ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

"PROGRAMACIÓN DE TEMPERATURAS CON SUS ALARMAS UTILIZANDO EL SENSOR INTELIGENTE DS1820 EN COMUNICACIÓN ONE-WIRE CON UN MICROCONTROLADOR."

TESINA DE SEMINARIO

Previa la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentado por:

Michael David Samaniego Villarroel

Marlon Manuel Carpio Salas

GUAYAQUIL – ECUADOR

AÑO 2010

.

AGRADECIMIENTO

A Dios.

A la familia.

A todas las personas que apoyaron en el desarrollo de este trabajo.

A todos quienes fomentan el desarrollo tecnológico en Ecuador.

DEDICATORIA

A Dios por la fortaleza que nos ha brindado al realizar este trabajo, por su infinito amor reflejado en nuestros seres queridos

A nuestra familia y seres queridos por su comprensión y apoyo incondicional, quienes siempre fomentaron diligencia y perseverancia con valores éticos, y a nuestros amigos, quienes han estado en esta etapa finalizando la educación superior.

TRIBUNAL DE SUSTENTACIÓN

Ing. Carlos Valdivieso

Profesor de Seminario de Graduación

Ing. Hugo Villavicencio V.

Delegado del Decano

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de esta tesina, nos corresponde exclusivamente; y el patrimonio intelectual del mismo a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL".

(Reglamento de exámenes y títulos profesionales de la ESPOL)

Michael David Samaniego Villarroel

Marlon Manuel Carpio Salas

RESUMEN

El principal objetivo es mejorar el control de diversos sistemas electrónicos y computacionales, logrando evitar mediante alarmas compuestas por un sensor de temperatura digital, junto a un teclado 4x4 por el cual se ingresan los rangos de temperaturas máximo y mínimo proporcionados por el usuario, determinando así las alarmas en caso de una desestabilidad del sistema, usando como herramienta principal el microcontrolador 16F887, que recibe el dato censado por el DS 1820 y lo compara con los rangos para activar las alarmas. El proyecto que se describe a continuación trata la programación de temperaturas con sus alarmas utilizando el sensor inteligente DS1820 en comunicación one-wire con un microcontrolador.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El objetivo de este proyecto es diseñar e implementar un sistema de control de temperatura a través de la comunicación entre el sensor inteligente DS1820 con su protocolo de comunicación One-wire y el microcontrolador 16F887 el cual maneja los datos proporcionados por el sensor para mostrarlos por la pantalla de visualización LCD.

En el primer capítulo, se menciona una descripción general del proyecto, las partes y funciones del mismo, aplicaciones en el campo industrial y proyectos similares como el sensor LM35 que maneja distintos parámetros de temperatura pero sin usar el protocolo One-Wire..

En el segundo capítulo, se da un detalle sobre las herramientas de hardware: sensor DS1820, el PIC 16F887, el modulo de LCD, el teclado 4x4 hecho por botoneras y el programador PICKIT 2. Además de las herramientas de software: MIKROC Pro for PIC, y las librerías con las rutinas One-Wire que proporcionan información necesario para lograr que el sistema funcione correctamente.

El tercer capítulo, trata del diseño e implementación del proyecto, empezando con una prueba para conocer el funcionamiento del sensor inteligente DS1820 usando las librerías One-Wire dadas por el programa MikroC pro for PIC, la cual nos da las pautas a seguir para el desarrollo del software que se implementará junto con el teclado 4x4 y la pantalla de visualización LCD.. Se desarrolló un diagrama de bloques que detalla los elementos de nuestro proyecto, el diagrama de flujo del controlador y las funciones detalladas para la obtención de los datos obtenidos por el DS1820.

En el cuarto y último capítulo, se muestran el esquema y simulación en PROTEUS, también se muestran las pruebas realizadas y conexiones con algunos sensores. Como en PROTEUS se encontraron todos los dispositivos a usarse para la simulación fue sencillo obtener un resultado favorable al momento de realizar las pruebas tanto en el protoboard y en la placa luego de sus respectivas pruebas para obtener un producto final.

CAPÍTULO 1

1. DESCRIPCIÓN GENERAL DEL PROYECTO

1.1. Antecedentes

En el mundo, se vive un proceso tecnológico de continuo cambio, promoviendo la innovación, la calidad en muchos aspectos para mejorar los distintos sectores productivos de los países en desarrollo, esto se logra con el análisis detallado de los problemas provocados por la falta de recursos para una inmediata solución, por lo cual se debe usar nuevos avances tecnológicos para prevenir distintos problemas de sobrecalentamiento de dispositivos en el sector electrónico-industrial.

Controlar la temperatura del ambiente para evitar que dispositivos electrónicos, industriales y servidores al momento de sobrecalentarse para evitar daños y pérdidas de información configurando alarmas en base a las temperaturas máximas que el dispositivo puede tener un correcto funcionamiento.

Este proyecto tiene como finalidad buscar contrarrestar el sobrecalentamiento de dispositivos electrónicos de información programando alarmas con un microcontrolador 16f887 y usando el protocolo ONE-WIRE definido en el sensor inteligente DS1820.

1.2. Descripción del Proyecto

Para realizar el proyecto utilizamos el sensor inteligente **DS1820** el cual nos va a permitir obtener las lecturas de la temperatura del sistema, mediante su protocolo **ONE-WIRE,** este protocolo es en un bus, un maestro y varios esclavos de una sola línea de datos en la que se alimentan. Por supuesto, necesita una referencia a tierra común a todos los dispositivos. El Sistema comprenderá en el ingreso mediante un teclado de las temperaturas máximas y mínimos con el cual va a trabajar el sistema propuesto en un ambiente de trabajo. Mientras se toman las lecturas mediante el sensor inteligente DS1820 el microcontrolador se encarga de comparar la temperatura obtenida con los rangos previamente ingresados por el usuario inicialmente, y mostrados en la LCD, posteriormente de haber sido realizado este proceso se determina si la temperatura no ha excedido los parámetros que el usuario ingreso, si ha sobrepasado el valor máximo se temperatura permitido se activa un ventilador colocado anexo al sistema para enfriar el dispositivo medido y así procurar mantener al sistema de una manera estable.

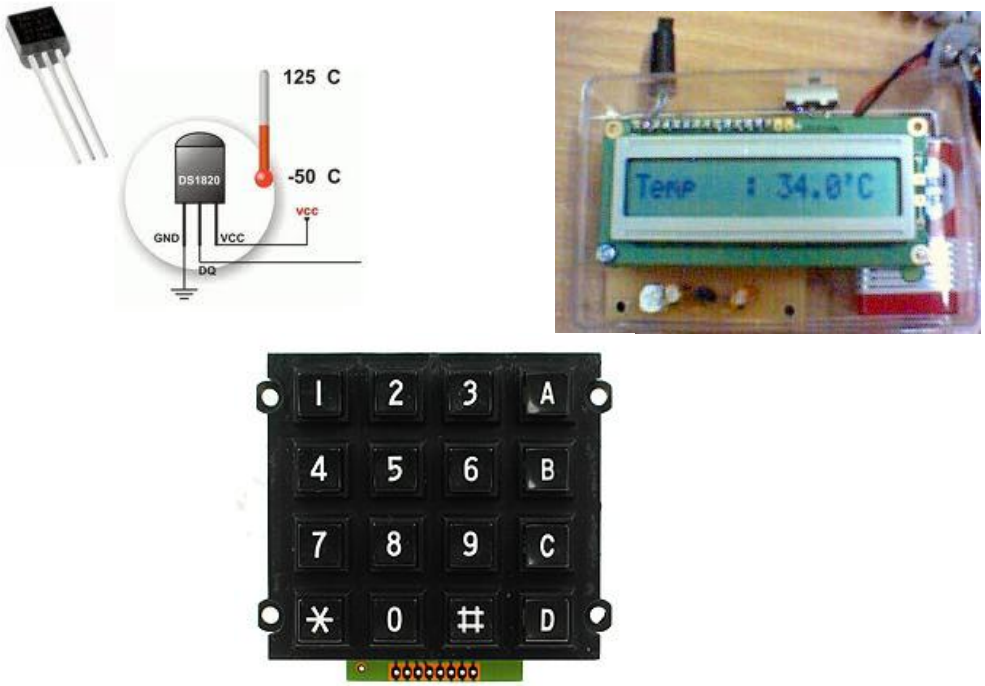


FIGURA 1.1.- Descripción del proyecto

El Sistema es programado mediando el **MikroC pro for pic,** este software programa los pic o microcontrolares en lenguaje c, proporcionando de una herramienta al usuario para determinar, proveer y realizar distintas tareas o requerimientos de acuerdo a las exigencias del proyecto.

Es una nueva tecnología aplicada al desarrollo e innovación de sistemas de monitoreo constante de diferentes dispositivos electrónicos e industriales, mediante alarmas previamente establecidas en el sistema y cumpliendo con las debidas especificaciones de las tarjetas o elementos electrónicos, que dependen de temperaturas moderadas para evitar el calentamiento excesivo y evitar que los equipos sufran daños internos que provocan pérdidas de dinero al usuario.

1.3. Aplicaciones

La aplicación para el Sistema de Alarmas de Temperatura usando el Sensor de Temperatura DS1820 es básicamente controlar la temperatura del ambiente en el cual se encuentre instalado equipos tales como servidores, equipos de comunicaciones, industriales etc. El sensor de temperatura (DS1820), es un dispositivo ideal para controlar y monitorear la temperatura de recintos amplios en los que simplemente con la instalación de un bus de un conductor se vincule todos los DS1820 dispuestos en los puntos en que se desee controlar la temperatura.

Periódicamente una computadora principal interrogara en caso de tener más de un DS1820 y esa información la almacenará para comparar con los valores establecidos previamente por el usuario para que en el caso tomar las medidas preventivas y las acciones de control necesarias para contrarrestar algún peligro en dispositivos como equipos de refrigeración, servidores, conexión de calefactores etc.

1.4. Proyectos similares

1.4.1 SENSOR DE TEMPERATURA CON EL INTEGRADO LM35

Descripción:

El sensor de temperatura LM35 tiene una precisión calibrada de 1ºC y un rango de captura de -55º a +150ºC.

El sensor presenta diferentes encapsulados pero el más común es el to-92, siendo similar a un transistor simple de tres pines, dos de ellos para alimentarlo y el tercero entrega un valor de tensión proporcional a la temperatura medida por el dispositivo.

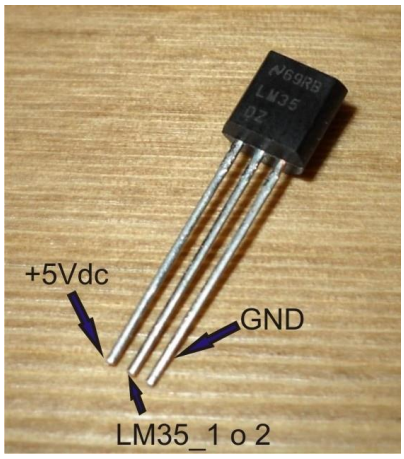


FIGURA 1.2.- Encapsulado LM35 to-92

La salida es lineal y equivale a 10mV/ºC por lo tanto:

y +1500mV = 150ºC

y +250mV = 25ºC

y -550mV = -55ºC

Funcionamiento:

Al medir la temperatura se utiliza un voltímetro calibrado para traducir el rango de la temperatura en voltaje. El LM35 funciona en el rango de alimentación comprendido entre 4 y 30 voltios. Podemos conectarlo a un conversor Analógico/Digital y tratar la medida digitalmente, almacenarla o procesarla con un Microontrolador o similar.

Aplicaciones:

El sensor de temperatura puede ser usado para compensar un dispositivo de medida sensible a la temperatura ambiente, refrigerar partes delicadas de un robot o para monitorear temperaturas en el transcurso de un trayecto de exploración.

Ejemplo de circuito de prueba circuito de prueba:

El siguiente montaje es un medidor de temperatura de 4 canales usando un PIC16F876 y un LCD para mostrar los datos. Para una medida real con el conversor Analógico/Digital se ha dispuesto un generador de tensión de referencia ajustable modelo LM336 externo al pic, con lo cual la medida de temperatura es exacta. El rango de medidas que soporta este montaje es solo de temperaturas positivas y abarca desde 0º a +150ºC.

FIGURA 1.3.- En el LCD se muestran los 4 canales T1=RA0, T2=RA1, T3=RA2, T4=RA5.

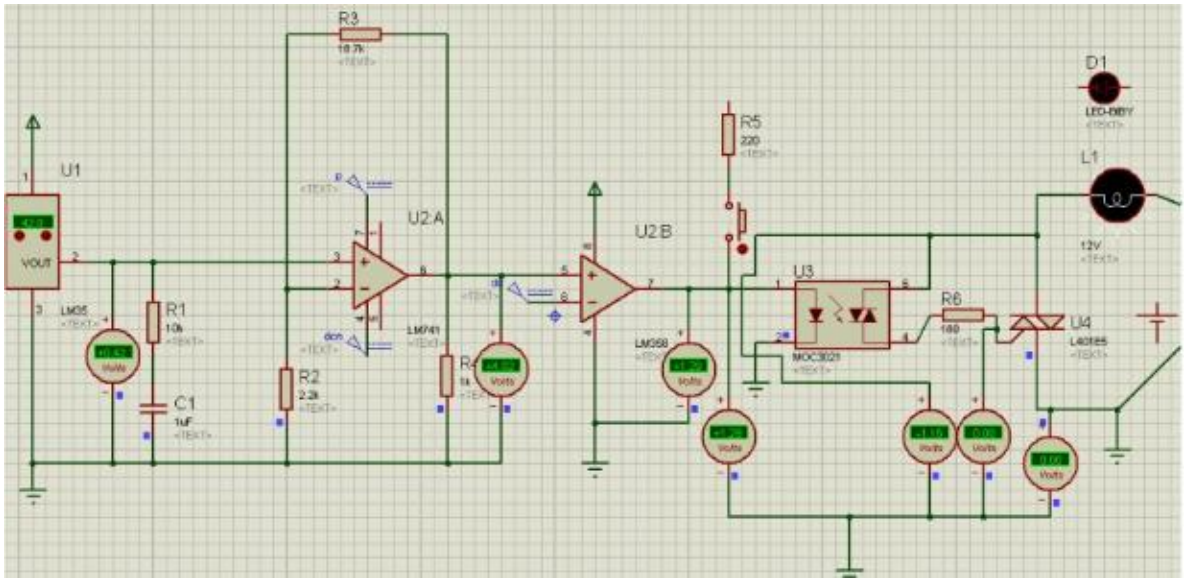


FIGURA. 1.4.- Circuito armado en el simulador Proteus

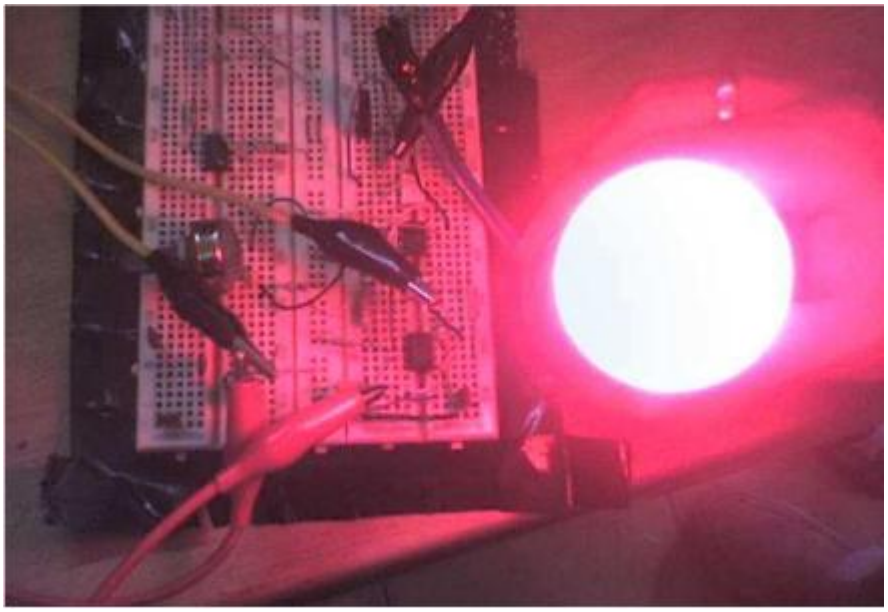


FIGURA. 1.5.- Prueba de temperatura con un cautin.

En la industria se utiliza este circuito para evitar el calentamiento de distintos dispositivos , ayudar a configurar alarmas, y llegar a corregir errores en un sistema de medición de temperatura inteligente mediante el cual se toman decisiones según los datos obtenidos dependiendo de alarmas y requerimientos previamente configurados.

CAPÍTULO 2

2. FUNDAMENTO TEÓRICO

2.1. Requerimientos para aplicación del Proyecto

El proyecto se lo puede dividir en dos partes esenciales: Software y Hardware. El software para la programación de temperaturas con sus alarmas utilizando el sensor inteligente ds1820 en comunicación one-wire es el MikroC Pro for PIC y para la simulación del sistema se usa la herramienta Proteus versión 7.7 Service Pack 2.

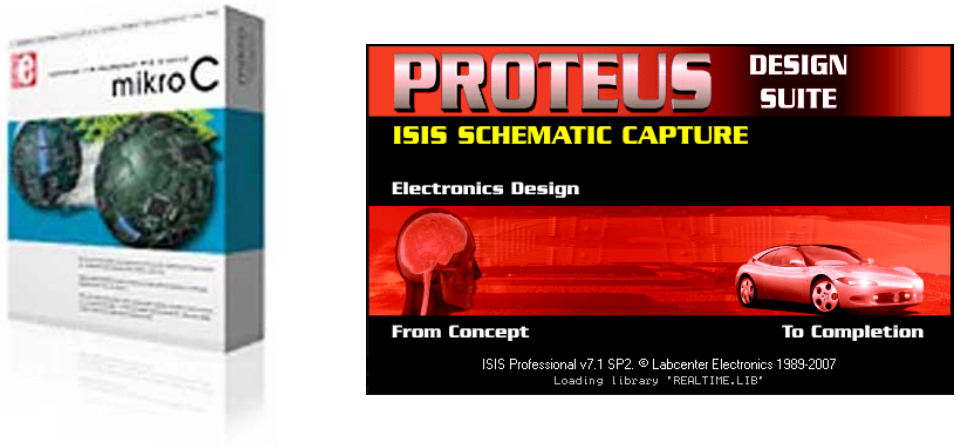


FIGURA 2.1.- Requerimientos del Proyecto (Software)

En el Hardware estamos usando el sensor de temperatura **DS1820** junto con el micro-controlador 16F887 el cual nos permite el ingreso de los valores mínimos y máximos de temperatura por el teclado matricial 4x4 y mostrar en la pantalla LCD 2x16 para el monitoreo del sistema, al final si la temperatura del ambiente sobrepasa la establecida la máxima se enciende un ventilador para retornar la estabilidad al sistema.

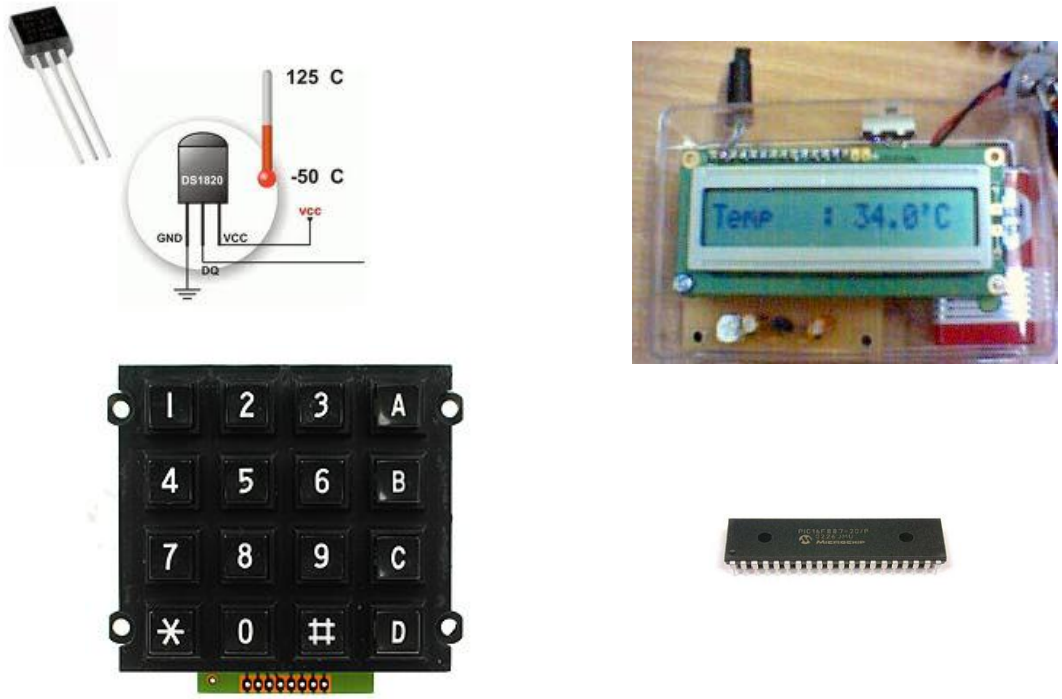


FIGURA 2.2.- Requerimientos del Proyecto (Hardware)

2.2. Herramientas de software

2.2.1. MIKROC PRO for PIC

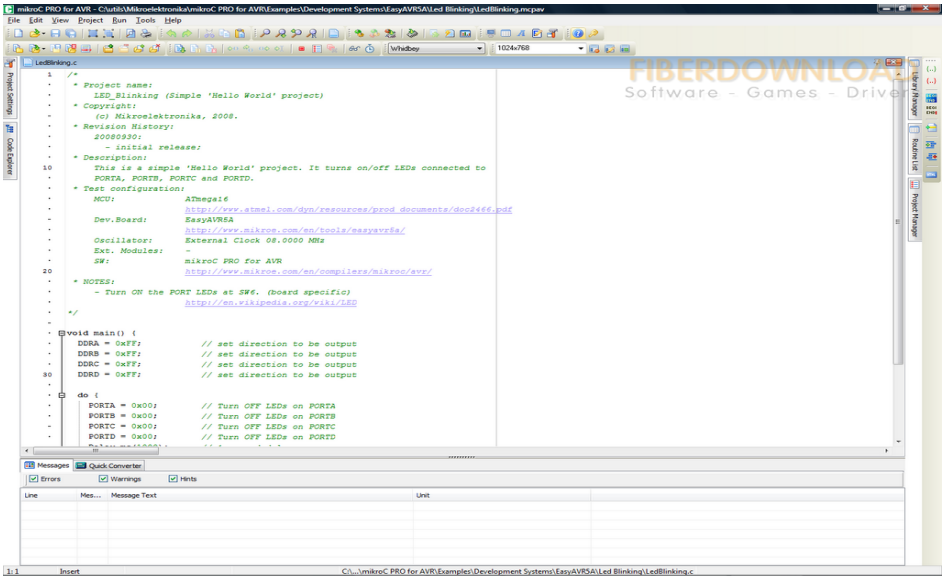


FIGURA 2.3.- Entorno de MIKROC PRO for Pic

El ya conocido MikroC Pro for Pic, perteneciente a MIKROELECTRONICA, muy formal y estructurado con un entorno de trabajo más elaborado, en este lenguaje podemos destacar el uso de la librería del protocolo one-wire para nuestro proyecto.

mikroC PRO for PIC organiza aplicaciones en los proyectos que consisten en un solo fichero de proyecto (fichero con extensión***.mcppi***) o en uno o más ficheros fuentes (ficheros con extensión ***.c***). Los ficheros fuentes son denominados cabeceras en lenguaje de programación *mikroC*. El compilador *mikroC PRO for PIC* permite manejar varios proyectos a la vez. Los ficheros fuentes se pueden compilar sólo si forman parte del proyecto.

Un fichero de proyecto contiene lo siguiente:

Nombre del proyecto y la descripción opcional;

Dispositivo destino (tipo de microcontrolador) utilizado;

Frecuencia de reloj del microcontrolador;

Lista de ficheros fuentes de proyecto;

Ficheros binarios (***.mcl***); y

Otros ficheros.

En esta guía referente vamos a crear un nuevo proyecto, escribir código, compilarlo en *mikroC PRO for PIC* y comprobar los resultados. El propósito de este ejemplo es hacer los diodos LED parpadear en el puerto PORTC del microcontrolador, por lo que será fácil comprobarlo.

2.2.1.1. ¿Qué es un Compilador?

Un **compilador** es un programa informático que traduce un programa escrito en un lenguaje de programación a otro lenguaje de programación, generando un programa equivalente que la máquina será capaz de interpretar. Usualmente el segundo lenguaje es lenguaje de máquina, pero también puede ser simplemente texto. Este proceso de traducción se conoce como compilación.

Un compilador es un programa que permite traducir el código fuente de un programa en lenguaje de alto nivel, a otro lenguaje de nivel inferior (típicamente lenguaje de máquina). De esta manera un programador puede diseñar un

programa en un lenguaje mucho más cercano a cómo piensa un ser humano, para luego *compilarlo* a un programa más manejable por una computadora.

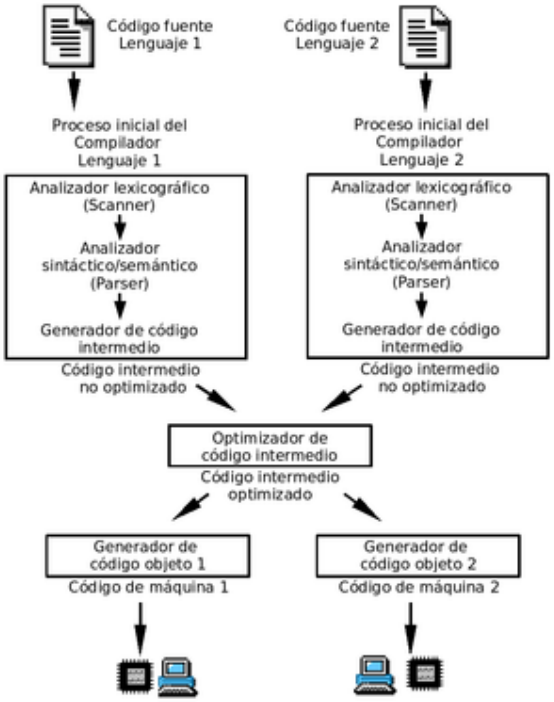


FIGURA 2.4.- Diagrama a bloques de la operación de un buen compilador

2.2.2. PROTEUS

PROTEUS es una herramienta software que permite la simulación de circuitos electrónicos con microcontroladores. Sus reconocidas prestaciones lo han convertido en el más popular simulador software para microcontroladores PIC.

Esta herramienta permite simular circuitos electrónicos complejos integrando inclusive desarrollos realizados con microcontroladores de varios tipos, en una herramienta de alto desempeño con unas capacidades graficas impresionantes.

Presenta una filosofía de trabajo semejante al SPICE, arrastrando componentes de una barra e incrustándolos en la aplicación, es muy sencillo de manejar y presenta una interfaz gráfica amigable para un mejor manejo de las herramientas proporcionadas por el Proteus.

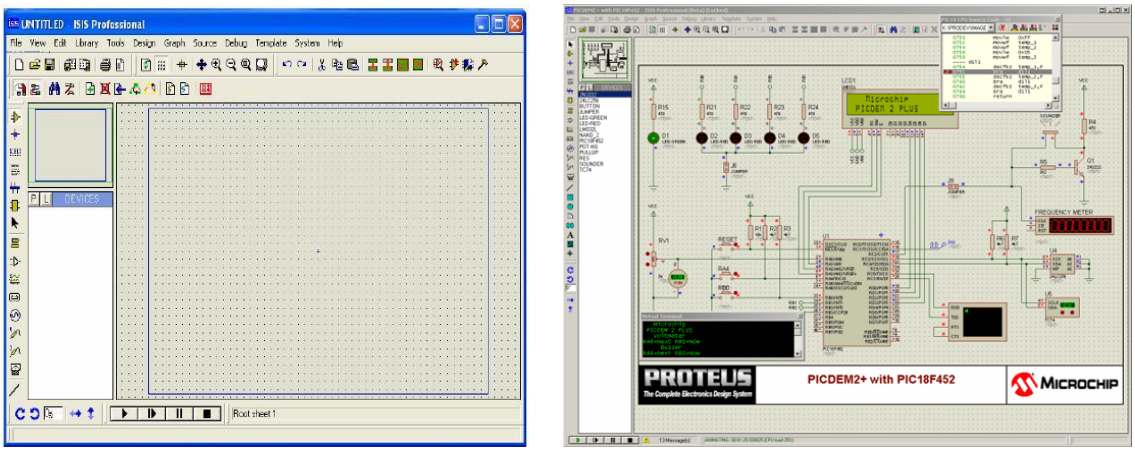


FIGURA 2.5.- Interfaz Gráfica Proteus

2.2.2.1. ARES o **A**dvanced **R**outing and **E**diting **S**oftware (*Software de Edición y Ruteo Avanzado*); es la herramienta de enrutado ,ubicación y edición de componentes, se utiliza para la fabricación de placas de circuito impreso, permitiendo editar generalmente, las capas superficial (Top Copper), y de soldadura (Bottom Copper).

2.2.2.1.1. *Forma Manual*

Ejecutando ARES directamente, y ubicando cada componente en el circuito. Tener cuidado al DRC, Design Rules Checker (Verificador de Reglas de Diseño)

2.2.2.1.2. *Forma Automática*

El propio programa puede trazar las pistas, si se guarda previamente el circuito en ISIS, y haciendo clic en el ícono de ARES, en el programa, el programa compone la Netlist

2.2.2.1.3. Método 1 (Autorouter)

1. Poner SOLO los componentes en la board
2. Especificar el área de la placa (con un rectángulo, tipo "Board Edge")
3. Hacer clic en "Autorouter", en la barra de botones superior
    1. Editar la estrategia de ruteo en "Edit Strategies"
4. Hacer clic en "OK"

2.2.2.1.4. Método 2 (Electra Autorouter)

Utilizando el módulo Electra (Electra Auto Router), el cual, una vez colocados los componentes trazará automáticamente las pistas realizando varias pasadas para optimizar el resultado.

Con Ares además se puede tener una visualización en 3D del PCB que se ha diseñado, al haber terminado de realizar la ubicación de piezas, capas y ruteo, con la herramienta "3D Visualization", en el menú output, la cual se puede demorar, solo haciendo los trazos un periodo de tiempo un poco más largo que el de los componentes, los cuales salen al empezar la visualización en 3D.

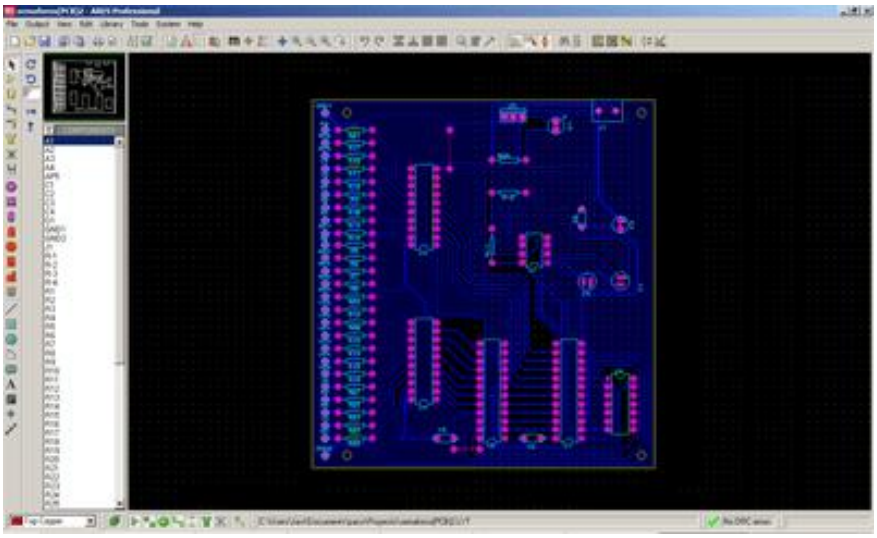


FIGURA 2.6.- Interfaz Gráfica ARES

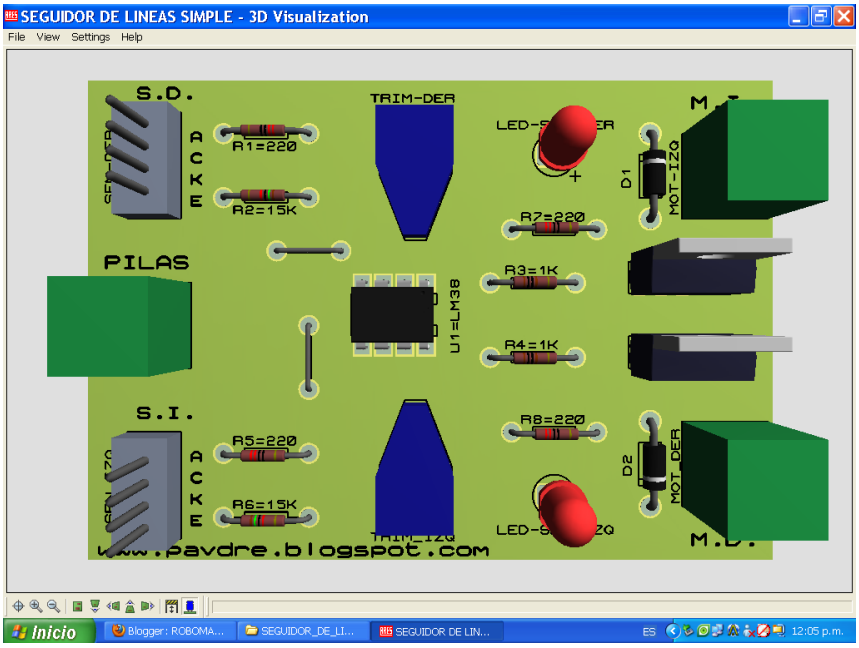


FIGURA 2.7.- Vista 3D Pistas en Ares

2.3. Herramientas de hardware

2.3.1. Sensor de Temperatura Inteligente DS1820

Es un dispositivo en encapsulado "tipo transistor" PR35 o "tipo integrado" SSOP (en lugar del tradicional botón) y permite medir temperaturas desde 55°C to +125°C en incrementos de 0.5°C con 9 bits de precisión en un tiempo típico de 200 ms.

El sistema opera sobre la ya tradicional interfaz de un conductor (1 wire bus), no siendo imprescindible alimentación externa y teniendo un número de serie en ROM de 64 bits, lo que permite tener un conjunto de termómetros

conectados por medio del bus de un conductor y ser interrogados de a uno por su número de serie, como si se trataran de botones.

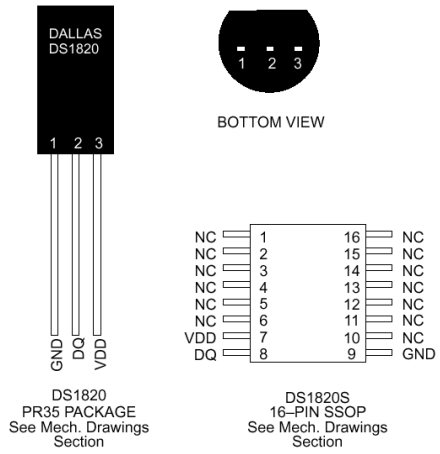


FIGURA 2.8.- Presentación Circuital

2.3.1.1. PRINCIPIO DE FUNCIONAMIENTO.

El DS1820, tiene, además del número de serie y de la interfaz de un conductor, un circuito medidor de temperatura y dos registros que pueden emplearse como alarmas de máxima y de mínima temperatura.

Este termómetro se basa en un par de osciladores de frecuencia proporcional a la temperatura El oscilador de frecuencia proporcional a la alta temperatura actúa como habilitación (gate) del conteo del oscilador de frecuencia proporcional a la baja temperatura. Existe un circuito (Slope Accumulator) encargado de

compensar las alinealidades de la variación de frecuencia de los osciladores con la temperatura.

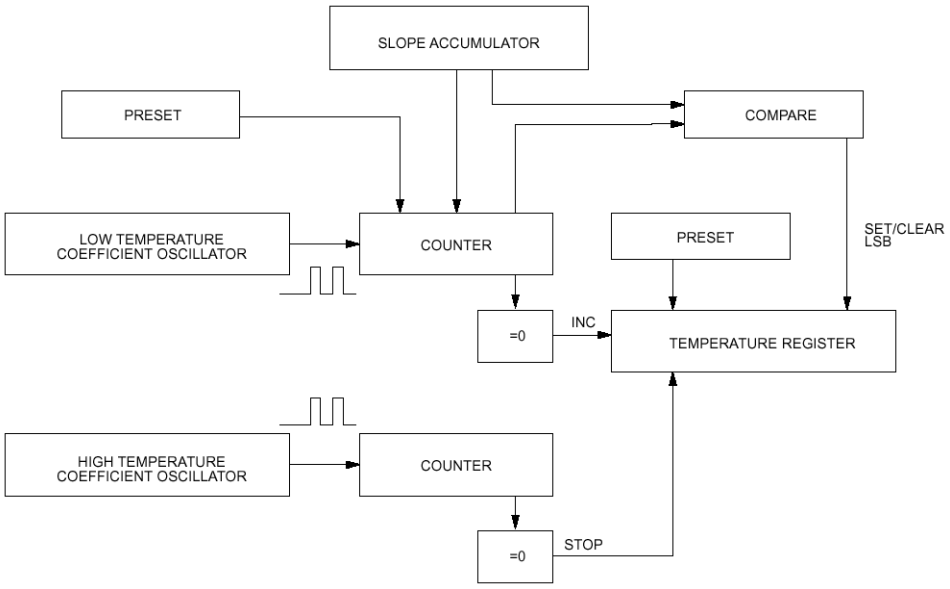


FIGURA 2.9.- Diagrama interno del DS1820

A los comandos tradicionales de los botones como: lectura de ROM, búsqueda de ROM, coincidencia de ROM, salteo de ROM, se agregan nuevos comandos por el bus de un conductor, como: Convertir temperatura, leer, copiar o escribir la memoria temporaria (scratchpad), buscar alarmas.

Estas alarmas son comparadas con el valor de temperatura medido inmediatamente de terminada la medición, es decir que el flag de alarma será actualizado después de cada medición.

DETAILED PIN DESCRIPTIONS Table 1

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

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

TABLA 2.1.- Descripción de los Pines

2.3.1.2. MEDICIÓN DE TEMPERATURA.

La temperatura se obtiene en un formato de módulo y signo de nueve bits.

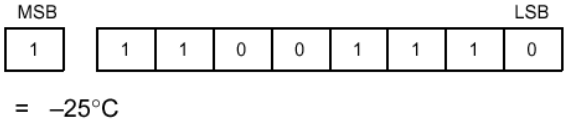


FIGURA 2.10.- Representación de una medición.

Se observa que el bit más significativo (MSB) corresponde al signo y que el bit menos significativo tiene un peso de 0.5 °C, el subsiguiente en sentido creciente 1°C, el bit 2 estará asociado a 2°C, hasta el bit 7 cuyo peso será de 64°C. En la Fig. 2.10 se ve la representación de –25°C.

Para la comparación con los valores de máxima y mínima se toman sólo los 8 bits más significativos (incluyendo al signo), descartando el 0.5°C.

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

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

TABLA 2.2.- Tabla Relación Temperatura/Datos

2.3.1.3. PROTOCOLO ONE-WIRE

1-Wire es un protocolo de comunicaciones en serie diseñado por Dallas Semiconductor. Está basado en un bus, un maestro y varios esclavos de una sola línea de datos en la que se alimentan. En principio los dispositivos de este tipo se alimentan con corrientes parásitas, aunque necesitan referencia a tierra.

Especificaciones

La línea de datos/alimentación requiere una resistencia de pull-up conectada a la alimentación y que le proporciona ésta.

Reinicio del bus

Se mantiene la señal de datos a 0 voltios durante 480 microsegundos. Se reinician todos los dispositivos conectados al bus (les retira la alimentación). Los dispositivos reiniciados indican su presencia manteniendo la señal de datos a 0 voltios durante 60 microsegundos.

Envió y recepción de datos

Para enviar un bit a 1 el maestro se lleva a 0 voltios la línea de datos durante 1-15 microsegundos. Para enviar un bit a 0 el maestro se lleva a 0 voltios la línea de datos durante 60 microsegundos.

Los dispositivos esclavos leen el bit aproximadamente a los 30 microsegundos después del flanco de bajada de cada bit.

Cuando el maestro lee los datos del dispositivo esclavo el pone 0 voltios durante 1-15 microsegundos en la línea de datos y a partir de ese momento el esclavo no hace nada (la señal pasa a vale 5 voltios) si quiere mandar un 1 lógico o mantiene la señal en 0 voltios hasta los 60 microsegundos si quiere mandar un 0 lógico.

Los datos se envían o reciben en grupos de 8 bits. Para iniciar una comunicación se reinicia el bus. El protocolo puede incluir detección de errores transmitiendo códigos de detección de errores (CRC).

Como en el bus puede haber muchos dispositivos el protocolo incluye el direccionamiento de los mismos empleando un código único de 64 bits de los cuales el byte más significativo indica el tipo de dispositivo, y el último es un código de detección de errores (CRC) de 8 bits.

Los comandos que pueden interpretar los dispositivos esclavos dependerán de estos. Para encontrar los dispositivos presentes en el bus el maestro puede enviar un comando de enumeración que responderán todos los dispositivos.

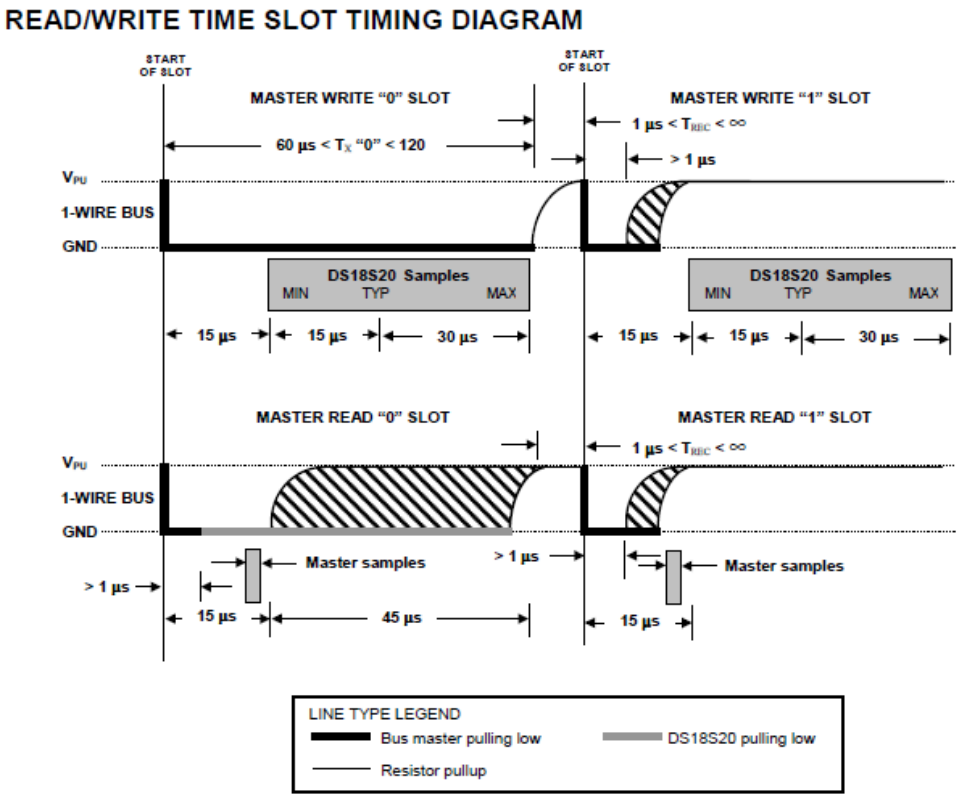


FIGURA 2.11.- Diagrama de Tiempo para Lectura/Escritura

INITIALIZATION TIMING Figure 10

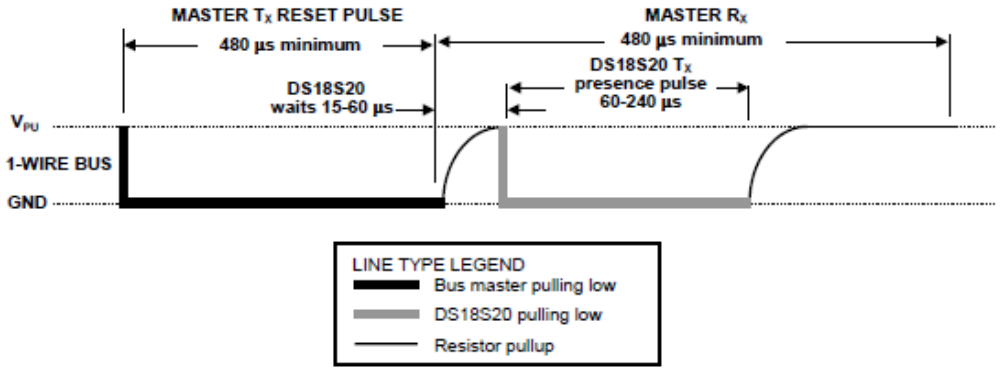


FIGURA 2.12.- Inicialización de Tiempo

2.3.2. Microcontrolador 16F887

2.3.2.1. Arquitectura Interna

- Generalidades
- Memoria de Programa
- Memoria de Datos
- EEPROM
- Registros de Control (SFR)

Generalidades : Características Generales

- Arquitectura RISC

  - 35 instrucciones

  - Instrucciones de un solo ciclo excepto las de salto

- Frecuencia de operación de 0-20MHz (DC-200ns)
- Manejo de Interrupciones

- 8 niveles de Pila (Stack)
- Oscilador interno de precisión calibrado en fábrica al 1% de error
- Frecuencias seleccionable por software entre 8MHz-31KHz
- Voltaje de alimentación entre 2.0-5.5V

  - Consumo de 220uA(2V, 4MHz), 11uA (2.0V, 32KHz), 50nA (en modo de stand-by)
- Modo SLEEP para ahorro de energía
- BOR(Brown-out Reset) reset por baja de voltaje con opción de control por software
- 35 pines de entrada/salida
  - Corriente de suministro/drenaje suficiente para manejar LED directamente
  - Resistores de pull-up programables individualmente
  - Interrupción por cambio en pin
- 8K de memoria FLASH. EL chip puede reprogramarse hasta 100.000 veces
- Opción de programación en circuito (In-circuit serial Programing.

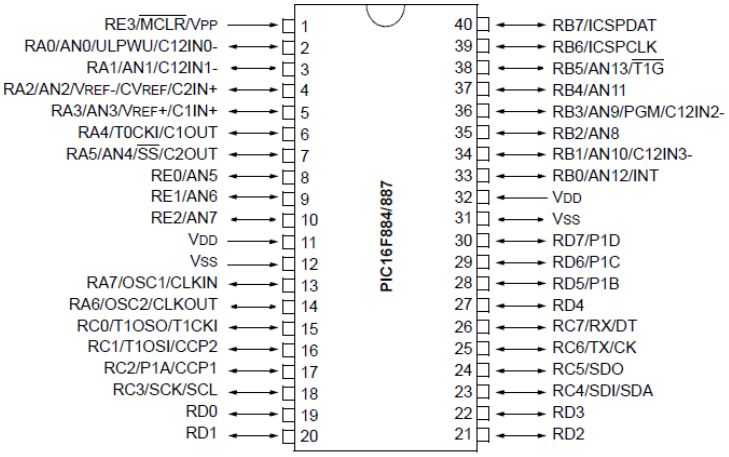


FIGURA 2.13.- Pic 16F887

Descripción de los Pines – Pines Funcionales

- Como puede observarse en el pin RA3: RA3/AN3/Vref+/C1IN+
- Significa que tiene varias funcionalidades:

  RA3 Port A entrada/salida digital 3

  AN3 Entrada analógica 3
- Vref+ Referencia de voltage positiva
- C1IN+ Entrada positiva de comparador C1
- Estas caractrísticas no pueden usarse en forma simultánea pero pueden cambiarse durante la operación .

Name	Number (DIP 40)	Function	Description
RE3/MCLR/Vpp	1	RE3	General purpose input Port E
		MCLR	Reset pin. Low logic level on this pin resets microcontroller.
		Vpp	Programming voltage
RA0/AN0/ULPWU/C12IN0-	2	RA0	General purpose I/O port A
		AN0	A/D Channel 0 input
		ULPWU	Stand-by mode deactivation input
		C12IN0-	Comparator C1 or C2 negative input
RA1/AN1/C12IN1-	3	RA1	General purpose I/O port A
		AN1	A/D Channel 1
		C12IN1-	Comparator C1 or C2 negative input
RA2/AN2/Vref-/CVref/C2IN+	4	RA2	General purpose I/O port A
		AN2	A/D Channel 2
		Vref-	A/D Negative Voltage Reference input
		CVref	Comparator Voltage Reference Output
		C2IN+	Comparator C2 Positive Input
RA3/AN3/Vref+/C1IN+	5	RA3	General purpose I/O port A
		AN3	A/D Channel 3
		Vref+	A/D Positive Voltage Reference Input
		C1IN+	Comparator C1 Positive Input
RA4/T0CKI/C1OUT	6	RA4	General purpose I/O port A
		T0CKI	Timer T0 Clock Input
		C1OUT	Comparator C1 Output
RA5/AN4/SS/C2OUT	7	RA5	General purpose I/O port A
		AN4	A/D Channel 4
		SS	SPI module Input (*Slave Select*)
		C2OUT	Comparator C2 Output
RE0/AN5	8	RE0	General purpose I/O port E
		AN5	A/D Channel 5
RE1/AN6	9	RE1	General purpose I/O port E
		AN6	A/D Channel 6
RE2/AN7	10	RE2	General purpose I/O port E
		AN7	A/D Channel 7
Vdd	11	+	Positive supply
Vss	12	-	Ground (GND)

Name	Number (DIP 40)	Function	Description
RA7/OSC1/CLKIN	13	RA7	General purpose I/O port A
		OSC1	Crystal Oscillator Input
		CLKIN	External Clock Input
RA6/OSC2/CLKOUT	14	OSC2	Crystal Oscillator Output
		CLKO	Fosc/4 Output
		RA6	General purpose I/O port A
RC0/T1OSO/T1CKI	15	RC0	General purpose I/O port C
		T1OSO	Timer T1 Oscillator Output
		T1CKI	Timer T1 Clock Input
RC1/T1OSO/T1CKI	16	RC1	General purpose I/O port C
		T1OSI	Timer T1 Oscillator Input
		CCP2	CCP1 and PWM1 module I/O
RC2/P1A/CCP1	17	RC2	General purpose I/O port C
		P1A	PWM Module Output
		CCP1	CCP1 and PWM1 module I/O
RC3/SCK/SCL	18	RC3	General purpose I/O port C
		SCK	MSSP module Clock I/O in SPI mode
		SCL	MSSP module Clock I/O in I^2C mode
RD0	19	RD0	General purpose I/O port D
RD1	20	RD1	General purpose I/O port D
RD2	21	RD2	General purpose I/O port D
RD3	22	RD3	General purpose I/O port D
RC4/SDI/SDA	23	RC4	General purpose I/O port A
		SDI	MSSP module *Data* input in SPI mode
		SDA	MSSP module *Data* I/O in I^2C mode
RC5/SDO	24	RC5	General purpose I/O port C
		SDO	MSSP module *Data* output in SPI mode
RC6/TX/CK	25	RC6	General purpose I/O port C
		TX	USART Asynchronous Output
		CK	USART Synchronous Clock
RC7/RX/DT	26	RC7	General purpose I/O port C
		RX	USART Asynchronous Input
		DT	USART Synchronous Data

Name	Number (DIP 40)	Function	Description
RD4	27	RD4	General purpose I/O port D
RD5/P1B	28	RD5	General purpose I/O port D
		P1B	PWM Output
RD6/P1C	29	RD6	General purpose I/O port D
		P1C	PWM Output
RD7/P1D	30	RD7	General purpose I/O port D
		P1D	PWM Output
Vss	31	-	Ground (GND)
Vdd	32	+	Positive Supply
RB0/AN12/INT	33	RB0	General purpose I/O port B
		AN12	A/D Channel 12
		INT	External Interrupt
RB1/AN10/C12INT3-	34	RB1	General purpose I/O port B
		AN10	A/D Channel 10
		C12INT3-	Comparator C1 or C2 Negative Input
RB2/AN8	35	RB2	General purpose I/O port B
		AN8	A/D Channel 8
RB3/AN9/PGM/C12IN2-	36	RB3	General purpose I/O port B
		AN9	A/D Channel 9
		PGM	Programming enable pin
		C12IN2-	Comparator C1 or C2 Negative Input
RB4/AN11	37	RB4	General purpose I/O port B
		AN11	A/D Channel 11
RB5/AN13/T1G	38	RB5	General purpose I/O port B
		AN13	A/D Channel 13
		T1G	Timer T1 External Input
RB6/ICSPCLK	39	RB6	General purpose I/O port B
		ICSPCLK	Serial programming Clock
RB7/ICSPDAT	40	RB7	General purpose I/O port B
		ICSPDAT	Programming enable pin

TABLA 2.3.- Descripción de Pines del 16F887

Unidad Central de Procesamiento CPU

El CPU reconoce 35 instrucciones a diferencia de otros microcontroladores con más de 200 instrucciones.

El tiempo de ejecución de cada instrucción es de 4 ciclos de reloj, excepto las de salto que demoran dos. Esto significa que si la velocidad de operación del microcontrolador es de 20MHz el tiempo de ejecución de cada instrucción es de 200ns. Es decir que el programa ejecutará 5'000.000 de instrucciones por segundo.

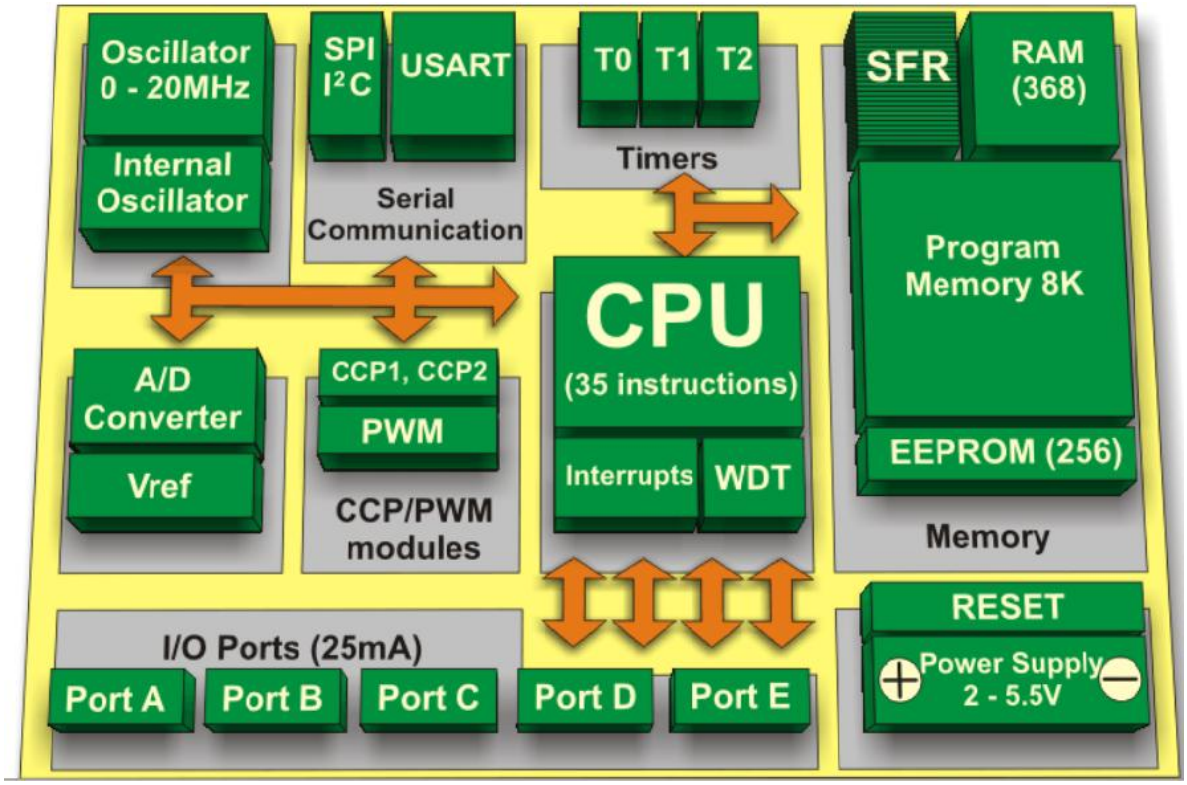


FIGURA 2.14.- Diagrama de Bloques del 16F88

El microcontrolador PIC16F887 tiene tres tipos de memoria:

- ROM
- RAM
- EEPROM

Bancos de Datos

La memoria RAM se divide en cuatro bancos de datos. Antes de acceder a un registro debe seleccionarse el banco en donde está ubicado dicho registro. Los BITS 5 y 6 del registro STATUS son utilizados para la selección de bancos. Los registros SFR más comunes tienen la misma dirección en todos los bancos permitiendo su fácil acceso.

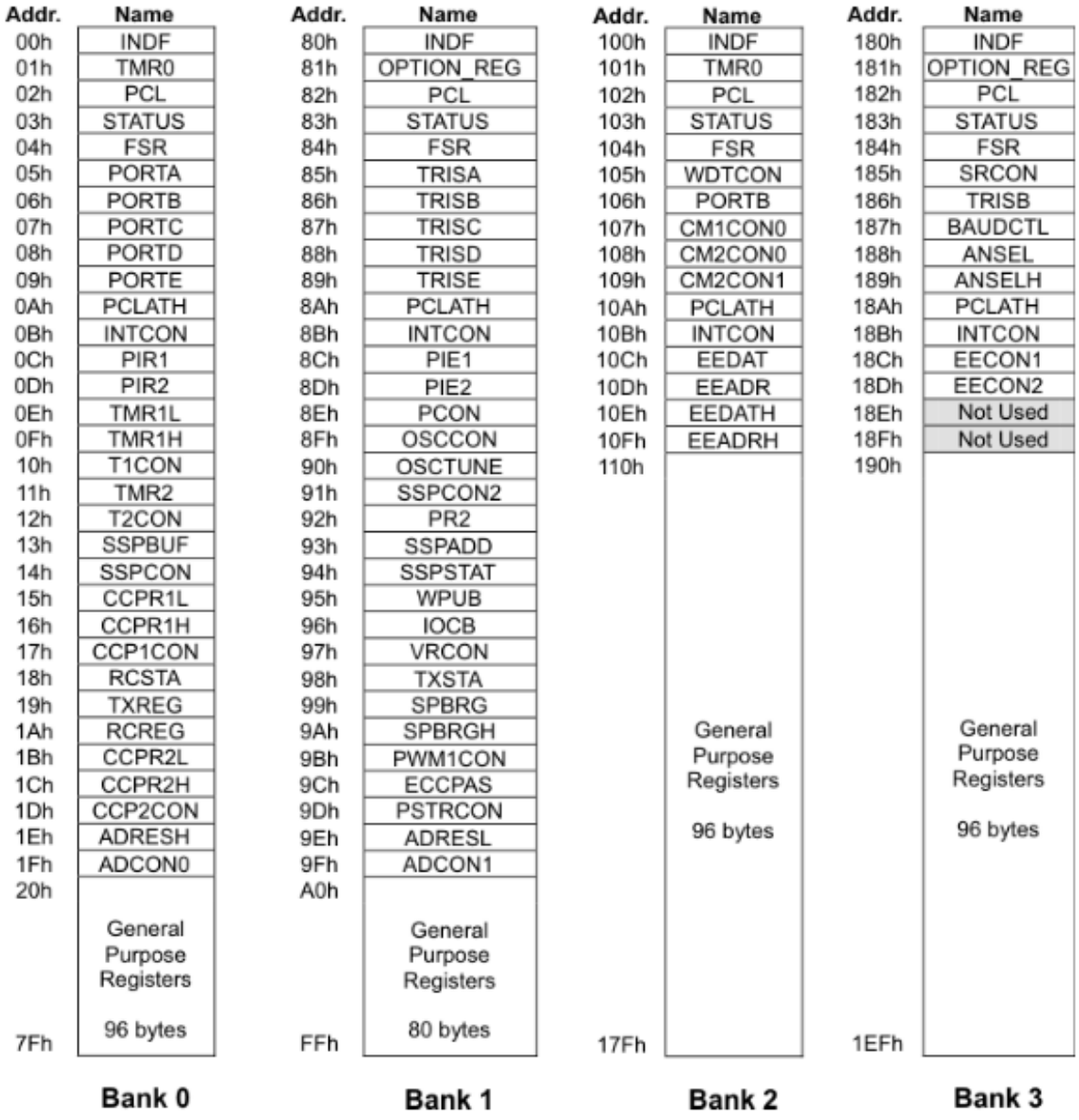


FIGURA 2.15.- Banco de Memoria de Datos del 16F887

Uso de interruptores, botoneras y teclados

Los pines del PORTB son comúnmente usados para detectar pulsadores de teclados. En el esquema presentado, los pines RB0-RB3 se configuran como entradas con resistencias a tierra (Pull-Down), mientras que x, y, z (columnas) son salidas en alto. Cuando se presione un pulsador ocasiona una interrupción que define la fila que la ocasionó. Inmediatamente en el programa principal con la fila detectada se cambian x, y, z a entradas y se lee su estado para determinar la columna. Con fila y columna definidas se identifica la tecla apretada.

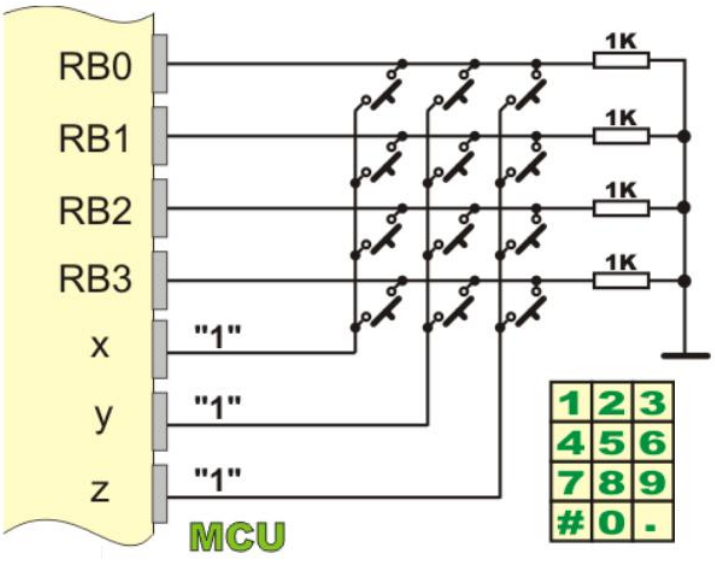


FIGURA 2.16.- Descripción y Uso de Teclado y Botoneras del 16F887

Temporizadores

Los temporizadores del PIC16F887 son tres:

- TMR0
- TMR1
- TMR2

TMR0

El TMR0 tiene una amplia gama de aplicaciones prácticas. Muy pocos programas no lo usan de alguna manera. Es muy conveniente de usarlo en programas para generar pulsos, medir la duración de pulsos externos o contar los pulsos recibidos.

El módulo TMR0 tiene las siguientes características:

- Contador/temporizador de 8-bits
- Preescalador compartido con el WDT de 8-bits
- Reloj interno programable o fuente de reloj externa
- Interrupción por desbordamiento
- Selección de flanco de reloj externo programable
- La próxima diapositiva representa al TMR0 con todos los bits que intervienen en su operación. Estos bits son almacenados en el registro OPTION_REG

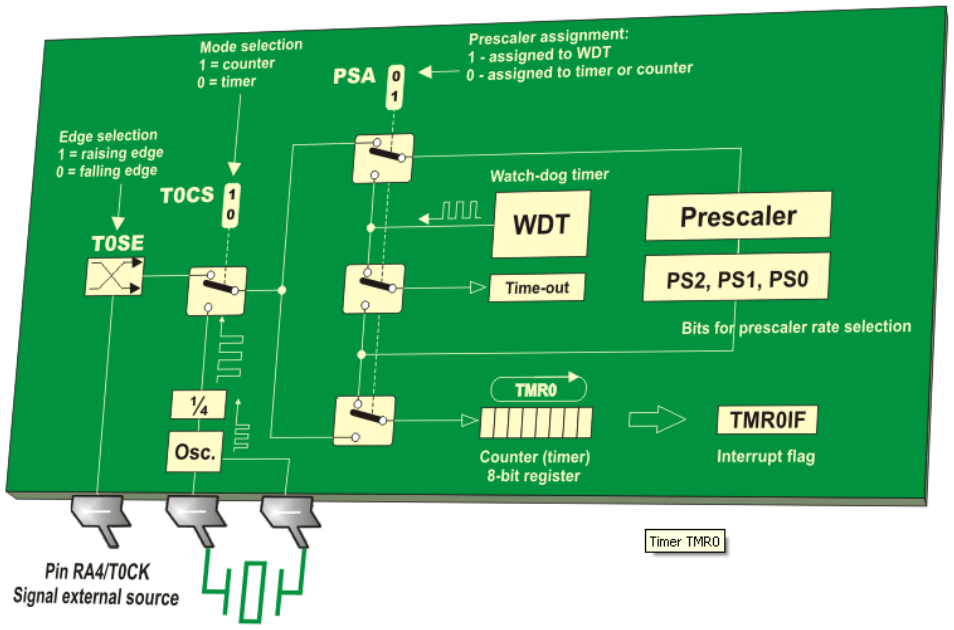


FIGURA 2.17.- Diagrama del TIMER 0

Cuando el microcontrolador entra en el modo SLEEP, el oscilador se detiene, por tanto el TMR0 no puede contar ni causar interrupciones para salir del modo SLEEP.

Cuando el preescalador está asignado al TMR0, cualquier escritura al TMR0 pondrá en cero al preescalador. En forma similar cuando el preescalador está asignado al WDT, la instrucción CLRWDT pondrá en cero tanto el WDT como el preescalador. Cuando se quiera cambiar la asignación del preescalador debe hacérselo utilizando unas pequeñas rutinas recomendadas por el fabricante, las mismas que pueden observarse en la hoja de datos del PIC 16F887.

TMR1

El timer TMR1 es un contador de 16 bits y combina dos registros el TMR1L y el TMR1H. Puede contar hasta 65.535 generar una bandera de sobrecarga y producir una interrupción (en caso de estar habilitada).

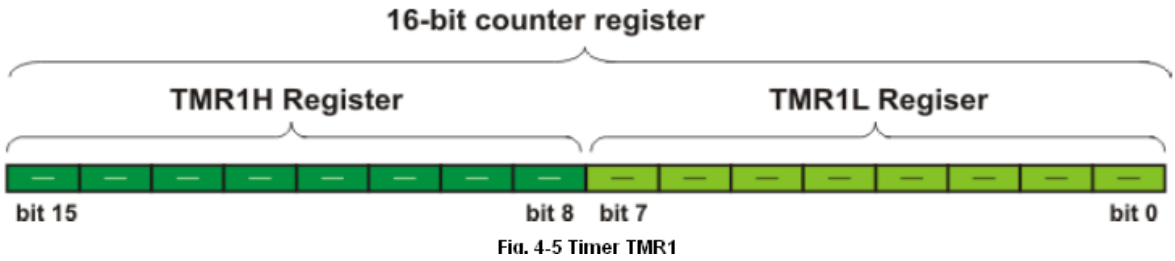


FIGURA 2.18.- TIMER 1

En forma similar al TMR0 , este registro puede leerse y escribirse en cualquier momento. En caso de overflow puede causar una interrupción (en caso de estar habilitada).

El timer TMR1 puede operar también como temporizador o como contador aunque a diferencia del TMRO cada módulo tiene funciones diferentes.

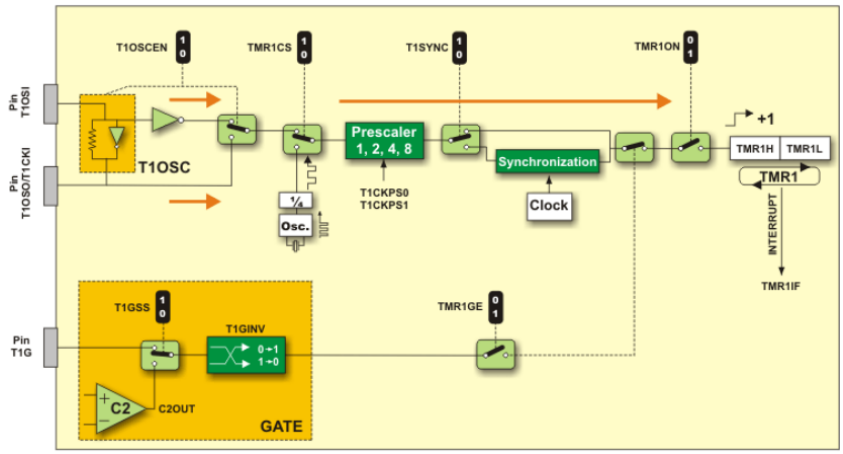


FIGURA 2.19.-Operación del TIMER 1

Uso adecuado del TMR1

Como no es posible apagar el preescalador debe escogerse adecuadamente su valor programando los bits T1CKPS1 y T1CKPS0 del registro T1CON .

La fuente de reloj debe de seleccionarse con TMR1CS (del registro T1CON) :

- TMR1CS = 0 La fuente de reloj para el TMR1 es el reloj interno (FOSC/4).
- TMR1CS =1 La fuente de reloj para el TMR1 es un reloj externo en el pin T1CKI .

Colocando T1OSCEN=1 (del registro T1CON) se habilita el TMR1 y los registros TMR1H and TMR1L se incrementan por cada entrada de reloj, haciendo T1OSCEN=0 se para el conteo.

El preescalador se borra haciendo cero o escribiendo los registros del TMR1.

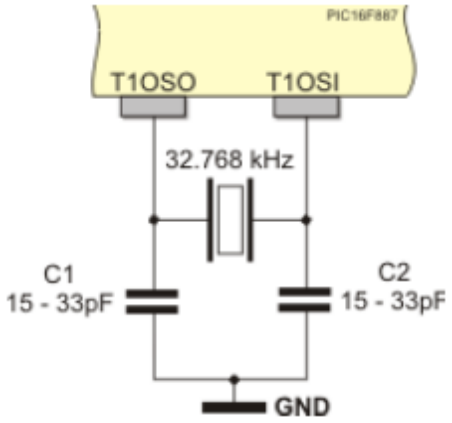
Cuando el TMR1 se desborda la bandera TMR1IF se hace uno y el TMR1 empieza a contar nuevamente desde cero.

Preescalador del TMR1

El TMR1 tiene su propio preescalador que permite divisiones del reloj de entrada para 1, 2, 4 u 8. El preescalador no puede ser leído o escrito directamente, sin embargo el preescalador es automáticamente encerado cuando se escriben los registros TMR1H o TMR1L.

Oscilador del TMR1

Los pines RC0/T1OSO and RC1/T1OSI son usados para registrar pulsos provenientes de periféricos, pero tienen una función adicional. Pin RC1 es configurado como entrada y pin RC0 es configurado como salida, del oscilador adicional de cuarzo LP, en forma simultánea. Este circuito adicional es para operar a bajas frecuencias (hasta los 200KHz), más precisamente para utilizar cristales de cuarzo de 32.768 KHz.. Tales cristales son usados en relojes de cuarzo porque es facil obtener pulsos de un segundo de duración dividiendo simplemente esta frecuencia. Dado que este oscilador no depende del reloj interno, puede operar aún en el modo SLEEP. Se lo habilita con T1OSCEN =1 (bit del registro T1CON). El usuario deberá proveer por software un retardo de tiempo (de unos cuantos milisegundos) para asegurar la operación adecuada del oscilador.



OSCILLATOR	FREQUENCY	C1	C2
LP	32 kHz	33 pF	33 pF
	100 kHz	15 pF	15 pF
	200 kHz	15 pF	15 pF

FIGURA 2.20.- Conexión del Oscilador del TIMER1

TMR2

El TMR2 es un registro de 8-bits que opera comparando su valor con PR2 hasta alcanzarlo. Posee un preescalador y un post escalador.

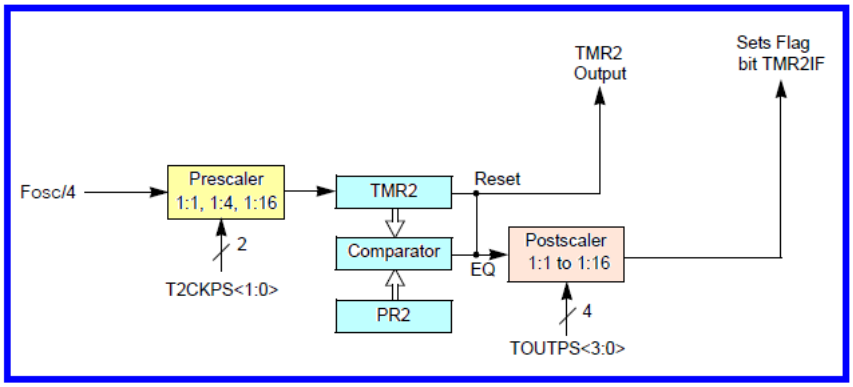


FIGURA 2.21.- Diagrama del TIMER 2

Operación del TMR2

- Su entrada de reloj es el reloj interno (FOSC/4) que pasa por el preescalador (con opciones de 1:1, 1:4 hasta 1:16). La salida del preescalador es usada para para incrementar el registro TMR2. Los valores de TMR2 y PR2 se comparan constantemente para determinar su igualdad, TMR2 incrementará desde cero hasta que su valor sea igual a PR2. Cuando ocurre la igualdad:
  - TMR2 regresa a cero en el próximo ciclo de incremento
  - Luego de generada la señal EQ, esta pasa por el postescalador del TMR2 (1:1 a 1:16) el cual divide la señal de acuerdo al valor escogido. Ej. Si el preescalador es de 1:5, cada 5 señales EQ provocarán un uno en la bandera.
- Los registros TMR2 y PR2 pueden leerse y escribirse.

- Tras un RESET, TMR2 se hace 00h y PR2 se hace FFh.
- TMR2 se habilita con TMR2ON=1 y deshabilita con TMR2ON=0.
- El preescalador del TMR2 se controla con los bits T2CKPS (registro T2CON)
- Los contadores del preescaldor y postescaldor se hacen cero cuando:
  - Se escribe sobre el TMR2
  - Cuando se escribe sobre el T2CON

- Cuando ocurre un RESET: Power-on Reset, MCLR Reset, Watchdog Timer Reset, or Brown-out Reset).

2.3.3. Pantalla LCD 2x16

La Pantalla LCD es uno de los periféricos más empleados para la presentación de mensajes, variables y casi cualquier información proveniente de un microcontrolador. Gracias a su flexibilidad, buena visibilidad y precio reducido se ha convertido en el estándar de visualización más utilizado con los microcontroladores.

LCD Alfanumérica

Pantalla en la cual, se pueden presentar caracteres y símbolos especiales en las líneas predefinidas del LCD. Su especificación viene dada como cantidad de caracteres por columna y número de filas. Por ejemplo: 2 x 16, 4 x 20.

LCD Gráfica

Pantalla en la cual, se pueden presentar caracteres, símbolos especiales y gráficos. Su especificación viene dada en píxeles. Por ejemplo 128 x 64.

TERMINALES DE CONEXIÓN:

Los terminales de conexión de las pantallas LCD de caracteres han sido estandarizados, siendo generalmente compatibles pin a pin con lo mostrado en la tabla :

Terminal	Nombre	Función	Descripción
1	Vss	Energía	Referencia 0 V. GND
2	Vdd	Energía	+5 V DC
3	Vee	Ajuste Contraste	Variable de 0 a 5 V
4	RS	Comando	Selección de Dato/Comando
5	R/W	Comando	Control de Lectura/Escritura
6	E	Comando	Habilitación
7	D0	E/S	DATO LSB
8	D1	E/S	DATO
9	D2	E/S	DATO
10	D3	E/S	DATO
11	D4	E/S	DATO
12	D5	E/S	DATO

13	D6	E/S	DATO
14	D7	E/S	DATO MSB

TABLA 2.4.- Tabla de Descripción de Terminales de Conexión del LCD

Esquema de conexiones:

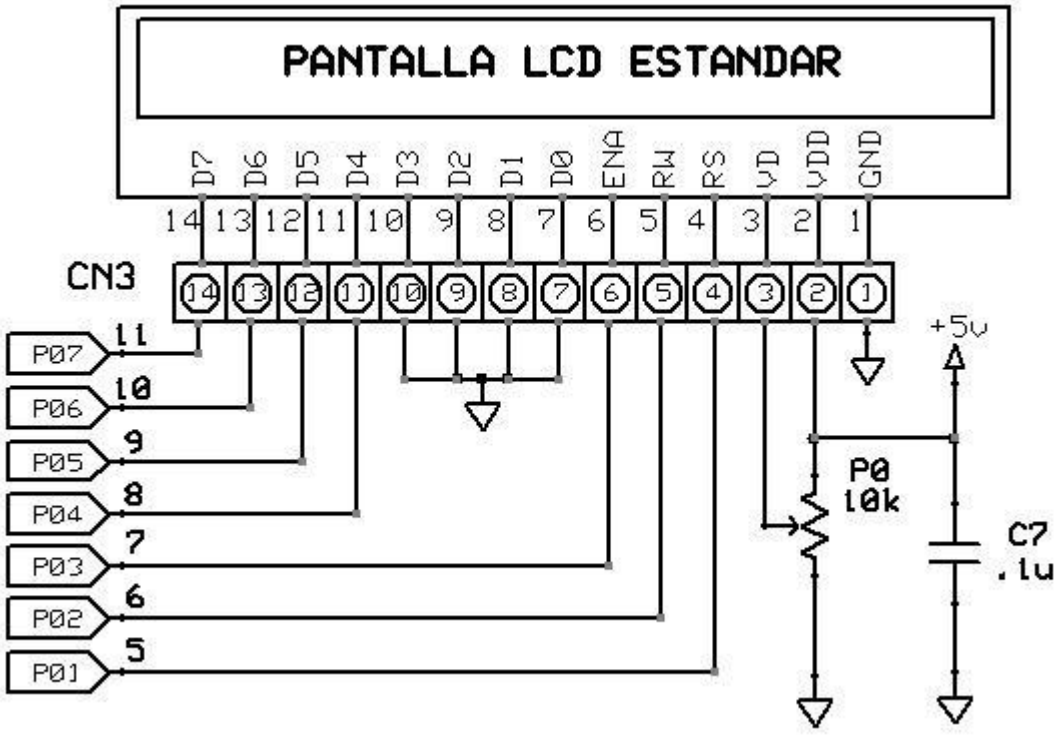


FIGURA 2.22 .- Esquema de conexiones de la pantalla LCD.

CAPÍTULO 3

3. DISEÑO E IMPLEMENTACIÓN DEL PROYECTO

A continuación se detalla el proceso del diseño y la implementación del proyecto. Nuestra prueba inicial nos ayudó a entender el funcionamiento de las funciones del sensor de temperatura con el protocolo ONE-WIRE para obtener la lectura del ambiente.

Ya con las bases del proyecto diseñamos el diagrama de bloques, donde el PIC 16F887 es el controlador el cual se encarga se obtener los datos de la temperatura censados por el DS1820 para visualizarlo en la pantalla LCD. Se desarrolló el algoritmo del microcontrolador, con las funciones principales que se muestran en el diagrama de flojo y la programación principal.

Antes de desarrollar las funciones del programa principal se estableció las funciones del sensor DS1820 con el protocolo one-wire a usar con la finalidad de estructurar el sistema de lectura entre el microcontrolador y el sensor.

3.1. Prueba inicial

Como primer paso realizamos pruebas con el sensor de temperatura DS1820 con el código de prueba basado en mikroC pro for pic, en conjunto con la pantalla de visualicación LCD y el microcontrolador 16F887.

El programa lleva por nombre One-wire y realiza la obtención de la temperatura del ambiente y mostrando por la pantalla LCD, luego este podrá ser manipulado para la obtención del proyecto final.

3.1.1. Código de prueba en Mikro C pro por pic

```
// LCD module connections
    sbit LCD_RS at RB4_bit;
    sbit LCD_EN at RB5_bit;
    sbit LCD_D4 at RB0_bit;
    sbit LCD_D5 at RB1_bit;
    sbit LCD_D6 at RB2_bit;
    sbit LCD_D7 at RB3_bit;

    sbit LCD_RS_Direction at TRISB4_bit;
    sbit LCD_EN_Direction at TRISB5_bit;
    sbit LCD_D4_Direction at TRISB0_bit;
    sbit LCD_D5_Direction at TRISB1_bit;
    sbit LCD_D6_Direction at TRISB2_bit;
    sbit LCD_D7_Direction at TRISB3_bit;
// End LCD module connections
```

```
// Set TEMP_RESOLUTION to the corresponding resolution of used DS18x20 sensor:
// 18S20: 9  (default setting; can be 9,10,11,or 12)
// 18B20: 12

        const unsigned short TEMP_RESOLUTION = 9;

        char *text = "000.0000";
        unsigned temp;

        void Display_Temperature(unsigned int temp2write) {
         const unsigned short RES_SHIFT = TEMP_RESOLUTION - 8;
        char temp_whole;
        unsigned int temp_fraction;

 // check if temperature is negative
          if (temp2write & 0x8000) {
                    text[0] = '-';
                    temp2write = ~temp2write + 1;
         }

 // extract temp_whole
          temp_whole = temp2write >> RES_SHIFT;

 // convert temp_whole to characters
          if (temp_whole/100)
                   text[0] = temp_whole/100  + 48;
        else
                    text[0] = '0';
```

```
                text[1] = (temp_whole/10)%10 + 48;          // Extract tens digit
                text[2] =  temp_whole%10     + 48;          // Extract ones digit

// extract temp_fraction and convert it to unsigned int
                 temp_fraction  = temp2write << (4-RES_SHIFT);
                temp_fraction &= 0x000F;
                temp_fraction *= 625;

// convert temp_fraction to characters
                  text[4] =  temp_fraction/1000    + 48;
                 // Extract thousands digit
                text[5] = (temp_fraction/100)%10 + 48;
                // Extract hundreds digit
                text[6] = (temp_fraction/10)%10  + 48;
               // Extract tens digit
                  text[7] =  temp_fraction%10      + 48;
              // Extract ones digit

// print temperature on LCD
                  Lcd_Out(2, 5, text);
      }
```

PROGRAMA PRINCIPAL

```
      void main() {
      ANSEL  = 0;                        // Configure AN pins as digital I/O
      ANSELH = 0;
      Lcd_Init();                        // Initialize LCD
      Lcd_Cmd(_LCD_CLEAR);               // Clear LCD
      Lcd_Cmd(_LCD_CURSOR_OFF);          // Turn cursor off
      Lcd_Out(1, 1, " Temperature:   ");
```

```
 // Print degree character, 'C' for Centigrades
        Lcd_Chr(2,13,223);  // different LCD displays have different char code for
    degree
 // if you see greek alpha letter try typing 178 instead of 223

        Lcd_Chr(2,14,'C');

 //--- main loop
        do {
         //--- perform temperature reading
         Ow_Reset(&PORTA, 5);                    // Onewire reset signal
        Ow_Write(&PORTA, 5, 0xCC);              // Issue command SKIP_ROM
        Ow_Write(&PORTA, 5, 0x44);              // Issue command CONVERT_T
        Delay_us(120);

         Ow_Reset(&PORTA, 5);
        Ow_Write(&PORTA, 5, 0xCC);              // Issue command SKIP_ROM
        Ow_Write(&PORTA, 5, 0xBE);              // Issue command
        READ_SCRATCHPAD

        temp =  Ow_Read(&PORTA, 5);
        temp = (Ow_Read(&PORTA, 5) << 8) + temp;

  //--- Format and display result on Lcd
          Display_Temperature(temp);

         Delay_ms(500);
        } while (1);
}
```

3.2. Descripción del proyecto final

3.2.1. Diagrama de bloques

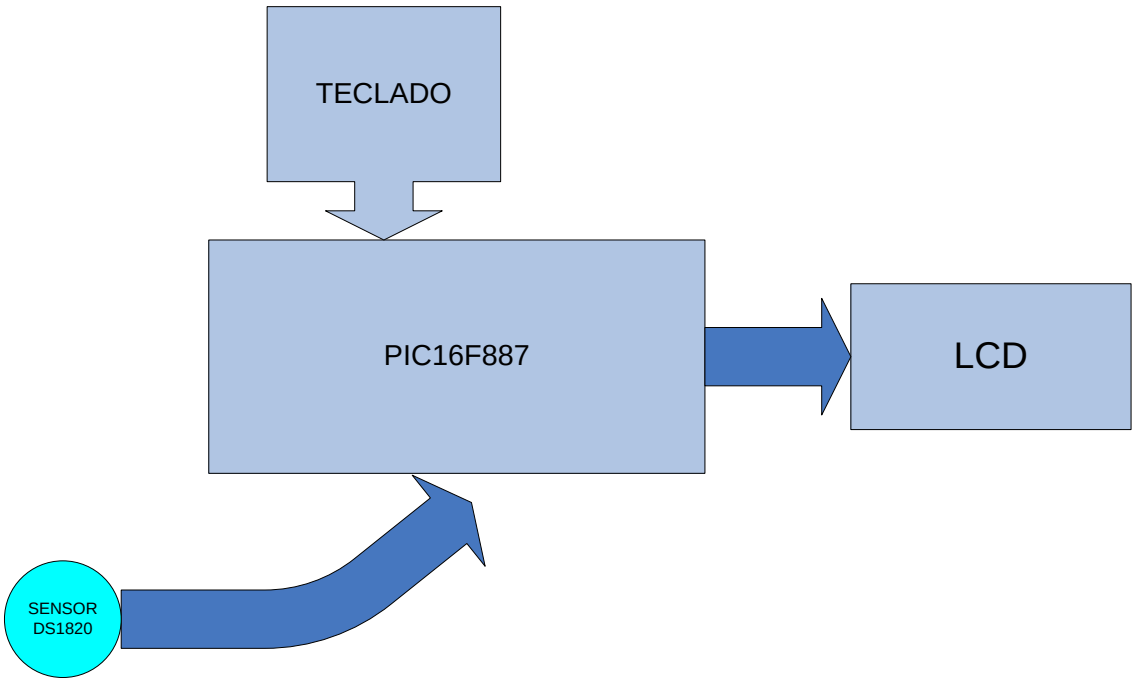


FIGURA 3.1.-Diagrama de bloques del proyecto

Podemos observar que el PIC 16F887 controla los datos obtenidos por el sensor de temperatura DS1820 y a su vez los datos ingresados por el teclado, con la finalidad de comparador los datos de cada uno de ellos y mostrar las alarmas respectivas en la LCD con su respectiva temperatura censada.

3.3. Algoritmo del microcontrolador

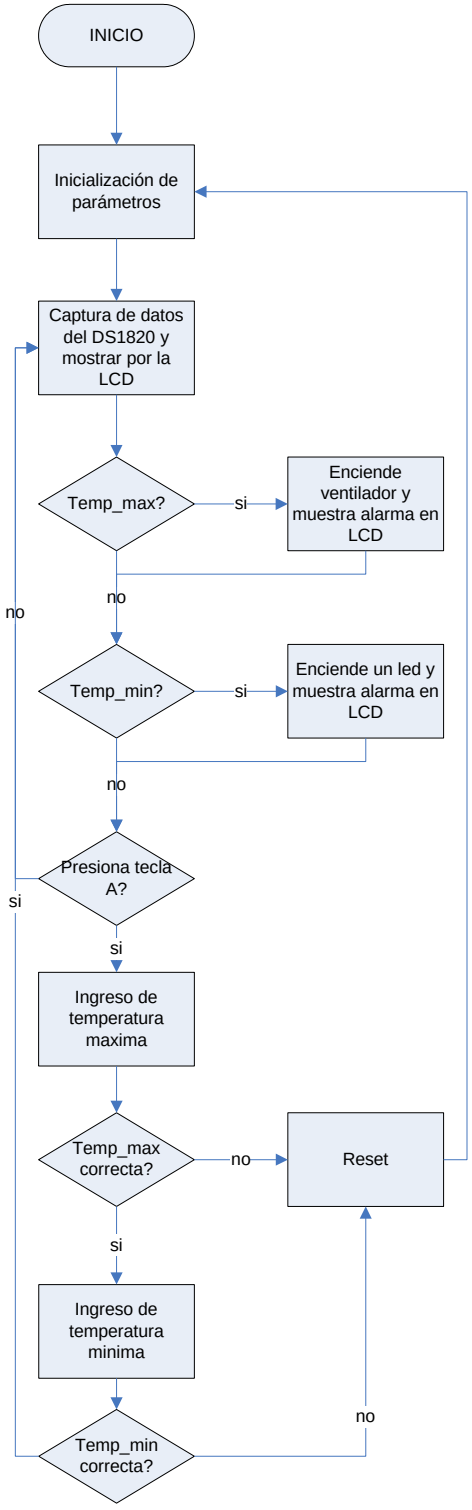


FIGURA 3.2.- Algoritmo del controlador

En el algoritmo del controlador se inicializa primero los puertos, luego el sensor de temperaura empieza a obtener los datos censados del ambiente y al presionar la tecla A del teclado se ingresan el rango de temperatura permitido, y se guardan en dos variables en Tmax y Tmin que luego son comparadas con el valor obtenido por el sensor, al salirse del rango se activan las alarmas respectivas y mostradas en la pantalla LCD.

3.4. Programa principal del Microcontrolador

```
void main() {
  ANSEL  = 0;                                  // Configure AN pins as digital I/O
  ANSELH = 0;
  C1ON_bit = 0;                                // Disable comparators
  C2ON_bit = 0;

  TRISC = 0X00;
  PORTC = 0X00;

  Lcd_Init();                                  // Initialize LCD
  Lcd_Cmd(_LCD_CLEAR);                         // Clear LCD
  Lcd_Cmd(_LCD_CURSOR_OFF);                    // Turn cursor off
  Lcd_Out(1, 1, "PROYECTO TEMP.");
  Lcd_Out(2, 1, "  CON ONEWIRE");
  Delay_ms(2000);
  MensajeInicial(1);
  Keypad_Init();                           // Initialize Keypad

  //--- Main loop
  do {
    //--- Perform temperature reading
    Ow_Reset(&PORTA, 1);                       // Onewire reset signal
    Ow_Write(&PORTA, 1, 0xCC);                 // Issue command SKIP_R
    Ow_Write(&PORTA, 1, 0x44);                 // Issue command CONVER
    Delay_us(120);

    Ow_Reset(&PORTA, 1);
    Ow_Write(&PORTA, 1, 0xCC);                 // Issue command SKIP_ROI
    Ow_Write(&PORTA, 1, 0xBE);                 // Issue command READ_SC   CHP

    Delay_ms(10);

    temp =  Ow_Read(&PORTA, 1);
    temp = (Ow_Read(&PORTA, 1) << 8) + temp;

    //--- Format and display result on Lcd
    Display_Temperature(temp);
    valtemp=temp>>1;
```

```
if (valtemp>=val1 && tmax)
{
   eval=1;
   tmax=0;
   tmin=1;
   Lcd_Cmd(_LCD_CLEAR);                              // Clear LCD
   Lcd_Cmd(_LCD_CURSOR_OFF);                         // Turn cursor off
   Lcd_Out(1, 1, " *ALARMA* ");
   Lcd_Out(2, 1, "Temp. Max. Excedida");
   Delay_ms(2000);
   PORTC=0X01;
   MensajeInicial(0);
}
else if (valtemp<=val2 && tmin)
{
   eval=1;
   tmax=1;
   tmin=0;
   Lcd_Cmd(_LCD_CLEAR);                              // Clear LCD

    Lcd_Cmd(_LCD_CURSOR_OFF);                         // Turn cursor off
    Lcd_Out(1, 1, " *INFORMACION* ");
    IRP_bit=1;
    Lcd_Out(2, 1, "Temp. Minima");
    IRP_bit=0;
    Delay_ms(2000);
    PORTC=0X02;
    MensajeInicial(0);
}
else if (valtemp>val2 && valtemp<val1)
{
     PORTC=0X00;
     if (eval)
     {
        MensajeInicial(1);
     }
     eval=0;
     tmax=1;
     tmin=1;
```

```
}
kp = Keypad_Key_Click();              // Store key code in kp variable
if (kp != 0)
{
   tecla();
}
if (ing == 10)
{
   //INGRESO DE TEMPERATURA MAXIMA
   Lcd_Cmd(_LCD_CLEAR);                              // Clear LCD
   Lcd_Cmd(_LCD_CURSOR_OFF);                         // Turn cursor off
   IRP_bit=1;
   Lcd_Out(1, 1, "Ingrese Temp Maxima: ");
   IRP_bit=0;
   kp=0;
   ingreso();
   val1 = ing*100;
   Lcd_Chr(2, 7, kp);
   kp=0;
   ingreso();
   val1 = val1 + (ing*10);
   Lcd_Chr(2, 8, kp);
   kp=0;
   ingreso();
   val1 = val1 + ing;
   Lcd_Chr(2, 9, kp);
   kp=0;
   ingreso();
   if (ing==14)
   {
      Lcd_Chr(2, 10, kp);
      kp=0;
      ingreso();
      val1 = val1 + (ing/10);
      Lcd_Chr(2, 11, kp);
   }
   Delay_ms(1000);
   //INGRESO DE TEMPERATURA MINIMA
```

```
        Lcd_Cmd(_LCD_CLEAR);                              // Clear LCD
        Lcd_Cmd(_LCD_CURSOR_OFF);                         // Turn cursor off
        IRP_bit=1;
        Lcd_Out(1, 1, "Ingrese Temp Minima: ");
        IRP_bit=0;
        kp=0;
        ingreso();
        val2 = ing*100;
        Lcd_Chr(2, 7, kp);
        kp=0;
        ingreso();
        val2 = val2 + (ing*10);
        Lcd_Chr(2, 8, kp);
        kp=0;
        ingreso();
        val2 = val2 + ing;
        Lcd_Chr(2, 9, kp);
        kp=0;
        ingreso();
        if (ing==14)
        {
           Lcd_Chr(2, 10, kp);
           kp=0;
           ingreso();
           val1 = val1 + (ing/10);
           Lcd_Chr(2, 11, kp);
        }
        Delay_ms(1000);
        MensajeInicial(1);
        tmax=1;
        tmin=1;
     }
     Delay_ms(500);
   } while (1);
}
```

3.5. Funciones implementadas en el Microcontrolador

3.5.1. Inicialización

Primero inicializan los puertos del microcontrolador para la utilización de los modulos a utilizar, tales como la pantalla LCD y el teclado.

```
unsigned short kp, cnt, oldstate = 0, val1=150, val2=0, ing=0, eval=0, tmax=0, tmin=0, salir=0;
char txt[6];

// Keypad module connections
char  keypadPort at PORTD;
// End Keypad module connections

// LCD module connections
sbit LCD_RS at RB4_bit;
sbit LCD_EN at RB5_bit;
sbit LCD_D4 at RB0_bit;
sbit LCD_D5 at RB1_bit;
sbit LCD_D6 at RB2_bit;
sbit LCD_D7 at RB3_bit;

sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB0_bit;
sbit LCD_D5_Direction at TRISB1_bit;

// End LCD module connections


//  Set TEMP_RESOLUTION to the corresponding resolution of used DS18x20 sensor:
//  18S20: 9  (default setting; can be 9,10,11,or 12)
//  18B20: 12
const unsigned short TEMP_RESOLUTION = 9;

char text[9] = "000.0000";
unsigned temp,valtemp=0;
```

3.5.2. Funciones TECLA, INGRESO, MENSAJE INICIAL Y DISPLAY_TEMPERATURE.

TECLA: Esta función se encarga de obtener mediante un teclado de 4x4 el valor de la tecla presionada y convertirla para la lectura del microcontrolador.

```
void tecla()
{
 switch (kp)
      {
      case  1: kp = 49; ing = 1; break; // 1        // Uncomment this block for keypad4x4
      case  2: kp = 50; ing = 2; break; // 2
      case  3: kp = 51; ing = 3; break; // 3
      case  4: kp = 65; ing = 10;break; // A
      case  5: kp = 52; ing = 4; break; // 4
      case  6: kp = 53; ing = 5; break; // 5
      case  7: kp = 54; ing = 6; break; // 6
      case  8: kp = 66; ing = 11;break; // B
      case  9: kp = 55; ing = 7; break; // 7
      case 10: kp = 56; ing = 8; break; // 8
      case 11: kp = 57; ing = 9; break; // 9
      case 12: kp = 67; ing = 12;break; // C
      case 13: kp = 42; ing = 15;break; // *
      case 14: kp = 48; ing = 0; break; // 0
      case 15: kp = 46; ing = 14;break; // #
      case 16: kp = 68; ing = 13;break; // D
      }
}
```

INGRESO: Al momento de presionar la tecla del teclado 4x4 se almacena en esta función para su posterior utilización en los parámetros del programa.

```
void ingreso()
{
       do
        {
           kp = Keypad_Key_Click();             // Store key code in kp variable
        } while(!kp);
        tecla();
}
```

MENSAJE INICIAL: Esta función maneja la LCD , primero limpia la pantalla y luego procede a escribir el mensaja referencial donde se va a mostrar la temperatura obtenida por el sensor DS1820.

```
void MensajeInicial(unsigned mostrar)
{
        if (mostrar)
        {
           Lcd_Cmd(_LCD_CLEAR);                              // Clear LCD
        }
        else
        {
           Lcd_Out(2, 1, "                                ");
        }
        Lcd_Cmd(_LCD_CURSOR_OFF);                          // Turn cursor off
        if (mostrar)
        {
           Lcd_Out(1, 1, " Temperatura:                   ");
        }
        Lcd_Chr(2,13,223);
        // Different LCD displays have different char code for degree
        Lcd_Chr(2,14,'C');
}
```

DISPLAY_TEMPERATURE: Esta función se encarga de obtener los datos provenientes del DS1820, el cual vienen en 9 bits y hacer la conversión respectivas de los datos para guardarla en una variable text y poder mostrarla en la pantalla LCD.

```
void Display_Temperature(unsigned int temp2write) {
  const unsigned short RES_SHIFT = TEMP_RESOLUTION - 8;
  char temp_whole;
  unsigned int temp_fraction;

  // Check if temperature is negative
  if (temp2write & 0x8000) {
     text[0] = '-';
     temp2write = ~temp2write + 1;
     }

  // Extract temp_whole
  temp_whole = temp2write >> RES_SHIFT ;

  // Convert temp_whole to characters
  if (temp_whole/100)
     text[0] = temp_whole/100  + 48;
  else
     text[0] = '0';

  text[1] = (temp_whole/10)%10 + 48;             // Extract tens digit
  text[2] =  temp_whole%10     + 48;             // Extract ones digit

  // Extract temp_fraction and convert it to unsigned int
  temp_fraction  = temp2write << (4-RES_SHIFT);
  temp_fraction &= 0x000F;
  temp_fraction *= 625;

  // Convert temp_fraction to characters
  text[4] =  temp_fraction/1000    + 48;         // Extract thousands digit
  text[5] = (temp_fraction/100)%10 + 48;         // Extract hundreds digit
  text[6] = (temp_fraction/10)%10  + 48;         // Extract tens digit
  text[7] =  temp_fraction%10      + 48;         // Extract ones digit

  // Print temperature on LCD
  //IRP_bit=1;
  Lcd_Out(2, 5, text);
  //IRP_bit=0;
}
```

CAPÍTULO 4

4. SIMULACIÓN Y PRUEBAS

A continuación describiremos las simulaciones y pruebas desarrolladas en nuestro proyecto, cabe recalcar en PROTEUS se simuló el funcionamiento del sistema con el sensor DS1820 utilizando el protocolo one-wire, en conjunto con la pantalla LCD y el teclado 4x4.

Mostraremos primero las pruebas que realizamos utilizando las herramientas proporcionadas del programa PROTEUS, tales como el sensor de temperatura, los botones para armar el teclado 4x4, resistencias, pantalla LCD y demás elementros. También se anexan fotos del proyecto implementado en protoboard y además la prueba realizada en la placa con el DS1820.

4.1. Simulación en Proteus

A través de las herramientas proporcionadas por Proteus podemos observar como se establecen las alarmas del sistema mediante el teclado 4x4, y el microcontrolador se encarga se determinar si se encienden o no las alarmas dependiendo de la temperatura del ambiente.

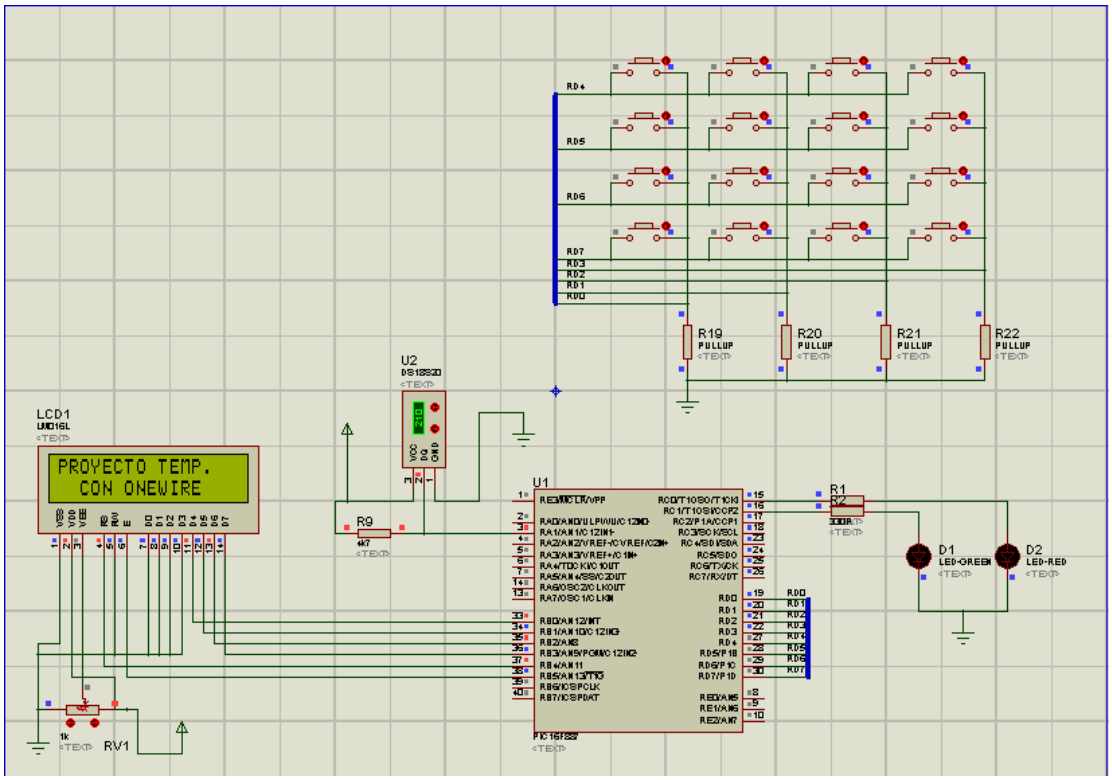


FIGURA 4.1.- Simulación en PROTEUS

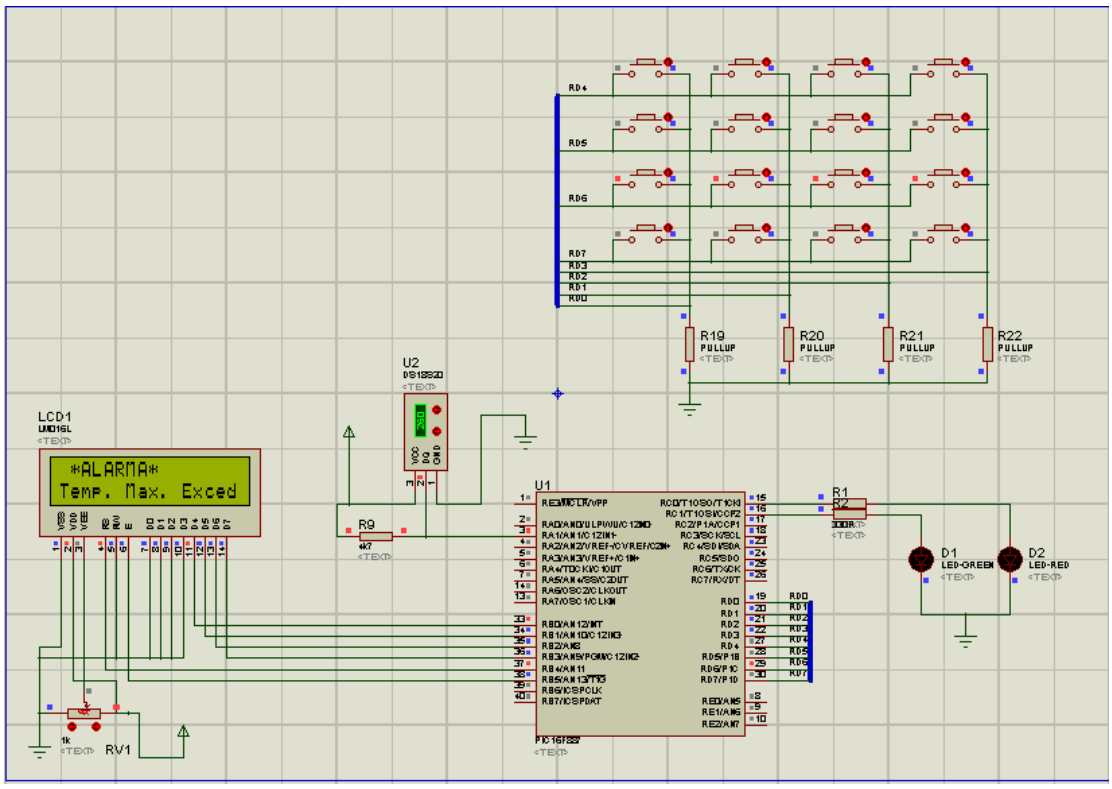


FIGURA 4.2.- Simulación de la alarma del sistema con el sensor DS1820

Podemos observar una prueba del sistema de alarma constituido por el sensor inteligente de temperatura DS1820 junto con el teclado 4x4 y la pantalla LCD la cual muestra la alarma cuando la temperatura del sistema sobrepasa el valor de la temperatura máxima ingresado por el usuario.

4.2. Implementación en protoboard

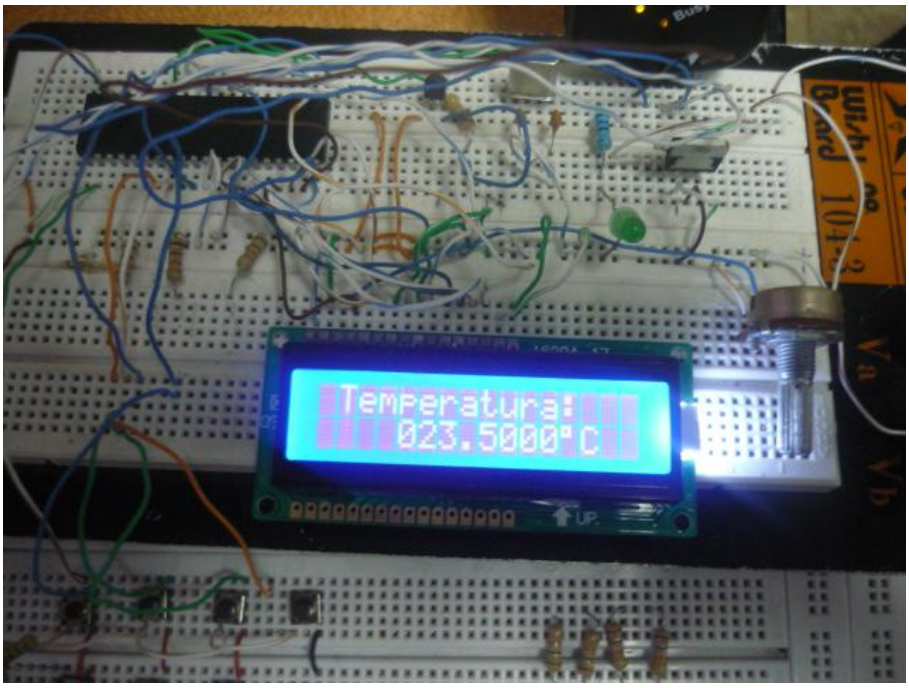


FIGURA 4.3.- Sistema de Alarma con el Sensor DS1820

En esta fotografía mostramos el funcionamiento del sensor DS1820 proporcionando la temperatura del ambiente y mostrandola en la pantalla LCD.

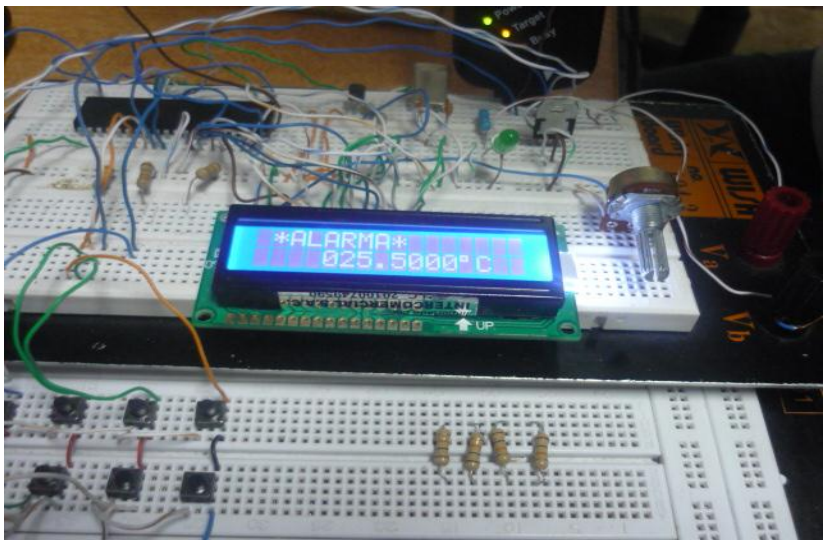


FIGURA 4.4.- Alarma con el Sensor DS1820

4.3. ESQUEMA DE CONEXIONES DEL CONTROLADOR

A continuación mostramos la simulación en Proteus con la ayuda de las herramientas del programa como esta conectado el controlador con los distintos dispositivos a emplear para el control de temperatura con alarmas.

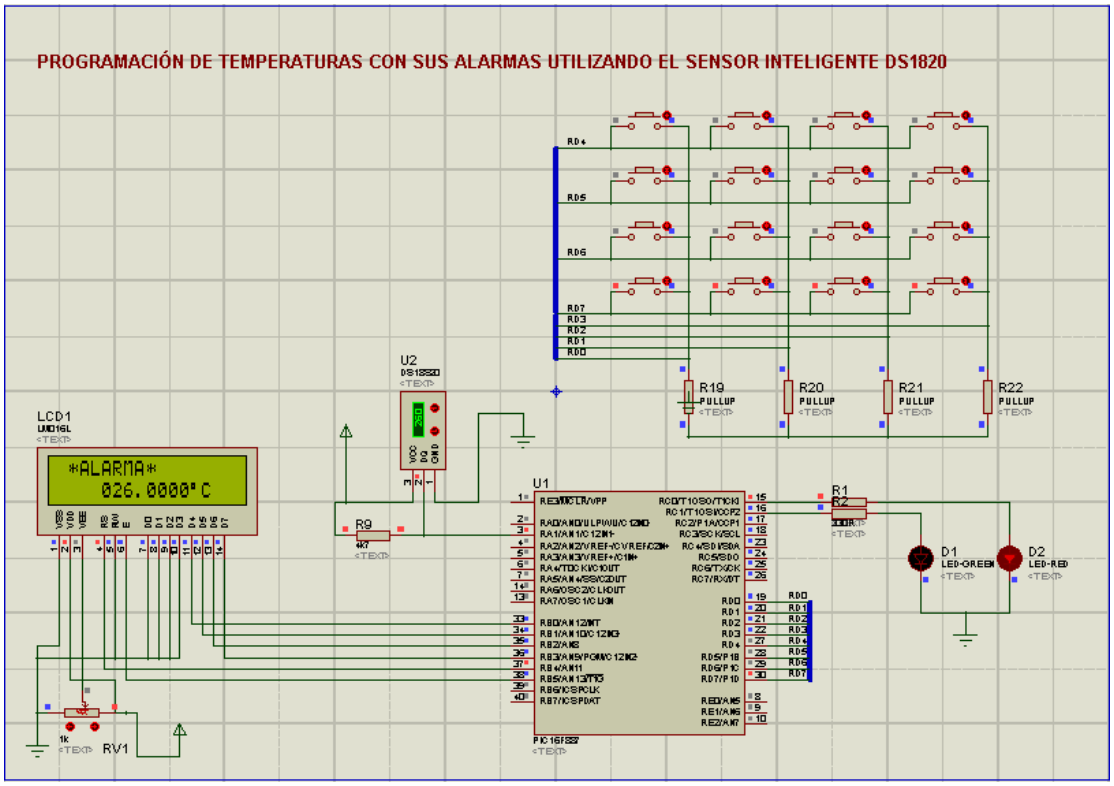


FIGURA 4.5.- Esquema de Conexión del Microcontrolador

CONCLUSIONES Y RECOMENDACIONES

1. Logramos construir un sistema que permite el control de la temperatura en un sistema cerrado a través del sensor inteligente DS1820, a través de dispositivos como los microcontroladores para manipular los datos obtenidos y proporcionar las alarmas necesarias al sistema.
2. El sensor de temperatura DS1820 utiliza el protocolo de comunicación one-wire que permite realizar una comunicación serial asincrónica entre un dispositivo maestro y uno o varios dispositivos esclavos, utilizando un único pin de E/S del microcontrolador.
3. Los valores de la temperatura máxima y mínima se guardan en distintas variables para luego su posterior comparación y comprobar si el sistema está estable, si se desestabiliza el sistema, se encenderá un ventilador para lograr volver a su estado estable.
4. Las rutinas del protocolo one-wire proporcionadas por el programa mikroc pro for pic nos permiten convertir los datos proporcionados

por el sensor DS1820 de bits a valores tipo char, para estos poder enviar a las funciones que permiten la visualización de los mensajes en la pantalla LCD.

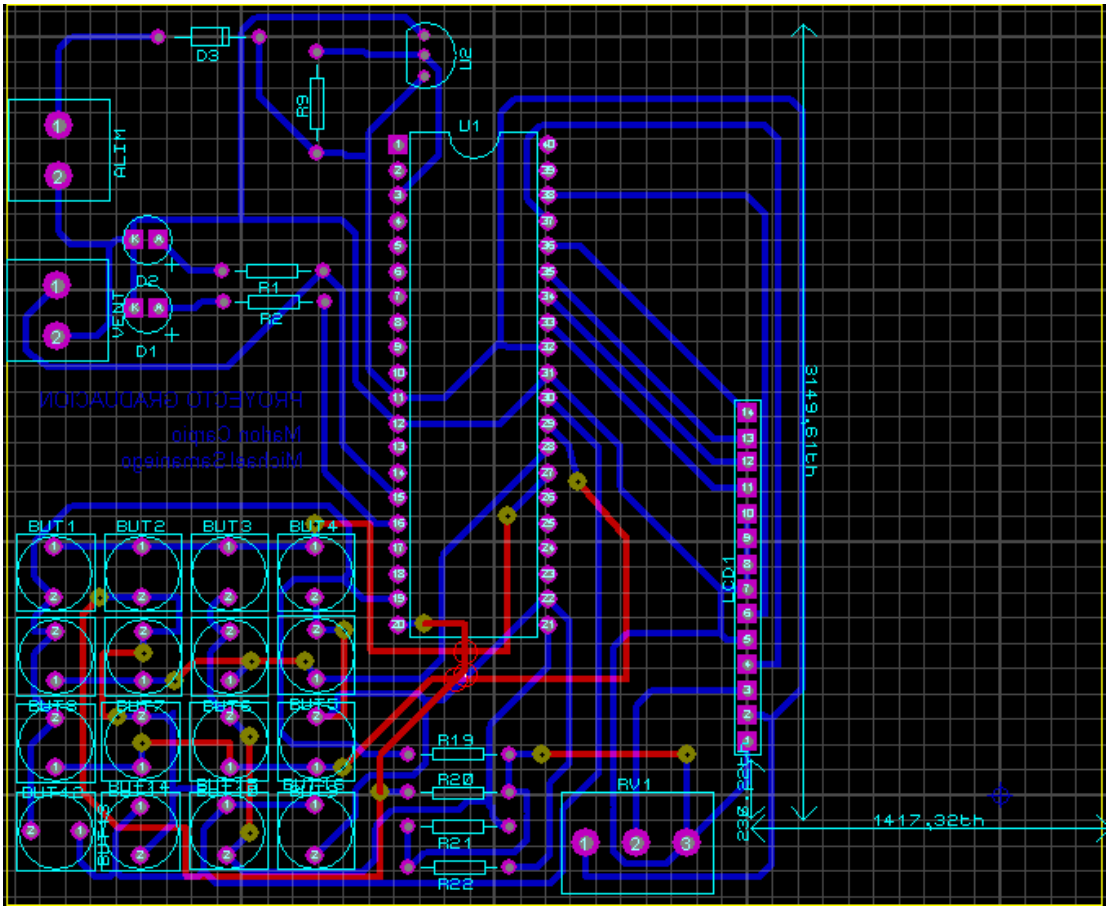
5. Cuando se ingresa los valores de los rangos máximos y mínimo de temperatura por el teclado 4x4, se debe procurar que estos valores sean acordes a los parámetros del sensor de temperatura DS1820 que solo soporta valores de temperatura entre -55 ^{o}C y +125 ^{o}C para que el sistema tenga un perfecto funcionamiento.
6. Verificar que el microcontrolador trabaje con una frecuencia de al menos 4Mhz, debido que las rutinas de la librería one-wire requieren ese parámetro para la utilización de termómetros digitales.
7. Crear un modelo adecuado de comandos para que la comunicación entre el sensor y el microcontrolador sea eficiente, esto es respetando el tiempo que el sensor necesita para la captura de datos.
8. Es necesario un voltaje levemente mayor para encender el ventilador durante la alarma máxima.

ANEXOS

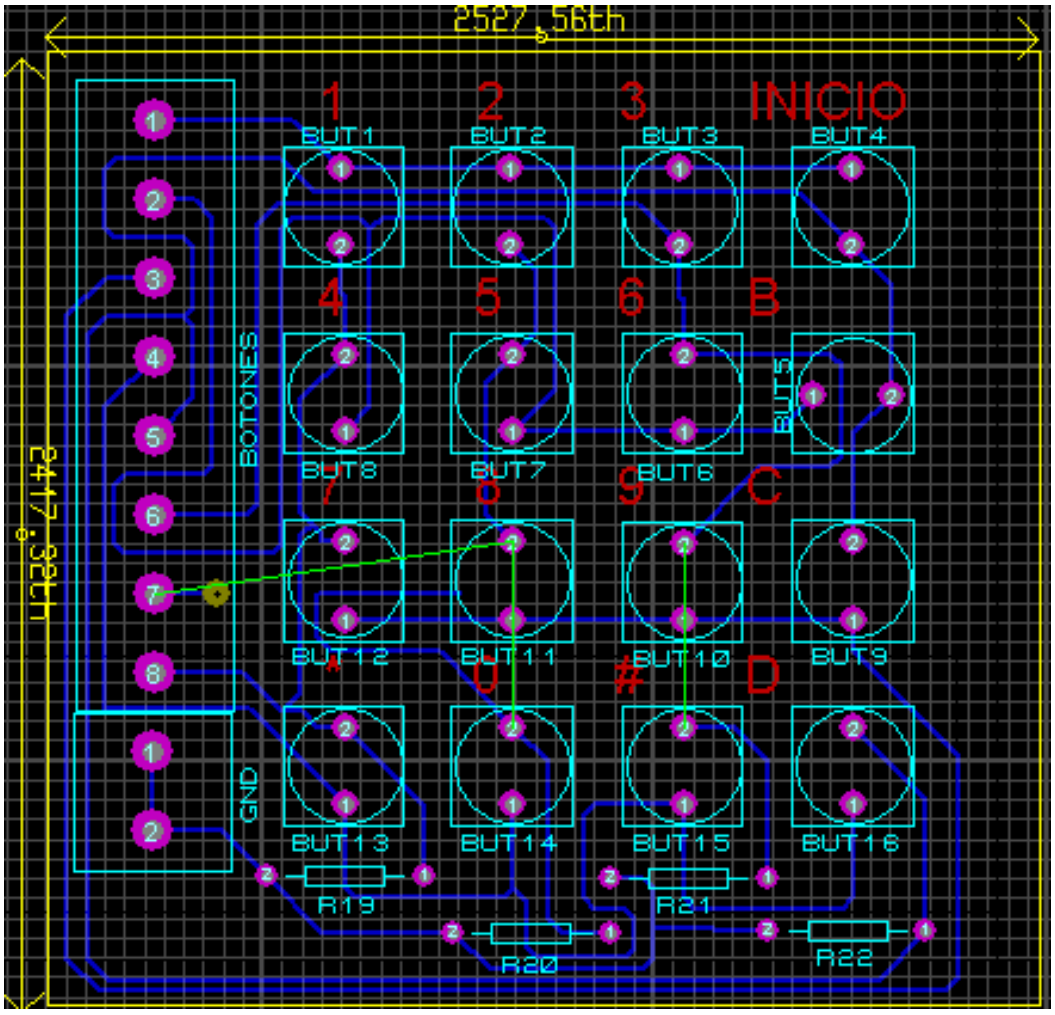
ANEXO A: DISEÑO DE LA TARJETA ELECTRÓNICA

El diseño se realizó utlizando las herramientas de PROTEUS en conjunto con ARES, cabe mencionar que se diseño dos placas, una que se compone del microcontrolador con el sensor de temperatura y la pantalla LCD y otra placa donde exclusivamente se encuentra el teclado 4x4.

DISEÑO PLACA CON MICROCONTROLADOR Y SENSOR



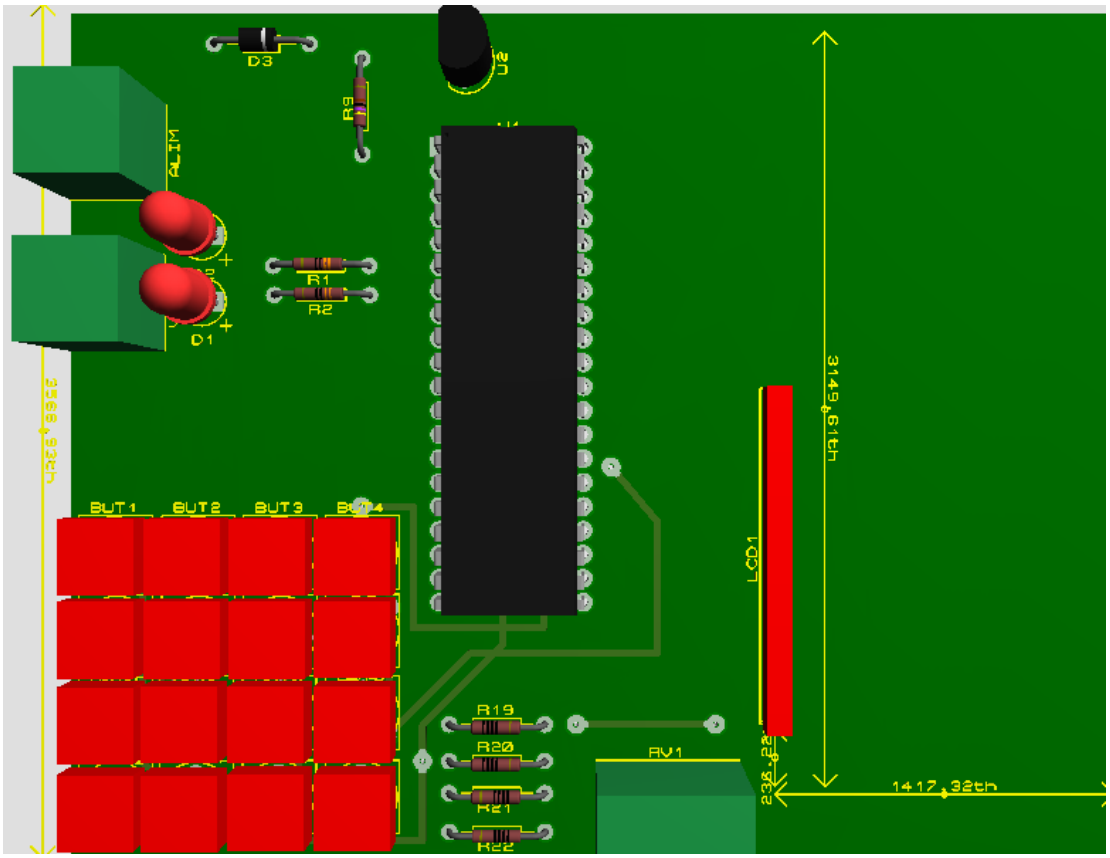
DISEÑO TECLADO 4X4



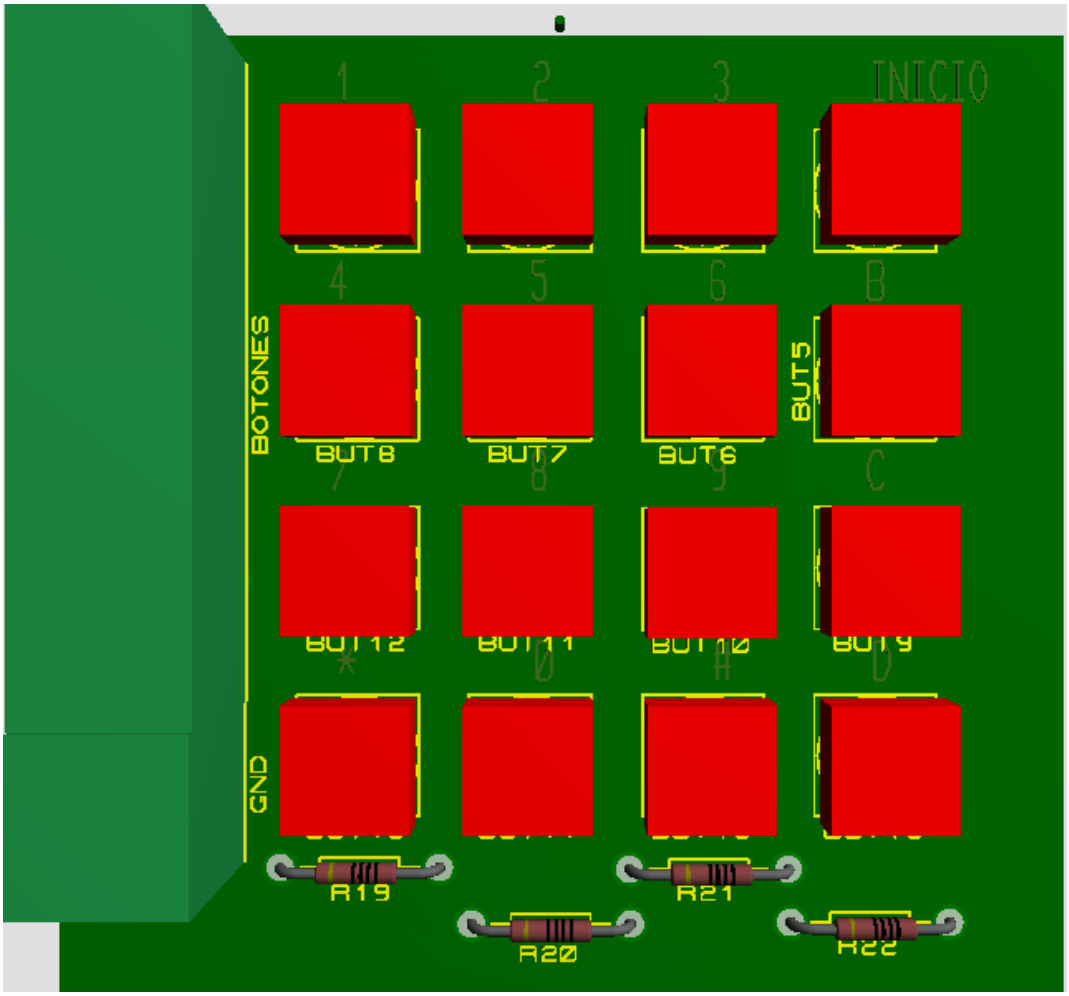
ANEXO B: VISTA 3D DEL DISEÑO.

Esta herramienta de ARES nos permite visualizar como quedaría el diseño de la tarjeta :

VISTA 3D MICROCONTROLADOR



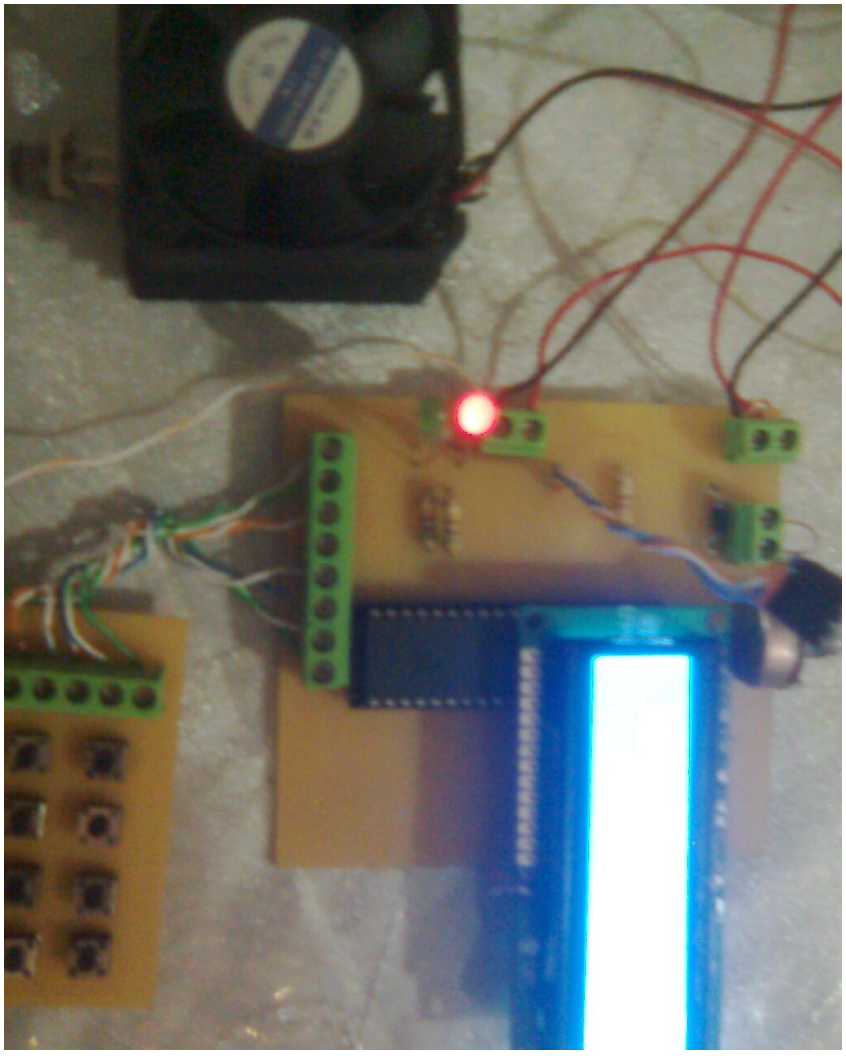
VISTA 3D TECLADO 4X4



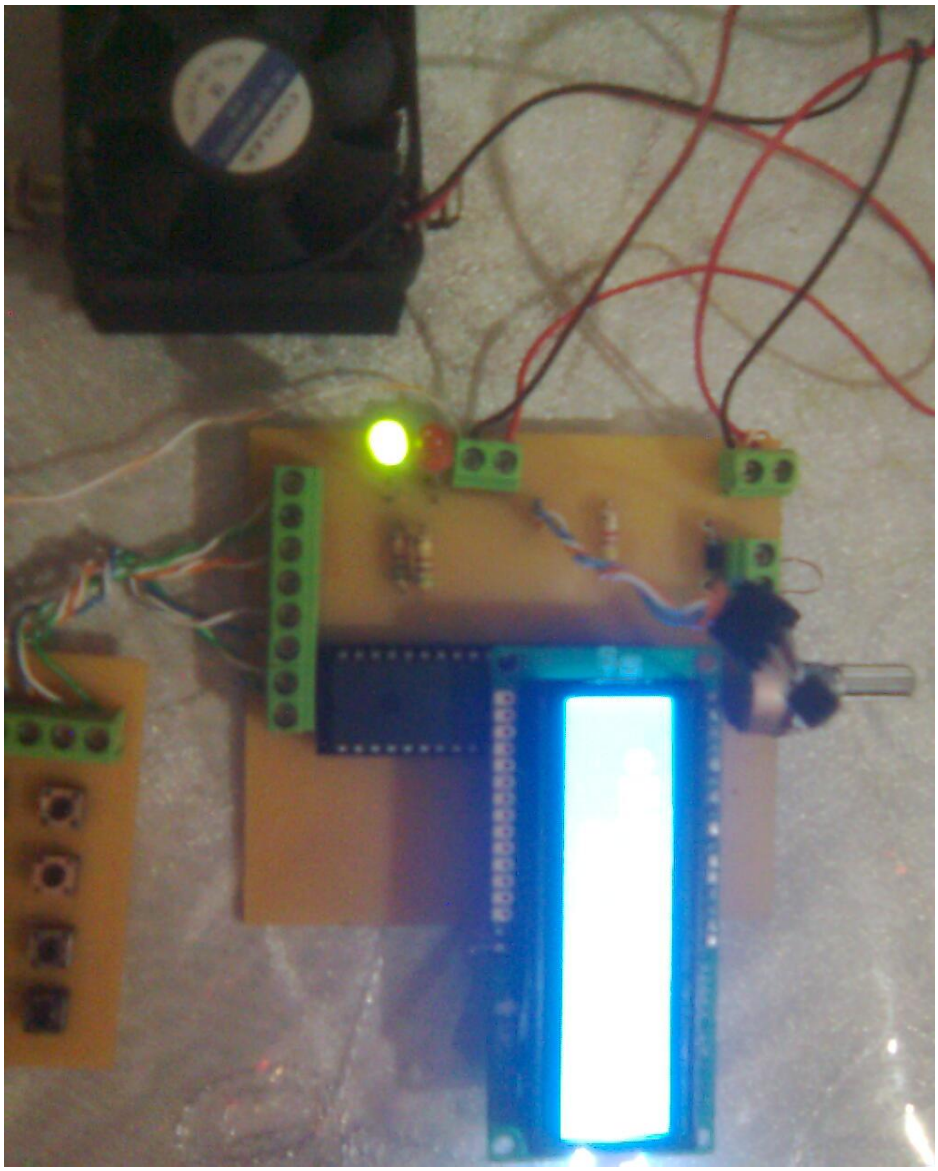
ANEXO C: FOTOGRAFÍAS DE LA TARJETA ELECTRÓNICA.

Estas fotografías muestran el resultado de un eficaz uso de las herramientas de PROTEUS Y ARES además de la paciencia y destreza que requiere soldar lo elementos.

ALARMA DE TEMPERATURA ALTA



ALARMA DE TEMPERATURA BAJA



BIBLIOGRAFÍA

1. Wikipedia, 1-Wire :

   http://es.wikipedia.org/wiki/1-Wire , **Fecha de Consulta** : 20/11/2010 .

2. Scribd , Protocolo 1 Wire ;

   http://www.scribd.com/doc/24421918/Protocolo-1-Wire ; **Fecha de Consulta:** 21/11/2010.

3. Mikroelectrónica , PIC Microcontrollers – Programming in C ;

   http://www.mikroe.com/eng/chapters/view/79/capitulo-1-el-mundo-de-los-microcontroladores/ ; **Fecha de Consulta :** 23/11/2010.

4. Mikroelectrónica , MikroC pro for Pic;

   http://www.mikroe.com/eng/products/view/7/mikroc-pro-for-pic/ ; **Fecha de Consulta:** 23/11/2010.

5. Microchip , Hoja de Datos PIC 16F887 ;

   http://ww1.microchip.com/downloads/en/DeviceDoc/41291F.pdf ; **Fecha de Consulta:** 25/11/2010 .

6. Systronix , Hoja de Datos Sensor DS1820, DS18S20 ;

   http://www.systronix.com/Resource/ds1820.pdf ; Fecha de Consulta: 25/11/2010 .

7. Ing. Marcelo E. Romeo, Dispositivos de Medición de Temperatura y Herramientas de Desarrollo ;

   meromeo@elecron.frba.utn.edu.ar ; **Fecha Consulta :** 29/11/2010