UNIVERSIDAD POLITÉCNICA DE VALENCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

PROYECTO FINAL DE CARRERA

MONITOR REMOTO DE TEMPERATURA Y HUMEDAD

CÓDIGO P.F.C.: DISCA-115

ALUMNO: **JAVIER LLUESMA JUAN**

DIRECTOR: **ÁNGEL RODAS JORDÁ**

Indice

1. Objetivos

Objetivo general

Realizar el diseño y la implementación de un dispositivo que sea capaz de captar dos variables como son la temperatura ambiente y la humedad relativa.
Una vez captadas serán transmitidas de forma inalámbrica a un ordenador, para ser interpretadas y visualizadas.

Objetivos específicos

- ✓ Estudiar las características que debiera tener el dispositivo.
- ✓ Diseñar dicho dispositivo teniendo en cuenta las diferentes etapas que lo conforman.
- ✓ Realizar el estudio de cada una de las etapas.
- ✓ Implementar de manera experimental el dispositivo.
- ✓ Optimizar el funcionamiento de este, adecuándolo a los requerimientos reales.

2. Introducción

El proyecto que se va a llevar a cabo, consiste en el diseño e implementación de un dispositivo que permite el monitoreo constante y de forma automática de dos variables, como son la humedad relativa y la temperatura ambiente (consideradas de mayor relevancia en el funcionamiento de un invernadero).
Dichas variables serán capturadas por medio de un sensor.

Una vez obtenidos los valores, estos serán clasificados mediante el *firmware* del microcontrolador, para ser transmitidos de forma inalámbrica a un ordenador (podría considerarse una estación de medición), donde serán recepcionados y visualizados.

3. Marco teórico

Hoy en día, en la mayoría de los invernaderos de algunas zonas, la toma de datos se realiza de forma manual. Si bien, éstos se guardan en un ordenador, no es un monitoreo constante que indique los cambios bruscos. Es decir, no posibilita un análisis más profundo en relación a la variación de las condiciones dentro de un invernadero.

El monitoreo de forma automática, si nos posibilita cierto control y estudio más a fondo de éste. Las variables generalmente utilizadas en los invernaderos son las de temperatura, humedad ambiental y humedad del suelo, aunque esta última no será de nuestro interés. Instaurando nuevas herramientas, las variables podrían ser captadas mediante sensores especializados y los datos serían enviados a un microcontrolador el cual los prepararía para ser remitidos a través de una tarjeta de transmisión inalámbrica, y luego ser recibidos por una tarjeta receptora de datos conectada a un ordenador.

Fig. 3.1.

4. Componentes del M.R.I.

A continuación se muestra un diagrama de bloques genérico (Fig. 4.1) para facilitar el entendimiento y la visión global del proyecto.

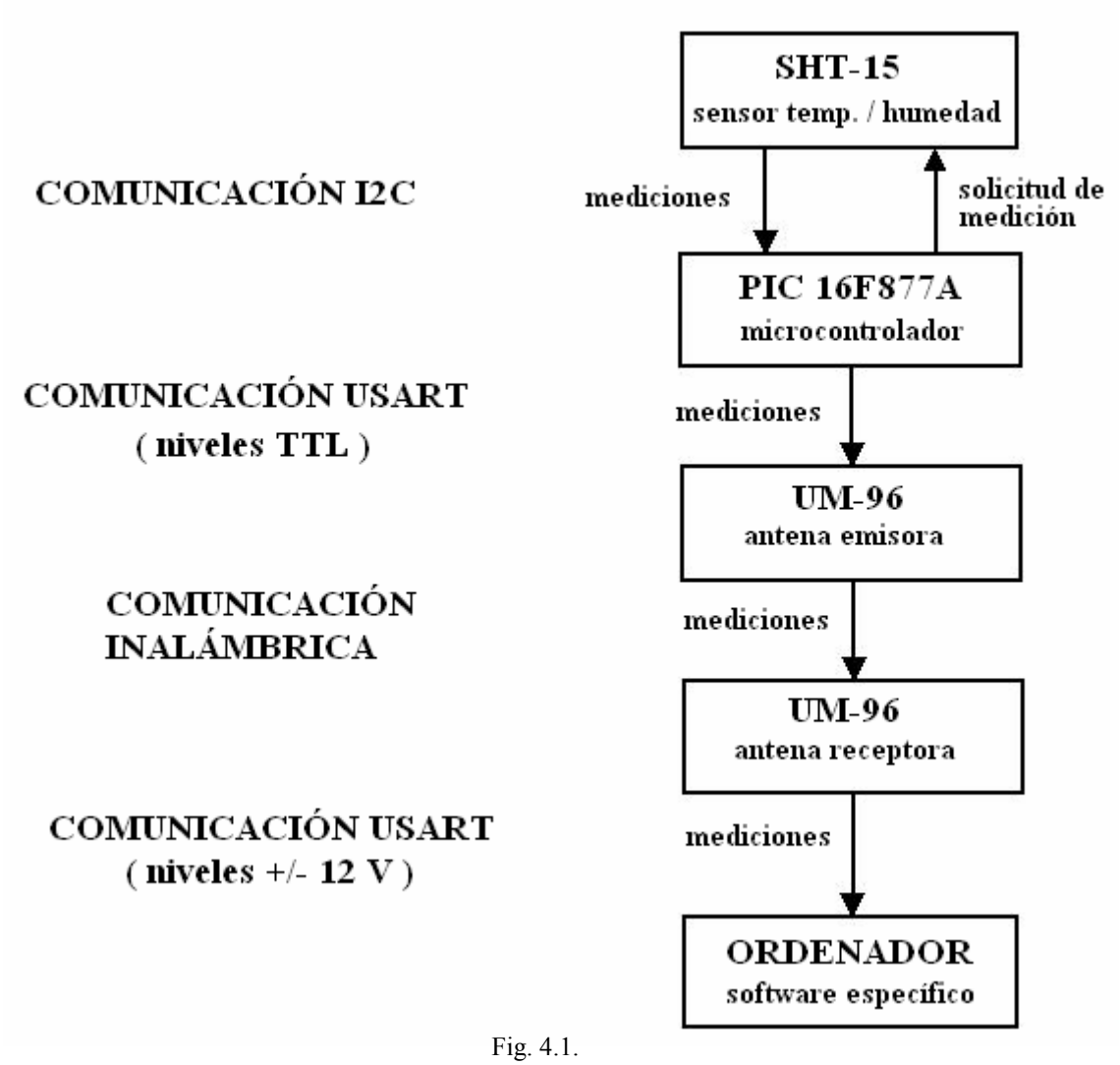


Fig. 4.1.

En el diagrama se puede observar que, mediante comunicación I2C, el microcontrolador solicita valores de medición al sensor y éste, tras un tiempo de medición y también bajo comunicación I2C, le transfiere dicha información, que a su vez es transmitida al módulo UM-96 vía USART (niveles TTL).
Este primer módulo UM-96 hace las veces de emisor, siendo el encargado de transmitir vía inalámbrica los valores de la medición a un segundo módulo que hace de receptor.
Éste último, una vez recibe la información, la envía al ordenador utilizando la comunicación USART, en este caso con niveles RS-232. El ordenador, en primer lugar, trata la información recibida para, seguídamente, visualizarla por pantalla.

4.1. MicroControlador. PIC 16F877A

La función del microcontrolador dentro del circuito es la de gestionar prácticamente todos los componentes. Como su palabra indica, es un controlador, el corazón del circuito.
Según el *firmware* que tenga programado, va a marcar las pautas de todo aquello que ocurra a "su alrededor".
Por una parte, será el encargado de recibir información del exterior, tendrá la capacidad de procesar y organizar dicha información y, finálmente, dará salida de la información que se precise como y cuando estime oportuno.
Para este circuito se ha buscado solución para gestionar información de entrada y salida de una manera óptima, y la elección del PIC 16F877 de la empresa Microchip ha parecido la más certera por sus características, que más adelante se explican.

Dando una visión general de los microcontroladores, se podría decir que son circuitos integrados programables, capaces de ejecutar las instrucciones que existen en su memoria. Principalmente constan de un microprocesador, líneas de entrada y salida, memorias RAM y ROM. Para su funcionamiento requiere de alimentación, de un oscilador, y de un programa de instrucciones.

La función principal del microcontrolador es interpretar combinaciones de bits y generar señales digitales internas y o externas, para ejecutar de manera continua una secuencia de instrucciones que permita controlar un sistema o subsistema electrónico. Estos dispositivos vienen con un juego de instrucciones reducido, además de su pequeño encapsulado con pocos pines y poco consumo, lo cual los hacen muy utilizables.

La mayoría de los microcontroladores constan con las siguientes características:

- ✓ Procesador o CPU: es quien procesa todos los datos que pasan por el bus.
- ✓ Memoria ROM: es la memoria no volátil, que es donde se guardan los programas
- ✓ Memoria RAM: o memoria volátil, que es donde se guardan los datos.
- ✓ Oscilador: que sincroniza todo el funcionamiento del sistema.
- ✓ Puertos de entrada y salida: Es por donde se comunica el microcontrolador con los periféricos externos.
- ✓ Convertidores análogo digital(A/D), digital análogo (D/A): como su nombre los dice convierten señales de análogas a digital y viceversa.
- ✓ Temporizadores: sirven para controlar periodos de tiempo.
- ✓ Comparadores: como su nombre lo indica comparan señales analógicas.
- ✓ Moduladores de ancho de pulso: esta función modula en PWM.
- ✓ Puerto USART: comunicación serie de transmisión y recepción.
- ✓ Controladores de interrupciones
- ✓ Perro guardián o watch dog: contador que resetea el microcontrolador cada ves que se desborda el stack.
- ✓ Protección ante fallo de alimentación: resetea el microcontrolador cada ves que la alimentación baja de un cierto limite.

Microchip ha dividido sus microcontroladores en tres grandes subfamilias de acuerdo al número de bits de su bus de instrucciones:

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

Tabla 4.1.

Variantes principales

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

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

El microcontrolador PIC16F877

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

- ✓ Arquitectura Harvard
- ✓ Tecnología RISC
- ✓ Tecnología CMOS

Estas características se conjugan para lograr un dispositivo altamente eficiente en el uso de la memoria de datos y programa y por lo tanto en la velocidad de ejecución.

Empaquetados

Aunque cada empaquetado tiene variantes, especialmente en lo relativo a las dimensiones del espesor del paquete, en general se pueden encontrar paquetes tipo PDIP (Plastic Dual In Line Package), PLCC (Plastic Leaded Chip Carrier) y QFP (Quad Flat Package), los cuales se muestran en las siguientes ilustraciones (Fig. 4.2.):

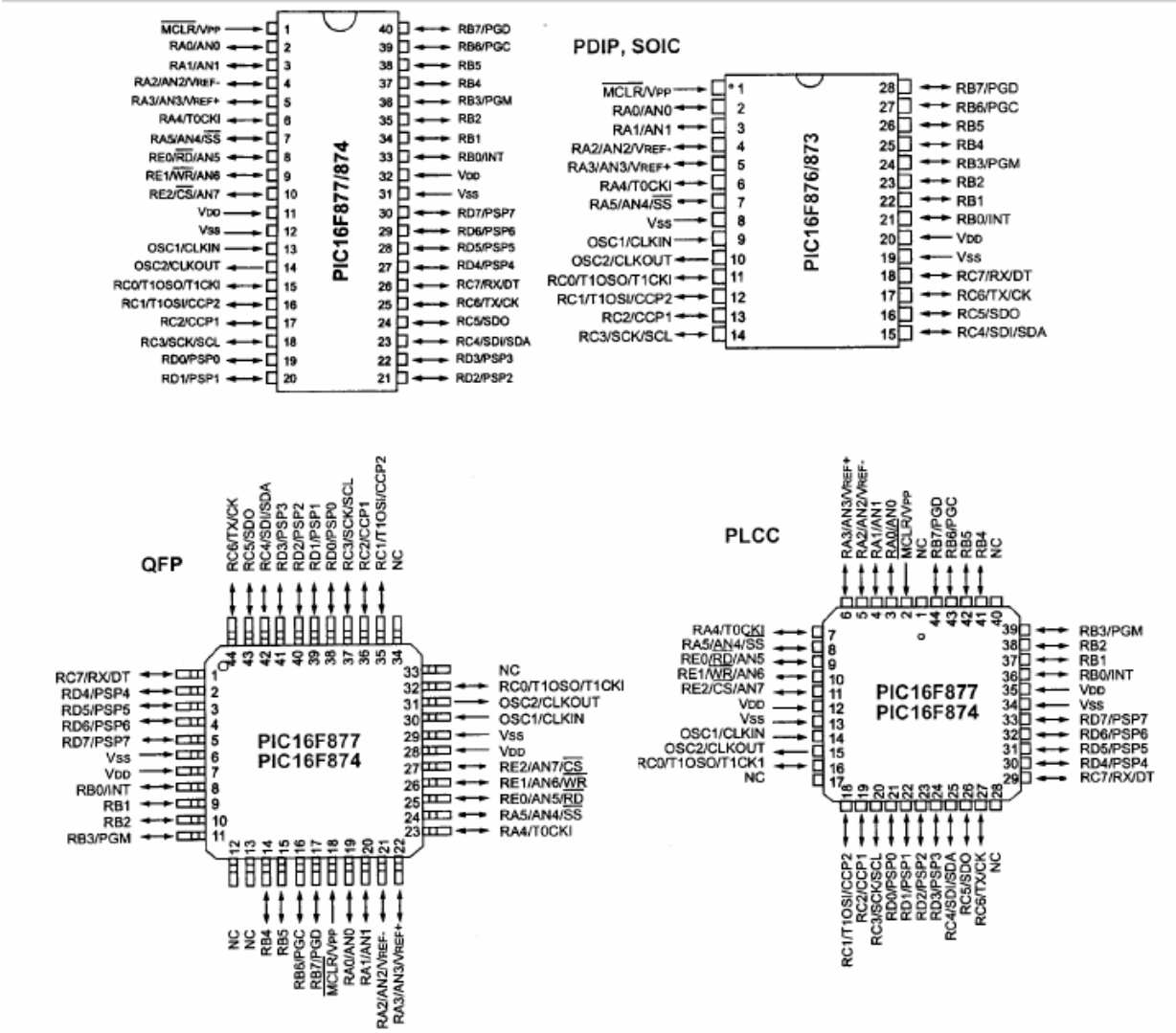


Fig. 4.2.

Nomenclatura

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

Tipo de memoria	Rango de voltaje	
	Estándar	Extendido
EPROM	PIC16**C**XXX	PIC16**LC**XXX
ROM	PIC16**CR**XXX	PIC16**LCR**XXX
Flash	PIC16**F**XXX	PIC16**LF**XXX

Tabla 4.2.

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

Rango de voltaje	EPROM		ROM		Flash	
Estándar	C	4.5 a 6v	CR	4.5 a 6v	F	4.5 a 6v
Extendido	LC	2.5 a 6v	LCR	2.5 a 6v	LF	2 a 6v

Tabla 4.3.

Oscilador

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

Los modos de operación se muestran en la siguiente lista:

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

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

Modo	Frecuencia típica	Capacitores recomendados	
		C1	C2
LP	32 khz 200 khz	68 a 100 pf 15 a 30 pf	68 a 100 pf 15 a 30 pf
XT	100 khz 2 Mhz 4 Mhz	68 a 150 pf 15 a 30 pf 15 a 30 pf	150 a 200 pf 15 a 30 pf 15 a 30 pf
HS	8 Mhz 10 Mhz 20 Mhz	15 a 30 pf 15 a 30 pf 15 a 30 pf	15 a 30 pf 15 a 30 pf 15 a 30 pf

Tabla 4.4.

Cristal externo

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

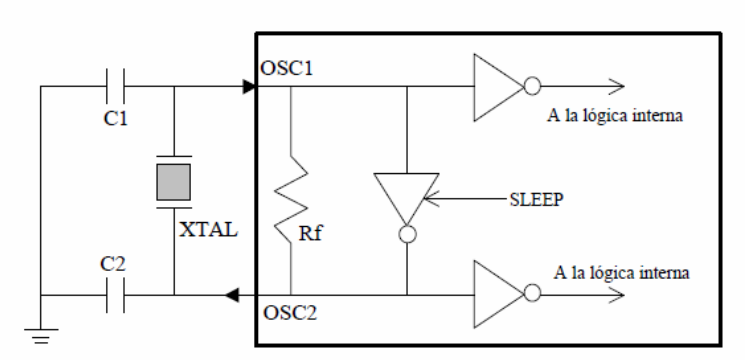


Fig. 4.3.

Circuito RC externo

En los modos RC y EXTRC el PIC puede generar su señal oscilatoria basada en un arreglo RC externo conectado a la patita OSC1 como se muestra en la siguiente figura.

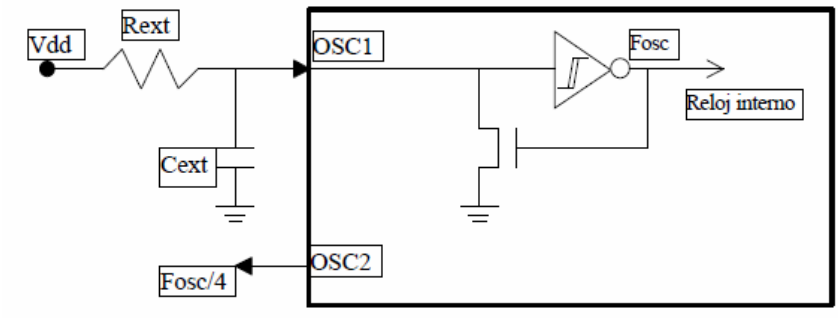


Fig. 4.4.

Este modo sólo se recomienda cuando la aplicación no requiera una gran precisión en la medición de tiempos.

Rangos.- La frecuencia de oscilación depende no sólo de los valores de Rext y Cext, sino también del voltaje de la fuente Vdd. Los rangos admisibles para resistencia y capacitor son:

Rext: de 3 a 100 Kohms
Cext: mayor de 20 pf

Oscilador externo.- También es posible conectar una señal de reloj generada mediante un oscilador externo a la patita OSC1 del PIC.
Para ello el PIC deberá estar en uno de los tres modos que admiten cristal (LP, XT ó HS). La conexión se muestra en la siguiente figura.

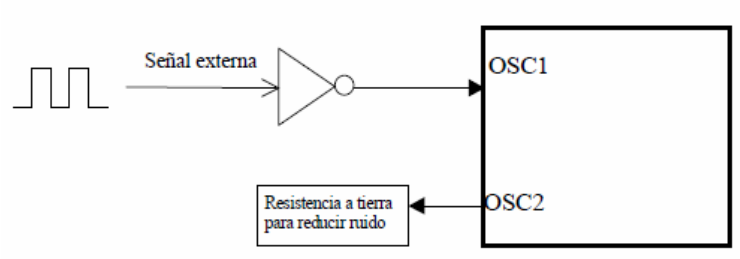


Fig. 4.5.

Oscilador interno de 4Mhz.- En el modo INTRC el PIC usa un arreglo RC interno que genera una frecuencia de 4 Mhz con un rango de error calibrable de ± 1.5%. Para calibrar el error de oscilación se usan los bits CAL3, CAL2 , CAL1 Y CAL0 del registro OSCCAL.

Calibración del oscilador interno.- El fabricante ha colocado un valor de calibración para estos bits en la última dirección de la memoria de programa. Este dato ha sido guardado en la forma de una instrucción RETLW XX. Si no se quiere perder este valor al borrar el PIC (en versiones EPROM con ventana) primero se deberá leer y copiar. Es una buena idea escribirlo en el empaquetado antes de borrar la memoria).

Características generales

- ✓ CPU RISC
- ✓ Sólo 35 instrucciones que aprender
- ✓ Todas las instrucciones se ejecutan en un ciclo de reloj, excepto los saltos que requieren dos
- ✓ Frecuencia de operación de 0 a 20 MHz (DC a 200 nseg de ciclo de instrucción)
- ✓ Hasta 8k x 14 bits de memoria Flash de programa
- ✓ Hasta 368 bytes de memoria de datos (RAM)
- ✓ Hasta 256 bytes de memoria de datos EEPROM
- ✓ Hasta 4 fuentes de interrupción
- ✓ Stack de hardware de 8 niveles
- ✓ Reset de encendido (POR)
- ✓ Timer de encendido (PWRT)
- ✓ Timer de arranque del oscilador (OST)
- ✓ Sistema de vigilancia Watchdog timer.
- ✓ Protección programable de código
- ✓ Modo SEP de bajo consumo de energía
- ✓ Opciones de selección del oscilador

- ✓ Programación y depuración serie "In-Circuit" (ICSP) a través de dos patitas
- ✓ Lectura/escritura de la CPU a la memoria flash de programa
- ✓ Rango de voltaje de operación de 2.0 a 5.5 volts
- ✓ Alta disipación de corriente de la fuente: 25mA
- ✓ Rangos de temperatura: Comercial, Industrial y Extendido
- ✓ Bajo consumo de potencia:
  - Menos de 0.6mA a 3V, 4 Mhz
  - 20 μA a 3V, 32 Khz
  - menos de 1μA corriente de standby

Periféricos

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

Diagrama de bloques

En la siguiente figura (Fig. 4.6) se muestra a manera de bloques la organización interna del PIC16F877, Se muestra también junto a este diagrama su diagrama de patitas, para tener una visión conjunta del interior y exterior del Chip.

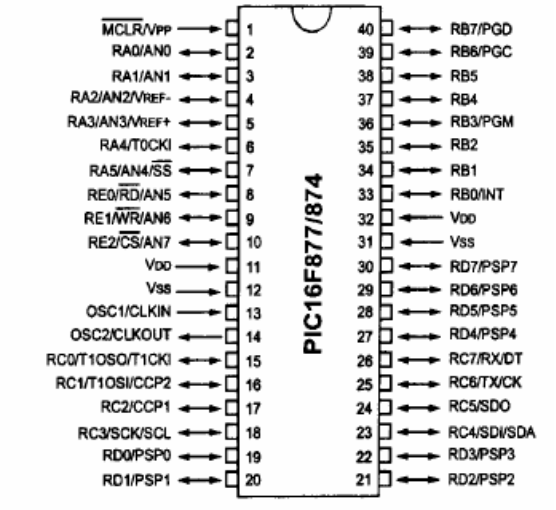


Fig. 4.6.

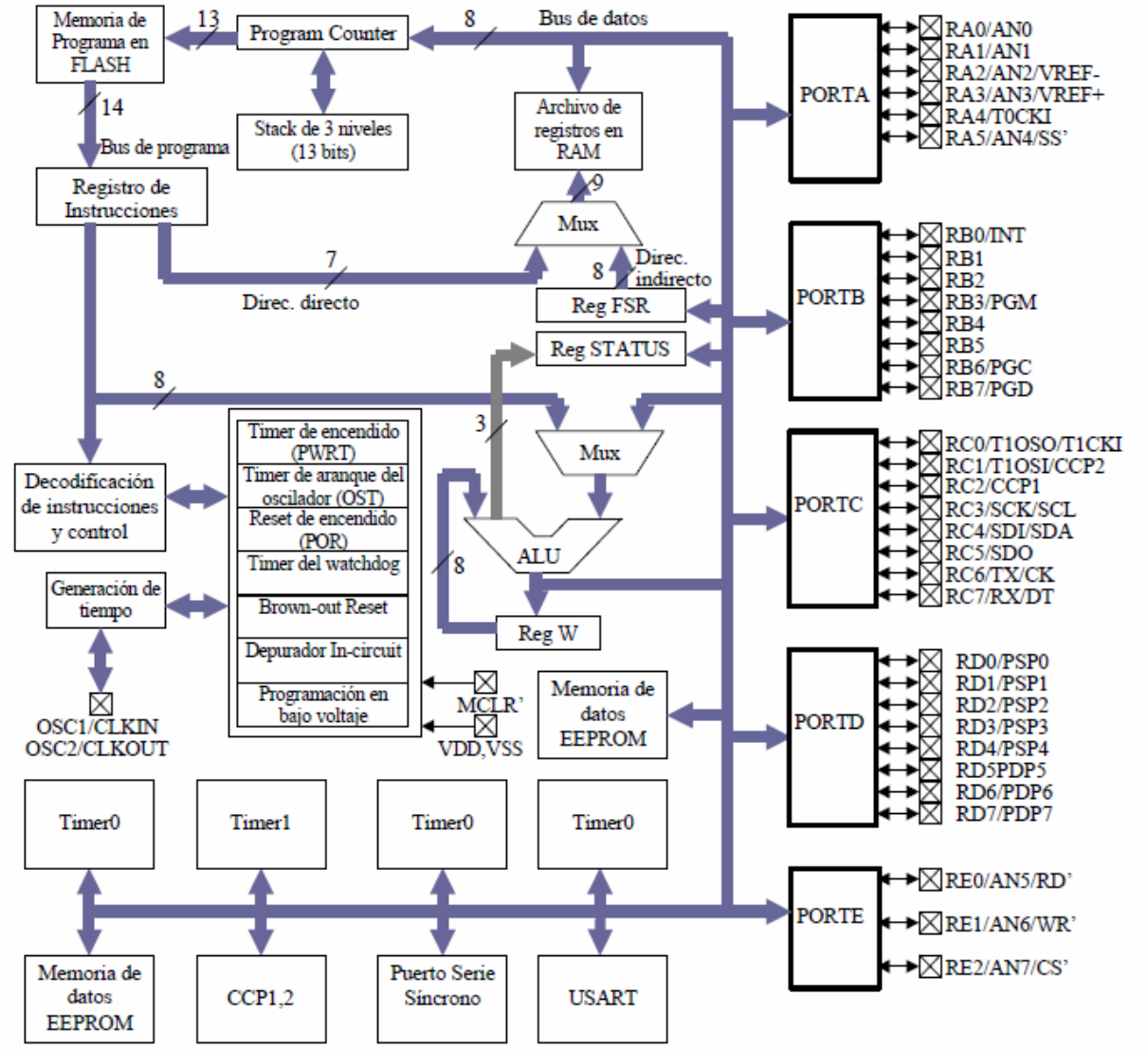


Fig. 4.7.

Descripción de la CPU

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

Ciclo de instrucción

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

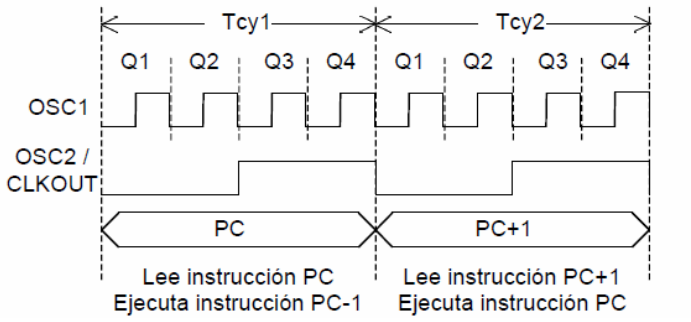


Fig. 4.8.

Como puede verse, cada ciclo de instrucción (Tcy) se compone a su vez de cuatro ciclos del oscilador (Tosc). Cada ciclo Q provee la sincronización para los siguientes eventos:

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

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

Registros de la CPU

Registro PC.- Registro de 13 bits que siempre apunta a la siguiente instrucción a ejecutarse. En la siguiente sección se dan mayores detalles en el manejo de este registro.
Registro de Instrucción.- Registro de 14 bits. Todas las instrucciones se colocan en el para ser decodificadas por la CPU antes de ejecutarlas.
Registro W.- Registro de 8 bits que guarda resultados temporales de las operaciones realizadas por la ALU.
Registro STATUS.- Registro de 8 bits, cada uno de sus bits (denominados ***Banderas***) es un indicador de estado de la CPU o del resultado de la última operación como se indica en la siguiente figura:

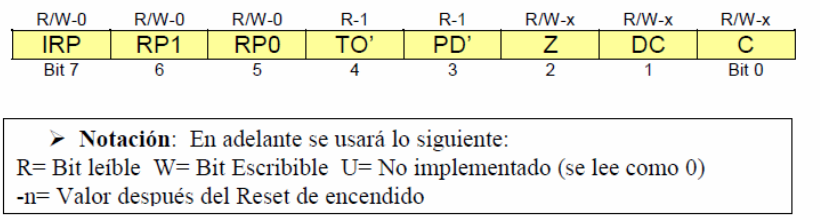


Fig. 4.9.

Z.- Este bit se ***pone*** (=1) para indicar que el resultado de la última operación fue cero, de lo contrario se ***limpia*** (=0).
C.- Bit de acarreo/préstamo' de la última operación aritmética (en el caso de resta, se guarda el préstamo invertido.
CD.- Acarreo/Préstamo' proveniente del cuarto bit menos significativo.
Funciona igual que el bit C, pero para operaciones de 4 bits.

Conjunto de Instrucciones

En la siguiente tabla se resumen las 35 instrucciones que reconoce la CPU de los PIC de medio rango, incluyendo su mnemónico, tiempo de ejecución, código de máquina y afectación de banderas:

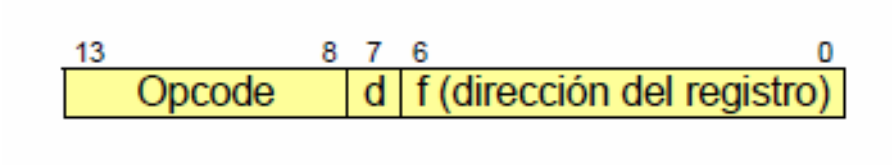
Mnemónico	Descripción	Ciclos	Código de Máquina	Banderas afectadas
Operaciones con el archivo de registros orientadas a bytes				
ADDWF f,d	Suma f + W	1	00 0111 dfff ffff	C,DC,Z
ANDWF f,d	W AND f	1	00 0101 dfff ffff	Z
CLRF f	Limpia f	1	00 0001 1fff ffff	Z
CLRW	Limpia W	1	00 0001 0xxx xxxx	Z
COMF f,d	Complementa los bits de f	1	00 1001 dfff ffff	Z
DECF f,d	Decrementa f en 1	1	00 0011 dfff ffff	Z
DECFSZ f,d	Decrementa f, escapa si 0	1(2)	00 1011 dfff ffff	
INCF f,d	Incrementa f en 1	1	00 1010 dfff ffff	Z
INCFSZ f,d	Incrementa f, escapa si 0	1(2)	00 1111 dfff ffff	
IORWF f,d	W OR f	1	00 0100 dfff ffff	Z
MOVF f,d	Copia el contenido de f	1	00 1000 dfff ffff	Z
MOVWF f	Copia contenido de W en f	1	00 0000 1fff ffff	
NOP	No operación	1	00 0000 0xx0 0000	
RLF f,d	Rota f a la izquierda	1	00 1101 dfff ffff	C
RRF f,d	Rota f a la derecha	1	00 1100 dfff ffff	C
SUBWF f,d	Resta f – W	1	00 0010 dfff ffff	C,DC,Z
SWAPF f,d	Intercambia nibbles de f	1	00 1110 dfff ffff	
XORWF f,d	W EXOR f	1	00 0110 dfff ffff	Z
Operaciones con el archivo de registros orientadas a bits				
BCF f,b	Limpia bit b en f	1	01 00bb bfff ffff	
BSF f,b	Pone bit b en f	1	01 01bb bfff ffff	
BTFSC f,b	Prueba bit b en f, escapa si 0	1(2)	01 10bb bfff ffff	
BTFSS f,b	Prueba bit b en f, escapa si 1	1(2)	01 11bb bfff ffff	
Operaciones con literales y de control				
ADDLW k	Suma literal k + W	1	11 111x kkkk kkkk	C,DC,Z
ANDLW k	k AND W	1	11 1001 kkkk kkkk	Z
CALL k	Llamado a subrutina	2	10 0kkk kkkk kkkk	
CLRWDT	Limpia timer del watchdog	1	00 0000 0110 0100	TO',PD'
GOTO k	Salto a la dirección k	2	10 1kkk kkkk kkkk	
IORLW k	k OR W	1	11 0000 kkkk kkkk	Z
MOVLW k	Copia literal a W	1	11 00xx kkkk kkkk	
RETFIE	Retorna de interrupción	2	00 0000 0000 1001	
RETLW k	Retorna con literal k en W	2	11 01xx kkkk kkkk	
RETURN	Retorna de subrutina	2	00 0000 0000 1000	
SLEEP	Activa Modo standby	1	00 0000 0110 0011	TO'PD'
SUBLW k	Resta k - W	1	11 110x kkkk kkkk	C,CD,Z
XORLW k	k EXOR W	1	11 1010 kkkk kkkk	Z

Tabla 4.5.

Formato General de las Instrucciones

Cada instrucción en lenguaje de máquina (binario) del PIC contiene un código de operación (*opcode*) el cual puede ser de 3 a 4 o 6 bits, dependiendo del tipo de instrucción.
A continuación se describe el formato para cada tipo de instrucción de los PIC de rango medio:

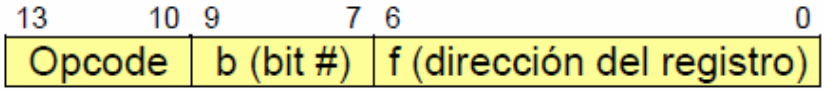
Operaciones con el archivo de registros orientadas a bytes



El bit **d** especifica el destino del resultado de la operación:

d = 0: destino W
d = 1 : destino f
f = Dirección de 7 bits del archivo de registros

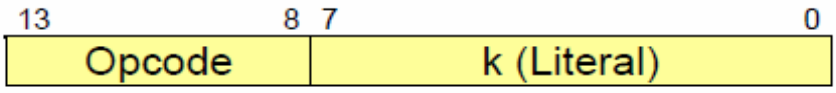
Operaciones con el archivo de registros orientadas a bits



b = Especificación en tres bits del bit sobre el que se va a operar
f = Dirección de 7 bits del archivo de registros

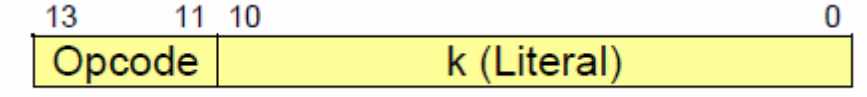
Operaciones con literales y de control

Formato general:



k : Literal = Valor de un operando de 8 bits

Formato para CALL y GOTO:



k : Literal = Valor de un operando de 8 bits

Organización de la memoria del PIC

Los PIC tienen dos tipos de memoria: Memoria de Datos y Memoria de programa, cada bloque con su propio bus: Bus de datos y Bus de programa; por lo cual cada bloque puede ser accesado durante un mismo ciclo de oscilación.
La Memoria de datos a su vez se divide en:

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

La Memoria de Programa

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

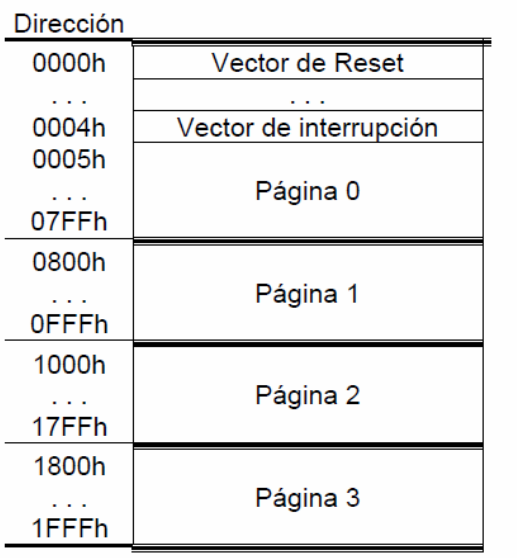
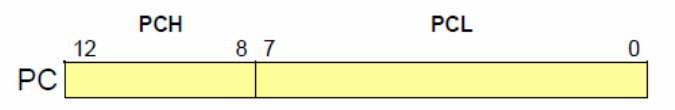


Fig. 4.10.

Observación 1: No todos los PIC tienen implementado todo el espacio de 8K de memoria de programa (Consultar las hojas de datos del PIC específico).
Observación 2: El fabricante puede grabar datos de calibración en localidades de memoria de programa por lo que se deberán anotar en papel antes de borrar los dispositivos con ventana transparente.
Vector de Reset.- Cuando ocurre un reset el contenido del PC es forzado a cero, ésta es la dirección donde la ejecución del programa continuará después del reset, por ello se le llama "**dirección del vector de reset**".
Vector de interrupción.- Cuando la CPU acepta una solicitud de interrupción ejecuta un salto a la dirección 0004h, por lo cual a esta se le conoce como "*dirección del vector de interrupción*". El registro PCLATH no es modificado en esta circunstancia, por lo cual habrá que tener cuidado al manipular el PC dentro de la Rutina de Atención a la Interrupción (Interrupt Service Routine (ISR)).

Manejo del Contador del Programa (PC)

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



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

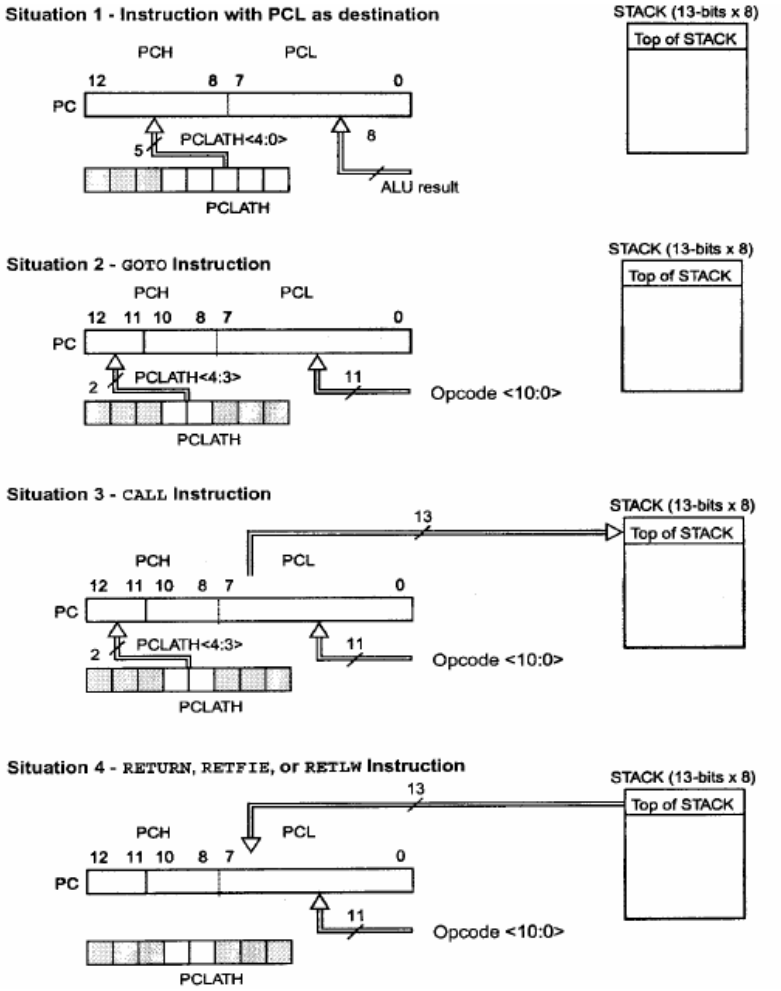


Fig. 4.11.

- **Nota**: PCLATH nunca es actualizado con el contenido de PCH.

Memoria de Stack

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

- ✓ **Nota 1**: PCLATH no se modifica en ninguna de estas operaciones
- ✓ **Nota 2**: Cuando el apuntador de stack ya está en la posición 8 y se ejecuta otro CALL se reinicia a la posición 1 sobrescribiendo en dicha posición. No existe ningún indicador que avise de esta situación. Así que el usuario deberá llevar el control para que esto no ocurra.

La Memoria de Datos

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

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

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

Bancos de memoria

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

La selección de bancos se basa en la siguiente tabla:

Direccionamiento Indirecto (IRP)	RP1:RP0	Banco
0	0 0	0
	0 1	1
1	1 0	2
	1 1	3

Tabla 4.6.

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

Direccionamiento Directo

Para acceder una posición de memoria mediante direccionamiento directo, la CPU simplemente usa la dirección indicada en los 7 bits menos significativos del código de operación y la selección de banco de los bits RP1:RP0 como se ilustra en la siguiente figura.

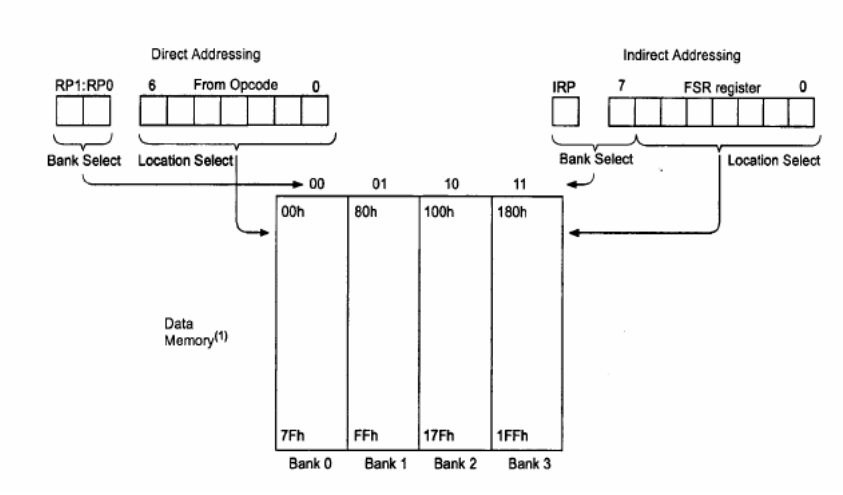


Fig. 4.12.

Direccionamiento indirecto

Este modo de direccionamiento permite acceder una localidad de memoria de datos usando una dirección de memoria *variable* a diferencia del direccionamiento directo, en que la dirección es fija. Esto puede ser útil para el manejo de tablas de datos.
El registro INDF.- En la figura anterior se muestra la manera en que esto se realiza. Para hacer posible el direccionamiento indirecto se debe usar el registro INDF. Cualquier instrucción que haga un acceso al registro INDF en realidad accede a la dirección apuntada por el registro FSR (File Select Register).
La selección de banco en el caso de direccionamiento indirecto se realiza mediante los bits IRP (STATUS<7>) y el bit 7 del registro FSR, como se muestra en la figura.
El registro INDF mismo al leerse de manera indirecta (con FSR=0) producirá un cero. Y al escribirse de manera indirecta no es afectado.

El Archivo de Registros

Aunque el archivo de registros en RAM puede variar de un PIC a otro, la familia del PIC16F87x coincide casi en su totalidad. En la siguiente figura se muestra a detalle

el mapa de este archivo de registros y su organización en los cuatro bancos que ya se describieron (en la página siguiente):

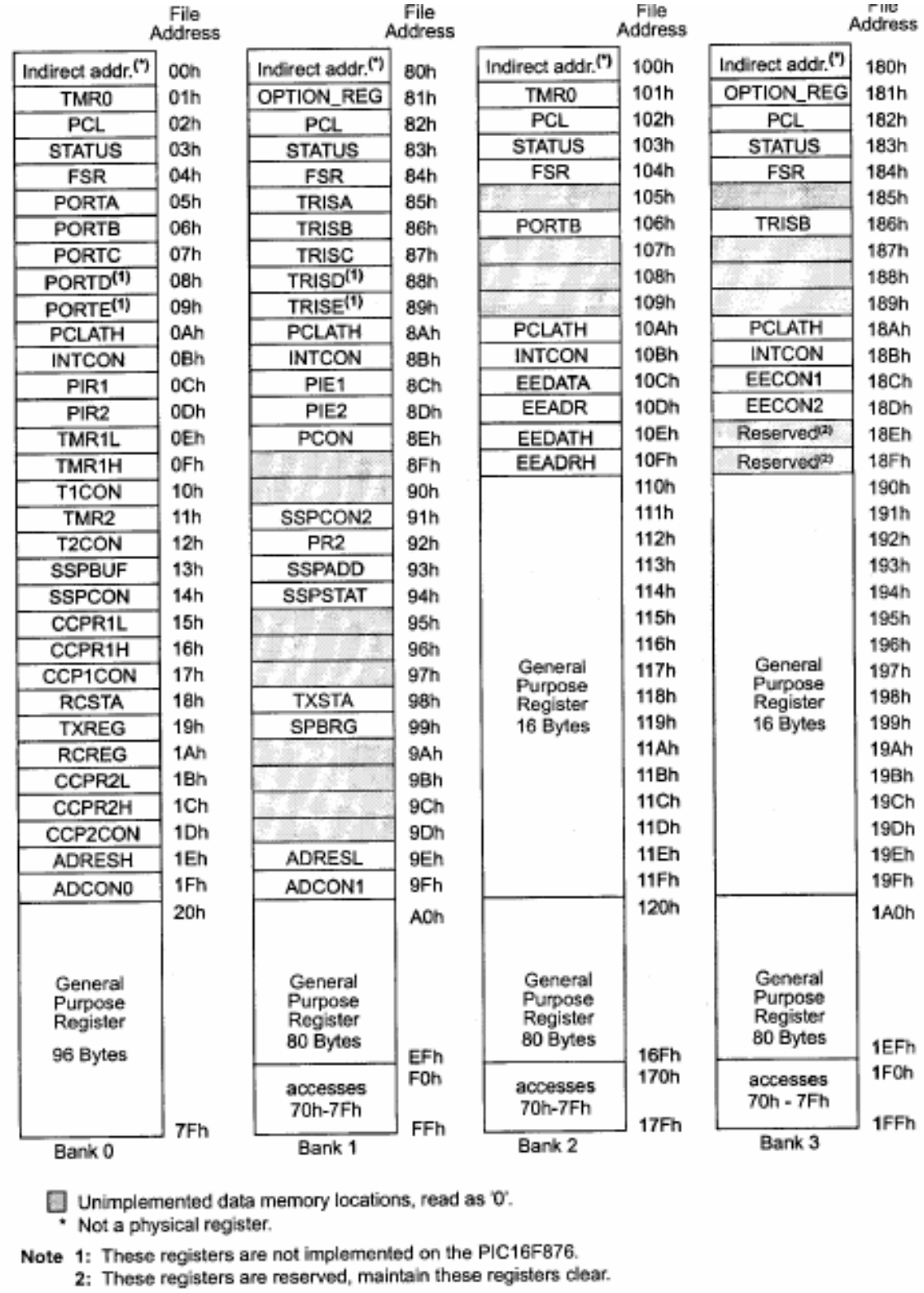


Fig. 4.13.

Paginación

Para saltar entre una página y otra, los bits más significativos del PC deberán ser modificados. Debido a que las instrucciones GOTO y CALL sólo pueden direccionar un bloque de 2K (pues usan una dirección de 11 bits) deben existir otros dos bits que completen los 13 bits del PC para moverse sobre los 8K de memoria de programa.
Estos dos bits extra se encuentran en un SFR denominado PCLATH (Program Counter Latch High) en sus bits PCLATH<4:3>. Por esto antes de un GOTO o un CALL el usuario deberá asegurarse que estos bits apunten a la página deseada.
Si las instrucciones se ejecutan secuencialmente el PC cruza libremente los límites de página sin necesidad de que el usuario escriba en el PCLATH

4.2. Sensor de temperatura / humedad. SHT15

El sensor de temperatura y humedad (Fig 4.14) es el encargado de suministrar toda la información necesaria al microcontrolador, según se le vaya solicitando.
Tiene la capacidad de poder medir dichas variables ambientales y poder transmitirlas para su posterior procesamiento.

Para tal función en el circuito se utilizará el sensor SHT15, de la empresa Sensirion. Este dispositivo incluye un polímero capacitivo como elemento sensor de humedad y un sensor de temperatura bandgap, esto unido a un conversor análogo digital de 14 bits y como salida una interfaz serial con código de redundancia cíclica para la detección de errores.

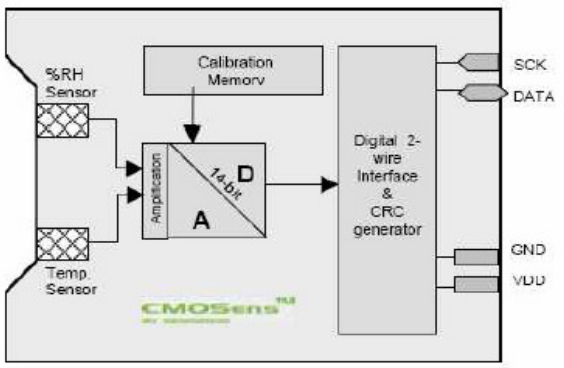


Fig 4.14. *Esquema interno del sensor SHT15*

En los siguientes gráficos se muestra la precisión que tiene el sensor.

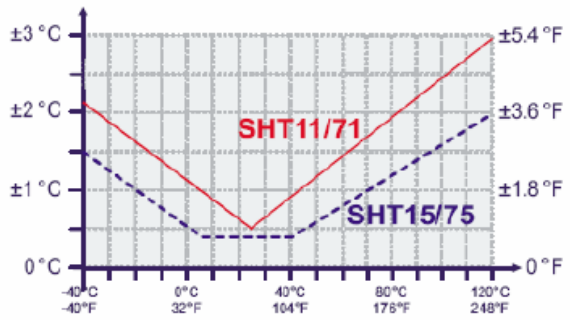


Grafico de comportamiento para la medición de temperatura.

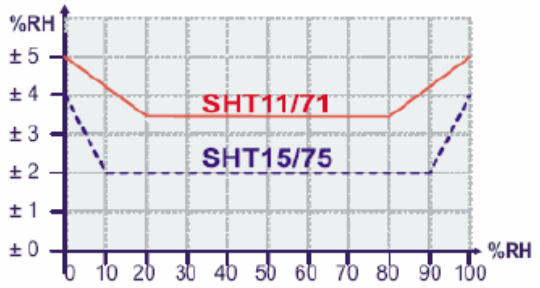


Grafico de comportamiento para la medición de humedad.

Fig. 4.15.

Comunicación con el sensor SHT15

Este sensor ocupa una interfase serial de dos hilos, uno que funciona de clock y otro para los datos:

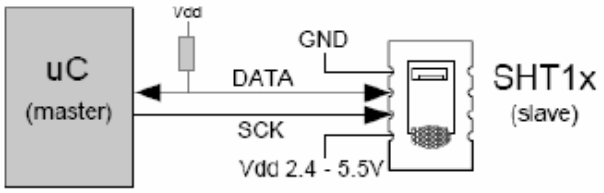


Fig. 4.16. *Conexión del sensor con el microcontrolador.*

Ocupando una comunicación muy similar al I2C, pero no es compatible con interfaces de este tipo de comunicación. La línea de clock se utiliza para sincronizar el microcontrolador y el sensor. La línea de datos se utiliza tanto para dar instrucciones al SHT como para obtener datos de este.
Para iniciar la comunicación con el sensor hay que ingresar una instrucción de la siguiente forma:

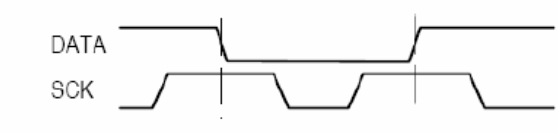


Fig. 4.17 *Trama de inicio del sensor*

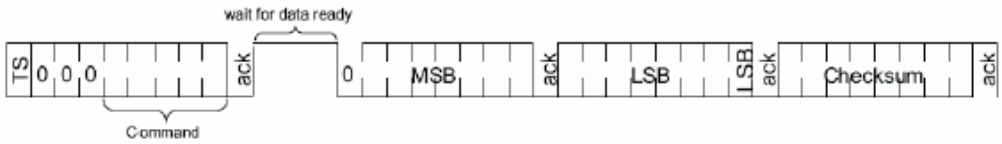
Luego se envían 3 ceros consecutivos seguidos de la instrucción que se desea aplicar:

- ✓ Medición de temperatura O O O I I
- ✓ Medición de humedad O O I O I
- ✓ Leer la condición del registro O O I I I
- ✓ Escribir la condición del registro O O I I O

Para la recepción de la medida, sea de temperatura como de humedad, luego de enviar el código como se describió anteriormente, el controlador tiene que esperar a que se efectué la medida, esto demora 11/55/210 ms para una medición de 8/12/14 bit, lo cual puede variar mas menos 15% según el oscilador interno. Con el fin de señalar la finalización de la medición el SHT15 tira hacia abajo la línea de datos, por lo cual el controlador tiene que esperar esta señal para empezar a cambiar de nuevo el sck.

Trama para la entrega de la medición de humedad. Fig. 4.18.

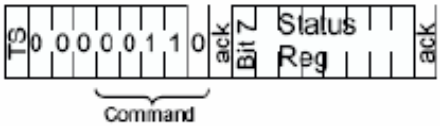
El sensor responde con tres bytes, 2 de datos y uno de corrección de errores o checksum. Esta corrección de errores se hace a través de código de redundancia cíclica. Si es que no se utiliza la comprobación de errores terminaría de transmitir luego de la medición. El sensor vuelve a reposo automáticamente luego de finalizada la medición y la comunicación.



Trama completa de comunicación con el sensor. Fig. 4.19

Registro Status

Algunas de las funciones avanzadas del sensor se encuentran disponibles a través de este registro…



Trama de configuración del registro status
Fig: 4.20.

Bit	Tipo	Descripción	por defecto
7		Reservado	0
6	R	Fin de la bateria(deteccion de bajo voltaje) 0 para Vdd > 2,47V 1 para Vdd < 2,47V	x
5		Reservado	0
4		Reservado	0
3		Solo para pruebas, no utilizar	0
2	R/W	Calentador	0 apagado
1	R/W	no reload from OTP	0 reload
0	R	1 = 8bit Humedad relativa/12bit Temperatura 0 = 12bit Humedad relativa/14bit Temperatura	0 12 bit humedad relativa 14 bit temperatura

Cuadro detallado de los bit del registro status

Tabla 4.7.

Resolución de la medición: El valor por defecto de la temperatura y la humedad es de 14 y 12 bit, lo cual se puede reducir a 12 y 8 bit, siendo especialmente útil para aplicaciones de alta velocidad o muy baja potencia.
Fin de batería: su función es detectar voltajes por debajo de los 2,47V con una precisión más menos 0,05V.
Calentador: es un elemento de calefacción que puede aumentar la temperatura del sensor en aproximadamente 5ºC (9ºF), para este efecto el consumo de energía aumentara en 8mA a 5V.
Esto se puede utilizar para humedades muy altas, evitando la condensación en el sensor, lo cual produciría fallas en su medida. Eso si que mientras se calienta el sensor se muestran temperaturas mas altas y humedad relativa mas baja que sin el calentador.

Características eléctricas

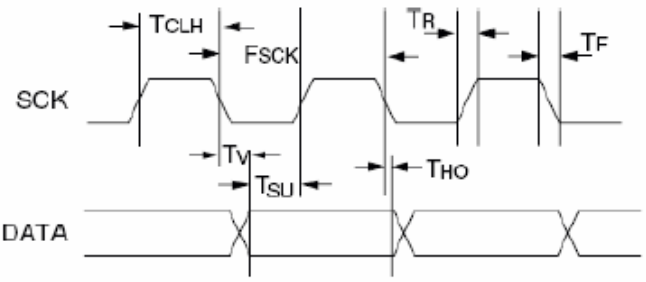
Parámetros	Condiciones	Mínimo	Típico	Máximo	Unidades
Alimentación		2,4	5	5.5	V
Corriente	Midiendo		550		µA
	promedio	2	28		µA
	sleep		0,3	1	µA
Bajo nivel de voltaje de salida		0		20%	Vdd
Alto nivel de voltaje de salida		75%		100%	Vdd
Bajo nivel de voltaje de entrada		0		20%	Vdd
Alto nivel de voltaje de entrada		80%		100%	Vdd

Tabla 4.8.

	Parameter	Conditions	Min	Typ.	Max.	Unit
F_{SCK}	SCK frequency	VDD > 4.5 V			10	MHz
		VDD < 4.5 V			1	MHz
T_{RFO}	DATA fall time	Output load 5 pF	3.5	10	20	ns
		Output load 100 pF	30	40	200	ns
T_{CLK}	SCK hi/low time		100			ns
T_V	DATA valid time			250		ns
T_{SU}	DATA set up time		100			ns
T_{HO}	DATA hold time		0	10		ns
T_R/T_F	SCK rise/fall time			200		ns

Tabla de tiempos de respuesta.

Tabla 4.9.



Esquema de tiempos de respuesta. Fig. 4.21.

Conversión a los valores físicos.

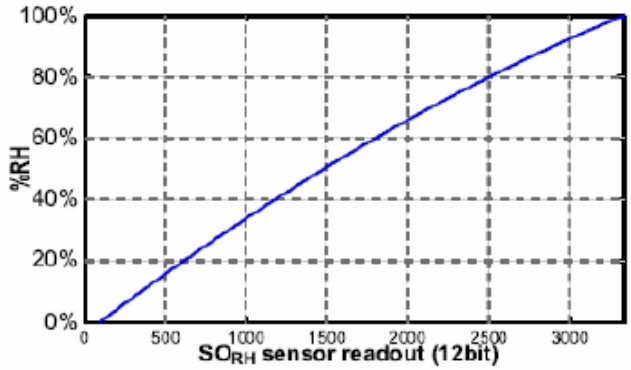
Humedad Relativa: para compensar la no linealidad del sensor de humedad y obtener la plena exactitud se debe aplicar la siguiente formula:

$$RH_{linear} = c_1 + c_2 \bullet SO_{RH} + c_3 \bullet SO_{RH}^{\ 2}$$

SO_{RH}	c_1	c_2	c_3
12 bit	-4	0.0405	$-2.8 * 10^{-6}$
8 bit	-4	0.648	$-7.2 * 10^{-4}$

Formula para calcular la humedad relativa

Estas medidas no tienen dependencia con el voltaje de entrada.



Curva de respuesta de la humedad

Fig. 4.22.

Para temperaturas notoriamente diferentes a los 25ºC se debe aplicar esta otra formula:

$$RH_{true} = (T_{oC} - 25) \bullet (t_1 + t_2 \bullet SO_{RH}) + RH_{linear}$$

SO_{RH}	t_1	t_2
12 bit	0.01	0.00008
8 bit	0.01	0.00128

Formula para calcular la humedad relativa a temperaturas superiores a los 25ºC

Tabla 4.10.

Temperatura: esta medida a diferencia de la anterior es mucho más lineal y basta con aplicar la siguiente formula:

$$\text{Temperature} = d_1 + d_2 \bullet SO_T$$

VDD	d_1 [°C]	d_1 [°f]
5V	-40.00	-40.00
4V	-39.75	-39.50
3.5V	-39.66	-39.35
3V	-39.60	-39.28
2.5V	-39.55	-39.23

SO_T	d_2 [°C]	d_2 [°f]
14bit	0.01	0.018
12bit	0.04	0.072

Formula para calcular la temperatura

Tabla 4.11.

4.3. Conexión inalámbrica: HAC-UM96.

La solución inalámbrica HAC-UM96 aportada al circuito consta de 2 módulos, uno emisor y otro receptor. Su función es la de transmitir la información obtenida a partir de unas mediciones al ordenador.
Uno tendrá la función de enviar la información desde el propio circuito, y la otra, instalada junto al ordenador, deberá recibir de forma correcta dicha información y transmitirla al ordenador vía USART.
A continuación se explican las características de este módulo…

Disposición pines del módulo.

Primero que nada revisemos los pines que se encuentran disponibles en el módulo:

Pin N°	Nombre	Descripción	Nivel	Conectado al Terminal	Notas
1	GND	Tierra de la fuente		Tierra de la fuente	
2	Vcc	Voltaje de Alimentación	+3.3~5.5V		
3	RxD/TTL	Recepción serial de datos	TTL	TxD	COM1
4	TxD/TTL	Transmisión serial de datos	TTL	Rxd	
5	SGND	Tierra de la señal			
6	A(TxD)	A del RS-485 o TxD del RS-232		A(RxD)	COM2
7	B(RxD)	B del RS-485 o RxD del RS-232		B(TxD)	
8	SLEEP	Sleep control (input)	TTL	Señal para ahorro de energía	Baja eficiencia t>15ms
9	RESET	Reset (input)	TTL	Señal para reiniciar el sistema	Aplicar Impulso negativo para reiniciar

Tabla 4.12.

Seleccionando el canal y modo de transmisión de datos.

Antes de comenzar a utilizar el UM96 el usuario necesita hacer algunas configuraciones sencillas basadas en sus propias necesidades.
El módulo UM96 posee un grupo de 5 jumpers (JP2) definidos como ABCDE respectivamente.
Asumiendo que cortocircuitando el jumper es el modo 0 y dejando el jumper sin corto circuito es el modo 1, la configuración del módulo es como sigue:

Selección del Canal

Los jumpers ABC del conector JP2 proveen 8 posiciones las cuales se utilizan para escoger entre los canales 0 a 7.

Canal N°	Frecuencia	Canal N°	Frecuencia
CBA=000(0)	430.2000 Mhz	CBA=100(4)	434.6940 Mhz
CBA=001(1)	431.4288 Mhz	CBA=101(5)	434.2332 Mhz
CBA=010(2)	431.7360 Mhz	CBA=110(6)	433.1580 Mhz
CBA=011(3)	430.5072 Mhz	CBA=111(7)	433.9260 Mhz

Tabla 4.13.

El UM96 provee 2 puertos seriales. El COM1 (Pin3 y PIN4 del JP1) el cual esta fijo como puerto serial USART con niveles TTL y el COM2 (Pin6 y Pin7 del JP1) el cual puede configurarse utilizando el jumper D del JP2
• D = 1 (sin corto circuito) → COM2 = RS-485
• D = 0 (con corto circuito) → COM2 = RS-232

Los datos recibidos desde el aire por el UM96 son entregados simultáneamente en los puertos COM1 y COM2.
Para transmitir datos sólo se puede utilizar uno de los 2 puertos COM. La transmisión no se puede hacerse simultáneamente para ambos puertos.

Selección de la paridad

El UM96 soporta transmisión con paridad Even (8E1) y sin paridad (8N1) lo cual se puede seleccionar con el jumper E del JP2.
• E=1 (sin corto circuito) → Paridad: 8E1
• E=0 (con corto circuito) → Paridad: 8N1

Conectando el UM96 al PC

Para conectar el UM96 al PC primero configuramos el módulo de la siguiente forma:

JP1		JP2	
GND	GND	A	sin cortocircuito
Vcc	+5V	B	sin cortocircuito
TXD	No conectado	C	sin cortocircuito
RXD	No conectado	D	con cortocircuito
SGND	Conectado al PIN 5 del conector DB9 que va al PC (ver figura)	E	con cortocircuito
A(TXD)	Conectado al PIN 2 del conector DB9 que va al PC (ver figura)		
B(RXD)	Conectado al PIN 3 del conector DB9 que va al PC (ver figura)		
SLEEP	No conectado		
RESET	No conectado		

Tabla 4.14.

A continuación se puede observar la distribución de los diferentes *pines* del UM96 conectados al puerto serie del ordenador.

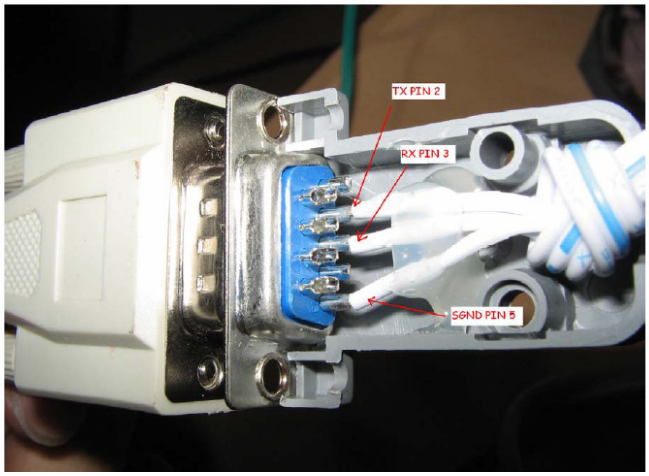


Fig 4.23

En la siguiente imagen se puede observar la distribución de los *jumpers,* anteriormente citados.

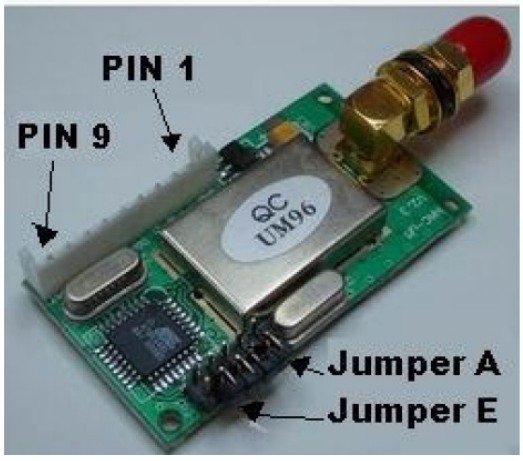


Fig. 4.24.

5. Diseño e implementación del M.R.I.

A continuación se explica el proceso llevado a cabo para el desarrollo del hardware, del software y del firmware.

5.1. Hardware

El diseño e implementación del monitor esta determinado por la realización de una etapa de estudio, que dio como resultado que las variables más relevantes a controlar en un invernadero son la temperatura ambiente, la humedad ambiente y la humedad del suelo (como ya se ha comentado anteriormente, esta última variable no se ha tenido en cuenta); al igual es importante que estas variables puedan ser transmitidas de forma inalámbrica por el tipo de emplazamiento que utilizan los invernaderos en la actualidad. Según estos requerimientos, el monitor estaría dividido en 5 etapas principales:

- ✓ Captura de variables.
- ✓ Ordenar variables.
- ✓ Transmisión.
- ✓ Recepción.
- ✓ Procesar variables.

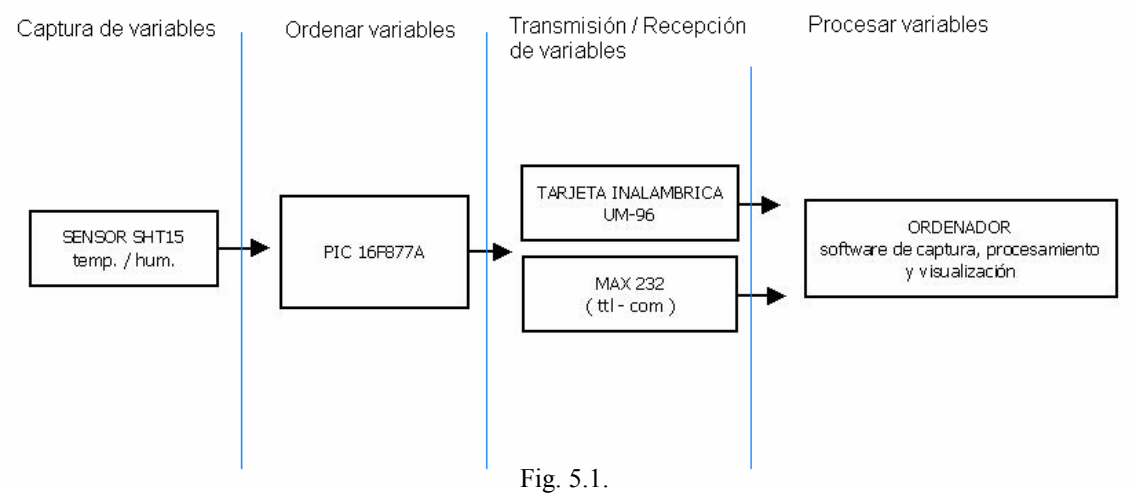


Fig. 5.1.

Captura de variables

Es aquí donde se van a tomar las variables relevantes para el funcionamiento del invernadero, los cuales son: la temperatura ambiente, la humedad ambiente y la humedad del suelo. Aquí se utilizará un sensor que capture estas variables para posteriormente reunirlas en el siguiente módulo.
Se utilizo para la captura de temperatura y humedad relativa el sensor SHT15, que para comunicarse con el siguiente módulo (microcontrolador) utiliza la comunicación I2C.

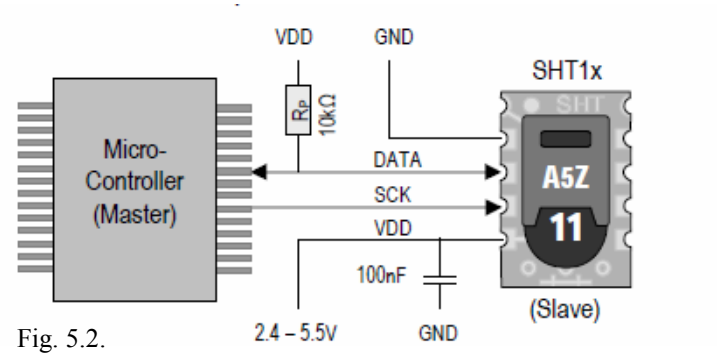


Fig. 5.2.

Ordenación de variables

Es donde se toman las variables y ordenan para su posterior transmisión. Para esta función se utilizará el microcontrolador 16F877A. Esta programado en Assembler, y el programa lo que hace es cada segundo toma los datos que envía el sensor SHT15 y los envía por el puerto serial asíncrono que posee ó USART.

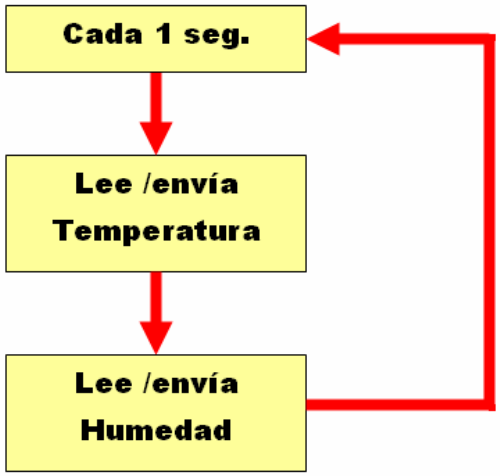


Fig. 5.3.

Transmisión y recepción

Es donde se envían los datos de forma inalámbrica para su posterior recepción, con tarjetas diseñadas para este motivo considerando su flexibilidad, alcance y modulación. Para esto fueron utilizadas las tarjetas UM96, que cumplen con los requerimientos antes mencionados, como lo son la facilidad de conexión tanto al microcontrolador, como a la comunicación RS232; no necesitando hardware aparte para su conexión a estos, aunque por seguridad se utiliza el componente *MAX-232*. Tienen un alcance de 500 metros, que es distancia suficiente para llegar a su central. Tienen varios canales de comunicación, lo que facilita el hecho de que haya varios módulos funcionando a la par, con una modulación GFSK.
La tarjeta de recepción se conecta a un adaptador de RS232 a USB, para garantizar la compatibilidad con ordenadores que no tengan el puerto RS232, como es el caso de los ordenadores actuales.

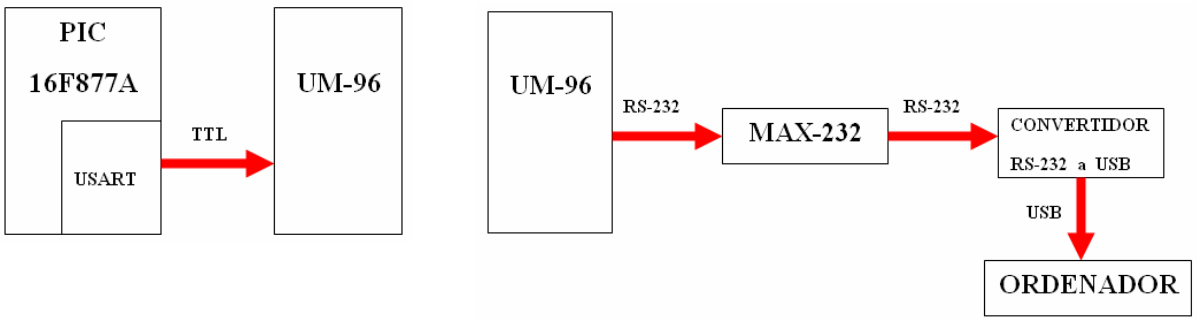


Fig. 5.4.

Procesamiento de variables

En esta última etapa se recepciona los datos para su procesamiento y posterior visualización por pantalla. Para esto se utilizará un ordenador, con un software que sea capaz de recopilar los datos, procesarlos y mostrarlos por pantalla. El software utilizado es Visual Basic, debido a su entorno fácil de utilizar para el usuario.
El programa lo que hace es abrir el puerto serie, que crea virtualmente el cable de adaptación de RS232 a USB, lee los datos que le están entrando, los procesa y los muestra mediante 2 ventanas creadas en la pantalla.

Finalmente, el esquema del circuito queda de la siguiente manera…

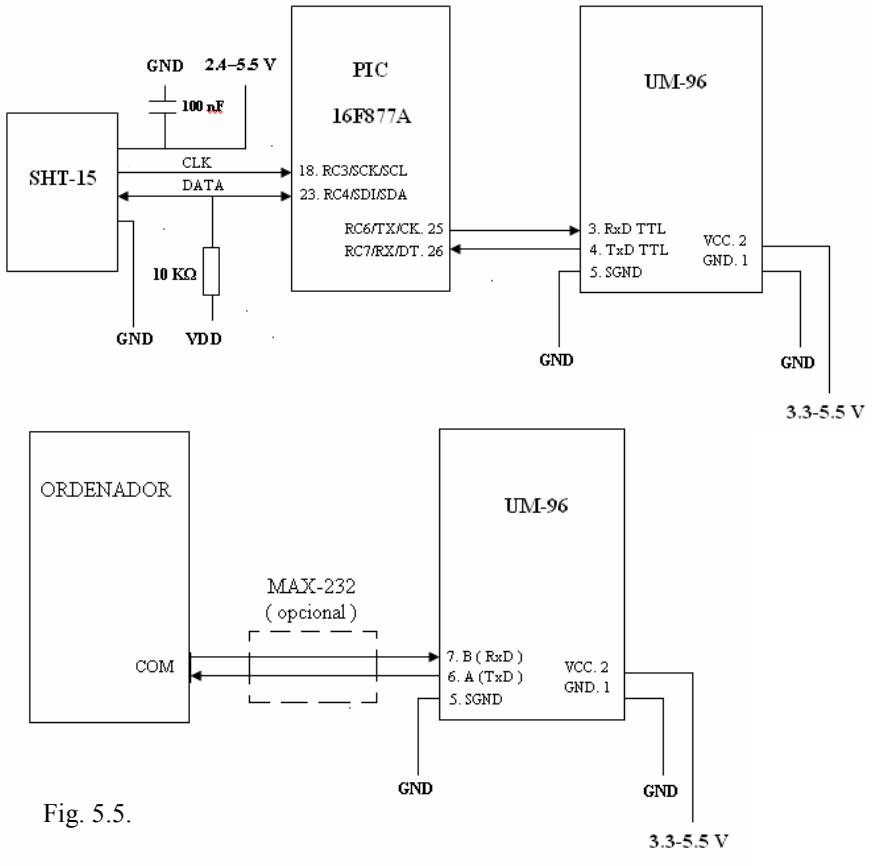


Fig. 5.5.

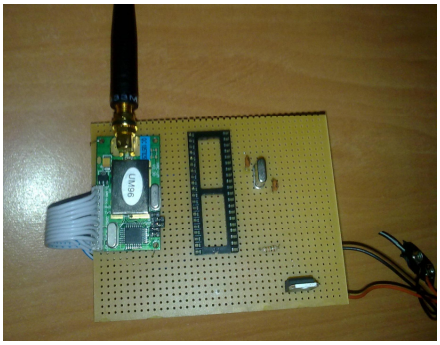


Fig. 5.6 Tarjeta Emisora

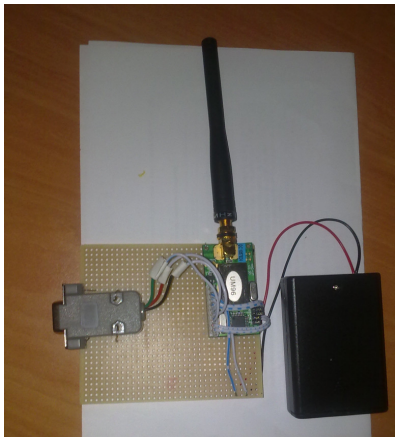


Fig. 5.7 Tarjeta Receptora

Cabe destacar que en la fotografía de la tarjeta emisora no aparece el sensor (SHT15), ya que se ha realizado un cambio de "última hora".
Debido a la inexperiencia en lo que a la soldadura respecta, se ha optado por cambiar el sensor, concretamente al SHT75. El principal motivo radica en la comodidad de soldar dicho sensor a la tarjeta emisora, es decir, la diferencia entre el sensor SHT75 y el SHT15 es el diseño, técnicamente se trata del mismo sensor.

5.2. Software

Para el desarrollo del software se ha utilizado Visual Basic, ya que trabaja sobre un framework ó marco común de librerías independiente de la versión del sistema operativo, .NET Framework, a través de Visual Basic .NET .
Además de facilitar mucho la programación, ya que consta de un editor de código, un depurador, un compilador, y un constructor de interfaz gráfica o GUI.

El control que facilitado aún más el desarrollo de la aplicación ha sido MSCOMM32.OCX, ya que gracias a sus diferentes métodos y funciones se ha conseguido capturar de forma muy cómoda los datos de entrada provenientes del circuito a través del puerto serie.
A continuación se muestra la pantalla donde se visualizan los valores obtenidos en las mediciones de temperatura y humedad.

Fig 5.8.

Como se puede observar, la pantalla donde se visualizan los datos es muy simple, consta de una ventana para mostrar la temperatura, otra ventana para mostrar la humedad, un botón para actualizar dichos valores y un botón para conectar el circuito al puerto serie que se indique (Com1, Com2…etc).

A continuación se muestra el código empleado en el software, así como las explicaciones oportunas.

```
Private Sub Conectar_Click(ByVal eventSender As System.Object, ByVal
eventArgs As System.EventArgs) Handles Conectar.Click

        'comprueba que el puerto este cerrado para poder abrirlo
            If MSComm1.PortOpen = False Then
                  'determina el puerto que hemos seleccionado
                  If Combo1.SelectedIndex = 0 Then
                        MSComm1.CommPort = 1
                        Else
                        MSComm1.CommPort = 2
                  End If
              MSComm1.InBufferSize = 8
              MSComm1.Settings = "9600,N,8,1"
              MSComm1.InputMode =
              MSCommLib.InputModeConstants.comInputModeText

' al recibir los 2 bytes de la temperatura y 1 byte de la humedad.
              MSComm1.RThreshold = 3
              MSComm1.NullDiscard = True
              MSComm1.RTSEnable = False
              MSComm1.DTREnable = True

              MSComm1.InBufferSize = 3
              MSComm1.OutBufferSize = 512

              MSComm1.InputLen = 3
              MSComm1.PortOpen = True
              Me.Text = "Conectado por el puerto " & MSComm1.CommPort

              End If
       End Sub
```

```
    Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal
eventArgs As System.EventArgs) Handles MyBase.Load

        Combo1.Items.Add("Com1")
        Combo1.Items.Add("Com2")
        Combo1.SelectedIndex = 0
    End Sub
```

```
    'El evento OnComm se genera siempre que cambia el valor de la
propiedad 'CommEvent e indica que se ha producido un evento o un error
en la comunicación.
      En este caso, el código de esta subrutina es idéntica a la del
botón Actualizar


    Private Sub MSComm1_OnComm(ByVal eventSender As System.Object,
ByVal eventArgs As System.EventArgs) Handles MSComm1.OnComm
        Dim i As Short
        Dim Valor As String
        Dim Entrada As String
        Dim Temp_Alta_Bin As String
        Dim Temp_Baja_Bin As String
        Dim Temp_Alta_Dec As Integer
        Dim Temp_Baja_Dec As Integer
        Dim Tempe As Integer
        Dim Tempe_Valida As Integer
        Dim Hume_Bin As String
        Dim Hume_Dec As Integer
        Dim Hume_Linear As Integer
        Dim Hume_Compensada As Integer


        Select Case MSComm1.CommEvent
            Case 2

' Almacenamos en las variables de entrada los bytes leídos

            Entrada = MSComm1.Input
            Temp_Alta_Bin = Entrada (23) & Entrada (22) & Entrada (21)
            & Entrada (20) & Entrada (19) & Entrada (18) & Entrada (17)
            & Entrada (16)
            Temp_Baja_Bin = Entrada (15) & Entrada (14) & Entrada (13)
            & Entrada (12) & Entrada (11) & Entrada (10) & Entrada (9)
            & Entrada (8)

' Convertimos dichas variables de entrada a número entero
            Temp_Alta_Dec = Cint(Temp_Alta_Bin)
            Temp_Baja_Dec = Cint(Temp_Baja_Bin)

' Obtenemos el valor número del contenido de 2 bytes
            Tempe = ( Temp_Alta_Dec * 256 ) + Temp_Baja_Dec

' Finalmente se obtiene la temperatura real
            Tempe_Valida = (-40.1) + (0.04*Tempe)

' Almacenamos en la variable de entrada el byte leído
            Hume_Bin = Entrada (7) & Entrada (6) & Entrada (5) &
            Entrada (4) & Entrada (3) & Entrada (2) & Entrada (1) &
            Entrada (0)

' Convertimos dicha variable de entrada a número entero
            Hume_Dec = Cint(Hume_Bin)

' Finalmente se obtiene la humedad real ( compensada )
            Hume_Linear = (-2.0468) + (0.5872*Hume_Dec) +
            ((-4.0845 ^ (-4))*Hume_Dec^2)

            Hume_Compensada = (Tempe_Valida-25) *
            ((0.01+(0.00128*Hume_Dec))-Hume_Linear
```

```
' Se muestra por pantalla dichos valores, es decir, la medición real
            Tempe.Text = Tempe_Valida
            Hume.Text = Hume_Compensada

        End Select
    End Sub
```

5.3. Firmware

Se conoce como *firmware* el programa, en este caso programado en Assembler, que lleva grabado el microcontrolador para que lleve a cabo todas aquellas funciones que deseamos que realice en el circuito, junto a los demás componentes.

5.3.1. Programación

A continuación se muestra un diagrama (Fig. 5.9.) donde se expresa gráficamente el algoritmo desarrollado en el firmware.
Como se observa, trás el inicio del mismo, se procede a configurar tanto el sensor como el microcontrolador para que puedan comunicarse. Acto seguido, se le manda un byte de *Start* para establecer una comunicación entre ambos. Una vez comunicados, se obtienen los valores de temperatura, primero, y de humedad, segundo, para su tratamiento y visualización. Con el fin de cerrar la comunicación se envía un byte de *Stop*.
El bucle, en principio infinito, vuelve a empezar enviando un byte de *Start*, creando de esta forma otra comunicación con el sensor, y así sucesivamente.

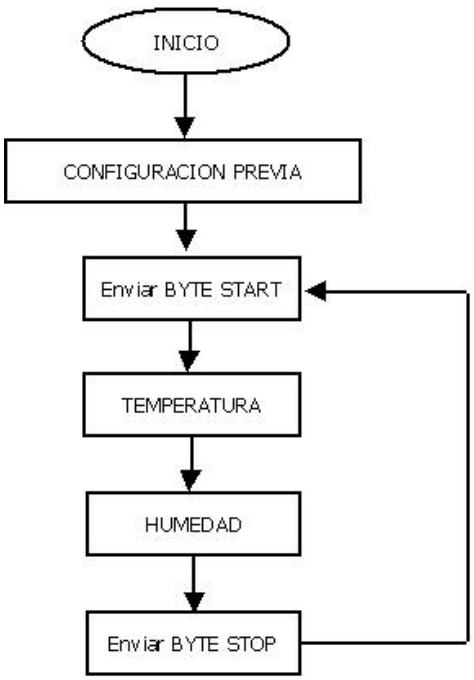


Fig. 5.9.

A. MODULO "CONFIGURACION PREVIA"

Parte de código por el cual quedan configurados los modos USART e I2C del microcontrolador para poder ser utilizados tanto en la medición de valores (I2C) como en la transmisión de dichos valores (USART), y el registro interno del sensor SHT15 (Register Status).

1.- Código empleado en el programa principal…

```
CALL conf_I2C
CALL conf_USART
CALL conf_SHT15
```

2.- Código empleado en las subrutinas…

conf_I2C

```
BCF STATUS, RP0                 ; Acceso a banco 00
BCF STATUS, RP1
CLRF PORTC                      ; Limpia las puertas C
BSF STATUS, RP0                 ; Acceso a banco 01
BCF STATUS, RP1
MOVLW b'11111111'               ;Configura todas las puertas C como salidas
MOVWF TRISC
MOVLW b'11001111'               ;Predivisor de 128 asociado al watchdog
MOVWF OPTION_REG
MOVLW b'10000000'               ; Velocidad standard con niveles i2C
MOVWF SSPSTAT
MOVWL .9                        ; Velocidad del bus I2C a 100 KHz
MOVWF SSPADD
BCF STATUS, RP0                 ; Acceso a banco 00
MOVLW b'00101000'               ; Configuración del registro SSPCON
MOVWF SSPCON
RETURN
```

conf_USART

```
BSF     STATUS,RP0              ; Acceso a banco 01
BCF     STATUS,RP1
MOVLW   b'10000000'             ; RC7/RX entrada, RC6/TX salida
MOVWF   TRISC
MOVLW   b'00100000'             ; Configuración del registro USART
```

```
MOVFW   TXSTA
MOVLW   .33                     ; Asignación de 9600 baudios
MOVFW   SPBRG
BCF     STATUS,RP0              ; Acceso al banco 00
MOVLW   b'10000000'             ; Configuración del registro RCSTA
MOVWF   RCSTA
RETURN
```

conf_SHT15

```
CALL    byte_start              ; Se inicia la comunicación con el sensor
MOVLW   b‘00000110’             ; Se accede al sensor y se le comunica que
CALL    send_byte               ; se va a escribir en su registro interno
MOVLW   b‘00000001’             ; Configuración de su registro interno
CALL    send_byte
CALL    byte_stop               ; Se finaliza la comunicación con el sensor
RETURN
```

Mención aparte requiere la configuración del registro interno del sensor.
El primer byte enviado es *00000110* , es decir, los primeros 3 ceros referencia la dirección del dispositivo en cuestión, el sensor, que al ser único esclavo, tiene como dirección *000*, y el siguiente bloque *00110* significa que se va a escribir en el registro interno del sensor ‘ Status Register’.
El siguiente byte enviado es *00000001* , es decir, cada uno de los 8 bits que componen el registro interno del sensor. Queda configurado el sensor de manera que el valor de temperatura lo obtendremos en forma de 8 bits y el de humedad en forma de 12 bits.

B. MODULO “Enviar BYTE START”

Se trata, por una parte, de configurar los diferentes registros del PIC 16F877A y, por otra, mandar una secuencia de bits para comunicar con el sensor (Fig. 5.10.).

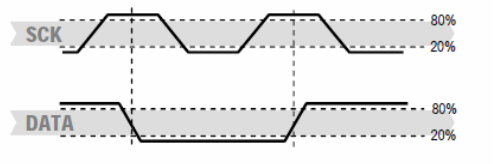


Fig. 5.10.

1.- Código empleado en el programa principal…

CALL BYTE_START

2.- Código empleado en la subrutina…

```
BYTE_START
        ;
        ; Configuración de los diferentes registros del PIC 16F877A
        ;
        BCF PIR1, SSPIF                 ; Desactiva el bit SSPIF del registro PIR1
        BSF STATUS, RP0                 ; Acceso al banco 01
        BCF STATUS, RP1
        BSF SSPCON2, SEN                ; Activa la secuencia de inicio
        BCF STATUS, RP0                 ; Acceso al banco 00

WAIT
        BTFSS PIR1, SSPIF               ; Espera a que termine de transmitir la
                                        secuencia
        GOTO WAIT

        BSF   STATUS,RP0                ; Acceso a banco 01
        BCF   TRISC,4                   ; RC4 = salida ( reloj CLK con el sensor )
        BCF   STATUS,RP0                ; Acceso a banco 00
        ;
        ; Comunicación con el sensor
        ;
        BSF   PORTC,3                   ; SDA = 1
        BSF   PORTC,4                   ; SCK = 1
        BCF   PORTC,3                   ; SDA = 0
        BCF   PORTC,4                   ; SCK = 0
        BSF   PORTC,4                   ; SCK = 1
        BSF   PORTC,3                   ; SDA = 1
        BCF   PORTC,4                   ; SCK = 0
        RETURN
```

C. MODULO “TEMPERATURA”

Tal y como se ha comentado anteriormente, en esta parte del programa lo que se ha desarrollado es:

- ✓ Comunicar al sensor que se quiere leer el valor de temperatura,
- ✓ La propia lectura de la medición obtenida y
- ✓ Su posterior envío al ordenador para poder ser visualizado.

Petición de medición.

Para comunicar al sensor que se quiere recibir el valor de temperatura, bastará con enviar un byte con el código correspondiente, es decir, *00000011*.

Lectura de las mediciones.

Tras una espera teórica de 80 ms (lo que tarda el sensor en realizar la medición para 12 bits) se habrá recibido en el buffer de entrada, *SSPBUF*, el valor de la parte alta de la medición de la temperatura, al cual accederemos para copiar el valor en una variable temporal (ej. TMP_MSB). Tras enviar el ACK correspondiente, se vuelve a acceder para almacenar el nuevo valor de *SSPBUF* (en este caso, la parte baja de la medición) en otra variable temporal (ej. TMP_LSB). Por último, y tras mandar el correspondiente ACK, volveremos a acceder para leer el CHECKSUM recibido. Este último no hace falta almacenarlo, pero si el envío del ACK.
Cada vez que se accede al registro *SPPBUF* para ser leído, éste se vacía y "da paso" a que se almacene un nuevo valor.
De esta forma, se tiene en 2 variables temporales la medición obtenida de la temperatura.

Envío de las mediciones.

Ya que se cuenta con el modo USART debídamente configurado, se procede a copiar, primero, el contenido de la primera variable temporal al registro W del microcontrolador y mandarlo vía USART al ordenador y, seguídamente, copiar el contenido de la segunda variable temporal al registro W para mandarlo también al ordenador.

A continuación, y para mejor entendimiento, se muestra el pseudocódigo de dichos procesos:

CABECERA PROGRAMA

CONFIGURACION PREVIA

ENVIAR BYTE_START comunicación con el sensor

```
Enviar_BYTE ( 00000011 )        ; Comunicar al sensor que se quiere leer la temperatura
Esperar a recibir el ACK
Esperar_Tiempo_Medicion         ; al menos los teóricos 80 ms

Leer Buffer_Entrada             ; Leer y Almacenar la parte alta del valor
Enviar ACK
Almacenar valor en TMP_MSB

Leer Buffer_Entrada             ; Leer y Almacenar la parte baja del valor
Enviar ACK
Almacenar valor en TMP_LSB

Leer Buffer_Entrada             ;  Leer el Checksum. No hace falta almacenar.
Enviar ACK

Enviar TMP_MSB                  ; Enviar las 2 variables temporales vía USART al pc
Enviar TMP_LSB
```

1.- Código empleado en el programa principal…

CALL TEMPERATURA

2.- Código empleado en la subrutina…

```
TEMPERATURA
        ;
        ; Rutina empleada para la medición y posterior envío de la temperatura
        ;
        MOVLW   b'00000011'     ; Comunicar que se quiere leer la temperatura
        CALL    send_byte

        CALL    read_byte       ; Se lee la parte alta y se almacena en TMP_MSB
        MOVWF   TMP_MSB
        CALL    read_byte       ; Se lee la parte baja y se almacena en TMP_LSB
        MOVWF   TMP_LSB
        CALL    read_byte       ; Se lee el CHECKSUM

        MOVF    TMP_MSB         ; Se copia el contenido de TMP_MSB al registro W
        CALL    tx_dato         ; y se manda al ordenador vía USART

        MOVF    TMP_LSB         ; Se copia el contenido de TMP_LSB al registro W
        CALL    tx_dato         ; y se manda al ordenador vía USART
```

```
        CALL    delay               ; retardo de 1 segundo

send_byte
        ;
        ; Rutina empleada para mandar un byte vía I2C
        ;
        BCF     PIR1, SSPIF         ; Desactiva el ACK
        MOVWF   SSPBUF              ; LLeva el dato al buffer de salida

send_byte_w
        BTFSS   PIR1, SSPIF         ; ACK recibido ??
        GOTO    send_byte_w         ; No. Esperar
        RETURN


read_byte
        ;
        ; Rutina empleada para leer un byte vía I2C
        ;
        BCF     PIR1, SSPIF         ; Limpia el bit de recepción completa
        BSF     STATUS, RP0         ; Acceso a banco 01
        BCF     STATUS, RP1
        BSF     SSPCON2, RCEN       ; Activa modo receptor
        BCF     STATUS, RP0         ; Acceso a banco 00

read_byte_w
        BTFSS   PIR1, SSPIF         ; ¿ Byte recibido ??
        GOTO    read_byte_w         ; No. Esperar
        BCF     PIR1, SPPIF         ; Limpia el bit de recepción completa
        BSF     STATUS, RP0         ; Acceso a banco 01
        BCF     STATUS, RP1
        BCF     SPPCON2, ACKDT      ; Limpia el bit ACK
        BSF     SSPCON2, ACKEN      ; Inicia la generación del ACK
        BCF     STATUS, RP0         ; Acceso a banco 00

ack_w
        BTFSS   PIR1, SSPIF         ; ¿ ACK enviado ??
        GOTO    ack_w               ; No. Esperar
        MOVF    SSPBUF, W           ; Copia al registro W el contenido del buffer
        RETURN                      ; de entrada

tx_dato
        ;
        ; Rutina empleada para enviar un byte vía USART
        ;
        BCF     PIR1,TXIF           ; Limpia el bit de transmisión completa
        MOVWF   TXREG               ; Copia el byte a transmitir al buffer de salida
        BSF     STATUS,RP0          ; Acceso a banco 01
```

```
        BCF         STATUS,RP1

tx_dat_w
        BTFSS       TXSTA,TRMT          ; ¿Byte transmitido ??
        GOTO        tx_dat_w            ; No, esperar
        BCF         STATUS,RP0          ; Si, vuelta a Banco 00
        RETURN
```

En la cabecera del programa quedan definidas las distintas variables temporales utilizadas, de la siguiente forma…

```
TMP_MSB     EQU   0x20
TMP_LSB     EQU   0x21
```

C. MODULO "HUMEDAD"

De forma análoga al apartado de la temperatura, y tal y como se ha comentado anteriormente, en esta parte del programa lo que se ha desarrollado es:

- ✓ Comunicar al sensor que se quiere leer el valor de humedad,
- ✓ La propia lectura de la medición obtenida y
- ✓ Su posterior envío al ordenador para poder ser visualizado.

Petición de medición.

Para comunicar al sensor que se quiere recibir el valor de temperatura, bastará con enviar un byte con el código correspondiente, es decir, *00000101*.

Lectura de las mediciones.

Tras una espera teórica de 20 ms (lo que tarda el sensor en realizar la medición para 8 bits) se habrá recibido en el buffer de entrada, *SSPBUF*, el valor de la medición de la humedad, al cual accederemos para copiar el valor en una variable temporal, por ejemplo, TMP_HUM. Tras enviar el ACK correspondiente, se vuelve a acceder para leer el CHECKSUM recibido. Este último no hace falta almacenarlo, pero si el envío del ACK.
Cada vez que se accede al registro *SPPBUF* para ser leído, éste se vacía y "da paso" a que se almacene un nuevo valor.
De esta forma, se tiene en una variable temporal la medición obtenida de la humedad.
Envío de las mediciones.

Ya que se cuenta con el modo USART debídamente configurado, se procede a copiar, primero, el contenido de la variable temporal al registro W del microcontrolador y mandarlo vía USART al ordenador.

A continuación, y para mejor entendimiento, se muestra el pseudocódigo de dichos procesos…


```
Enviar_BYTE ( 00000101 )          ; Comunicar al sensor que se quiere leer la humedad
Esperar a recibir el ACK
Esperar_Tiempo_Medicion           ; al menos los teóricos 20 ms

Leer Buffer_Entrada               ; Leer y Almacenar el valor medido
Enviar ACK
Almacenar valor en TMP_HUM

Leer Buffer_Entrada               ;  Leer el Checksum. No hace falta almacenar.
Enviar ACK

Enviar TMP_HUM                    ; Enviar la variable temporal vía USART al pc
```

1.- Código empleado en el programa principal…

CALL HUMEDAD

2.- Código empleado en la subrutina…

```
HUMEDAD
        ;
        ; Rutina empleada para la medición y posterior envío de la humedad
        ;
        MOVLW   b'00000101'     ; Comunicar que se quiere leer la humedad
        CALL    send_byte

        CALL    read_byte       ; Se lee el valor medido y se almacena en
                                ; TMP_HUM
        MOVWF   TMP_HUM
        CALL    read_byte       ; Se lee el CHECKSUM

        MOVF    TMP_HUM         ; Se copia el contenido de TMP_HUM al reg. W
        CALL    tx_dato         ; y se manda al ordenador vía USART

        CALL     delay          ; retardo de 1 segundo
```

Las *rutinas read_byte, send_byte* y *tx_dat* se han adjuntado en el apartado de la temperatura.
En la cabecera del programa queda definida la variable temporal utilizada, de la siguiente forma…

```
TMP_HUM     EQU   0x22
```

D. MODULO "Enviar BYTE STOP"

Solo requiere configuración alguna por parte del PIC 16F877A.

1.- Código empleado en el programa principal…

```
    CALL BYTE_STOP
```

2.- Código empleado en la subrutina…

```
BYTE_STOP
        ;
        ; Configuración de los diferentes registros del PIC 16F877A
        ;
        BCF PIR1, SSPIF                 ; Desactiva el bit SSPIF del registro PIR1
```

```
        BSF STATUS, RP0          ; Acceso al banco 01
        BCF STATUS, RP1
        BSF SSPCON2, PEN         ; Activa la secuencia de Stop
        BCF STATUS, RP0          ; Acceso al banco 00
WAIT
        BTFSS PIR1, SSPIF        ; Espera a que termine de transmitir la
                                 ; secuencia
        GOTO WAIT
        RETURN
```

5.3.2. Herramientas utilizadas

A. MPLAB

Este es un software de la empresa Arizona Microchip Technology, este software permite escribir el programa para los microcontroladores, sea en lenguaje assembler como en C, además de crear proyectos, compilar, simular el programa, para finalmente programar el dispositivo, contando con un programador para estos.

Este programa contiene todo lo necesario para la realización de cualquier proyecto, por ejemplo permite editar el archivo fuente en lenguaje ensamblador, además de ensamblarlo y simularlo en pantalla, pudiendo ejecutarlo posteriormente en modo paso a paso y ver como evolucionarían los registros internos, la memoria RAM la EEPROM y la memoria del programa, según se fueran ejecutando las instrucciones. Ademas este entorno es similar a estar utilizando cualquier emulador.

Partes del MPLAB

- ✓ Editor: incorporado para permitir escribir y editar programas u otros archivos de texto.
- ✓ Project Manager: Organiza los distintos archivos relacionados con un programa en un proyecto. Permite crear un proyecto, editar y simular un programa.
  Además crea archivos objeto y permite bajar archivos hacia emuladores o simuladores de hardware.
- ✓ Simulador: simulador de eventos discretos que permite simular programas con ilimitados breakpoint (puntos de parada), examinar ó modificar registros, observar variables, tiempos y simular estímulos externos.
- ✓ Ensamblador: Genera varios tipos de archivos objetos y relacionados, para programadores microchip y universales.
- ✓ Linker: Permite unir varios archivos objetos en uno solo, generados por el ensamblador o compiladores C.
- ✓ Programador: El MPLAB puede trabajar con varios tipos de programadores, el usuario debe seleccionar con cual va a trabajar, haciendo clic en la opción Programmer/ Select Programmer.

Para crear un nuevo proyecto hay que seguir los siguientes pasos, teniendo al principio la siguiente pantalla.

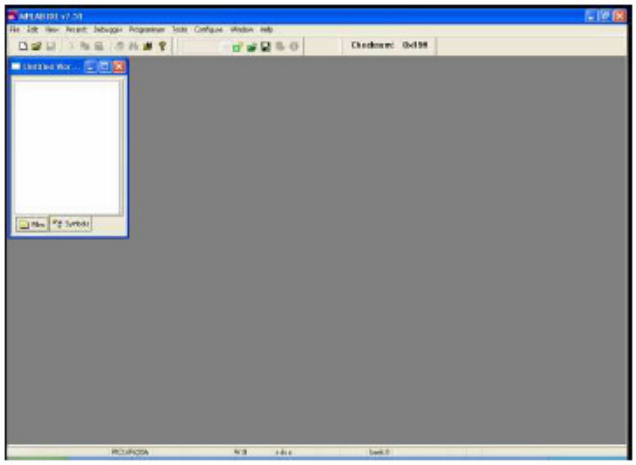


Fig. 5.11.

Vamos en el menú a Project / Project Wizard.

Fig. 5.12.

Hacemos clic en siguiente, y seleccionamos el dispositivo a utilizar, en este caso el PIC 16F877A.

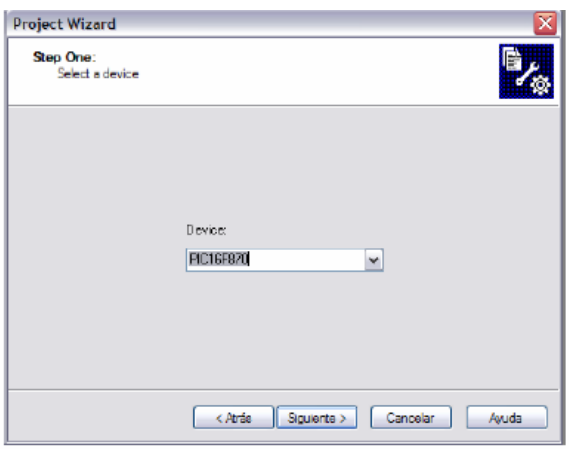


Fig. 5.13.

Luego se define el lenguaje a usar en la programación:

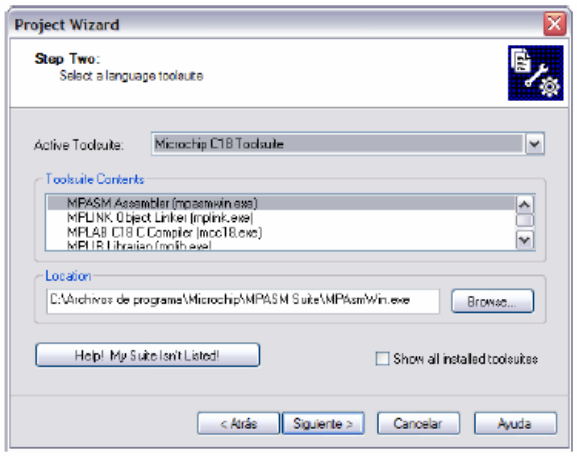


Fig. 5.14.

En la siguiente ventana, se escoge el nombre del proyecto y la dirección en la cual va a quedar guardada. Se recomienda que quede lo mas cerca de C:, ya que si la ruta es muy larga el compilador puede arrojar errores:

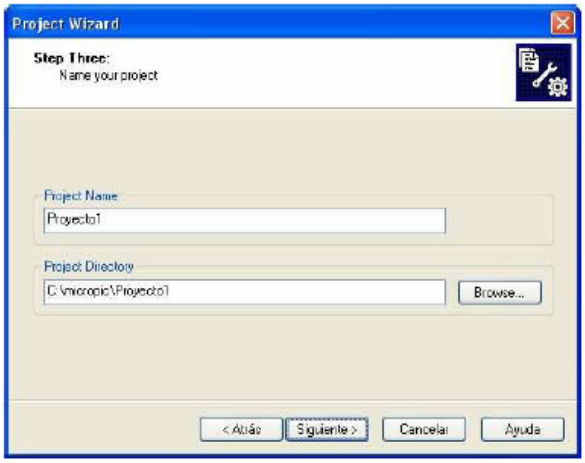


Fig. 5.15.

En la siguiente ventana permite agregar archivos existentes al proyecto, si no existe ninguno simplemente se coloca siguiente:

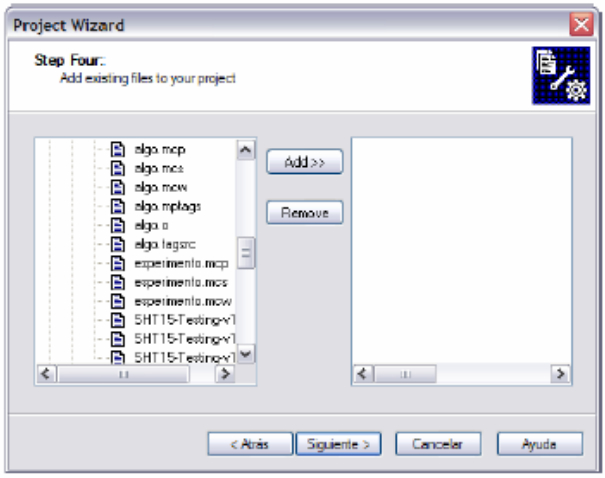


Fig. 5.16.

Y para finalizar aparece un resumen de lo que se hizo.

Fig. 5.17.

Luego si es que no se tenía un archivo previo para escribir el código se crea uno con extensión .asm o .c según el lenguaje de programación que se este utilizando.

En este circuito se ha optado por la programación en ensamblador del microcontrolador, principalmente, por haber tratado con más experiencia con este tipo de programación.
Además, se cuenta con la ventaja de que el código objeto generado por el ensamblador es más eficiente y ocupa menos tamaño que el generado por un compilador C.
En un microcontrolador, la generación de procesos y la utilización de recursos deben ser muy controlada.

Una vez se tiene desarrollado el *firmware*, se procede a su grabación dentro del microcontrolador. Proceso que se consigue mediante un grabador y su correspondiente software, en este caso PicProg2006.

B. PicProg2006

Software desarrollado por la empresa Velleman, que viene incluido en el programador VM134.
Mediante este software podemos abrir un fichero .hex, desarrollado en el proceso anterior, y grabarlo dentro del microcontrolador. Los pasos son los siguientes:

Una vez hayamos ejecutado la aplicación, tendremos la siguiente pantalla:

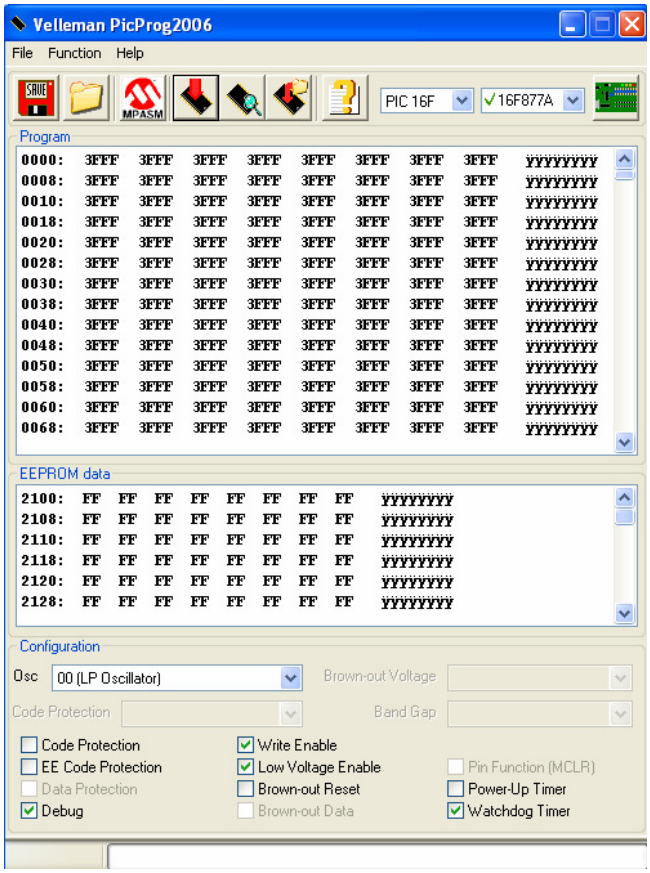


Fig. 5.18.

En el menú, vamos a File / Load File…

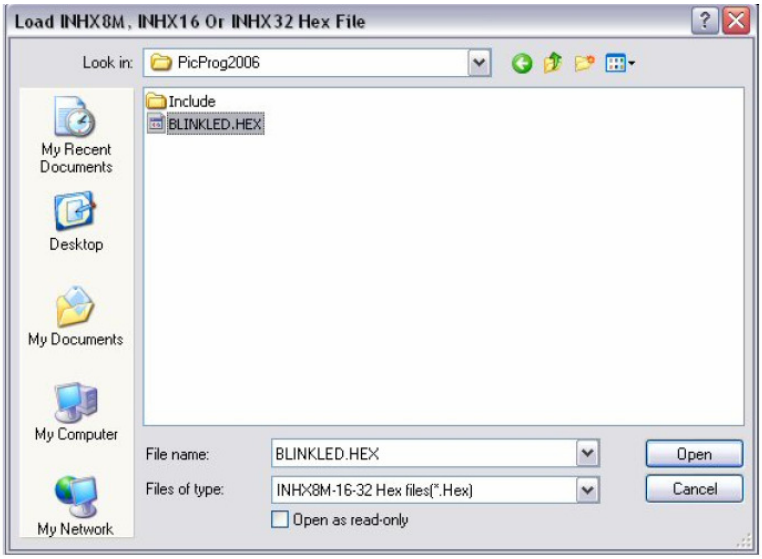


Fig. 5.19.

Una vez se tenga carado el fichero .hex, se procede a la configuración oportuna del PIC. Se deberá configurar tanto el tipo de reloj (oscilador) que se tenga, así como las diferentes opción para el código

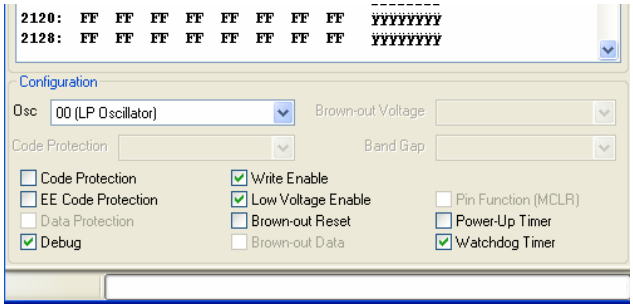
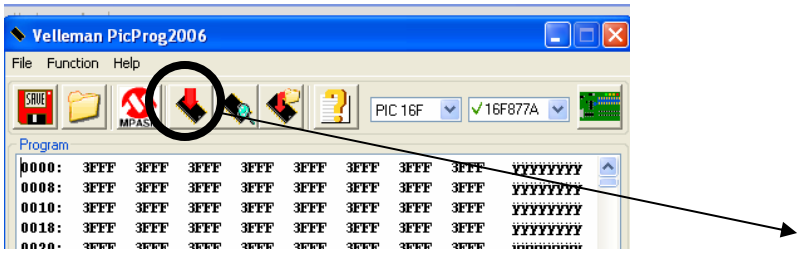


Fig. 5.20.

A continuación, se selecciona el siguiente acceso directo para proceder a la escritura del PIC.



Para escribir en el PIC

Fig. 5.21.

El software procede a la escritura del PIC.

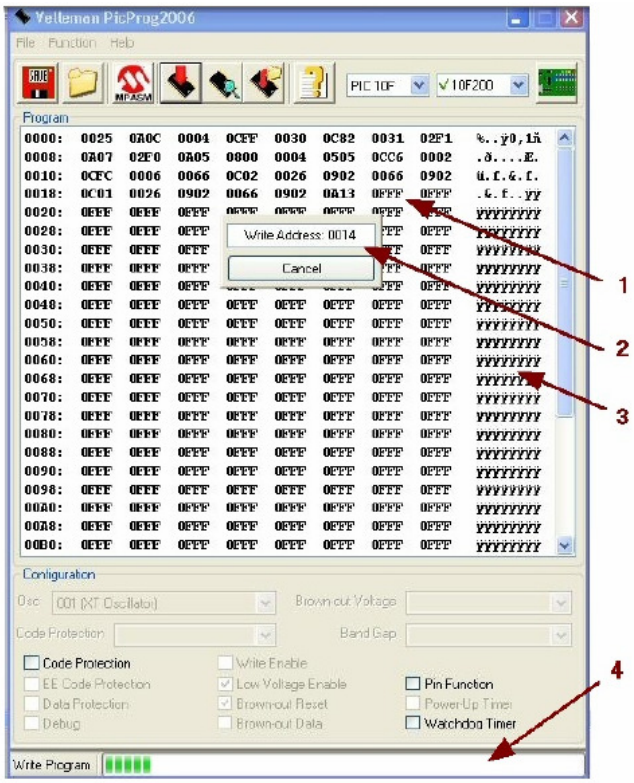


Fig. 5.22.

1. El código HEX que se guardará en el controlador.
2. La variante ASCII del código.
3. Contador de direcciones: le indica donde se efectúa la memoria la lectura o la escritura.
4. Barra de progresión: Sigue visualmente el porcentaje del procedimiento de programación o aprendizaje.

A continuación, se quita el PIC del zócalo y lo se coloca en el circuito, perfectamente programado.

En el Anexo del CD-ROM se adjunta la documentación completa de dicho software.

C. VM123. El programador.

El programador VM134 (Fig. 5.23..) se conecta al puerto serie del ordenador, alimentado a 15 V, y los pines ICSP debídamente configurados.

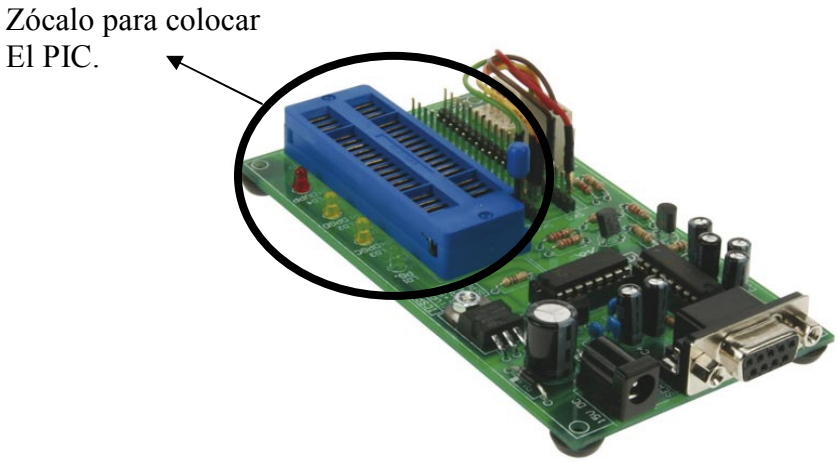


Fig. 5.23.

6. Pruebas de campo.

Las distintas pruebas de campo realizadas a medida que se iba desarrollando el proyecto han sido…

FASE 1

Mostrar por pantalla, mediante una ventana en VB, cualquier información recibida a través de puerto serie (COM).
En este caso, se realizó, para 2 ordenadores, una ventana en VB con la opción de enviar y recibir caracteres, y mostrarlos según se van enviando de un ordenador a otro.

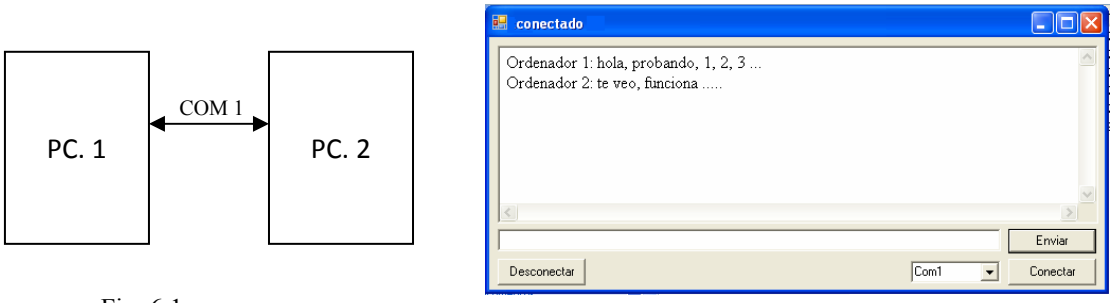


Fig. 6.1.

FASE 2

En esta fase se conecta al microcontrolador un pulsador y un diodo LED, que según se pulse el primero se va apagando ó encendiendo el segundo.

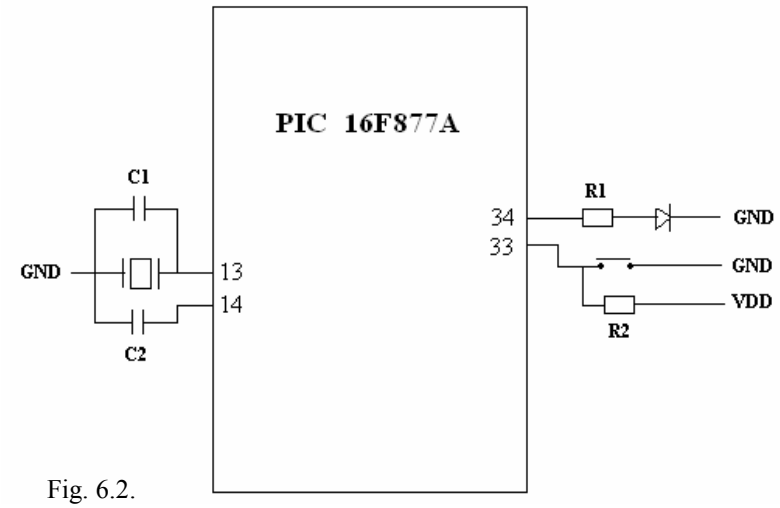


Fig. 6.2.

FASE 3

En esta fase se conecta el PIC a la solución inalámbrica UM-96, y se le mandan 2 valores establecidos en el propio *firmware* a la ventana VB del ordenador (con su propia antena receptora UM-96).

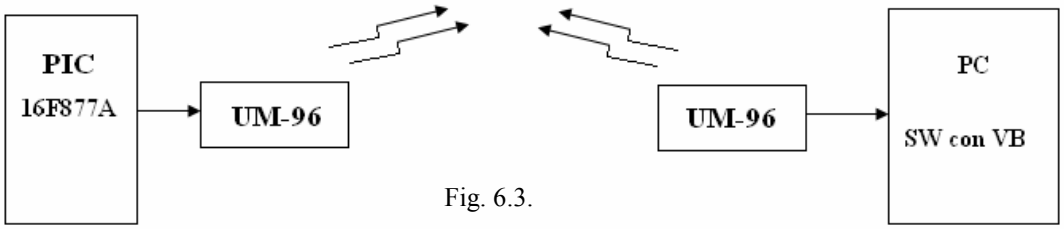


Fig. 6.3.

Fig. 6.4.

FASE 4

Considerada como la Fase Final ó en el proyecto en sí.
Es la Fase 3 pero añadiéndole el sensor al microcontrolador, y en lugar de enviar un '15' y un '55' fijo desde el propio *firmware*, se le manda automáticamente los valores que se capturen del sensor.

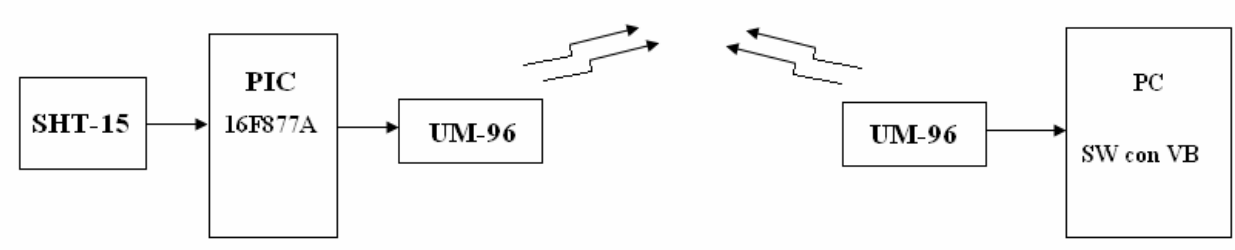


Fig. 6.5

7. Coste económico.

UDS	CONCEPTO	€ / UD	IMPORTE
1	Microcontrolador PIC 16F877A	9,63 €	9,63 €
1	Zócalo CI 40p	0,35 €	0,35 €
1	MAX 232	0,85 €	0,85 €
1	Zócalo CI 16p	0,18 €	0,18 €
1	Cristal 20 MHz	0,37 €	0,37 €
2	Condensadores 15 pF	0,03 €	0,06 €
4	Condensadores 100 nF	0,21 €	0,84 €
1	Diodo 1N4148	0,03 €	0,03 €
2	Placa BOARD	2,55 €	5,10 €
2	7805 CV	0,45 €	0,90 €
4	Pilas CR2032	0,84 €	3,36 €
2	Porta-Pilas 2xBoton	0,61 €	1,22 €
1	Juego de cables	7,84 €	7,84 €
1	Diodo LED Rojo	0,23 €	0,23 €
1	Soluciones inalámbricas UM-96	58,00 €	58,00 €
1	Sensor SENSIRION SHT-15	33,78 €	33,78 €
1	Adaptador USB / RS232	14,24 €	14,24 €
1	DB Hembra SOLDAR	0,81 €	0,81 €

TOTAL (iva inc.)	162,59 €

Adicionalmente, se ha aportado al proyecto un programador de PIC's VM134 por valor de 55,70 € y un cable eléctrico universal THOMSON por valor de 9,59 €.

8. Conclusiones y trabajos futuros.

Para el diseño de un monitor o cualquier otro dispositivo, es esencial realizar un estudio previo, para determinar las necesidades que tendrá el dispositivo, las variables a considerar, rangos, metodología, etc. para precisar los objetivos que debe cumplir el artefacto y diseñarlo de la manera más adecuada para cumplir estos objetivos, considerando las exigencias que tendrá el dispositivo durante su funcionamiento.

La optimización que deberá efectuarse en el diseño e implementación del instrumento, es un proceso que es realizado durante el trabajo con el mismo, debido a que es necesario un ensayo del funcionamiento "in situ" del dispositivo. La optimización de un dispositivo de este tipo, requiere del análisis de variables del medio externo, que solo pueden ser determinadas cuando el monitor ya esté operando en su ubicación definitiva.

El monitoreo de registros de variables tan fundamentales como son humedad y temperatura en invernaderos, dan paso a que se puedan implementar en base a estos, elementos de control automático en el proceso. Este tipo de registros, permite la optimización de los recursos que posee el agricultor para lograr las variables óptimas para el desarrollo de las plantas, al menor costo posible.

Las tarjetas UM96 son de gran utilidad en dispositivos de transmisión inalámbrica, ya que su flexibilidad nos permite un gran ámbito de uso de estas, además de tener en la misma familia, una gran gama de distancias a alcanzar, lo cual puede bajar costos para quien no necesite tanta distancia para alcanzar sus objetivos, además de poseer varios canales que permiten el uso de varios módulos funcionando de forma de paralela.

La utilización del monitor, permite un estudio continuo en las variables del invernadero, lo cual da la posibilidad de un análisis profundo al como se esta llevando a cabo el proceso, realizar curvas de comportamiento diarios, mensuales y anuales. Al tener este tipo de información a disposición siempre va en post del desarrollo y mejora de un proceso como lo es el cultivo de plantas, hortalizas y flores dentro de invernaderos.

Por tanto, una posible ampliación del proyecto sería almacenar todas las variables captadas en una base de datos, para su posterior análisis, y un posible proyecto futuro sería una serie de accionadores que trabajaran dentro del invernadero, y que actuaran siempre en función del valor de las variables previamente recogidas mediante este proyecto, teniendo en cuenta tal y como se ha comentado en la página 34 (punto 5.1) que por motivos de diseño se ha optado por cambiar al sensor SHT75 y que, por motivos de tiempo, ha sido imposible implantar en dicho proyecto.