Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas

AUTOMATIZACIÓN DE LA TEMPERATURA AMBIENTE DE UN CENTRO DE COMPUTO UTILIZANDO EL PIC16F877 Y EL SENSOR LM35D

ANGEL ATILIO MALTEZ CIFUENTES

Asesorado por: Ing. Guillermo Rafael Sánchez Barrios

Guatemala, octubre de 2005

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

AUTOMATIZACIÓN DE LA TEMPERATURA AMBIENTE DE UN CENTRO DE COMPUTO UTILIZANDO EL PIC16F877 Y EL SENSOR LM35D

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

ANGEL ATILIO MALTEZ CIFUENTES
ASESORADO POR: ING. GUILLERMO RAFAEL SANCHEZ B.

AL CONFERÍRSELE EL TÍTULO DE
INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, OCTUBRE DE 2005

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	
VOCAL II	Lic. Amahán Sánchez Álvarez
VOCAL III	Ing. Julio David Galicia Celada
VOCAL IV	Br. Kenneth Issur Estrada Ruiz
VOCAL V	Br. Elisa Yazminda Vides Leiva
SECRETARIA	Inga. Marcia Ivonne Véliz Vargas

TRIBUNAL QUE PRACTICO EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADOR	Ing. Luis Alberto Vettorazzi España
EXAMINADOR	Inga. Ligia Maria Pimentel Castañeda
EXAMINADOR	Inga. Elizabeth Domínguez Alvarado
SECRETARIA	Inga. Marcia Ivonne Véliz Vargas

HONORABLE TRIBUNAL EXAMINADOR

Cumpliendo con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a consideración mi trabajo de graduación titulado:

AUTOMATIZACIÓN DE LA TEMPERATURA AMBIENTE DE UN CENTRO DE COMPUTO UTILIZANDO EL PIC16F877 Y EL SENSOR LM35D

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas con fecha febrero de 2004.

Ángel Atilio Maltez Cifuentes

Guatemala, 8 de Octubre de 2005

Ingeniero
Carlos Azurdia
Coordinador de Tesis
Escuela de Ciencias y Sistemas
Facultad de Ingeniería

Estimado Ingeniero Azurdia:

Por este medio me permito hacer de su conocimiento que he procedido a revisar el trabajo de tesis del estudiante Ángel Atilio Maltez Cifuentes, titulado **AUTOMATIZACIÓN DE LA TEMPERATURA AMBIENTE DE UN CENTRO DE COMPUTO UTILIZANDO EL PIC16F877 Y EL CENSOR LM35D.**

Considerando que dicho trabajo cumple con los objetivos propuestos para su desarrollo.

Atentamente,

Ing. Guillermo Rafael Sánchez Barrios

AGRADECIMIENTOS A:

Dios y la Virgen Maria	Por haberme permito vivir y luchar para alcanzar este triunfo, por la sabiduría y fuerza que me brindaron.
Mis padres	Por darme la vida y confiar en mí, motivándome a alcanzar este éxito.
Mis tías	Tía Fulvia y Dori por que sin su ayuda no lo hubiera logrado.
Mi esposa y mi hijo	Por la paciencia y compresión que tuvieron en todos los momentos de trabajo en mi carrera sin importar que los dejaba solos.
Mis hermanos y familia política	Por estar pendientes y felices de mi éxito.
Todas las personas	Que me brindaron su apoyo y ayuda durante toda mi carrera en especial a mis amigos de grupo.

DEDICADO A:

Dios y la Virgen Maria	Por que sin ellos no lo hubiera echo.
Mis padres	Como respuesta a la confianza puesta en mí y que este triunfo se convierta en una de sus más grandes satisfacciones.
Mi esposa y mi hijo	Por ser instrumentos de inspiración y lucha y mis grandes amores.
Mis Hermanos y sus familias	Como un ejemplo de lucha, cariñosamente.
Mis Abuelitas	Elva Maltez y Rosa Cifuentes (Mama Rosa) recordándolas siempre con mucho cariño.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
ºC	Grados Centígrados o Celsius
ºF	Grados Fahrenheit
mA	Mili Amperios
µA	Micro Amperios
A	Amperios
V o Volts	Voltios
mV	Mili Voltios
Mhz	*Megahertz*
Khz	*Kilohertz*
Seg	Segundos
Nseg	Nano Segundo
Pf	Pico Faradios
KB	*Kilobytes*
Hex	Hexadecimal
Tcy	Ciclo de instrucción
Vi	Voltaje de Entrada
Vfe	Tensión de fondo de escala
Ta	Tiempo de apertura del circuito

GLOSARIO

- **Actuador** — Es un integrado que realiza un proceso al recibir una señal.
- **Aplicación construida** — El acto de ensamblar o compilar todos los componentes en un proyecto de una aplicación.
- **Archivo “hex”** — Representación *ASCII* del código de máquina. Un archivo HEX está compuesto de registros que le especifican al microcontrolador datos o instrucciones que serán ubicados en un dispositivo de memoria programable.
- **Sensores de temperatura** — Dispositivo que permite determinar la temperatura de un ambiente cerrado.
- **Centro de cómputo** — Lugar donde se almacena una serie de computadora que actúan como servidores de información de datos.

- **Circuito integrado** — Dícese, así, al conjunto de dispositivos que generan salidas en base al voltaje que se le aplica en sus entradas.
- **"*CMOS*"** — Integrado semiconductor de oxido de metal.
- **Código fuente** — Archivo de texto que es procesado por un lenguaje ensamblador o un compilador para producir un archivo de objeto intermedio, o código de máquina que pueda ejecutarse en un microcontrolador.
- **Código objeto** — Código intermedio producido por un lenguaje ensamblador o compilador. Este código no contiene direcciones para la memoria del programa o las variables del *RAM* incluidas en el mismo, pero contiene demarcadores para las direcciones que el *linker* debe determinar cuando coloca el código en la memoria del programa.
- **Codiseño** — Diseño de un conjunto de sistemas que actuaran con la decisión de una computadora para realizar un proceso.

- **Compilador** — Programa que convierte las proposiciones de un lenguaje de alto nivel en un código de máquina que puede ejecutarse en un microcontrolador. Una proposición de un lenguaje de alto nivel usualmente genera una cantidad de instrucciones de código de máquina.

- **Controlador** — Es un dispositivo para controlar un proceso.

- **Cosimulación** — Nos indicara que la decisión tomada fue o no la correcta.

- **"*Display*"** — Dispositivo que permite visualizar la salida binaria del PIC en un número ordinario.

- **Emulador** — Instrumento de hardware usado en lugar del microcontrolador en una aplicación. El emulador permite que el código sea, rápidamente, descargado, ejecutado y probado, tal como será ejecutado en la aplicación final.

- **"*FLASH*"** — Memoria tipo *PROM* que es de alta velocidad y se puede escribir varias veces en ella.

- **Hardware** — Partes o componentes físicos que integran una herramienta; inclusive ella misma como una unidad.

- **Herramienta** — Recurso o mecanismo que puede ser utilizado para facilitar la solución de cualquier problema que el ser humano tenga que afrontar en un momento determinado.

- ***"IC-Prog"*** — Software permite cargar el programa compilado en *mplab* al PIC.

- ***"IDE"*** — - Entorno de Desarrollo Integrado - Aplicación que usa múltiples funciones y características para el desarrollo del código. Las distintas herramientas pueden ser usadas sin cambiar a otros programas.

- ***"IIC"*** — Es un bus desarrollado por *Philips* semiconductor, bidireccional a dos hilos que puede conectar varios microcontroladores mediante un hardware simple. Además, el número máximo de controladores que se pueden conectar al bus, está limitado por una capacidad de 400pF.

- **Instrucciones** — Conjunto de pasos que se escriben para realizar un proceso en determinado dispositivo.

- **Interrupción** — Proceso que se activa al momento de suceder un determinado evento.

- **Lenguaje ensamblador** — Programa que convierte instrucciones nemotécnicas en un código de máquina que pueda ejecutarse en un microcontrolador.

- ***"Linker"*** — Programa que convierte el código de un objeto en un código de máquina ejecutable. El *linker* determina dónde se ubicará el código en la memoria y cómo será usada la *RAM* por las variables.

- **Memoria de programa** — La memoria de un microcontrolador que contiene una serie de instrucciones para ejecutar una aplicación.

- **Microcontrolador** — Dispositivo electrónico que contiene todas las características de una computadora.

- **Modo de desarrollo** — Establece la herramienta que ejecutará el código. Para esta guía hemos usado el *MPLAB*, el simulador de software.

- ***"MPLAB"*** — *Software* encargado de compilar código fuente, que no es más que el conjunto de instrucciones para el PIC y que tienen extensión *.ASM.*

- **Metodología** — Es el proceso, técnicas o enfoques empleados en la solución de un problema o en la creación de algo: un procedimiento particular o un conjunto de procedimientos.

- ***"*PIC*"*** — Es un microcontrolador basado en memoria *EPROM/FLASH* desarrollado por *Microchip Technology* y los cuales son programables.

- **Programa de computadora** — Secuencia de instrucciones u ordenes que una computadora ejecuta para poder llevar a cabo una tarea especifica.

- **Programador de dispositivo** — Instrumento de hardware que toma el código de máquina desde un archivo y lo programa en un microcontrolador programable o un chip de memoria programable.

- ***"PROM"*** — - *Programmable Read Only Memory* o Memoria Programable Solamente para Lectura -.

- **Proyecto o aplicación** — Grupo de archivos usados para crear una aplicación, junto con instrucciones a lenguajes ensambladores, compiladores y *linkers* para convertir los archivos en un código de máquina ejecutable. Los archivos pueden ser archivos fuente de ensamble, archivos fuente del compilador, bibliotecas, archivos de objeto pre-compilados, y archivos de instrucciones llamados escrituras del *linker*.

- ***"PWM"*** — Modulación de anchura de impulsos.

- **Quemador** — Es un circuito que permite cargar un programa al microcontrolador.

- ***"RAM"*** — (*Random Access Memory* o Memoria de Acceso Aleatorio).

- **Registro de archivos** — El área de la memoria *RAM* interna de PIC usada para almacenaje de variables.

- **Registro de función especial** — Ubicaciones internas de un PIC que pueden ser usadas para afectar la operación interna del controlador o un periférico. Entre los ejemplos se pueden incluir los registros de control de interrupción, los *timers* y los registros I/O.

- ***"Relay"*** — Dispositivo que deja pasar corriente de un lado al momento de cerrar un *switch* interno.
- **Resistencia** — Es un limitador de corriente que se opone al flujo de electrones.
- **Servidor** — Computadora que lleva el control de los datos y a la cual accedan varios usuarios desde diferentes puntos de ubicación.
- ***"Smtp"*** — Protocolo para la transferencia de mensajes electrónicos.
- ***"Software"*** — Es un programa de computadora que permite al programador interactuar con la computadora.
- **Temperatura ambiente** — Se le llama temperatura ambiente, al la temperatura registrada en un cuarto cerrado.
- ***"Vdd"*** — Voltaje de alimentación del integrado.
- **Voltaje** — Es una fuerza magnetomotriz que mueve electrones.

RESUMEN

El aporte del presente trabajo de investigación, es presentar al lector la forma que puede implementar un sistema para controlar y monitorear la temperatura que existe en un centro de cómputo donde se encuentran los servidores y, así, poder garantizar el perfecto funcionamiento y rendimiento de cada uno de ellos.

Esto puede garantizarles, a los encargados de centro de cómputo, la seguridad y confianza, además, grandes beneficios debido a que existen algunos momentos en los que la temperatura ambiente puede llegar a disminuir por algún error, debido a que se apagó el aire acondicionado o este se detuvo por daños físico, por lo cual puede llegar a provocarnos grandes pérdidas si los servidores llegaran a calentarse y estos apagarse, automáticamente, por los dispositivos de seguridad que poseen.

A todas estas automatizaciones para la realización de este tipo de tareas, se les conoce como automatizaciones de toma de decisiones.

El presente trabajo tiene como principales objetivos:

- servir como herramienta alternativa para la toma automática de decisiones con el simple hecho de recibir una entrada;
- servir como base o prototipo para realizar automatizaciones controladas por medio de microcontroladores programables.

OBJETIVOS

Generales

1. Implementar un proyecto o aplicación capaz de tomar una decisión en base a la lectura de la temperatura registrada en un ambiente cerrado.

2. Alcanzar un nivel fiable de la temperatura para que los equipos que se encuentran en el centro de cómputo puedan trabajar a su máximo rendimiento.

3. Promover la automatización para toma de decisiones demostrando que su costo es bastante bajo a comparación del beneficio que se podrá obtener para determinados procesos y que su fabricación no requiere de mucho conocimiento en electrónica.

Específicos

1. Evaluar herramientas y técnicas para la implementación de este tipo de proyectos, de manera que se busque la mejor solución al mejor precio sabiendo que su funcionamiento será el mismo.

2. Determinar un estudio técnico de implementación para comprender que procesos hay que automatizar al momento de realizar un proyecto de esta magnitud.

INTRODUCCIÓN

Dada la necesidad de mantener un conjunto de computadoras siempre funcionando a su máximo rendimiento, se a creado este tipo de procesos, los cuales son capaces de controlar la temperatura ambiente en la que se encuentra un centro de computo completamente cerrado, debido a esto se han dado una serie problemas bastantes serios al momento de que dichos servidores se apaguen lo cual no puede llegar a suceder cuando las empresas realizan miles de transacciones en lapsos de tiempos muy cortos o que dependen de que sus sistemas estén encendidos los 365 días del año.

Es por eso que ha surgido esta idea para garantizar que en determinado momento se puede tomar una decisión automática sin que haya una persona cerca para la realización de un evento que avise, arranque el aire o suene una alarma, tomando en cuenta lo que esta pasando debido a la variación de temperatura.

Las condiciones ambientales de alta humedad, también, pueden ocasionar la corrosión de los componentes internos de una computadora y la degradación de algunas de sus propiedades esenciales, como la resistencia eléctrica o la conductividad térmica. En casos extremos, la humedad puede ocasionar cortocircuitos, provocando desde la pérdida de datos hasta el daño físico de algunos componentes del sistema. Generalmente, los ordenadores suelen encontrarse en ambientes con un nivel aceptable de humedad relativa, como

pueden ser oficinas donde las condiciones ambientales se regulan a través de sistemas de aire acondicionado.

La construcción de un este dispositivo se enfoca en la temperatura ambiente para que no llegue a ocurrir un accidente que acabe deteniendo el sistema de una empresa a la cual le generarían miles de pérdidas.

1. MICROCONTROLADORES

1.1 Qué es un microcontrolador

Un microcontrolador es un circuito integrado programable que contiene todos los componentes de un computador, se emplea para realizar una tarea determinada para la cual ha sido programado.

Dispone de procesador, memoria para el programa y los datos, líneas de entrada y salida de datos y suele estas asociado a múltiples recursos auxiliares. Puede controlar cualquier cosa y suele estar incluido en el mismo dispositivo que controla.

- Máquinas expendedora de productos.
- Controles de acceso tanto de personas como de objetos
- Máquinas herramientas, motores, temporizadores.
- Sistemas autónomos de control, incendio, humedad, temperatura. etc.
- Telefonía, Automatismos, Medicina, Automoción, etc.

Básicamente consta de un programa más o menos complejo que da las pautas para realizar un trabajo ayudado por unos sensores y actuadores que recogen la información y transmiten las instrucciones.

A es este chip se le conoce también como microcomputadora, porque tiene todos los componentes y recursos necesarios para serlo, es decir:

Una CPU quien interpreta las instrucciones de programa.

Una memoria *PROM* el cual memoriza permanentemente las instrucciones de programa. Otros modelos de microcontroladores tienen memoria de programa de tipo *EEPROM* y otros de tipo *FLASH*

Una memoria *RAM* utilizada para memorizar las variables utilizadas para el programa.

Una serie de líneas de E/S para controlar dispositivos externos o recibir pulsos de sensores, switches, etc.

Una serie de dispositivos auxiliares para su funcionamiento, como puede ser generador de reloj, bus, contador, etc.

1.2 Tipos de arquitectura de microcontroladores

1.2.1 Arquitectura cerrada

Cada modelo se construye con un determinado *CPU*, cierta capacidad de memoria de datos, cierto tipo y capacidad de memoria de instrucciones, un número de E/S y un conjunto de recursos auxiliares muy concreto. El modelo no admite variaciones ni ampliaciones. La aplicación a la que se destina debe encontrar en su estructura todo lo que precisa y, en caso contrario, hay que desecharlo. Microchip ha elegido principalmente este modelo de arquitectura.

1.2.2 Arquitectura abierta

Estos microcontroladores se caracterizan porque, además de disponer de una estructura interna determinada, pueden emplear sus líneas de E/S para sacar al exterior los buses de datos, direcciones y control, con lo que se posibilita la ampliación de la memoria y las E/S con circuitos integrados externos. *Microchip* dispone de modelos PIC con arquitectura abierta, sin embargo, esta alternativa se escapa de la idea de un microcontrolador incrustado y se asemeja a la. Solución que emplean los clásicos microprocesadores. En nuestra opinión, los verdaderos microcontroladores responden a la arquitectura cerrada y permiten resolver una aplicación con un solo circuito integrado y a precio muy reducido.

La mayoría de los sistemas de control incrustados requieren *CPU*, memoria de datos, memoria de instrucciones, líneas de E/S, y diversas funciones auxiliares como temporizadores, comunicación serie y otras. La capacidad y el tipo de las memorias, el número de líneas de E/S y el de temporizadores, así como circuitos auxiliares, son parámetros que dependen exclusivamente de la aplicación y varían mucho de unas situaciones a otras.

1.3 Familias de microcontroladores

1.3.1 PIC16C5X Familia base

La familia 16C5X se consolido como la base en el desarrollo de nuevas tecnologías ofreciendo la solución costeable más efectiva.

Estos microcontroladores cuentan con un conjunto de instrucciones de 12-BIT de ancho y se ofrecen en empaquetados de 18, 20 y 22 pines. En opciones de empaquetado SOIC o SSOP. Con bajo voltaje de operación hasta de 2 volt, hacen de esta familia un elemento ideal para ser operado por aplicaciones a baterías.

1.3.2 PIC16C5XX Familia de rango medio

La familia de rango medio ofrece un amplio rango de opciones, desde empaquetados de 18 hasta 44 pines, así como un alto nivel de integración de periféricos. Esta familia cuenta con un conjunto de instrucciones de 14-BIT de ancho y la capacidad de manejar interrupciones.

1.3.3 PIC 17CXX Familia de rango alto

Esta familia de alto rendimiento ofrece la más alta velocidad de ejecución de todos los microcontroladores de 8-BIT de la industria. Utilizando una arquitectura de instrucciones de 16-BIT, mejora el conjunto de instrucciones y las capacidades de interrupción.

1.4 Aplicaciones de los microcontroladores

Cada vez existen más productos que incorporan un microcontrolador con el fin de aumentar sustancialmente sus prestaciones, reducir su tamaño y costo, mejorar su fiabilidad y disminuir el consumo.

Algunos fabricantes de microcontroladores superan el millón de unidades de un modelo determinado producidas en una semana. Este dato puede dar una idea de la masiva utilización de estos componentes.

Los microcontroladores están siendo empleados en multitud de sistemas presentes en nuestra vida diaria, como pueden ser juguetes, horno microondas, frigoríficos, televisores, computadoras, impresoras, módems, el sistema de arranque de nuestro vehiculo, etc. Y otras aplicaciones con las que seguramente no estaremos tan familiarizados como instrumentación electrónica, control de sistemas en una nave espacial, etc. Una aplicación típica podría emplear varios microcontroladores para controlar pequeñas partes del sistema. Estos pequeños controladores podrían comunicarse entre ellos y con un procesador central, probablemente más potente, para compartir la información y coordinar sus acciones, como, de hecho, ocurre ya habitualmente en cualquier PC.

1.5 Mercado de los microcontroladores

Aunque en el mercado de la microinformática la mayor atención la acaparan los desarrollos de los microprocesadores, lo cierto es que se venden cientos de microcontroladores por cada uno de aquéllos. Existe una gran diversidad de microcontroladores. Quizá la clasificación más importante sea entre microcontroladores de 4, 8, 16 ó 32 bits. Aunque las prestaciones de los microcontroladores de 16 y 32 bits son superiores a los de 4 y 8 bits, la realidad es que los microcontroladores de 8 bits dominan el mercado y los de 4 bits se resisten a desaparecer.

La razón de esta tendencia es que los microcontroladores de 4 y 8 bits son apropiados para la gran mayoría de las aplicaciones, lo que hace absurdo emplear micros más potentes y consecuentemente más caros. Uno de los sectores que más tira del mercado del microcontrolador es el mercado automovilístico. De hecho, algunas de las familias de microcontroladores actuales se desarrollaron pensando en este sector, siendo modificadas posteriormente para adaptarse a sistemas más genéricos. El mercado del automóvil es además uno de los más exigentes: los componentes electrónicos deben operar bajo condiciones extremas de vibraciones, choques, ruido, etc. y seguir siendo fiables. El fallo de cualquier componente en un automóvil puede ser el origen de un accidente.

En cuanto a las técnicas de fabricación, cabe decir que prácticamente la totalidad de los microcontroladores actuales se fabrican con tecnología *CMOS*. Esta tecnología supera a las técnicas anteriores por su bajo consumo y alta inmunidad al ruido. La distribución de las ventas según su aplicación es la siguiente:

- Una tercera parte se absorbe en las aplicaciones relacionadas con las computadoras y sus periféricos.
- La cuarta parte se utiliza en las aplicaciones de consumo (electrodomésticos, juegos, tv, vídeo, etc.)
- El 16% de las ventas mundiales se destinó al área de las comunicaciones.
- Otro 16% fue empleado en aplicaciones industriales.
- El resto de los microcontroladores vendidos en el mundo, aproximadamente un 10% fueron adquiridos por las industrias de automoción.

También los modernos microcontroladores de 32 bits van afianzando sus posiciones en el mercado, siendo las áreas de más interés el procesamiento de imágenes, las comunicaciones, las aplicaciones militares, los procesos industriales y el control de los dispositivos de almacenamiento masivo de datos, y automatización de temperaturas que es nuestro caso a esa aplicación lo orientaremos.

1.6 Qué microcontrolador utilizar

A la hora de escoger el microcontrolador a emplear en un diseño concreto hay que tener en cuenta multitud de factores, como la documentación y herramientas de desarrollo disponibles y su precio, la cantidad de fabricantes que los producen y por supuesto las características del microcontrolador (tipo de memoria de programa, número de temporizadores, interrupciones, etc.).

Primero se ve el costo. Como es lógico, los fabricantes de microcontroladores compiten duramente para vender sus productos. Y no les va demasiado mal ya que sin hacer demasiado ruido venden 10 veces más microcontroladores que microprocesadores. Para que nos hagamos una idea, para el fabricante que usa el microcontrolador en su producto una diferencia de precio en el microcontrolador de algunos quetzales es importante (el consumidor deberá pagar además el costo del empaquetado, el de los otros componentes, el diseño del hardware y el desarrollo del software). Si el fabricante desea reducir costos debe tener en cuenta las herramientas de apoyo con que va a contar: emuladores, simuladores, ensambladores, compiladores, etc. Es habitual que muchos de ellos siempre se decanten por microcontroladores pertenecientes a una única familia.

Aplicación. Antes de seleccionar un microcontrolador es imprescindible analizar los requisitos de la aplicación:

- Procesamiento de datos: puede ser necesario que el microcontrolador realice cálculos críticos en un tiempo limitado. En ese caso debemos asegurarnos de seleccionar un dispositivo suficientemente rápido para ello. Por otro lado, habrá que tener en cuenta la precisión de los datos a manejar: si no es suficiente con un microcontrolador de 8 bits, puede ser necesario acudir a microcontroladores de 16 ó 32 bits, o incluso a hardware de coma flotante. Una alternativa más barata y quizá suficiente es usar librerías para manejar los datos de alta precisión.

- Entrada Salida: para determinar las necesidades de entrada/salida del sistema es conveniente dibujar un diagrama de bloques del mismo, de tal forma que sea sencillo identificar la cantidad y tipo de señales a controlar. Una vez realizado este análisis puede ser necesario añadir periféricos hardware externos o cambiar a otro microcontrolador más adecuado a ese sistema.

- Consumo: algunos productos que incorporan microcontroladores están alimentados con baterías y su funcionamiento puede ser tan vital como activar una alarma antirrobo. Lo más conveniente en un caso como éste puede ser que el microcontrolador esté en estado de bajo consumo pero que despierte ante la activación de una señal (una interrupción) y ejecute el programa adecuado para procesarla.

- Memoria: para detectar las necesidades de memoria de nuestra aplicación debemos separarla en memoria volátil (*RAM*), memoria no volátil (*ROM, EPROM*, etc.) y memoria no volátil modificable (*EEPROM*).

Este último tipo de memoria puede ser útil para incluir información específica de la aplicación como un número de serie o parámetros de calibración.

El tipo de memoria a emplear vendrá determinado por el volumen de ventas previsto del producto: de menor a mayor volumen será conveniente emplear *EPROM, OTP* y *ROM*. En cuanto a la cantidad de memoria necesaria puede ser imprescindible realizar una versión preliminar, aunque sea en pseudo-código, de la aplicación y a partir de ella hacer una estimación de cuánta memoria volátil y no volátil es necesaria y si es conveniente disponer de memoria no volátil modificable.

- Ancho de palabra: el criterio de diseño debe ser seleccionar el microcontrolador de menor ancho de palabra que satisfaga los requerimientos de la aplicación. Usar un microcontrolador de 4 bits supondrá una reducción en los costes importante, mientras que uno de 8 bits puede ser el más adecuado si el ancho de los datos es de un bite. Los microcontroladores de 16 y 32 bits, debido a su elevado coste, deben reservarse para aplicaciones que requieran sus altas prestaciones (Entrada/Salida potente o espacio de direccionamiento muy elevado).

- Diseño de la placa: la selección de un microcontrolador concreto condicionará el diseño de la placa de circuitos. Debe tenerse en cuenta que quizá usar un microcontrolador barato encarezca el resto de componentes del diseño.

Los microcontroladores más populares se encuentran, sin duda, entre las mejores elecciones:

8048 *(*Intel*).* Es el padre de los microcontroladores actuales, el primero de todos. Su precio, disponibilidad y herramientas de desarrollo hacen que todavía sea muy popular.

8051 *(*Intel y otros*).* Es sin duda el microcontrolador más popular. Fácil de programar, pero potente. Está bien documentado y posee cientos de variantes e incontables herramientas de desarrollo.

***80186, 80188 y 80386 EX* (Intel).** Versiones en microcontrolador de los populares microprocesadores 8086 y 8088. Su principal ventaja es que permiten aprovechar las herramientas de desarrollo para PC.

68HC11 *(*Motorola y Toshiba*).* Es un microcontrolador de 8 bits potente y popular con gran cantidad de variantes.

683xx *(*Motorola). Surgido a partir de la popular familia 68k, a la que se incorporan algunos periféricos. Son microcontroladores de altísimas prestaciones.

PIC (MicroChip). Familia de microcontroladores que gana popularidad día a día. Fueron los primeros microcontroladores *RISC* para la aplicación de automatización de temperatura tomaremos este en el modelo PIC 16f877 que se explica mas adelante.

Es preciso resaltar en este punto que existen innumerables familias de microcontroladores, cada una de las cuales posee un gran número de variantes.

1.7 Descripción del microcontrolador PIC 16f877 de Microchip

1.7.1 Características

- *CPU RISC*
- ARQUITECTURA HARVARD
- 35 INSTRUCCIONES DE ENSAMBLADOR
- EJECUCIÓN EN 1 CICLO MÁQUINA
- MÁXIMA VEL 20MHZ
- MEMORIA PROGRAMA 8K-PALABRAS (*FLASH*)
- 368 KB MEM RAM DE DATOS
- 1 *WATCHDOG*
- 3 *TIMERS* (8,16,8 BITS)
- 2 CANALES *PWM*
- 8 CONVERTIDORES A/D de 10 BITS
- PUERTOS SERIES
- SPI
- IIC
- *USART*
- 256 KB MEM *EEPROM* DE DATOS
- PUERTO PARALELO ESCLAVO DE 8 BITS
- 33 PINES E/S EN 5 PUERTOS
- PILA *HARDWARE*

- *DEBUGGER IN CIRCUIT*

Puerto A

- PUERTO DE E/S DE 6 PINES
- RA0 o RA0 y AN0
- RA1 o RA1 y AN1
- RA2 o RA2, AN2 y VREF-
- RA3 o RA3, AN3 y VREF+
- RA4 o RA4 (Salida en colector abierto) y T0CKI(Entrada de reloj del modulo Timer0)
- RA5 o RA5, AN4 y SS (Selección esclavo para el puerto serie asíncrono)

Puerto B

- PUERTO E/S 8 PINES
- RESISTENCIAS *PULL-UP* PROGRAMABLES
- RB0 o Interrupción externa
- RB4-7 Interrupción por cambio de flanco
- RB5-RB7 y RB3 o programación y *debugger* en circuito

Puerto C

- PUERTO E/S 8 PINES
- RC0 o RC0, T1OSO (Timer1 salida oscilador) y T1CKI (Entrada de reloj del modulo Timer1).
- RC1-RC2 o *PWM/COMP/CAPT*
- RC1 o T1OSI (entrada *osc timer1*)

- RC3-4 o IIC
- RC3-5 o SPI
- RC6-7 o USART

Puerto D

- PUERTO E/S 8 PINES
- BUS DE DATOS EN PPS (Puerto paralelo esclavo)

Puerto E

- PUERTO E/S 3 PINES
- RE0 o RE0 y AN5 y *Read* de PPS
- RE1 o RE1 y AN6 y *Write* de PPS
- RE2 o RE2 y AN7 y CS de PPS

Registro de dirección

- Se denominan:
  - TRISA
  - TRISB
  - ...
- 0 significa *Output* (salida)
- 1 significa *Input* (entrada)

1.7.2 Canales para llegar al PIC

En primer lugar necesitamos de un software de ambiente agradable en entorno Windows donde poder realizar y simular nuestros programas, para esto tenemos el *IDE (Integrate Development Enviroment)* es decir, Ambiente de Desarrollo Integrado más conocido como *MPLAB*.

Este software será el encargado de compilar nuestro código fuente, que no es más que el conjunto de instrucciones que editemos en el *MPLAB* y que tienen extensión .ASM.

MPLA es un software proporcionado gratuitamente por la microchip, acá editamos nuestras diferentes directivas e instrucciones de programa, es acá donde plasmamos nuestra capacidad e intuición de programación.

Una vez bien depurado nuestro programa, este genera un archivo de salida con extensión .hex.

Este archivo .hex se carga primero en una interfase de usuario, siendo el mas conocido y usado el ProPic. En el ProPic se configura tipo de puerto de comunicación, en nuestro caso el paralelo, el tipo de procesador que usamos, es decir el modelo de PIC, tipo de oscilador que me dará la frecuencia de reloj.

EL archivo .hex contiene el código de operación (Código OP) que será enviado a la memoria de programa (*FLASH* o *EEPROM*) dentro del PIC por medio del cable paralelo y del circuito de programación que forma parte del entrenador.

El archivo .hex no es un archivo en formato binario y no refleja directamente el contenido que deberá tener la memoria de programa (*FLASH* o *EEPROM*) del PIC. Pero los formatos reflejarán directamente cuando sean transferidos al PIC en forma de bajo nivel y con algunas instrucciones más.

Sin entrar en detalles es útil saber que tal formato es directamente reconocido por el hardware del PIC que promueve durante la programación la conversión en binario del código de operación (código OP). Si nos preguntamos porque código binario, es simplemente porque como buena microcomputadora, al igual que cualquier computador, procesador y sistema digital, el PIC entiende solamente código binario.

Es por esta razón que se necesita siempre de un software de aplicación que compila o traduce a binario la serie de instrucciones que editamos y conocemos como código fuente.

Asumiremos entonces que el Código de operación o Código OP, es simplemente el código de nuestras instrucciones en binario.

Ejemplo:

00 0001 0000 0000B

En notación hexadecimal será:
0100H

Vemos que este código tiene una extensión de 14 bits, que es la extensión de la memoria de programa. Este código, completamente sin sentido para los humanos, es lo que el PIC esta preparado para entender. Para facilitar la comprensión del programador (nosotros), se recurre a un instrumento y convención para tornar la instrucción más comprensible.

La primera convención es la que asocia el código OP (un total de 35 para o Pic16x84; letra "x" puede ser F, C ó CR) a una sigla nemónica, o sea una inicial que nos permitirá recordar fácilmente el significado de la instrucción.

Volviendo a nuestro ejemplo el código OP 0100H corresponde a la instrucción nemónica CLRW que es una forma abreviada de la instrucción *CLEAR* W *REGISTER* (veremos posteriormente lo que significa).

De esta manera todos los neumónicos o siglas de las instrucciones de programación tienen su respectivo código OP.

1.7.3 Funcionamiento en el interior

Viendo la arquitectura interna del PIC sus componentes principales son: el *CPU*, memoria *RAM*, memoria *FLASH* o *EEPROM*, puertos E/S.

Una vez que tenemos grabado el código de operación dentro de la memoria de programa o simplemente nuestro programa, el PIC, está listo para realizar su función encomendada. Todo el funcionamiento interno del PIC se trata de manejo y configuración de bits de archivos.

Estos archivos se encuentran implementados en la memoria *RAM* y cada uno tiene una longitud de 8 bits, cada bite de cada archivo cumple una función, por esta razón a estos archivos que ocupan las primeras posiciones de de la memoria *RAM* se les llama Registros de función específica o SFR *(Special Function Register).*

También existen posiciones de la memoria *RAM* que no están ocupados por los SFR, y nos sirven para implementar nuestros propios registros, por esto se les llama Registros de propósito general GPR *(General Purpose Registe*r). En resumen existe un uso intensivo de los registros de la memoria *RAM* en su ínter actuación con la memoria de programa y la *CPU*.

1.7.3.1 Temporizadores

Se emplean para controlar periodos de tiempo (temporizadores) y para llevar la cuenta de acontecimientos que suceden en el exterior (contadores). Para la medida de tiempos se carga un registro con el valor adecuado y a continuación dicho valor se va incrementando o decrementando al ritmo de los impulsos de reloj o algún múltiplo hasta que se desborde y llegue a 0, momento en el que se produce un aviso.

Cuando se desean contar acontecimientos que se materializan por cambios de nivel o flancos en alguna de las patitas del microcontrolador, el mencionado registro se va incrementando o decrementando al ritmo de dichos impulsos.

1.7.3.2 *Watchdog* o perro guardián

Cuando el computador personal se bloquea por un fallo del software u otra causa, se pulsa el botón del *reset* y se reinicializa el sistema. Pero un microcontrolador funciona sin el control de un supervisor y de forma continuada las 24 horas del día. El perro guardián consiste en un temporizador que, cuando se desborda y pasa por 0, provoca un reset automáticamente en el sistema.

Se debe diseñar el programa de trabajo que controla la tarea de forma que refresque o inicialice al perro guardián antes de que provoque el reset. Si falla el programa o se bloquea, no se refrescará al perro guardián, y al completar su temporización, "ladrará y ladrará" hasta provocar el *reset*.

1.7.3.3 Protección ante fallo de alimentación o *"Brownout"*

Se trata de un circuito que resetea al microcontrolador cuando el voltaje de alimentación (VDD) es inferior a un voltaje mínimo *("brownout").* Mientras el voltaje de alimentación sea inferior al de *brownout* el dispositivo se mantiene reseteado, comenzando a funcionar normalmente cuando sobrepasa dicho valor.

1.7.3.4 Conversor A/D (CAD)

Los microcontroladores que incorporan un conversor A/D (Analógico/Digital) pueden procesar señales analógicas, tan abundantes en las aplicaciones. Suelen disponer de un multiplexor que permite aplicar a la entrada del CAD diversas señales analógicas desde las patitas del circuito integrado.

1.7.3.5 Conversor D/A (CDA)

Transforma los datos digitales obtenidos del procesamiento del computador en su correspondiente señal analógica que saca al exterior por una de las patitas de la cápsula. Existen muchos efectores que trabajan con señales analógicas.

1.7.3.6 Comparador analógico

Algunos modelos de microcontroladores disponen internamente de un amplificador operacional que actúa como comparador entre una señal fija de referencia y otra variable que se aplica por una de las patitas de la cápsula. La salida del comparador proporciona un nivel lógico 1 ó 0 según una señal sea mayor o menor que la otra.

También hay modelos de microcontroladores con un módulo de tensión de referencia que proporciona diversas tensiones de referencia que se pueden aplicar en los comparadores.

1.7.3.7 Modulador de anchura de impulsos o *PWM*

Son circuitos que proporcionan en su salida impulsos de anchura variable, que se ofrecen al exterior a través de las patitas del encapsulado.

1.7.3.8 Puertos de E/S digitales

Todos los microcontroladores destinan algunas de sus patitas a soportar líneas de E/S digitales. Por lo general, estas líneas se agrupan de ocho en ocho formando Puertos. Las líneas digitales de los puertos pueden configurarse como entrada o como Salida cargando un 1 ó un 0 en el bite correspondiente de un registro destinado a su configuración.

1.7.3.9 Puertos de comunicación

Con objeto de dotar al microcontrolador de la posibilidad de comunicarse con otros dispositivos externos, otros buses de microprocesadores, buses de sistemas, buses de redes y poder adaptarlos con otros elementos bajo otras normas y protocolos. Algunos modelos disponen de recursos que permiten directamente esta tarea, entre los que destacan:

UART, adaptador de comunicación serie asíncrona.

USART, adaptador de comunicación serie síncrona y asíncrona.

Puerta paralela esclava para poder conectarse con los buses de otros microprocesadores. USB (*Universal Serial Bus*), que es un moderno bus serie para los PC.Bus I^2C, que es un interfaz serie de dos hilos desarrollado por *Philips*. CAN (*Controller Area Network*), para permitir la adaptación con redes de conexionado multiplexado desarrollado conjuntamente por *Bosch* e Intel para el cableado de dispositivos en automóviles.

2. SENSORES

2.1 Qué son sensores

Los sensores son cada vez más comunes en nuestra vida diaria. Un coche, por ejemplo, utiliza docenas de ellos para permitirnos controlar sus funciones básicas. Sin embargo, este tipo de sensores están muy limitados, puesto que, colocados estáticamente en un lugar, adolecen de la capacidad de analizar o actuar sobre los datos que detectan, y simplemente, su misión se limita a enviar las mediciones que han registrado a un procesador central.

En definitiva, los sensores todavía podrían dar mucho más de sí. Así lo cree toda una industria tecnológica que está detrás de ellos, y son cada vez más las empresas y los equipos de investigadores que trabajan en el desarrollo de este tipo de dispositivos. En este sentido, compañías como la cadena de supermercados británicos *Tesco* o la compañía petrolífera *Shell* han instalado sistemas de primera generación para controlar y chequear el estado de los expendedores de gasolina en sus estaciones de servicio.

Los sensores nos permiten estar seguros y confiados que en determinado momento leerán la información que esta pasando en el centro de cómputo, y esto permitirá enviar un mensaje o activar una alarma, en nuestro caso para la aplicación a desarrollar encender el aire acondicionado, y si la temperatura disminuye aun más activar una alarma. Esto nos permite tener la perfecta tranquilidad que nunca se apagaran los servidores si la temperatura llegara a aumentar.

2.2 Tipos de Sensores

2.2.1 Termómetros de Resistencia

Los termómetros de resistencia se usan ampliamente para medir temperaturas en un intervalo de 260 a 750 ºC. El funcionamiento de los termómetros de resistencia se basa en la propiedad que tienen las sustancias de cambiar su resistencia eléctrica al variar la temperatura.

El termómetro de éste tipo se sumerge en el medio cuya temperatura mide, conociendo la variación de la resistencia del termómetro en función de la temperatura, se puede juzgar acerca de la temperatura del medio donde el mismo se halla. Además ha de tenerse en cuenta que en la mayoría de los termómetros de resistencia, la longitud del elemento sensible constituye varios centímetros y por eso, si en el medio se manifiestan gradientes de temperatura, con tal termómetro se mide cierta temperatura media de aquellas capas del medio en las que se halla su elemento sensible.

Antes se estudiaba que los materiales mas favorables para la fabricación de termómetros de resistencia eran solamente los metales puros, sin embargo las investigaciones han demostrado que también una serie de semiconductores pueden utilizarse en calidad de materiales para fabricar tales termómetros.

Es sabido que la mayoría de los metales tienen un coeficiente térmico positivo de resistencia eléctrica, el cual alcanza 0.4 -0.6 ºC en los metales puros, esto se explica por el hecho de que el numero de portadores de corriente, ósea, de electrones de conducción, es muy grande en los metales y no depende de la temperatura.

La resistencia eléctrica del metal crece al aumentar la temperatura, debido a la dispersión creciente de los electrones en las heterogeneidades de la red cristalina, motivada por el incremento de las oscilaciones térmicas de iones alrededor de sus posiciones de equilibrio. En los semiconductores se observa otro cuadro, el numero de electrones de conducción crece bruscamente con el aumento de la temperatura, por eso la resistencia eléctrica de los semiconductores típicos también disminuye bruscamente a esa misma magnitud (de ordinario, con arreglo a una ley exponencial) al calentarlos, además el coeficiente térmico de la resistencia eléctrica de los semiconductores es más alto en un orden que el de los metales puros.

2.2.2 Termómetros de Planito

El platino puro responde en sumo grado a todos los principales requisitos que se le exigen a los metales puros para la fabricación de los elementos sensibles (ES) de los termómetros de resistencia. Los termómetros con ES de alambre de platino de 0.05 a 0.10 milímetros de diámetro, se emplean en la práctica de laboratorio y en la industria para medir temperaturas de -260 a 750 ºC.

Durante el uso de éstos termómetros para medir temperaturas de -260 a -180 ºC es necesario tener en cuenta que en este caso se miden resistencias muy pequeñas, sobre todo en la parte inferior del intervalo. Por eso al medir bajas temperaturas con tales termómetros, es preciso emplear junto con éstos, aparatos que permitan medir centésimas de *ohm* con alta precisión.

En algunos casos se utiliza para medir temperaturas muy altas, por ejemplo en la practica metereologica, de hasta 1065 ºC, en éste caso se debe tomar en consideración que el platino a estas temperaturas comienza a atomizarse, por esta razón para disminuir la influencia del atomización del platino, y por consiguiente aumentar la vida útil del termómetro, el elemento sensible del termómetro de resistencia destinado a medir temperaturas de hasta 1100 ºC, se fabrica de alambre de platino de cerca de 0.5 milímetros de diámetro.

2.2.3 Termómetros bimetálicos

Todos los termómetros bimetálicos deben ser seleccionados considerando el fluido del proceso y las condiciones ambientales donde irá a operar el instrumento, una aplicación impropia puede dañar el termómetro, causar fallas en el equipo o instalación así cómo ocasionar daños al personal.

2.3 Sensor lm35d

Este sensor permite determinar la temperatura en grados centígrados de un ambiente cerrado, este puede determinar una temperatura máxima de 150 grados centígrados. El sensor lm35d será el que se utilizara para el desarrollo de nuestra aplicación.

El circuito integrado lm35d es un sensor de temperatura cuya tensión de salida es linealmente proporcional con la temperatura en la escala Celsius (centígrada). Posee una precisión aceptable para la aplicación requerida, no necesita calibración externa, posee sólo tres terminales, permite el censado remoto y es de bajo costo.

2.3.1 Funcionamiento del Sensor lm35d

La serie del lm35d fácilmente permite calibrar, los sensores de temperatura de circuito integrados. Operando con 2-terminales *zener*, los lm35 tienen un voltaje directamente proporcional con temperatura absoluta a +10 mV/°K. Con el dispositivo opera encima de un rango actual de 400 µA a 5 MA que virtualmente ningún cambio en actuación menos de 1Ohm impedancia dinámica. Cuando se calibra a las 25°C los lm35d tiene típicamente menos de 1°C error encima de un 100°C rango de temperatura. Otros sensores diferentes los lm35d tienen un rendimiento lineal.

Las aplicaciones para los lm35d incluyen casi cualquier tipo de temperatura que se da cuenta de encima de un -55°C a +150°C rango de temperatura. La impedancia baja y la hechura del rendimiento lineal que unen a lectura o circuitería del mando son especialmente fáciles.

Lm35d/lm35a/lm35 están disponibles empaquetado en TO-46 transistor hermético.

- Factor de escala : 10mV/ºC (garantizado entre 9,8 y 10,2mV/ºC)
- Rango de utilización : -55ºC < T < 150ºC
- Precisión de : ~1,5ºC (peor caso)
- No linealidad : ~0,5ºC (peor caso)

2.3.2 Diagrama y Diseño de un lm35d

La figura 2.1 muestra el diagrama del lm35d, este integrado, consta de 3 patas, una de voltaje, el cual puede ser de 4 voltios a 20 voltios, este voltaje permite determinar la magnitud de salida de voltaje del sensor, la patita del centro es la salida y esta puede variar de 0 a 1.5 milivoltios, variando en un rango de 10milivotios por grado centígrado de variación, para que el este reflejo de voltaje sede la entrada de voltaje deberá de estar a por lo menos 12 voltios. La otra patita del integrado va conectada a tierra. En la figura 2.2 nos muestra la forma conectar cada patita del integrado.

Figura 1. Diagrama del Sensor lm35d

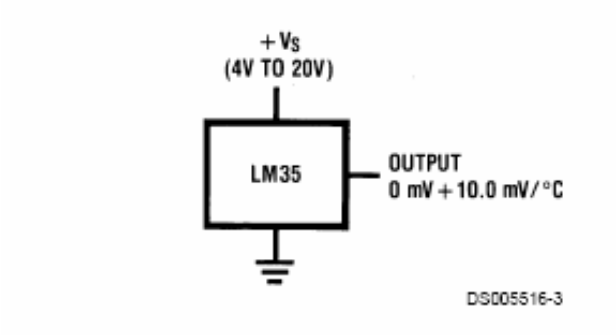
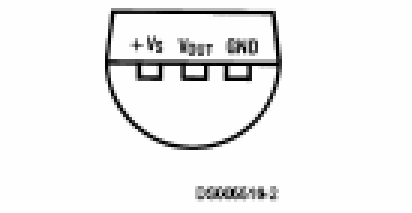
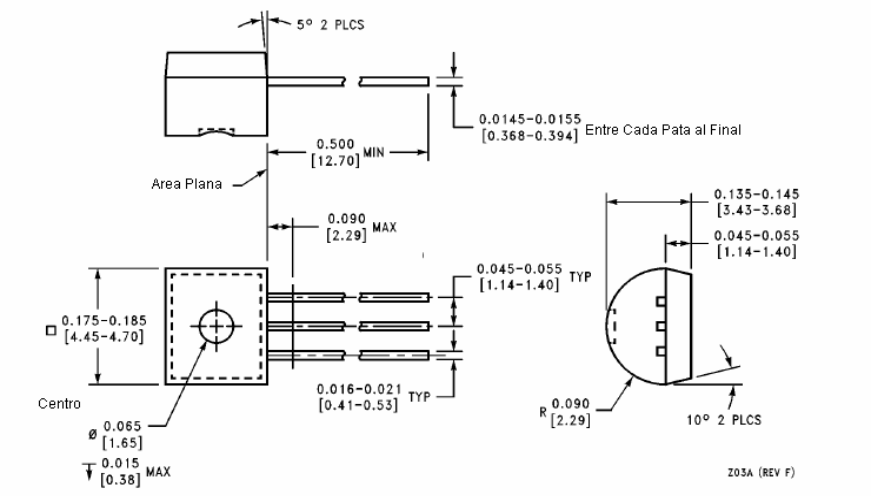


Figura 2. Diagrama del Integrado lm35caz, lm35dz, lm35cz



La figura 3 muestra las dimensiones del sensor lm35d.

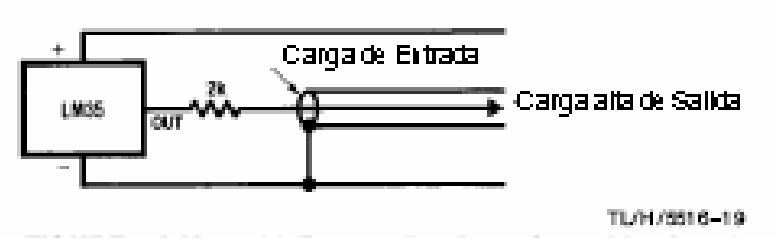
Figura 3. Dimensiones del Sensor Lm35az, Lm35dz



2.3.3 Medición de temperatura usando lm35d

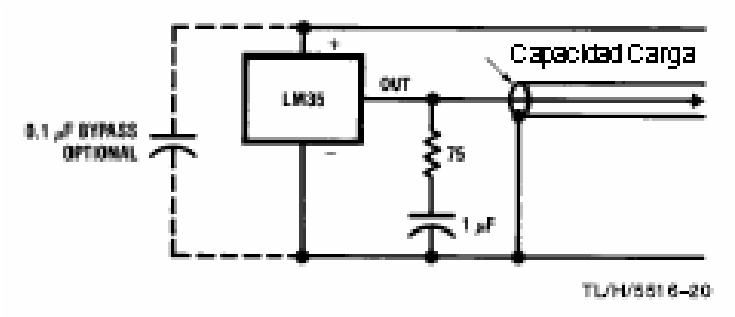
El lm35d, puede ser aplicado fácilmente, usando algunos circuitos eléctricos, de la manera que se pueda registrar la temperatura correctamente, este sensor puede medir una temperatura de hasta 0.01 grado centígrado. Esto indica que podemos medir la temperatura de cualquier nivel en un ambiente cerrado. Si la temperatura llegara a variar en algún momento debido a la circulación del aire, el sensor lm35 llegar a registrar este margen de variación de temperatura. A continuación se presenta algunos diagramas de las formas en las que se puede conectar el sensor lm35 según la forma que se leerá la temperatura.

Figura 4. Aplicación de lectura de temperatura usando un tubo de capacidad de carga



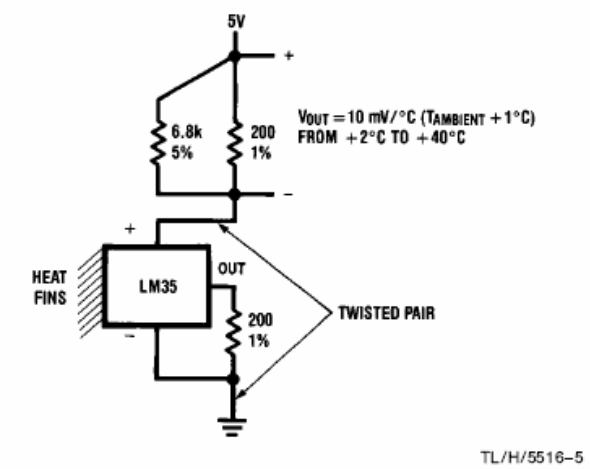
En la figura 2.4 se puede ver la como el sensor lm35d lee la temperatura desde un dispositivos de carga alta con un resistor que permite determinar la temperatura de un oscilador.

Figura 5. Aplicación de lectura de temperatura usando un tubo de capacidad de carga y capacitares



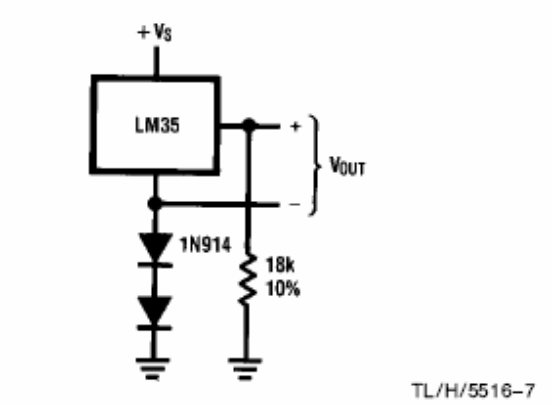
En la figura 5 se puede ver como se conecta el sensor lm35d impidiendo la tolerancia de fallos de lectura de temperatura, colocando capacitares de 1 micro faradios, lo cual hace mas certero la lectura de temperatura.

Figura 6. Lm35d conectado con sensores Remotos



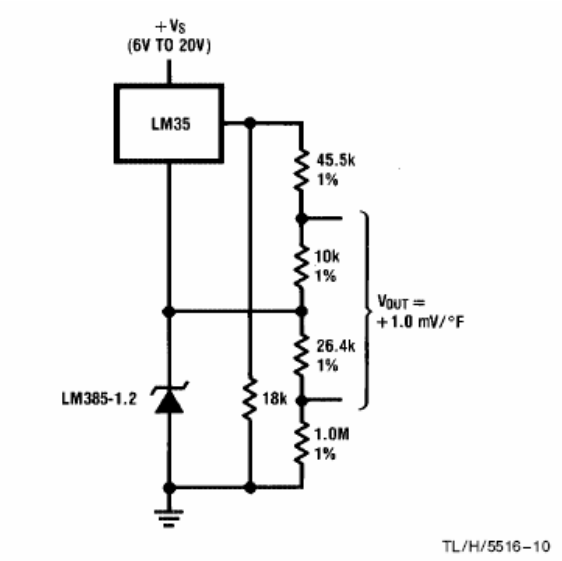
En la figura 6 nos muestra como conectar un sensor lm35d usando dos lectores de temperatura remotos. Esto nos permite determinar temperatura de 2 a 40 grados centígrados, estas aplicaciones son diseñadas para determinar la temperatura de ambientes no cerrados donde la variación sea demasiado pequeña.

Figura 7. Lm35d conectado de forma Simple



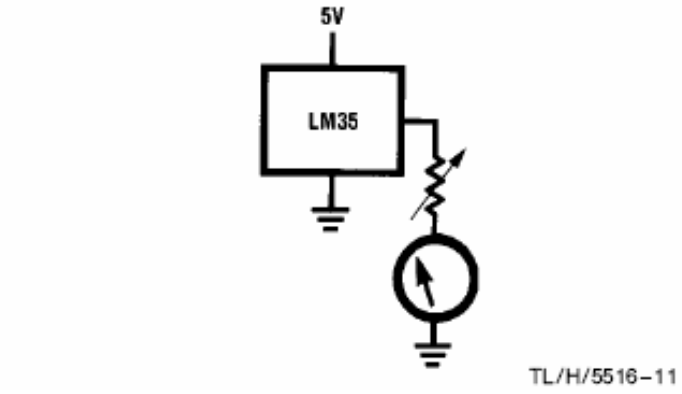
En la figura 7 el circuito permite conectar el lm35d de forma simple y sencilla para leer temperaturas ambiente de -55 grados centígrados a 150 grados centígrados.

Figura 8. lm35d como lector de temperatura Fahrenheit



En la Figura 8 el circuito nos permite leer la temperatura de un ambiente cerrado en grados Fahrenheit, esto sirve para países que no trabajan con el sistema centígrado, los cuales son los países europeos, usa, etc.

Figura 9. Lm35d con lector de temperatura análogo



El diagrama de la figura 9 muestra como conectar un lector de temperatura análogo hacia un sensor de temperatura lm35d, esos sensores análogos son muy utilizados en la industria de la refrigeración.

Existen infinidad de aplicaciones en las cuales se puede aplicar el sensor lm35d, en nuestro caso utilizaremos el diagrama 7 que es un lector de temperatura sencillo.

2.4 Temperatura normal en un servidor

Lo primero que se debe hacer es un estudio de las temperaturas a las que funciona nuestro procesador en los cuales se instalara la aplicación, debido a que según la marca y el modelo o ya sea una PC o un servidor sus temperaturas son diferentes según la situación del procesamiento. Para esto es extremadamente importante tener en cuenta que ésta puede variar dependiendo de los siguientes factores:

- Temperatura ambiente
- Porcentaje de uso de la CPU y tiempo que esta sometida a grandes cargas
- Uso intensivo de discos duros y lectores de CD (lo cuales suelen producir mucho calor)
- Condiciones de la habitación (humedad, ventilación, etc...)

A continuación se escribe un cuadro con las temperaturas de trabajo de una PC de escritorio; debemos tomar en cuenta que estas pueden variar según la época del año (ya sea verano e invierno) y el tipo de procesador que se utilice.

Trabajando en modo de máximo rendimiento (P0):

Tabla I. Máximo rendimiento del CPU

Uso	**Tipo de Operación**	**Temperatura**
Uso Moderado	Trabajo habitual con escritorios, editores,…	50ºC a 55ºC
Uso alto	Por ejemplo recompilar el kernel	50ºC a 60ºC
Uso muy alto	Por ejemplo comprimir Divx de un DVD	55ºC a 68ºC

Trabajando en modo de bajo rendimiento (P1):

Tabla II. Bajo rendimiento del CPU

Uso	**Tipo de Operación**	**Temperatura**
Uso Moderado	Trabajo habitual con escritorios, editores,…	50ºC a 55ºC
Uso alto	Por ejemplo recompilar el kernel	50ºC a 55ºC
Uso muy alto	Por ejemplo comprimir Divx de un DVD	55ºC

Como se puede observar en estos cuadros, cuando se hace trabajar a la máquina a máxima carga durante un tiempo muy largo, y en el caso de estas mediciones en circunstancias desfavorables, la temperatura tiende a subir mucho. Esto puede provocar que la máquina alcance temperaturas no deseadas, y en nuestro caso, que el procesador se ponga a 68ºC puede llegar a detenerse el funcionamiento de un servidor, aunque según sus especificaciones técnicas esta no se considere una temperatura excesiva para su funcionamiento.

Una vez realizado un estudio de las temperaturas a las que funciona nuestro procesador es necesario decidir que rango de temperaturas consideramos adecuado para él. En este punto es muy importante ser lo más realista posible y adecuarnos al estudio realizado en el punto anterior, es decir, por mucho que se quiera, un procesador no va a funcionar a 75ºC a no ser que cambie el sistema de refrigeración de la máquina por uno más potente. Es por eso que la aplicación a desarrollar no permitirá exceder la temperatura en un margen deseado.

3. Metodología de desarrolló de la aplicación usando pic16f877 y sensor lm35d

3.1 Definición del problema

El problema realmente es que existen muchas entidades privadas y publicas que no tienen un grado de seguridad sobre la temperatura de sus sistemas de computo, esto es debido a que como cuentan con aire acondicionada lo dejan todo a este sistema sin ponerse a pensar que sucedería si realmente el aire es apagado por error o se descompone en un momento donde no exista alguien tan cerca para encontrar una solución la cual implicaría que los servidores llegaran a pararse y dejar el sistema sin funcionar lo que implica que se pueden llegar a tener perdidas si fuera una empresa multinacional que realiza operaciones en todo momento.

3.2 Descripción de la solución

El proyecto a implementar, es un medidor de temperatura para ambientes cerrados, útil en centros de cómputo, permitiendo así que la temperatura se mantenga en un nivel en el cual los servidores puedan trabajar a su máximo rendimiento, sin llegar a calentarse, esto es debido a que mientras mas caliente este el procesador, su rendimiento disminuye.

El circuito será controlado por un PIC 16f877 este llevara internamente el programa que permitirá administrar los procesos e interrupciones, además el PIC leerá la temperatura por medio de un sensor de lm35d el cual es un integrado que nos permite leer de 0 a 150 grados Celsius. La temperatura podrá programarse en el PIC debido a que esta varía según los servidores y las marcas. La programación del PIC se desarrollara a través de un quemador y el programa fuente será compilado en la aplicación *MPLAB*.

Al momento de registrar la temperatura mínima el PIC deberá de tomar la decisión de que hacer (encender una alarma o activar el aire acondicionado) si la temperatura llega mas abajo de lo deseado entonces se deberá de enviar un mensaje de correo urgente al administrador o encargado para que este sepa que esta pasando y tome por entendido que debe de hacer para no provocar ninguna falla en los servidores.

El PIC leerá la señal del sensor en la pata 2 luego se ara la conversión para determinar que temperatura esta leyendo y así empezar el funcionamiento. La salida será en la pata 14 del PIC, esta enviara una salida de 5 voltios la cual amplificaremos para activar un releí y se active el aire acondicionado o la alarma.

También se enviara la señal al puerto paralelo de una computadora que simulara un servidor de correo para enviar el correo a las personas encargadas, así podrá llegar la que mas cerca se encuentre si en dado caso surgiría un problema.

3.3 Restricciones del proyecto

La única restricción que podría existir es que se necesitaría Internet en todo momento para asegurar el perfecto funcionamiento, debido a que si la temperatura baja mucho como se describió anteriormente, se estaría enviando un mensaje a las personas correspondientes.

3.4 Equipo a utilizar

Necesitamos una computadora a la cual se conecte el circuito con un simulador de correo o servidor de correo.

Conexión a Internet las 24 horas del día los 365 días del año para poder enviar correos en cualquier momento, permitiendo garantizar el perfecto funcionamiento de nuestra aplicación.

Un integrado PIC 16f877, un sensor lm35d, dos *Display* para ver la temperatura, dos integrados 77LS48 un amplificador de señal, y un Releí.

Un quemador para el PIC, una Fuente para el quemador.

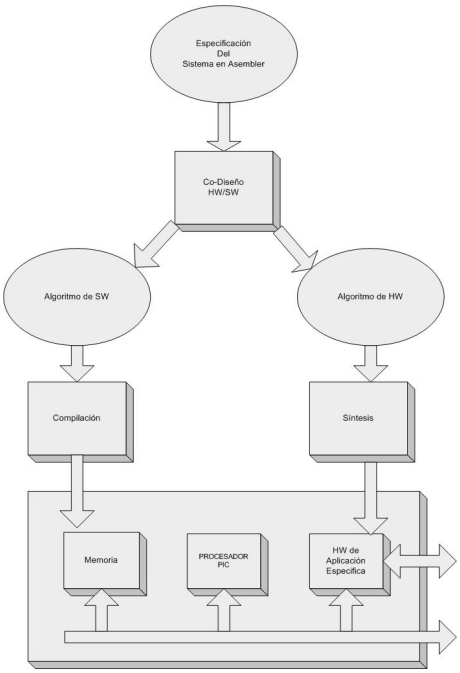
El programa de compilación del código fuente *MPLAB* y el programa para cargar nuestro software al PIC.

Una placa de cobre para montar el circuito.

3.5 Metodología

La metodología de diseño propuesta para nuestra aplicación se muestra en la figura 10. La cual es una metodología de carácter general, llamada también metodología de codiseño, es fácilmente extensible a la utilización de código asembler.

Figura 10. Metodología Codiseño



La palabra codiseño implica la especificación e implementación de un sistema a través del desarrollo simultáneo e integrado de módulos, correspondiendo estos a diferentes áreas de diseño. Específicamente codiseño hardware-software se refiere al diseño e implementación concurrente de partes software y hardware como elementos de un solo sistema.

Una definición más formal podría ser la interrogante dada por *Franke y Purvis.* “Codiseño Hw/Sw es el proceso de diseño de un sistema que combina las perspectivas hardware y software desde los estados primarios, para aprovechar la flexibilidad del diseño y la localización eficiente de las funciones”.

A continuación se justifica el por qué del la metodología de codiseño.

- Aumenta la diversidad y complejidad de aplicaciones que exigen alta eficiencia de desarrollo y elevado desempeño.

- Espacio de diseño muy amplio, necesidad de explorar dentro de éste de una manera eficiente una buena solución

- Necesidad de explorar más y nuevas alternativas de diseño para lograr mejores sistemas.

- Tendencia hacia las soluciones denominas ‘*Embedded Systems*” o sistemas embebidos. Las cuales son arquitecturas de circuitos integrados que incorporan elementos microprocesador que por naturaleza, exigen como solución un sistema hardware-software.

- Exigencia de los sistemas actuales para satisfacer rigurosas restricciones de desempeño y confiabilidad entre otras.

- Necesidad de manejar los procesos de manera coherente, con la obvia exigencia de simplificar costos de diseño.

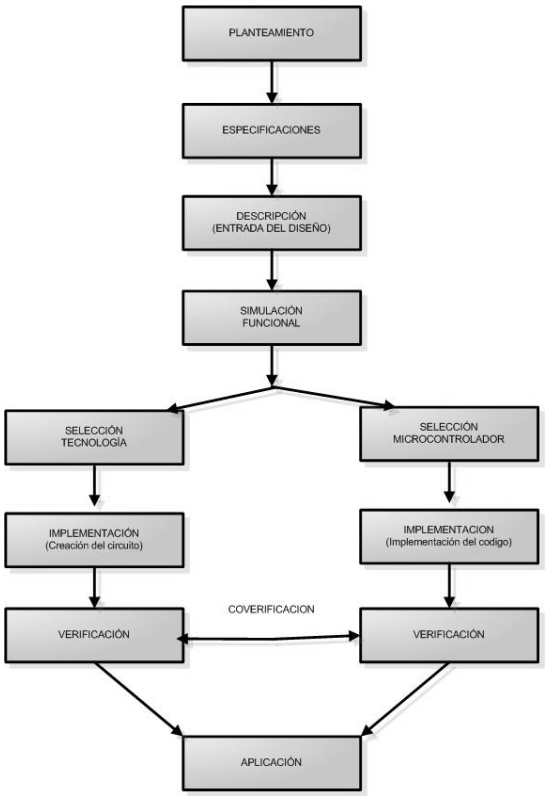
- Facilitar de manera confiable el diseño de *SOCs (‘Systems on a Chip’).*

Una Metodología se deriva de los procesos obvios de diseño donde de manera estructurada y sistemática se lleva el planteamiento de un problema a una solución óptima que respeta los requerimientos y restricciones originales.

3.6 Descripción de las etapas de desarrollo para el proyecto

3.6.1 Diseño de sistema bajo la metodología elegida

Figura 11. Metodología de codiseño a implementar



3.6.2 Planteamiento

Lo que se necesita es llegar a desarrollar una aplicación en la cual podamos medir la temperatura de un laboratorio o centro de cómputo, así determinar si estas están trabajando a su máximo rendimiento cada una de ellas y poder realizar una acción si la temperatura varía demasiado. De modo que el sistema se mantenga el cien por ciento en línea.

3.6.3 Especificaciones

La temperatura se mide en grados centígrados y esta debe de ser registrada por un dispositivo para que cuando aumente mas de los normal que necesita un servidor para que trabaje a su máximo rendimiento, nivele la temperatura encendiendo el aire acondicionado, si la temperatura en lugar de bajar un grado, sube uno más, este debería de enviar un correo a la persona encargada para que esta tome nota del asunto.

3.6.4 Descripción

Como entrada al sistema determinaremos la temperatura que existe en el ambiente, esta temperatura será registrada por un sensor en el cual determinaremos los nivel mas altos o bajos que podrían existir, la temperatura debería de ser la de un ambiente cerrado debido a que un ambiente en donde exista mucha ventilación podría ocasionar que no se registre la que realmente sea.

3.6.5 Selección de la tecnología Hardware

La tecnología ha elegir en el caso del hardware es el microcontrolador PIC16F877 este en un dispositivo electrónico que tiene todas la características de una computadora y nos ayudara a la toma de decisiones.

Para la lectura de la temperatura se a decido utilizar el sensor lm35d debido a que es un sensor fácil de conseguir, como y llena todas la perspectivas de desarrollo.

3.6.7 Selección de la tecnología de software

En el caso del software el código se desarrollara en *asembler* debido a que este tipo de controlador solo acepta este tipo de instrucción, además utilizaremos un software que permite compilar este código a código hexadecimal para el traspaso al PIC.

3.6.8 Descripción a nivel del sistema

- Entradas: Como única entrada tendremos la temperatura ambiente registrada dentro del cuarto de cómputo.
- Salidas: La salida principal será la activación del aire acondicionado para bajar el nivel de temperatura. Otra salida en el sistema será el envió de un mensaje ala persona encargada.
- Validaciones: Se definirá la temperatura ambiente que debería aceptar como mínima un servidor sobre esta temperatura se hará dicha validación.

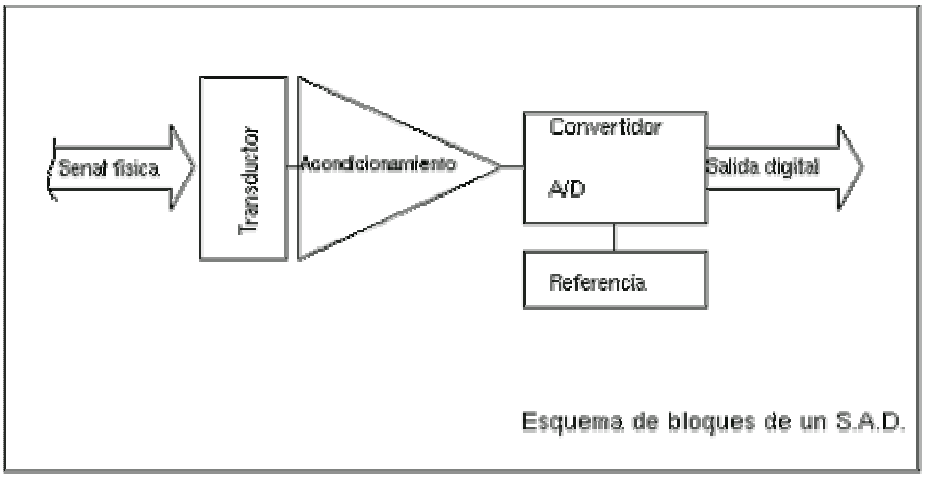
3.7 Programación del PIC

Para programar el PIC necesitamos definir la forma en la que leeremos los datos de entradas y como manejaremos los convertidores A/D. Es importante tener en cuenta que es lo que estamos recibiendo y como lo procesaremos para tomar la decisión de realizar una operación en un determinado tiempo.

3.7.1 Adquisición de datos

Un sistema de adquisición de datos es un equipo que nos permite tomar señales físicas del entorno y convertirlas en datos que posteriormente podremos procesar y presentar. A veces el sistema de adquisición es parte de un sistema de control, y por tanto la información recibida se procesa para obtener una serie de señales de control la grafica de la figura 12 nos muestra el proceso de convertir una señal física a una decisión digital para ejecutar un actuador.

Figura 12. Diagrama que muestra el proceso de convertir una señal física a una decisión digital



3.7.1.1 Estructura del sistema de adquisición de datos

En la figura 12 podemos ver los bloques que compondrán nuestro sistema de adquisición de datos. Como vemos, los bloques principales son estos:

- El transductor.
- El acondicionamiento de señal.
- El convertidor analógico-digital.
- La etapa de salida (interfaz con la lógica).

El transductor es un elemento que convierte la magnitud física que vamos a medir en una señal de salida (normalmente tensión o corriente reflejando la temperatura) que puede ser procesada por nuestro sistema. Salvo que la señal de entrada sea eléctrica, podemos decir que el transductor es un elemento que convierte energía de un tipo en otro.

Por tanto, el transductor debe tomar poca energía del sistema bajo observación, para no alterar la medida. El acondicionamiento de señal es la etapa encargada de filtrar y adaptar la señal proveniente del transductor a la entrada del convertidor **analógico / digital**. Esta adaptación suele ser doble y se encarga de:

- Adaptar el rango de salida del transductor al rango de entrada del convertidor. (Normalmente en tensión).
- Acoplar la impedancia de salida de uno con la impedancia de entrada del otro.

La adaptación entre los rangos de salida del convertidor y el de entrada del convertidor tiene como objetivo el aprovechar el margen dinámico del convertidor, de modo que la máxima señal de entrada debe coincidir con la máxima que el convertidor (pero no con la máxima tensión admisible, ya que para ésta entran en funcionamiento las redes de protección que el convertidor lleva integrada). Por otro lado, la adaptación de impedancias es imprescindible ya que los transductores presentan una salida de alta impedancia, que normalmente no puede excitar la entrada de un convertidor, cuya impedancia típica suele estar entre 1 y 10 k. El convertidor Analógico / Digital es un sistema que presenta en su salida una señal digital a partir de una señal analógica de entrada, (normalmente de tensión) realizando las funciones de **cuantificación** y **codificación.**

La cuantificación implica la división del rango continuo de entrada en una serie de pasos, de modo que para infinitos valores de la entrada la salida sólo puede presentar una serie determinada de valores. Por tanto la cuantificación implica una pérdida de información que no podemos olvidar.

La codificación es el paso por el cual la señal digital se ofrece según un determinado código binario, de modo que las etapas posteriores al convertidor puedan leer estos datos adecuadamente. Este paso hay que tenerlo siempre en cuenta, ya que puede hacer que obtengamos datos erróneos, sobre todo cuando el sistema admite señales positivas y negativas con respecto a masa, momento en el cual la salida binaria del convertidor nos da tanto la magnitud como el signo de la tensión que ha sido medida. La etapa de salida es el conjunto de elementos que permiten conectar el s.a.d con el resto del equipo, y puede ser desde una serie de buffer digitales incluidos en el circuito convertidor, hasta un interfaz para el encendido del aire acondicionado o una RS 232, RS 485 o *Ethernet* para conectar a un ordenador o estación de trabajo en cual será nuestro caso para el envió de mensajes.

3.7.2 Convertidor Análogo/Digital

A continuación se describen las características esenciales que hemos de tener en cuenta para realizar nuestras medidas de un modo decente.

3.7.2.1 Características básicas del convertidor Análogo/Digital

Las características que no debemos olvidar son éstas:

- Impedancia de entrada
- Rango de entrada
- Número de bits
- Resolución

- Tensión de fondo de escala
- Tiempo de conversión
- Error de conversión

Hay una serie de características que son comunes a otros tipos de circuitos que no se detallaran, aunque siempre hay que tener en cuenta, como la impedancia de entrada, *fan-out*, etc. **Número de bits:** Es el número de bits que tiene la palabra de salida del convertidor, y por tanto es el número de pasos que admite el convertidor. Así un convertidor de 8 bits sólo podrá dar a la salida 2^8=256 valores posibles. **Resolución:** Es el mínimo valor que puede distinguir el convertidor en su entrada analógica, o dicho de otro modo, la mínima variación, Vi, en el voltaje de entrada que se necesita para cambiar en un bit la salida digital. Por lo tanto, tenemos que:

$$Vi = \frac{Vfe}{(2^n - 1)}$$

Donde n es el número de bits del convertidor, y Vfe la tensión de fondo de escala, es decir, aquella para la que la salida digital es máxima y pueda activar el aire acondicionado. La tensión de fondo de escala depende del tipo de convertidor, pero normalmente se fija a nuestro gusto, en forma de una tensión de referencia externa. Por ejemplo, un convertidor de 8 bits con una tensión de fondo de escala de 2V tendrá una resolución de:

$$\frac{2V}{2^8 - 1} = 7.84 \frac{mV}{paso}$$

En cambio, para el mismo convertidor, si cambiamos la tensión de referencia, y por tanto la de fondo de escala, la resolución será de:

$$\frac{5V}{2^8 - 1} = 19.6 \frac{mV}{paso}$$

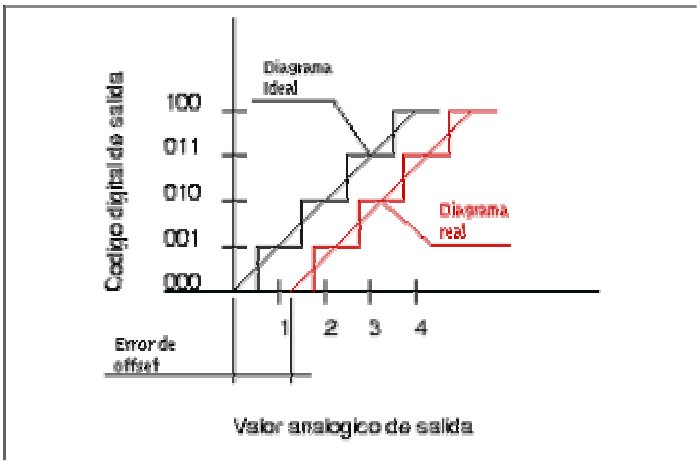
3.7.2.2 Tipos de errores en los convertidores Análogo/Digital

Un convertidor no es un circuito perfecto, sino que presenta una serie de errores que debemos tener en cuenta. Algunos de los que más importancia tiene son los siguientes:

Error de *offset*: El error de *offset* es la diferencia entre el punto nominal de *offset* (cero) y el punto real de *offset*. Concretamente, para un convertidor A/D este punto es el punto central de todos aquellos valores de la entrada que nos proporcionan un cero en la salida digital del convertidor. Este error afecta a todos los códigos de salida por igual, y puede ser compensado por un proceso de ajuste.

Error de cuantificación: Es el error debido a la división en escalones de la señal de entrada, de modo que para una serie de valores de entrada, la salida digital será siempre la misma. Este valor se corresponde con el escalonado de la función de transferencia real, frente a la ideal. Podemos verlo en esta figura 13 que cada valor digital tiene un error de cuantificación de +- ½ LSB (bite menos significativo). Por tanto, cada código digital representa un valor que puede estar dentro del ½ LSB a partir del punto medio entre valores digitales continuos.

Figura 13. Errores de cuantificación



Error de linealidad (linealidad integral): Este error es la manifestación de la desviación entre la curva de salida teórica y la real, de modo que para iguales incrementos en la entrada, la salida indica distintos incrementos.

Error de apertura: Es el error debido a la variación de la señal de entrada mientras se está realizando la conversión. Este error es uno de los más importantes cuando se están muestreando señales alternas de una frecuencia algo elevada, (como por ejemplo el muestreo de voz) pero tiene poca importancia cuando medimos señales cuasi-continuas, como temperatura, presión, o nivel de líquidos. Para minimizar este tipo de error se usan los circuitos de muestreo y retención. Este error es importante, ya que si no lo tenemos en cuenta raramente podemos digitalizar adecuadamente señales alternas. Si consideramos un error que no afecte a la precisión total de la conversión, (por lo que habrá de ser menor que ½ LSB) la frecuencia máxima de muestreo deberá ser:

$$Fmax = \frac{1}{Ta * \pi * 2^{n+1}}$$

En esta fórmula Ta es el tiempo de apertura del circuito de muestreo y retención, o bien el tiempo total de conversión si el anterior no existe, y n el nº de bits del convertidor.

3.7.3 Realización del código

Después de haber determinado como recibiremos los datos y como será convertida la señal análoga digital a través de un puerto de PIC realizaremos el programa que utilizaremos en nuestra aplicación.

3.7.3.1 Variables del código utilizado

Antes de empezar a escribir código para un PIC se necesitan color tres cosas importes:

```
list p=16f877
RADIX HEX
include "p16f877.inc"
```

El *list* nos indica el tipo de procesador que vamos a utilizar, el *RADIX HEX* la base numérica y por ultimo se incluyen las directivas del PIC que la que usamos será PIC16F877.

Otros comandos utilizados son:

1. *ADRESH.* Parte alta del resultado de la conversión.
2. *ADRESL*. Parte baja del resultado de la conversión.
3. *ADCON0*. Registro de Control 0.
4. *ADCON1*. Registro de Control 1.

En la pareja de registro *ADRESH: ADRESL* se deposita el resultado de la conversión, que al estar compuesta por 10 bits, sólo son significativos 10 de los bits de dicha pareja.

El registro *ADCON0* controla la operación del Convertidor Analógico-Digital, mientras que el *ADCON1* sirve para configurar las patitas del Puerto A como entradas analógicas o E/S digitales. Se seleccionará el canal 0 (RA0) para introducir la señal analógica a convertir.

El *BIT GO / DONE* es el bite de estado de la conversión. Poniéndolo a 1 se inicia la conversión y mientras esté a 1 está realizándose dicha operación. Cuando *GO / DONE* pasa a 0 confirma el final de la conversión y la puesta del resultado en la pareja de registros *ADRESH: ADRESL*. El bite *ADON* sirve para activar el Convertidor A / D poniéndolo a 1 y para inhibir su funcionamiento poniéndolo a 0.

3.7.3.2 Implementación del programa

```
list p=16f877
RADIX    HEX
include "p16f877.inc"
errorlevel 1, -302
```

```
contador equ  0x20
BCD_0         equ  0x21
BCD_1         equ  0x22
BCD_2         equ  0x23
L_BYTE        equ  0x24
H_BYTE        equ  0x25
TEMP          equ  0x26
L_BYTE2       equ  0x27
H_BYTE2       equ  0x28
GRADOS        equ  0x29
COUNT_L       equ  0x2A
COUNT_H       equ  0x2B
CONTADOR2     equ  0x2C
SUMA_VAL_L    equ  0x2D
SUMA_VAL_H    equ  0x2E
DATO1         equ  0x2F
DATO2         equ  0x30
VAL_DIV       equ  0x31
SEGUNDO       equ  0x32
      org     0x000
      nop
      goto inicio
      org     0x005
velocidad
      banksel STATUS
      bcf  STATUS,C
      banksel DATO1
      movlw   b'00000011'
      andwf   DATO2,F
      swapf   DATO2,F
      movf    DATO2,W
      banksel CCP1CON
      iorwf   CCP1CON,F
      movf    DATO1,W
      banksel CCPR1L
      movwf   CCPR1L
      return
Bits16aBCD_C
      bcf  STATUS,C
      movlw   0X10
      movwf   contador
      clrf  BCD_0
      clrf  BCD_1
      clrf  BCD_2
```

```
ciclo16
        rlf    L_BYTE,F
        rlf    H_BYTE,F
        rlf    BCD_2,F
        rlf    BCD_1,F
        rlf    BCD_0,F
        decfsz      contador,F
        goto        ajuste_decimal
        retlw       0
ajuste_decimal
        movlw       BCD_2
        movwf       FSR
        call ajuste_BCD
        movlw       BCD_1
        movwf       FSR
        call ajuste_BCD
        movlw       BCD_0
        movwf       FSR
        call ajuste_BCD
        goto        ciclo16
ajuste_BCD
        movlw       0x03
        addwf       0x00,W
        movwf       TEMP
        btfsc       TEMP,0x03
        movwf       0x00
        movlw       0x30
        addwf       0x00,W
        movwf       TEMP
        btfsc       TEMP,7
        movwf       0x00
        retlw       0x00
DELAY_100MS
        movlw       0x82
        movwf       COUNT_H
        clrf  COUNT_L
DELAY_SEGUNDO
        movlw       0x0D
        movwf       SEGUNDO
        loop_SEGUNDO
        call DELAY_100MS
        decfsz  SEGUNDO, f
        goto        loop_SEGUNDO
```

```
        return
D_LOOP
        decfsz      COUNT_L, f
        goto        D_LOOP
        decfsz      COUNT_H, f
        goto        D_LOOP
        return
```

Esto sirve para pasar del valor que genero el ADC (que seria Y) al valor real en grados, bajo la suposición de que 1.5 volteos equivalen a 150 grados y 0 volteos a cero grados la temperatura se debe mantener dentro de 32 y 45 grados la formula a seguir es Y - GRADOS * 2 + (GRADOS / 22).

```
CONVIERTE_GRADOS
        banksel     SUMA_VAL_L
        clrf  SUMA_VAL_L          ;limpiar las variables
        clrf  SUMA_VAL_H
        clrf  GRADOS
        clrf  CONTADOR2
        clrf  VAL_DIV
SUMA_GRADOS
        banksel     H_BYTE
        movf        H_BYTE,W              ;Hacemos una copia de los datos leídos por el ADC
        movwf       H_BYTE2
        movf        L_BYTE,W
        movwf       L_BYTE2
        movlw       0x01
        addwf       GRADOS,1              ;Incremento en uno el contador de grados
        clrf  SUMA_VAL_L
        clrf  SUMA_VAL_H
        movlw       0x02
        movwf       CONTADOR2;Vamos a multiplicar GRADOS por 2
                             ; Para ello realizamos una serie de sumas
SUMA
        movf        GRADOS,W
        addwf       SUMA_VAL_L,1        ;SUMA_VAL_L = SUMA_VAL_L + GRADOS
        btfss       STATUS,C
        goto        NO_SUMAR
        movlw       0x01
        addwf       SUMA_VAL_H,1
NO_SUMAR
        decfsz      CONTADOR2,f
        goto        SUMA
        clrf  CONTADOR2
        movf        GRADOS,W
```

```
	movwf	VAL_DIV		;tiene el valor que queremos dividir dentro de 22
	; Para ello realizamos una serie de restas hasta que el residuo sea negativo
RESTAR_F
	movlw	0x01
addwf CONTADOR2,1  ; Incremento en uno al contador  CONTADOR = CONTADOR + 1
	movlw	0x16		; W = 16h = 22 decimal
	subwf	VAL_DIV,1	; VAL_DIV = VAL_DIV - W
	btfsc	STATUS,C	; Verificar si el BIT C esta en uno, si es uno salte ya
;que es menor
	goto	RESTAR_F
	movlw	0x01
	subwf	CONTADOR2,1	; ajusto el valor de CONTADOR
	movf	CONTADOR2,W	; sumamos el resultado de la división
	addwf	SUMA_VAL_L,1
	btfss	STATUS,C
	goto	NO_ACARREO2
	movlw	0x01
	addwf	SUMA_VAL_H,1
NO_ACARREO2
		; Ahora restamos los valores y vemos si ya es negativo, de no serlo se
;Repetira todo el proceso
		; Al valor que genero el ADC le resto el valor que yo encontré con la
		 ; Formula anterior
	movf	SUMA_VAL_L,W
	subwf	L_BYTE2,F
	btfsc	STATUS,C	;Si el resultado es negativo quiero restar 1 a la parte alta
	goto	NO_ACARREO3
	;Primero debo verificar si la parte alta tiene un valor diferente de cero
	movlw	0XFF
	andwf	H_BYTE2,W
	btfsc	STATUS,Z	Si se genero un cero H_BYTE  tiene cero, entonces salto al
;Final de la conversion
	goto	FIN_COMP
	movlw	0x01
	subwf	H_BYTE2,F
NO_ACARREO3
	movf	SUMA_VAL_H,W
	subwf	H_BYTE2,F
	btfsc	STATUS,C	;Si el resultado es negativo salto a FIN_COMP porque ya
; Encontró el valor de grados
	goto	SUMA_GRADOS

FIN_COMP	; Aquí termina la comparación, en GRADOS queda el valor real de
; Grados leído
```

```
            ;le resto 1 para ajustar el valor
        movlw       0x01
        subwf       GRADOS,F
        btfss       STATUS,C    ;si el resultado es negativo indica que paso de cero a 255
;debo corregir el dato en cero
        clrf  GRADOS        ;lo limpio, todo a cero  o ponerle el valor cero
RANGOS
                            ; Ahora veo si el GRADOS es inferior al rango de 32
        movlw       .00             ;este valor se puede cambiar por el del rango inferior
        subwf       GRADOS,W
        btfss       STATUS,C
        goto        ACTIVA_PIND
                            ; Ahora veo si GRADOS es superior al rango de 45
        movlw       .37             ;este valor se puede cambiar por el del rango superior, se
;coloca un valor mayor
        subwf       GRADOS,W
        btfsc       STATUS,C
        goto        DESACTIVA_PIND
ACTIVA_PIND
        bsf  PORTC,3            ;enciendo la lámpara, en el pin 3 del puerto C
        goto        sale55
DESACTIVA_PIND
        bcf  PORTC,3            ;apagar la lámpara,
        goto        sale55
sale55
        movf        GRADOS,W
        movwf       L_BYTE
        clrf  H_BYTE
        call Bits16aBCD_C
; Los resultados quedan en BCD_2 al BCD_0  ,  los dos menos significativos
; En BCD_2 y los más en BCD_0
; ejemplo  para el numero 198765      65 en BCD_2,  87 en BCD_1 y 19 en BCD_0 en
;formato BCD
        banksel     PORTD
        clrf  PORTD
        movf        BCD_2,W             ;envió el valor al puerto D
        movwf       PORTD
        return

;-----------  PROGRAMA PRINCIPAL  ----------------------
inicio
        banksel ADCON0
        movlw       b'11000001'
        movwf       ADCON0
```

```
        movlw       b'10000000'
        banksel ADCON1
        movwf       ADCON1
        banksel INTCON
        bsf  INTCON, PEIE
        bsf  INTCON, GIE
            ; Configuración del modulo PWM basado en la lectura del convertidor ADC
            ; Con una presicion de 8 bits
        banksel     T2CON
        movlw       b'00000111'   ;habilitar el TIMER2 con predivisor de 16 BITS
        movwf       T2CON
        movlw       .255          ;carga periodos al PR2
        banksel     PR2
        movwf       PR2
        movlw       b'0001100'    ;modulo PWM configurado
        banksel     CCP1CON
        movwf       CCP1CON
        banksel     CCPR1L
        clrf  CCPR1L              ;limpiar el registro de trabajo del PWM
             ; configuramos el puerto C para controlar el modulo PWM
        banksel TRISC     ;registro de control del puerto C
        movlw       b'00000000'   ;modulo ccp1 de salida
        movwf       TRISC
                            ; configuramos el puerto D para  los displays
        banksel TRISD
        movlw       b'00000000'
        movwf       TRISD         ;el puerto D será de salidas digitales
        banksel PORTD
        clrf    PORTD
Inicio_conversion
        banksel     H_BYTE2
        clrf  H_BYTE
        clrf  L_BYTE
        clrf  GRADOS
        clrf  SUMA_VAL_H
        clrf  SUMA_VAL_L
        clrf  CONTADOR2
        banksel     PIR1
        bcf  PIR1,ADIF
        banksel ADCON0
        bsf  ADCON0,GO

espera
        btfss       PIR1,ADIF
```

```
goto        espera
bcf  PIR1,ADIF

banksel     ADRESH
movf        ADRESH,W
banksel     DATO2
movwf       DATO2         ;mueve el dato de referencia para PWM
banksel H_BYTE
movwf       H_BYTE
banksel     ADRESL
movf        ADRESL,W
banksel     DATO1
movwf       DATO1         ;mueve el dato de referencia para PWM
banksel L_BYTE
movwf       L_BYTE
call velocidad      ;procedimiento que cambia la velocidad del PWM
call CONVIERTE_GRADOS
call DELAY_SEGUNDO  ;retardo entre conversiones
goto        inicio_conversion
end
```

3.7.4 Configurar programa en *MPLAB*

El *MPLAB* es un entorno de desarrollo integrado que permite escribir y codificar los microcontroladores PIC de Microchip para ejecutarlos. El *MPLAB* incluye un editor de texto, funciones para el manejo de proyectos, un simulador interno y una variedad de herramientas que lo ayudarán a mantener y ejecutar su aplicación. También provee una interfase de usuario para todos los productos con lenguaje *Microchip*, programadores de dispositivos, sistemas emuladores y herramientas de tercer orden.

El *MPLAB* permite realizar las siguientes tareas:

- Manejar el escritorio *MPLAB*

- Crear un nuevo archivo de código fuente para el ensamble e ingresarlo a un nuevo proyecto para el 16F84
- Identificar y corregir los errores simples
- Ejecutar el simulador interno
- Marcar puntos de interrupción
- Crear ventanas de observación
- Manejar ventanas para el seguimiento de errores

Pasos para cargar un programa en *MPLAB*

Paso 1: Instalación
Paso 2: Configurar el Modo de Desarrollo
Paso 3: Crear un nuevo Proyecto simple
Paso 4: Abrir un nuevo Archivo fuente simple
Paso 5: Ensamble del Archivo fuente
Paso 6: Ejecución de su programa

Pasó 1: Instalación

Descargue los archivos del software de instalación y ejecute el archivo MPxxxxx.EXE. Estos archivos pueden ser transferidos a diskettes si desea instalar el *MPLAB* en otra computadora. De acuerdo a la versión que ha descargado, los nombres de los archivos pueden variar levemente. Por ejemplo, la versión 4.00 del *MPLAB* tendría los siguientes archivos:

MP40000.EXE - MP40000.WO2 - MP40000.WO3 - MP40000.WO4 - MP40000.WO5 - MP40000.WO6

Cuando ejecuta el archivo .EXE, comenzará la instalación del *MPLAB* en su sistema. Seguidamente deberá elegir los componentes del *MPLAB* que desea instalar en su sistema.

Pasó 2: Configurar el Modo de Desarrollo

El escritorio básico del *MPLAB* se asemeja al de las aplicaciones de *Windows* (como pudo ver en la pantalla previa). Tiene una barra de menú en el margen superior, una barra de herramientas, y también una barra de estado en el margen inferior. Podrá advertir que la barra de estado incluye información sobre cómo se ha configurado el sistema.

Pasó 3: Crear un nuevo Proyecto simple

El simulador se ejecutará desde el mismo archivo, llamado "archivo hex", el cual puede ser programado en el micro PIC. Para que se ejecute el simulador, primero deberá crear un archivo de código fuente y realizar el montaje del código fuente.

Pasó 4: Abrir un nuevo Archivo fuente simple

Use la opción de menú "Archivo>Abrir...", y abra el archivo creado con el programa fuente.

Pasó 5: Ensamble del Archivo fuente

El ensamble del archivo puede realizarse de varias maneras. Aquí describiremos un método. Use el ítem de menú "Proyecto>Construir todo".

De este modo ejecutará el lenguaje ensamblador *MPASM* en el trasfondo usando las configuraciones guardadas con el proyecto anteriormente. Una vez completado el proceso de ensamble, aparecerá la siguiente ventana "Resultados de Construcción":

Pasó 6: Ejecución de su programa

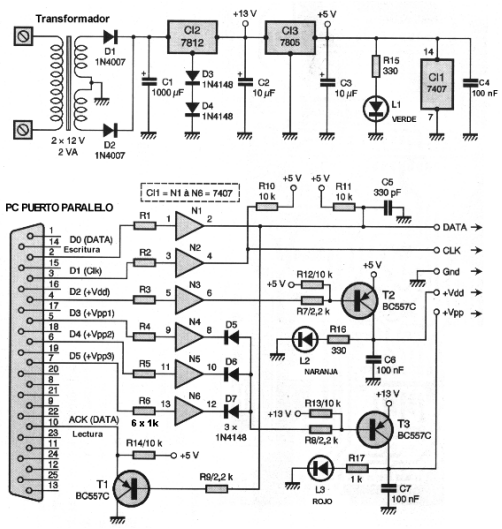
Use "*Debug*>Ejecutar>*Reset*" para iniciar el sistema. El contador del programa se reseteará a cero, que es el vector de reset en el 16F84. La línea del código fuente en esta dirección será destacada con una barra oscura. También advertirá que en la barra de estado, la PC se establecerá en 0x00.

3.8 Cargar el programa al PIC a través del quemador

Esta parte no es mas que cargar el programa al PIC para que este pueda entender los ingresos de información que puede realizar y que acción tomar para generar un proceso.

Al haber compilado el programa en *MPLAB* y generado el código en hexadecimal nos queda por ultimo cargar el programa al PIC. Esto se hace a través del software de microchip llamado *IC-Prog* de microchip y del quemador del PIC para el cual el diagrama se muestra en la figura 14.

Figura 14. Diagrama del quemador



La construcción del quemador puede realizarse en un *protoboard* debido a que solo se utilizara una vez o cuando se deseen hacer cambios al programa para que realice otras. El programador puede adquirirse o comprarlo debido a que ya existen algunos hechos.

Como podemos ver el PIC se coloca en el para cargar el programa este quemador va conectado al puerto paralelo de la computadora donde se encuentra el código hexadecimal. Los pasos para poder cargar el programa al PIC son los siguientes.

Pasos para cargar el programa

- Abrimos el software *IC-Prog* 1.05 de microchip
- Si es la primera ver que lo utilizamos elegimos la opción *ProPic* 2 *programmer*. Y colocamos un check a *invert VCC*.
- Abrimos el programa compilado en *MPLAB* con extensión HEX.
- Elegimos el tipo de modelo de PIC a quemar.

- Si nuestro PIC usara un reloj de cuarzo de 10 MHZ debemos de elegir la opción RC en oscilador. Y en el código CP off
- Quitamos el check a todas las opciones.
- Borramos el contenido del PIC
- Chequeamos que este bien conectado.
- Presionamos el botón de programar todo y esperamos.
- Si el mensaje es positivo entones el traspaso del programa fue correcto.
- Cerramos el programa.

Al concluir el proceso de carga del programa al PIC desde el software *IC-Prog* nos indicara que todo fue un éxito entonces procedemos a quitar el l PIC del quemador y a construir el circuito de la aplicación de control de temperatura en ambientes cerrados.

3.9 Construcción del hardware "circuito"

Para la construcción del circuito se necesita una los siguientes materiales.

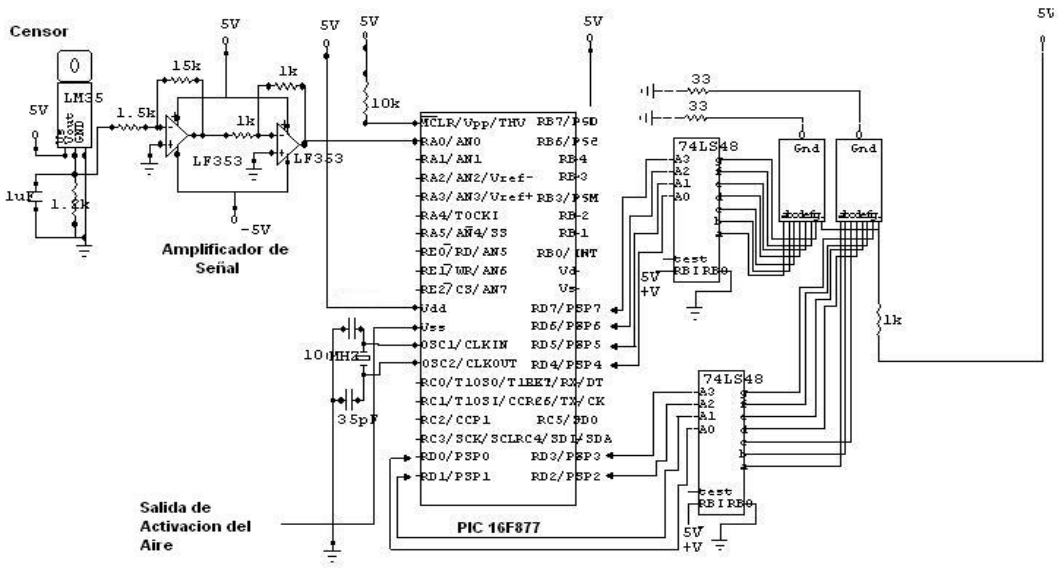
- Una placa de cobre.
- Cuatro metros de estaño.
- Un PIC 16f877.
- Un sensor lm35d.
- Dos *display* para el despliegue de temperatura.
- Dos convertidores de análogo a digital 77LS48.
- Un optó acoplador 3347.
- Dos *leds*.

- Un quemador de PIC.
- Una fuente de poder para alimentar el circuito

A continuación definiremos el diseños que deberá de tener el circuito para la construcción de nuestra aplicación de forma que el sensor pueda detectar la temperatura ambiente del lugar donde se encuentra nuestro equipo a proteger.

3.9.1 Diseño del circuito

Figura 15. Diagrama del circuito



Explicación del Diagrama de la figura 15: El PIC recibe voltaje de 5V, en la pata 11 y voltaje tierra en la pata 14. Las salidas para los *display* son las siguientes.

Puerto D, nos permite enviar la salida al *display* de las unidades. La salida de este número se da en binario.

Pata 19, salida A
Pata 20, salida B
Pata 21, salida C
Pata 22, salida D

Puerto E, nos permite enviar la salida al *display* de las decenas. La salida al igual que el puerto anterior es binaria.

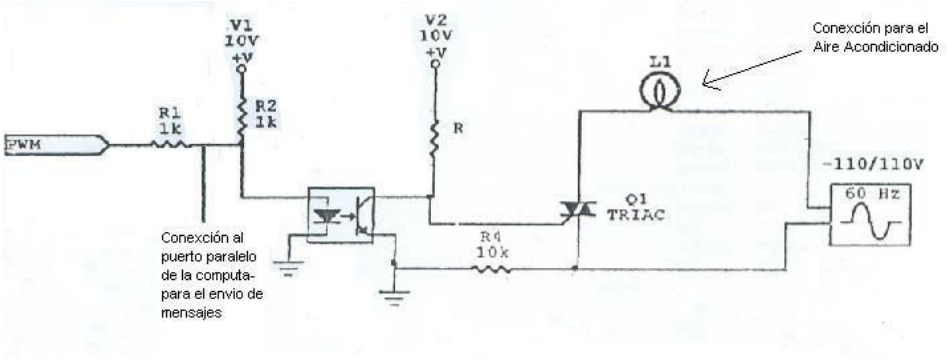
Pata 27, salida A
Pata 28, salida B
Pata 29, salida C
Pata 30, salida D

En el diagrama anterior se muestra la forma de cómo conectar el sensor lm35d al PIC. El sensor esta conectado en la pata 1 a 5voltios y en la pata 2 a tierra, la pata del centro es la que nos indica la salida del sensor en Micro voltios esta señal ingresa al PIC por medio de la pata 2.

Activar el aire acondicionado:

El circuito para activar el aire acondicionado es el siguiente.

Figura 16. Diagrama para activar el aire y enviar mensaje



Como podemos ver en la figura 16 la salida del PIC seda en el *PWD* pata 12 de integrado. Al momento de dar una señal esta salida, se cierra el releí y la corriente circula lo cual permite activar el aire, encender una lámpara, o una alarma. La otra salida del *PWD* no permite que la aplicación de recepción de señal envié correos electrónicos a las personas encargadas del centro de computo.

3.9.2 Construcción del circuito

Primero se deberá de dibujar el circuito de la forma en la que se diseñara para la placa de cobre. De manera que se pueda tener un orden bien definido.

Después de tener el diseño del circuito pasamos a dibujarlo en la placa ya sea con un marcador para subrayar CDS o con las calcomanías especiales para placas.

A continuación llenamos un trasto de plástico duro con el ácido para deshacer las el cobre en la placa, y realizamos movimiento de manera horizontal para ir ayudando a remover el cobre de la placa a modo que solo queden las líneas marcadas del diseño. Si en dado caso el cobre esta ya muy sucio y no se logra remover podemos vaciar el acido y volver a llenar el trasto con acido nuevo esto acelerara el proceso de botado del cobre.

Limpieza de placa: Al finalizar este proceso debemos de limpiar la placa, para mejor funcionamiento lo podemos hacer con un trapo con tiner así lograremos remover el marcador o la calcomanía.

Probar continuidad: Después de haber limpiado la placa debemos de probar la continuidad en las líneas de cobre para estar seguros que esta no fallara.

Perforación de hoyos: Luego debemos de perforar los hoyos donde estarán colocados nuestro integrados, esto se hace con una broca de 1ml de ancho, se deberá de tener cuidad que no se elimine el cobre en la parte trasera de la placa.

Introducir integrados y soldar: Al terminar de abrir los hoyos en la placa lo que nos queda es introducir los integrados y soldar con estaño las patitas por el lado de atrás de la placa.

3.10 Pruebas

Esta sección tiene como objetivo el poder explicar a fondo como funciona el circuito y cuales deberían de ser los procesos que nuestra aplicación realizara en un determinado momento según el cambio de temperatura que existiese.

3.10.1 Series de pruebas del circuito

Debemos de recordar que nuestro circuito de automatización es aplicado únicamente a ambientes cerrados donde el cambio de temperatura puede registrarse sin ningún problema y que no sea una temperatura completamente variable.

Primero podemos probar lo cambios de temperaturas en nuestro circuito colocando un dispositivo que genere calor, el cual puede ser una lámpara infrarroja, un cautín o una pistola de soldar. Esto nos ayudara a poder ver como los *display* van cambiando según la temperatura registrada.

El dispositivo se deberá de colocar a la distancia mas cercana al sensor debido a que el calor puede variar y no llegar a activar la opción elegida por quien lo construyo.

3.10.2 Activar dispositivo

Los dispositivos que esta aplicación puede llegar a activar son los siguientes. El aire acondicionado, una alarma, una luz de emergencia, etc.

Debido a que nuestra aplicación se basa en registrar la temperatura de un centro de cómputo manteniéndola a lo máximo 28 grados Celsius sin dejar que suba a más ya que esto puede hacer que los computadores trabajen a un bajo rendimiento o se apaguen por seguridad, activaremos lo que es el aire acondicionado si este estuviera apagado. Es importante mantener el nivel de la temperatura estable y menos de 25 grados Celsius para garantizar el perfecto rendimiento de las computadoras.

La activación del aire se dará en lugar de encender una luz tal como lo demuestra el circuito de la figura 3.7. Solo deberá de cortarse el alambre de corriente del aire colocando un lado del alambre en la pata 4 y otro en la pata 6 del opto acoplador el permitirá para la corriente al momento de recibir la señal del PIC. En la pata 2.

3.10.3 Envío de Mensajes

Para poder enviar mensajes se puede sacar la salida de activación del aire del PIC y leerla en el puerto paralelo de una computadora que tendrá un programa realizado en *Visual Basic* y este a su vez una base de datos con las personas que deberá de contactar al momento de registrar este cambio de temperatura para alertarlas. El envío de mensajes se puede realizar por medio del puerto del *smtp*.

El programa de *Visual Basic* estará esperando la lectura del pulso de activación del PIC y así poder enviar el mensaje. El código de esta aplicación es el siguiente.

```
// Procedimiento para enviar comandos SMTP al Puerto del 25 del Host al
// Al que se conectara
Private Enum SMTP_State
   MAIL_CONNECT
   MAIL_HELO
   MAIL_FROM
   MAIL_RCPTTO
   MAIL_DATA
   MAIL_DOT
   MAIL_QUIT
End Enum

//Variable para determinar el estado del SMTP
Private m_State As SMTP_State
//Procedimiento para enviar el mensaje
Private Sub cmdSend_Click()
   Winsock1.Connect Trim$(txtHost), 25
   m_State = MAIL_CONNECT
End Sub

// Procedimiento para establecer un hilo de conexión con el host al que se le
// enviara el mensaje
Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
   Dim strServerResponse   As String
   Dim strResponseCode     As String
   Dim strDataToSend       As String

   //Recibiendo datos del hilo establecido
     Winsock1.GetData strServerResponse
   Debug.Print strServerResponse
   strResponseCode = Left(strServerResponse, 3)

  // Definir solo los comandos en códigos a utilizar
   If strResponseCode = "250" Or _
     strResponseCode = "220" Or _
     strResponseCode = "354" Then
      Select Case m_State
         Case MAIL_CONNECT
```

```
    // Cambiando el estado de la conexión
    m_State = MAIL_HELO
        //Removiendo espacios en blanco
    strDataToSend = Trim$(txtSender)
 //Recibir el correo electrónico
    strDataToSend = Left$(strDataToSend, _
                InStr(1, strDataToSend, "@") - 1)
    // Enviar mensaje al servidor
    Winsock1.SendData "HELO " & strDataToSend & vbCrLf
        Debug.Print "HELO " & strDataToSend

     Case MAIL_HELO
    //Cambiar el estado de la sesión
    m_State = MAIL_FROM
Winsock1.SendData "MAIL FROM:" & Trim$(txtSender) & vbCrLf
    Debug.Print "MAIL FROM:" & Trim$(txtSender)
 Case MAIL_FROM
    //cambiar el estado de la sesión
    m_State = MAIL_RCPTTO
 Winsock1.SendData "RCPT TO:" & Trim$(txtRecipient) & vbCrLf
    Debug.Print "RCPT TO:" & Trim$(txtRecipient)
Case MAIL_RCPTTO
    //cambiar el estado de la sesión
    m_State = MAIL_DATA
    Winsock1.SendData "DATA" & vbCrLf
    Debug.Print "DATA"
Case MAIL_DATA
    //cambiar el estado de la sesión
    m_State = MAIL_DOT
    // Enviar el cuerpo del mensaje completo
    Winsock1.SendData "Subject:" & txtSubject & vbLf
    Debug.Print "Subject:" & txtSubject
    Dim varLines    As Variant
    Dim varLine     As Variant
    // Parsea el mensaje
    varLines = Split(txtMessage, vbCrLf)
    For Each varLine In varLines
        Winsock1.SendData CStr(varLine) & vbLf
        Debug.Print CStr(varLine)
    Next
    //Enviar símbolos de información al servidor
    Winsock1.SendData "." & vbCrLf
    Debug.Print "."
Case MAIL_DOT
```

```
                //cambiar el estado de la sesión
            m_State = MAIL_QUIT
            Debug.Print "QUIT"
        Case MAIL_QUIT
// Cerrar la conexión
            Winsock1.Close
    End Select
  Else
    // Si no responde el servidor
    Winsock1.Close
    If Not m_State = MAIL_QUIT Then
     MsgBox "SMTP Error: " & strServerResponse, _
            vbInformation, "SMTP Error"
    Else
      MsgBox "Message sent successfuly.", vbInformation
    End If
  End If
End Sub

//Procedimiento para detectar si el hilo establecido con el servidor dio error
Private Sub Winsock1_Error(ByVal Number As Integer, Description As String, ByVal Scode As Long, ByVal Source
As String, ByVal HelpFile As String, ByVal HelpContext As Long, CancelDisplay As Boolean)
  MsgBox "Winsock Error number " & Number & vbCrLf & _
      Description, vbExclamation, "Winsock Error"
End Sub
```

3.11 Costo de desarrollo

El costo total de realizar un proyecto de automatización de la temperatura de un habiente cerrado es el siguiente.

Tabla III. Tabla de costos

Cant.	Descripción	P. Unitario	Total
2	Placas de cobre de 30cms cuadrados	22.50	45.00
4	Metros estaño	2.50	10.00
1	PIC16F877	160.00	160.00
1	Sensor lm35d	25.00	25.00
2	*Display* para el despliegue de temperatura	8.00	16.00
2	Convertidores de análogo a digital 7748	25.00	50.00
1	Un optó acoplador 3347	11.00	11.00
2	*Leds*	0.50	1.00
1	Quemador de PIC	55.00	55.00
1	Fuente de poder para alimentar el circuito	125.00	125.00
5	Metros De cables de 8 hilos categoría 5	3.50	17.50
1	Protoboard para realizar pruebas	125.00	125.00
	Total		**Q640.50**

3.11.1 Beneficio

El beneficio de realizar este gasto será bastante alto, debido a que la empresa no necesitara gastar o pagar a un empleado que este pendiente de que el aire funcione.

Si no mas bien avisar alguien o notificarle que esta habiendo un cambio de temperatura en el centro de computo y esta persona como gerente puede determinar en ese mismo momento porque esta sucediendo esto y así antes de que los servidores se detengan el problema puede llegar a solucionar se no crear perdidas a la empresa.

CONCLUSIONES

1. Debido a que la aplicación realizada permite estar controlando la temperatura del centro de computo, garantiza que en el momento de que el aire acondicionado se apague o el nivel de temperatura exceda mas de lo máximo permitido, no existiría problema alguno, pues el aire se encendería, automáticamente, y además, el correo que se envía al gerente o persona encargada garantiza, que vea qué está pasando antes de que la temperatura aumente mas y los servidores se apaguen.

2. Existen momentos en que las personas de vigilancia de los centros de computo por la noche apagan el aire debido a que hay lugares donde la temperatura exterior es demasiado alto y siente mucho frió y luego, se les olvida subirlo a la mañana siguiente, pero con esta aplicación ese problema estaría resuelto pues encendería automáticamente el aire acondicionado.

3. Hay muchas aplicaciones que se pueden llegar a desarrollar por medio del los microcontroladores muchas pueden llegar a ser demasiado útiles y a veces hasta inteligentes para el proceso de automatización y toma de decisiones. Es por eso que es importante definir una metodología de desarrollo y cuáles seran las entradas posibles que un proceso pueda llegar a tener.

4. Se debe de tomar en cuenta qué tipo de aplicación se realizará para poder determinar que tipo de microcontrolador es más útil, debido a que existen variedades que pueden ser más baratos para lo que realmente lo necesitamos.

5. Lamentablemente, el desarrollo de este tipo de aplicación requiere del conocimiento de programación de lenguaje de maquina lo que con lleva a que sea difícil el mantenimiento de dichos programas para aprovechar al máximo todo lo que el microcontrolador puede llegar a desarrollar.

6. Existen metodologías que pueden ser alternativas para ser tomadas en cuenta al momento de realizar una aplicación de este tipo. Así, de esta manera, la persona encargadas para emprender el proyecto pueden llegar a desarrollarlo sin ninguna dificultad ni problema, además, existe mucha ayuda y bibliografías para realizarlos.

7. Este tipo de aplicaciones con llevan a poder automatizar la empresa y poder estar al alcance de la tecnología, utilizando al máximo nuestros recursos y beneficiándose el crecimiento.

RECOMENDACIONES

1. Muchos de los problemas que existen cuando los sistemas se caen o no hay comunicación con ellos, se dan debido a que los servidores se sobre calientan al máximo y por ello llegan a apagarse por seguridad y, a veces, nadie se da cuenta y creen que el problema está en la red. Es por eso importante saber que estos servidores cuentan con una temperatura ambiente que les permita no dejar de funcionar nunca, para que no se lleguen a tener perdidas dentro de las empresas.

2. Se le recomienda que la persona a realizar este tipo de aplicación tenga un leve conocimiento de electrónica digital sin ser experto, debido a que se necesita entender como trabaja cada uno de los dispositivos y que tipos de voltaje hay que introducirles para su funcionamiento.

3. Por otro lado, la persona debe entender como se maneja el código de maquina o lenguaje *asembler* debido a que la mayoría de los microcontroladores se programan de esta manera con la serie de instrucciones dadas.

4. El desarrollo de esta aplicación está orientado sólo a cuartos cerrados donde la temperatura pueda ser fácil de percibir debido a que en lugares donde varia mucho o que estén bien ventilados la temperatura puede llegar a variar demasiado rápido y no puede ser registrada, se recomienda que el circuito desarrollado o el sensor este lo mas cerca posible de los servidores o computadores a proteger para que pueda medir la temperatura ambiente que poseen dichas maquinas.

5. Para la carga del programa al microcontrolador se necesita de un buen quemador el cual puede ser creado o puede comprarse alguno, debido a que existe muchas empresas que ya los venden como tarjetas de computaras y solo es de conectarlas en alguno de sus buses periféricos, para facilitar las cargas o modificaciones de los programas.

6. Actualmente, el área de la computación no solo implica programar computadores y realizar sistemas de gestión, sino que es importante que se puedan realizar procesos o tareas que automaticen las industrias, realizando procesos capaces de tomar las decisiones por si solos, garantizando un perfecto funcionamiento en cada uno de ellos. Por lo cual, se recomienda realizar estudios para desarrollar dichas aplicaciones

BIBLIOGRAFÍA

1. ANGULO Usategui, José María y Ignacio Angulo Martínez. **Microcontroladores PIC: Diseño práctico de aplicaciones,** 1era edición. Ed. McGraw-Hill. 221 pp.

2. MARTIN Cuenca, Eugenio y José Maria Angulo Usategui. **Microcontroladores PIC: La clave del diseño,** 1era edición. Thomson Paraninfo, S.A. 2003.

3. ANGULO Martínez Ignacio, José María Angulo Usategui, Susana Romero Yesa. **Microcontroladores PIC 16F87X,** McGraw-Hill. 2000.

4. PADILLA Gil, Antonio J., Fernando Remiro Domínguez, Luis Miguel Cuesta García. **Electrónica Digital y Micro programable,** McGraw-Hill

5. DE Grediaga Olivo, Angel y Pablo Baeza Nadal. **Diseño Electrónico Con Microcontroladores,** AUTOR-EDITOR.

6. MANZO Quintas, Manuel. **Circuitos Electrónicos Digitales,** Editorial Universidad de Alcalá de Henares. 1995

7. NACIONAL Semiconductor, **Manual del Sensor LM35D Precision Temperatura Sensors,** (Edición mayo 1999)

8. Pagina: www.ing.unp.edu.ar/electronica/graduados/calafate/capitulo5.html, Titulo**: Departamento de Electrónica - Facultad de Ingeniería - U.N.P.S.J.B.,** Fecha Consulta: 05/03/2004

9. Pagina: http://www.esi2.us.es/~aguirre/LabMic_esp.html, Titulo: **LabMic1,** Fecha Consulta: 11/03/2004

10. Pagina:http://www.todorobot.com.ar/proyectos/picprog/picprog.htm, Titulo: **Programador PIC 16F8xx,** Fecha Consulta: 08/04/2004

11. Pagina:http://www.monografias.com/trabajos12/microco/microco.shtml, Titulo: **Microcontroladores,** Fecha Consulta: 16/04/2004

12. Pagina: http://bulma.net/body.phtml?nIdNoticia=1825, Titulo: **Control de la temperatura del procesador con ACPI,** Fecha Consulta: 17/04/2004

13. Pagina: http://manuales.elo.utfsm.cl/datasheet/national/htm/nsc02934.htm, Titulo: **National P/N LM35D -Precision Centigrade Temperature Sensor,** Fecha Consulta: 17/04/2004

14. Pagina:http://www.hispamp3.com/noticias/noticia.php?noticia=2004040111 1348, Titulo: **Sensores, Una nueva revolución tecnológica,** Fecha Consulta: 17/04/2004

15. Pagina: http://www.domotica.net/Tipos_de_Sensores.htm, Titulo: **Tipos de Sensores - Mercado - ,** Fecha Consulta: 17/04/2004

APÉNDICE A

A.1 La familia del PIC16F877

El microcontrolador PIC16F877 de *Microchip* pertenece a una gran familia de microcontroladores de 8 bites (bus de datos) que tienen las siguientes características generales que los distinguen de otras familias:

- Arquitectura *Harvard*
- Tecnología *RISC*
- Tecnología *CMOS*

Estas características se conjugan para lograr un dispositivo altamente eficiente en el uso de la memoria de datos y programa y por lo tanto en la velocidad de ejecución. Microchip ha dividido sus microcontroladores en tres grandes subfamilias de acuerdo al número de bits de su bus de instrucciones:

Tabla IV. Subfamilias de acuerdo al número de bits

Subfamilia	Bits del bus de instrucciones	Nomenclatura
Base – Line	12	PIC12XXX y PIC14XXX
Mid – Range	14	PIC16XXX
High – End	16	PIC17XXX y PIC18XXX

A.1.1 Variantes principales

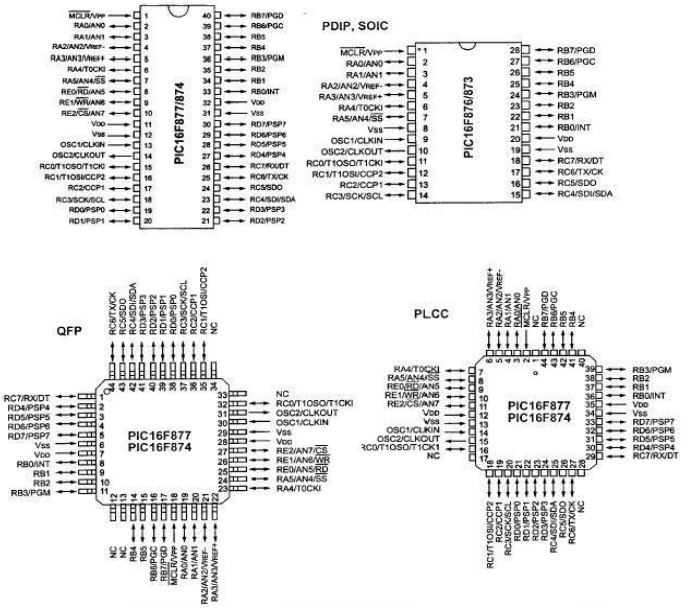
Los microcontroladores que produce *Microchip* cubren un amplio rango de dispositivos cuyas características pueden variar como sigue:

- Empaquetado (desde 8 patitas hasta 68 patitas)
- Tecnología de la memoria incluida (*EPROM, ROM, Flash*)
- Voltajes de operación (desde 2.5 v. Hasta 6v)
- Frecuencia de operación (Hasta 20 Mhz)

A.1.2 Empaquetados

Aunque cada empaquetado tiene variantes, especialmente en lo relativo a las dimensiones del espesor del paquete, en general se pueden encontrar paquetes tipo *PDIP (Plastic Dual In Line Package), PLCC (Plastic Leaded Chip Carrier) y QFP (Quad Flat Package),* los cuales se muestran en las figura 17.

Figura 17. Distintos empaquetados de los microcontroladores



A.1.3 Nomenclatura

Además de lo mostrado en la tabla anterior, en el nombre específico del microcontrolador pueden aparecer algunas siglas como se muestra en la siguiente tabla:

Tabla V. Nomenclatura de los microcontroladores

Tipo de Memoria	Rango de Voltaje	
	Estándar	Extendido
EPROM	PIC16**C**XXX	PIC16LCXXX
ROM	PIC16CRXXX	PIC16LCRXXX
FLASH	PIC16FXXX	PIC16LFXXX

En la siguiente tabla se especifican los rangos de voltaje estándar y extendido manejados por los dispositivos

Tabla VI. Rango de voltaje de los microcontroladores

Rango de Voltaje	*EPROM*		*ROM*		*FLASH*	
Estándar	C	4.5 a 6 V	CR	4.5 a 6 V	F	4.5 a 6 V
Extendido	LC	2.5 a 6 V	LCR	2.5 a 6 V	LF	2 a 6 V

A.1.5 Oscilador

Los PIC de rango medio permiten hasta 8 diferentes modos para el oscilador. El usuario puede seleccionar alguno de estos 8 modos programando 3 bits de configuración del dispositivo denominados: *FOSC2, FOSC1 y FOSC0.* En algunos de estos modos el usuario puede indicar que se genere o no una salida del oscilador (*CLKOUT*) a través de una patita de Entrada/Salida. Los modos de operación se muestran en la siguiente lista:

- LP Baja frecuencia (y bajo consumo de potencia)
- XT Cristal / Resonador cerámico externos, (Media frecuencia)
- HS Alta velocidad (y alta potencia) Cristal/resonador
- RC Resistencia / capacitor externos (mismo que *EXTRC* con *CLKOUT*)
- *EXTRC* Resistencia / capacitor externos
- *EXTRC* Resistencia / capacitor externos con *CLCKOUT*
- *INTRC* Resistencia / capacitor internos para 4 MHz
- *INTRC* Resistencia / capacitor internos para 4 MHz con *CLKOUT*

Los tres modos LP, XT y HS usan un cristal o resonador externo, la diferencia sin embargo es la ganancia de los *drivers* internos, lo cual se ve reflejado en el rango de frecuencia admitido y la potencia consumida. En la siguiente tabla se muestran los rangos de frecuencia así como los capacitores recomendados para un oscilador en base a cristal.

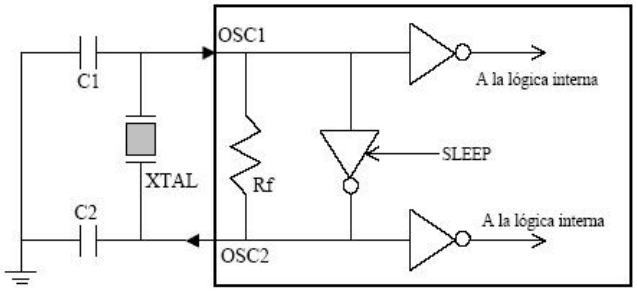
Tabla VII. Rangos de frecuencias recomendados para el oscilador

MODO	Frecuencia Típica	Capacitares Recomendados	
		C1	C2
LP	32 khz	68 a 100 pf	68 a 100 pf
	200 khz	15 a 30 pf	15 a 30 pf
XT	100 khz	68 a 150 pf	150 a 200 pf
	2 Mhz	15 a 30 pf	15 a 30 pf
	4 Mhz	15 a 30 pf	15 a 30 pf
HS	8 Mhz	15 a 30 pf	15 a 30 pf
	10 Mhz	15 a 30 pf	15 a 30 pf
	20 Mhz	15 a 30 pf	15 a 30 pf

A1.5.1 Cristal externo

En los tres modos mostrados en la tabla anterior se puede usar un cristal o resonador cerámico externo. En la figura 18 se muestra la conexión de un cristal a las patitas OSC1 y OS2 del PIC.

Figura 18. Conexión de un cristal



A.2 Características generales del PIC16F877

La siguiente es una lista de las características que comparte el PIC16F877 con los dispositivos más cercanos de su familia:

Tabla VIII. Familia del PIC

PIC16F873	PIC16F874	PIC16F876	PIC16F877

- CPU *RISC*
- Sólo 35 instrucciones que aprender
- Todas las instrucciones se ejecutan en un ciclo de reloj, excepto los saltos que requieren dos
- Frecuencia de operación de 0 a 20 MHz (DC a 200 nseg de ciclo de instrucción)
- Hasta 8k x 14 bits de memoria Flash de programa
- Hasta 368 bites de memoria de datos (*RAM*)
- Hasta 256 bites de memoria de datos *EEPROM*
- Hasta 4 fuentes de interrupción
- *Stack* de hardware de 8 niveles
- Reset de encendido (*POR*)

- *Timer* de encendido (*PWRT*)
- *Timer* de arranque del oscilador (*OST*)
- Sistema de vigilancia *Watchdog timer*.
- Protección programable de código
- Modo SEP de bajo consumo de energía
- Opciones de selección del oscilador
- Programación y depuración serie "*In-Circuit*" (*ICSP*) a través de dos patitas
- Lectura/escritura de la *CPU* a la memoria flash de programa
- Rango de voltaje de operación de 2.0 a 5.5 voltios
- Alta disipación de corriente de la fuente: 25mA
- Rangos de temperatura: Comercial, Industrial y Extendido
- Bajo consumo de potencia:
  - Menos de 0.6mA a 3V, 4 Mhz
  - 20 µA a 3V, 32 Khz
  - menos de 1µA corriente de *standby*.

A.2.1 Periféricos

- ***Timer0***: Contador/Temporizador de 8 bits con pre-escalador de 8 bits
- ***Timer1***: Contador/Temporizador de 16 bits con pre-escalador
- ***Timer0****:* Contador/Temporizador de 8 bits con pre-escalador y post-escalador de 8 bits y registro de periodo.
- **Dos módulos de Captura, Comparación y *PWM***
- **Convertidor Analógico/Digita**l: de 10 bits, hasta 8 canales
- Puerto Serie Síncrono (*SSP*)
- Puerto Serie Universal (*USART/SCI*).
- Puerto Paralelo Esclavo (*PSP*): de 8 bits con líneas de protocolo

A.3 Diagrama de Bloques del PIC16F877

En la figura 19 muestra el diagrama del PIC16F8977 en forma externa y en la figura 20 se ven los bloques y la organización interna del PIC16F877, esto no ayuda a tener una visión conjunta del interior y exterior del *Chip*.

Figura 19. Diagrama de patitas del PIC16F877

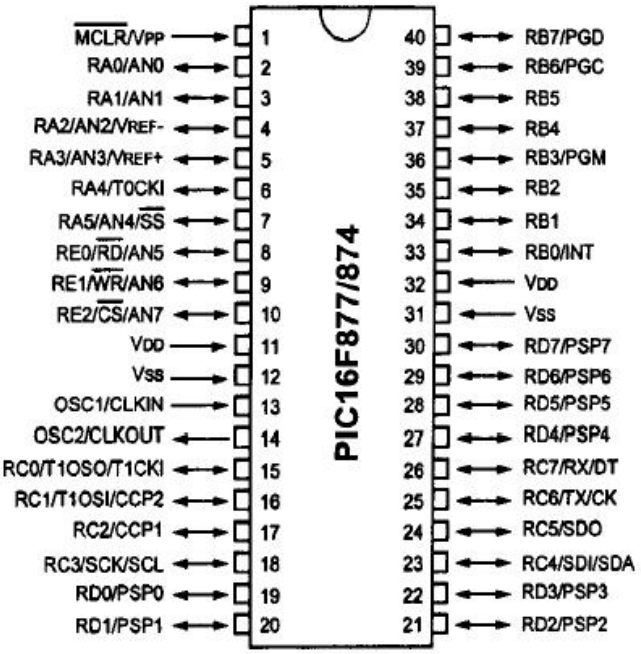
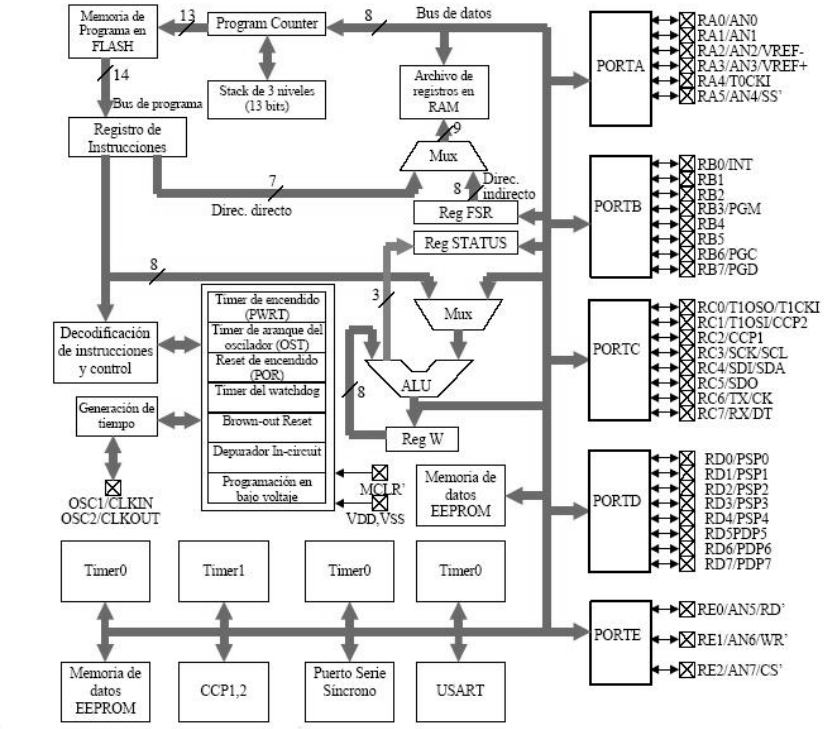


Figura 20. Diagrama de bloques del PIC16F877



A.4 Descripción del CPU del PIC16F877

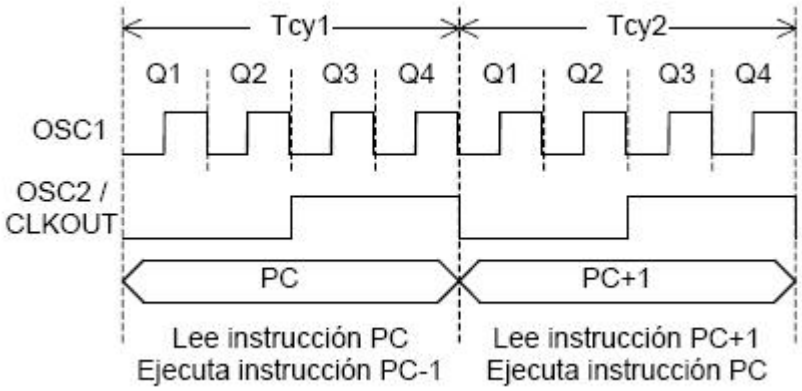
La *CPU* es la responsable de la interpretación y ejecución de la información (instrucciones) guardada en la memoria de programa.

Muchas de estas instrucciones operan sobre la memoria de datos. Para operar sobre la memoria de datos además, si se van a realizar operaciones lógicas o aritméticas, requieren usar la Unidad de Lógica y Aritmética (ALU). La ALU controla los bits de estado (Registro *STATUS*), los bits de este registro se alteran dependiendo del resultado de algunas instrucciones.

A.4.1 Ciclo de instrucción

El registro *Program Counter* (PC) es gobernado por el ciclo de instrucción como se muestra en la siguiente en la figura A.5. Cada ciclo de instrucción la *CPU* lee (ciclo *Fetch*) la instrucción guardada en la memoria de programa apuntada por PC y al mismo tiempo ejecuta la instrucción anterior, esto debido a una ***cola de instrucciones*** que le permite ejecutar una instrucción mientras lee la próxima:

Figura 21. Diagrama de los ciclos de instrucción



Como puede ver en la figura 21, cada ciclo de instrucción (*Tcy*) se compone a su vez de cuatro ciclos del oscilador (*Fosc*). Cada ciclo Q provee la sincronización para los siguientes eventos:

Q1: Decodificación de la instrucción
Q2: Lectura del dato (si lo hay)
Q3: Procesa el dato
Q4: Escribe el dato

Debido a esto cada ciclo de instrucción consume 4 ciclos de reloj, de manera que si la frecuencia de oscilación es Fosc, Tcy será 4/Fosc.

A.4.2 Registros de la CPU

Registro PC: Registro de 13 bits que siempre apunta a la siguiente instrucción a ejecutarse. En la siguiente sección se dan mayores detalles en el manejo de este registro.

Registro de Instrucción: Registro de 14 bits. Todas las instrucciones se colocan en el para ser decodificadas por la *CPU* antes de ejecutarlas.

Registro W: Registro de 8 bits que guarda resultados temporales de las operaciones realizadas por la ALU

Registro STATUS: Registro de 8 bits, cada uno de sus bits (denominados ***Banderas***) es un indicador de estado de la *CPU* o del resultado de la última operación.

A.5 Organización de la memoria del PIC16F877

Los PIC tienen dos tipos de memoria: memoria de datos y memoria de programa, cada bloque con su propio bus: bus de datos y bus de programa; por lo cual cada bloque puede ser accesado durante un mismo ciclo de oscilación.

La memoria de datos a su vez se divide en

- Memoria *RAM* de propósito general
- Archivo de Registros (*Special Function Registers (SFR*))

A.5.1 La Memoria de Programa

Los PIC de rango medio poseen un registro Contador del Programa (PC) de 13 bits, capaz de direccionar un espacio de 8K x 14, como todas las instrucciones son de 14 bits, esto significa un bloque de 8k instrucciones. El bloque total de 8K x 14 de memoria de programa está subdividido en 4 páginas de 2K x 14.

Observación1: No todos los PIC tienen implementado todo el espacio de 8K de memoria de programa.

Observación2: El fabricante puede grabar datos de calibración en localidades de memoria de programa por lo que se deberán anotar en papel antes de borrar los dispositivos con ventana transparente.

Vector de Reset.- Cuando ocurre un reset el contenido del PC es forzado a cero, ésta es la dirección donde la ejecución del programa continuará después del reset, por ello se le llama "**dirección del vector de reset**".

Vector de interrupción.- Cuando la *CPU* acepta una solicitud de interrupción ejecuta un salto a la dirección 0004h, por lo cual a esta se le conoce como "**dirección del vector de interrupción**". El registro *PCLATH* no es modificado en esta circunstancia, por lo cual habrá que tener cuidado al manipular el PC dentro de la Rutina de Atención a la Interrupción *(Interrupt Service Routine (ISR)).*

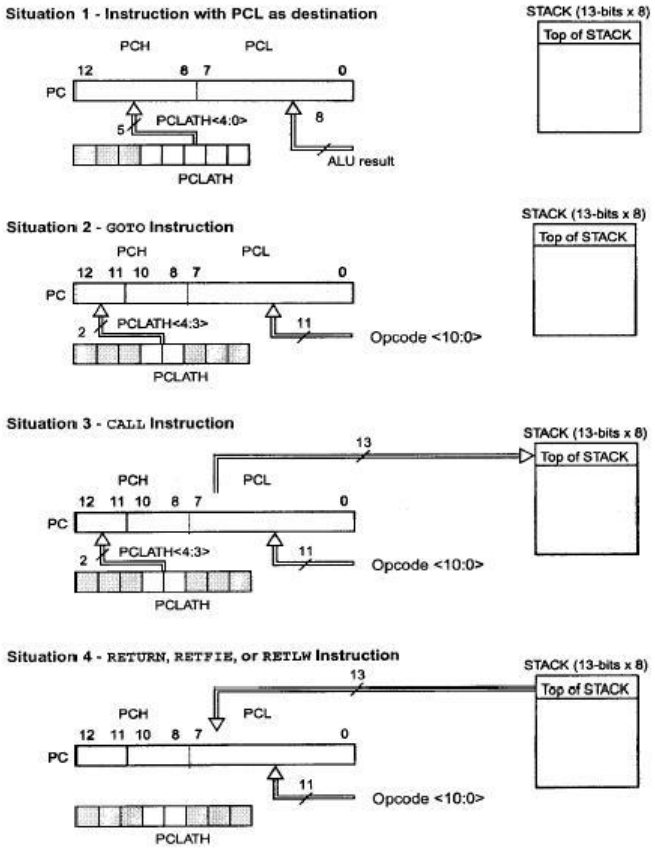
A.5.2 Manejo del Contador del Programa (PC)

El registro contador del programa (*PC*) especifica la dirección de la instrucción que la *CPU* buscará (*fetch*) para ejecutarla. El *PC* consta de 13 bits, separados en dos partes: como se muestra en la figura 22.

Figura 22. Diagrama de las partes del PC

El bite de orden bajo es llamado el registro PCL, mientras que el bite de orden alto es llamado registro PCH. Este último contiene los bits PC<12:8> y ***no se puede leer o escribir directamente*** Todas las actualizaciones al registro PCH deben ser hechas a través del registro *PCLATH*. En la figura 23 se ilustran las cuatro situaciones y las maneras correspondientes en que el PC puede ser actualizado.

Figura 23. Diagrama situaciones en que el PC puede ser actualizado



A.5.3 Paginación

Para saltar entre una página y otra, los bits más significativos del PC deberán ser modificados. Debido a que las instrucciones *GOTO* y *CALL* sólo pueden direccional un bloque de 2K (pues usan una dirección de 11 bits) deben existir otros dos bits que completen los 13 bits del PC para moverse sobre los 8K de memoria de programa.

Estos dos bits extra se encuentran en un *SFR* denominado *PCLATH* (*Program Counter Latch High*) en sus bits *PCLATH*<4:3>. Por esto antes de un *GOTO* o un *CALL* el usuario deberá asegurarse que estos bits apunten a la página deseada.

Si las instrucciones se ejecutan secuencial menté en la PC se cruza libremente los límites de página sin necesidad de que el usuario escriba en el *PCLATH*

A.5.4 Memoria de *Stack*

La memoria de *stack* es un área de memoria completamente separada de la memoria de datos y la memoria de programa. El *stack* consta de 8 niveles de 13 bits cada uno. Esta memoria es usada por la *CPU* para almacenar las direcciones de retorno de subrutinas. El apuntador de *stack* no es ni legible ni escribible. Cuando se ejecuta una instrucción *CALL* o es reconocida una interrupción el PC es guardado en el stack y el apuntador de *stack* es incrementado en 1 para apuntar a la siguiente posición vacía. A la inversa, cuando se ejecuta una instrucción *RETURN, RETLW* o *RETFIE* el contenido de la posición actual del *stack* es colocado en el PC.

La memoria de datos consta de dos áreas mezcladas y destinadas a funciones distintas:

- Registros de Propósito Especial (*SFR*)
- Registro de Propósito General (*GPR*)

Los *SFR* son localidades asociadas específicamente a los diferentes periféricos y funciones de configuración del PIC y tienen un nombre específico asociado con su función. Mientras que los *GPR* son memoria *RAM* de uso general.

A.5.5 Bancos de memoria

Toda la memoria de datos está organizada en 4 ***bancos*** numerados 0,1, 2 y 3. Para seleccionar un banco se debe hacer uso de los bits del registro STATUS<7:5> denominados IRP, RP1 y RP0. Hay dos maneras de acceder a la memoria de datos:

Direccionamiento directo e indirecto. La selección de bancos se basa en la siguiente tabla.

Tabla IX. Selección de bancos

Direccionamiento Indirecto (IRP)	RP1:RP0	Banco
0	00	0
	01	1
1	10	2
	11	3

Cada banco consta de 128 bites (de 00h a 7Fh). En las posiciones más bajas de cada banco se encuentran los SFR, y arriba de éstos se encuentran los GPR. Toda la memoria de datos está implementada en RAM estática.