1. OBJECTIVO DEL PROYECTO

La necesidad de desarrollar un dispositivo capaz de realizar un diagnóstico más rápido que los actuales sistemas para detección del cáncer de mama , ha propiciado que en LEITAT nos hayamos planteado desarrollar un prototipo de un dispositivo innovador capaz de realizar un diagnostico en tan solo 5 minutos, cuando los sistemas actuales tardan unas 24 horas.

El objetivo de LEITAT es desarrollar una plataforma electrónica que interprete la información del biosensor, capaz de detectar por la sangre el tipo de proteína CA-15-3, que es un indicador de una posible afectación de cáncer de mama.

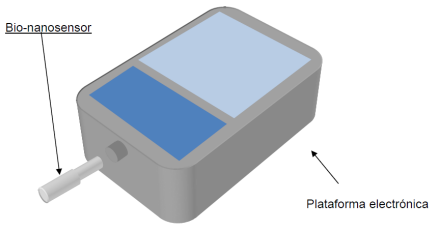

Figura 1. Plataforma de diagnóstico integrada.

El presente proyecto expone como se ha realizado la parte analítica del prototipo, si bien cabe señalar que se hace referencia a la parte de control para poder entender perfectamente el desarrollo del software que acompaña al dispositivo electrónico.

Personalmente me he planteado como objetivo adquirir la experiencia necesaria como programador de microcontroladores, para así aportar mis conocimientos a futuros proyectos de bioingeniería, nanotecnología, medicina, etc. que tenga por delante en la empresa.

2. JUSTIFICACIÓN DEL PROYECTO

La motivación por llegar a completar el software que sea capaz de diagnosticar una afectación por cáncer de mama, me ha llevado a ser en parte autodidacta y así saber programar microcontroladores PIC. Y por otra parte los conocimientos adquiridos en la universidad, especialmente en los siguientes campos: electrónica digital, electrónica analógica, informática industrial y microrobótica, han sido clave para poder ser autodidacta y para completar el desarrollo de la electrónica que acompaña al microcontrolador.

El objetivo de realizar un software cada vez mejor desarrollado es el hecho por el que día a día he trabajado y seguiré trabajando después de este proyecto, ya que en el mundo de la programación de microcontroladores no para de haber innovaciones, lo que es un factor de motivación más para seguir aprendiendo.

Luego he descubierto que el campo de la ingeniería biomédica es el indicado para aplicar los conocimientos que he adquirido durante la realización de este proyecto, ya que siempre se presenta la ocasión de realizar dispositivos que tengan que controlar una serie de parámetros que solo uno o más microcontroladores pueden ser capaces de controlar con precisión, como el detectar una proteína específica dada una muestra.

3. DECRIPCIÓN TECNOLÓGICA

El término “biosensor” hace referencia a un dispositivo compuesto fundamentalmente por dos elementos: un elemento biológico (en este caso una enzima) y un elemento sensor (en este caso electroquímico). En esencia, en un biosensor el material biológico de reconocimiento interactúa con el analítico (sustancia a detectar) y da una respuesta detectable por el elemento sensor, que convierte los cambios en las biomoléculas en una señal de salida (en este caso de tipo eléctrico).

El material biológico utilizado son enzimas, se tratan de proteínas que catalizan las reacciones bioquímicas en los seres vivos, manteniéndose sin cambios al final la reacción. Básicamente destacan por su gran selectividad a un sustrato dado (la molécula transformada) y por agilizar la velocidad de la reacción química (lo que es directamente proporcional a la concentración de la enzima).

Siguiendo con el tema anterior, existen los bio-nanosensores comos dispositivos de coste reducido, capaces de detectar en tiempo real y con buena sensibilidad i selectividad agentes químicos y biológicos.

Un bio-nanosensor esta formado por dos partes: un *receptor biológico* preparado para detectar específicamente una substancia y por un *transductor* que interpreta una reacción de reconocimiento biológico que produce el receptor y lo traduce en una señal cuantificable. Esta reacción es provocada mediante la introducción de nanomaterials.

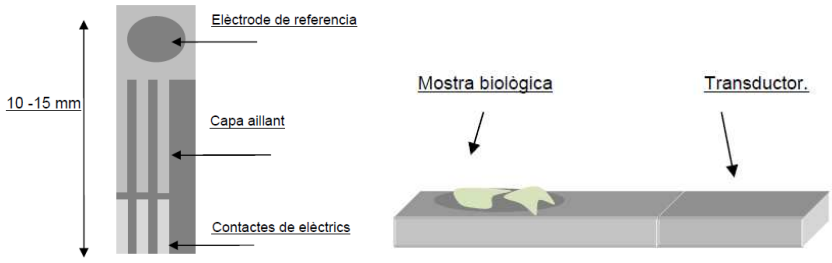

Figura 2. Estructura del transductor.

El funcionamiento se basa en la transformación de las señales derivadas de la reacción catalítica entra la muestra y los nanomateriales en una señal eléctrica reconocible/cuantificable, que será analizado y diagnosticado por la plataforma electrónica que le acompaña.

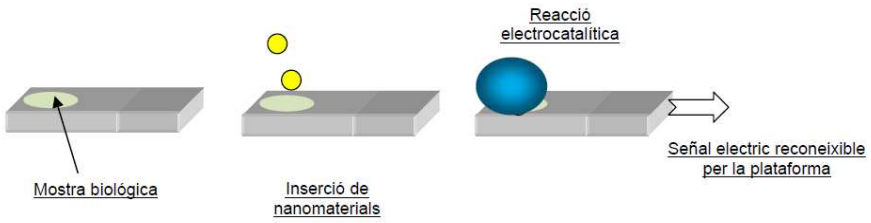

Figura 3. Funcionamiento del transductor.

4. DISPOSITIVO ELECTRÓNICO

El desarrollo del dispositivo se ha desglosado en 5 partes, de las cuales 2 forman parte de este proyecto: “Procesamiento de la señal mediante el PIC16F87XA” y “Acondicionamiento de la señal”. Las otras 3 partes (Alimentación CC/CC, Potensiostato y Interfaz usuario) las desarrollarán mis compañeros de trabajo.

4.1 Acondicionamiento de señal

Las señales que nos proporciona el transductor vienen a ser corrientes del orden de µA que pueden tener valores negativos y positivos. Dado que el PIC utiliza como medida analógica tensiones, se diseña un convertidor I-V que transforme estas corrientes dentro del rango de tensión que va desde los -5V hasta los 5V. Por otra banda, el PIC ha de ser capaz de saber cuando la señal que le llega es positiva y cuando esta señal es negativa, por lo que se adhiere un circuito de cambio de signo al diseño del convertidor I-V para cambiar de signo las señales negativas para que puedan ser tratadas por el DAC, ya que el PIC no puede tratar señales negativas. Se añade también al diseño un seguidor de tensión cuya salida es la entrada para las señales positivas hacia el CAD. Finalmente, se suma al diseño un circuito comparador para poder elegir el canal de entrada de la señal hacia el conversor analógico-digital (DAC) del PIC, según la señal que nos da el biosensor es positiva o negativa.

4.1.1 Conversión I-V

En la figura 4 podemos ver el diseño del convertidor de corriente a tensión. Elaborar este convertidor pasa por calcular el valor de la resistencia R1, como veremos a continuación el valor calculado de R1 permite detectar corrientes hasta un máximo de ± 204,8µA, este

valor se ha escogido a partir de establecer una resolución para el DAC de 0,2 µA / bit, resolución que se ha creído adecuada porque las variaciones de la señal que nos proporciona el biosensor son bastantes pequeñas.

$$I_{\max} = I_{LSB} \cdot 2^N = 0,2\mu A \cdot 2^{10} = 204,8\mu A \quad \text{(Expresión 1)}$$

Sabiendo la corriente máxima que se puede llegar a detectar y sabiendo que la tensión máxima a la que puede llegar una medida es 5V (tanto si es positiva como negativa), se puede calcular la resistencia de entrada del convertidor I-V.

$$R_1 = \frac{V_{cc}}{I_{\max}} = \frac{5V}{204,8\mu A} = 24,41K\Omega \quad \text{(Expresión 2)}$$

Para implementar el convertidor I-V se ha elegido el OP07C como amplificador operacional, básicamente porque es adecuado por el bajo ruido que genera y por su buena precisión para señales muy pequeñas. Se alimenta Vcc+ a 12 V y Vcc- a -12V, ya que si alimentamos a una tensión Vcc+ < 12V y una tensión Vcc- > -12V la tensión de salida se recorta 2V del convertidor I-V. Así aseguramos que la señal de salida se mueva entre -5V y +5V.

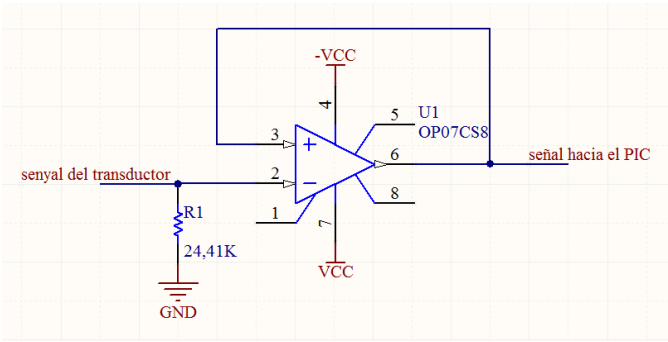

Figura 4. Convertidor I-V.

4.1.2 Adaptación del Nivel de Tensión

En la salida del convertidor I-V tenemos la señal proporcional a la corriente que suministra el transductor. Esta señal de salida esta dentro un rango de tensión que va de -5 V a +5 V, que corresponde al rango de corriente de -204,8 µA a +204,8 µA. Cuando la señal es positiva se activa el canal AN0 del convertidor ADC del PIC y cuando es negativa se activa el canal AN1, donde la señal mediante un cambio de signo se coge positiva sabiendo que se ha de tratar posteriormente como señal negativa. Seguidamente el CAD onvierte la señal analógica a digital. La activación de uno u otro canal esta condicionada por un amplificador diferencial de ganancia 1 y alimentando Vcc+ a 5 V y Vcc- a -5 V, donde en la salida se tendrá 5 V (selecciona canal AN0, señal positiva) o 0V (selecciona canal AN1, señal negativa).

Los diodos a la salida de los operacionales ejercen una función de protección en las entradas analógicas del PIC, para evitar que se produzcan tensiones menores o iguales a -0,3V a la salidas de estos, ja que el PIC no soporta estas tensiones en sus entradas por posibles cortocircuitos (ver el apartado de características eléctricas del datasheet del PIC16F876A).

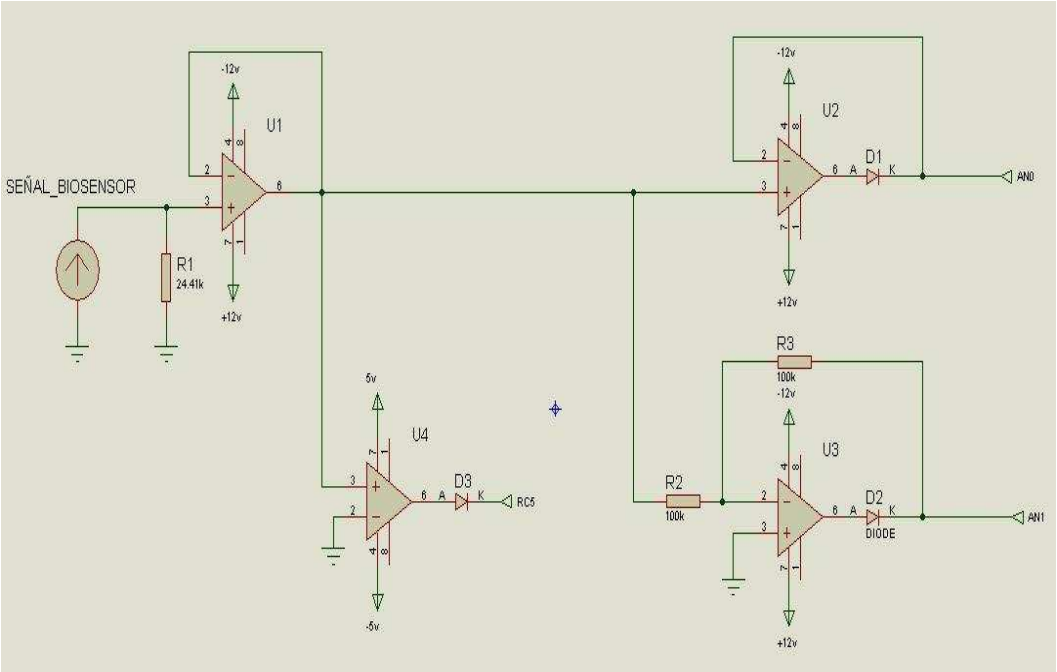

Figura 5. Circuito Acondicionador de la señal.

Todos los operacionales utilizados son del mismo modelo, el OPC07C; así se ha elegido por las características explicadas antes. Todos los operacionales se alimentan a ±12V por el motivo explicado antes, excepto el que forma el comparador que se alimenta a ±5V, ya que aunque la señal de salida quede recortada, el PIC siempre tomará el valor analógico de 3V como un 1 digital y el valor analógico 0V como un 0 digital (ver el apartado de características eléctricas del datasheet del PIC16F876A).

4.2 Microcontrolador y herramientas de desarrollo

La adquisición, el procesado y la respuesta se elaboran dentro de un microcontrolador de la familia PIC16F87XA que posee unas características técnicas suficientes para el desarrollo del prototipo. A continuación se adjunta un cuadro con las principales características del microcontrolador.

Key Features	PIC16F874A	PIC16F876A
Operating Frequency	DC – 20 MHz	DC – 20 MHz
Resets (and Delays)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
Flash Program Memory (14-bit words)	4K	8K
Data Memory (bytes)	192	368
EEPROM Data Memory (bytes)	128	256
Interrupts	15	14
I/O Ports	Ports A, B, C, D, E	Ports A, B, C
Timers	3	3
Capture/Compare/PWM modules	2	2
Serial Communications	MSSP, USART	MSSP, USART
Parallel Communications	PSP	—
10-bit Analog-to-Digital Module	8 input channels	5 input channels
Analog Comparators	2	2
Instruction Set	35 Instructions	35 Instructions
Packages	40-pin PDIP 44-pin PLCC 44-pin TQFP 44-pin QFN	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin QFN

Tabla 1. Características de dos microcontroladores de la familia PIC16F87XA.

Los criterios de selección que se han tenido en cuenta para la selección del microcontrolador son:

- Convertidor A/D de 10 bits: Dado que se requiere una buena resolución, es necesario un CAD con un amplio rango de salida.

- Dos TIMERS: Dado que hay diferentes unidades de tiempos que controlar es necesario la utilización de dos temporizadores. El Timer 1 destinado a controlar tiempos de microsegundos, utilizado para controlar el tiempo de adquisición de datos, y el Timer 2 destinado a controlar tiempos de segundos, utilizado para controlar los tiempos de polarización.

- Entradas/Salidas: El dispositivo necesita 3 salidas destinadas a controlar 6 LEDS, 2 salidas para aplicar al electrodo una tensión positiva y otra negativa, 11 salidas destinadas a controlar el LCD, 1 entrada destinada a colocar el electrodo, 1 entrada destinada al botón de START, 1 entrada para detectar el signo de la señal obtenida por el CAD, 2 entradas para adquisición de datos a través del CAD.

Me he decantado por utilizar el PIC16F876A, por su pequeño tamaño lo que reduce las dimensiones de la placa donde irá soldado junto a otros componentes electrónicos. Por otro lado con este PIC se llega justo a las entradas/salidas necesarias, por lo que no se puede adaptar otras posibles aplicaciones.

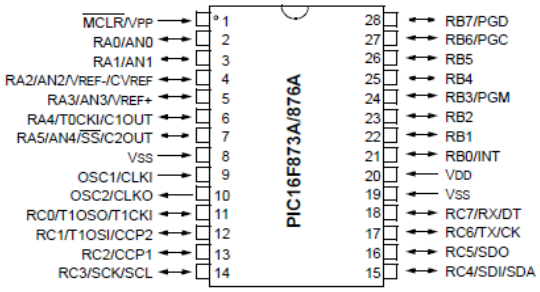

Figura 6. Esquema del microcontrolador PIC16F876A

Para el proceso de programación y simulación se utiliza el programa MPLAB IDE v8.43. En esta herramienta es donde se diseñan las diferentes funciones a realizar por el PIC. Se

puede programar en assembler sin problemas, ya que este software incluye el compilador para código máquina.

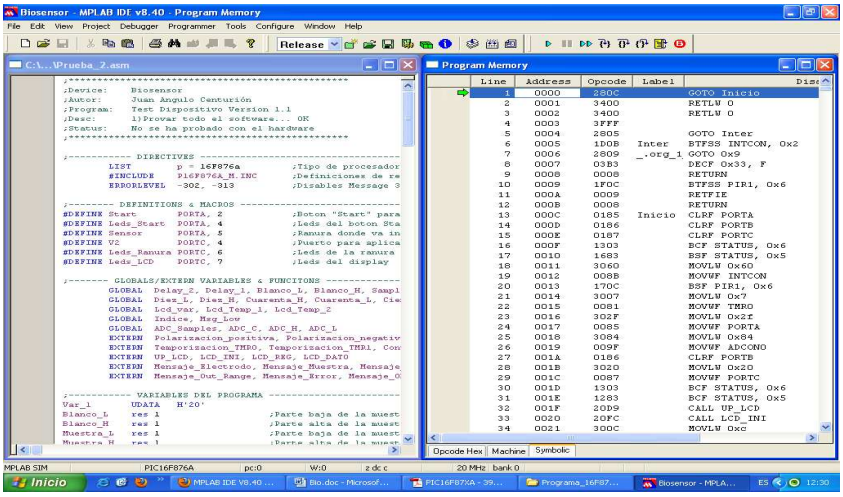

Figura 7. Programa MPLAB IDE v8.43

El programador del PIC se elige de acuerdo a la familia a la que pertenece el PIC. Para el PIC16F876A se ha escogido el programador MPLAB ICD 2, con entrada de datos USB. Este programador recibe del MPLAB el fichero .HEX, como resultado de la compilación del programa principal y sus funciones, y lo programa en el PIC.

Figura 8. Programador MPLAB ICD 2.

Cuando la simulación se vuelve tediosa por el aumento de rutinas o por lo complicadas de algunas de ellas, como por ejemplo las rutinas que utiliza el LCD o las rutinas que utiliza el CAD, es necesario una herramienta de simulación mas rápida y gráfica. Para ello hemos

escogido el simulador PIC SIMULATOR IDE, que nos permita visualizar el PIC y las funciones que necesitemos.

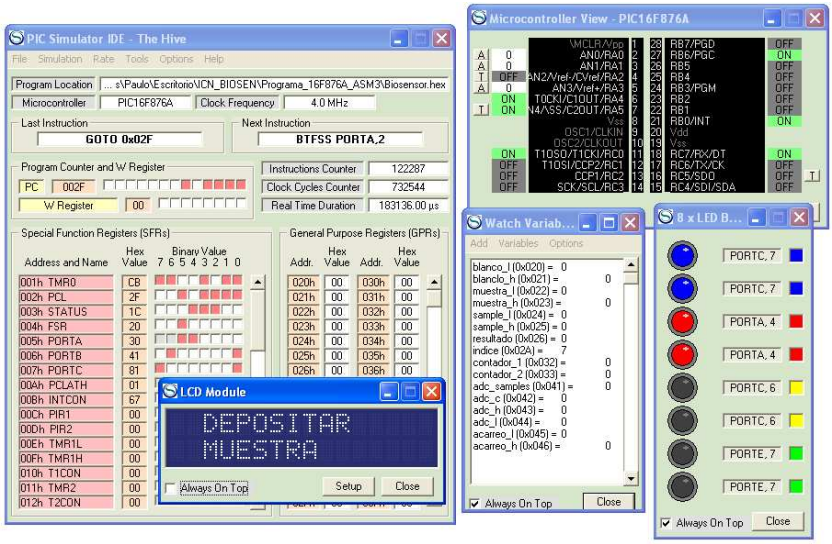

Figura 9. Simulador PIC SIMULATOR IDE con algunas de sus funciones.

Para diseñar y simular el acondicionador de señal se ha utilizado Proteus (en la figura 5 se observa el diseño hecho en Proteus), que es un software que permite simular el comportamiento del acondicionador para diferentes valores de corrientes, y permite también diseñar el circuito acondicionador en PCB.

4.3 Adquisición de datos

La obtención de la información que nos da el biosensor se ha de hacer por dos veces. En la primera adquisición se obtiene información sobre una muestra llamada “muestra blanca”, que es una muestra destinada a compararse con la segunda muestra, llamada “muestra de test”. En la segunda adquisición se obtiene información sobre la muestra del paciente (muestra de test), la cual se comparará con la muestra blanca para determinar su verdadero valor.

La información, como ya se ha descrito antes, consiste en unas señales eléctricas muy pequeñas (corrientes en µA), que han sido transformadas a niveles de tensión; estas señales están dentro de un rengo de valores positivos y/o negativos. Toda esta transformación es realizada por el acondicionador de la señal (Figura 5), que acaba en tres terminales: dos terminales llamados AN0 y AN1 que son las entradas hacia el CAD del PIC para las señales positivas y negativas respectivamente, y un terminal llamado RC5 que es el bit 5 del Puerto C, usado como detector de signo de la señal.

Es decir, los terminales AN0 y AN1 vienen a ser las entradas analógicas seleccionadas hacia el convertidor analógico/digital, el cual cogerá la señal del canal AN0 si esta es positiva, o del canal AN1 si es negativa.

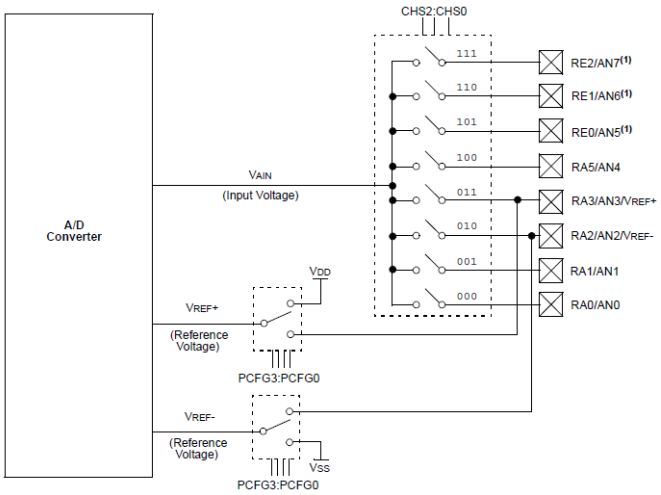

Figura 10. Esquema del Convertidor Analógico/Digital (CAD).

El CAD recibe señales que están dentro de un rango que está condicionado por el fondo de escala, el cual ha resultado ser de 204,8µA (Expresión 1) al decidir un valor por bit de 0,2µA. He elegido este último valor ya que por una parte aborda con gran margen los rangos de los diferentes estados en los que las muestras pueden estar (dentro de este rango están incluidas las corrientes que entran dentro del análisis que son: +10 µA, ±40 µA y -100 µA), y por otra parte porque he considerado suficiente la precisión que se obtiene en cada medida.

Estos valores de corriente se convierten a valores de tensión cuya tensión de referencia lo marca esta vez el PIC, que es de 5V, a partir del cual podemos saber que la resolución del CAD de 10 bits, viene dada por:

$$resCAD = \frac{V_{ref}}{2^N} = \frac{5V}{2^{10}} = \frac{5V}{1024} = 4,88mV \qquad \text{(Expresión 3)}$$

La precisión por bit es bastante buena, como se puede apreciar en el resultado de la Expresión 3. Así podemos tenemos la relación de 0,2µA = 4,88mV.

El sistema sample&hold que posee toda entrada configurada como analógica muestreará las tensiones que se generen en ellas, luego el convertidor las convertirá a digital siguiendo un proceso que se explicará posteriormente. Para saber que valor digital le corresponde a cada valor de tensión se tiene la siguiente expresión:

$$OutputCode = F.S.\times\frac{V_{IN}}{V_{REF}} = 1024\times\frac{V_{IN}}{5V} = 204,8\times V_{IN} \qquad \text{(Expresión 4)}$$

Así sabiendo que 5V equivalen a 204,8µA, aplicando la ecuación 4, obtenemos el código de salida para las corrientes desde 0µA hasta a 204,8µA.

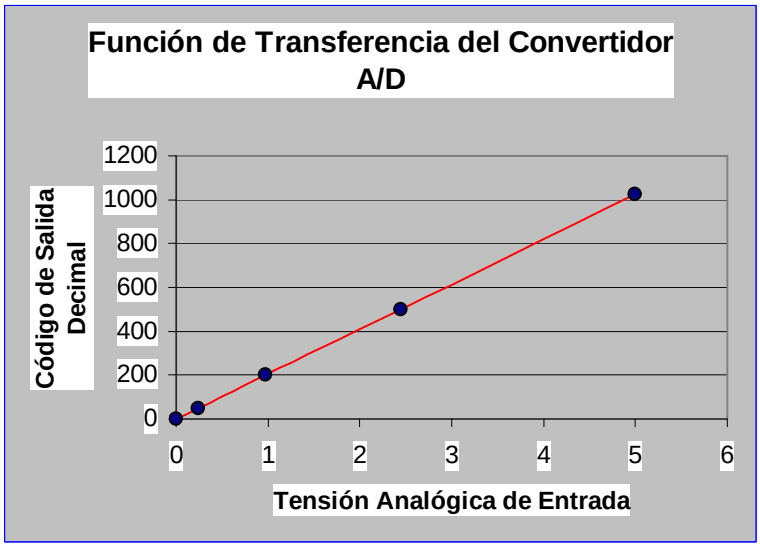

Figura 11. Función de Transferencia del CAD.

A continuación se explica detalladamente los pasos del programa principal, en esta explicación no se incluyen definiciones de memoria o configuraciones generales, porque no he creído importante explicarlo en la memoria, solo se hacen referencias a ellas. En el programa entero, que aparecerá en el Anexo a esta memoria, aparecerán todas sus partes explicadas. Los pasos que sigue el programa para hacer el diagnóstico están en negrita para dejar claro cuales son y no confundir al lector con otros subtítulos.

1. Introducir el electrodo en la ranura de conexión

Se encienden los 2 LEDS que rodean la pantalla del dispositivo para que el paciente esté atento a posibles mensajes, estos LEDS no se apagarán hasta el final del proceso. Aparece en la pantalla un mensaje que indica que se introduzca el electrodo, el electrodo se introduce en la ranura cuando los 2 LEDS que rodean la ranura estén encendidos, cuando el dispositivo reconozca el electrodo se apagaran los LEDS de la ranura, seguidamente se borra el mensaje de la pantalla. El código seria el siguiente:

```
bsf     Leds_LCD            ;Se encienden los LEDS del display.
call    Mensaje_Electrodo   ;Mensaje:  Introducción del electrodo.
bsf     Leds_Ranura         ;Se encienden los LEDS de la ranura.
btfss   Sensor              ;Se espera ha que el electrodo
  goto  $-1                 ;se haya introducido.
bcf     Leds_Ranura         ;Se apagan los LEDS de la ranura.
movlw  0x01                 ;Se carga comando de borrado.
call    LCD_REG             ;Borrar pantalla LCD.
```

2. Depositar muestra en el área del electrodo

Aparece un mensaje en la pantalla que indica que se deposite la muestra (el proceso es el mismo para los dos tipos de muestras). No hay un detector de muestra especifico, más bien es el paciente quien ha de pulsar el botón de START una vez depositada la muestra, para ello se encienden 2 LEDS que rodean el botón después de aparecer el mensaje.

```
Next
    call    Mensaje_Muestra     ;Mensaje: Colocar muestra.
    bsf     Leds_Start          ;Se encienden LEDS del START.
```

3. Pulsar botón de inicio de medida

Una vez colocada la muestra el paciente ha de pulsar el botón de START. Una vez pulsado, se apagan los LEDS del botón, se borra la pantalla, se pone en marcha el proceso de polarización del electrodo, se selecciona el canal dependiendo del signo de la corriente proporcionada por el transductor, obtenemos la señal convertida a bits que nos da el CAD y desactivamos la polarización negativa.

```
    btfss   Start               ;Se espera a que se de al START
     goto   $-1                 ;para empezar el test.
    btfsc   Start               ;Una vez se deja de pulsar el botón
     goto   $-1                 ;empieza el test
    bcf     Leds_Start          ;Se apagan los LEDS del START
    movlw   0x01                ;Se carga comando de borrado.
    call    LCD_REG             ;Borrar pantalla LCD.
    call    Polarizacion_positiva       ;Polarización de +1,35V.
```

```
call    Polarizacion_negativa       ;Polarización de -1V.
call    Acondicionador_de_señal     ;Seleccionamos el canal del CAD.
call    Control                     ;Obtenemos los datos de la muestra.
bcf     V2                          ;Desactivación de -1V.
```

Bien, ahora que hemos visto el programa que se antepone a la adquisición de datos, veremos las funciones a las que se llaman: Polarizacion_positiva, Polarizacion_negativa, Acondicionador_de_señal y Control. Las funciones Mensaje_Electrodo, Mensaje_Muestra y LCD_REG se describirán más adelante ya que son funciones referentes al LCD y de este tipo de funciones se repiten en todo el programa, por lo que su análisis se hará al acabar de analizar todo el programa principal.

La subrutina para polarizar a +1.35V

Esta función consiste en aplicar una tensión entre dos electrodos del bionanosensor durante 60 segundos, ni uno mas ni uno menos. Al acabar este tiempo se dejará de aplicar la tensión inmediatamente.

La temporización se ha realizado con el temporizador Timer 0 (TMRO) del PIC y dos contadores. La configuración del Timer 0 se lleva acabo al modificar los registros: TMRO, INTCON, OPTION_REG. Los contadores son dos variables inicializadas de tal manera que se cumpla el tiempo previsto al llegar a 0 ambas variables.

Lo primero es habilitar la interrupción del Timer 0, así una vez acabe el tiempo establecido en un contador saltará la interrupción y se decrementará el otro contador. En el apartado de “Configuraciones” del programa principal (ver Anexo) tenemos la configuración del

registro INTCON activando el bit TMR0IE de este, para posibilitar la interrupción del Timer 0.

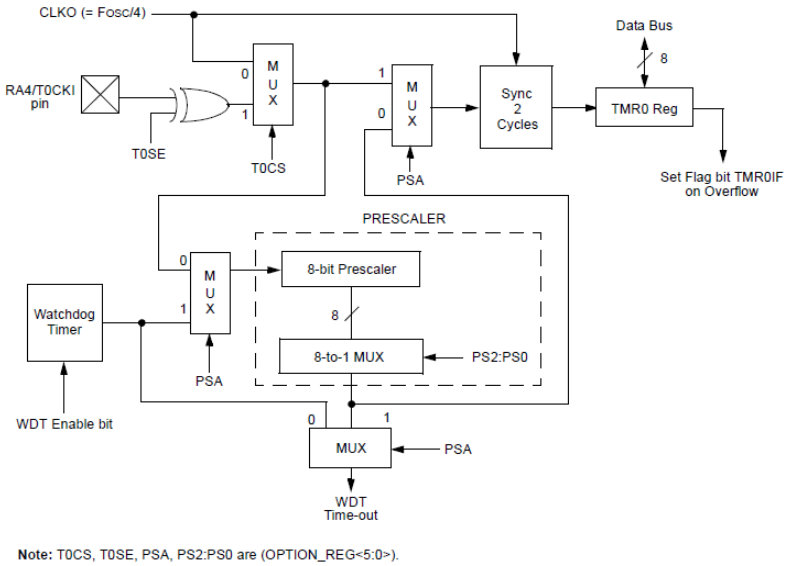

Figura 12. Esquema del TIMER0 & WDT prescaler.

Luego establecemos una serie de opciones para el Timer 0, esto se hace modificando el registro OPTION_REG (mirar Datasheet PIC16F87XA, pag. 23). La activación de los bits PS2, PS1 y PS0 de este registro nos permite tener el prescaler máximo (1:256), lo que nos permite dividir la frecuencia de reloj del PIC por 256, pudiendo temporizar tiempos grandes.

Establecemos el tiempo que temporizara el Timer 0, aplicando la siguiente expresión:

$$Temporiación\_Timer0 = T_{OSC} \times \left(2^n - V_{TMR0}\right) \times \Pr escaler$$

$$Temporiación\_Timer0 = \frac{1}{4MHz} \times \left(2^8 - 2\right) \times 256 \approx 65ms \qquad \text{(Expresión 5)}$$

Donde: “Tosc”, es el periodo del oscilador del PIC que trabaja a una frecuencia de 4MHz, “n” es el tamaño en bits del registro TMR0 (por tanto 2^n es el valor máximo al que puede llegar TMR0), “V_{TMR0}” es el valor inicial de TMR0.

Una ves tenemos que TMR0=2 para obtener un tiempos en este registro de 65ms, hemos de temporizar este tiempo las veces necesarias que nos indiquen los contadores para poder llegar a temporizar 60s.

Utilizamos 2 contadores dado que con un solo contador no llegamos al tiempo requerido ya que hemos de repetir 922 veces el tiempo del Timer 0, y los contadores pueden repetir este tiempo un máximo de 256 veces, dado que son registros de 8 bits.

$$Temporización\_60s = Temporización\_Timer0 \times Contador1 \times Contador2$$

(Expresión 6)

Así tomando un valor para el Contador1=230 y para el Contador2=4, obtenemos los 60s que necesitamos.

$$Temporización\_60s = 65ms \times 230 \times 4 \approx 60s$$

La rutina para polarizar a +1,35V durante 60s es la siguiente:

```
Polarizacion_positiva
        bsf     V1              ;Se aplican los +1,35V
        movlw   .4              ;Cargamos con el valor decimal de 4
```

```
movwf  Contador2            ;el Contador2.
movlw  .230                 ;Cargamos con el valor decimal de 230
movwf  Contador1            ;el Contador1.
call   Temporizacion_TMR0   ;Temporizamos 230 veces el TMR0.
decfsz Contador2, F         ;Hemos temporizado 60s?
 goto  $-4                  ;No, volvemos a temporizar.
bcf    V1                   ;Si, dejamos de dar tensión.
return                      ;Volvemos al programa principal
```

En esta rutina se hace un llamado a otra "Temporización_TMR0". En esta rutina entra en acción el Timer 0 y el registro INTCON. Lo primero ha hacer en esta rutina es activar el bit GIE del registro INTCON que habilita las interrupciones generales, para que una vez ha pasado los 65ms se active el bit TMR0IF.

```
Temporizacion_TMR0
    bsf    INTCON, GIE      ;Activamos interrupciones generales.
    movlw  .2               ;Inicialización del TMR0 para que llegue a
    movwf  TMR0             ;temporizar 65ms.
    btfss  INTCON, TMR0IF   ;¿Se ha acabado los 62,024ms?
     goto  $-1              ;No, aún no.
    bcf    INTCON, TMR0IF   ;Si, borramos flag de interrupción Timer 0.
    movf   Contador1, W     ;¿Se ha repetido el tiempo
    sublw  .0               ;del Timer0 las veces establecidas por el
```

```
btfss   Zero            ;Contador 1?
 goto   $-9             ;No, aún no, volvemos a temporizar.
return                  ;Si, volvemos a la rutina de polarización.
```

Cuando entramos en la interrupción hemos de comprobar si la interrupción ha sido realmente del Timer 0, si es así decrementamos el valor del Contador2 cada vez que se produzca una interrupción por Timer 0. En el código vemos también que puede existir una interrupción por final de conversión del CAD.

```
Inter
        btfss   INTCON, TMR0IF  ;¿Interrupción por final TIMER0?
         goto   $+3             ;No, miramos si es por el ADC.
        decf    Contador1, F    ;Si, se decrementa Contador1.
         return                 ;Vuelta a la rutina Temporización_TMR0.
        btfss   PIR1, ADIF      ;¿Interrupción por ADC?
         retfie                 ;No, falsa interrupción.
        return                  ;Si, vuelta a la función Control.
```

La subrutina para polarizar a -1.0V

Esta función consiste en aplicar una tensión entre dos electrodos del bionanosensor durante 5minutos. Al acabar este tiempo se dejará de aplicar la tensión inmediatamente. Esta función es muy similar a la descrita anteriormente, solo cambian los valores que se cargan en los contadores. Así, cogiendo como referencia la Expresión 6 miramos cuales pueden ser los valores para cada contador, así tenemos Contador1=154 y Contador2=30.

$$Temporización\_5\,\text{min} = 65ms \times 154 \times 30 \approx 300s = 5\,\text{min}$$

El código es muy similar al de la función de “Polarización_positiva”.

```
Polarizacion_negativa
        bsf     V2              ;Activación     de -1V
        movlw   .30             ;Cargamos con el valor decimal de 30
        movwf   Contador2       ;el Contador2.
        movlw   .154            ;Cargamos con el valor decimal de 154
        movwf   Contador1       ;el Contador1
        call    Temporizacion_TMR0      ;Temporizamos 154 veces el TMR0.
        decfsz  Delay_1, F      ¿Hemos temporizado 5 min?
        goto    $-4             ;No, volvemos a temporizar
        return                  ;Si, vuelta al programa principal.
```

Una vez hemos acabado de temporizar volvemos al programa principal sin desactivar al polarización negativa, ya que es necesario que se mantenga dicha polarización mientras cogemos las muestras con el convertidor analógico a digital (CAD). Una vez acabada la conversión se desactivará la polarización negativa.

La subrutina para controlar el CAD

Esta función es la que mas tiempo ha llevado diseñar ya que intervienen muchos factores, como la configuración de los bits de control del CAD, respetar los tiempos de muestreos y de conversión, sistema de filtrado y interrupción por ADC.

Hay, entre otros, 2 registros de control del CAD, estos son ADCON0 y ADCON1. El ADCON0 se utiliza para encender el convertidor, para poner en marcha el convertidor y para seleccionar el canal analógico por donde entrará la señal a ser convertida a digital. El ADCON1 se utiliza para configurar los canales del puerto A para ser analógicos o digitales según convenga al programador y para seleccionar el formato del resultado de la conversión (justificado hacia la derecha o hacia la izquierda).

La combinación de ambos registros se utiliza para seleccionar la frecuencia de conversión, o lo que es lo mismo determinar el tiempo de conversión por bit TAD. El tiempo de conversión total ha de ser como mínimo de 12TAD para 10 bits. Por tanto ya que utilizamos un reloj para el PIC de 4 MHz tenemos un periodo de oscilación de Tosc = 250ns, y yendo a la tabla hemos de elegir la configuración idónea de los bits ADCS2 del registro ADCON1 y los bits ADCS1 y ADCS0 del registro ADCON0 para que se cumpla como mínimo un TAD = 1,6 µs (ver apartado “características eléctricas” del Datasheet PIC16F87XA) necesario para que el CAD convierta correctamente.

AD Clock Source (TAD)		Maximum Device Frequency
Operation	ADCS2:ADCS1:ADCS0	
2 Tosc	000	1.25 MHz
4 Tosc	100	2.5 MHz
8 Tosc	001	5 MHz
16 Tosc	101	10 MHz
32 Tosc	010	20 MHz
64 Tosc	110	20 MHz
RC(1, 2, 3)	x11	(Note 1)

Note 1: The RC source has a typical TAD time of 4 µs but can vary between 2-6 µs.
2: When the device frequencies are greater than 1 MHz, the RC A/D conversion clock source is only recommended for Sleep operation.
3: For extended voltage devices (LF), please refer to **Section 17.0 “Electrical Characteristics”**.

Tabla 2. TAD vs. Frecuencias máximas del PIC.

De la Tabla 3 se ha elegido la configuración correcta, ya que nos da un TAD>1,6µs

$$T_{AD} = 8 \times T_{OSC} = 8 \times 250ns = 2\mu s \qquad \text{(Expresión 7)}$$

Lo primero que se ha hace es determinar las entradas analógicas del convertidor, de las cuales utilizaremos RA0 o RA1 del Puerto A (ver Figura 10) para la entrada de la señal positiva o negativa respectivamente, para ello se configura el registro ADCON 1 desactivando el bit PCFG3, activando el PCFG2 y desactivando los bits PCFG1 y PCFG0, luego configurar el CAD para que el resultado de la conversión esté justificado a la derecha, para ello se activan el bit ADFM del registro ADCON 1, esto facilitará que los valores que resulten de las operaciones que se hagan con las muestras sean sencillos de manejar.

En el apartado de “Configuraciones” del programa principal (ver Anexo), se realizan estas configuraciones entre otras, como la configuración necesaria (para poder realizar varias funciones) que modifica cada bit del puerto A como entrada o salida a través del registro TRISA, poniendo a 1 los bits que queremos como entrada y a 0 los bits que queremos como salida.

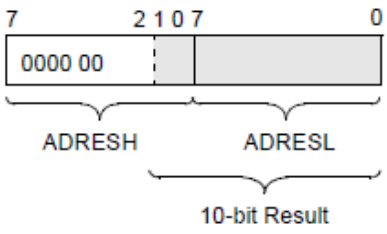

Figura 13. Justificación a la derecha del resultado de la conversión A/D.

Para terminar de configurar el CAD antes de empezar las conversiones, hay que configurar dos registros INTCON y PIE1 en el apartado “Configuraciones”. El primer registro habilita

interrupciones periféricas activando el bit PEIE, y el segundo habilita la interrupción del CAD activando el bit ADIE.

En la declaración de variables que se hace al principio del programa principal en el apartado de “Variables de Programa” (ver Anexo), se declaran tres variables que contendrán los valores de cada conversión que se haga y que están ligadas al proceso de filtrado como se verá en breve.

- ADC_Samples: Esta variable contiene el número de muestras que se harán, en total serán 255 muestras.

- ADC_C; Esta variables contiene el valor del carry después de la conversión.

- ADC_H: Esta variable contiene la parte alta del valor de la conversión.

- ADC_L: Esta variable contiene la parte baja del valor de la conversión.

Como última condición para empezar la conversión, se determina el tiempo de adquisición para tenerlo en cuenta justo antes de empezar la conversión. Para ello se ha de mirar el modelo de la entrada analógica hacia el convertidor que nos proporciona el Datasheet.

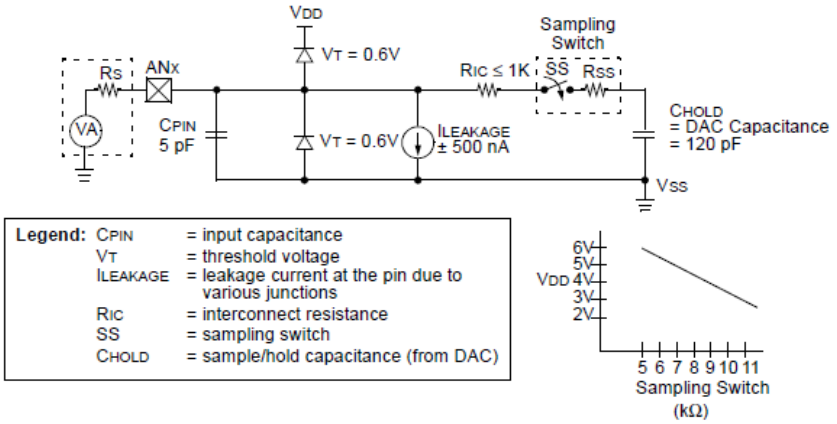

Figura 14. Modelo de entrada analógica.

Para calcular este tiempo se utiliza una expresión que considera un error máximo de 1/2LSB, lo cual ja va bien ya que es admisible. Redondeando tomaremos un tiempo de 20µS de muestreo para cada muestra que hagamos.

$$
\begin{aligned}
T_{ACQ} &= \text{Amplifier Settling Time} + \text{Hold Capacitor Charging Time} + \text{Temperature Coefficient} \\
&= T_{AMP} + T_C + T_{COFF} \\
&= 2\,\mu s + T_C + [(\text{Temperature} - 25°C)(0.05\,\mu s/°C)] \\
T_C &= C_{HOLD}(R_{IC} + R_{SS} + R_S)\ln(1/2047) \\
&= -120\,pF\,(1\,k\Omega + 7\,k\Omega + 10\,k\Omega)\ln(0.0004885) \\
&= 16.47\,\mu s \\
T_{ACQ} &= 2\,\mu s + 16.47\,\mu s + [(50°C - 25°C)(0.05\,\mu s/°C) \\
&= 19.72\,\mu s
\end{aligned}
$$

(Expresión 8)

Bien ahora ya podemos empezar la conversión. Ésta empieza al poner el bit GO=1 del registro ADCON0, una vez acabada cada conversión el resultado se sumará en los registros ADRESH y ADRESL y la adquisición/conversión de la siguiente muestra empezará. Se harán un total de 256 muestras, con el objetivo de realizar un filtrado (Expresión 9) para así poder eliminar el posible ruido proveniente de los operacionales que se infiltre en la medida. A esta función se la ha llamado “Control”.

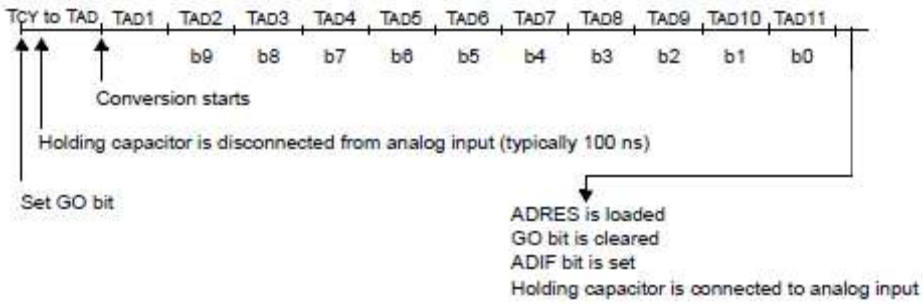

Figura 15. Ciclos de conversión del CAD.

El filtro consiste en capturar un total de 256 muestras y hacer una media entre ellas, para asegurar que el valor que tomemos sea el más próximo a la señal en la entrada del convertidor. En este apartado entran en juego las variables antes descritas: ADC_C, ADC_H, ADC_L, estan variables se irán incrementando conforme el número de muestras convertidas se aproxime a 256, al final se cogerá como válido los valores que contienen las variables ADC_H y ADC_C, la primera como parte baja del resultado y la segunda como parte alta del resultado. Lo que se ha hecho es desplazar un registro (eliminar) el registro ADC_L del resultado final, esto es lo mismo que dividir por 256 el resultado.

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\downarrow$$

$$\bar{x} = \frac{\sum_{i=1}^{256} x_i}{256} \qquad \text{(Expresión 9)}$$

El valor obtenido de hacer dicha media, es el que será analizado posteriormente para determinar un diagnóstico

```
Control
        clrf    ADC_Samples         ;Muestras = 0,
        clrf    ADC_C               ;Carry del resultado = 0.
        clrf    ADC_H               ;Parte alta del resultado = 0.
        clrf    ADC_L               ;Parte baja del resultado = 0.
lb_ADC_1
        bsf     INTCON, GIE         ;Habilita interrupciones generales.
        call    Temporizacion_TMR1          ;Tiempo de muestreo de 20uS.
        bsf     ADCON0, GO          ;Inicia la conversión.
        btfss   PIR1, ADIF          ;¿Interrupción por final de conversión?
         goto   $-1                 ;No, aún sigue en proceso de conversión.
        Bank1                       ;Si, miramos resultado…
        movf    ADRESL, W
        Bank0
        addwf   ADC_L, f            ;ADC_L = ADC_L + ADRESL
        btfsc   Carry               ;Ha habido carry?
        incf    ADC_H, f            ;Si hay carry, ADC_H =ADC_H+1.
        movf    ADRESH, w
        addwf   ADC_H, f            ;ADC_H = ADC_H + ADRESH
        btfsc   Carry               Ha habido carry?
        incf    ADC_C, f            ;Si hay carry, ADC_C=ADC_C+1
        decfsz  ADC_Samples, F      ;¿Control del Numero de Muestras > 256?
```

```
 goto    lb_ADC_1          ;No, se ejecuta siguiente conversión.
movf    ADC_H, W          ;Si, se guarda la parte baja del resultado
movwf  Sample_L           ;filtrado en una variable para su análisis.
movwf  INDF               ;Parte baja de la conversión guardada.
incf      FSR, F          ;Incrementamos puntero.
movf    ADC_C, W          ;Se guarda parte alta del resultado filtrado
movwf  Sample_H           ;en una variable para su análisis.
movwf INDF                ;Parte alta de la conversión guardada.
incf      FSR, F          ;Incrementamos puntero.
bcf       PIR1, ADIF      ;Borrar flag de interrupción por ADC.
return                    ;De vuelta al programa principal.
```

En la función anterior se utiliza el puntero FSR, este puntero debe ser inicializado en la posición 20H de memoria RAM, ya que las posiciones de memoria que van desde la 20H hasta la 24H, están reservadas para guardar los valores de los resultados de la muestra Blanca (Blanca_L=20H, Blanca_H=21H, Muestra_L=22H, Muestra_H=24H) y la muestra de Test. Este puntero también sirve como guía en el programa principal para saber que cuando hemos llegado a la posición 25H hemos acabado los tests de las dos muestras.

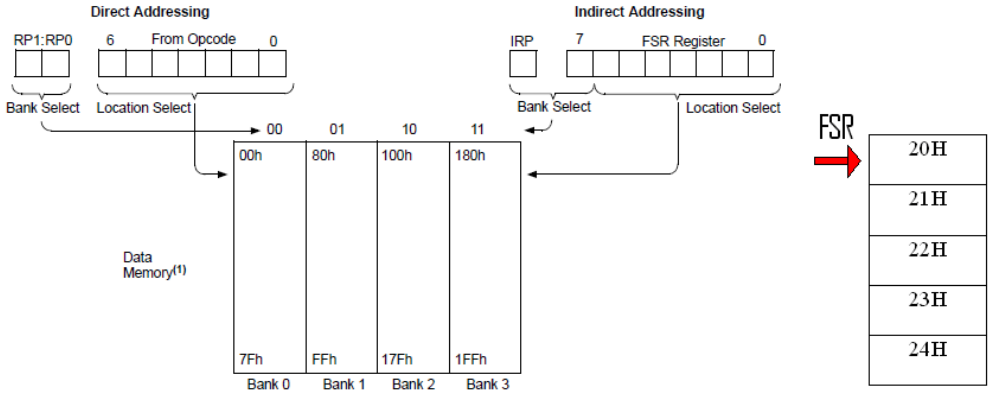

Figura 16. Direccionamiento directo/indirecto. Posiciones de Memoria correspondiente a las 2 muestras.

En la función “Control” también se hace una llamada a la rutina de “Temporización_TMR1”. Esta rutina utiliza el Timer 1 con el fin de temporizar los 20µs necesarios, cada vez que se haga una adquisición.

El Timer 1 esta formado por 2 registros, el TIMER1L y el TIMER1H. En estos dos registros se cargan los valores adecuados para temporizar los 20µs. La forma de saber que valores hemos de poner en ambos registros consiste en saber que el Timer 1 se desbordará cuando lleguemos al valor FFFF, por tanto hemos de cargar los dos registros que forman el temporizador con unos valores que nos permita temporizar 20µs, teniendo en cuenta que por cada cuenta que haga el Timer 1 estaremos temporizando 1µs.

Tomando el valor hexadecimal del número decimal 25 y restando el valor máximo al que llega el temporizador, obtenemos el valor que hemos de cargar en los dos registros del Timer1.

$$25_{(10} \rightarrow 19_{(16}$$

$$FFFF_{(16} - 0019_{(16} = FFE6_{(16}$$

Por tanto, tenemos que el registro TIMER1L=E6 y en el TIMER1H=FF. Así cuando TIMER1L llegue a FF, se desbordará el Timer 1 y se activará el flag TMR1IF del registro PIR1.

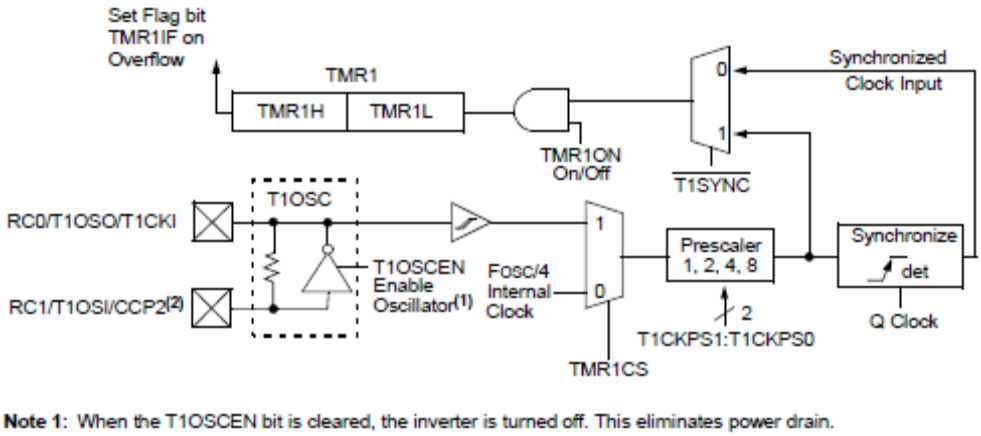

Figura 17. Diagrama de bloques del Timer 1.

La configuración del Timer 1 se reduce en dejar todos los bits a 0 de registro T1CON. Así tendremos, un prescaler de 1:1 y reloj interno (el Timer 1 tiene la misma frecuencia que la del reloj del PIC, así cada cuenta del Timer1 es de 1 µs).

Para empezar a temporizar activamos el bit TMR1ON del registro T1CON. La rutina es la siguiente:

```
Temporizacion_TMR1
        movlw  h'FF'              ;Cargamos un 255 en el registro
        movwf  TMR1H              ;TMR1H.
        movlw  h'EB'              ;Cargamos un 235 en el registro
        movwf  TMR1L              ;TMR1L.
        bsf    TMR1ON, T1CON      ;Ponemos en marcha el Timer 1
        btfss  PIR1, TMR1IF       ;¿El Timer 1 ha sobrepasado el valor FFFF?
         goto  $-1                ;No, aún no.
        bcf    PIR1, TMR1IF       ;Si, borramos el flag de final de tiempo.
        bcf    T1CON, TMR1ON      ;Desactivamos el Timer 1.
        Return                    ;Volvemos a la función de “Control”
```

5. PROCESADO DE LA MUESTRA

Una vez obtenida la medida, se procede al análisis. En primer lugar, una vez obtenida la muestra blanca, se determina el origen de la siguiente medida desplazando el origen de ésta según el nivel de medida que nos da la muestra blanca.

En la siguiente tabla se muestran los estados en los que se puede encontrar la muestra después de ser analizada.

Bit Signo	Resultado (µA)	Estado	Resultado W*
1	Mayor que 10	Fuera de Rango	1
1	Menor que 10	Riesgo Bajo	2
0	Menor que 40	Riesgo Bajo	2
0	Mayor que 40 y Menor que 100	Riesgo Alto	3
0	Mayor que 100	Fuera de Rango	1

Tabla 3. Resultado del análisis de muestras.

*Nota: El resultado en W indica el valor del registro acumulador en el programa para cada diagnóstico.

El bit de signo esta asignado al bit 5 del Puerto C. Este bit es un indicador del signo de la señal que capturamos con el CAD, cuando este bit esta a 0, la señal es negativa, cuando este bit esta a 1, la señal es positiva.

El resultado del diagnostico se muestra a través de la pantalla LCD mediante tres mensajes:

“Fuera de Rango”

Este mensaje aparecerá en el display si una vez obtenida la muestra blanca, ésta tiene un valor más grande que +10µA; o si una vez obtenidas tanto la muestra blanca como la muestra de test (debidamente modificada posteriormente), ésta última tenga un valor más pequeño que -100µA .

“Riesgo Bajo”

Este mensaje aparecerá en el display si una vez obtenida la muestra de test (debidamente modificada posteriormente), ésta tiene un valor que esta entre el siguiente rango [-40µA, +10µA].

“Riesgo Alto”

Este mensaje aparecerá en el display si una vez obtenida la muestra de test (debidamente modificada posteriormente), ésta tiene un valor que esta entre el siguiente rando [-100µA, -40µA].

4. Miramos si las muestras están dentro del rango establecido.

Una vez hemos capturado la señal que nos determina el valor de la muestra blanca, miramos si el valor esta o no fuera de rango. Si el valor esta fuera de rango se muestra el mensaje y se acaba el programa, si por el contrario esta dentro del rango se vuelve al punto 2 del programa para capturar la muestra de test.

Una vez hemos capturado la señal que nos determina el valor de la muestra de test, modificamos su valor restándole el de la muestra blanca, seguidamente se pasa al análisis de la muestra modificada para determinar en cual de los tres estados se encuentra la muestra.

```
        btfsc   FSR, 2              ;Miramos que muestra toca (blanca o test).
        goto    Comp_2              ;Vamos a analizar la muestra test.
Comp_1
        call    Analisis            ;Análisis de la corriente.
        sublw   .2                  ;O una vez obtenido el resultado analítico
        btfsc   Zero                ; verificamos si la muestra blanca está o no
        goto    $+3                 ;dentro del rango permitido (tabla 2),

        call    Mensaje_Out_Range   ;si estamos fuera de rango mostramos error
        goto    Fin                 ;y se acaba el test,
        call    Mensaje_OK          ;si estamos dentro de rango mostramos ok.
        goto    Next                ;Volvemos al punto 2.
```

```
Comp_2
    call    Resta                   ;Modificación de la muestra de test.
    call    Analisis                ;Analisis muestra test
    movwf   resultado               ;En la muestra test guardamos muestra
    sublw   .1                      ;y se mira si estamos dentro del rangos (tabla 2),
    btfss   Zero                    ;si estamos dentro pasamos a dar resultados,
    goto    $+3
    call    Mensaje_Out_Range       ;si estamos fuera de rango mostramos error.
    goto    Fin
    movf    resultado,W             ;Guardamos muestra para compararla
    sublw   .2                      ;y determinar el resultado (tabla 2).
    btfss   Zero                    ;Miramos el resultado de la comparación y
    goto    $+3
    call    Mensaje_Riesgo_Bajo     ;si hay riesgo bajo mostramos mensaje,
    goto    Fin
    call    Mensaje_Riesgo_Alto     ;o si hay riego alto mostramos mensaje,
Fin
    bcf     Leds_LCD                ;Se apagan los leds del display.
    end                             ;Fin
```

Bien, hemos visto todo el diseño del análisis de la señal, ahora veremos las funciones de “Análisis” y “Resta” a las que se hace referencia en el código anterior.

La función “Análisis” se encarga de establecer el rango de los dos tipos de muestra, basa su funcionamiento en lo establecido en la Tabla 2.

Para establecer los valores decimales de las corrientes se hace una simulación del circuito acondicionador (Figura 5) por medio del programa “Proteus”, para saber que valor de tensión nos proporciona el acondicionador para cada valor de corriente del análisis. Una vez obtenido esos valores de tensiones se aplica la Expresión 4. Así tenemos las siguientes relaciones:

$$100\mu A \rightarrow 2{,}44V \rightarrow 500_{(10}$$

$$40\mu A \rightarrow 0{,}976V \rightarrow 200_{(10}$$

$$10\ \mu A \rightarrow 0{,}244V \rightarrow 50_{(10}$$

Un apunte a señalar. En algunos comentarios de código de la siguiente función, aparece la palabra “Comparación”. Ésta hace referencia a una resta entre la muestra obtenida y las muestras a detectar que vienen definidas por la Tabla 2.

A continuación se muestra un ejemplo de la comparación entre una posible señal negativa obtenida, por ejemplo 50µA, y la señal de 100µA.

$$50\mu A \rightarrow 1,22V \rightarrow 250_{(10}$$
$$100\mu A \rightarrow 2,44V \rightarrow 500_{(10}$$

La comparación consiste en una simple resta, las señales se toman como positivas, aunque éstas sean negativas, ya que el signo de ambas ya se tiene en cuenta en el bit de signo, lo que determina si se hace la comparación A1 si el bit de signo es 0 (señal capturada

negativa) o que se haga la comparación A2 si el bit de signo es 1 (señal capturada positiva).

Volviendo al ejemplo, se resta 100µA - 50µA y obtenemos el resulta de 50µA ($250_{(16)}$), por tanto el análisis nos dará que el paciente esta en “Riesgo Alto”.

$$+500_{(10} \rightarrow 0000000111110100$$
$$-250_{(10} \rightarrow 1111111100000110$$
$$+250_{(10} \rightarrow 0000000011111010$$

El código de la función “Análisis” es el siguiente:

```
Analisis
    movlw  h'32'            ;Parte alta del valor binario de 10uA.
    movwf Diez_L
    movlw  h'00'            ;Parte baja del valor binario de 10uA.
    movwf  Diez_H
    movlw  h'C8'            ;Parte alta del valor binario de 40uA.
    movwf  Cuarenta_L
    movlw  h'00'            ;Parte baja del valor binario de 40uA.
    movwf  Cuarenta_H
    movlw  h'F4'            ;Parte alta del valor binario de 100uA.
    movwf  Cien_L
```

```
        movlw  h'01'            ;Parte baja del valor binario de 100uA.
        movwf  Cien_H
        btfsc  Signo            ;Miramos el signo de la señal.
         goto  A2               ;Es positiva vamos a la comparación A2.
A1                              ;Es negativa entramos en comparación A1.
        movf   Cien_H, W        ;Comparamos la parte alta de 100uA
        subwf  Sample_H, W      ;con la parte alta del valor de la muestra
        btfsc  Zero             ;Si coinciden las partes altas…
         goto  $+6              ;… vamos a comparar la parte baja
        movf   Sample_H, W      ;Si no coinciden vemos si la muestra
        sublw  .0               ;es mas grande o mas pequeña que 100uA.
        btfss  Zero             ;¿Es más pequeña?
         goto  Out              ;No, estamos fuera de rango vamos a Out.
        goto   $+5              ;Si, pasamos a comparar parte baja.
        movf   Cien_L, W        ;Comparamos la parte baja de 100uA
        subwf  Sample_L, W      ;con la parte baja del valor de la muestra.
        btfsc  Carry            ;¿Es mas grande la muestra que 100uA?
         goto  Out              ;Si, estamos fuera de rango vamos a Out.
        movf   Cuarenta_H, W    ;No, comparamos la muestra con 40uA.
        subwf  Sample_H, W      ;Comparamos las partes altas.
        btfss  Zero             ;¿Es mas grande la muestra que 40uA?
         goto  Ralto            ;Si, estamos en riesgo alto, vamos a Ralto.
```

```
        movf   Cuarenta_L, W      ;No, comparamos las partes bajas,
        subwf  Sample_L, W        ;restamos…
        btfsc  Carry              ;¿Es mas grande la muestra que 40uA?
         goto  Ralto              ;Si, estamos en riesgo alto, vamos a Ralto.
        goto   Rbajo              ;No, riesgo bajo, vamos a Rbajo.
A2
        movf   Diez_H,W           ;Comparamos la parte alta de 10uA
        subwf  Sample_H,W         ;con la parte alta del valor de la muestra.
        btfss  Zero               ;¿La muestra es mas grande que 10uA?
         goto  Out                ;Si, fuera de rango, vamos a Out.
        movf   Diez_L,W           ;No, comparamos la parte baja de 10uA
        subwf  Sample_L,W         ;con la parte baja del valor de la muestra.
        btfss  Carry              ;¿La muestra es mayor o igual a 10uA?
         goto  Rbajo              ;No, riesgo bajo, vamos a Rbajo.
        goto   Out                ;Si, fuera de rango, vamos a Out.
Out
        retlw  .1                 ;Devuelve valor de fuera de rango.
Rbajo
        retlw  .2                 ;Devuelve valor de riesgo bajo.
Ralto
        retlw  .3                 ;Devuelve valor de riesgo alto.
```

La función “Resta” es la encargada suministrar el “offset” necesario a la muestra de test para que la medida de ésta sea correcta. Este “offset” viene determinado por el valor de la

muestra blanca. Es decir, se coge el valor de la muestra blanca para luego restárselo a la muestra de test, y así tener el valor correcto de la medida del paciente.

En principio se ha decido que la manera de comparar sea una resta, si bien cabe aclarar que los del ICN no tienen aún claro que sea así más adelante.

Los valores esperados en la muestra blanca están dentro del rango de 0 a 10µA, teniendo en cuenta este apunte, realizamos el código de la función “Resta”.

Si por ejemplo obtenemos un valor para la muestra blanca de 45µA (Offset = 45 µA), y luego obtenemos un valor para la muestra de test (sin compensar Offset) de 345µA, obtenemos una muestra de test (con compensación del Offset) de +300µA.

```
Resta
        comf    Blanco_L, F     ;Hacemos complemento A2
        comf    Blanco_H, F     ;de la muestra blanca
        incf    Blanco_L, F     ;para poder restar
        movf    Blanco_L, W     Restamos parte baja,
        addwf   Muestra_L, F    ;resultado en Muestra_L
        btfsc   Carry           ;Hay acarreo de parte baja?
        incf    Acarreo_L, F    ;Si, guardamos acarreo
        movf    Blanco_H, W     ;Restamos parte alta
        addwf   Muestra_H, F    ;resultado en Muestra_H
        btfsc   Carry           ;Hay acarreo de parte alta?
```

```
incf    Acarreo_H, F        ;Sí, guardamos acarreo


btfsc   Acarreo_L, 0        ;Hay acarreo de parte baja?
incf    Muestra_H, F        ;Si, incrementamos Muestra_H
btfsc   Acarreo_H, 0        ;Hay acarreo de parte alta?
 goto   $+4                 ;Si, Resultado - y acabamos comparación.
comf    Muestra_L, F        ;No, Complemento A1 Muestra_L \
comf    Muestra_H, F        ;Complemento A1 Muestra_H      /  Res +
incf    Muestra_L, F
movf    Muestra_L, W        ;Guardamos la muestra para el análisis
movwf   Sample_L            ;en las variables SampleL y SampleH.
movf    Muestra_H, W
movwf   Sample_H
return                      ;Volvemos al programa principal
```

Bien, lo último que queda por explicar es el tema de visualización de datos, ya que a lo largo de todo el programa se hacen llamadas para visualizar diferentes mensajes.

La visualización de datos se hace a través de una pantalla LCD, se utiliza todo el puerto B (PB0 … PB7), configurando todos sus pines como salida, para la salida de datos del PIC hacia el display, y los tres primeros bits del puerto C (PC0, PC1, PC2) se configuran como salidas para poder ejercer un control sobre el controlador de LCD.

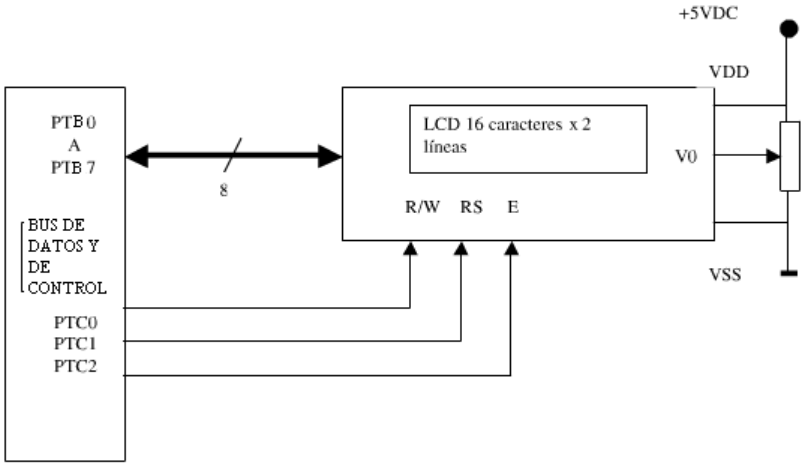

Figura 18. Esquema de la conexión del LCD con el Puerto B.

Lo primero, antes de mostrar cualquiera de los mensajes del programa principal, es configurar el LCD, y dejarlo listo para su uso. En el apartado de “Configuración del LCD” del programa LCD_Driver (ver Anexo) se hace el llamado a las siguientes rutinas que configuran el módulo LCD: “UP_LCD” y ”LCD_INI”. Por otra parte en el apartado de “Definiciones & Configuraciones” del programa LCD_Driver (ver Anexo) se define “OFF_COMANDO” cuya función es borrar el bit 0 del Puerto C para que el LCD tome lo que hay en su entrada de información como comandos y no como datos, también se define “DISABLE” cuya función es borrar el bit 2 del Puerto C para desactivar el display. Y en el apartado de “Variables de Programa” del programa principal (ver Anexo) se declaran las variables “Lcd_Temp1” y “Lcd_Temp2”.

```
UP_LCD
        OFF_COMANDO          ;RS=0 -> Modo Comando
        DISABLE              ;E=0   -> Desactiva señal de Enable
        return
```

Una vez tenemos el LCD en modo comando, enviamos 3 veces el código de instrucción con un intervalo temporal de 5ms. Así el LCD queda borrado y el cursor en la primera posición.

```
LCD_INI
        movlw b'00111000'       ;Código de instrucción
        call    LCD_REG
        call    LCD_DELAY       ;Temporiza
        movlw b'00111000'       ;Código de instrucción
        call    LCD_REG
        call    LCD_DELAY       ;Temporiza
        movlw b'00111000'       ;Código de instrucción
        call    LCD_REG
        call    LCD_DELAY       ;Temporiza
        movlw b'00000001'       ;Borra LCD y Home.
        call    LCD_REG
        return
LCD_DELAY:
        clrwdt
        movlw .10
        movwf   Lcd_Temp_1
        clrf    Lcd_Temp_2
```

```
LCD_DELAY_1:
        decfsz  Lcd_Temp_2,F
         goto   LCD_DELAY_1
        decfsz  Lcd_Temp_1,F
         goto   LCD_DELAY_1
        return
```

El display esta listo para recibir el comando que coloque el cursor en la posición del display que se crea conveniente.

Bien, una vez configurado el display, pasaremos a ver el procedimiento de mostrar los mensajes por pantalla. Para ello solo tomaremos como ejemplo una de las tantas llamadas a mensajes que se hacen en el programa. Así si en el programa principal hacemos la siguiente llamada “call Mensaje_Electrodo”, ésta nos llevará a la rutina “Mensaje_Electrodo que mostrará el siguiente mensaje por el display:

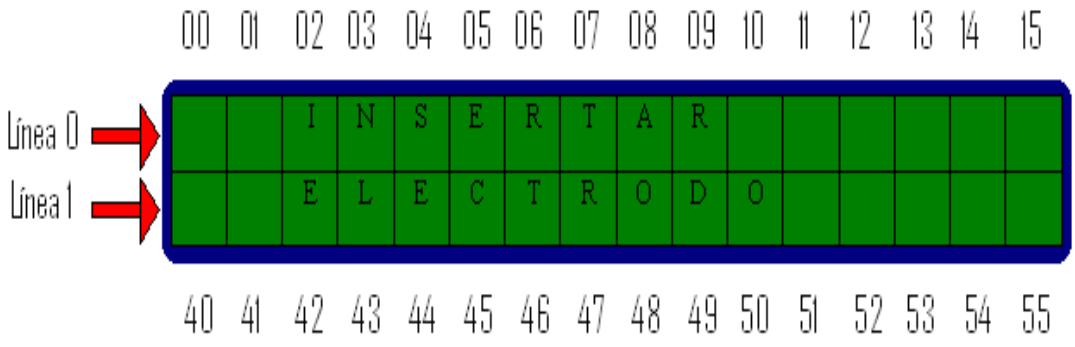

Figura 19. Esquema del LCD con sus posiciones de memoria RAM, mostrando un mensaje.

```
Mensaje_Electrodo
        incf    PCLATH, F       ;Incrementamos parte alta puntero.
        movlw   0x82            ;Comando que coloca cursor LINEA 0.
        call    LCD_REG         ;Envía comando anterior.
        movlw   LOW  Msg_01     ;Coge la dirección definida por Msg_01.
        call    Write_Message   ;Escribe en el LCD el mensaje.
        movlw   0xC2            ;Comando que coloca cursor LINEA 1.
        call    LCD_REG         ;Envía comando anterior.
        movlw   LOW  Msg_02     ;Coge la dirección definida por Msg_02.
        call   Write_Message    ;Escribe en el LCD el mensaje.
        return                  ;Vuelta al programa principal
```

Lo único que tiene de diferente esta rutina de mensaje con las otras rutinas, es que en la primera línea aparece la instrucción “incf PCLATH, F” que incrementa el contador de programa PC. Esta línea de código se utiliza para incrementar el registro PCH, obteniendo PCH=1, ya que no es accesible desde código sino es a través del registro PCLATH. Esto lo hacemos porque los mensajes empiezan en la posición 175H y acaban en la posición 1C2H de la memoria de programa del PIC, y con la instrucción “movlw Msg_01” tenemos el registro PCL=75. En consecuencia el PC apuntará a la dirección PC=175H.

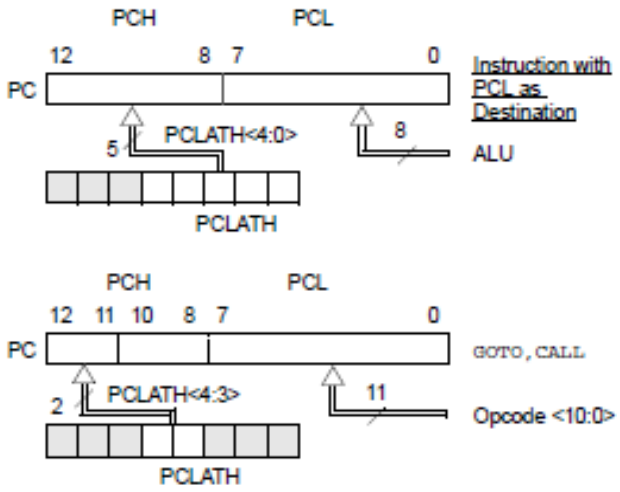

Figura 20. Carga del contador de programa PC, en diferentes situaciones.

Después está el tema de los comandos. En la rutina anterior se envían dos. El comando 0x82 posiciona el cursor en el tercer carácter de la línea 0 colocando el cursor en la posición 02H de la memoria que posee el LCD y activando el bit de busy (bit 7 del puerto B) que indica que el display esta ocupado gestionando ese comando. Por tanto en el puerto B tendríamos 10000010b = 82H (ver Figura 19)

El comando 0xC2 posiciona el cursor en el tercer carácter de la línea 1 colocando el cursor en la posición 42H de la memoria que posee el LCD y activando el bit de busy (bit 7 del Puerto B) que indica que el display esta ocupado gestionando ese comando. Por tanto en el puerto B tendríamos 11000010b = C2H (ver Figura 19).

También aparece la instrucción “movlw LOW Msg_01” que lo que hace es coger la parte baja de la dirección donde empieza el primer carácter del mensaje mostrado en la figurta 19, luego se vera como con la acción de un puntero iremos incrementando de posición para poder visualizar todos los caracteres del mensaje.

En la rutina “Mensaje_Electrodo”, en el momento que queremos enviar un comando por el Puerto B hacia el display llamamos a la rutina “LCD_REG”. Esta rutina envía el comando presente en el acumulador W.

```
LCD_REG
        OFF_COMANDO          ;Desactiva RS (modo comando).
        movwf  PORTB         ;Código de comando.
        call       LCD_BUSY  ;LCD libre?.

        goto       LCD_E     ;Si.Genera pulso de E.
        …
LCD_E
        ENABLE               ;Activa E.
        nop                  ;Pulso de 2µs al ejecutar 2 instrucciones.
        DISABLE              ;Desactiva E.
        movlw .14            ;Cargamos valor decimal de tiempo
        movwf  Lcd_Temp_1    ;para que le LCD asimile la orden.
LCD_E_1
        decfsz  Lcd_Temp_1,F ;Pierde 40 uS para la constante tiempo Tc
         goto    LCD_E_1     ;de los nuevos módulos LCD de winteck.
        Return
```

Una vez tenemos el comando en el Puerto B, antes de activar la señal “enable” y escribir el comando en el LCD, miramos si el LCD esta libre leyendo el bit de busy. A todo esto se ha de saber que el comando permanece en el lacht de salida del Puerto B, antes de llamar a la rutina “LCD_Busy”, rutina que leerá a través del latch de entrada el bit de busy y esperara que sea cero para volver a la rutina LCD_REG, entonces el LCD ya podrá coger los datos del lacth de salida del Puerto B activando la señal de “enable”. A esta señal le sigue un tiempo de 40µs, que es el tiempo que tarda el LCD en asimilar los datos/comandos.

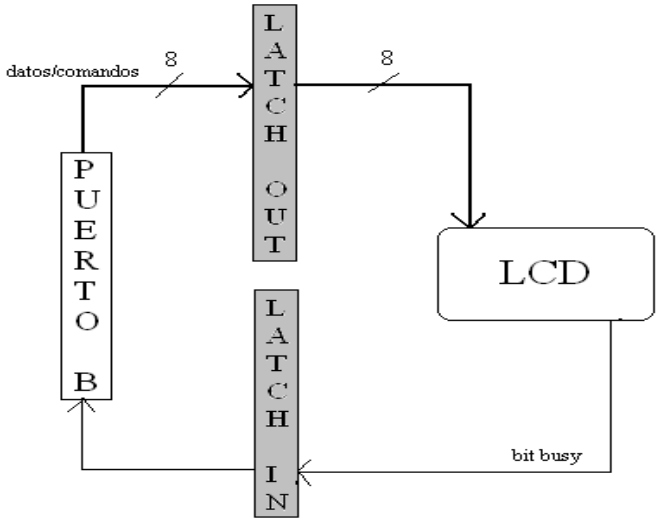

Figura 21. Esquema de Lectura/Escritura del LCD.

A continuación se muestra la rutina “LCD_Busy”, en ella para elegir entre el latch de entrada o latch de salida, según se quiera leer del LCD o escribir en el, solo se ha de configurar el Puerto B como entrada o salida.

```
LCD_BUSY
        LEER                    ;Pone el LCD en Modo RD.
        Bank1                   ;Selecciona Banco 1.
        movlw  H'FF'            ;Configura Puerto B
```

```
        movwf  PORTB            ;como entrada.
        Bank0                   ;Selecciona banco 0 de memoria.
        ENABLE                  ;Activa el LCD.
        nop                     ;1µs.
LCD_BUSY_1
        btfsc   PORTB,7         ;Chequea bit de Busy
         goto   LCD_BUSY_1      ;LCD ocupado
        DISABLE                 ;LCD libre, desactiva LCD.
        Bank1                   ;Selecciona banco1 de  memoria.
        clrf    PORTB           ;Configura Puerto B  como salida.
        Bank0                   ;Selecciona banco 0 de memoria.
        ESCRIBIR                ;Pone LCD en modo WR
        Return
```

En el apartado de “Definiciones y Configuraciones” del programa LCD_Driver (ver Anexo) se define LEER cuya función es activar el bit 1 del Puerto C para poner el LCD en modo lectura, se define también ESCRIBIR cuya función es borrar el bit 1 del Puerto C para poner el LCD en modo escritura.

Cuando hacemos un “enable”, estamos dando la orden al LCD para que el comando u dato que se encuentre en el “Latch Out” sea admitido por el LCD. Así si era un comando lo que había en el Puerto B se ejecutarán acciones como posicionar el cursor, borrado de pantalla, etc., y si era un dato lo que había en el Puerto B se visualizará este por el display.

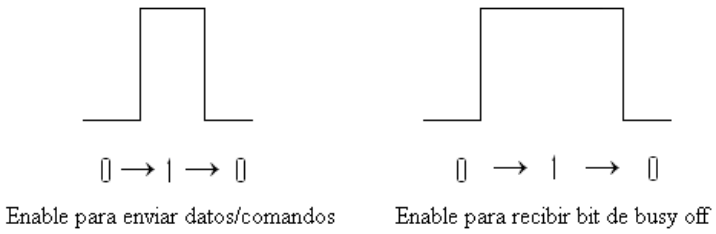

Figura 22. Pulso de enable. Orden para lectura u escritura.

Volviendo a la rutina "Mensaje_Electrodo", para mostrar el mensaje por el display, una vez tenemos la dirección donde empieza el mensaje llamamos a la rutina "Write_Message". Esta rutina se utiliza para mostrar cualquier mensaje que se quiera mostrar.

```
Write_Message
        movwf  Msg_Low          ;Dirección baja del mensaje a mostrar.
        clrf   Indice           ;Índice = 0
lb_WM_01
        call   Messages         ;Vamos a la tabla.
        movwf  Lcd_var          ;Carácter del mensaje
        xorlw  0xFF             ;Control de Fin del Mensaje.
        btfsc  Zero             ;¿Fin del mensaje?
        goto   lb_WM_End        ;Si, salimos de la rutina.
        movf   Lcd_var, W       ;No, mostramos el dato
        call   LCD_DATO         ;por la pantalla LCD.
        incf   Indice,f         ;Incremento puntero para prox. carácter.
        goto   lb_WM_01         ;Vamos a la tabla a coger el carácter.
```

```
lb_WM_End
        return                  ;Vuelve a la rutina de “Mensaje_xxx”
```

La anterior rutina trabaja con tres variables: Msg_Low, Indice y Lcd_Var, estas varibles se definen en el apartado de “Variables de Programa” del programa principal (ver Anexo). En la variable Msg_Low se guarda la dirección de memoria donde está el primer carácter del mensaje que queremos mostrar, la variable Indice se utiliza como una especie de puntero que se ira incrementando para luego apuntar a la dirección donde se encuentra cada carácter del mensaje a mostrar a través del contador de programa PC (ver Figura 20) , y en la variable Lcd_Var es donde se guarda el carácter para luego ser mostrado por pantalla llamando a la rutina “LCD_DATO”. Cuando la variable Lcd_Var contenga el valor 0xFF (este valor se utiliza como comando indicador de que no quedan más caracteres del mensaje para mostrar) volvemos a la rutina del mensaje a mostrar, en este caso “Mensaje_Electrodo”.

Ahora veremos la rutina “Messages”, que contiene todos los mensajes que se han de mostrar por el display. Esta rutina es común para todos los mensajes que se han de mostrar.

```
Messages
        movf    Msg_Low, W      ;Dirección del mensaje a mostrar
        addwf   Indice, W       ;Sumamos el puntero a la dirección.
        movwf   PCL             ;Movemos dirección al PC.
Msg_01
        retlw   'I'             ;Sale de la rutina con el carácter en W.
        retlw   'N'             ;Sale de la rutina con el carácter en W.
        retlw   'S'             ;Sale de la rutina con el carácter en W.
```

```
        retlw   'E'             ;Sale de la rutina con el carácter en W.
        retlw   'R'             ;Sale de la rutina con el carácter en W.
        retlw   'T'             ;Sale de la rutina con el carácter en W.
        retlw   'A'             ;Sale de la rutina con el carácter en W.
        retlw   'R'             ;Sale de la rutina con el carácter en W.
        retlw   0xFF            ;Sale de la rutina con el comando en W.
Msg_02
        retlw   'E'             ;Sale de la rutina con el carácter en W.
        retlw   'L'             ;Sale de la rutina con el carácter en W.
        retlw   'E'             ;Sale de la rutina con el carácter en W.
        retlw   'C'             ;Sale de la rutina con el carácter en W.
        retlw   'T'             ;Sale de la rutina con el carácter en W.
        retlw   'R'             ;Sale de la rutina con el carácter en W.
        retlw   'O'             ;Sale de la rutina con el carácter en W.
        retlw   'D'             ;Sale de la rutina con el carácter en W.
        retlw   'O'             ;Sale de la rutina con el carácter en W.
        retlw   0xFF            ;Sale de la rutina con el comando en W.
Msg:03
        …
```

La tabla de mensajes continua, en el apartado de “Mensajes a mostrar por el LCD” del programa “LCD_Driver” (ver Anexo) se muestran todos los mensajes.

Una vez tenemos un carácter, que a través del acumulador (W) se ha cargado en el la variable Lcd_Var, llamando a la función “LCD_DATO” mostramos el carácter por pantalla.

```
LCD_DATO
        OFF_COMANDO          ;Desactiva RS (modo comando)
        movwf PORTB          ;Valor ASCII a sacar por portb
        call   LCD_BUSY      ;Espera a que se libere el LCD
        ON_COMANDO           ;Activa RS (modo dato).
        goto   LCD_E          ;Genera pulso de E
```

El dato se mantendrá en el latch de salida del Puerto B, mientras esperamos recibir el “bit de busy off “ por el latch de entrada, una vez recibido se da al pulso de “enable” y aparecerá el carácter ASCII en la pantalla del display.

6. RESULTADOS

El software desarrollado cumple con las condiciones planteadas por el cliente (el ICN). Lo único que quedaría por mejorar seria la comparación entre las dos muestras que se analizan en el programa. En este software la comparación consiste en restar a la muestra de test el valor que nos ha dado antes la muestra blanca, eso se hace para tener en cuenta el desfase que nos da la medida en muestra blanca en la muestra d test. En el caso ideal este desfase en la medida debería de ser cero. De todas formas el ICN aún no tiene claro el sistema para comparar las dos muestras, por lo que han decidido que una resta entre ambas por ahora es admisible.

Se puede afirmar que el software esta completo, salvo por el dato explicado antes. Eso por cuanto a la parte de software, la parte hardware lo lleva el resto del equipo de LEITAT y aún no esta acabada por lo que el prototipo final no esta terminado. Solo la parte hardware que he realizado esta montada en placa, aunque aún no se ha podido integrar.

Como resultado experimental he ido probando el programa en el hardware que he ido montando en una protoboard, obteniendo un funcionamiento estupendo de las diversas funciones que ha de hacer el futuro prototipo.

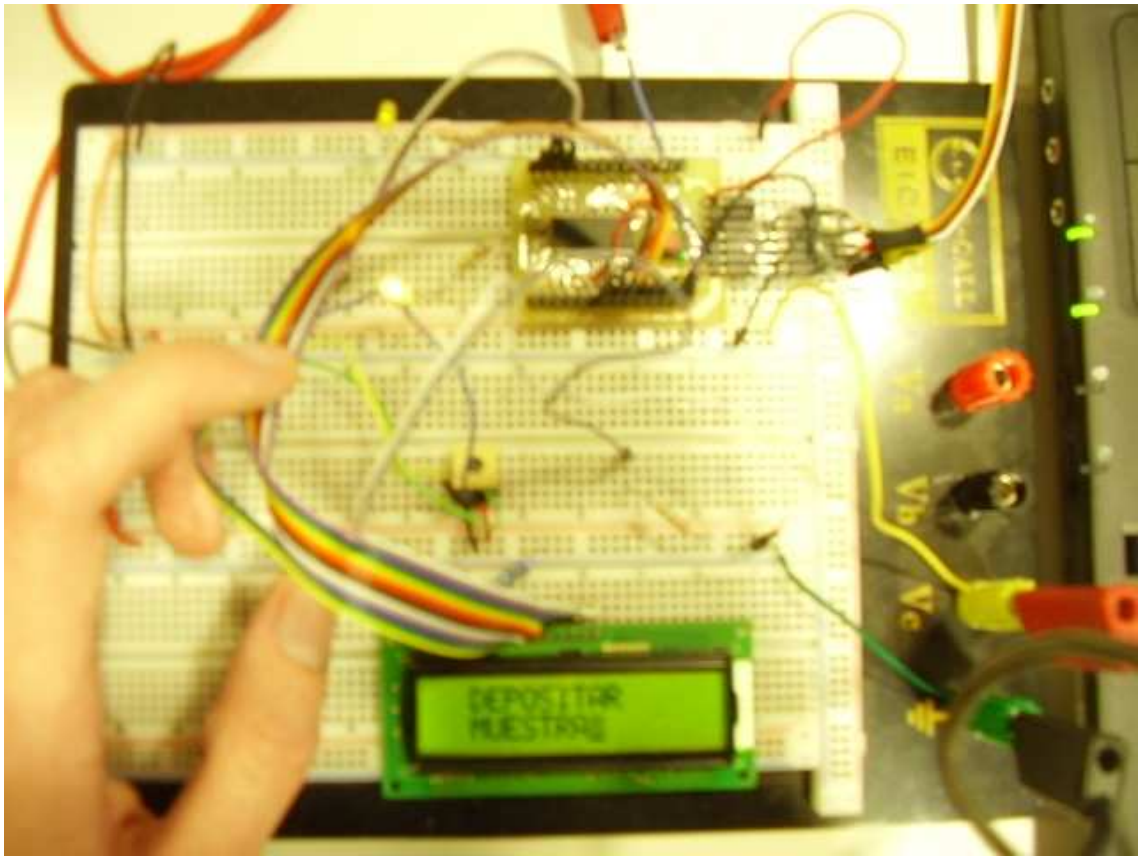

Figura 23. Montaje en protoboard del PIC más el hardware para realizar medidas.

7. CONCLUSIONES

El programa cargado en el PIC y probado con el hardware que se ha podido montar funciona perfectamente, lo que será critico es juntar la etapa del acondicionamiento de señal, ya que el circuito acondicionador puede verse modificado mas adelante con el objetivo de mejorar sus prestaciones ya que aún no se ha podido probar el circuito al estar sin hacer las demás partes del hardware.

El código en lenguaje assembler permite un control exhaustivo (bit a bit) de todos los registros del PIC que se utilizan en el programa. Como el programa ya esta terminado se puede considerar pasar el programa a código C con el objetivo de hacerlo aún más versátil y fácil a la hora de programar, con la idea de añadir librerías que realicen funciones que en assembler son difíciles de montar, aunque eso signifique no tener el control absoluto de algunas de todo el programa.

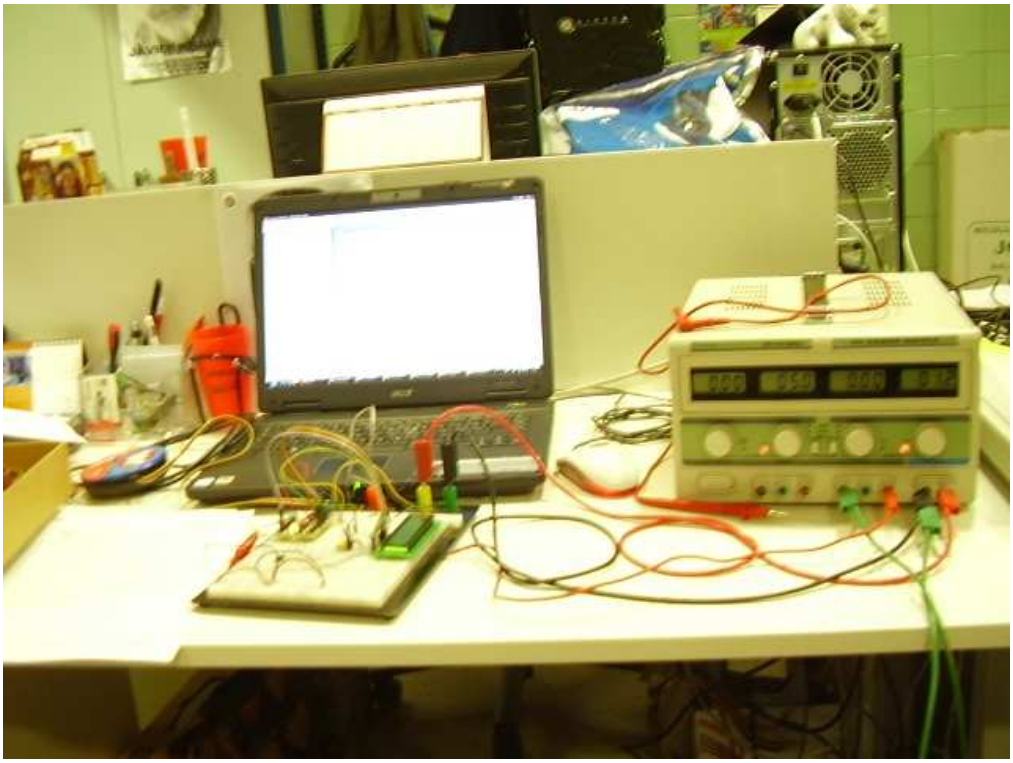

Figura 24. Puesto de trabajo.

8. BIBLIOGRAFÍA

[1] José Mª Angulo Usategui, Ignasio Angulo Martínez, *Microcontroladores PIC. Diseño práctico de aplicaciones*. Primera y segunda parte. Tercera Edición. Mc Graw Hill 2003.

[2] Microchip Tecnology Inc. *PIC16F87XA Family Datasheet*. DS39582B. 2003.

[3] Microchip Tecnology Inc. AN546. *Using the Analog-to-Digital (A/D) Converter*.. DS00546E. 1997.

[4] Microchip Tecnology Inc. AN556. *Implementing a Table Read*. DS00556E. 2000.

[5] Microchip Tecnology Inc. AN580. *Using Timer1 in Asynchronous Clock Mode*. DS00580C. 1997.

[6] Microchio Tecnology Inc. AN693. *Understanding A/D Converter Performance Specifications*. DS00693A. 2000.

[7] Microchip Tecnology Inc. *Operational Amplifier Topologies and DC Specifications*. DS00722A. 1999.

[8] Microchip Tecnology Inc. *MPASM™ Assembler, MPLINK™ Object Linker, MPLIB™ Object Librarian User's Guide*. DS33014J. 2005.

[9] Texas Instruments. *OP07C, OP07D PRECISION OPERATIONAL AMPLIFIERS*. SLOS099D. Febrero 2002.

[10] Sunplus Tecnology Co. *16COM/40SEG Controller/Driver*. SPLC780. 2003.

[11] Miguel A. Pérez, Juan C. Alvarez, *Instrumentación Electrónica*. Segunda Edición. Thomson.

[12] Virginia Espinosa Duró, *Instrumentació 1*. Dossier de la asignatura. Invierno 2006.