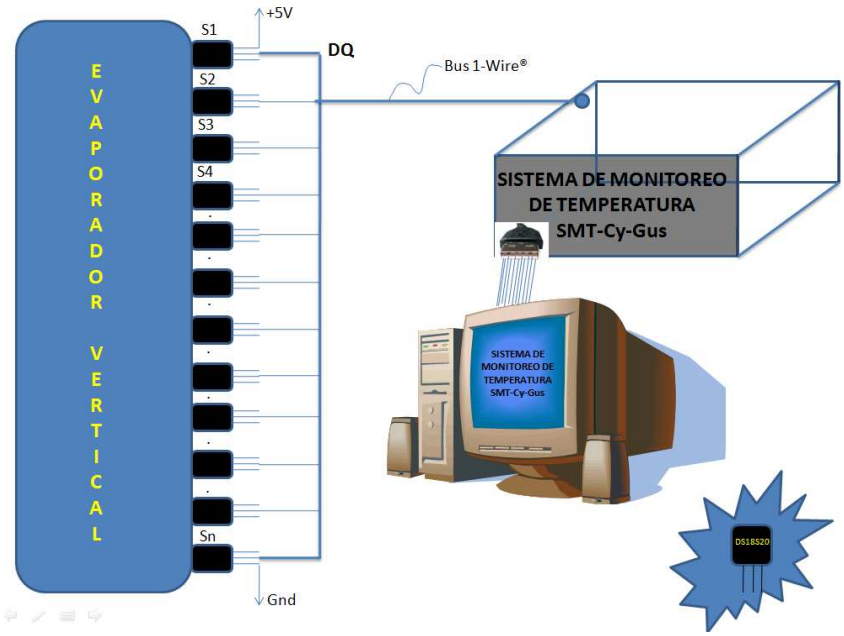


Figura 1. Representación gráfica del Sistema de Monitoreo de Temperatura SMT-Cy-Gus

Fuente: Elaboración Propia

1.2 Objetivo General

Implementar un sistema de monitoreo de temperatura en el equipo evaporador-condensador vertical del laboratorio de ingeniería química haciendo uso del microcontrolador PIC18F2550.

1.3 Objetivos Específicos

- Evaluar las características actuales del proceso de medición de temperatura en el equipo para determinar las debilidades presentes en el mismo.
- Seleccionar el sensor de temperatura 1-Wire® a emplear para realizar las mediciones.
- Diseñar una tarjeta de adquisición de datos con comunicación USB, en donde se reciban los valores provenientes de los sensores de temperatura 1-wire®.

- Diseñar un software en visual studio C# 2008 que permita mostrar los datos recibidos por la tarjeta de adquisición de datos, mediante una gráfica de perfil de temperatura.
- Realizar un conjunto de pruebas para validar el funcionamiento del sistema.

1.4 Justificación

Con la implementación del presente sistema automatizado, el desarrollo de las prácticas en el laboratorio de ingeniería química será más dinámico, sencillo, y retornará resultados más cercanos a los esperados teóricamente.

Además, debido a la utilización de sensores de temperatura 1-Wire® se obtendrán las siguientes ventajas:

- Por su pequeño tamaño, se podrán conectar hasta 100 sensores en el equipo en el cual se realizarán las mediciones, sabiendo que cada sensor representa un punto de medición.
- Cada sensor digital de temperatura será instalado firmemente en cada punto del equipo evaporador-condensador vertical, colocados equidistantes con la idea de ocupar toda la longitud del equipo.
- El Costo de cada sensor es 40 veces más económico que un termómetro digital.

El presente estudio representa un aporte a la línea de investigación y al desarrollo tecnológico en el área de conocimiento.

1.5 Alcance

La presente investigación pretende resolver una problemática con solución de ingeniería básica, haciendo uso de herramientas poderosas como lo son la tecnología 1–Wire®, tecnología USB, entorno de programación orientado a objetos Visual Studio C# 2008.

Algunas funcionalidades del software son:

- Permitir la visualización en tiempo real del valor cada sensor
- Visualizar serial único de cada dispositivo
- Crear la gráfica del perfil de temperatura
- Almacenar valores de temperatura en un archivo de texto.

CAPÍTULO II
MARCO TEÓRICO

2.1 ANTECEDENTES

La investigación aquí propuesta tiene como basamento teórico los distintos trabajos especiales de grado realizados en la Universidad de Carabobo y otras instituciones, a través de los cuales se han estudiado las diferentes herramientas en el área de control e instrumentación digital. Estos se nombran a continuación:

- "Desarrollo de Practicas para el laboratorio de microprocesadores utilizando microcontroladores" (2006) realizado por los bachilleres Lemus Henry J y Istillarte Carlos L. Para optar al título de Ingeniero Electricista. En este proyecto se explica el manejo de los microcontroladores, así como aplicaciones típicas [1].

- "Desarrollo de instrumentación virtual con fines didácticos empleando controles ActiveX para la utilización de circuitos integrados con capacidad de comunicación 1-Wire®" (2007) realizado por Castro Zambrano José R. y Díaz Mattern Jessica J. Para optar al título de Ingeniero Electrónico. En este proyecto se desarrollaron cuatro controles ActiveX con capacidad de comunicación 1-Wire®, fueron programados en Visual Basic 6.0, y a su vez empleado como una herramienta didáctica, capaz de leer, manipular o accionar variables desde el instrumento virtual creado en la PC [2].

- "Incorporación de la red 1-Wire® a los Procesos de automatización industrial, basados en el estándar "OPC" OLE para el control de procesos" (2008) realizado por Ing. José M. Rodríguez S. Para optar al título de magister en Ingeniería Eléctrica. La presente investigación permitió dar a conocer a los dispositivos 1-Wire® a nivel industrial, así como también el nuevo protocolo OPC (OLE para el control de procesos), demostrando su gran versatilidad, seguridad, robustez y alto rendimiento. Se utilizó el lenguaje de programación Delphi® para Windows 32 bits. Además se creó un Servidor de datos OPC, el cual

actualmente está siendo usado en la Universidad de Carabobo en materias referentes a instrumentación y control [3].

- "Construcción de un electroencefalógrafo portátil" (2009) realizado por los bachilleres Eduardo Antonio Rojas Nastrucci y José Antonio Vargas Peiko. Para optar al título de Ingeniero Electricista. El presente trabajo de grado consiste en la realización de un electroencefalógrafo portátil de un canal y con conectividad USB, el cual da a conocer el funcionamiento y características del protocolo USB [4].

- "Desarrollo de un tacómetro digital con comunicación USB para los motores del laboratorio de máquinas eléctricas de la Facultad de Ingeniería en la Universidad de Carabobo" (2009) realizado por los bachilleres Hernández Humberto y Pulido Gregory. Para optar al título de Ingeniero Electricista. En este proyecto se presenta información de programación de microcontroladores, comunicación serial y USB [5].

- "Desarrollo de un sistema de instrumentación para el proyecto BAJA de la organización SAEUC Venezuela" (2009) realizado por los bachilleres Cárdenas R. Jorge G y Márquez B. Navier E. Para optar al título de Ingeniero Electricista. El presente trabajo grado consiste en el diseño e implementación de un tablero de medida de velocidad, nivel de combustible y revoluciones del motor cuyo control se realizó mediante la programación en Mikrobasic de un PIC [6].

- "Diseño y construcción de un módulo para pantallas a base de LEDS RGB, que permita la comunicación vía USB para la visualización de datos transmitidos desde una computadora" (2010) realizado por los bachilleres Fernando Botello y José María Rojas. Para optar al título de Ingeniero Electricista. En este proyecto se explica detalladamente la programación USB tanto en el microcontrolador PIC como en la PC, haciendo uso del software Easy Hid Wizard el cual genera ambos códigos automáticamente [7].

2.2 BASES TEÓRICAS

2.2.1 Instrumentación Virtual

Muchas veces la obtención de una medida requiere la utilización de varios instrumentos, unos generan estímulos sobre el dispositivo que se pretende medir y otros recogen la respuesta a estos estímulos. Este conjunto de instrumentos, que hace posible la realización de la medida, recibe el nombre de sistema de instrumentación. Todo sistema de instrumentación consta de unos instrumentos, un sistema de interconexión de éstos y un controlador inteligente que gestiona el funcionamiento de todo el sistema y da las órdenes para que una medida se obtenga correctamente.

El concepto de instrumentación virtual nace a partir del uso de la computadora personal, como forma de reemplazar instrumentos de medición ya sean analógicos o digitales que se emplean en físico, por instrumentos creados por software, permitiendo a los usuarios interactuar con la computadora como si estuviesen manipulando un instrumento real o en físico.

El concepto de instrumentación virtual o digital implica adquisición de señales, el procesamiento, análisis, almacenamiento, distribución y despliegue de los datos e información relacionados con la medición de una o varias señales, interfaz hombre-máquina, visualización, monitoreo y supervisión remota del proceso, la comunicación con otros equipos, enmarcados en funciones de control.

Contemporáneamente, con el ingreso y avance de la computadora personal, los instrumentos adquieren el máximo potencial debido a las prestaciones de los sistemas de la actualidad. De esta forma se abren camino dos nuevos conceptos muy importantes: La instrumentación virtual y los sistemas de adquisición de datos.

La instrumentación virtual es un concepto introducido por la compañía National Instruments en el año 2001. En el año de 1983, Truchard y Kodosky, de National Instruments, decidieron enfrentar el problema de crear un software que permitiera utilizar el computador personal (PC) como instrumento para realizar mediciones. Tres años fueron necesarios para crear la primera versión del software que permitió, de una manera gráfica y sencilla, diseñar un instrumento en la PC. De esta manera surge el concepto de instrumento virtual (IV), definido como, "un instrumento que no es real, se ejecuta en una computadora y tiene sus funciones definidas por software." (National Instruments, 2001). A este software le dio el nombre de *Laboratory Virtual Instrument Engineering Workbench*, más comúnmente conocido por las siglas LabVIEW.

2.2.2 Tecnología 1-Wire®

Dallas-Maxim™ Semiconductor, uno de los fabricantes de semiconductores más grande del mundo, desarrolló una poderosa tecnología denominada 1-Wire®, la cual, utiliza un sólo conductor más el retorno o común para efectuar las comunicaciones y la transmisión de energía entre un dispositivo maestro y múltiples esclavos. Una red digital de dispositivos 1-Wire® está conformada por un maestro y uno o más esclavos que poseen un único terminal de datos de tipo open drain al que se conecta una resistencia de pull-up alimentada por +5v nominales. Una de las características de ésta tecnología, es que cada dispositivo esclavo tiene una única e irrepetible identificación codificada en la memoria ROM, lo cual, facilita las acciones típicas de búsqueda en una red, el direccionamiento y transferencia de información entre múltiples dispositivos.

2.2.2.1 El Bus 1-Wire®

El bus 1-Wire®, permite realizar una comunicación serial asincrónica entre un dispositivo maestro y uno o varios dispositivos esclavos, utilizando un único terminal de

entrada/salida (E/S), la comunicación se realiza a través de Microcontroladores y/o interfaz de comunicación a un computador ó PC.

Algunas características de este bus, según lo recopilado por (José M. Rodríguez S., 2006), son:

- Utiliza niveles de alimentación compatibles con las tecnologías CMOS/TTL en un rango de operación desde 2.8 Vdc hasta 6 Vdc.
- El dispositivo maestro y los dispositivos esclavos transmiten información en forma bi-direccional, pero, sólo en una dirección a la vez. De ésta manera la comunicación se realiza en forma "*half dúplex*".
- Toda la información transmitida a través del bus, es leída o escrita comenzando por el bit menos significativo (LSB).
- No se requiere del uso de una señal de reloj para realizar la sincronización, ya que, cada componente 1-Wire® posee un oscilador interno que se sincroniza con el del maestro cada vez que en la línea de datos aparece un flanco de bajada ó transición beta (de nivel lógico alto a nivel lógico bajo).
- La alimentación de los esclavos se realiza utilizando el voltaje propio del bus. Para ello, cada circuito esclavo posee un rectificador de media onda y un capacitor. Durante los períodos en los cuales no se efectúa comunicación, la línea de datos se encuentra en estado alto debido a una resistencia de PullUp; en esa condición, un diodo entra en conducción y carga a un capacitor. Cuando el voltaje de la red cae por debajo de la tensión del capacitor, el diodo se polariza en inverso evitando que el capacitor se descargue. La carga que queda almacenada en el capacitor es la que finalmente alimenta al circuito integrado conectado como esclavo.
- La red de dispositivos 1-Wire®, en general, tiene capacidad para manejar hasta 100 dispositivos esclavos distribuidos a lo largo de una distancia de 200 metros.

- Todas las tensiones presentes en el bus mayores a 2.2 Vdc, son consideradas un (1) lógico mientras que como un (0) lógico se interpreta cualquier voltaje menor de 0.8 Vdc.
- La transferencia de información se realiza a una velocidad de 16.3 Kbps en modo "Estándar" y hasta a 142 Kbps en modo "Overdrive".

2.2.2.2 Elementos que componen una red 1-Wire®

La red 1-Wire® utiliza una topología maestro-esclavo, en la cual, existe un único dispositivo maestro y uno o más esclavos. Según la nota de aplicación 1796 de la empresa Maxim (Dallas-Maxim, 2002), esta red se compone de tres elementos principales: (1) un maestro de bus con un software de control como el visor de dispositivos TMEX iButton®, (2) el cableado y los conectores asociados, y (3) los dispositivos 1-Wire®. Por otra parte, la red 1-Wire®, brinda un estricto control sobre la comunicación debido a que ningún nodo de interconexión está autorizado a transmitir información a menos que se lo indique el dispositivo maestro, además, no está permitida la comunicación directa entre los esclavos. La implementación del maestro 1-Wire® puede hacerse a través de diferentes medios, siendo los más utilizados un computador, un microcontrolador o un circuito integrado de aplicación específica ASIC. En cada caso, se requiere de determinados recursos hardware. Para la implementación basada en un computador, se necesita un adaptador de puerto de comunicaciones, denominado "Host Adapter". Este dispositivo permite establecer la conexión física entre el puerto de computador y la red 1-Wire®. Existen adaptadores de re para los puertos paralelo, serial RS-232C y USB, siendo estos dos últimos los más utilizados.

2.2.2.3 Protocolo de comunicaciones 1-Wire®

Se puede describir al protocolo de comunicaciones 1-Wire® como una secuencia de transacciones de información, la cual, se desarrolla según los siguientes pasos: (1)

Inicialización, (2) Comandos y funciones de ROM, (3) Comandos y funciones de control y memoria, (4) Transferencia de bytes o datos (Parallax, 2004). A continuación se describe cada secuencia.

Inicialización: Las comunicaciones en el bus 1-Wire® comienzan con una secuencia de un pulso de reset y presencia. El pulso de reset provee una forma conveniente de iniciar las comunicaciones, ya que, con él se sincronizan los dispositivos esclavos presentes en el bus. Un *reset* es un pulso que genera el maestro al colocar la línea de datos en estado lógico bajo por aproximadamente 480 µs. una vez liberado, el bus retorna al nivel alto y luego de un tiempo comprendido entre 15 a 60 µs, los dispositivos esclavos transmitirán un pulso de presencia. Este consiste en forzar la línea de datos a nivel bajo durante un tiempo entre 60 a 240 µs. El maestro, esperará por los pulsos de presencia con el bus en estado alto, a través de la resistencia de PullUp, por un tiempo de al menos 240 µs. Cuando se trabaja al bus a velocidad estándar, los pulsos de reset y presencia tendrán las características como las mostradas en la figura 2.

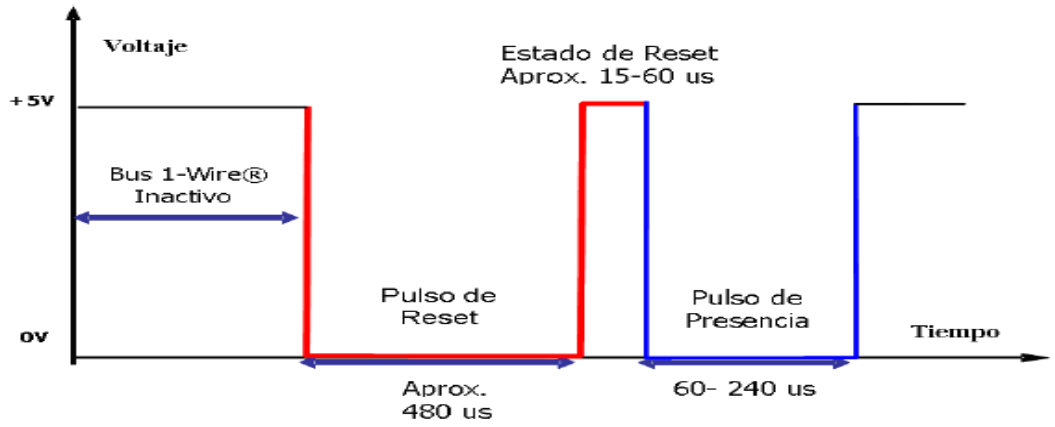


Figura 2: Representación gráfica de un pulso de reset y de presencia

Fuente: José M. Rodríguez S, Comunicación 1-Wire®, 2008 [3].

Comandos y funciones de ROM: Una vez que el dispositivo maestro recibe el pulso de presencia de los dispositivos esclavos, puede enviar un comando de ROM. Los comandos de ROM son comunes a todos los dispositivos 1-Wire® y se relacionan con la

búsqueda, lectura y utilización de la dirección de 64 bits que identifica a esclavos. La tabla 1, muestra los comandos de ROM comúnmente utilizados con los dispositivos 1-Wire®.

Comandos y funciones de control y memoria: Son funciones propias de cada dispositivo 1-Wire®. Incluyen comandos para leer/escribir en localidades de memoria, leer memorias *scratchpad*, controlar el inicio de la conversión de un ADC, iniciar la medición de una temperatura o manipular el estado de un bit de salida, entre otros. Cada dispositivo define un conjunto de comandos propios a su funcionalidad.

Transferencia de datos: La lectura y escritura de datos en el bus 1-Wire® se hace por medio de Slot[38] la generación de éstos es responsabilidad del dispositivo maestro. Cuando el maestro lee información del bus, debe forzar la línea de datos a un estado bajo durante al menos 4 µs y esperar adicionalmente 15 µs para leer el estado presente en la línea. El estado lógico de la línea en ese momento, estará determinado por el dispositivo esclavo. La figura 3 (a), muestra el proceso de lectura de un bit (slot de lectura), mientras que la figura 3 (b), muestra el proceso de escritura de un bit (slot de escritura).

COMANDO	VALOR	DESCRIPCIÓN
Read ROM	$33	Lee la identificación de 64 bits del dispositivo. Puede ser utilizado solamente cuando existe un sólo dispositivo esclavo presente en la red 1-Wire®.
Match ROM	$55	Este comando, seguido de una identificación de 64 bits, permite seleccionar a un dispositivo esclavo en particular.
Skip ROM	$CC	Direcciona a un dispositivo sin necesidad de conocer la identificación, puede ser utilizado solamente cuando existe un sólo esclavo conectado en la red 1-Wire®.
Search ROM	$F0	Lee los 64 bits de identificación de los dispositivos esclavos conectados en la red. Se utiliza un proceso de eliminación para distinguir a cada dispositivo conectado.

Tabla 1. Comandos de ROM comúnmente utilizados en chips 1-Wire®

Fuente: (Parallax, 2004)

2.2.2.4 Componentes que integran la tecnología 1-Wire®

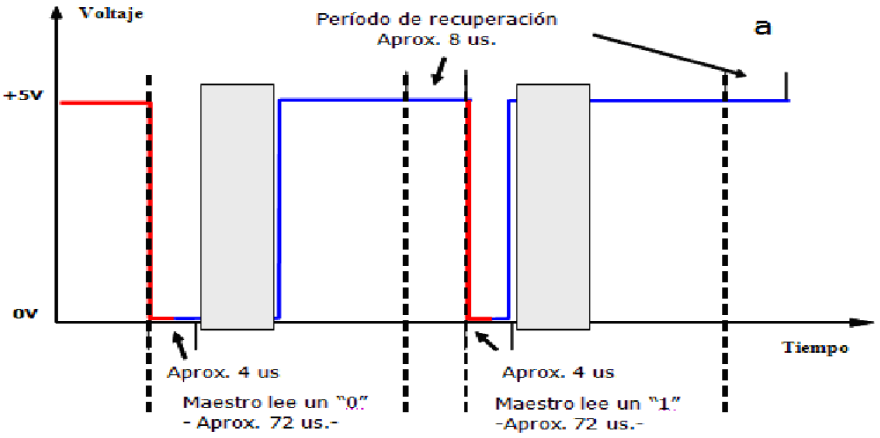
Actualmente, existen alrededor de treinta (30) diferentes circuitos integrados que trabajan con ésta tecnología (Dallas Semiconductor/Maxim, 2005). La tabla 2.2, muestra algunos de los componente más importantes, los ibuttons® aparecen en la tabla entre paréntesis ( ).

Código Hex[21]	CÓDIGO DEL PRODUCTO	DESCRIPCIÓN
01	(DS1990A)*, DS2401	Serial de identificación 1-Wire®.
04	(DS1994), DS2404	Reloj y alarmas con memoria de 4K NVRAM.
05	DS2405	Interruptor simple N.O. y direccionable (SPST).
06	(DS1993)	4K NVRAM de memoria no volátil.
08	(DS 1992)	1K NVRAM de memoria no volátil.
09	(DS1982), DS2502	1K EPROM de memoria EPROM.
0B	(DS1985), DS2505	16K EPROM de memoria EEPROM.
0C	(DS1996)	64K a 256K NVRAM de memoria no volátil.
10	DS1820, 18S20	Medidor de temperatura con alarmas y disparos.
12	DS2406, DS2407	Interruptor doble N.O. con 1K EPROM.
1D	DS2423	4K NVRAM de memoria con contadores externos.
1F	DS2409	Acoplador de RED 1-Wire® de 2 canales.
20	DS2450	Cuatro convertidores A/D de 16 bits resolución.
21	(DS1921), (DS1921H),	Registradores "loggers" de temperatura.
22	DS1822	Medidor de temperatura digital.
23	(DS1973), DS2433	4K EEPROM de memoria con contadores.
24	(DS1904), DS2415	Reloj de tiempo real.
26	DS2438	Convertidor A/D dual con medición de Temp.
27	DS2417	Reloj de tiempo real con interrupciones externas.
28	DS 18B20	Medidor de temperatura con resolución ajustable.
2C	DS2890	Potenciómetro digital de un canal.
30	DS2760	Medidor de temp. y un canal A/D precisión mV.

Tabla 2. Circuitos Integrados con Tecnología 1-Wire®

Fuente: (Dallas Semiconductor/Maxim, 2005)

3(a)



3 (b)

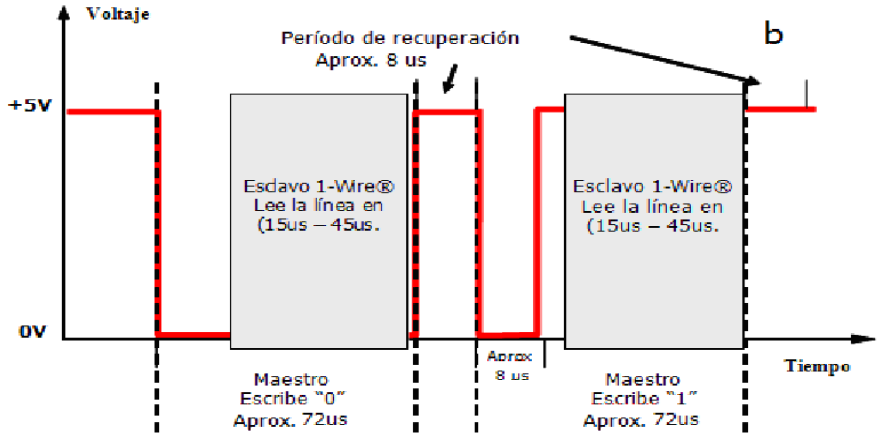


Figura 3: Transmisión de bits bajo el protocolo 1-Wire®

(a) Lectura de un (0) y un (1) desde el maestro.

(b) Escritura de un (0) y un (1) desde el maestro.

Fuente: Modificado de (Parallax, 2004)

2.2.3 Comparación del Bus 1-Wire® con tecnologías similares

En esta sección, se realiza una breve comparación entre los buses seriales que poseen características similares a las del 1-Wire® a fin de resaltar, tanto las ventajas como las desventajas de utilizar esta tecnología, para ello, se presenta una matriz de comparación entre diferentes buses de comunicación serial, ver tabla 2.3 y tabla 2.4. En esta matriz, cada casilla agrupa a buses con características similares de acuerdo a: los métodos de direccionamiento y sincronización, el modo de transmisión y requerimientos de adaptación de impedancia "impedance matching" de las líneas de transmisión. Los espacios dejados en blanco indican que no existe ningún bus serial conocido que cumpla con la intersección de las características mostradas.

<table>
<tr><td></td><td colspan="3">DIRECCIONAMIENTO (SELECCIÓN)</td><td></td></tr>
<tr><td>SINCRONIZACIÓN</td><td colspan="2">POR PROTOCOLO</td><td>LÍNEA DE HABILITACIÓN</td><td>ADAPTADOR IMPEDANCIA</td></tr>
<tr><td rowspan="2">AUTO-SINCRONIZADA</td><td>1-Wire®, LIN bus, SensorPath™</td><td></td><td></td><td>NO REQUERIDA</td></tr>
<tr><td></td><td>RS-485, LVDS, CAN, USB 2.0, FireWire®</td><td></td><td rowspan="2">REQUERIDA</td></tr>
<tr><td rowspan="2">LÍNEA DE RELOJ</td><td></td><td></td><td></td></tr>
<tr><td>I²C, SMBus™</td><td></td><td>SPI™, MICROWIRE™</td><td>NO REQUERIDA</td></tr>
<tr><td></td><td>SINGLE ENDED</td><td>DIFERENCIAL</td><td>SINGLE ENDED</td><td></td></tr>
<tr><td></td><td colspan="3">MODO DE TRANSMISIÓN</td><td></td></tr>
</table>

Tabla 3. Comparación entre diferentes buses de comunicación serial.

Fuente: (Dallas Semiconductor, 2006)

INDICADOR	1-Wire®	LIN Bus	Sensor Path™	I²C
Extensión de la Red	PCB[29] o expandido hasta 200m	40 m	Dentro del PCB. Aprox. 1 Metro.	Dentro del PCB Aprox. 1 metro.
Interfaz de Hardware Soportada	Controladores para PC por puerto paralelo, serial y/o USB. Controlador para I²C Terminales de I/O de uso general en microcontroladores	Terminales de I/O de uso general en microcontroladores	Circuitos integrado tecnologia Super I/O. Terminales de I/O de uso general en microcontroladores	Terminales de I/O de uso general en microcontroladores Por hardware especializado en algunos microcontroladores
Soporte de Software para desarrollo	SDKs gratuitos para varias plataformas incluyendo microcontroladores	Gratuito para microcontroladores Freescale™	No disponible	Gratuito para microcontroladores
Fuente de alimentación	Por la línea de datos Fuente local	Por la línea de datos	Fuente local	Fuente local
Velocidad de comunicación máxima	Estándar 15 kbps Alta velocidad 125 kbps	Hasta 20 kbps	Dependiente de los datos 20 kbps	Estándar 100 kbps Rápida 400 kbps Alta Velocidad 3.4 Mbps
Reconocimiento de dispositivos	Función de búsqueda de seriales	No aplica, direccionamiento por mensaje	No soportado	Direcciones de esclavos en el firmware
Tipos de dispositivos	Gran variedad incluyendo seriales de ID, instrumentación, memorias seguras	Limitado a funciones de aplicaciones automotrices	Limitado a sensores de temperatura y convertidores ADC[3]	Amplia gama de productos incluyendo memorias, reloj de tiempo real, convertidores ADC[3]
Detección de errores	Soportada mediante algoritmo CRC[9] de 8 y 16 bits	Soportada mediante "Check – Sum"	Verificación de 1 bit de paridad	No soportada
Formato de direccionamiento	56 bits con serial único de 48 bits, identificador de familia de 8 bit y código CRC[9]	Identificador de mensajes de 8 bits incluyendo 2 bits de paridad	Dirección de 9 bits, 3 para identificación del esclavo y 6 para la operación a realizar	Dirección de 7 bits, un bit adicional para tipo de operación

Tabla 4. Características de los buses de comunicación serial similares a 1-Wire®

Fuente: Modificado de (Dallas Semiconductor, 2006) y (Dallas Semiconductor, 2004)

En la tabla 4, se observa que:

- El bus SensorPath® a diferencia de los demás está limitado físicamente a la tarjeta de circuito impreso PCB que la contiene.

- Bajo ciertas condiciones y utilizando un apropiado hardware y software controlador de red, el alcance del bus 1-Wire® puede ser expandido significativamente.
- Los buses 1-Wire® y LIN Bus permiten la alimentación de los eslavos por la propia línea de datos mediante un método llamado “alimentación parásita”, esto evita la necesidad de instalar fuentes de alimentación locales en cada dispositivo conectado al bus.
- La característica de reconocimiento de la red 1-Wire® permite al maestro identificar los números, tipos y direcciones de los dispositivos eslavos en la red, permitiendo una re-configuración dinámica de la misma.
- Dentro del grupo simple de los buses de bajo costo, los buses 1-Wire® e I^2C poseen mayor cantidad de dispositivos comparados con el LIN Bus y el SensorPath®.
- 1-Wire® ofrece soporte tanto para desarrollar aplicaciones embebidas basadas en Microcontroladores como para aplicaciones basadas en el uso del computador.
- El bus I^2C requiere de una línea adicional de reloj y una fuente de poder externa, lo que le resta versatilidad.
- El mecanismo de direccionamiento e identificación única de 64 bits de los dispositivos 1-Wire® supera ampliamente a los utilizados en los otros buses analizados. El identificador de carácter físico se encuentra dentro del circuito integrado y mediante el código de familia se puede determinar el tipo de dispositivo presente, por consiguiente, el tipo de servicio que provee. La dirección del dispositivo posibilita la unicidad en el direccionamiento y por tanto la ubicación del mismo en la red, adicionalmente el código CRC, utilizado para la verificación de errores en la comunicación digital.
- La principal desventaja del 1-Wire® como red de comunicación es que presenta una velocidad inferior a otros buses de comunicación, por ejemplo, I^2C. Sin embargo, aventaja a éste en cuanto a costos de implementación, flexibilidad, seguridad y alcance físico.

2.2.4 API de programación 1-Wire®

Un API[4] "*Application Program Interface*" o interfaz de programa de aplicación, es la plataforma que proporciona el fabricante para acceder a los dispositivos y la red 1-Wire® desde un computador. Básicamente se definen como un conjunto de subprogramas o funciones de bajo nivel programadas en un entorno que depende tanto del lenguaje de programación como del sistema operativo utilizado, tal como se muestra en la figura 4.

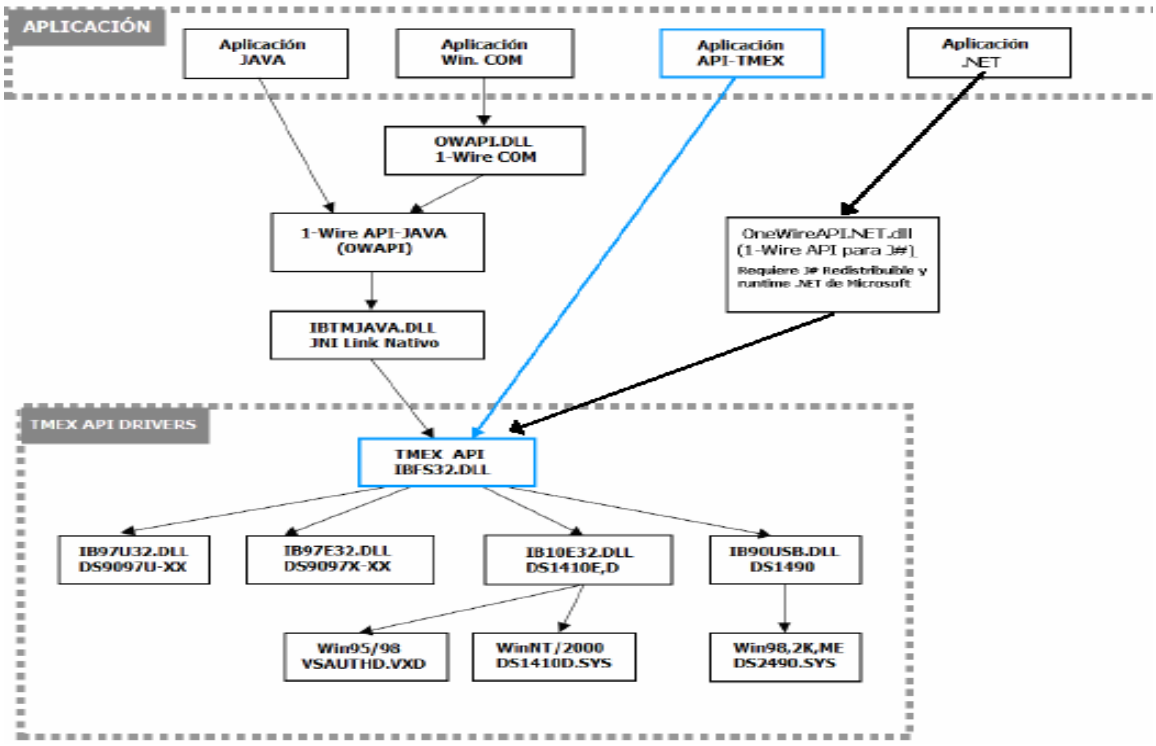


Figura 4. Librerías API requeridas para cada tipo de aplicación.

Fuente: (Dallas Semiconductor/Maxim, 2005)

Actualmente el fabricante de dispositivos 1-Wire®, dispone de cinco diferentes API, las cuales se describen a continuación:

- **1-Wire® Public Domain (PD):** Conjunto completo de códigos y funciones abiertas en lenguaje "C", las cuales soportan la conexión con un PC a través de un adaptador tipo serial denominado DS9097U ó equivalente.
- **1-Wire® API for JAVA (OWAPI):** Conjunto completo de códigos y funciones abiertas en lenguaje JAVA, los cuales, soportan la conexión con un PC a través de un adaptador tipo serial denominado DS9097U ó equivalente y casi la totalidad de los dispositivos 1-Wire®.
- **1-Wire® API.NET:** Conjunto completo de funciones independientes del lenguaje (DLL), para ser utilizada con la nueva plataforma de Microsoft .NET. En realidad el API.NET resulta de la compilación del "API for Java" realizado con la versión de J# de Microsoft.
- **1-Wire® API TMEX:** Conjunto completo de funciones independientes del lenguaje (DLL's), proveen soporte a los dispositivos y adaptadores de red 1-Wire® que trabajen bajo la plataforma Windows™ de 32 Bits, son al mismo tiempo, las funciones que se utilizarán para acceder a los diferentes dispositivos en la red 1-Wire®. El API TMEX, está diseñado para trabajar en aplicaciones multiprocesos-multitareas y es soportado por los sistemas operativos Windows™ 32 bits.

Debido a que la mayoría de los dispositivos 1-Wire® poseen memoria, las funciones de memoria entrada/salida (I/O) son tratadas como un grupo de funciones API aunque las mismas no se apliquen a todos los dispositivos.

2.2.5 Definición de ActiveX

Un objeto ActiveX se define como el que se adhiere al Modelo de Objeto Componente (Component Object Model COM), quien representa una arquitectura de programación que permite construir aplicaciones a partir de componentes de software, definido por Microsoft. Es usado en ambientes Windows™ por lenguajes de

programación Orientado a Objetos como Visual Basic, Visual C#, C++, Delphi entre otros, como también en la Web mediante el uso de Active Server Pages (ASP) y el lenguaje VBScript. La tecnología COM provee muchos beneficios, incluyendo integración más fácil, escalabilidad y reusabilidad. También ofrece la ventaja e independencia del lenguaje y compatibilidad entre distintas plataformas.

2.2.6 Características del objeto ActiveX

Un objeto que cumpla con este modelo tiene las siguientes características:

- Un objeto ActiveX está aplicado como código binario, por consiguiente, puede estar escrito en cualquier lenguaje fuente.
- El objeto está encapsulado en un archivo ejecutable o en una biblioteca de vínculo dinámico.
- El objeto contiene datos de dos tipos: datos de presentación, que se requieren para presentar la pantalla o para imprimir, y datos internos. Puede considerar los dos tipos de datos como propiedades que son privadas para el objeto.
- El objeto contiene también funciones para manipular sus datos.
- El objeto proporciona una interfaz estándar para que otros objetos se comuniquen con él.
- El objeto participa en la disposición en formación, proceso de pasar argumentos de funciones y valores de retorno entre procesos y máquinas.

2.2.7 Programación Orientada a Objetos (POO)

Un lenguaje se dice que está basado en objetos si soporta objetos como una característica fundamental del mismo. El elemento fundamental de la POO es, como su nombre lo indica, el **objeto** y se define como un conjunto complejo de datos y programas que poseen estructura y forman parte de una organización.

La programación orientada a objetos tiene como conceptos elementales los conceptos de objeto y clase.

2.2.7.1 Clase y Objeto

Una clase es un ente abstracto que permite declarar las propiedades y los métodos de objetos similares. Las clases presentan el estado de los objetos a los que representan mediante variables denominadas *atributos.* Cuando se crea un objeto el compilador crea en la memoria dinámica un espacio para tantas variables como atributos tenga la clase a la que pertenece el objeto.

Un lenguaje de programación orientado a objetos debe permitir al programador realizar definiciones de clases, y construir objetos a partir de esas clases.

Un objeto también se define como una entidad que posee sus características propias (propiedades) y un conjunto de acciones que es capaz de realizar (métodos) que pueden ser invocadas externamente. Un objeto además de un estado interno, presenta una interfaz para poder interactuar con el exterior.

Entonces, en la programación orientada a objetos, un programa no es más que una colección de objetos interactuando entre sí. Esa interacción se produce a través de mensajes. Un **mensaje** es un pedido al objeto de que ejecute uno de sus métodos y consta del nombre del objeto y el nombre del método a ejecutar.

2.2.7.2. Estructura de un Objeto

Un objeto se compone de: Relaciones, Propiedades y Métodos.

Las **relaciones** permiten que el objeto se inserte en la organización y están formadas esencialmente por punteros a otros objetos.

Las **propiedades** son elementos expuestos por el componente a fin de que el programador pueda personalizarlo, modificando sus atributos con el fin de adaptarlo a sus necesidades, por ejemplo, estableciendo colores, dimensiones, títulos, enlaces a bases de datos , conexiones con servidores, etc. Las propiedades e un objeto pueden ser heredadas a otros objetos que dependan de él dentro del código.

Los **métodos** son las operaciones que pueden realizarse sobre el objeto, que normalmente estarán incorporados en forma de programas (código) que el objeto es capaz de ejecutar y que también puede heredarse de una clase a otra.

2.2.7.3. Eventos

Un evento de un objeto es una señal generada por el objeto, ante un suceso concreto, y que la aplicación puede aprovechar para ejecutar alguna acción. Dicho suceso puede ser externo, como la pulsación de una tecla, o bien interno como la recepción de una señal por parte del Sistema Operativo.

2.2.7.4. Beneficios que se obtienen del desarrollo con POO

Entre las preocupaciones que han existido desde el comienzo en el desarrollo de software se encuentran la falta de portabilidad del código y reusabilidad, código que es difícil de modificar, ciclos de desarrollo largos y mantenimiento del software.

Todos estos problemas aún no han sido solucionados en forma completa. Pero como los objetos son portables (teóricamente) y la herencia permite la reusabilidad del código orientado a objetos, es más sencillo modificar un código existente porque los objetos no interaccionan excepto a través de mensaje; en consecuencia un cambio en la

codificación de un objeto no afectará la operación con otro objeto siempre que los métodos respectivos permanezcan intactos.

La introducción de tecnología de objetos como una herramienta conceptual para analizar, diseñar e implementar aplicaciones, permite obtener aplicaciones más modificables, fácilmente extensibles y a partir de componentes reusables. Esta reusabilidad del código disminuye el tiempo que se utiliza en el desarrollo y hace que el desarrollo del software sea más intuitivo porque toda persona piensa naturalmente en términos de objetos más que en términos de algoritmos de software.

2.2.8 Entorno de Programación de visual Studio C#

Visual C# es la versión del lenguaje de programación C orientado a objetos, posee características similares a los lenguajes de programación: Java y C++. Al igual que visual Basic está orientado al sistema operativo de Windows™ con todas sus características (manejo de ventanas, controles, íconos, gráficos, funciones de la API, la apariencia con respecto a visual Basic 6.0, es muy parecida.

El lenguaje que maneja este entorno es el lenguaje "C", se ha comprobado que es el lenguaje con gran nivel de dificultad con respecto al BASIC y borland. Con visual C# se pueden crear aplicaciones (*.exe), librerías dinámicas (*dll), controles ActiveX (*ocx), entre otras cosas. Posee la opción de conectarse a base de datos, mediante programas como SQL server, Acces.

Se trabajará con la versión express descargada de la página de Microsoft (Visual Studio 2008), como se observa en la figura 2.5

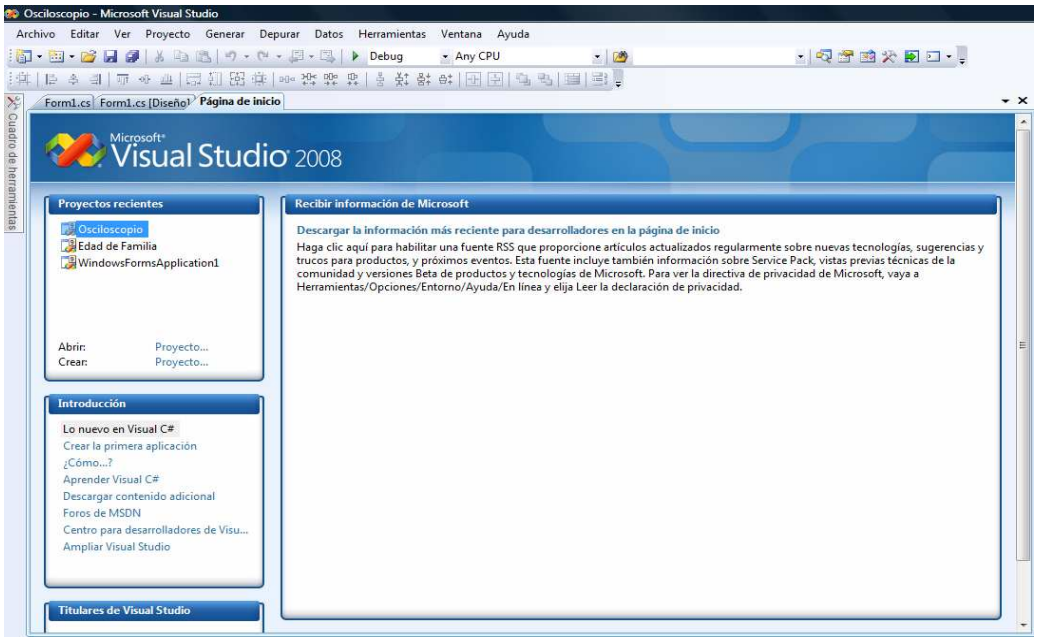


Figura 5. Ventana principal al abrir Visual Studio 2008

Fuente: Elaboración Propia

Para la creación de un proyecto en el entorno C#, se realiza lo siguiente:

Figura 6. Cómo crear un nuevo proyecto

Fuente: Elaboración Propia

Al hacer click izquierdo en el lugar señalado, se abrirá la ventana en donde se le colocará el nombre del nuevo proyecto.

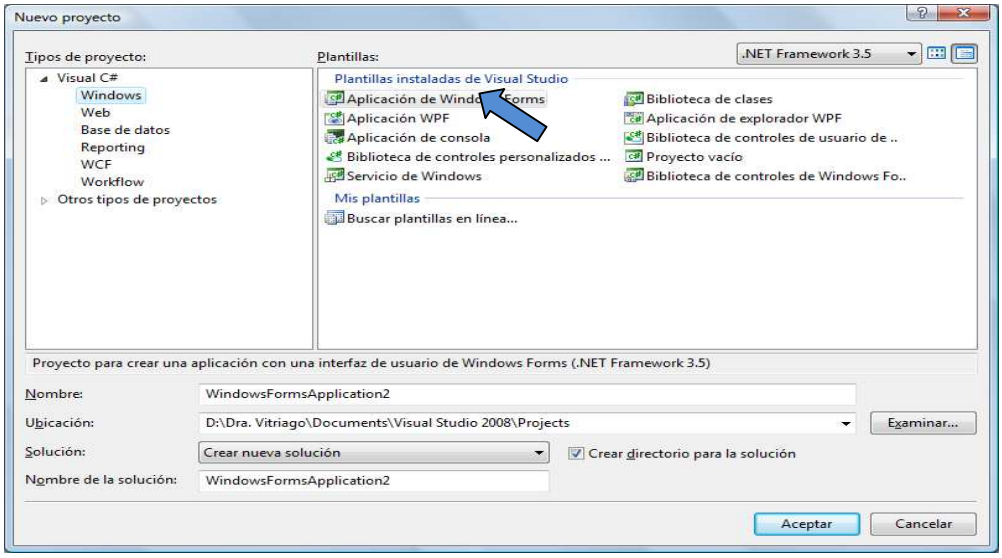


Figura 7. Ventana para darle nombre al nuevo proyecto

Fuente: Elaboración Propia

Allí se introducirá el nombre deseado al proyecto, colocaremos “Prueba 1”.

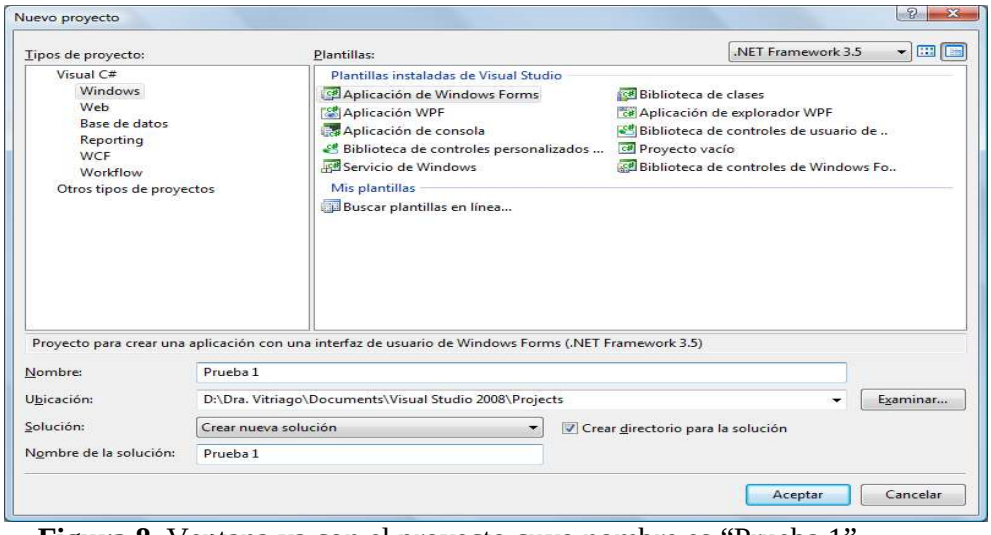


Figura 8. Ventana ya con el proyecto cuyo nombre es “Prueba 1”

Fuente: Elaboración Propia

Al presionar aceptar, se observa el espacio de trabajo

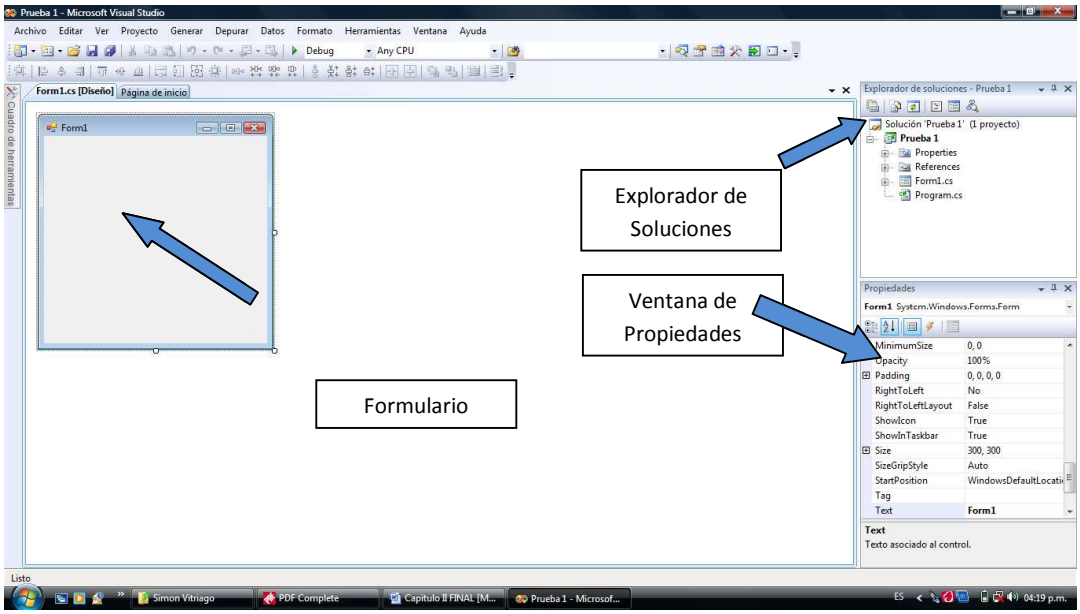


Figura 9. Espacio de trabajo en el entorno VS 2008

Fuente: Elaboración Propia

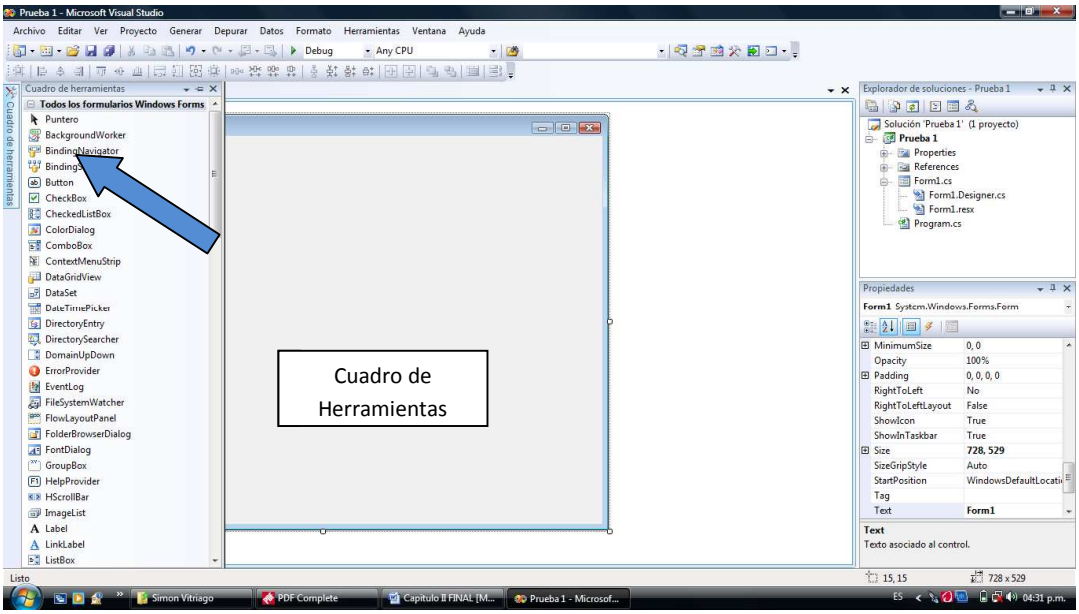


Figura 10. Cuadro de Herramientas de VS 2008

Fuente: Elaboración Propia

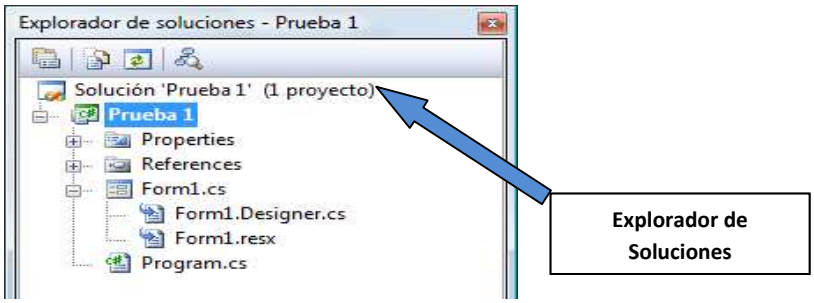


Figura 11: Explorador de Soluciones

Fuente: Elaboración Propia

- **Formulario:** Representa el espacio de trabajo en donde se arrastran los controles y se crea la interfaz que se le presentará al usuario. Por defecto, posee el nombre de "Form1.cs". El formulario también es un objeto, pero se puede decir que es un objeto contenedor de más objetos.

- **Propiedades:** Son atributos o características que posee un objeto, ya sea nombre, color, estilo o tipo de letra, ancho, alto, habilitación, posición, visibilidad, apariencia 2D o 3D, estilo del borde, entre otras.

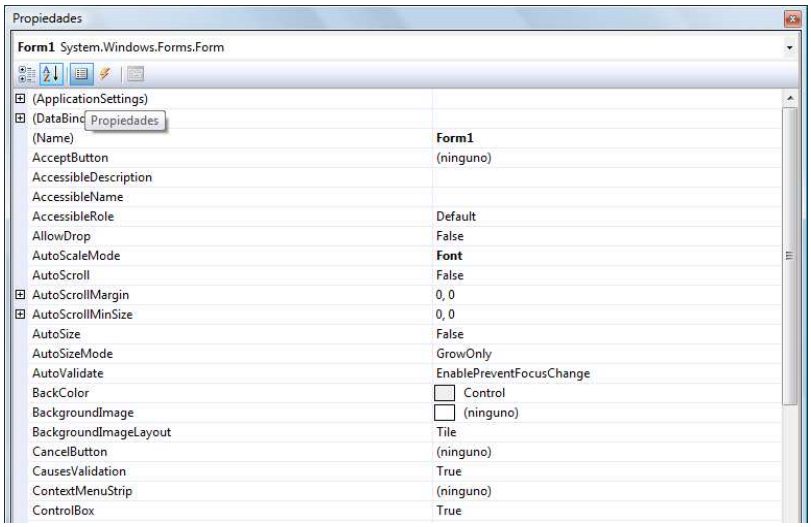


Figura 12: Propiedades del formulario

Fuente: Elaboración Propia

Al programar en Visual Studio C# 2008 se nos presentan dos entornos de trabajo: el de diseño, y el de Codificación.

Al hacer doble click al formulario que aparece por defecto:

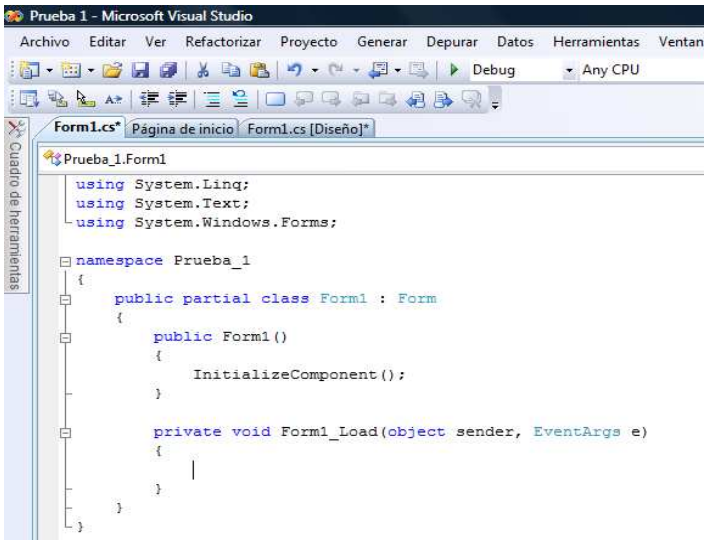


Figura 13. Espacio de trabajo en el cual se codifica cada objeto del entorno VS2008

Fuente: Elaboración Propia

Existen ActiveX que permiten simular instrumentos de medición, calderas, válvulas, tuberías, que pueden ser adaptados tanto a Visual C#, como a Visual Basic, Delphi y java.

Se utilizaran los ActiveX llamados “iocomp”. Descargándolos de su página oficial http://www.iocomp.com.

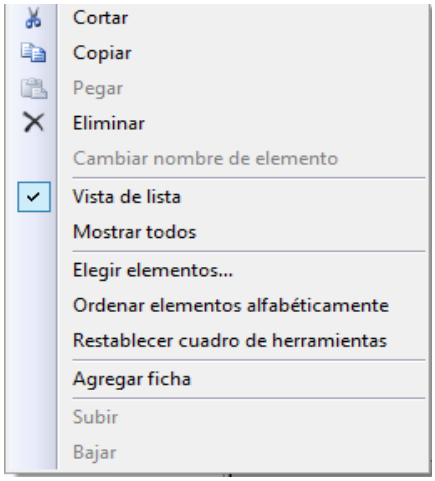


Figura 14: Ventana desplegable de cuadro de herramientas (al hacer click derecho)

Fuente: Elaboración Propia

Al hacer click derecho en la zona de cuadro de herramientas, se despliega esta ventana (ver Figura 14), en ella seleccionamos “Elegir elementos…”

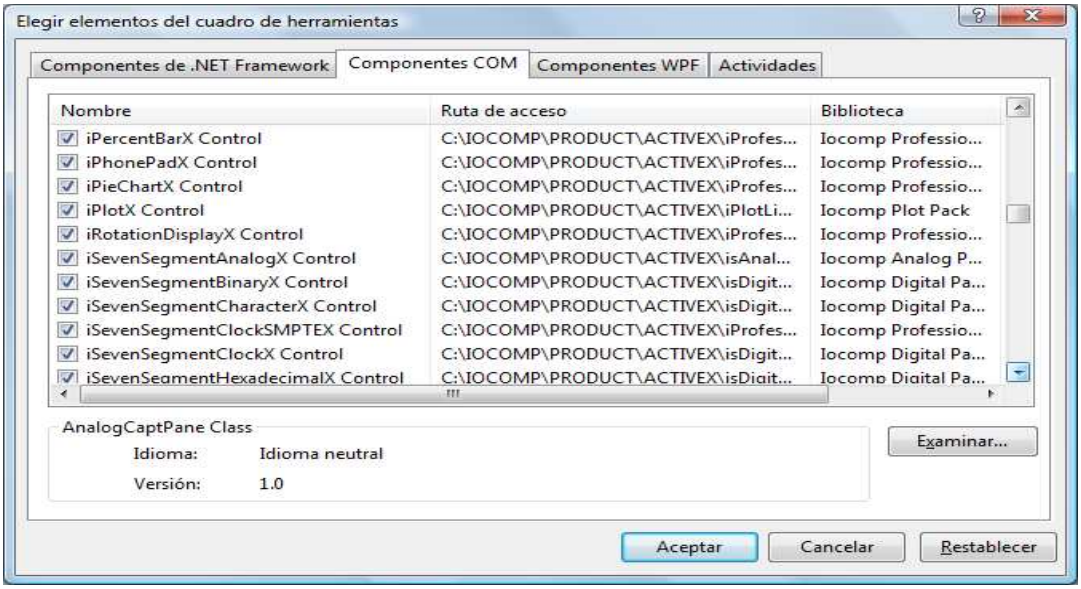


Figura 15. Ventana en la cual se seleccionan los ActiveX instalados, para ser añadidos al cuadro de herramientas

Fuente: Elaboración Propia

2.2.9. Tecnología USB

El Universal Serial Bus (bus universal en serie) o Conductor Universal en Serie (CUS), abreviado comúnmente USB, es un puerto que sirve para conectar periféricos a un ordenador. Fue creado en 1996 por siete empresas (que actualmente forman el consejo directivo): IBM, Intel, Northern Telecom, Compaq, Microsoft, Digital Equipment Corporation y NEC.

El diseño del USB tenía en mente eliminar la necesidad de adquirir tarjetas separadas para poner en los puertos bus ISA o PCI, y mejorar las capacidades plug-and-play permitiendo a esos dispositivos ser conectados o desconectados al sistema sin necesidad de reiniciar. Sin embargo, en aplicaciones donde se necesita ancho de banda para grandes transferencias de datos, o si se necesita una latencia baja, los buses PCI o PCIe salen ganando. Igualmente sucede si la aplicación requiere de robustez industrial. A favor del bus USB, cabe decir que cuando se conecta un nuevo dispositivo, el servidor lo enumera y agrega el software necesario para que pueda funcionar (esto dependerá ciertamente del sistema operativo que esté usando el ordenador).

El USB no puede conectar los periféricos porque sólo puede ser dirigido por el drive central así como: ratones, teclados, escáneres, cámaras digitales, teléfonos móviles, reproductores multimedia, impresoras, discos duros externos entre otros ejemplos, tarjetas de sonido, sistemas de adquisición de datos y componentes de red. Para dispositivos multimedia como escáneres y cámaras digitales, el USB se ha convertido en el método estándar de conexión. Para impresoras, el USB ha crecido tanto en popularidad que ha desplazado a un segundo plano a los puertos paralelos porque el USB hace mucho más sencillo el poder agregar más de una impresora a un ordenador.

Algunos dispositivos requieren una potencia mínima, así que se pueden conectar varios sin necesitar fuentes de alimentación extra. La gran mayoría de los concentradores incluyen fuentes de alimentación que brindan energía a los dispositivos conectados a ellos, pero algunos dispositivos consumen tanta energía que necesitan su

propia fuente de alimentación. Los concentradores con fuente de alimentación pueden proporcionarle corriente eléctrica a otros dispositivos sin quitarle corriente al resto de la conexión (dentro de ciertos límites).

En el caso de los discos duros, es poco probable que el USB reemplace completamente a los buses (el ATA (IDE) y el SCSI), pues el USB tiene un rendimiento más lento que esos otros estándares. Sin embargo, el USB tiene una importante ventaja en su habilidad de poder instalar y desinstalar dispositivos sin tener que abrir el sistema, lo cual es útil para dispositivos de almacenamiento externo. Hoy en día, una gran parte de los fabricantes ofrece dispositivos USB portátiles que ofrecen un rendimiento casi indistinguible en comparación con los ATA (IDE). Por el contrario, el nuevo estándar Serial ATA permite tasas de transferencia de hasta aproximadamente 150/300 MB por segundo, y existe también la posibilidad de extracción en caliente e incluso una especificación para discos externos llamada eSATA. El USB casi ha reemplazado completamente a los teclados y ratones PS/2, hasta el punto de que un amplio número de placas base modernas carecen de dicho puerto o solamente cuentan con uno válido para los dos periféricos.

Características de Transmisión:

Pin	Nombre	Color del Cable	Descripción
1	Vcc	Rojo	+5V
2	D-	Blanco	Data -
3	D+	Verde	Data +
4	Gnd	Negro	Tierra

Tabla 5: Descripción de los pines de un conector USB

Fuente: Elaboración Propia

Figura 16. Cables de datos de un conector USB

En la (figura 16) se puede observar que los cables de datos de un conector USB representan un par trenzado, éste último para así reducir el ruido e interferencias que se puedan originar durante la transferencia de datos en el cable USB.

Los dispositivos USB se clasifican en cuatro tipos según su velocidad de transferencia de datos:

- **Baja velocidad (1.0):** Tasa de transferencia de hasta 1.5 Mbps (192 KB/s). Utilizado en su mayor parte por dispositivos de interfaz humana (Human Interface Device, en inglés) como los teclados, los ratones, las cámaras web, etc.

- **Velocidad Completa (1.1):** Tasa de transferencia de hasta 12 Mbps (1.5 MB/s), según este estándar pero se dice en fuentes independientes que habría que realizar nuevamente las mediciones. Ésta fue la más rápida antes de la especificación USB 2.0, y muchos dispositivos fabricados en la actualidad trabajan a esta velocidad. Estos dispositivos dividen el ancho de banda de la conexión USB entre ellos, basados en un algoritmo de impedancias LIFO.

- **Alta velocidad (2.0):** Tasa de transferencia de hasta 480 Mbps (60 MB/s) pero por lo general de hasta 125 Mbps (16 MB/s). Está presente casi en el 99% de los ordenadores actuales. El cable USB 2.0 dispone de cuatro líneas, un par de datos, una de corriente y una de toma de tierra.

- **Super alta velocidad (3.0):** Tiene una tasa de transferencia de hasta 4.8 Gbps (600 MB/s). Esta especificación es diez veces más veloz que la anterior 2.0 y se lanzó a mediados de 2009 por Intel, según se estima, o quizá por otra empresa de Hardware,

de acuerdo con información recabada de Internet. Aunque actualmente cualquier distribución GNU/Linux es capaz de soportar el nuevo estándar, sin embargo, aún no hay hardware disponible. La velocidad del bus es diez veces más rápida que la del USB 2.0, debido a que han incluido 5 conectores extra, desechando el conector de fibra óptica propuesto inicialmente, y será compatible con los estándares anteriores. Se espera que los productos fabricados con esta tecnología lleguen al consumidor entre 2009 y 2015.

Las señales del USB se transmiten en un cable de par trenzado con impedancia característica de 90Ω ± 15%, cuyos hilos se denominan D+ y D-. Estos, colectivamente, utilizan señalización diferencial en full dúplex para combatir los efectos del ruido electromagnético en enlaces largos. D+ y D- suelen operar en conjunto y no son conexiones simples. Los niveles de transmisión de la señal varían de 0 a 0’3V para bajos (ceros) y de 2’8 a 3’6 V para altos (unos) en las versiones 1.0 y 1.1, y en ±400 mV en alta velocidad (2.0). En las primeras versiones, los alambres de los cables no están conectados a masa, pero en el modo de alta velocidad se tiene una terminación de 45 Ω a tierra o un diferencial de 90 Ω para acoplar la impedancia del cable. Este puerto sólo admite la conexión de dispositivos de bajo consumo, es decir, que tengan un consumo máximo de 100 mA por cada puerto; sin embargo, en caso de que estuviese conectado un dispositivo que permite 4 puertos por cada salida USB (extensiones de máximo 4 puertos), entonces la energía del USB se asignará en unidades de 100 mA hasta un máximo de 500 mA por puerto.

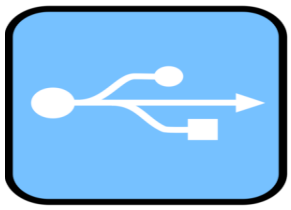


Figura 17: Símbolo del USB

Fuente: http://es.wikipedia.org/w/index.php?oldid=38132575

Figura 18: Memoria USB (Pendrive)

Fuente: http://es.wikipedia.org/w/index.php?oldid=38132575

Figura 19: Conector USB tipo A macho

Fuente: http://es.wikipedia.org/w/index.php?oldid=38132575

Figura 20: Prolongador USB

Fuente: http://es.wikipedia.org/w/index.php?oldid=38132575

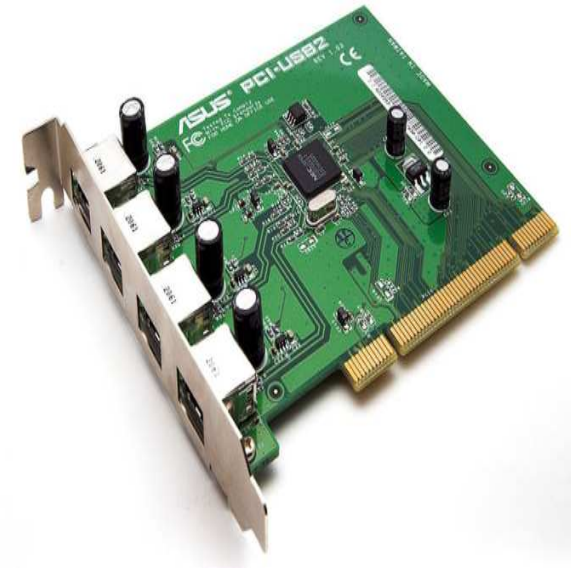


Figura 21: Tarjeta PCI-USB 2.0

Fuente: http://es.wikipedia.org/w/index.php?oldid=38132575

Figura 22: Adaptador USB a PS/2

Fuente: http://es.wikipedia.org/w/index.php?oldid=38132575

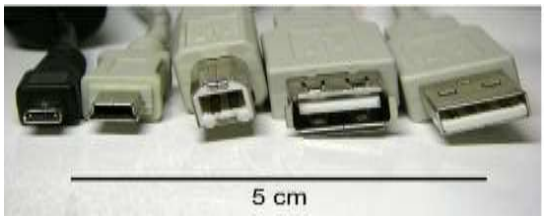


Figura 23: Tipos diferentes de conectores USB (de izquierda a derecha): micro USB macho, mini USB tipo B macho, tipo B macho, tipo A hembra, tipo A macho.

Fuente: http://es.wikipedia.org/w/index.php?oldid=38132575

Figura 24: Una memoria USB como ésta implementará normalmente la clase de dispositivo de almacenamiento masivo USB

Fuente: http://es.wikipedia.org/w/index.php?oldid=38132575

CAPÍTULO III
MARCO METODOLÓGICO

En el presente capítulo se detallan los métodos, técnicas y procedimientos empleados para determinar requerimientos, la lógica de la programación, para el desarrollo de los programas e implementación.

3.1. Tipo de investigación

El desarrollo de la investigación: implementación de un sistema de monitoreo de temperatura en el equipo evaporador-condensador vertical del laboratorio de ingeniería química haciendo uso del microcontrolador PIC18F2550, califica como un proyecto factible, tal como lo define el manual de trabajos de grado de la UPEL (2002): “Consiste en la investigación, elaboración y desarrollo de una propuesta de un modelo operativo viable para solucionar problemas, requerimientos o necesidades de organizaciones o grupos sociales; puede referirse a la formulación de políticas, programas, tecnologías, métodos o procesos”.

Con referencia a lo anterior, las razones por las que el proyecto es un proyecto factible, se debe a que se sustenta en la investigación, desarrollo e implementación de un sistema de monitoreo de temperatura en el equipo evaporador-condensador vertical del laboratorio de ingeniería química haciendo uso del microcontrolador PIC18F2550.

3.2. Técnicas de Investigación

3.2.1. Técnica para fijar requerimientos: Entrevista no estructurada (no formalizada)

Esta técnica se utilizó para fijar los requerimientos según la empresa Roso Electric Supply a través de la asesoría con el Ing. José M. Rodriguez S, quien posee sólidos conocimientos de electrónica en las áreas digitales y de programación así como de instrumentación virtual, 1.Wire®, ActiveX, USB.

3.2.2. Métodos y Técnicas para establecer la lógica de programación

3.2.2.1. Técnica de Diagrama de bloque

El Diccionario de la Real Academia Española (2005) señala que un diagrama es un dibujo en el que se muestran las relaciones entre las diferentes partes de un conjunto o sistema. Para hacer un Bosquejo del sistema se elaboró un diagrama utilizando bloques que permitiesen la fácil comprensión de los elementos del mismo.

3.2.2.2. Algoritmo

De acuerdo con G. Brassard y P. Bratley (1997), un algoritmo es un método para resolver un problema a través de una secuencia de pasos lógicos que llevará a cumplir un objetivo o solución teniendo en cuenta que debe ser definido, finito y eficiente.

Para solucionar computacionalmente un problema se debe:

- Seleccionar un modelo adecuado para el problema (representación del modelo).

- Concebir con respecto a dicho modelo un algoritmo que dé solución al problema (diseño del algoritmo).

- Programar el algoritmo en algún lenguaje de programación y ejecutar el programa en una computadora (programación del algoritmo).

Estructura Básica: inicio, constantes (datos inalterables), variables (datos alterables), ingresar datos (datos ingresados por el usuario que se guardarán en las variables), proceso de operaciones (ejecución de algoritmo sobre las variables y constantes), mostrar resultados (resultados de la operación algorítmica) y fin.

Este método representó la base fundamental sobre la cual se elaboraría el código fuente posteriormente en concordancia con los diagramas de flujo.

3.2.2.3. Diagramas de flujo como técnica de Diseño de Algoritmo

El diagrama de flujo es una representación gráfica para la definición, el análisis, o la solución de un problema. CEO (2006). El diagrama de flujo constituyó la técnica para diseñar el algoritmo ya que permitió recrear el código fuente en completa analogía con los pasos que se indican en las notas de aplicaciones de la página web http://maxim-ic.com.

Estos diagramas están sometidos a una estandarización de sus símbolos según la Norma ISO 5807, donde los símbolos más comunes son:

- ***Flecha.*** Indica el sentido y trayectoria del proceso de información o tarea.

- ***Rectángulo*.** Se usa para representar un proceso determinado.

- ***Rombo*.** Se utiliza para representar una condición, por lo que a su salida, el proceso tendrá dos caminos posibles.

- ***Círculo*.** Representa un punto de conexión entre procesos. Se utiliza cuando es necesario dividir un diagrama de flujo en varias partes, por ejemplo por razones de espacio o simplicidad. Una referencia debe de darse dentro para distinguirlo de otros.

Existen además una variedad de formas especiales para denotar las entradas, las salidas, los almacenamientos, etc. En los diagramas de flujo se presuponen los siguientes aspectos:

- Existe siempre un camino que permite llegar a una solución
- Existe un único inicio del proceso
- Existe un único punto de fin para el proceso de flujo, salvo del rombo que indica una comparación con dos caminos posibles.

3.3. Fases Metodológicas:

3.3.1. Fase I. Evaluar las características actuales del proceso de medición de temperatura en el equipo para determinar las debilidades presentes en el mismo.

- Verificar con qué instrumento se está realizando la medición de temperatura en el equipo.
- Determinar en cuántos puntos se están realizando las mediciones de temperatura en el equipo.
- Conocer las debilidades e inconformidad del proceso de medición de temperatura actual en el equipo.

3.3.2. Fase II. Seleccionar el sensor de temperatura 1-Wire® a emplear para realizar las mediciones.

- Elaborar una lista de los sensores digitales de temperatura 1-Wire® existentes en el mercado.
- Seleccionar el sensor digital de temperatura 1-Wire® que se encuentre disponible en el mercado y de ser posible el de menor costo.

3.3.3. Fase III. Diseñar una tarjeta de adquisición de datos con comunicación USB, en donde se reciban los valores provenientes de los sensores de temperatura 1-wire®.

- Programar el PIC18F2550 para que permita leer los sensores de temperatura 1-Wire® y además permita establecer la comunicación USB de la tarjeta de adquisición de datos.

- Realizar el montaje en físico de la tarjeta de adquisición de datos la cual consta principalmente de: PIC18F2550, resonador de 20Mhz, conector USB hembra tipo B, resistencia de Pull-up para la comunicación con los sensores digitales de temperatura 1-Wire®.

3.3.4. Fase IV. Diseñar un software en visual studio C# 2008 que permita mostrar los datos recibidos por la tarjeta de adquisición de datos, mediante una gráfica de perfil de temperatura.

- Abrir el entorno de POO (Programación Orientado a Objetos) Microsoft Visual C# 2008, crear los formularios a utilizar, añadir los ActiveX iocomp para ser usados en la creación del software.
- Añadir la librería “usblibrary.dll”, a las referencias del entorno de POO, para así tener acceso al ActiveX “UsbHidPort”, con el cual se podrá establecer la comunicación USB desde la PC por medio del software a realizar.
- Elaborar en el entorno de POO el código que permita recibir desde la tarjeta de adquisición de datos el valor de temperatura de cada sensor, para ser mostrado en el software en tiempo real.
- Crear en un formulario la gráfica de temperatura de cada sensor.

3.3.5. Fase V. Realizar un conjunto de pruebas para validar el funcionamiento del sistema.

- Realizar mediciones de temperatura en diversos puntos con la termocupla tipo k.
- Realizar mediciones de temperatura en los mismos puntos anteriores empleando los sensores digitales de temperatura 1-Wire®.
- Comparar ambas mediciones y determinar si los resultados del sistema son correctos.

CAPÍTULO IV
ANÁLISIS DE LOS RESULTADOS

Según la metodología utilizada para el desarrollo de esta investigación, se aplicaron diversas técnicas y se utilizaron varios instrumentos que permitieron lograr los resultados esperados. Los resultados obtenidos se explican a continuación, en función de las fases descritas en el marco metodológico (capítulo III).

4.1 Evaluar las características actuales del proceso de medición de temperatura en el equipo para determinar las debilidades presentes en el mismo.

- **Verificar con qué instrumento se está realizando la medición de temperatura en el equipo.**

Actualmente en el equipo evaporador-condensador vertical se está realizando la medición de temperatura a través de una termocupla tipo K marca FLUKE.

- **Determinar en cuántos puntos se están realizando las mediciones de temperatura en el equipo.**

En el equipo evaporador-condensador vertical se están realizando las mediciones de temperatura solo en tres puntos, justamente en los orificios que posee el equipo para conectar la termocupla tipo k.

- **Conocer las debilidades e inconformidad del proceso de medición de temperatura actual en el equipo.**

Con respecto a información que no se obtiene mediante el proceso de medición de temperatura actual se tiene:

- ✓ Medición de temperatura en muchos más puntos del equipo evaporador-condensador vertical, para así establecer el perfil de temperatura deseado.
- ✓ Valor promedio de temperatura en el equipo de forma automática.
- ✓ Permitir observar la gráfica de temperatura de cada punto de medición.

La principal debilidad que posee el proceso de medición de temperatura actual es la necesidad de conectar y desconectar la termocupla tipo k para así obtener la medición, teniendo en cuenta que al fijar sensores de temperatura al equipo se obtendrá el valor de temperatura en cualquier momento sin necesidad de conectar ni desconectar ningún dispositivo medidor de temperatura.

4.2 Seleccionar el sensor de temperatura 1-Wire® a emplear para realizar las mediciones.

- ❖ Elaborar una lista de los sensores digitales de temperatura 1-Wire® existentes en el mercado.

Descripción del Dispositivo 1-Wire®	Modelo	Disponibilidad en el Mercado
Termómetro Digital 1-Wire® de Alta Precisión	DS1820 ó DS18S20	SI
Termómetro Digital 1-Wire® de Resolución Programable	DS18B20	NO

Tabla 6: Lista de Sensores digitales de temperatura 1-Wire®

Fuente: Elaboración Propia

- ❖ Seleccionar el sensor digital de temperatura 1-Wire® que se encuentre disponible en el mercado y de ser posible el de menor costo.

El dispositivo seleccionado fue el Termómetro Digital 1-Wire® de Alta Precisión (DS1820), debido a que se encontraba disponible en el mercado y cumple con los requisitos para la implementación de nuestro proyecto.

Algunas de sus características son las siguientes:

- ✓ Por ser de tecnología 1-Wire® para establecer la comunicación con el Microcontrolador simplemente se necesita un pin de los puertos del mismo a la patita "DQ" del sensor.
- ✓ Puede ser alimentado desde la línea de datos "DQ". El rango de suministro de energía es 3.0V a 5.5V.
- ✓ Las mediciones de temperaturas van desde -55°C a +125°C (-67°F a +257°F).
- ✓ Posee una precisión de ±0.5°C desde -10°C a +85°C.
- ✓ Resolución del termómetro de 9 bits.
- ✓ Conversión de temperatura en 750ms (máximo).

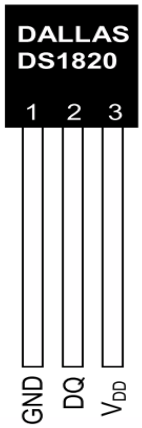


Figura 26: Termómetro Digital 1-Wire® de Alta Precisión

Fuente: Datasheet del dispositivo DS1820 (Ver Anexo A)

4.3 Diseñar una tarjeta de adquisición de datos con comunicación USB, en donde se reciban los valores provenientes de los sensores de temperatura 1-wire®.

Para el diseño de la tarjeta de adquisición de datos con comunicación USB, se hiso uso del PIC18F2550 el cual se muestra en la figura 27.

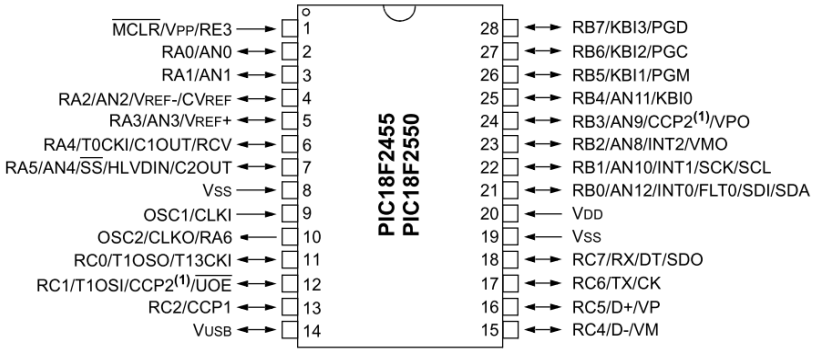


Figura 27: Microcontrolador PIC18F2550

Fuente: Datasheet del microcontrolador PIC18F2550 (Ver Anexo A)

Algunas características del microcontrolador PIC18F2550 son las siguientes:

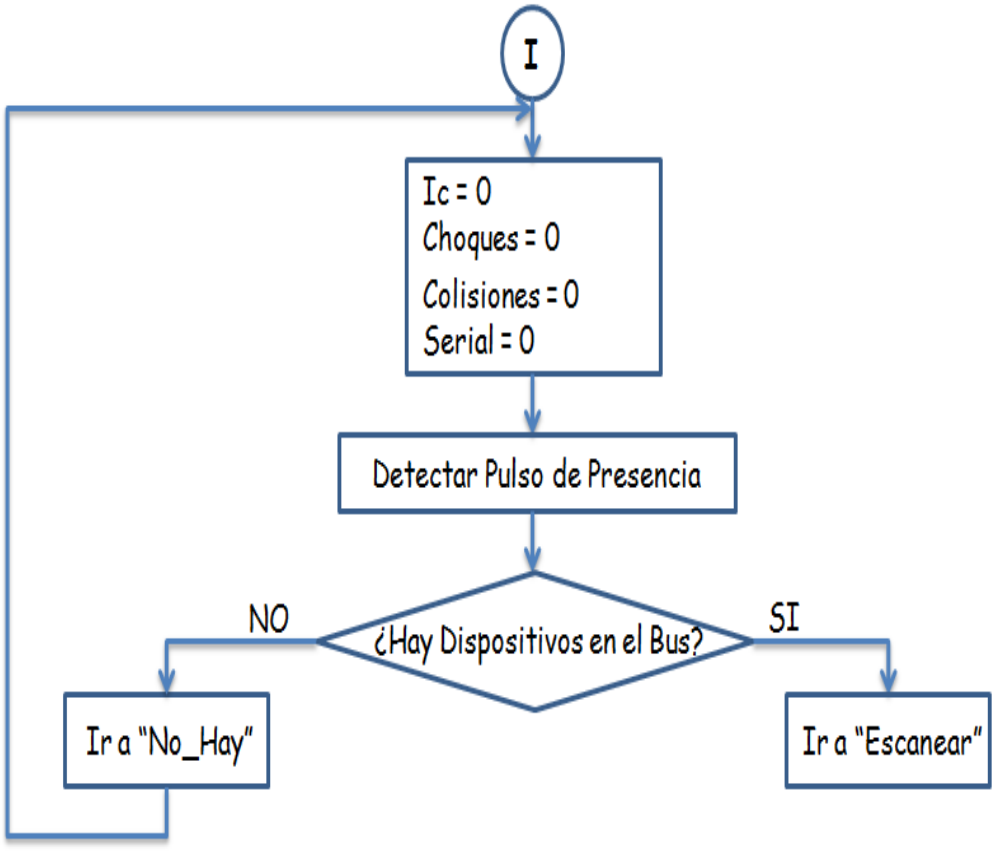
Características	PIC18F2550
Frecuencia de Operación	DC-48 MHz
Memoria de Programa (Bytes)	32768
Memoria de Programa (Instrucciones)	16384
Memoria de Datos (Bytes)	2048
Puertos I/O	Puertos A, B, C, (E)
Velocidad de Comunicación USB	Baja Velocidad: 1.5Mb/s Alta Velocidad: 12Mb/s
Transferencia que soporta el bus USB	Control, Interrupciones, Síncrono y Bulk
Endpoints que soporta el bus USB	32 Endpoints (16 Bidireccional)

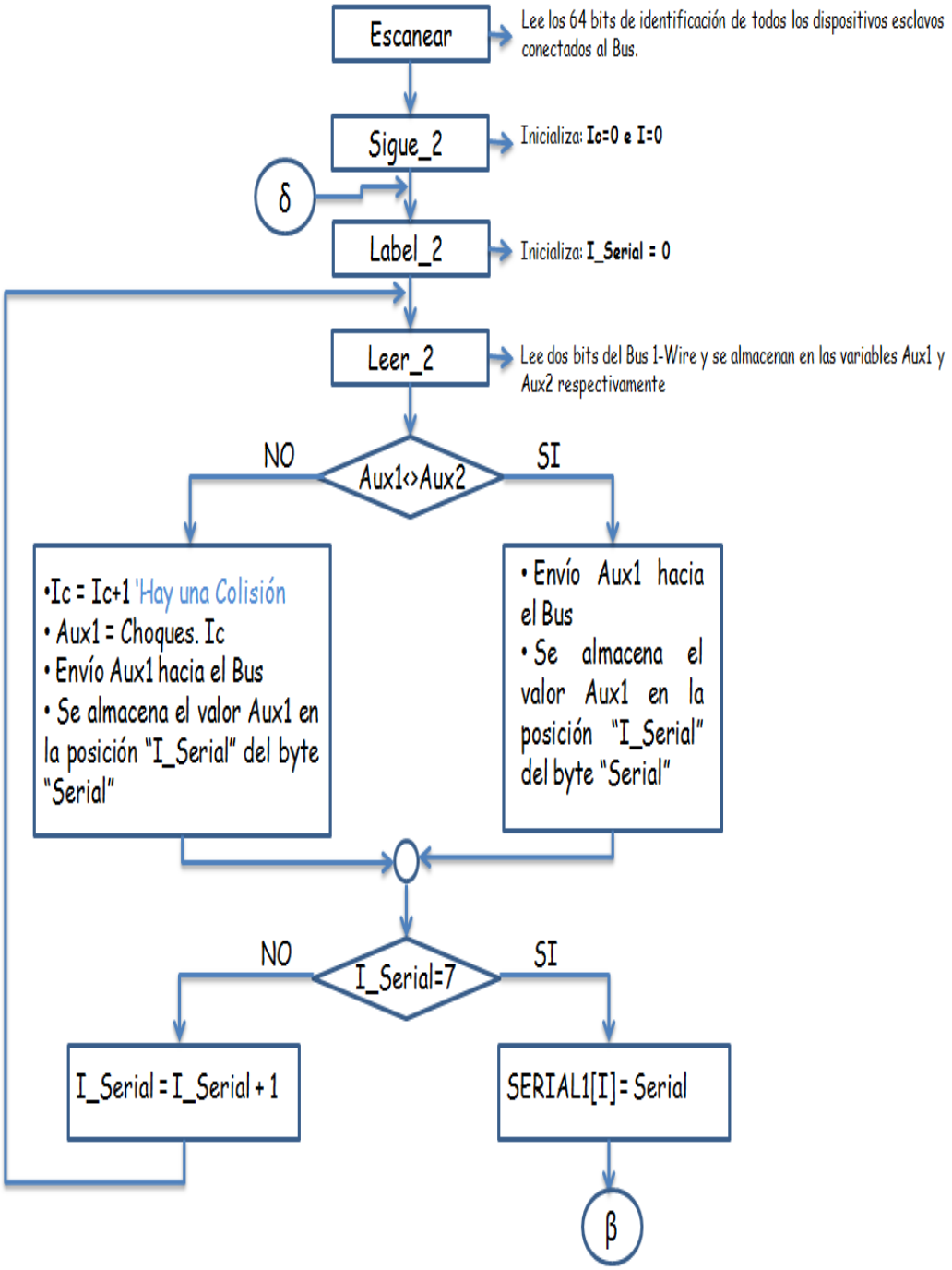
Tabla 7: Características del microcontrolador PIC18F2550

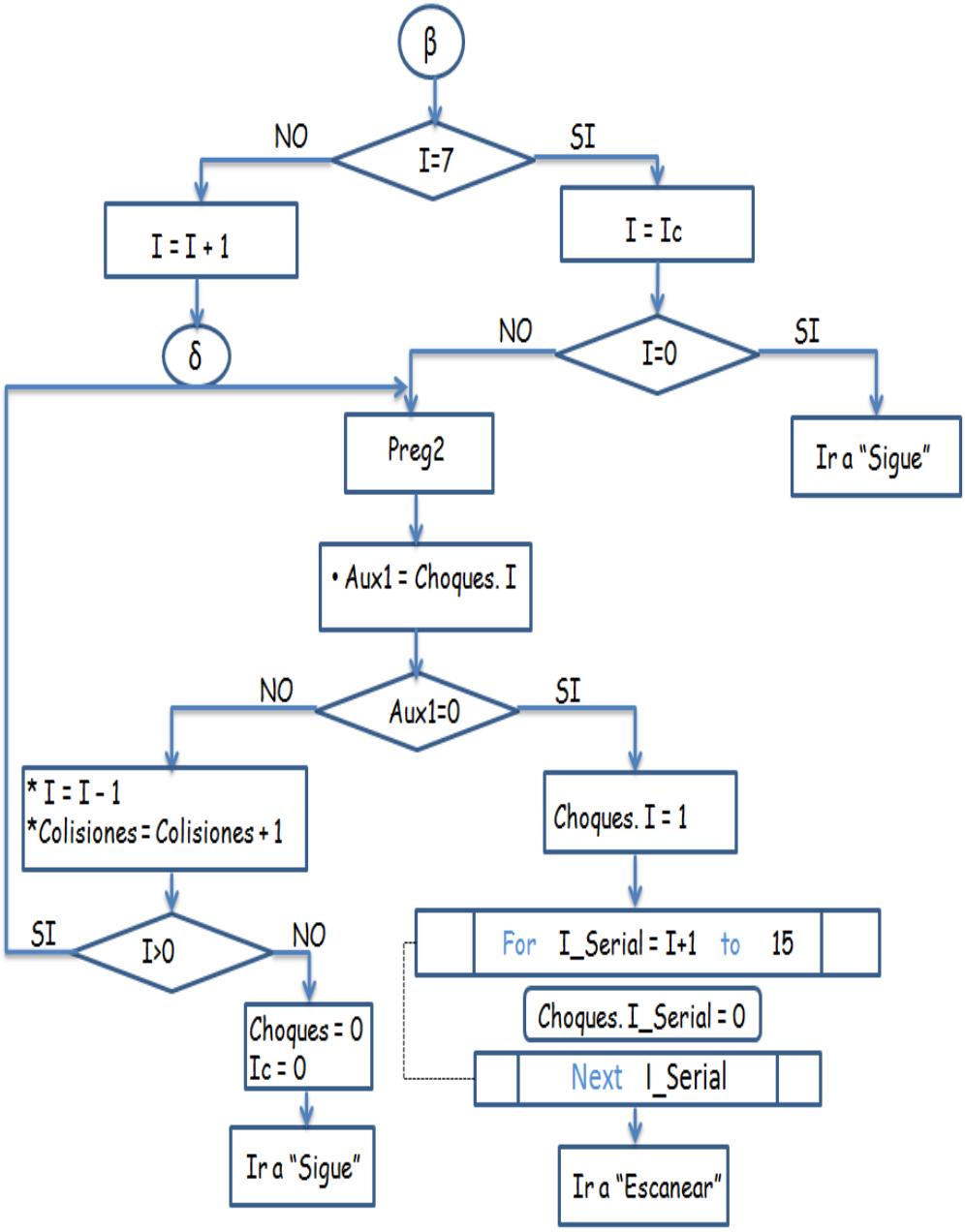
Fuente: Datos Extraídos del Datasheet del microcontrolador PIC18F2550 (Ver Anexo A)

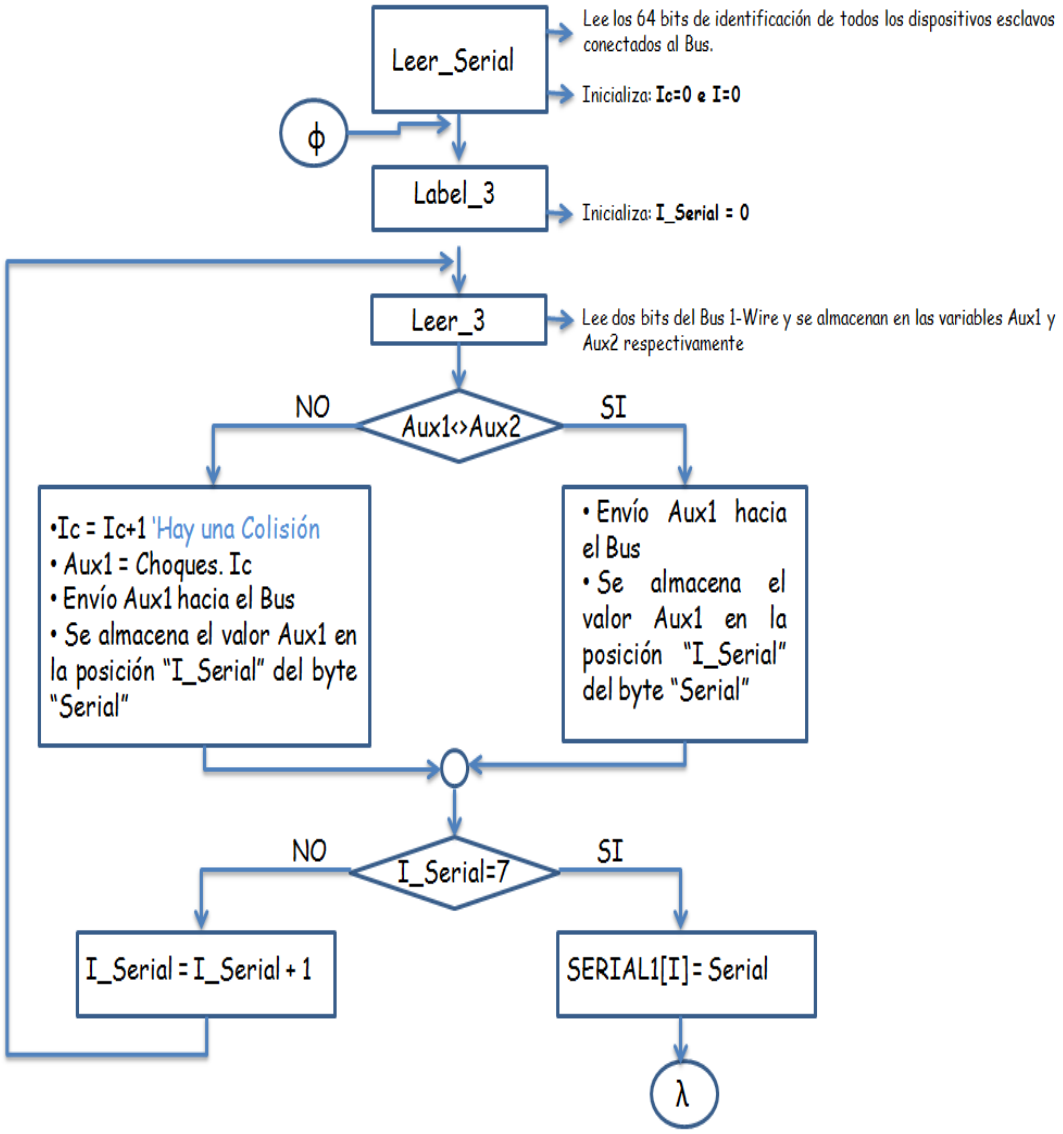
- **Programar el PIC18F2550 para que permita leer los sensores de temperatura 1-Wire® y además permita establecer la comunicación USB de la tarjeta de adquisición de datos.**

La programación del microcontrolador PIC18F2550 fue realizada en el programa PIC CCS, el cual se trabaja con el lenguaje de alto nivel C. Antes de mostrar el código realizado en el programa PIC CCS se procederá a mostrar el algoritmo de búsqueda de dispositivos 1-Wire® empleado para así poder distinguir a cada dispositivo 1-Wire® conectado al único bus de datos DQ. A continuación se presenta el diagrama de flujo del algoritmo de Búsqueda de dispositivos 1-Wire®:




Escanear
Lee los 64 bits de identificación de todos los dispositivos esclavos conectados al Bus.
Sigue_2
Inicializa: Ic=0 e I=0
δ
Label_2
Inicializa: I_Serial = 0
Leer_2
Lee dos bits del Bus 1-Wire y se almacenan en las variables Aux1 y Aux2 respectivamente
NO
Aux1<>Aux2
SI
•Ic = Ic+1 'Hay una Colisión
• Aux1 = Choques. Ic
• Envío Aux1 hacia el Bus
• Se almacena el valor Aux1 en la posición "I_Serial" del byte "Serial"
• Envío Aux1 hacia el Bus
• Se almacena el valor Aux1 en la posición "I_Serial" del byte "Serial"
NO
I_Serial=7
SI
I_Serial = I_Serial + 1
SERIAL1[I] = Serial
β


β
NO
I=7
SI
I = I + 1
I = Ic
δ
NO
I=0
SI
Preg2
Ir a "Sigue"
• Aux1 = Choques. I
NO
Aux1=0
SI
* I = I - 1
*Colisiones = Colisiones + 1
Choques. I = 1
SI
I>0
NO
For I_Serial = I+1 to 15
Choques = 0
Ic = 0
Choques. I_Serial = 0
Next I_Serial
Ir a "Sigue"
Ir a "Escanear"


Leer_Serial
Lee los 64 bits de identificación de todos los dispositivos esclavos conectados al Bus.
Inicializa: Ic=0 e I=0
ϕ
Label_3
Inicializa: I_Serial = 0
Leer_3
Lee dos bits del Bus 1-Wire y se almacenan en las variables Aux1 y Aux2 respectivamente
NO
Aux1<>Aux2
SI
•Ic = Ic+1 'Hay una Colisión
• Aux1 = Choques. Ic
• Envío Aux1 hacia el Bus
• Se almacena el valor Aux1 en la posición "I_Serial" del byte "Serial"
• Envío Aux1 hacia el Bus
• Se almacena el valor Aux1 en la posición "I_Serial" del byte "Serial"
NO
I_Serial=7
SI
I_Serial = I_Serial + 1
SERIAL1[I]= Serial
λ

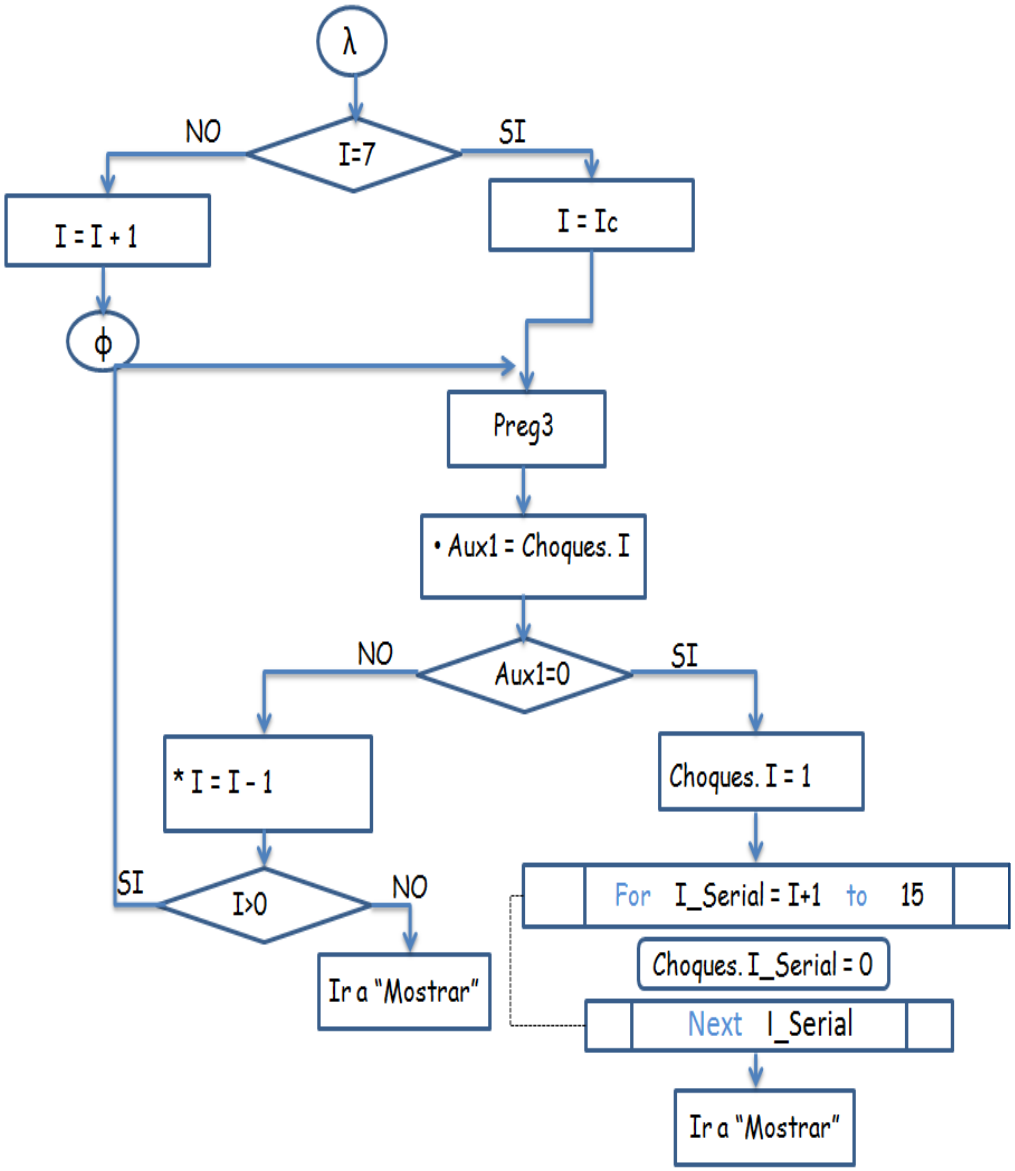


Figura 28. Diagrama de Flujo representando el Algoritmo de Búsqueda y Selección de dispositivos 1-Wire®

Fuente: Elaboración Propia

A continuación se presenta el diagrama de flujo del procedimiento del cálculo de temperatura del Termómetro Digital 1-Wire® de Alta Precisión (DS1820):

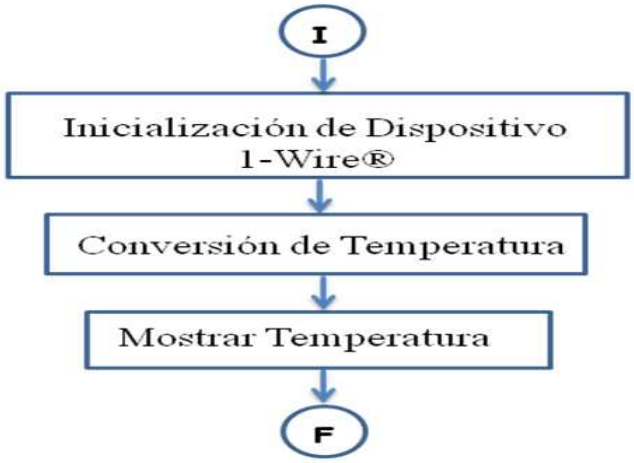


Figura 29. Diagrama de Flujo del procedimiento del cálculo de temperatura del Termómetro Digital 1-Wire® de Alta Precisión (DS1820)

Fuente: Elaboración Propia

En la figura 30, se observa el compilador que se empleó para realizar el código del PIC18F2550.

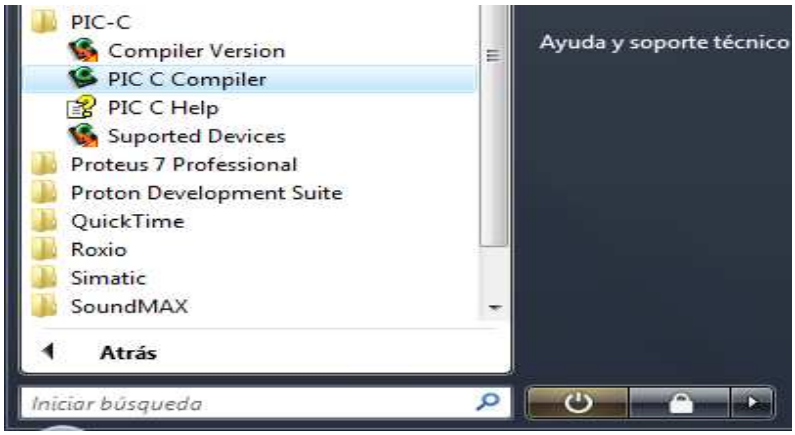


Figura 30. Compilador empleado para programar al PIC18F2550

Fuente: Elaboración Propia

Nota: El código fuente de la programación del PIC18F2550 se encuentra en el Apéndice A (Electrónico).

- **Realizar el montaje en físico de la tarjeta de adquisición de datos la cual consta principalmente de: PIC18F2550, resonador de 20Mhz, conector USB hembra tipo B, resistencia de Pull-up para la comunicación con los sensores digitales de temperatura 1-Wire®.**

Antes de proceder a realizar el diseño de la tarjeta de adquisición de datos, se realizó una simulación en Proteus. Como se muestra en la figura 31:

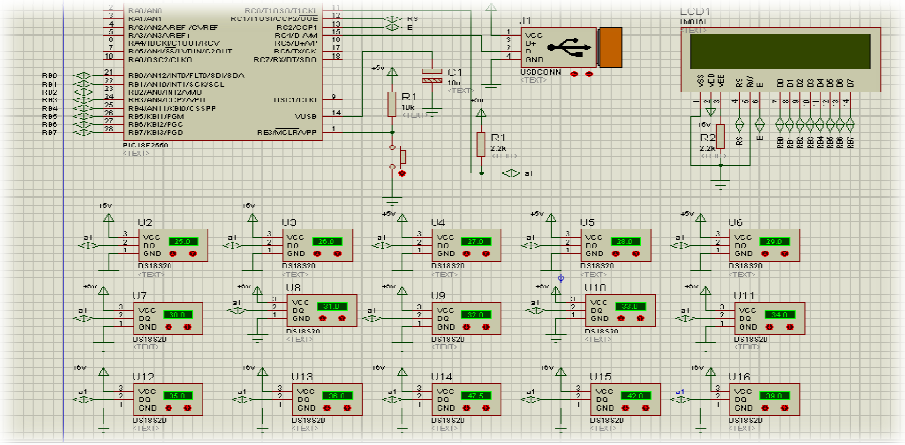


Figura 31. Circuito de la Tarjeta de Adquisición de Datos (TAD) realizado en Proteus

Fuente: Elaboración Propia

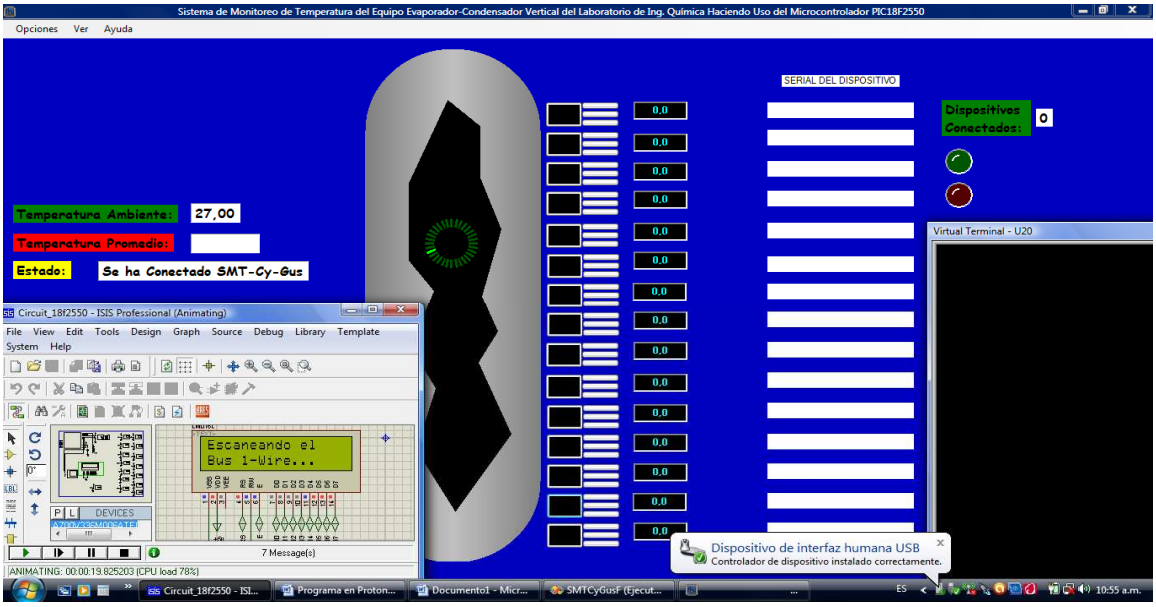
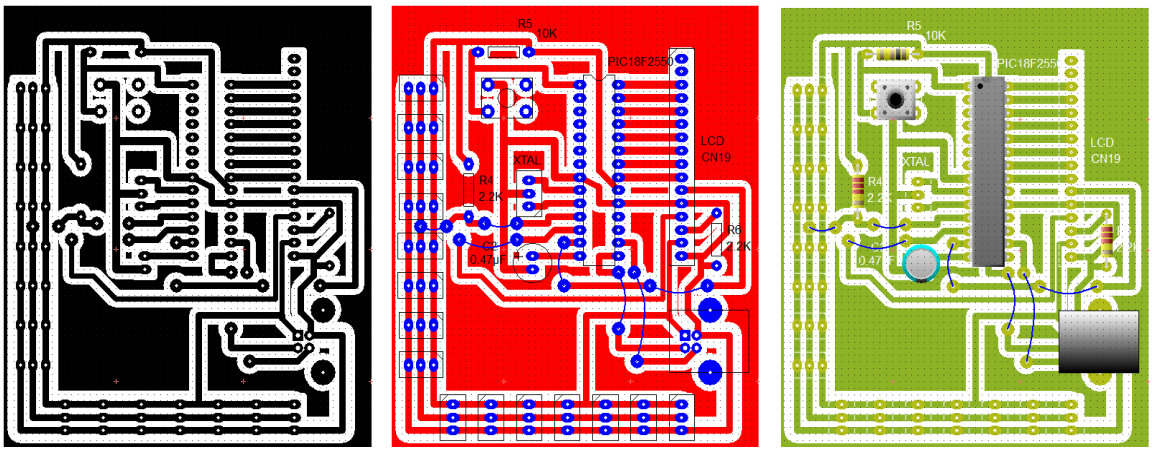


Figura 32. Simulando en Proteus el código del PIC18F2550

Fuente: Elaboración Propia

El diseño del esquemático del circuito impreso fue realizado en el software PCB Wizard. A continuación en la figura 33 se presentan las tres principales vistas del esquemático:



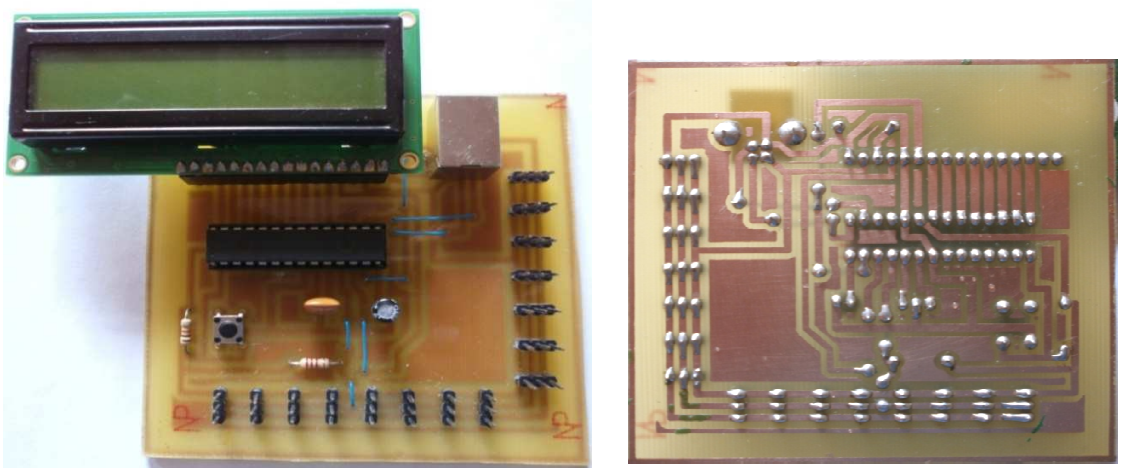
(a) **(b)** **(c)**

Figura 33. Esquemático de la Tarjeta de Adquisición de Datos (TAD)

(a). Vista "AirtWork", (b). Vista "Normal", (c). Vista "Real World"

Fuente: Elaboración Propia

Quedando el montaje en físico como se muestra en la figura 33.



(a) **(b)**

Figura 34. Tarjeta de Adquisición de Datos (TAD)

(a). Vista Superior, (b). Vista Inferior

Fuente: Elaboración Propia

Los componentes utilizados para diseñar la Tarjeta de adquisición de datos que se muestra en la figura 34.(a) se muestran en la tabla 8. Los precios fueron tomados de la empresa DIGITECH ELECTRONICS ubicada en Maracay.

Componente	Cantidad	Precio (Bsf)
Pantalla LCD 16x2	1	51,52
Conector Hembra USB Tipo B	1	7,37
Base de 28 Pines	1	1,76
PIC18F2550	1	79,80
Condensador Electrolítico de 0.47 µF/50V	1	0,62
Resonador de 20 MHz	1	5,5
Resistencia de 2.2 KΩ	2	0,62
Resistencia de 10 KΩ	1	0,31
Pulsador N.O	1	3,11
Base de 8 pines para LCD	2	4,5
Espadines de 3 pin	15	4,7
TOTAL		159,81

Tabla 8: Lista de Componentes utilizados para el diseño de la TAD

Fuente: Elaboración Propia

4.4 Diseñar un software en visual studio C# 2008 que permita mostrar los datos recibidos por la tarjeta de adquisición de datos, mediante una gráfica de perfil de temperatura.

- Abrir el entorno de POO (Programación Orientado a Objetos) Microsoft Visual C# 2008, crear los formularios a utilizar, añadir los ActiveX iocomp para ser usados en la creación del software.

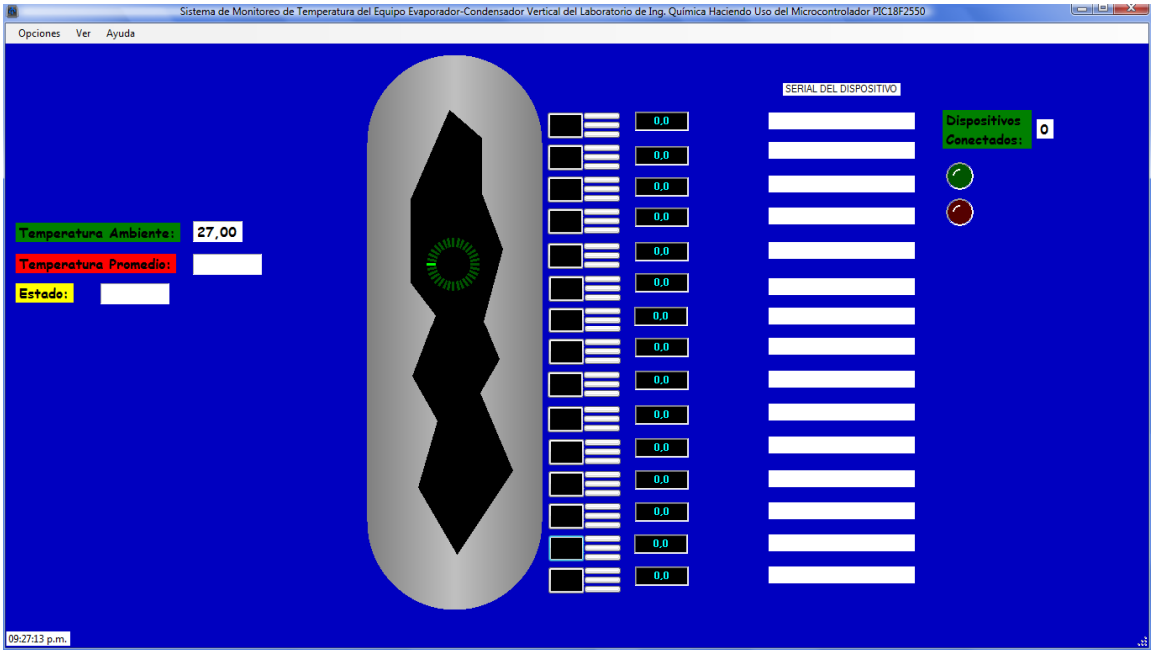


Figura 35. Formulario Principal (SMT_Cy_Gus.cs) del Software realizado en Visual Studio C# 2008

Fuente: Elaboración Propia

Como se observa en la Figura 35, se utilizó la herramienta MenuStrip, la cual nos permitió añadir un Menú “Opciones”, “Ver” y “Ayuda”.

El menú Opciones posee los siguientes submenús: “Cálculo de Temperatura Promedio”, “Limpiar Temperaturas”, “Almacenar Temperaturas en Archivo de Texto”, ”Cerrar”. En la figura 36 se puede observar lo antes descrito:

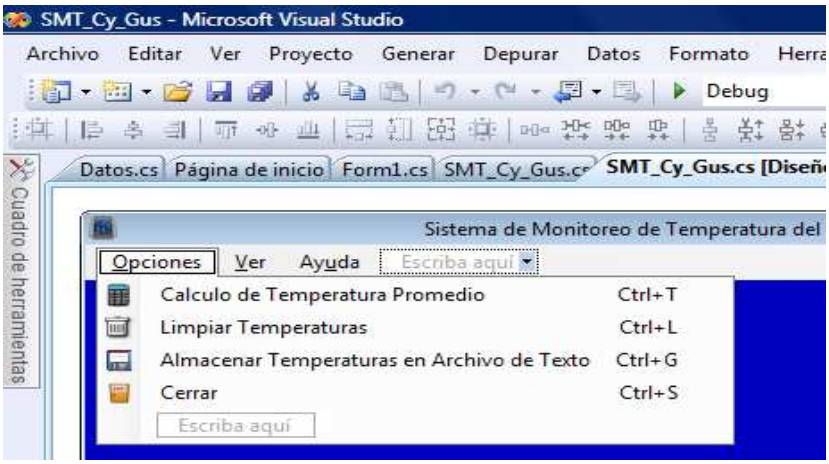


Figura 36. Menú Opciones y sus Submenús

Fuente: Elaboración Propia

El menú Ver posee un submenú: “Perfil de Temperatura”. En la figura 37 se puede observar lo antes descrito:

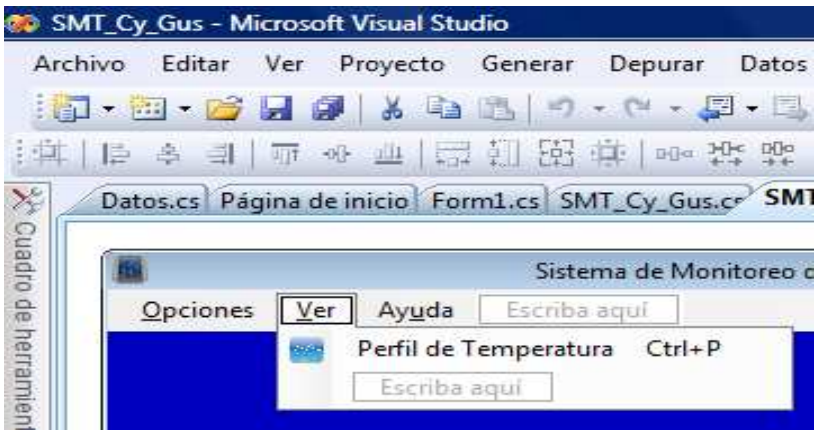


Figura 37. Menú Ver y su Submenú

Fuente: Elaboración Propia

El menú Ayuda posee un submenú: “Acerca de SMT-Cy-Gus”. En la figura 38 se puede observar lo antes descrito:

Figura 38. Menú Ayuda y su Submenú

Fuente: Elaboración Propia

- Añadir la librería “usblibrary.dll”, a las referencias del entorno de POO, para así tener acceso al ActiveX “UsbHidPort”, con el cual se podrá establecer la comunicación USB desde la PC por medio del software a realizar.

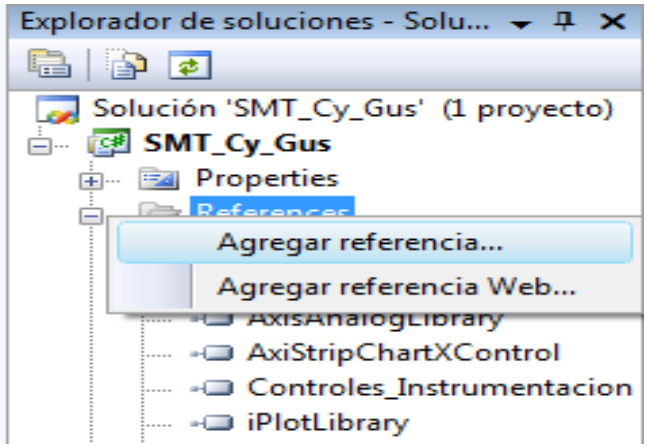


Figura 39. Procedimiento para agregar una librería a la aplicación en el entorno Visual Studio C# 2008

Fuente: Elaboración Propia

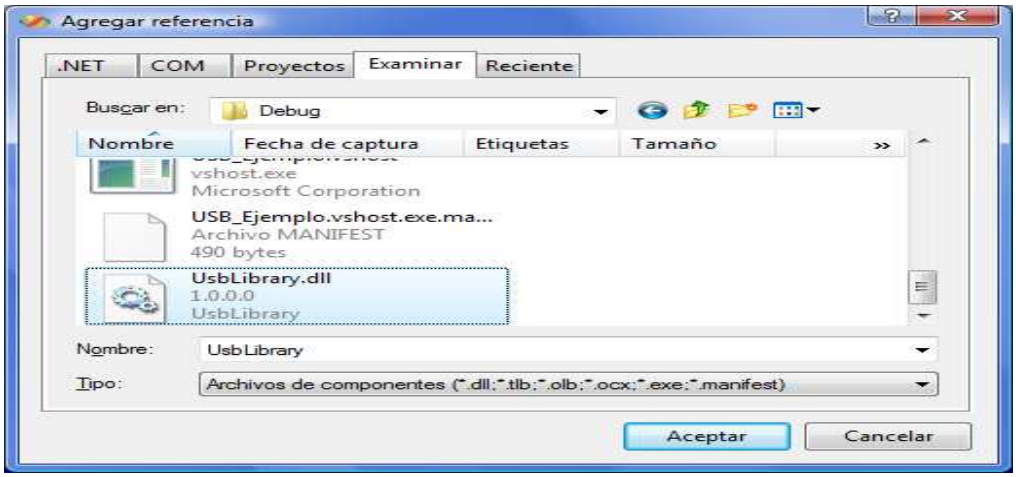


Figura 40. Añadiendo la librería "UsbLibrary.dll" a nuestra aplicación

Fuente: Elaboración Propia

La librería "UsbLibrary.dll", permite comunicar vía USB a nuestra aplicación con la tarjeta de adquisición de datos antes diseñada, por medio del ActiveX "UsbHidPort", el cual se muestra en la figura 41.

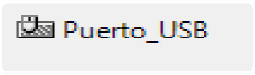


Figura 41. ActiveX UsbHidPort

Fuente: Elaboración Propia

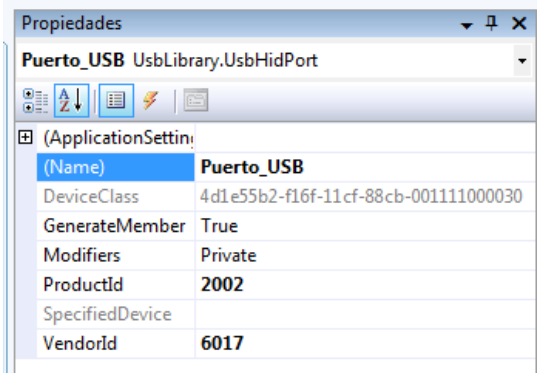


Figura 42. Propiedades del ActiveX UsbHidPort

Fuente: Elaboración Propia

El ActiveX UsbHidPort posee los siguientes eventos:

- **OnDataRecieved:**

El Nombre para nuestra aplicación de éste evento es **Puerto_USB_OnDataRecieved**, como su nombre lo indica éste evento es llamado al momento en que se reciben datos hacia la aplicación.

- **OnDataSend:**

El Nombre para nuestra aplicación de éste evento es **Puerto_USB_OnDataSend**, como su nombre lo indica éste evento es llamado al momento en que se envían datos desde la aplicación. Éste evento no posee código debido a que en ningún momento se envían datos desde la aplicación.

- **OnDeviceArrived:**

El Nombre para nuestra aplicación de éste evento es **Puerto_USB_ OnDeviceArrived**, como su nombre lo indica éste evento es llamado al momento en que se detecta la conexión de algún dispositivo USB al PC.

- **OnDeviceRemoved:**

El Nombre para nuestra aplicación de éste evento es **Puerto_USB_ OnDeviceRemoved**, como su nombre lo indica éste evento es llamado al momento en que se detecta la desconexión de algún dispositivo USB al PC.

- **OnSpecifiedDeviceArrived:**

El Nombre para nuestra aplicación de éste evento es **Puerto_USB_ OnSpecifiedDeviceArrived**, como su nombre lo indica éste evento es llamado al momento en que se detecta la conexión del dispositivo USB en específico (Dispositivo el cual posee el vendorID y ProductID que se le asignó al activeX HidPort) al PC. A continuación se presenta el código realizado dentro de éste evento:

- **OnSpecifiedDeviceRemoved:**

El Nombre para nuestra aplicación de éste evento es **Puerto_USB_ OnSpecifiedDeviceRemoved**, como su nombre lo indica éste evento es llamado al momento en que se detecta la desconexión de un dispositivo en específico USB al PC. A continuación se presenta el código realizado dentro de éste evento:

- Elaborar en el entorno de POO el código que permita recibir desde la tarjeta de adquisición de datos el valor de temperatura de cada sensor, para ser mostrado en el software en tiempo real.

El código para recibir los datos de valor de temperatura de cada sensor, está contenido en el evento **Puerto_USB_OnDataRecieved**.

- Crear en un formulario la gráfica de temperatura de cada sensor.

A continuación se presenta el formulario (Form1.cs) en el cual se presenta la Gráfica de Perfil de temperatura en la figura 43:

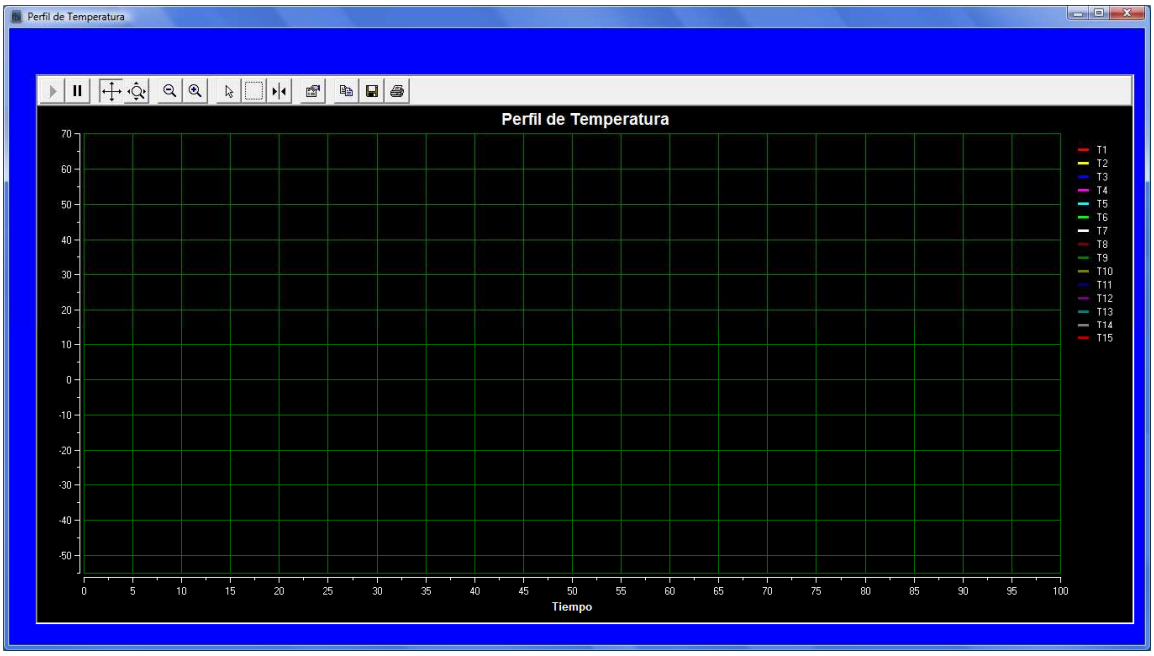
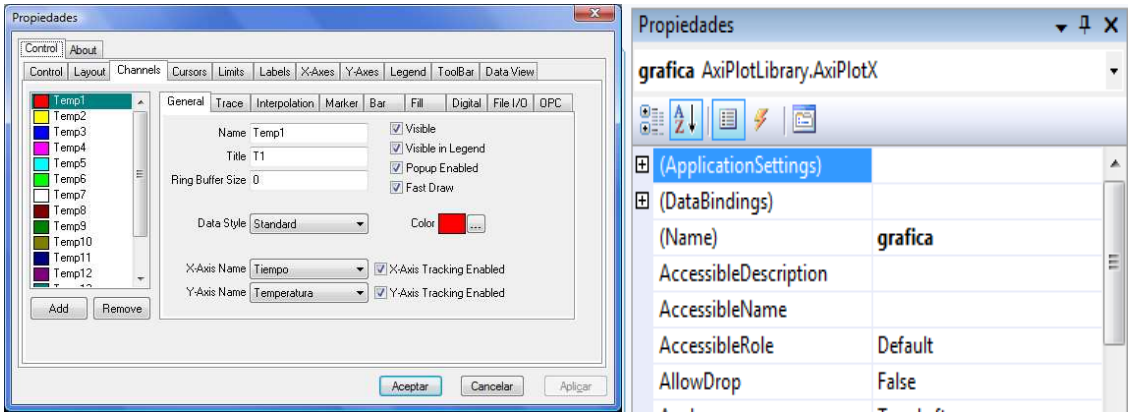


Figura 43. Formulario (Form2.cs) Perfil de Temperatura

Fuente: Elaboración Propia



(a) **(b)**

Figura 44. (a) y (b). Propiedades del ActiveX AxiPlotX

Fuente: Elaboración Propia

En el formulario Form2.cs se añadió un ActiveX timer1 el cual posee un solo evento (**timer1_Tick**). El ActiveX se presenta en la figura 45.

Figura 45. ActiveX Timer1

Fuente: Elaboración Propia

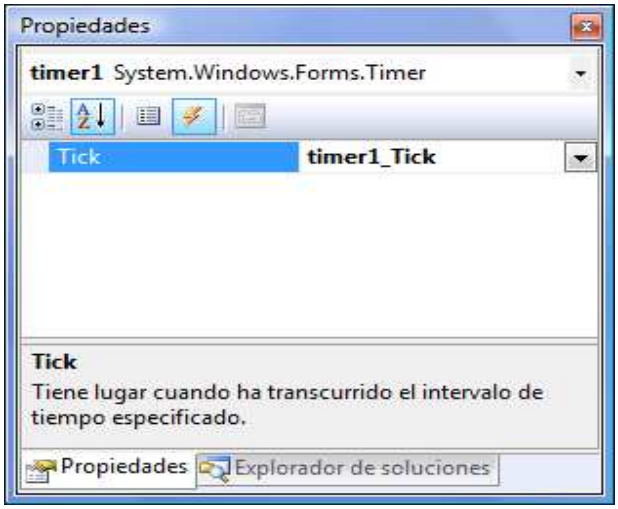


Figura 46. Propiedades del ActiveX Timer1

Fuente: Elaboración Propia

Nota: El código fuente del software se encuentra en el Apéndice B (Electrónico).

4.5 Realizar un conjunto de pruebas para validar el funcionamiento del sistema.

❖ Realizar mediciones de temperatura en diversos puntos con la termocupla tipo k.

Antes de proceder a realizar las mediciones de temperatura, fue necesario fijar la presión en el equipo Evaporador con 15 psig, variando la válvula de diafragma HV-11 observando el valor de presión en el manómetro PI-01.

En la figura 47 se puede observar la válvula de diafragma y en la figura 48 se observa el manómetro.

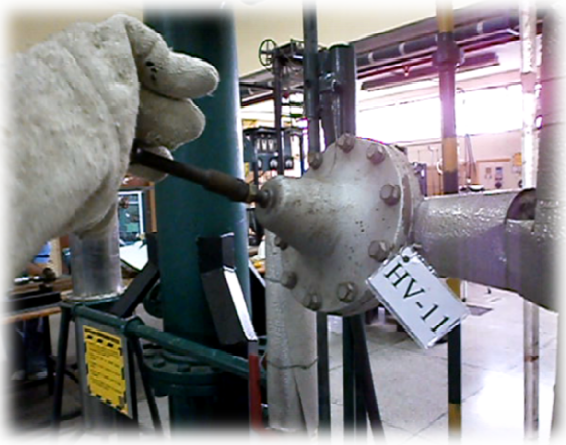


Figura 47. Válvula de diafragma HV-11

Fuente: Elaboración Propia

Figura 48. Manómetro PI-01

Fuente: Elaboración Propia

En el equipo Evaporador, se marcaron cuatro puntos alrededor del mismo para así tomar las mediciones de temperatura de pared como se muestra en la figura 49:

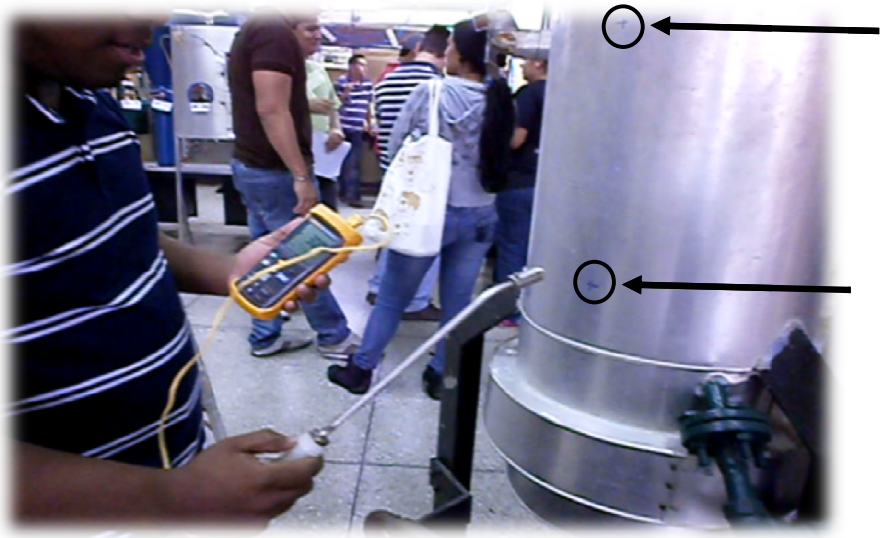


Figura 49. Puntos marcados en el Equipo Evaporador

Fuente: Elaboración Propia

Se tomaron mediciones en cada punto señalado en el equipo con la termocupla tipo K marca FLUKE, como se muestra a continuación:

Figura 50. (a) y (b). Ubicación de la termocupla tipo k en el primer punto más bajo del evaporador y el valor de temperatura en ese punto.

Fuente: Elaboración Propia

(a) **(b)**

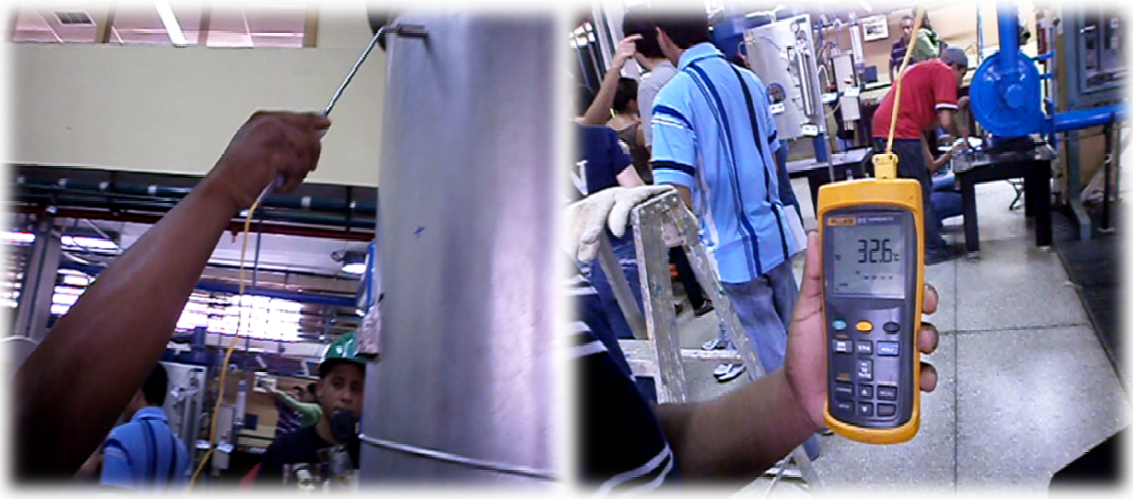
Figura 51. (a) y (b). Ubicación de la termocupla tipo k en el segundo punto del evaporador y el valor de temperatura en ese punto.

Fuente: Elaboración Propia

(a) **(b)**

Figura 52. (a) y (b). Ubicación de la termocupla tipo k en el tercer punto del evaporador y el valor de temperatura en ese punto.

Fuente: Elaboración Propia

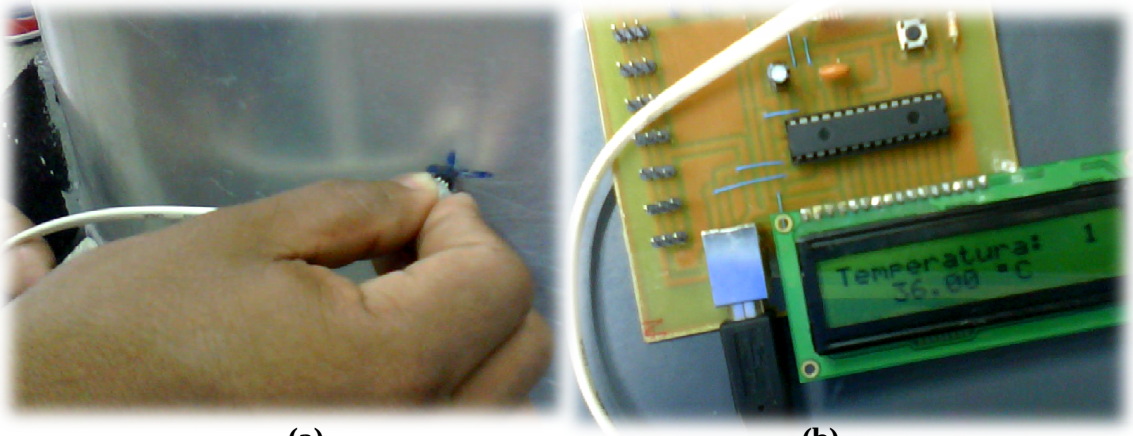


(a) **(b)**

Figura 53. (a) y (b). Ubicación de la termocupla tipo k en el cuarto punto del evaporador y el valor de temperatura en ese punto.

Fuente: Elaboración Propia

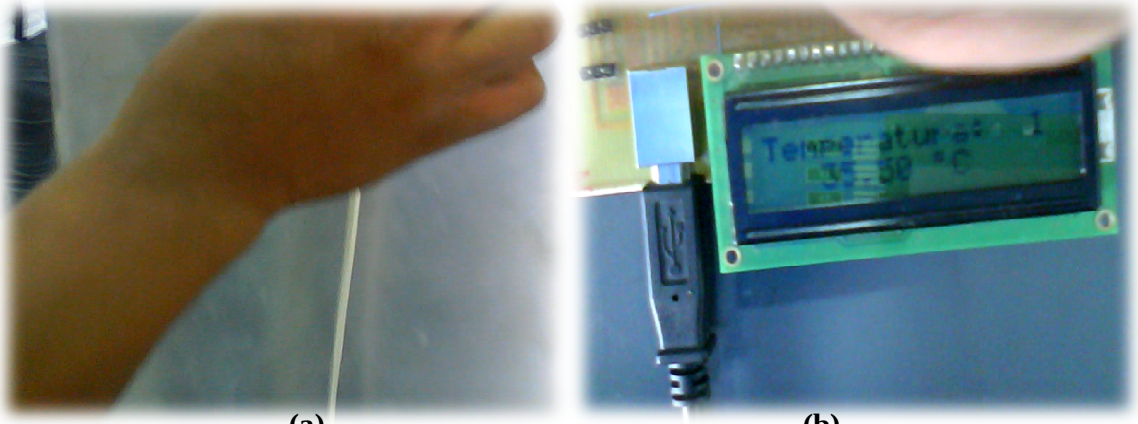
- Realizar mediciones de temperatura en los mismos puntos anteriores empleando los sensores digitales de temperatura 1-Wire®.



(a) **(b)**

Figura 54. (a) y (b). Ubicación del sensor en el primer punto más bajo del evaporador y el valor de temperatura en ese punto.

Fuente: Elaboración Propia



(a) **(b)**

Figura 55. (a) y (b). Ubicación del sensor en el segundo punto más bajo del evaporador y el valor de temperatura en ese punto.

Fuente: Elaboración Propia

❖ Comparar ambas mediciones y determinar si los resultados del sistema son correctos.

Estableciendo una tabla comparativa entre las dos primeras mediciones tanto con la termocupla tipo k y el termómetro de temperatura de alta precisión DS1820:

Valor Medido con Termocupla Tipo k	Valor Medido con Sensor DS1820 ó DS18S20	Diferencia entre Valores
36 °C	36 °C	0 °C
35.8 °C	35.5 °C	0.3 °C

Tabla 9: Tabla comparativa entre las mediciones con la termocupla tipo k y el sensor de temperatura DS1820.

Fuente: Elaboración Propia

Según la tabla 9, se puede observar que ambos dispositivos de medición de temperatura arrojaron valores cercanos entre sí.

En la primera medición dieron exactamente iguales a 36 °C, en la segunda medición hubo una diferencia de 0.3 °C, sin embargo se logró notar que si permanecíamos en contacto con la pared del equipo evaporador con el dispositivo de medición de temperatura, la temperatura seguía aumentando debido a que medida que transcurría el tiempo la temperatura aumentaba en la coraza del evaporador.

Cabe destacar que el material del filamento que posee la termocupla tipo k que sirve como contacto al momento de realizar la medición de temperatura de pared, es más vulnerable en lo que concierne a la transferencia de calor y por ende mucho más precisa su medición.

Además se puede notar en las características del sensor DS1820 que éste posee una precisión de ±0.5 °C en el rango de -10°C a 85°C, es por ello que no es posible obtener como medición valores como: 35.1, 35.2, 35.8, etc.

En términos generales se comprueba que el funcionamiento del sistema es correcto.

CAPÍTULO V
CONCLUSIONES

Se diseñó un sistema de medición de temperatura denominado SMT-Cy-Gus compuesto por una Tarjeta de Adquisición de Datos (TAD) y un software, por medio de la TAD es posible medir hasta en 15 puntos de temperatura por medio de los sensores de temperatura DS1820. Existen dos formas de visualizar los valores de temperatura medidos:

- Por medio de la pantalla LCD de 16x2 que contiene la TAD, en la cual muestra cantidad de sensores conectados al bus 1-wire® y el respectivo valor en °C de cada sensor.
- Por medio del software realizado, se permite apreciar en tiempo real el valor de temperatura en °C de cada sensor, así como también el serial único de cada dispositivo conectado al bus 1-wire®.

La TAD es alimentada únicamente por medio del conector USB tipo B, a su vez los sensores DS1820 conectados al bus 1-wire® son alimentados mediante una resistencia de pull-up de 2.2kΩ. Algunos logros obtenidos se presentan a continuación:

- Se realizó un estudio de las características actuales del proceso de medición de temperatura en el equipo, determinando que éste se realiza totalmente manual haciendo lenta la realización de las prácticas.

- Se seleccionó el sensor de temperatura que cumple con los requerimientos del sistema.

- Se estableció la comunicación entre el PIC18F2550 (Maestro) y los sensores DS1820 (Esclavos) bajo el protocolo 1-wire®.

- Se diseñó e implementó la TAD.

- Se creó el software en el entorno Visual Studio C# 2008, con capacidad de comunicación USB.

- Se estableció la comunicación entre la TAD y la PC (Visual Studio C# 2008) bajo el protocolo USB.

- Se puso en funcionamiento el sistema SMT-Cy-Gus en el Laboratorio de Ingeniería Química (LIQ) y se realizaron lecturas de la temperatura del equipo evaporador-condensador vertical.

- Se estableció la comparación entre las lecturas obtenidas a través de la termocupla tipo k y el sistema SMT-Cy-Gus, observando alta precisión entre ellas.

- Se añadieron las siguientes funcionalidades al software: Cálculo de temperatura promedio, almacenamiento de valores de temperatura en un archivo de texto, visualización de la gráfica de perfil de temperatura y cantidad de sensores conectados con sus respectivos seriales.

RECOMENDACIONES

Para el óptimo uso del sistema creado y para añadir mejoras al sistema, se recomienda lo siguiente:

- Realizar los agujeros necesarios en el equipo evaporador y fijar los sensores de temperatura 1-Wire®.
- Conectar la TAD a la PC sin ningún sensor instalado.
- Construir una caja a la TAD de manera de que se conserve su estado.
- Añadir al software la función de reconocer la ubicación de cada sensor en el equipo.
- Incluirle al sistema, medición de variables como: *Presión, nivel y humedad*.
- En caso de que se desee utilizar la TAD como dispositivo portátil, agregar al circuito de ésta, los componentes necesarios para que pueda ser alimentada con una batería.

ANEXO A

(Datos técnicos de los componentes electrónicos)

➢ **Microcontrolador 18F2550**

MICROCHIP PIC18F2455/2550/4455/4550

28/40/44-Pin, High-Performance, Enhanced Flash, USB Microcontrollers with nanoWatt Technology

Universal Serial Bus Features:

- USB V2.0 Compliant
- Low Speed (1.5 Mb/s) and Full Speed (12 Mb/s)
- Supports Control, Interrupt, Isochronous and Bulk Transfers
- Supports up to 32 Endpoints (16 bidirectional)
- 1-Kbyte Dual Access RAM for USB
- On-Chip USB Transceiver with On-Chip Voltage Regulator
- Interface for Off-Chip USB Transceiver
- Streaming Parallel Port (SPP) for USB streaming transfers (40/44-pin devices only)

Power-Managed Modes:

- Run: CPU on, peripherals on
- Idle: CPU off, peripherals on
- Sleep: CPU off, peripherals off
- Idle mode currents down to 5.8 µA typical
- Sleep mode currents down to 0.1 µA typical
- Timer1 Oscillator: 1.1 µA typical, 32 kHz, 2V
- Watchdog Timer: 2.1 µA typical
- Two-Speed Oscillator Start-up

Flexible Oscillator Structure:

- Four Crystal modes, including High Precision PLL for USB
- Two External Clock modes, up to 48 MHz
- Internal Oscillator Block:
  - 8 user-selectable frequencies, from 31 kHz to 8 MHz
  - User-tunable to compensate for frequency drift
- Secondary Oscillator using Timer1 @ 32 kHz
- Dual Oscillator options allow microcontroller and USB module to run at different clock speeds
- Fail-Safe Clock Monitor:
  - Allows for safe shutdown if any clock stops

Peripheral Highlights:

- High-Current Sink/Source: 25 mA/25 mA
- Three External Interrupts
- Four Timer modules (Timer0 to Timer3)
- Up to 2 Capture/Compare/PWM (CCP) modules:
  - Capture is 16-bit, max. resolution 5.2 ns (TCY/16)
  - Compare is 16-bit, max. resolution 83.3 ns (TCY)
  - PWM output: PWM resolution is 1 to 10-bit
- Enhanced Capture/Compare/PWM (ECCP) module:
  - Multiple output modes
  - Selectable polarity
  - Programmable dead time
  - Auto-shutdown and auto-restart
- Enhanced USART module:
  - LIN bus support
- Master Synchronous Serial Port (MSSP) module supporting 3-wire SPI (all 4 modes) and I^2C™ Master and Slave modes
- 10-bit, up to 13-channel Analog-to-Digital Converter module (A/D) with Programmable Acquisition Time
- Dual Analog Comparators with Input Multiplexing

Special Microcontroller Features:

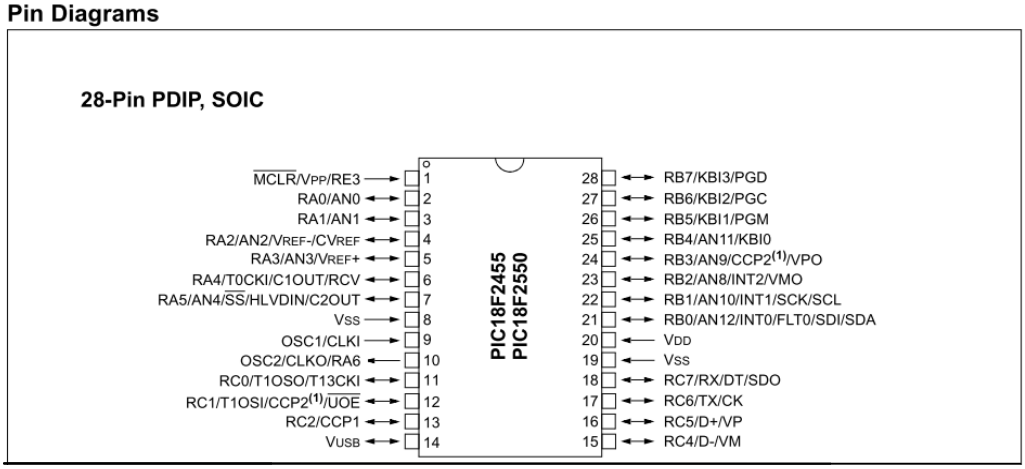
- C Compiler Optimized Architecture with optional Extended Instruction Set
- 100,000 Erase/Write Cycle Enhanced Flash Program Memory typical
- 1,000,000 Erase/Write Cycle Data EEPROM Memory typical
- Flash/Data EEPROM Retention: > 40 years
- Self-Programmable under Software Control
- Priority Levels for Interrupts
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 41 ms to 131s
- Programmable Code Protection
- Single-Supply 5V In-Circuit Serial Programming™ (ICSP™) via two pins
- In-Circuit Debug (ICD) via two pins
- Optional dedicated ICD/ICSP port (44-pin devices only)

• Wide Operating Voltage Range (2.0V to 5.5V)

Device	Program Memory		Data Memory		I/O	10-Bit A/D (ch)	CCP/ECCP (PWM)	SPP	MSSP		EAUSART	Comparators	Timers 8/16-Bit
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)					SPI	Master I^2C™			
PIC18F2455	24K	12288	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F2550	32K	16384	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F4455	24K	12288	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3
PIC18F4550	32K	16384	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3

Pin Diagrams

28-Pin PDIP, SOIC



PIC18F2455/2550/4455/4550

1.0 DEVICE OVERVIEW

This document contains device-specific information for the following devices:

- PIC18F2455
- PIC18F2550
- PIC18F4455
- PIC18F4550
- PIC18LF2455
- PIC18LF2550
- PIC18LF4455
- PIC18LF4550

This family of devices offers the advantages of all PIC18 microcontrollers – namely, high computational performance at an economical price – with the addition of high endurance, Enhanced Flash program memory. In addition to these features, the PIC18F2455/2550/4455/4550 family introduces design enhancements that make these microcontrollers a logical choice for many high-performance, power sensitive applications.

1.1 New Core Features

1.1.1 nanoWatt TECHNOLOGY

All of the devices in the PIC18F2455/2550/4455/4550 family incorporate a range of features that can significantly reduce power consumption during operation. Key items include:

- **Alternate Run Modes:** By clocking the controller from the Timer1 source or the internal oscillator block, power consumption during code execution can be reduced by as much as 90%.
- **Multiple Idle Modes:** The controller can also run with its CPU core disabled but the peripherals still active. In these states, power consumption can be reduced even further, to as little as 4% of normal operation requirements.
- **On-the-Fly Mode Switching:** The power-managed modes are invoked by user code during operation, allowing the user to incorporate power-saving ideas into their application's software design.
- **Low Consumption in Key Modules:** The power requirements for both Timer1 and the Watchdog Timer are minimized. See **Section 28.0 "Electrical Characteristics"** for values.

1.1.3 MULTIPLE OSCILLATOR OPTIONS AND FEATURES

All of the devices in the PIC18F2455/2550/4455/4550 family offer twelve different oscillator options, allowing users a wide range of choices in developing application hardware. These include:

- Four Crystal modes using crystals or ceramic resonators.
- Four External Clock modes, offering the option of using two pins (oscillator input and a divide-by-4 clock output) or one pin (oscillator input, with the second pin reassigned as general I/O).
- An internal oscillator block which provides an 8 MHz clock (±2% accuracy) and an INTRC source (approximately 31 kHz, stable over temperature and VDD), as well as a range of 6 user-selectable clock frequencies, between 125 kHz to 4 MHz, for a total of 8 clock frequencies. This option frees an oscillator pin for use as an additional general purpose I/O.
- A Phase Lock Loop (PLL) frequency multiplier, available to both the High-Speed Crystal and External Oscillator modes, which allows a wide range of clock speeds from 4 MHz to 48 MHz.
- Asynchronous dual clock operation, allowing the USB module to run from a high-frequency oscillator while the rest of the microcontroller is clocked from an internal low-power oscillator.

Besides its availability as a clock source, the internal oscillator block provides a stable reference source that gives the family additional features for robust operation:

- **Fail-Safe Clock Monitor:** This option constantly monitors the main clock source against a reference signal provided by the internal oscillator. If a clock failure occurs, the controller is switched to the internal oscillator block, allowing for continued low-speed operation or a safe application shutdown.
- **Two-Speed Start-up:** This option allows the internal oscillator to serve as the clock source from Power-on Reset, or wake-up from Sleep mode, until the primary clock source is available.

1.1.2 UNIVERSAL SERIAL BUS (USB)

Devices in the PIC18F2455/2550/4455/4550 family incorporate a fully featured Universal Serial Bus communications module that is compliant with the USB Specification Revision 2.0. The module supports both low-speed and full-speed communication for all supported data transfer types. It also incorporates its own on-chip transceiver and 3.3V regulator and supports the use of external transceivers and voltage regulators.

2.0 OSCILLATOR CONFIGURATIONS

2.1 Overview

Devices in the PIC18F2455/2550/4455/4550 family incorporate a different oscillator and microcontroller clock system than previous PIC18F devices. The addition of the USB module, with its unique requirements for a stable clock source, make it necessary to provide a separate clock source that is compliant with both USB low-speed and full-speed specifications.

To accommodate these requirements, PIC18F2455/2550/4455/4550 devices include a new clock branch to provide a 48 MHz clock for full-speed USB operation. Since it is driven from the primary clock source, an additional system of prescalers and postscalers has been added to accommodate a wide range of oscillator frequencies. An overview of the oscillator structure is shown in Figure 2-1.

Other oscillator features used in PIC18 enhanced microcontrollers, such as the internal oscillator block and clock switching, remain the same. They are discussed later in this chapter.

2.1.1 OSCILLATOR CONTROL

The operation of the oscillator in PIC18F2455/2550/4455/4550 devices is controlled through two Configuration registers and two control registers. Configuration registers, CONFIG1L and CONFIG1H, select the oscillator mode and USB prescaler/postscaler options. As Configuration bits, these are set when the device is programmed and left in that configuration until the device is reprogrammed.

The OSCCON register (Register 2-2) selects the Active Clock mode; it is primarily used in controlling clock switching in power-managed modes. Its use is discussed in **Section 2.4.1 "Oscillator Control Register"**.

The OSCTUNE register (Register 2-1) is used to trim

1. XT Crystal/Resonator
2. XTPLL Crystal/Resonator with PLL enabled
3. HS High-Speed Crystal/Resonator
4. HSPLL High-Speed Crystal/Resonator with PLL enabled
5. EC External Clock with FOSC/4 output
6. ECIO External Clock with I/O on RA6
7. ECPLL External Clock with PLL enabled and FOSC/4 output on RA6
8. ECPIO External Clock with PLL enabled, I/O on RA6
9. INTHS Internal Oscillator used as microcontroller clock source, HS Oscillator used as USB clock source
10. INTXT Internal Oscillator used as microcontroller clock source, XT Oscillator used as USB clock source
11. INTIO Internal Oscillator used as microcontroller clock source, EC Oscillator used as USB clock source, digital I/O on RA6
12. INTCKO Internal Oscillator used as microcontroller clock source, EC Oscillator used as USB clock source, FOSC/4 output on RA6

2.2.1 OSCILLATOR MODES AND USB OPERATION

Because of the unique requirements of the USB module, a different approach to clock operation is necessary. In previous PIC® devices, all core and peripheral clocks were driven by a single oscillator source; the usual sources were primary, secondary or the internal oscillator. With PIC18F2455/2550/4455/4550 devices, the primary oscillator becomes part of the USB module and cannot be associated to any other clock source. Thus, the USB module must be clocked from the primary clock source; however, the microcontroller core and other peripherals can be separately clocked from the second-

the INTRC frequency source, as well as select the low-frequency clock source that drives several special features. Its use is described in **Section 2.2.5.2 "OSCTUNE Register"**.

2.2 Oscillator Types

PIC18F2455/2550/4455/4550 devices can be operated in twelve distinct oscillator modes. In contrast with previous PIC18 enhanced microcontrollers, four of these modes involve the use of two oscillator types at once. Users can program the FOSC3:FOSC0 Configuration bits to select one of these modes:

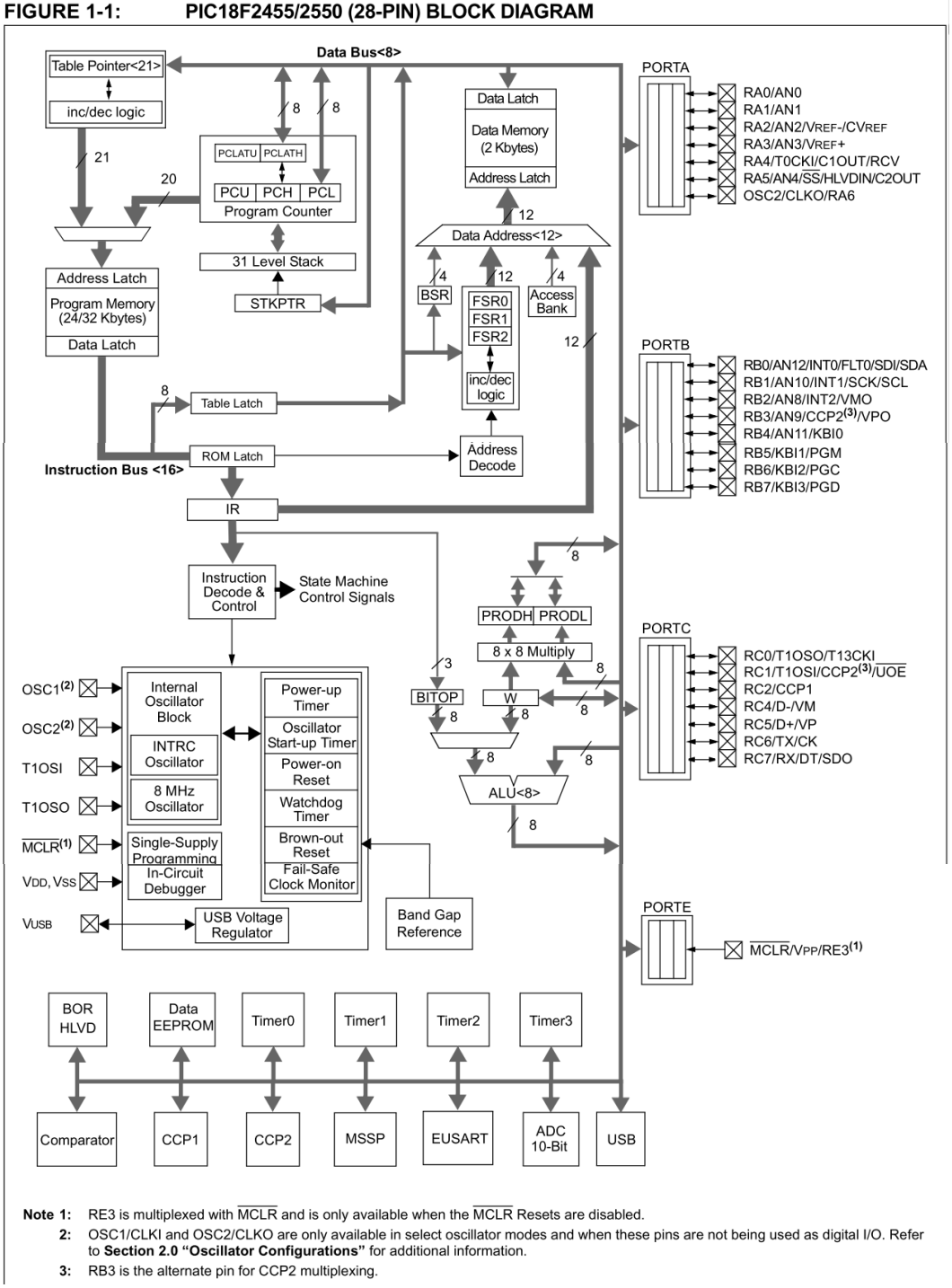
ary or internal oscillators as before.

Because of the timing requirements imposed by USB, an internal clock of either 6 MHz or 48 MHz is required while the USB module is enabled. Fortunately, the microcontroller and other peripherals are not required to run at this clock speed when using the primary oscillator. There are numerous options to achieve the USB module clock requirement and still provide flexibility for clocking the rest of the device from the primary oscillator source. These are detailed in **Section 2.3 "Oscillator Settings for USB"**.

TABLE 1-1: DEVICE FEATURES

Features	PIC18F2455	PIC18F2550	PIC18F4455	PIC18F4550
Operating Frequency	DC – 48 MHz	DC – 48 MHz	DC – 48 MHz	DC – 48 MHz
Program Memory (Bytes)	24576	32768	24576	32768
Program Memory (Instructions)	12288	16384	12288	16384
Data Memory (Bytes)	2048	2048	2048	2048
Data EEPROM Memory (Bytes)	256	256	256	256
Interrupt Sources	19	19	20	20
I/O Ports	Ports A, B, C, (E)	Ports A, B, C, (E)	Ports A, B, C, D, E	Ports A, B, C, D, E
Timers	4	4	4	4
Capture/Compare/PWM Modules	2	2	1	1
Enhanced Capture/ Compare/PWM Modules	0	0	1	1
Serial Communications	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART
Universal Serial Bus (USB) Module	1	1	1	1
Streaming Parallel Port (SPP)	No	No	Yes	Yes
10-Bit Analog-to-Digital Module	10 Input Channels	10 Input Channels	13 Input Channels	13 Input Channels
Comparators	2	2	2	2
Resets (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT
Programmable Low-Voltage Detect	Yes	Yes	Yes	Yes
Programmable Brown-out Reset	Yes	Yes	Yes	Yes
Instruction Set	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled
Packages	28-pin PDIP 28-pin SOIC	28-pin PDIP 28-pin SOIC	40-pin PDIP 44-pin QFN 44-pin TQFP	40-pin PDIP 44-pin QFN 44-pin TQFP

FIGURE 1-1: PIC18F2455/2550 (28-PIN) BLOCK DIAGRAM



Note 1: RE3 is multiplexed with $\overline{\text{MCLR}}$ and is only available when the $\overline{\text{MCLR}}$ Resets are disabled.

2: OSC1/CLKI and OSC2/CLKO are only available in select oscillator modes and when these pins are not being used as digital I/O. Refer to **Section 2.0 "Oscillator Configurations"** for additional information.

3: RB3 is the alternate pin for CCP2 multiplexing.

- **Termómetro Digital 1-Wire® de alta precisión D18S20**

DALLAS SEMICONDUCTOR

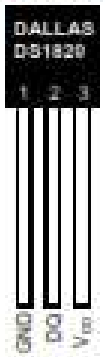
DS18S20
High Precision 1-Wire® Digital Thermometer

www.dalsemi.com

FEATURES

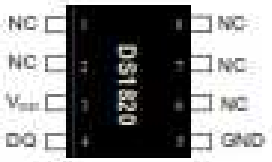
- Unique 1-wire interface requires only one port pin for communication
- Each device has a unique 64-bit serial code stored in an on-board ROM
- Multi-drop capability simplifies distributed temperature sensing applications
- Requires no external components
- Can be powered from data line. Power supply range is 3.0V to 5.5V
- Measures temperatures from –55°C to +125°C (–67°F to +257°F)
- ±0.5°C accuracy from –10°C to +85°C
- 9-bit thermometer resolution
- Converts temperature in 750 ms (max.)
- User-definable nonvolatile alarm settings
- Alarm search command identifies and addresses devices whose temperature is outside of programmed limits (temperature alarm condition)
- Applications include thermostatic controls, industrial systems, consumer products, thermometers, or any thermally sensitive system

PIN ASSIGNMENT



(BOTTOM VIEW)

TO-92
(DS18S20)



8-pin 150-mil SOIC
(DS18S20Z)

PIN DESCRIPTION

GND - Ground
DQ - Data In/Out
V_{DD} - Power Supply Voltage
NC - No Connect

DESCRIPTION

The DS18S20 Digital Thermometer provides 9–bit centigrade temperature measurements and has an alarm function with nonvolatile user-programmable upper and lower trigger points. The DS18S20 communicates over a 1-wire bus that by definition requires only one data line (and ground) for communication with a central microprocessor. It has an operating temperature range of –55°C to +125°C and is accurate to ±0.5°C over the range of –10°C to +85°C. In addition, the DS18S20 can derive power directly from the data line ("parasite power"), eliminating the need for an external power supply.

Each DS18S20 has a unique 64-bit serial code, which allows multiple DS18S20s to function on the same 1–wire bus; thus, it is simple to use one microprocessor to control many DS18S20s distributed over a large area. Applications that can benefit from this feature include HVAC environmental controls, temperature monitoring systems inside buildings, equipment or machinery, and process monitoring and control systems.

DETAILED PIN DESCRIPTIONS Table 1

8-PIN SOIC*	TO-92	SYMBOL	DESCRIPTION
5	1	GND	**Ground.**
4	2	DQ	**Data Input/Output pin.** Open-drain 1-wire interface pin. Also provides power to the device when used in parasite power mode (see "Parasite Power" section.)
3	3	V_{DD}	**Optional V_{DD} pin.** V_{DD} must be grounded for operation in parasite power mode.

*All pins not specified in this table are "No Connect" pins.

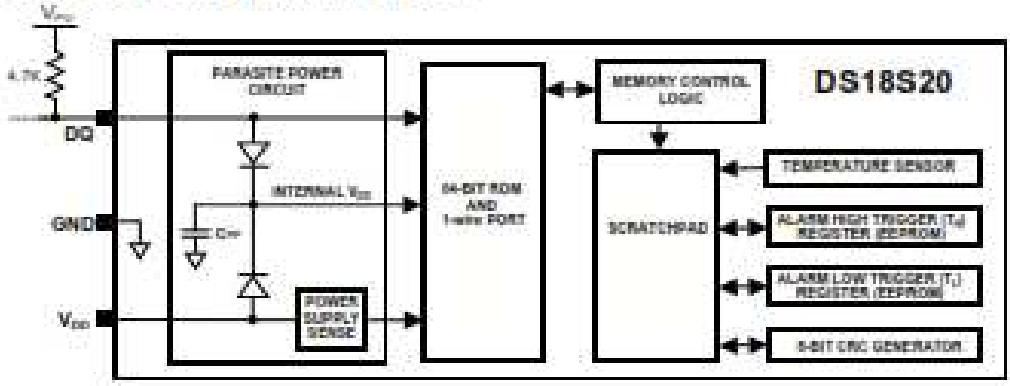
OVERVIEW

Figure 1 shows a block diagram of the DS18S20, and pin descriptions are given in Table 1. The 64-bit ROM stores the device's unique serial code. The scratchpad memory contains the 2-byte temperature register that stores the digital output from the temperature sensor. In addition, the scratchpad provides access to the 1-byte upper and lower alarm trigger registers (T_H and T_L). The T_H and T_L registers are nonvolatile (EEPROM), so they will retain data when the device is powered down.

The DS18S20 uses Dallas' exclusive 1-wire bus protocol that implements bus communication using one control signal. The control line requires a weak pullup resistor since all devices are linked to the bus via a 3-state or open-drain port (the DQ pin in the case of the DS18S20). In this bus system, the microprocessor (the master device) identifies and addresses devices on the bus using each device's unique 64-bit code. Because each device has a unique code, the number of devices that can be addressed on one bus is virtually unlimited. The 1-wire bus protocol, including detailed explanations of the commands and "time slots," is covered in the 1-WIRE BUS SYSTEM section of this datasheet.

Another feature of the DS18S20 is the ability to operate without an external power supply. Power is instead supplied through the 1-wire pullup resistor via the DQ pin when the bus is high. The high bus signal also charges an internal capacitor (C_{PP}), which then supplies power to the device when the bus is low. This method of deriving power from the 1-wire bus is referred to as "parasite power." As an alternative, the DS18S20 may also be powered by an external supply on V_{DD}.

DS18S20 BLOCK DIAGRAM Figure 1



OPERATION – MEASURING TEMPERATURE

The core functionality of the DS18S20 is its direct-to-digital temperature sensor. The temperature sensor output has 9-bit resolution, which corresponds to 0.5°C steps. The DS18S20 powers-up in a low-power idle state; to initiate a temperature measurement and A-to-D conversion, the master must issue a Convert T [44h] command. Following the conversion, the resulting thermal data is stored in the 2-byte temperature register in the scratchpad memory and the DS18S20 returns to its idle state. If the DS18S20 is powered by an external supply, the master can issue "read time slots" (see the 1-WIRE BUS SYSTEM section) after the Convert T command and the DS18S20 will respond by transmitting 0 while the temperature conversion is in progress and 1 when the conversion is done. If the DS18S20 is powered with parasite power, this notification technique cannot be used since the bus must be pulled high by a strong pullup during the entire temperature conversion. The bus requirements for parasite power are explained in detail in the POWERING THE DS18S20 section of this datasheet.

The DS18S20 output data is calibrated in degrees centigrade; for Fahrenheit applications, a lookup table or conversion routine must be used. The temperature data is stored as a 16-bit sign-extended two's complement number in the temperature register (see Figure 2). The sign bits (S) indicate if the temperature is positive or negative: for positive numbers S = 0 and for negative numbers S = 1. Table 2 gives examples of digital output data and the corresponding temperature reading.

Resolutions greater than 9 bits can be calculated using the data from the temperature, COUNT REMAIN and COUNT PER °C registers in the scratchpad. Note that the COUNT PER °C register is hard-wired to 16 (10h). After reading the scratchpad, the TEMP_READ value is obtained by truncating the 0.5°C bit (bit 0) from the temperature data (see Figure 2). The extended resolution temperature can then be calculated using the following equation:

$$TEMPERATURE = TEMP\_READ - 0.25 + \frac{COUNT\_PER\_C - COUNT\_REMAIN}{COUNT\_PER\_C}$$

Additional information about high-resolution temperature calculations can be found in Application Note 105: "High Resolution Temperature Measurement with Dallas Direct-to-Digital Temperature Sensors".

TEMPERATURE REGISTER FORMAT Figure 2

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
LS Byte	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}
	bit 15	**bit 14**	**bit 13**	**bit 12**	**bit 11**	**bit 10**	**bit 9**	**bit 8**
MS Byte	S	S	S	S	S	S	S	S

TEMPERATURE/DATA RELATIONSHIP Table 2

TEMPERATURE	DIGITAL OUTPUT (Binary)	DIGITAL OUTPUT (Hex)
+85.0°C*	0000 0000 1010 1010	00AAh
+25.0°C	0000 0000 0011 0010	0032h
+0.5°C	0000 0000 0000 0001	0001h
0°C	0000 0000 0000 0000	0000h
-0.5°C	1111 1111 1111 1111	FFFFh
-25.0°C	1111 1111 1100 1110	FFCEh
-55.0°C	1111 1111 1001 0010	FF92h

*The power-on reset value of the temperature register is +85°C

OPERATION – ALARM SIGNALING

After the DS18S20 performs a temperature conversion, the temperature value is compared to the user-defined two's complement alarm trigger values stored in the 1-byte T_H and T_L registers (see Figure 3). The sign bit (S) indicates if the value is positive or negative: for positive numbers S = 0 and for negative numbers S = 1. The T_H and T_L registers are nonvolatile (EEPROM) so they will retain data when the device is powered down. T_H and T_L can be accessed through bytes 2 and 3 of the scratchpad as explained in the MEMORY section of this datasheet.

T_H AND T_L REGISTER FORMAT Figure 3

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
S	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Only bits 8 through 1 of the temperature register are used in the T_H and T_L comparison since T_H and T_L are 8-bit registers. If the result of a temperature measurement is higher than T_H or lower than T_L, an alarm condition exists and an alarm flag is set inside the DS18S20. This flag is updated after every temperature measurement; therefore, if the alarm condition goes away, the flag will be turned off after the next temperature conversion.

The master device can check the alarm flag status of all DS18S20s on the bus by issuing an Alarm Search [ECh] command. Any DS18S20s with a set alarm flag will respond to the command, so the master can determine exactly which DS18S20s have experienced an alarm condition. If an alarm condition exists and the T_H or T_L settings have changed, another temperature conversion should be done to validate the alarm condition.

POWERING THE DS18S20

The DS18S20 can be powered by an external supply on the V_{DD} pin, or it can operate in "parasite power" mode, which allows the DS18S20 to function without a local external supply. Parasite power is very useful for applications that require remote temperature sensing or that are very space constrained. Figure 1 shows the DS18S20's parasite-power control circuitry, which "steals" power from the 1-wire bus via the DQ pin when the bus is high. The stolen charge powers the DS18S20 while the bus is high, and some of the charge is stored on the parasite power capacitor (C_{PP}) to provide power when the bus is low. When the DS18S20 is used in parasite power mode, the V_{DD} pin must be connected to ground.

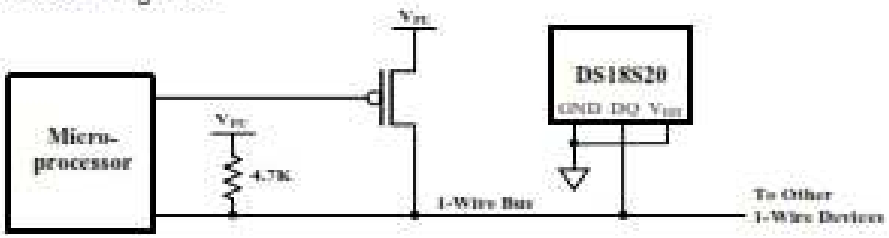
In parasite power mode, the 1-wire bus and C_{PP} can provide sufficient current to the DS18S20 for most operations as long as the specified timing and voltage requirements are met (refer to the DC ELECTRICAL CHARACTERISTICS and the AC ELECTRICAL CHARACTERISTICS sections of this data sheet). However, when the DS18S20 is performing temperature conversions or copying data from the scratchpad memory to EEPROM, the operating current can be as high as 1.5 mA. This current can cause an unacceptable voltage drop across the weak 1-wire pullup resistor and is more current than can be supplied by C_{PP}. To assure that the DS18S20 has sufficient supply current, it is necessary to provide a strong pullup on the 1-wire bus whenever temperature conversions are taking place or data is being copied from the scratchpad to EEPROM. This can be accomplished by using a MOSFET to pull the bus directly to the rail as shown in Figure 4. The 1-wire bus must be switched to the strong pullup within 10 µs (max) after a Convert T [44h] or Copy Scratchpad [48h] command is issued, and the bus must be held high by the pullup for the duration of the conversion (t_{CONV}) or data transfer (t_{WR} = 10 ms). No other activity can take place on the 1-wire bus while the pullup is enabled.

The DS18S20 can also be powered by the conventional method of connecting an external power supply to the V_{DD} pin, as shown in Figure 5. The advantage of this method is that the MOSFET pullup is not required, and the 1–wire bus is free to carry other traffic during the temperature conversion time.

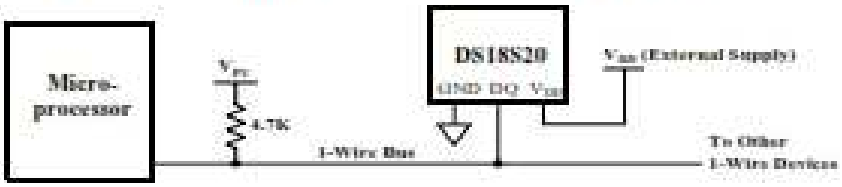
The use of parasite power is not recommended for temperatures above 100°C since the DS18S20 may not be able to sustain communications due to the higher leakage currents that can exist at these temperatures. For applications in which such temperatures are likely, it is strongly recommended that the DS18S20 be powered by an external power supply.

In some situations the bus master may not know whether the DS18S20s on the bus are parasite powered or powered by external supplies. The master needs this information to determine if the strong bus pullup should be used during temperature conversions. To get this information, the master can issue a Skip ROM [CCh] command followed by a Read Power Supply [B4h] command followed by a "read time slot". During the read time slot, parasite powered DS18S20s will pull the bus low, and externally powered DS18S20s will let the bus remain high. If the bus is pulled low, the master knows that it must supply the strong pullup on the 1-wire bus during temperature conversions.

SUPPLYING THE PARASITE-POWERED DS18S20 DURING TEMPERATURE CONVERSIONS Figure 4



POWERING THE DS18S20 WITH AN EXTERNAL SUPPLY Figure 5



64-BIT LASERED ROM CODE

Each DS18S20 contains a unique 64-bit code (see Figure 6) stored in ROM. The least significant 8 bits of the ROM code contain the DS18S20's 1-wire family code: 10h. The next 48 bits contain a unique serial number. The most significant 8 bits contain a cyclic redundancy check (CRC) byte that is calculated from the first 56 bits of the ROM code. A detailed explanation of the CRC bits is provided in the CRC GENERATION section. The 64-bit ROM code and associated ROM function control logic allow the DS18S20 to operate as a 1-wire device using the protocol detailed in the 1-WIRE BUS SYSTEM section of this datasheet.

64-BIT LASERED ROM CODE Figure 6

8-BIT CRC		48-BIT SERIAL NUMBER		8-BIT FAMILY CODE (10h)	
MSB	LSB	MSB	LSB	MSB	LSB

MEMORY

The DS18S20's memory is organized as shown in Figure 7. The memory consists of an SRAM scratchpad with nonvolatile EEPROM storage for the high and low alarm trigger registers (T_H and T_L). Note that if the DS18S20 alarm function is not used, the T_H and T_L registers can serve as general-purpose memory. All memory commands are described in detail in the DS18S20 FUNCTION COMMANDS section.

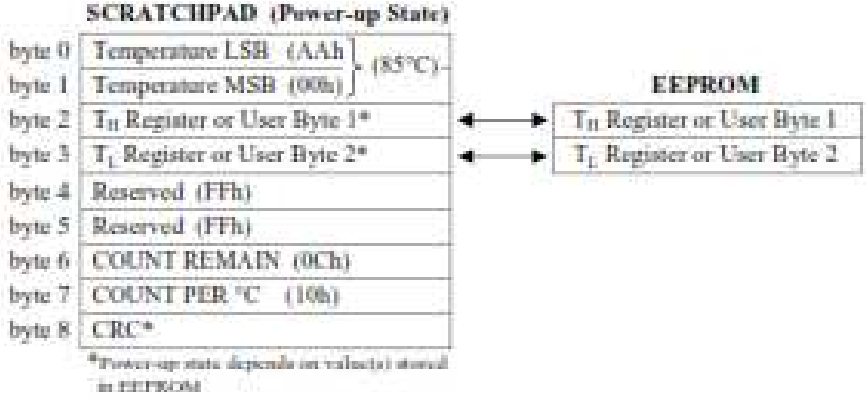
Byte 0 and byte 1 of the scratchpad contain the LSB and the MSB of the temperature register, respectively. These bytes are read-only. Bytes 2 and 3 provide access to T_H and T_L registers. Bytes 4 and 5 are reserved for internal use by the device and cannot be overwritten; these bytes will return all 1s when read. Bytes 6 and 7 contain the COUNT REMAIN and COUNT PER °C registers, which can be used to calculate extended resolution results as explained in the OPERATION – MEASURING TEMPERATURE section.

Byte 8 of the scratchpad is read-only and contains the cyclic redundancy check (CRC) code for bytes 0 through 7 of the scratchpad. The DS18S20 generates this CRC using the method described in the CRC GENERATION section.

Data is written to bytes 2 and 3 of the scratchpad using the Write Scratchpad [4Eh] command; the data must be transmitted to the DS18S20 starting with the least significant bit of byte 2. To verify data integrity, the scratchpad can be read (using the Read Scratchpad [BEh] command) after the data is written. When reading the scratchpad, data is transferred over the 1-wire bus starting with the least significant bit of byte 0. To transfer the T_H and T_L data from the scratchpad to EEPROM, the master must issue the Copy Scratchpad [48h] command.

Data in the EEPROM registers is retained when the device is powered down; at power-up the EEPROM data is reloaded into the corresponding scratchpad locations. Data can also be reloaded from EEPROM to the scratchpad at any time using the Recall E^2 [B8h] command. The master can issue "read time slots" (see the 1-WIRE BUS SYSTEM section) following the Recall E^2 command and the DS18S20 will indicate the status of the recall by transmitting 0 while the recall is in progress and 1 when the recall is done.

DS18S20 MEMORY MAP



INITIALIZATION

All transactions on the 1-wire bus begin with an initialization sequence. The initialization sequence consists of a reset pulse transmitted by the bus master followed by presence pulse(s) transmitted by the slave(s). The presence pulse lets the bus master know that slave devices (such as the DS18S20) are on the bus and are ready to operate. Timing for the reset and presence pulses is detailed in the 1-WIRE SIGNALING section.

ROM COMMANDS

After the bus master has detected a presence pulse, it can issue a ROM command. These commands operate on the unique 64–bit ROM codes of each slave device and allow the master to single out a specific device if many are present on the 1-wire bus. These commands also allow the master to determine how many and what types of devices are present on the bus or if any device has experienced an alarm condition. There are five ROM commands, and each command is 8 bits long. The master device must issue an appropriate ROM command before issuing a DS18S20 function command. A flowchart for operation of the ROM commands is shown in Figure 14.

SEARCH ROM [F0h]

When a system is initially powered up, the master must identify the ROM codes of all slave devices on the bus, which allows the master to determine the number of slaves and their device types. The master learns the ROM codes through a process of elimination that requires the master to perform a Search ROM cycle (i.e., Search ROM command followed by data exchange) as many times as necessary to identify all of the slave devices. If there is only one slave on the bus, the simpler Read ROM command (see below) can be used in place of the Search ROM process. For a detailed explanation of the Search ROM procedure, refer to the iButton Book of Standards at www.ibutton.com/ibuttons/standard.pdf. After every Search ROM cycle, the bus master must return to Step 1 (Initialization) in the transaction sequence.

READ ROM [33h]

This command can only be used when there is one slave on the bus. It allows the bus master to read the slave's 64-bit ROM code without using the Search ROM procedure. If this command is used when there is more than one slave present on the bus, a data collision will occur when all the slaves attempt to respond at the same time.

MATCH ROM [55h]

The match ROM command followed by a 64–bit ROM code sequence allows the bus master to address a specific slave device on a multi-drop or single-drop bus. Only the slave that exactly matches the 64–bit ROM code sequence will respond to the function command issued by the master; all other slaves on the bus will wait for a reset pulse.

SKIP ROM [CCh]

The master can use this command to address all devices on the bus simultaneously without sending out any ROM code information. For example, the master can make all DS18S20s on the bus perform simultaneous temperature conversions by issuing a Skip ROM command followed by a Convert T [44h] command. Note, however, that the Skip ROM command can only be followed by the Read Scratchpad [BEh] command when there is one slave on the bus. This sequence saves time by allowing the master to read from the device without sending its 64–bit ROM code. This sequence will cause a data collision on the bus if there is more than one slave since multiple devices will attempt to transmit data simultaneously.

ALARM SEARCH [ECh]

The operation of this command is identical to the operation of the Search ROM command except that only slaves with a set alarm flag will respond. This command allows the master device to determine if any DS18S20s experienced an alarm condition during the most recent temperature conversion. After

every Alarm Search cycle (i.e., Alarm Search command followed by data exchange), the bus master must return to Step 1 (Initialization) in the transaction sequence. Refer to the OPERATION – ALARM SIGNALING section for an explanation of alarm flag operation.

DS18S20 FUNCTION COMMANDS

After the bus master has used a ROM command to address the DS18S20 with which it wishes to communicate, the master can issue one of the DS18S20 function commands. These commands allow the master to write to and read from the DS18S20's scratchpad memory, initiate temperature conversions and determine the power supply mode. The DS18S20 function commands, which are described below, are summarized in Table 4 and illustrated by the flowchart in Figure 15.

CONVERT T [44h]

This command initiates a single temperature conversion. Following the conversion, the resulting thermal data is stored in the 2-byte temperature register in the scratchpad memory and the DS18S20 returns to its low-power idle state. If the device is being used in parasite power mode, within 10 µs (max) after this command is issued the master must enable a strong pullup on the 1-wire bus for the duration of the conversion (t_{conv}) as described in the POWERING THE DS18S20 section. If the DS18S20 is powered by an external supply, the master can issue read time slots after the Convert T command and the DS18S20 will respond by transmitting 0 while the temperature conversion is in progress and 1 when the conversion is done. In parasite power mode this notification technique cannot be used since the bus is pulled high by the strong pullup during the conversion.

WRITE SCRATCHPAD [4Eh]

This command allows the master to write 2 bytes of data to the DS18S20's scratchpad. The first byte is written into the T_H register (byte 2 of the scratchpad), and the second byte is written into the T_L register (byte 3 of the scratchpad). Data must be transmitted least significant bit first. Both bytes MUST be written before the master issues a reset, or the data may be corrupted.

READ SCRATCHPAD [BEh]

This command allows the master to read the contents of the scratchpad. The data transfer starts with the least significant bit of byte 0 and continues through the scratchpad until the 9th byte (byte 8 – CRC) is read. The master may issue a reset to terminate reading at any time if only part of the scratchpad data is needed.

COPY SCRATCHPAD [48h]

This command copies the contents of the scratchpad T_H and T_L registers (bytes 2 and 3) to EEPROM. If the device is being used in parasite power mode, within 10 µs (max) after this command is issued the master must enable a strong pullup on the 1-wire bus for at least 10 ms as described in the POWERING THE DS18S20 section.

RECALL E^2 [B8h]

This command recalls the alarm trigger values (T_H and T_L) from EEPROM and places the data in bytes 2 and 3, respectively, in the scratchpad memory. The master device can issue read time slots following the Recall E^2 command and the DS18S20 will indicate the status of the recall by transmitting 0 while the recall is in progress and 1 when the recall is done. The recall operation happens automatically at power-up, so valid data is available in the scratchpad as soon as power is applied to the device.

READ POWER SUPPLY [B4h]

The master device issues this command followed by a read time slot to determine if any DS18S20s on the bus are using parasite power. During the read time slot, parasite powered DS18S20s will pull the bus low, and externally powered DS18S20s will let the bus remain high. Refer to the POWERING THE DS18S20 section for usage information for this command.

DS18S20 OPERATION EXAMPLE 1

In this example there are multiple DS18S20s on the bus and they are using parasite power. The bus master initiates a temperature conversion in a specific DS18S20 and then reads its scratchpad and recalculates the CRC to verify the data.

MASTER MODE	DATA (LSB FIRST)	COMMENTS
TX	Reset	Master issues reset pulse.
RX	Presence	DS18S20s respond with presence pulse.
TX	55h	Master issues Match ROM command.
TX	64-bit ROM code	Master sends DS18S20 ROM code.
TX	44h	Master issues Convert T command.
TX	DQ line held high by strong pullup	Master applies strong pullup to DQ for the duration of the conversion (t_{conv}).
TX	Reset	Master issues reset pulse.
RX	Presence	DS18S20s respond with presence pulse.
TX	55h	Master issues Match ROM command.
TX	64-bit ROM code	Master sends DS18S20 ROM code.
TX	BEh	Master issues Read Scratchpad command.
RX	9 data bytes	Master reads entire scratchpad including CRC. The master then recalculates the CRC of the first eight data bytes from the scratchpad and compares the calculated CRC with the read CRC (byte 9). If they match, the master continues; if not, the read operation is repeated.

DS18S20 OPERATION EXAMPLE 2

In this example there is only one DS18S20 on the bus and it is using parasite power. The master writes to the T_H and T_L registers in the DS18S20 scratchpad and then reads the scratchpad and recalculates the CRC to verify the data. The master then copies the scratchpad contents to EEPROM.

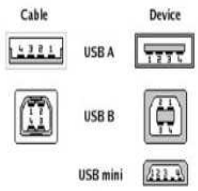
MASTER MODE	DATA (LSB FIRST)	COMMENTS
TX	Reset	Master issues reset pulse.
RX	Presence	DS18S20 responds with presence pulse.
TX	CCh	Master issues Skip ROM command.
TX	4Eh	Master issues Write Scratchpad command.
TX	2 data bytes	Master sends two data bytes to scratchpad (T_H and T_L).
TX	Reset	Master issues reset pulse.
RX	Presence	DS18S20 responds with presence pulse.
TX	CCh	Master issues Skip ROM command.
TX	BEh	Master issues Read Scratchpad command.
RX	9 data bytes	Master reads entire scratchpad including CRC. The master then recalculates the CRC of the first eight data bytes from the scratchpad and compares the calculated CRC with the read CRC (byte 9). If they match, the master continues; if not, the read operation is repeated.
TX	Reset	Master issues reset pulse.
RX	Presence	DS18S20 responds with presence pulse.
TX	CCh	Master issues Skip ROM command.
TX	48h	Master issues Copy Scratchpad command.
TX	DQ line held high by strong pullup	Master applies strong pullup to DQ for at least 10 ms while copy operation is in progress.

DS18S20 OPERATION EXAMPLE 3

In this example there is only one DS18S20 on the bus and it is using parasite power. The bus master initiates a temperature conversion then reads the DS18S20 scratchpad and calculates a higher resolution result using the data from the temperature, COUNT REMAIN and COUNT PER °C registers.

MASTER MODE	DATA (LSB FIRST)	COMMENTS
TX	Reset	Master issues reset pulse.
TR	Presence	DS18S20 responds with presence pulse.
TX	CCh	Master issues Skip ROM command.
TX	44h	Master issues Convert T command.
TX	DQ line held high by strong pullup	Master applies strong pullup to DQ for the duration of the conversion (t_{CONV}).
TX	Reset	Master issues reset pulse.
RX	Presence	DS18S20 responds with presence pulse.
TX	CCh	Master issues Skip ROM command.
TX	BEh	Master issues Read Scratchpad command.
RX	9 data bytes	Master reads entire scratchpad including CRC. The master then recalculates the CRC of the first eight data bytes from the scratchpad and compares the calculated CRC with the read CRC (byte 9). If they match, the master continues; if not, the read operation is repeated. The master also calculates the TEMP_READ value and stores the contents of the COUNT REMAIN and COUNT PER °C registers.
TX	Reset	Master issues reset pulse.
RX	Presence	DS18S20 responds with presence pulse.
-	-	CPU calculates extended resolution temperature using the equation in the OPERATION - MEASURING TEMPERATURE section of this datasheet.

- **Pines de los tipos de conectores USB**



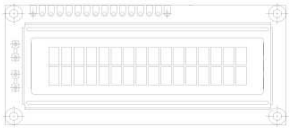
Pin	Signal	Color	Description
1	VCC		+5V
2	D-		Data -
3	D+		Data +
4	GND		Ground

- **Pantalla LCD de 16X2**

VISHAY

LCD-016M002B
Vishay

16 x 2 Character LCD



FEATURES

- 5 x 8 dots with cursor
- Built-in controller (KS 0066 or Equivalent)
- + 5V power supply (Also available for + 3V)
- 1/16 duty cycle
- B/L to be driven by pin 1, pin 2 or pin 15, pin 16 or A.K (LED)
- N.V. optional for + 3V power supply

MECHANICAL DATA

ITEM	STANDARD VALUE	UNIT
Module Dimension	80.0 x 36.0	mm
Viewing Area	66.0 x 16.0	mm
Dot Size	0.56 x 0.66	mm
Character Size	2.96 x 5.56	mm

ABSOLUTE MAXIMUM RATING

ITEM	SYMBOL	STANDARD VALUE			UNIT
		MIN.	TYP.	MAX.	
Power Supply	VDD-VSS	- 0.3	–	7.0	V
Input Voltage	VI	- 0.3	–	VDD	V

NOTE: VSS = 0 Volt, VDD = 5.0 Volt

ELECTRICAL SPECIFICATIONS

ITEM	SYMBOL	CONDITION		STANDARD VALUE			UNIT
				MIN.	TYP.	MAX.	
Input Voltage	VDD	VDD = + 5V		4.7	5.0	5.3	V
		VDD = + 3V		2.7	3.0	5.3	V
Supply Current	IDD	VDD = 5V		–	1.2	3.0	mA
Recommended LC Driving Voltage for Normal Temp. Version Module	VDD - V0	- 20 °C		–	–	–	V
		0°C		4.2	4.8	5.1	
		25°C		3.8	4.2	4.6	
		50°C		3.6	4.0	4.4	
		70°C		–	–	–	
LED Forward Voltage	VF	25°C		–	4.2	4.6	V
LED Forward Current	IF	25°C	Array	–	130	260	mA
			Edge	–	20	40	
EL Power Supply Current	IEL	Vel = 110VAC:400Hz		–	–	5.0	mA

DISPLAY CHARACTER ADDRESS CODE:

Display Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
DD RAM Address	00	01														0F
DD RAM Address	40	41														4F

PIN NUMBER	SYMBOL	FUNCTION
1	Vss	GND
2	Vdd	+ 3V or + 5V
3	Vo	Contrast Adjustment
4	RS	H/L Register Select Signal
5	R/$\overline{W}$	H/L Read/Write Signal
6	E	H →L Enable Signal
7	DB0	H/L Data Bus Line
8	DB1	H/L Data Bus Line
9	DB2	H/L Data Bus Line
10	DB3	H/L Data Bus Line
11	DB4	H/L Data Bus Line
12	DB5	H/L Data Bus Line
13	DB6	H/L Data Bus Line
14	DB7	H/L Data Bus Line
15	A/Vee	+ 4.2V for LED/Negative Voltage Output
16	K	Power Supply for B/L (OV)

BIBLIOGRAFÍA

REFERENCIAS BIBLIOGRÁFICAS Y ELECTRÓNICAS

[1] **Lemus Henry J e Istillarte Carlos L.** (2006). Desarrollo de Practicas para el laboratorio de microprocesadores utilizando microcontroladores. Trabajo especial de grado para optar al título de ingeniero electricista. Universidad de Carabobo. Carabobo, Venezuela.

[2] **Castro Zambrano José R. y Díaz Mattern Jessica J.** (2007). Desarrollo de instrumentación virtual con fines didácticos empleando controles ActiveX para la utilización de circuitos integrados con capacidad de comunicación 1-Wire®. Trabajo especial de grado para optar al título de ingeniero electrónico. UNEFA. Maracay, Venezuela.

[3] **José M. Rodríguez S.** (2008). Incorporación de la red 1-Wire® a los Procesos de automatización industrial, basados en el estándar "OPC" OLE para el control de procesos. Trabajo especial de grado para optar al título de magíster en Ingeniería Eléctrica. Universidad de Carabobo. Carabobo, Venezuela.

[4] **Eduardo Antonio Rojas Nastrucci y José Antonio Vargas Peiko.** (2009). Construcción de un electroencefalógrafo portátil. Trabajo especial de grado para optar al título de ingeniero electricista. Universidad de Carabobo. Carabobo, Venezuela.

[5] **Hernández Humberto y Pulido Gregory.** (2009). Desarrollo de un tacómetro digital con comunicación USB para los motores del laboratorio de máquinas eléctricas de la Facultad de Ingeniería en la Universidad de Carabobo. Trabajo especial de grado para optar al título de ingeniero electricista. Universidad de Carabobo. Carabobo, Venezuela.

[6] **Cárdenas R. Jorge G y Márquez B. Navier E.** (2009). Desarrollo de un sistema de instrumentación para el proyecto BAJA de la organización SAEUC Venezuela. Trabajo especial de grado para optar al título de ingeniero electricista. Universidad de Carabobo. Carabobo, Venezuela.

[7] **Fernando Botello y José María Rojas.** (2010). Diseño y construcción de un módulo para pantallas a base de LEDS RGB, que permita la comunicación vía USB para la visualización de datos transmitidos desde una computadora. Trabajo especial de grado para optar al título de ingeniero electricista. Universidad de Carabobo. Carabobo, Venezuela.