ESCUELA DE TECNOLOGÍAS

PROTOTIPO DE UN SISTEMA DE CONTROL AUTOMÁTICO DE TEMPERATURA

MARGOTH VALENCIA LALANGUI

2012

ESCUELA DE TECNOLOGÍAS

PROTOTIPO DE UN SISTEMA DE CONTROL AUTOMÁTICO DE TEMPERATURA

"Trabajo de Titulación presentado en conformidad a los requisitos establecidos
para optar por el título de
Tecnóloga en Redes y Telecomunicaciones"

Profesor Guía
Ing. Marco Cevallos
……

Autor/a
Margoth Valencia Lalangui

……..

2012

DECLARACIÓN DEL PROFESOR GUÍA

“Declaro haber dirigido este trabajo a través de reuniones periódicas con la estudiante orientando sus conocimientos para un adecuado desarrollo del tema escogido, y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación.”

……..

Marco Cevallos

C.I.170513310-4

DECLARACIÓN DE AUTORÍA DEL ESTUDIANTE

“Declaro que este trabajo es original, de mi autoría, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes.”

……..
Margoth Valencia
1720999513

RESUMEN

Debido a que las condiciones ambientales, inciden en el desarrollo de trabajo de un ser humano, se diseñó el Prototipo de Control Automático deTemperatura; el mismo que tiene como función principal mantener una temperatura óptima para el ser humano y para la conservación de materia prima.

Para el desarrollo de este prototipo se empleó el circuito integrado LM35, el mismo que es un sensor de temperatura cuya tensión de salida es linealmente proporcional con la temperatura en la escala Celsius, posee una precisión aceptable para la aplicación requerida, no necesita calibración externa, y es de bajo costo.

Este transductor censa la temperatura existente en el ambiente y emite pulsos eléctricos dependiendo de la temperatura existente; los mismos que son almacenados y procesados en el micro controlador PIC16F877A, para luego emitir una orden lógica de encendido previamente programada de acuerdo a la necesidad que se presente.

En este prototipo se utilizara un ventilador y un calefactor para mantener la temperatura ambiente, el mismo que debido a su costo esta remplazado por un led (diodo emisor de luz).Si la temperatura supera las condiciones adecuadas del lugar donde se encuentra operando el prototipo encenderá automáticamente un ventilador, y si la temperatura baja se encenderá el calefactor que en nuestro caso está representado por el LED.

Estas dos opciones están previamente programadas en la memoria interna del PIC, el funcionamiento óptimo del prototipo depende directamente de la calibración del sensor de temperatura. La información de la temperatura existente podrá ser visualizada por el usuario a través de un LCD y por medio de pulsadores se podrá variar los parámetros de temperatura necesarios para el lugar donde está siendo empleado el Prototipo de Control Automático.

SUMMARY

Since environmental conditions affect the development work of a human being was designed Prototype Automatic Temperature Control; the same as its main function is to maintain an optimum temperature for humans and for the conservation of materials.

For the development of this prototype was used LM35 integrated circuit is the same as a temperature sensor whose output voltage is linearly proportional to the temperature on the Celsius scale. It has an acceptable precision for the required application does not need external calibration, and is inexpensive.

This transducer census the temperature in the atmosphere and emits electrical pulses depending on the existing temperature, which we will be stored and processed in the PIC16F877A microcontroller and then issue an order on pre-programmed logic according to the need to be present.

In this prototype were used a fan and a heater to keep the temperature the same because of its cost is replaced by an LED (light emitting diode).
If the temperature exceeds the appropriate conditions of where the prototype is operating automatically turn on a fan, and if the temperature drops will turn the heater in our case is represented by the LED.

These two options are pre-programmed PIC's internal memory,the optimal operation of the prototype directly dependent on the temperature sensor calibration.The existing temperature information can be viewed by the user through an LCD and by means of buttons may vary the temperature parameters needed for the site is being used the Prototype of Automatic Control.

ÍNDICE

INTRODUCCIÓN

Este proyecto se basa en la necesidad de mantener una ambiente adecuado en áreas de trabajo cerradas dependiendo de la temperatura ambiente para su funcionamiento. Debido a esta necesidad surgió la idea de realizar un Prototipo de Control Automático de Temperatura.

Se empleó el micro controlador PIC16F877A, que resultó ser el más apropiado para el propósito especificado.

Para la construcción del proyecto se emplea un sensor de temperatura LM35 como termómetro digital, este dispositivo presenta una variación de 10mV por grado centígrado. Su alimentación puede ser de 4 a 30 Voltios, el rango de temperatura a sensar va desde -55 ºC a 150 ºC.

La temperatura mínima y máxima que aparece en el LCD son las que están grabadas previamente en el EEPROM del PIC, las mismas que se pueden variar con el uso de las teclas Enter, Subir y Bajar a conveniencia del usuario.

La temperatura actual es la que el PIC está sensando a cada instante con las variaciones de voltaje que proporciona el LM35.Si la temperatura baja del rango mínimo establecido será sensada y un led que representa a un calefactor se encenderá. Si la temperatura supera el rango máximo, en este caso se encenderá un ventilador y así se mantendrá la temperatura promedio logrando un ambiente con una temperatura atractiva para el usuario.

Los datos de la temperatura mínima, actual y máxima se visualizan por medio de un LCD (2 x 16) 2 líneas de texto de 16 caracteres".

Se puede modificar los rangos de temperatura por medio de los pulsadores para cada uno de los parámetros antes mencionados.

CAPITULO 1: MICROCONTROLADORES[1]

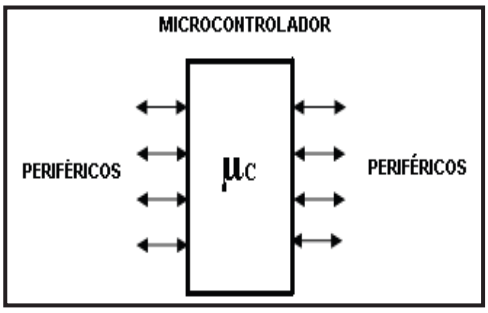
1.1. Conceptos:

Es un circuito integrado programable,que nos ofrece la posibilidad de un pequeño computador.

Sus líneas de entrada/salida soportan la conexión de los sensores y pulsadores del dispositivo a controlar y todos los recursos complementarios disponibles tienen como única finalidad atender sus requerimientos como se indica en la *Figura 1.1*

Una vez programado y configurado el microcontrolador realizará las tareas asignadas por el programador.

Figura 1.1 El Microcontrolador.



Fuente: José Angulo

1.2 Familia de los PIC

Microchip dispone de cuatro familias o gamas de microcontroladores, estas son:

---

[1]ANGULO, José Angulo, Ignacio. Microcontroladores PIC, Diseño Práctico de Aplicaciones, Editora McGraw-Hill, Págs. 1-4.

Tabla 1.1 Familia de los PICs

FAMILIA	PIC	NUM. DE BITS	CARACTERISTICAS
Gama Enana	PIC16F© XXX	12 ó 14 bits	Tamaño reducido, trabaja con una alimentación en DC desde 2.5 a 5.5 voltios.
Gama Baja	PIC 16C5X	12 bits	Recursos limitados, solo tienen manejo a nivel de bits.
Gama Media	PIC16F(C) XXX	14 bits	Gama variada y completa, tienen encapsulados de 18 a 68 pines, integran varios periféricos externos.
Gama Alta	PIC17CXXX	16 bits	Posee un sistema de gestión de interrupciones a gran velocidad y es de arquitectura abierta.

Dentro de la gama media, la serie del PIC16F87X y PIC 16F87XA, en general reúnen las mejores características de todas las gamas de familias, el que más recursos posee y mejor se adapta a las necesidades de este proyecto es el PIC16F877A.

1.3 Microcontrolador 16F877A[2]

La elección de este Microcontrolador se basó en las ventajas que nos ofrece estas son:

- Más resistencia a condiciones físicas externas.
- Recursos incorporados como conversores A/D, salida PWM, memoria de datos volátiles, etc.
- Protección ante fallos de alimentación.
- Código de protección programable.
- Programación de la memoria de programa más sencilla.
- Costos vs. Prestaciones más accesibles.

Entre más de cincuenta fabricantes de microcontroladores que se encuentran en el mundo, es muy difícil seleccionar el mejor". La selección se realizo según sus características específicas las que determinan el más conveniente.

Sin embargo, los factores siguientes son determinantes en su elección: sencillez de manejo, buena información, precio, buen promedio de parámetros (velocidad, consumo, tamaño, alimentación, código compacto), entre otros.

Para el caso de este proyecto se escogió uno de los microcontroladores del fabricante Microchip, basándose en la utilización más común en este tipo de proyectos, y por ser el más idóneo para la realización del mismo.

---

[2]ANGULO, José Angulo, Ignacio. Microcontroladores PIC, Diseño Práctico de Aplicaciones, Tomo II, Editora McGraw-Hill, Pág.4-5

Características[3]

Juego de 35 instrucciones de 14 bits de longitud, todas ellas se ejecutan en un ciclo de instrucción, menos las instrucciones de salto.
Frecuencia de trabajo hasta 20 MHz
Hasta 8k palabras de 14 bits para la memoria de código tipo flash.
Hasta 368 bytes de memoria de datos RAM.
Hasta 256 bytes de memoria de datos no volátil EEPROM.
Hasta 14 fuentes de interrupción interna y externa.
Pila con 8 niveles.
Modo de direccionamiento directo, indirecto y relativo.
Perro guardián (WDT).
Código de protección programable.
Modo SLEEP de bajo consumo.
Voltaje de alimentación comprendido entre 2V y 5.5V.
Bajo consumo (menos de 2mA a 5V y 4Mhz).

Como dispositivos Periféricos

Timer 0: temporizador- contador de 8bits con pre divisor.
Timer 1: temporizador-contador de 16 bits, con pre divisor.
Timer 2: temporizador de 8bits, con pre divisor y postdivisor.
Dos módulos de PWM, captura y comparación.
Conversor A/D de 10 bits.
Puerto serie sincrónico (SSP) con SPI e I2C.
Puerto serie asincrónico (USART).
Puerto Paralelo Esclavo (PSP).
Una interrupción externa.

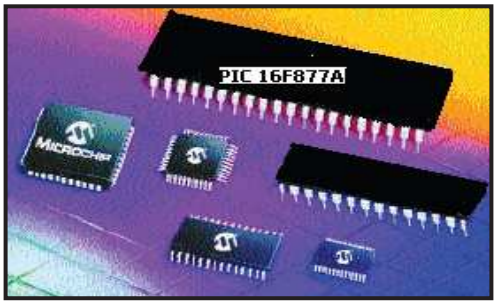
---

[3]ANGULO, José Angulo, Ignacio. Microcontroladores PIC, Diseño Práctico de Aplicaciones, Tomo II, Editora McGraw-Hill, Pág.22

Aspecto Externo

El PIC16F877A está fabricado con una tecnología CMOS, que consume muy poca corriente pero a la vez es susceptible a daños por estática por lo que es muy recomendable utilizar pinzas para manipular o una manilla antiestática y así poder transportar desde el grabador al protoboard o viceversa), su encapsulado es de tipo DIP de 40 pines, cuatro de ellos soportan la tensión de alimentación (2VDD,2VSS), otros dos reciben la señal del oscilador externo (cristal de cuarzo) y otro se utiliza para generar un reset (MCRL) o entrada de voltaje de programación verificación en el PIN 1 y los 33 pines restantes funcionan como líneas de E/S.
En la *Figura 1.2,* se presenta el aspecto externo del microcontrolador 16F877A.

Figura 1.2 Microcontrolador 16F877A.



Fuente: http//www.microchip.com

Organización de la memoria

El PIC 16F877A tienen dos tipos de memoria; Memoria de Programa y Memoria de Datos. Cada bloque con su propio bus: Bus de Datos y Bus de Programa.

Memoria de programa: conocida también como memoria de instrucciones, aquí se escribe las órdenes para que el CPU las ejecute. El PIC16F877A tiene una memoria de programa no volátil tipo FLASH, puede tener una capacidad de 8k palabras de 14 bits cada una, dicha memoria está dividida en dos

páginas de 4k cada una y esta direccionada con el PC, que tiene un tamaño de 13 bits.

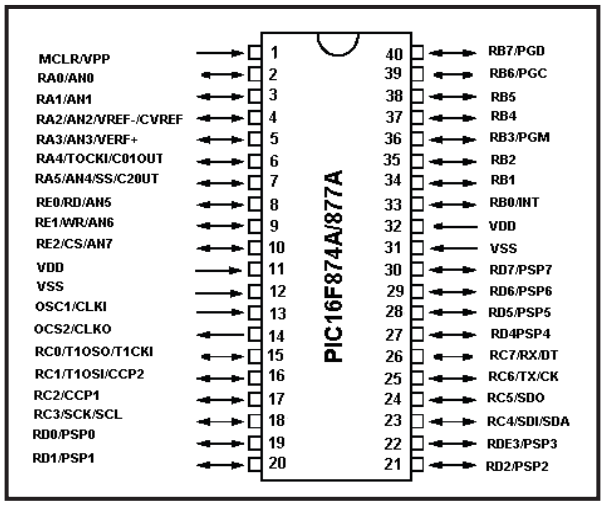
Memoria de Datos (RAM): Se alojan los registros operativos fundamentales en el funcionamiento del procesador y en el manejo de todos sus periféricos, además de registros que el programador puede usar para información de trabajo propio de la aplicación. La memoria de datos está divida en 4 bancos con 128 bytes cada uno.

En las posiciones iníciales de cada banco se encuentran los siguientes registros específicos que gobiernan al procesador y sus recursos. El total de la memoria es de 368x8 bytes.

Puertos de Entrada/Salida

El PIC 16F877A, tiene 33 líneas para comunicarse con el medio externo, todos estos puertos son multifuncionales, es decir, realizan diversas funciones según estén programadas. Sin embargo todas ellas tienen la capacidad de trabajar como líneas de E/ S digitales.

Figura 1.3 Distribución de pines del 16F877A.



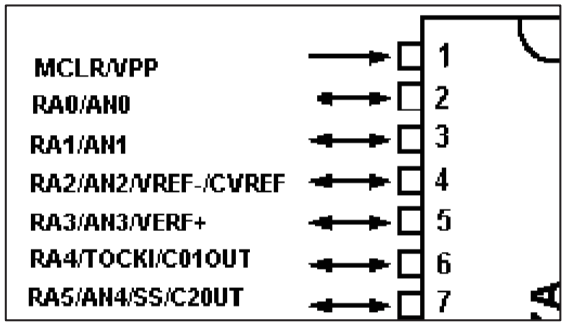
Fuente: http//datashet_downloads/16f877a/devicedoc

- **Puerto A**

Sólo dispone de 6 líneas denominadas RA0 – RA5, son bidireccionales y su sentido queda configurado según la programación realizada.

RA0/AN0, RA1/AN1Y y RA2/AN2/Vref-/CVref: además de líneas de E/S digitales también pueden actuar como los canales 0, 1 y 2 por lo que se puede aplicar una señal analógica al conversor A/D.
RA3/AN3/Vref +: también puede actuar como entrada de la tensión de referencia para los periféricos que la precisan.
RA4/TOCK1/C1OUT: actúa como E/S digital y como entrada de señal de reloj para el timer 0.
RA5/AN4/SS/C2OUT: tiene multiplexada tres funciones, E/S digital, canal 4 para el conversor A/D y selección del modo esclavo se trabaja con la comunicación serie sincrónica.

Figura 1.4 Distribución de pines del puerto A.



Fuente: http//datashet_downloads/16f877a/devicedoc

- **Puerto B**

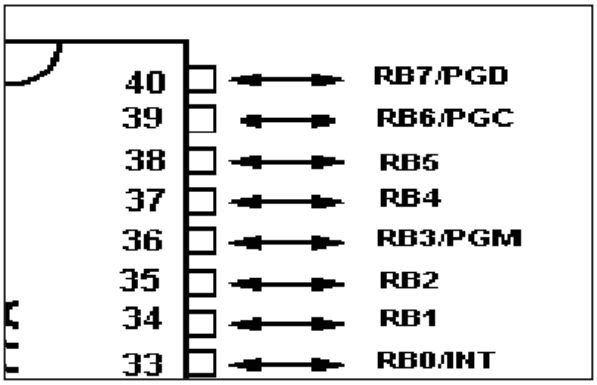
Dispone de 8 líneas bidireccionales cuya función se elige mediante la programación.

RBO/INT, RB1, RB2, RB3/PGM, RB4, RB5: son líneas de E/S digitales del pórtico B RB0/INT también sirve como entrada a una petición de interrupción externa.

RB6/PGC: es una línea de E/S digitales, también se introducen los pulsos de reloj, cuando se introduce la programación sincrónicamente en serie.

RB7/PGD: es una línea de E/S digitales, también se introducen los bits de datos

Figura 1.5 Distribución de pines del puerto B.



Fuente: http//datashet_downloads/16f877a/devicedoc

- **Puerto C**

Consta de 8 líneas bidireccionales cuyo sentido se configura mediante su programa, todas los pines tienen multiplexadas diferentes funciones.

RC0/T1OSO/T1CKI: esta línea puede actuar como E/S digital, como salida del timer 1 ó como entrada de impulsos para el timer 1.

RC1/T1OSI/CCP2: esta línea puede actuar como E/S digital, entrada al oscilador del timer 1, entrada del módulo de captura 2 , salida del comparador 2 , salida del PWM 2.

RC2/CCP1: esta línea puede actuar como E/S digital, entrada captura1, salida comparador 1, salida PWM1.

RC3/SCK/SCL: esta línea puede actuar como E/S digital, señal de reloj en modo SPI, señal de reloj en modo I2C.

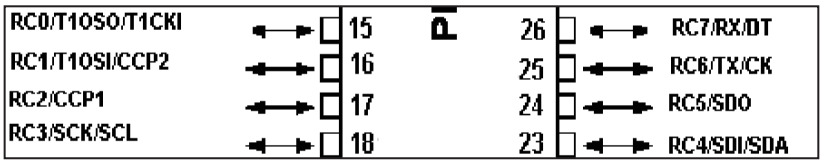
RC4/SDI/SDA: esta línea puede actuar como E/S digital, entrada de datos en modo SPI, línea de datos en modo I2C.

RC5/SDO: esta línea puede actuar como E/S digital, salida de datos en modo SPI

RC6/TX/CK: esta línea puede actuar como E/S digital, línea de transmisión en USART, señal de reloj sincrónica en transmisión serie.

RC7/RX/DT: esta línea puede actuar como E/S digital, línea de recepción del USART, línea de datos en transmisión serie sincrónica.

Figura 1.6 Distribución de pines del puerto C.



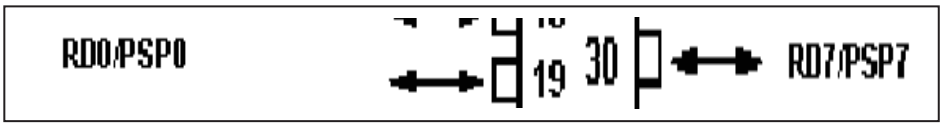
Fuente: http//datashet_downloads/16f877a/devicedoc

- **Puerto D**

Consta de 8 líneas bidireccionales, todos los pines disponen en su entrada de Un SmittchTrigger.

RD0/PSP–RD7/PSP7: además de usarse como líneas de E/S digitales normales, implementan un puerto paralelo esclavo de 8 líneas (PSP), que permiten la comunicación en paralelo con otros elementos del sistema.

Figura 1.7 Distribución de pines del puerto D.



Fuente: http//datashet_downloads/16f877a/devicedoc

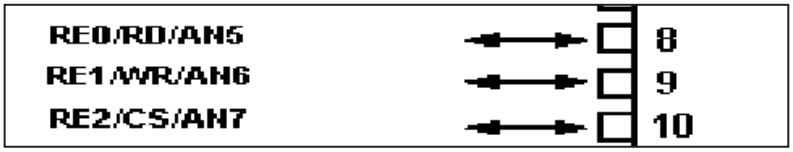
- **Puerto E**

RE0/RD/AN5: además de usarse como líneas de E/S digitales, señal de lectura en el modo de puerta paralela esclava, canal 5 del conversor A/D.

RE1/WR/AN6: además de usarse como líneas de E/S digitales, señal de escritura en modo PSP, canal 6 del conversor A/D.

RE2/CS/AN7 además de usarse como líneas de E/S digitales, selección de chip en el modo PSP, canal 7 del conversor A/D.

Figura 1.8 Distribución de pines del puerto E.

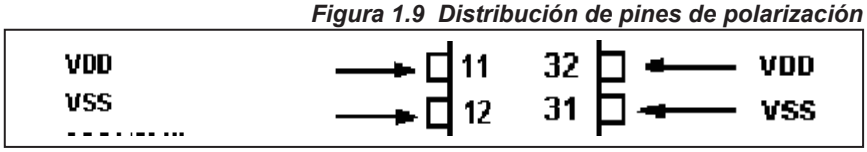


Fuente: http//datashet_downloads/16f877a/devicedoc

Pines de polarización

V_{DD}: tensión de alimentación positiva.
V_{SS}: tierra o negativo de la alimentación.

Figura 1.9 Distribución de pines de polarización



Fuente: http//datashet_downloads/16f877a/devicedoc

El reloj

Este es el parámetro fundamental a la hora de establecer la velocidad de ejecución de las instrucciones y el consumo de energía.
El microcontrolador 16F877A posee un oscilador interno RC de 4Mhz, permite también utilizar un oscilador externo de hasta 20 MHz

Re inicialización o reset

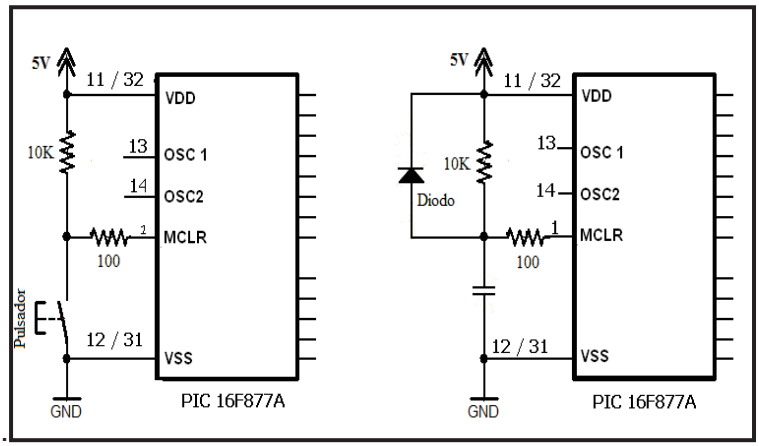
El PIC 16F877A dispone de diversas maneras de reinicializarse que se citan a continuación:

- Reset por conexión de la alimentación (POR: Power – onReset). El valor de la tensión de alimentación VDD sube entre 1,2 v a 1,7 v.
- Activación del pin MCLR (nivel bajo en dicho pin durante una operación Normal).
- Activación del pin MCLR estando en el PIC trabajando en modo SLEEP.

- Reset provocado por el desbordamiento del perro guardián en una operación normal.
- Reset provocado por el desbordamiento del perro guardián durante el estado de reposo.
- Reset provocado por una caída de voltaje (BORN: Brown – out- Reset) en VDD baja entre 3,8 V Y 4,2V.

Existen dos circuitos de conexión usados para el reset o re inicialización como se indica en la *Figura 1.4*

Figura 1.10 Circuitos usados para el Reset.



Fuente: http//www.microchip.com_reset/downloadtimer

Interrupciones

Las interrupciones son desviaciones asincrónicas del flujo de control del programa originadas por diversos sucesos que no están bajo el control de las instrucciones del programa. Estos sucesos pueden ser internos o externos al sistema y en diseños industriales son un recurso muy importante para atender acontecimientos físicos en tiempo real. Cuando se produce una interrupción se detiene la ejecución del programa en curso, se salva el valor del PC en la pila y se carga aquel con el valor 0004h que es el vector de interrupción.

Causas de la interrupción

Existen catorce posibles causas de interrupción.

Activación del pin RB0/INT.
Desbordamiento del temporizador TMR0.
Desbordamiento del temporizador TMR1.
Desbordamiento del temporizador TMR2.
Captura o comparación en él módulo CCP1.
Captura o comparación en él módulo CCP2.
Transferencia en el puerto serie sincrónico.
Transferencia en el puerto paralelo esclavo.
Colisión de bus en el puerto serie sincrónica.
Cambio de estado de una de las entradas RB4 - RB7 del pórtico B.
Finalización de la escritura en la EEPROM de datos.
Fin de la transmisión en el USART.
Fin de la recepción en el USART.
Fin de la conversión en el conversor A/D.

1.4 PANTALLA DE CRISTAL LÍQUIDO (LCD) [4]

Introducción

La pantalla de cristal líquido es uno de los visualizadores más utilizados en la actualidad debido a las importantes ventajas que ofrece, por ejemplo permite mostrar mensajes que indican al operario el estado de la máquina, o para dar instrucciones de manejo, mostrar valores, entre otros.

---

[4]BOYLESTAD, Robert L, Nashelsky Louis, Electrónica: Teoría de Circuitos y Dispositivos Electrónicos. Pag 59, 60,61

Es decir que permite la comunicación entre máquinas y humanos, ya que los mensajes que se visualizan en la pantalla del LCD pueden mostrar cualquier código ASCII, introduciendo el código correspondiente de cada uno de los caracteres a visualizar.

En el mercado existen varias presentaciones por ejemplo de 2 líneas por 8 caracteres, 2x16, 2x20, 4x20.

A pesar de que la variedad de modelos de LCD es muy grande, las líneas necesarias para su conexión y control son prácticamente las mismas. Para la visualización de nuestro proyecto utilizaremos un LCD 2x16, como se muestra en la *Figura 1.11* que a continuación indicaremos sus aspectos más importantes.

Figura 1.11 LCD 2 X16.

Fuente: http://www.x-robotics.com/lcd.html

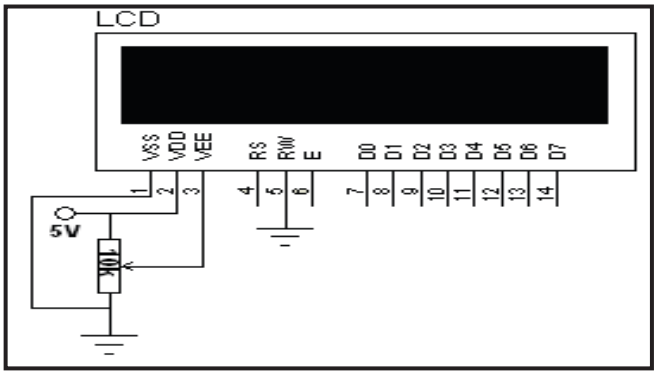
La pantalla de cristal líquido o LCD (Liquid Crystal Display) es un dispositivo microcontrolador de visualización gráfico para la presentación de caracteres, símbolos o incluso dibujos (en algunos modelos). En este caso dispone de 2 filas de 16 caracteres cada una y cada carácter dispone de una matriz de 5x7 puntos (pixeles).

Este dispositivo esta manejado internamente por un microcontrolador Hitachi 44780 y regula todos los parámetros de presentación, este modelo es el mas comúnmente usado.

Descripción de pines

En la *Figura 1.12* se puede visualizar la distribución del pines del LCD 2x16, y en la *Tabla 1.2* la función que cumple cada pin.

Figura 1.12 Distribución de pines del LCD.



Fuente: http://www.x-robotics.com/lcd.html

Tabla 1.2 Función de cada pin del LCD.

Pin	Símbolo	Descripción
1	Vss	Tierra de alimentación GND
2	Vdd	Alimentación de +5 Vcc
3	Vo	Ajuste del contraste del cristal líquido (0 a+5 V)
4	Rs	Selección del registro control/datos Rs=0 reg. Control RS= 1 reg. Datos
5	R/W	Lectura / Escritura en LCD R/W =0 escritura R/W =1 lectura
6	E	Habilitación
7	D0	Bit menos significativo (bus de datos bidireccional)
8	D1	Pin de comunicación entre un microcontrolador y el LCD.
9	D2	Pin de comunicación entre un microcontrolador y el LCD
10	D3	Pin de comunicación entre un microcontrolador y el LCD
11	D4	Pin de comunicación entre un microcontrolador y el LCD
12	D5	Pin de comunicación entre un microcontrolador y el LCD
13	D6	Pin de comunicación entre un microcontrolador y el LCD
14	D7	Bit más significativo (bus de datos bidireccional)

Funcionamiento

Para comunicarse con la pantalla LCD podemos hacerlo por medio de sus pines de entrada de dos maneras posibles, con bus de 4 bits o con bus de 8 bits, estos dos se diferencian en el tiempo de retardo, pues la comunicación a 4bits, primero envía los 4 bits más altos y luego los 4 bits más bajos, mientras que la de 8 bits envía todo al mismo tiempo, esto no es un inconveniente si consideramos que el LCD trabaja en microsegundos (µs). Para la aplicación del proyecto utilizaremos la conexión a 4 bits, debido a que se deben conectar pocos cables , sólo se debe conectar el bit de registro, el Enable y los 4 bits más altos de LCD, con esto es suficiente para enviar los mensajes. El compilador PBP soporta módulos LCD con controlador Hitachi 44780 o equivalentes, se conecta el LCD al PIC16F877A de la siguiente manera, en B3 para el registro R/S, en B2 el Enable, en el puerto B desde el B4 hasta el B7, los bits más altos del LCD, también se utilizó una resistencia de 10Ω conectado a la alimentación del backlight, sirve para evitar altas temperaturas, además el R/W del LCD se encuentra conectado a tierra , esto de debe por que la declaración de LCDOUT, es de escritura solamente.

Declaración de instrucciones para el LCD en PBP

La declaración LCDOUT sirve para mostrar ítems en una pantalla de cristal líquido, se utiliza escribiendo: LCDOUT seguido por $fe, esto se lo escribe siempre, junto a esto el comando a utilizar. En la siguiente *Tabla 1.3* muestra los comandos para la programación de nuestro PIC 16F877A en el compilador PBP.

Tabla 1.3 Comandos más utilizados para manejar el LCD.

Comando	Operación
$FE,1	Limpia el visor del LCD
$FE,2	Vuelve al inicio (comienzo de la primera línea)
$FE,$0C	Apagar el cursor

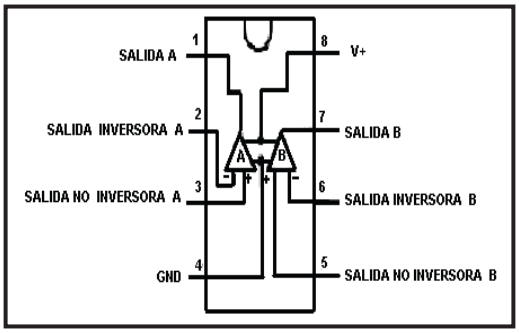
$FE,$0E	Subrayado del cursor activo
$FE,$0F	Parpadeo del cursor activo
$FE,$10	Mover el cursor una posición a la izquierda
$FE,$14	Mover el cursor una posición a la derecha
$FE,$80	Mover el cursor al comienzo de la primera línea
$FE,$C0	Mover el cursor al comienzo de la segunda línea
$FE,$94	Mover el cursor al comienzo de la tercera línea
$FE,$D4	Mover el cursor al comienzo de la cuarta línea

1.5 Circuito Integrado LM358 [5]

Es un circuito integrado que en su interior se encuentran dos amplificadores operacionales que trabaja en un rango de 3v a 32v a 500µA, una particularidad de este circuito integrado, es que puede ser alimentado con una sola fuente positiva.

En algunos proyectos se necesita un amplificador donde no se cuenta con fuentes +V -V y este integrado es una buena opción. Como se indica en la *Figura 1.13 se* muestra la distribución de pines.

Figura 1.13 Distribución de pines de LM358.



Fuente: http://www.sensores_lm35/view

[5] COSTALES, Alcívar Apuntes de microcontroladores.Pag 24, 25, 26, 27

1.6 SENSORES[6]

Un sensor, es un dispositivo diseñado para recibir información de una magnitud del exterior y transformarla en otra magnitud, normalmente eléctrica, que seamos capaces de cuantificar y manipular.

Normalmente estos dispositivos se encuentran realizados mediante la utilización de componentes pasivos (resistencias variables, PTC, LDR, etc. todos aquellos componentes que varían su magnitud en función de alguna variable), y la utilización de componentes activos.

A continuación se indican algunos tipos y ejemplos de sensores electrónicos:

- Sensores de temperatura: Termopar, Termistor, RTD.
- Sensores de deformación: Galga extensiométrica.
- Sensores de acidez: IsFET.
- Sensores de luz: fotodiodo, fotorresistencia, fototransistor.
- Sensores de sonido: micrófono.
- Sensores de imagen digital (fotografía): CCD o CMOS.
- Sensores de proximidad: sensor de proximidad.

Por lo general la señal de salida de estos sensores no es apta para su lectura directa y a veces tampoco para su procesado, por lo que se usa un circuito de acondicionamiento.

1.6.1 Sensor de Temperatura LM35

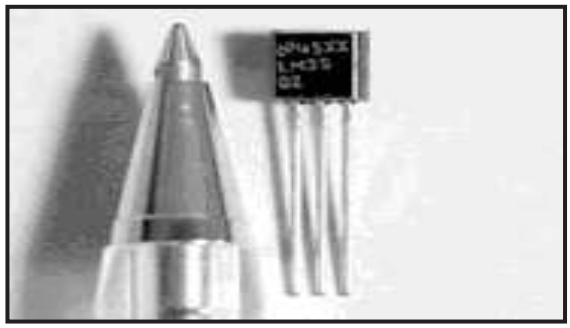
En la *Figura 1.14* se puede observar al sensor de temperatura LM35 con una precisión calibrada de 1ºC. Puede medir temperaturas en el rango que abarca desde -55º a + 150ºC. La salida es lineal y equivale a 10mV/ºC por lo tanto:

---

[6] http://es.wikipedia.org/wiki/Sensor.

- +1500mV = 150ºC
- +250mV = 25ºC
- -550mV = -55ºC

Figura 1.14 Sensor de Temperatura LM35.



Fuente: http://es.wikipedia.org/wiki/Sensor

Sus características más relevantes son:

- Precisión de ~1,5ºC (peor caso), 0.5ºC garantizados a 25ºC.
- No linealidad de ~0,5ºC (peor caso).
- Baja corriente de alimentación (60uA).
- Amplio rango de funcionamiento (desde -55º a + 150ºC).
- Bajo costo.
- Baja impedancia de salida.

Funcionamiento

Su tensión de salida es proporcional a la temperatura, en la escala Celsius, funciona en el rango de alimentación comprendido entre 4 y 30 voltios y, es de bajo costo. Como ventaja adicional, el LM35 no requiere de circuitos adicionales para su calibración externa cuando se desea obtener una precisión del orden de ±0.25 ºC a temperatura ambiente, y ±0.75 ºC en un rango de temperatura desde -55 a150 ºC.

La baja impedancia de salida, su salida lineal y su precisa calibración inherente hacen posible una fácil instalación en un circuito de control. Debido a su baja corriente de alimentación (60uA), se produce un efecto de auto calentamiento reducido, menos de 0.1 ºC en situación de aire estacionario.

Encapsulado

El sensor LM35 se presenta en diferentes tipos de encapsulados, pero el más común es el T0-92 (siendo el más idóneo para nuestra aplicación), de igual forma que un típico transistor con 3 pines, dos de ellas para alimentarlo y la tercera nos entrega un valor de tensión proporcional a la temperatura medida por el dispositivo.

Como se muestra en la *Figura 1.15*, el LM35 viendo las letras del encapsulado hacia arriba tenemos de izquierda a derecha que los pines son:

- VCC
- Vout
- GND.

Figura 1.15 Distribución de pines del LM35.

Fuente: http://es.wikipedia.org/wiki/Sensor

Usos:

El sensor de temperatura puede usarse para compensar un dispositivo de medida sensible a la temperatura ambiente, refrigerar partes delicadas del robot ó bien para lograr temperaturas en el transcurso de un trayecto de exploración.

1.7 Lenguajes de programación[7]

1.7.1 Lenguajes de bajo nivel

Se llaman de bajo nivel porque están muy cercanos al Hardware del ordenador. Este es el primer lenguaje que se utilizó para el lenguaje máquina, que consiste en un conjunto de instrucciones en binario, con ceros y unos, con los cuales se indica al ordenador qué hacer.

Este lenguaje es muy complicado y la posibilidad de cometer errores es muy alta, para solventar estas dificultades apareció el lenguaje ensamblador, que consiste en asignar una abreviatura a cada instrucción en binario, de forma que sea más fácil recordarla y menos probable equivocarse.
Sin embargo, con este lenguaje es necesario conocer muy bien el Hardware del ordenador.

1.7.2 Lenguajes de alto nivel:

Estos lenguajes se encuentran más cercanos al lenguaje natural (lenguaje humano) que al lenguaje máquina. Se trata de lenguajes independientes de la arquitectura del ordenador. Por lo que, en principio, un programa escrito en un lenguaje de alto nivel, se puede enviar de una máquina a otra sin ningún tipo de problema.

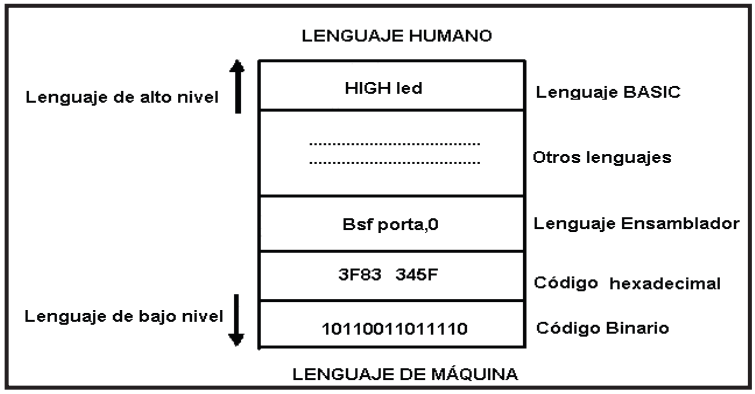
Estos lenguajes permiten al programador olvidarse por completo del funcionamiento interno de la máquina para la que están diseñando el programa. Tan solo necesitan un traductor que entienda el código fuente como las características de la máquina.

En la *Figura 1.16* se puede observar la diferencia de los dos tipos de lenguajes.

---

[7] http://es.wikipedia.org/wiki/lenguajesprogramacion.edu

Figura 1.16 Niveles de lenguajes de programación



Fuente: http://www.ucontrol.com.ar_wiki/programacion

1.7.3 Lenguaje de programación PBP

Se utilizó el programador PBP (Pic Basic Pro) que tiene muchas librerías y funciones como un compilador real por lo que los programas se ejecutan mucho más rápido.

PBP tiene una variedad de instrucciones utilizadas para programar microcontroladores. A continuación se describen las instrucciones utilizadas en el desarrollo del programa para el Sistema de Automático de Ventilación.

DEFINE: Algunos elementos, como el oscilador y las ubicaciones de los pines del LCD, están predefinidos en PBP. DEFINE, le permite al programa PBP cambiar estas definiciones si así se desea.

Ejemplo: DEFINE LCD_DREG PORT B
DEFINE LCD_DBIT 4

La primera instrucción define los pines del LCD desde el pórtico B4 hasta B7, la segunda instrucción nos indica que va empezar desde el puerto B4 hasta el B7.

IDENTIFICADORES: Un identificador es simplemente un nombre. Los identificadores son usados en PBP como etiquetas de líneas y nombres de variables. Un identificador es cualquier secuencia de letras, dígitos y símbolos, aunque no deben comenzar con un dígito.

Los identificadores no distinguen las letras mayúsculas de las minúsculas, por lo que dice la etiqueta se puede escribir de las siguientes formas; etiqueta, ETIQUETA, Etiqueta estos ejemplos son tratados como equivalentes. Aunque las etiquetas pueden tener cualquier número de caracteres de longitud PBP solamente reconoce los primeros 32.

VARIABLES: Es donde se guardan datos en forma temporal en el programa PBP, son creadas usando la palabra clave **VAR**, pueden ser bits, bytes ó word. El formato para crear una variable es el siguiente:
Etiqueta **VAR** tamaño

Etiqueta es cualquier identificador. Sus tamaños son: bit (valores 0 ó 1), byte (enteros de 0 a 255) ó word (enteros de 0 a 65535).Crean una variable y le asignan un tamaño ya sea BIT, BYTE O WORD.
Ejemplo: x **VAR BYTE**

PINES: A los pines se puede acceder de diferentes modos, el mejor camino para especificar un pin para una operación, es usando simplemente sus nombres PORT (A ó B) y un número de bit.

Ejemplo: PORTD.4=1
Esta instrucción indica, que el pórtico D4 se debe sacar 1 Lógico.

Para recordar fácilmente el uso de un pin, puede asignársele un nombre usando el comando **VAR**, de esta manera, el nombre puede ser utilizado luego en cualquier operación.
Ejemplo: Led**VAR** PORTD.4

ETIQUETAS DE LÍNEA (LABELS): Para marcar líneas que el programa puede desear referenciar con comandos como el GOTO, PBP usa etiquetas de línea. Cualquier línea PBP puede comenzar con una etiqueta de línea que es simplemente un identificador seguido por dos puntos (:).

Ejemplo: sensar**:**
PAUSE 1000
GOTO sensar

HIGH: Coloca el Pin especificado en valor alto (5 voltios) y lo convierte automáticamente en salida.

Ejemplo: led **VAR** PORTD.4
HIGH led

LOW: Coloca el pin especificado en valor bajo (0 voltios) y automáticamente lo convierte en salida.

Ejemplo: Led **VAR** PORTD.4
LOW led

GOTO: La ejecución del programa continúa en la declaración de la etiqueta.

Ejemplo: **GOTO**sensar
sensar:
HIGH led

PAUSE: Detiene el programa por el tiempo señalado en milisegundos por "Periodo". El Periodo tiene 16 bits, por lo que los retardos pueden ser de hasta 65.535 milisegundos. No coloca al microcontrolador en modo de baja potencia como las otras funciones de retardo (NAP y SLEEP), inclusive, consume mayor potencia, pero es más exacto ya que tiene la misma precisión que el reloj.

Ejemplo: **PAUSE** 1000

Este pause de 1000 nos indica que el programa, espera un segundo para ejecutar la siguiente instrucción.

IF...THEN puede operar de dos maneras:

De una forma, el **THEN** en un **IF...THEN** es esencialmente un GOTO. Si la condición es cierta, el programa irá hacia la etiqueta que sigue al **THEN**, si la condición es falsa, el programa va a continuar hacia la próxima línea después del **IF**...**THEN**. Otra declaración no puede ser puesta después del **THEN**, sino que debe ser una etiqueta. Todas las comparaciones no tienen signo, ya que PBP solo soporta operaciones sin signo.

Ejemplo: **IF** PORTD.4=1 **THEN** espere

En la segunda forma, **IF...THEN** puede ejecutar condicionalmente un grupo de declaraciones que sigan al **THEN**. Las declaraciones deben estar seguidas por un ELSE o un ENDIF para completar la estructura.

Ejemplo:
```
IF PORTD.4=1 THEN
        PAUSE 1000
ELSE
        PAUSE 5000

        ENDIF
```

EEPROM 5, [3,"K", 9,12]: Quiere decir colocar en la memoria EEPROM, dirección 5 el número 3, en la dirección 6 del carácter ASCII de K, es decir el número 75, y en la dirección 7 se guardará el numero 9 y así sucesivamente.
La declaración **EEPROM** define el contenido inicial en un chip EEPROM.

READ 5, **pepe:** Quiere decir leer la dirección 5 de la memoria **EEPROM** y guardar en la variable pepe, pero consiguiente pepe se carga con el número3. Además la declaración READ lee byte de un chip EEPROM

WRITE 7, pepe: En este caso la variable estaba cargada con el número 3 por consiguiente la dirección 7 de la memoria EEPROM se borra y se carga con el número 3. La declaración WRITE graba bytes en un chip EEPROM.

Operadores de comparación: Se usan en declaraciones **IF... THEN** para comparar una expresión con otra. Los operadores soportados se muestran en la Tabla 1.4

Tabla 1.4 Operadores.

Operador	Descripción
= o ==	Igual
<>ó !=	No igual
<	Menor
>	Mayor
<=	Menor o igual
>=	Mayor o igual

FOR … NEXT: El loop **FOR … NEXT** permite a los programas ejecutar un número de declaraciones (Body), un número de veces, usando una variable como contador.

FOR Count = Start **TO** End {STEP {-} Inc}
{Body}
NEXT {Count}

El valor de Start se asigna a la variable índice, Count, puede ser una variable de cualquier tipo. Se ejecuta el Body, que es opcional y puede ser omitido (quizás por un loop de demora).
El valor de Inc es sumado ó restado si se especifica "-" a Count. Si no se define una cláusula STEP, se incrementa Count en uno.

Ejemplo:
```
FOR contador = 1 TO 10

        PAUSE 1000
NEXT
```

RETURN y GOSUB: Estos sirven cuando se tiene muchas repeticiones de una línea o grupo de líneas de programa.

La declaración **RETURN**, envía de regreso al inicio del programa para continuar después del **GOSUB** ya que este tiene una subrutina de cualquier nombre que le hayamos asignado, en donde se especifica el tiempo con un **PAUSE** con su tiempo respectivo, lo cual no es necesario escribir este tiempo en el programa varias veces sino solo una vez en la subrutina.

Ejemplo:

```
Inicio:
        portb=%00001
        GOSUB x
        portb=%00010
        GOSUB x
        GOTO inicio
x:
        Pause 1000
RETURN
```

END: Detiene la ejecución del proceso y entra en modo de baja potencia. Todos los pines de I/O permanecen en el estado en que se encuentran, **END** trabaja ejecutando una instrucción **SLEEP** continua dentro de un loop.

Ejemplo:

```
END
```

TRIS (A,B,C,D,E): Esta declaración nos permite indicar al PIC cuales pines serán salida y cuales serán entradas.

Ejemplo:

```
TRISA = %00001111
```

Esto significa que los cuatro bits menos significativos son entradas y los cuatro bits más significativos son salidas.

CAPITULO 2

PROTOTIPO DE UN SISTEMA DE CONTROL AUTOMÁTICO DE TEMPERATURA

2.1. INTRODUCCIÓN

El prototipo del Control Automático de Temperatura es activado por la variación de la temperatura ambiente, la cual esta dentro de un rango de temperatura que lo establece el usuario con la manipulación de los pulsadores, los que permiten cambiar el valor de dicho rango, de esta manera el usuario mantendrá el control sobre la temperatura requerida en un área de trabajo cerrada, en los casos de un ambiente frio se encenderá la calefacción o de lo contrario, si el ambiente es muy caluroso se activará un ventilador.

2.2 DESARROLLO DEL SOFTWARE

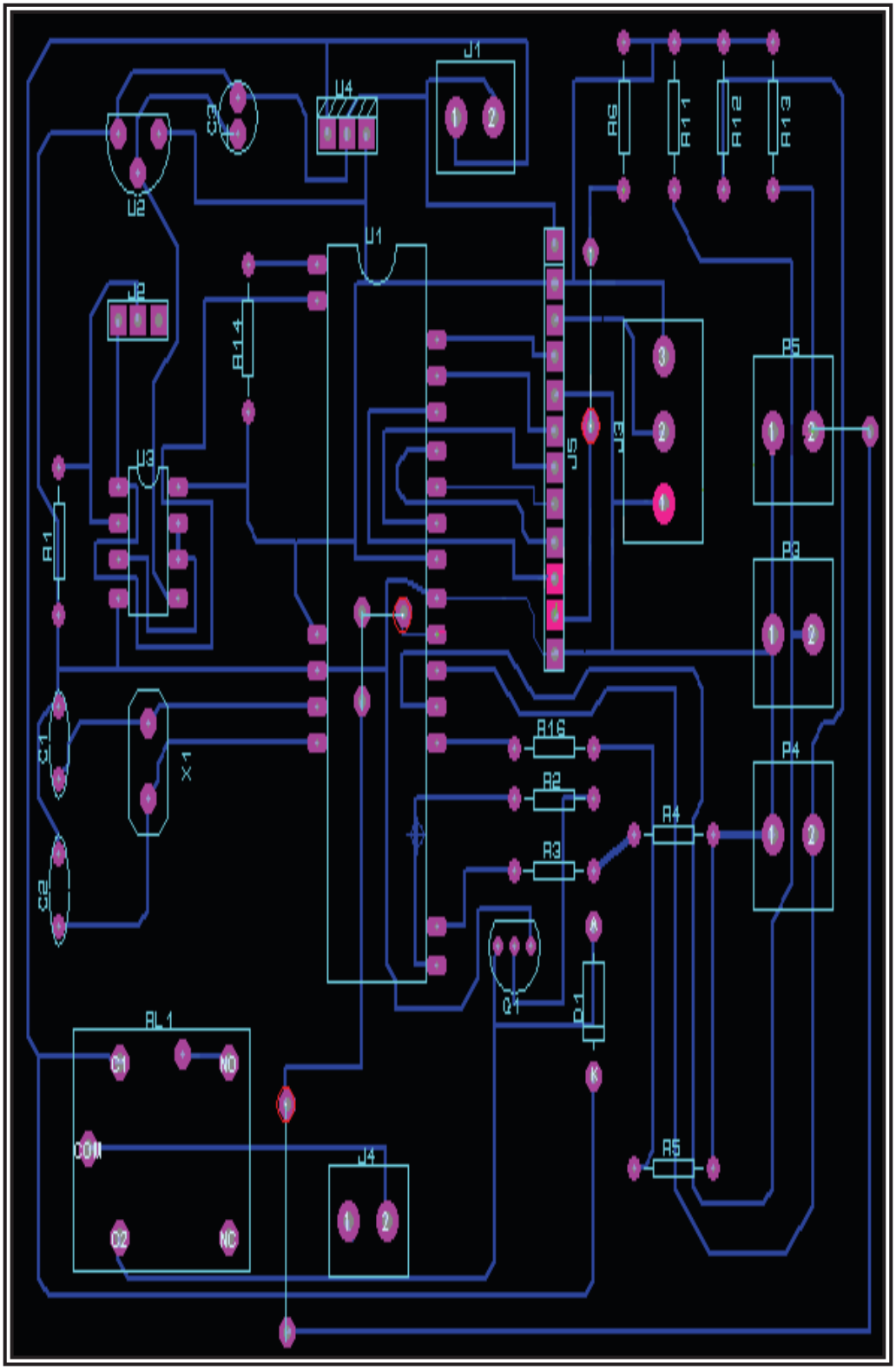
Antes de iniciar el desarrollo del software se debe tener claro cada una de las actividades que el sistema va a realizar.

Para la elaboración del software del microcontrolador se utilizó el compilador PIC Basic Pro (PBP), es un lenguaje de programación de alto nivel debido a su fácil programación. El lenguaje Pic Basic Pro es mucho más fácil de leer y escribir que el lenguaje ensamblador.

2.3 DISEÑO DEL CIRCUITO IMPRESO

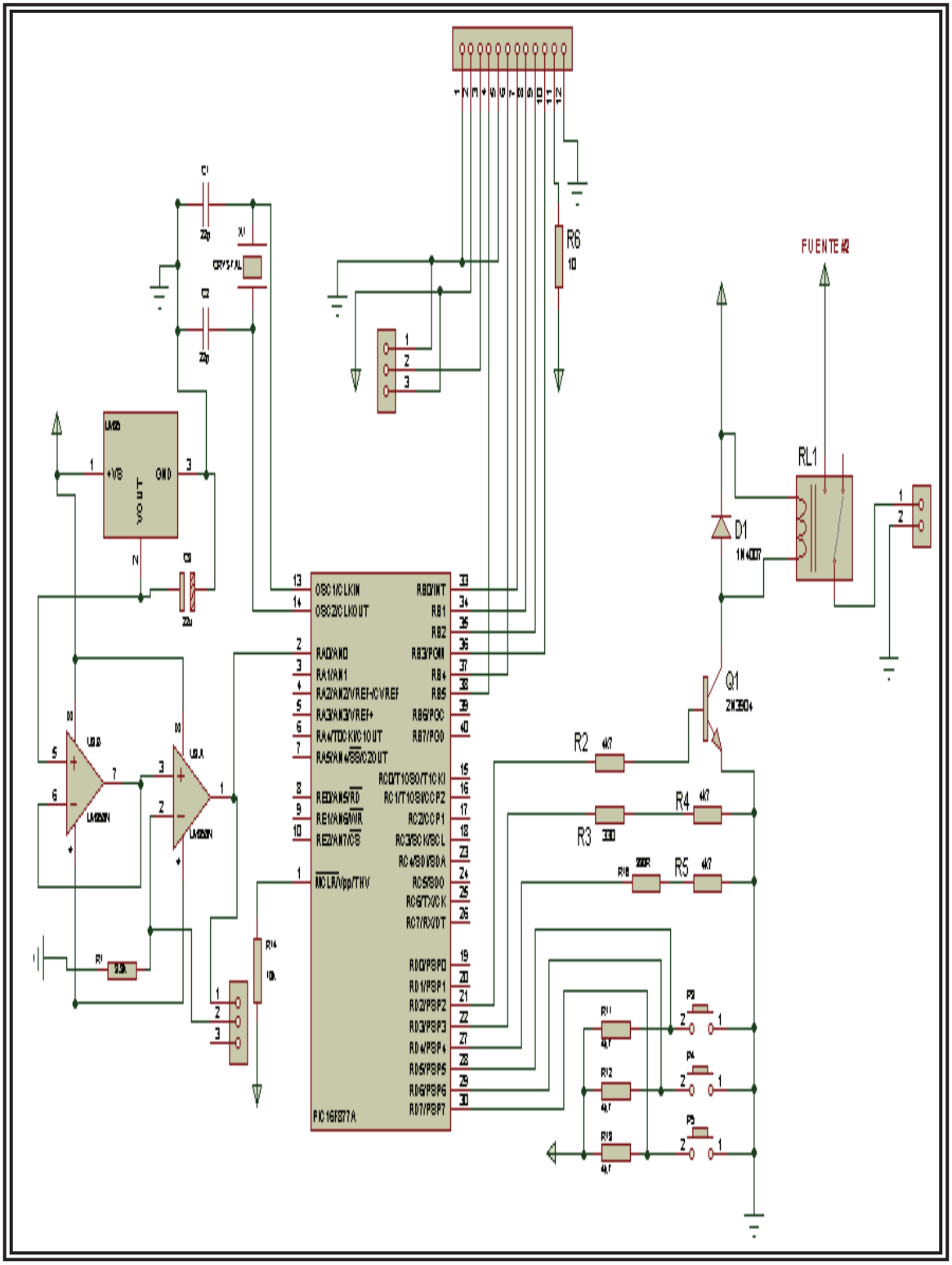
El diseño del ruteado de las tarjetas se ha llevado a cabo con la ayuda de los programas Eagle 4.11 y Proteus V7.1 SP2 (ARES e ISIS). El diagrama esquemático de la tarjeta principal se lo elaboró en Proteus ISIS y el ruteado de la misma en Proteus Ares. *(Figura 2.1 y 2.2* respectivamente)

Figura 2.1 Ruteado de la placa principal en Proteus ARES



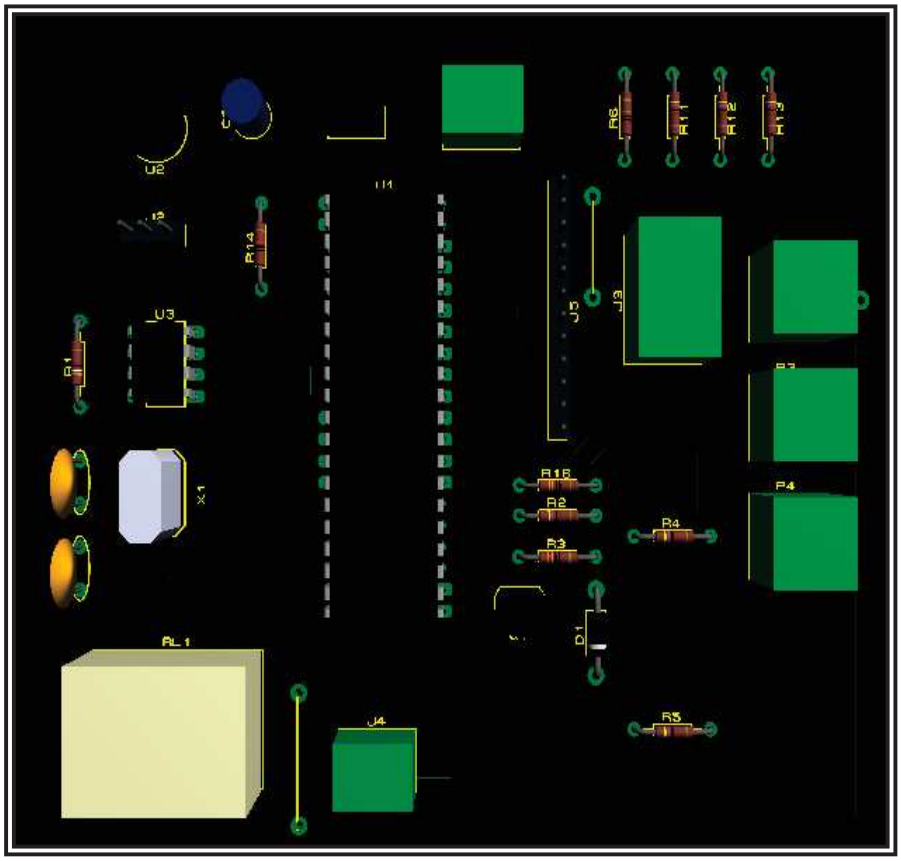
Fuente: Margoth Valencia Lalangui

Figura 2.2 Diagrama esquemático de la placa principal



Fuente: Margoth Valencia Lalangui

Figura 2.3 Diseño de la placa principal en 3D



Fuente: Margoth Valencia Lalangui

2.4 ELABORACIÓN DE LA TARJETA PRINCIPAL DEL PROTOTIPO DE CONTROL AUTOMÁTICO DE TEMPERATURA

Se procede a la elaboración del circuito de la placa principal

Los pasos a seguir son los siguientes:

- Teniendo ya el diseño de las pistas (ruteado) con la ayuda de los paquetes electrónicos Eagle y Proteus, empleando la herramienta ARES se procedió a la impresión en láser en un papel especial llamado papel termo transferible.

- En una baquelita de cobre con las dimensiones de 10cm x8cm para la placa principal, se procede a la adhesión del papel térmico sobre la misma, lo cual se realiza con una plancha casera a máxima temperatura durante 30 minutos.

- Terminado ya el planchado de la placa se retocó con marcador donde el papel térmico no se haya adherido correctamente.

- Luego se prepara el ácido disolviéndolo con un poco de agua, para colocar la placa de cobre con el diseño del circuito impreso hacia arriba. Mientras la placa de cobre se encuentre sumergida en cloruro férrico se debe agitar el recipiente donde se encuentra inclinándolo de lado a lado para que de esta manera el químico pueda disolver más rápido el cobre de la placa.

- Después de que el cloruro férrico haya consumido todo el cobre, se procede a sacar la placa del recipiente y a retirar la tinta con thinner y un trapo. Para dar mayor presentación al circuito impreso se lo debe lavar y secar con papel de cocina, de ser necesario pulir suavemente con viruta de acero.

- Con un probador de continuidad verificar que todas las pistas lleguen de un punto a otro. En caso de haber una pista cortada estañarla desde donde se interrumpe hasta el otro lado y colocar sobre ella un fino alambre telefónico, de ser una pista ancha colocar alambre más grueso o varios uno junto a otro.

- Se comienza a perforar en los lugares indicados, para esto se debe emplear una broca de 75mm de espesor, para los orificios de resistencias, capacitores y semiconductores, para los orificios de bornera y pines de los diferentes integrados y del PIC una broca de 1mm es la más adecuada. Aquí será de suma utilidad acertar al orificio central de la isla para que quede la hilera de perforaciones lo más pareja que sea posible.

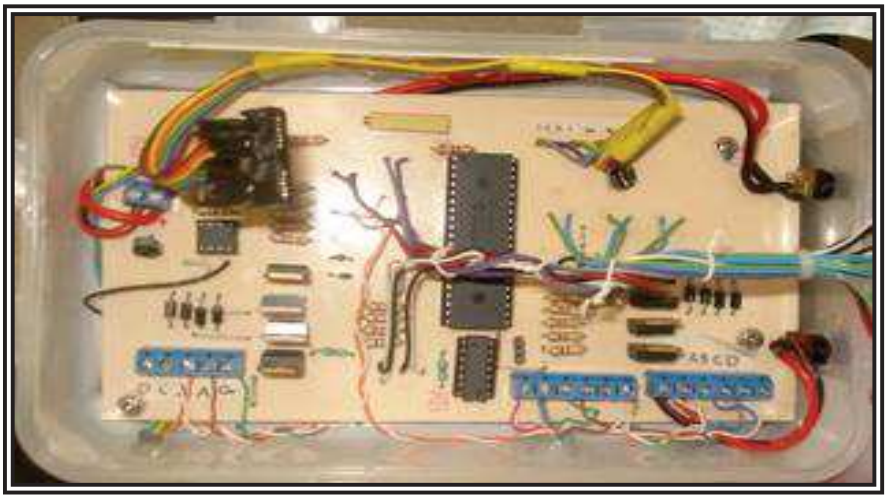
- Colocados los elementos en su lugar y posiciones respectivas se procede a soldar con un cautín de 40w, procurando no dejar pistas cristalinas o también llamadas sueldas frías. Tomando en cuenta que siempre se debe montar primero los componentes de menor espesor, comenzando por los puentes de alambre, diodos, resistencias, pequeños capacitores, transistores, pines de conexión y los zócalos de circuitos integrados. Lo más adecuado es montar zócalos para los circuitos integrados, para cuando sea necesario reemplazarlos en futuras reparaciones será simple quitar uno y colocar otro sin siquiera usar el cautín.

- Terminado ya de soldar todos los elementos se debe limpiar la placa del excedente de pomada con un cepillo y thinner.

Acabado final de la placa electrónica

En la *Figura 2.4* se muestra el resultado final de la placa principal, la misma que tiene mucha similitud con la simulación en 3D que nos proporcionó el programa Proteus ARES.

Figura 2.4 placa principal del Prototipo de Control Automático de Temperatura.



Fuente: Margoth Valencia Lalangui

La placa principal se la ha colocado en una tabla de 40cm x 60cm, en la que también ira la pantalla LCD, el ventilador y la fuente de voltaje todas aseguradas con tornillos tratando así de que se encuentren seguras y no se provoque algún daño por la transportación del prototipo. Ver figura 2.5.

Figura 2.5. Prototipo Control Automático de Temperatura.

Fuente: Margoth Valencia Lalangui

2.5. PARÁMETROS TÉCNICOS A CONSIDERAR

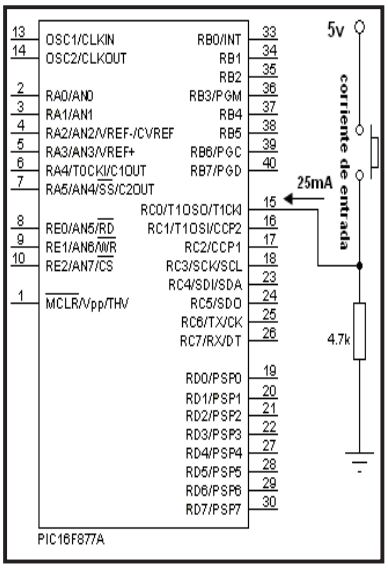
Es muy importante tomar en cuenta estas recomendaciones ya que si no se las sigue podría correr el riesgo de dañar el PIC:

1) El PIC 16F877A es de tecnología CMOS, esto quiere decir que consume muy poca corriente pero a la vez es muy susceptible a daños por estática, se recomienda utilizar una manilla antiestática y así poder transportar desde el grabador al protoboard y viceversa.

2) Se emplea un regulador de voltaje como el L7805C-V que nos entrega exactamente 5vDC, y no un adaptador de pared, ya que el voltaje de salida no siempre es el mismo que indica su fabricante.

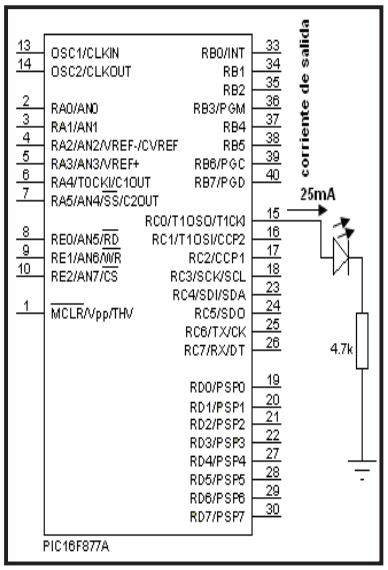
3) No se debe sobrepasar los niveles de corriente, tanto de entrada como de salida, el PIC puede entregar por cada uno de sus pines una corriente máxima de 25 mA. Así mismo soporta una corriente máxima de entrada de 25mA. Como se ven en las *Figuras 2.6a y 2.6b.*

Figura 2.6a Corriente de entrada.



Fuente: http://www.microchip.com

Figura 2.6b Corriente de salida.



Fuente: http://www.microchip.com

Para encender un led con la resistencia adecuada se debe considerar los siguientes valores:

El voltaje de salida del PIC, si es alimentado con 5Vdc= 5Vdc out.
Corriente que requiere un led para un encendido normal= 15 mA.

$$V = RxI$$
$$R = \frac{V}{I}$$
$$R = \frac{5V}{0{,}015A}$$

$$R = 333{,}33\Omega = 330\Omega$$

Para encender un led sin correr el riesgo de dañarlo se necesitará colocar una resistencia de 330Ω.

Para el correcto funcionamiento de la tecla se debe tomar en cuenta los siguientes valores:

La corriente de entrada que soporta el PIC por cada pin es 25mA, si el voltaje de alimentación del PIC, es 5Vdc.

$$V = RxI$$
$$R = \frac{V}{I}$$
$$R = \frac{5V}{0{,}025A}$$

$$R = 200\Omega = 220\Omega$$

Esto quiere decir que la resistencia mínima a colocarse sería de 220Ω para estar al límite de la capacidad que soporta el PIC, pero no es muy aconsejable colocar la resistencia justa, por lo que se recomienda utilizar una resistencia de 1kΩ a 10kΩ, así el PIC estaría trabajando tranquilamente con una corriente de entrada de 5mA o 0,5mA respectivamente. Pero en la aplicación hemos trabajado con resistencias de 4,7KΩ con una corriente de 0,00106 Amperios o 1,06 mA.

4) Cuando se necesite precisión en el trabajo del PIC 16F877A, se recomienda utilizar un oscilador externo de 4Mhz (de configuración HS) debido a que estos cristales son muy precisos en cuanto a la frecuencia que entregan, ya que el oscilador interno RC no posee una buena precisión.

2.6 PRUEBAS Y RESULTADOS

- Lo primero que se hizo, fue probar el proyecto montado en el protoboard, para verificar que las conexiones y los voltajes sean los correctos, y que este funcione de manera adecuada antes de transferirlo a una baquelita.

- La siguiente prueba fue transferir el proyecto a la baquelita y revisar que no existan pistas rotas o unidas por error, puentes faltantes, revisar que la polaridad de los elementos sea la correcta.

- Antes de alimentar la tarjeta principal se verificó con un multímetro el voltaje (5V) para evitar dañar los elementos por exceder el voltaje.

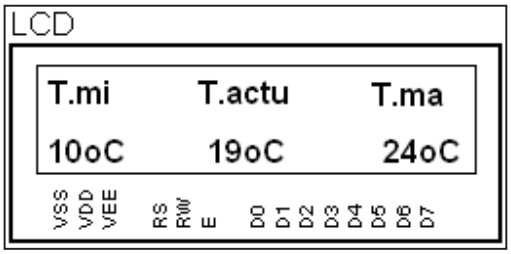
- Se probó el circuito utilizando una sola fuente de alimentación, y se presentó un inconveniente, al momento que el ventilador arrancaba la fuente no suministraba la corriente necesaria para todo el circuito, provocando que el Pic se reinicie. Por esta razón se optó por utilizar una fuente individual tanto para el ventilador como para la placa principal.

2.7 FUNCIONAMIENTO DEL PROTOTIPO

El primer paso a realizar es alimentar el circuito con los 5V que accionan al relé con una fuente de alimentación, y al momento de encender el prototipo, se observa el led verde titilar tres veces seguidas indicando que el PIC está trabajando perfectamente.

A continuación el PIC envía los datos al LCD como se indica en la *Figura 2.7,* tomando en cuenta que la temperatura actual es la que está sensando el sensor LM35.

Figura 2.7 Presentación de datos en el LCD



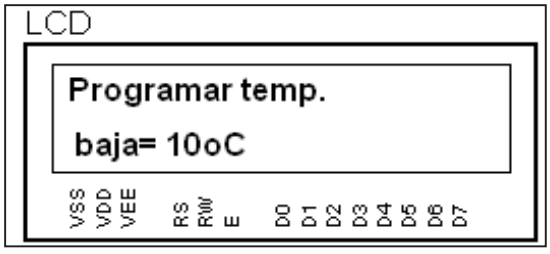
Fuente: Margoth Valencia Lalangui

La temperatura actual es la que el PIC está sensando a cada instante con las variaciones de voltaje que proporciona el LM35 por el cambio de temperatura que existe en el ambiente donde se encuentra dicho sensor.

La temperatura mínima y temperatura máxima que aparecen en el LCD son las que están grabadas previamente en el EEPROM del PIC, estas pueden variar con el uso de las teclas Enter, Subir y Bajar a conveniencia del usuario.

Al presionar la primera vez la tecla Enter tenemos en el LCD lo que indica en la *Figura 2.8* mientras que con el uso de las teclas subir y bajar variamos aquel valor.

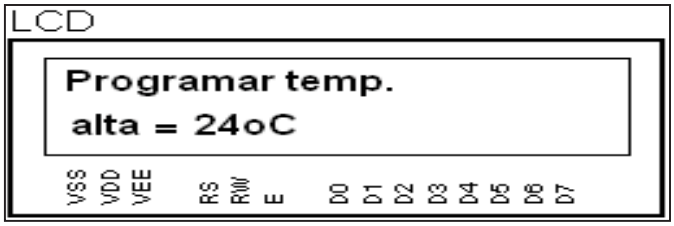
Figura 2.8 Pantalla informando temperatura baja.



Fuente: Margoth Valencia Lalangui

Presionando por segunda ocasión la tecla Enter el led titila una sola vez y el LCD cambia como se indica en la *Figura 2.9,* de igual manera al presionar las teclas subir o bajar variamos aquel valor.

Figura 2.9 Pantalla informando temperatura alta.



Fuente: Margoth Valencia Lalangui

Si la temperatura actual llega a disminuir por debajo del valor de la temperatura mínima el PIC ordenará enviar 1L al pin donde se encuentra conectado el led de color rojo, simbolizando que en ese momento debe encenderse un calefactor. El led permanecerá encendido hasta que la temperatura actual iguale o sobrepase el valor de la temperatura mínima.

Si la temperatura actual llega a exceder el valor de la temperatura máxima el PIC ordenará enviar 1L al pin donde se encuentra conectado el transistor que funciona como un interruptor de encendido y apagado del relé. La bobina del relé al energizarse provoca que su contacto normalmente abierto cambie a normalmente cerrado activando al ventilador, el mismo que permanecerá encendido hasta que la temperatura actual iguale o disminuya el valor de la temperatura máxima.

Se debe recalcar que nunca el ventilador y el calefactor se encenderán en el mismo instante, ya que la condición para que cualquiera de los dos funcione, es que el otro este apagado y así se cumple la finalidad del prototipo que es regular la temperatura.

CAPÍTULO III: CONCLUSIONES Y RECOMENDACIONES

3.1 CONCLUSIONES:

1. Se concluye que la implementación de un Prototipo de Control Automático de Temperatura en un ambiente cerrado, es de gran utilidad para lugares donde la temperatura varía constantemente.

2. La elaboración del software debe ser realizada siguiendo los comandos correspondientes para cada función que el microcontrolador desarrollara de acuerdo a los requerimientos del usuario.

3. Se concluye que la elaboración de la tarjeta principal se realizó de una manera eficaz y optimizando los recursos, ya que gracias al paquete electrónico Proteus se evitó gastos innecesarios debido a la simulación que nos ofrece dicho programa.

4. En el proceso de este proyecto de titulación se aplicó y desarrolló todos los conocimientos adquiridos en la materia de Electrónica y Domótica.

5. Para un ahorro de pines del PIC16F877A se conecta solo 4 bits para la comunicación entre el PIC y el LCD 2X16, en lugar de utilizar los 8 bits de comunicación debido a que la comunicación existente entre estos dos elementos se la está realizando de forma serial es decir el microcontrolador envía al LCD los datos uno a uno para su posterior presentación.

6. La fuente de voltaje debe tener una corriente que abastezca al PIC y al ventilador, ya que si no es suficiente al momento de arrancar el ventilador el PIC se reiniciará hasta que vuelva a tener la corriente necesaria.

3.2. RECOMENDACIONES:

1. La recomendación más relevante es tener cuidado al momento de conectar la alimentación a la placa principal, ya que el voltaje de alimentación es de 5V y si este se sobrepasa el PIC dejará de funcionar ocasionando daños en los elementos del prototipo.

2. Para calibrar la sensibilidad del termómetro digital se recomienda utilizar un potenciómetro de precisión.

3. Se recomienda utilizar fuentes individuales tanto para el PIC como para el ventilador, así se podrá escoger entre la variedad de ventiladores que hay en el mercado ya que el relé soporta hasta 125v.

4. Evitar que el sensor de temperatura LM35 este expuesto al contacto con el agua para que no sufra daños físicos.

5. Se recomienda que este prototipo sea implementado en lugares cerrados donde existe variación de temperaturas sean estas altas o bajas las mismas que generan malestar en el personal y disminuye su ritmo de trabajo.

ANEXOS

ANEXO I

ANALISIS DE COSTOS

DETALLE	COSTO
Paquetes electrónicos	20
Diseño de software	60
Programación del microcontrolador	10
Adquisición de elementos	80
Fotocopias	35
Internet	50
Impresiones	75
Varios	90
TOTAL	**420**

ANEXO 2

DESCRIPCIÓN DEL PROGRAMA DISEÑADO Y REALIZADO EN MICRO CODE STUDIO

DEFINE LCD_DREG PORTB

Esta primera línea de instrucción nos permite definir el puerto del PIC que utilizaremos para la conexión del bus de datos del LCD (Puerto B).

DEFINE LCD_DBIT 0

Esta instrucción nos indica desde que pin del puerto B vamos a conectar el bus de datos del LCD (B.0 al B.4).

DEFINE LCD_RSREG PORTB

Define en que puerto va a ser conectado el bit del registro del LCD (Puerto B).

DEFINE LCD_RSBIT 5

El bit de registro va a estar conectado en el pin B.5

DEFINE LCD_EREG PORTB

Define en que puerto va a ser conectado el bit de enable del LCD (Puerto B).

DEFINE LCD_EBIT 4

El bit de enable va a estar conectado en el pin B.4

DEFINE ADC_BITS 8

Fija el número de BITS del resultado.

DEFINE ADC_CLOCK 3
Fije EL CLOCK (rc = 3)

DEFINE ADC_SAMPLEUS 50
Fija el tiempo de muestreo en µs. ADC_SAMPLEUS es el número de microsegundos que el programa espera entre fijar el canal y comenzar la conversión análoga/digital.

TRISA = %00000001	Indica que solo A.0 es entrada, el resto son salidas
TRISE = %00000000	Todo el puerto A es salida.
ADCON1 =%00001110	El puerto A.0 es conversor A/D los demás Digitales
TRISC = %00000000	Todo el puerto C es salida
TRISD = %11100000	Los pines D.5, D.6 y D.7 son entradas el resto son salidas
Dato **VAR BYTE**	Crea una variable y asigna un tamaño de 8 bits
tempbaj **VAR BYTE**	Crea una variable y asigna un tamaño de 8 bits
tempalt **VAR BYTE**	Crea una variable y asigna un tamaño de 8 bits
x **VAR BYTE**	Crea una variable y asigna un tamaño de 8 bits
Ventilador **VAR** portD.2	Asignamos un nombre al puerto D.2
Calefactor **VAR** portD.3	Asignamos un nombre al puerto D.3
Led **VAR** portD.4	Asignamos un nombre al puerto D.4
enter **VAR** portD.5	Asignamos un nombre al puerto D.5
bsubir **VAR** portD.6	Asignamos un nombre al puerto D.6
bbajar **VAR** portD.7	Asignamos un nombre al puerto D.7
EEPROM 0,[22,26]	Guardamos desde la dirección 0 de la EEPROM el número 22 y en la dirección 1 el número 26.

Inicio:	Nos indica el inicio del programa
FOR x =1 **TO** 3	Crear un lazo de tres repeticiones
HIGH led	Sacar 1L en led
PAUSE 200	Led permanezca encendido 200 ms
LOWled Sacar 0L en led.	
PAUSE 200	Led permanezca apagado 200ms.
NEXT	Terminado el lazo, continúa con el resto del programa
READ 0, tempbaj	Lee la EEPROM 0 y lo guarda en la variable tempbaj.
READ 1, tempalt	Lee la EEPROM 1 y lo guarda en la variable tempalt.
sensar:	Nombre de subrutina.
ADCIN 0, dato	Leer el canal 0 (A0) y guarde en dato
LCDOUT $fe, 1 ,"T.miT.actu T.ma"	Limpiar LCD y sacar texto
dato = dato /2	El dato dividir para 2
LCDOUT $fe,$c6,**DEC** dato,"oC"	Sacar por el LCD el valor de dato en decimal.
LCDOUT $fe,$c0,**DEC** tempbaj,"oC"	Sacar por el LCD el valor de tempbaj en decimal.
LCDOUT $fe,$cc,**DEC** tempalt,"oC"	Sacar por el LCD el valor de tempalt en decimal.
FOR x = 1 **TO** 50	Crear un lazo de 50 repeticiones

```
    IF enter =0 THEN grabar1a        Si la condición es verdadera saltar a
                                     grabar1a, caso contrario continuar con el
                                     programa.

    PAUSE 10                         Detiene el programa 10 ms.
    NEXT
    IF dato < tempbaj THEN calentar      Si las condiciones son verdades
    IF dato > tempalt THEN enfriar       ir a sus respectivas subrutinas, caso
                                         contrario continuar con el programa
    LOW calefactor: LOW ventilador   Apagar los 2 pines
    GOTO sensar                      Regresar a la subrutina sensar.
Calentar:                            Nombre de subrutina.
    HIGH calefactor: LOW ventilador  Enciende el calefactor y apaga el
                                     ventilador.
GOTO sensar                          Regresar a la subrutina sensar.

Enfriar:
    HIGH ventilador: LOW calefactor  Enciende el ventilador y apaga el
                                     calefactor
    GOTO sensar                      Regresar a la subrutina sensar.

grabar1a:                            Nombre de subrutina.
    GOSUB soltar                     Saltar a soltar y regresar.

grabar1:                             Nombre de subrutina.
    LCDOUt $fe, 1 ,"Programar temp."
    LCDOUT $fe,$c0,"baja= ",DEC tempbaj ," oC"
    PAUSE 100
    IF bbajar=0 THEN restar1         Si las condiciones son verdaderas
    IF bsubir=0 THEN sumar1          ir a sus respectivas subrutinas,
```

IF enter=0 **THEN** grabarA	caso contrario continuar con el
GOTO grabar1	programa
restar1:	Nombre de subrutina.
GOSUB soltar	Saltar a soltar y regresar.
IF tempbaj< 1 **THEN** grabar1	Si la condición es verdadera saltar a grabar1, caso contrario continuar con el programa.
tempbaj= tempbaj -1	Disminuir 1 a la variable tempbaj y grabar el resultado de la resta en tempbaj.
GOTO grabar1	Regresar a la subrutina grabar1
sumar1:	Nombre de subrutina.
GOSUB soltar	Saltar a soltar y regresar.
IF tempbaj> 40 **THEN** grabar1	Si la condición es verdadera saltar a grabar1, caso contrario continuar con el programa.
tempbaj= tempbaj + 1	Sumar 1 a la variable tempbaj y garbar el resultado de la suma en tempbaj.
GOTO grabar1	Regresar a la subrutina grabar1.
grabarA:	Nombre de la subrutina
GOSUB soltar	Saltar a soltar y regresar.
WRITE 0,tempbaj	Escribir en la dirección 0 de la EEPROM el valor de la variable tempbaj.

```
grabar2:                                    Nombre de la subrutina
    LCDOUT $fe, 1 ,"Programar temp."
    LCDOUT $fe,$c0,"Alta= ",dectempalt ," oC"
    PAUSE 100
    IF bbajar=0 THEN restar2          Si una condición es verdadera saltar a
    IF bsubir=0 THEN sumar2a          su respectiva subrutina, caso contrario
    IF enter=0 THEN grabarB           continuar con el programa.
    GOTO grabar2                            Regresar a la subrutina grabar2

restar2:                                    Nombre de la subrutina
    GOSUB soltar                            Saltar a soltar y regresar
    IF tempalt< 5 THEN grabar2              Si la condición es verdadera saltar a
                                            sensar, caso contrario continuar con
                                            el programa.

    tempalt= tempalt -1                     Restar 1 a la variable tempalt y
                                            garbar el resultado de la resta en
                                            tempalt.
    GOTO grabar2                            Regresar a la subrutina grabar2

sumar2:                                     Nombre de la subrutina

    GOSUB soltar                            Saltar a soltar y regresar
    IF tempalt> 50 THEN grabar2             Si la condición es verdadera saltar a
                                            sensar, caso contrario continuar con
                                            el programa.
    tempalt= tempalt + 1                    Sumar 1 a la variable tempalt y
                                            garbar el resultado de la suma en
                                            tempalt.
    GOTO grabar2                            Regresar a la subrutina grabar2
```

grabarB:	Nombre de la subrutina
GOSUB soltar	Saltar a soltar y regresar.
WRITE 1,tempalt	Escribir en la dirección 1 de la EEPROM el valor de la variable tempalt.
GOTO inicio	Regresar a la subrutina inicio.
soltar:	Nombre de la subrutina
HIGH led	El programa provoca un
LOW led	parpadeo del led.
PAUSE 150	
PAUSE 100	
RETURN	Regresa una línea debajo del gosub que le envió a esta subrutina.
END	Fin del programa.

GLOSARIO DE TERMINOS

CCD: C*harge-coupled device:*

CMOS: *Complementary Metal Oxide Semiconductor (*"Metal Óxido Semiconductor Complementario")

EEPROM*: Electrically-Erasable Programmable Read-Only Memory*

LCD: Liquid Crystal Display

LDR: Light Dependent Resistor

PBP: Pic Basic Pro

PIC: ***P****eripheral* ***I****nterface* ***C****ontroller* (Controlador de Interfaz Periférico).

PWM: P*ulse-width modulatione*

RAM: *Random Access Memory*

BIBLIOGRAFÍA

Libros:

1. Angulo, José Angulo, Ignacio.
   Microcontroladores PIC, Diseño Práctico de Aplicaciones, Tomo II
   .
2. BOYLESTAD, Robert L, Nashelsky Louis,
   Electrónica: Teoría de Circuitos y Dispositivos Electrónicos, II Edicion

3. COSTALES, Alcívar
   Apuntes de Microcontroladores.

4. REYES, Carlos
   Aprenda rápidamente a programar Microcontroladores.

Direcciones de Internet:

1. Microcontroladores
   http://www. monografias.com

2. Pantalla de cristal líquido (LCD)
   http://www.x-robotics.com/lcd.html

3. PIC 16F877A
   http://www.microchip.com/downloads/en/DeviceDoc/39582b.pdf.

4. Sensores
   http://es.wikipedia.org/wiki/Sensor.

5. Sensor de temperatura LM35
   http://www.ucontrol.com.ar/wiki/index.php?title=LM35