Escola Tècnica Superior Enginyeria
Universitat Rovira i Virgili
Departament d'Enginyeria Electrònica Elèctrica i Automàtica

Afinador de bajo eléctrico gestionado por Microcontrolador

TITULACIÓN: Ingenieria Técnica Industrial en Electrónica Industrial

AUTOR: Daniel Díez Díaz-Calonge
DIRECTOR: José Luis Ramírez Falo

FECHA: Enero de 2009

ÍNDICE

1 MEMORIA DESCRIPTIVA **3**
- **1.1 Objetivo** **3**
- **1.2 Antecedentes** **3**
- **1.3 Titular del Proyecto** **3**
- **1.4 Introducción al bajo eléctrico** **3**
  - **1.4.1 El bajo eléctrico** **4**
    - ***1.4.1.1 Rango de frecuencias*** **4**
    - ***1.4.1.2 Pastillas*** **7**
- **1.5 Afinador de bajo eléctrico** **9**
  - **1.5.1 La afinación** **9**
  - **1.5.2 Tratamiento y adecuación** **9**
  - **1.5.3 Muestreo y análisis** **10**
    - ***1.5.3.1 Interrupción externa*** **10**
  - **1.5.4 Comunicación de resultados** **11**
- **1.6 Posibles soluciones y soluciones adoptadas** **11**
  - **1.6.1 Tratamiento de la señal del bajo** **11**
  - **1.6.2 Entorno de trabajo** **13**
  - **1.6.3 Entorno de simulación** **16**

2 MEMORIA DE CÁLCULO **18**
- **2.1 Adecuación de la señal** **18**
  - **2.1.1 Forma y amplitud** **18**
  - **2.1.2 Circuito de adecuación** **18**
    - ***2.1.2.1 El AO 741*** **19**
    - ***2.1.2.2 Divisor de Tensión*** **20**
    - ***2.1.2.3 Rectificador de onda*** **21**
- **2.2 Muestreo de la onda** **22**
  - **2.2.1 Adquisición de la onda** **22**
  - **2.2.2 Timer 1** **23**
    - ***2.2.2.1 Configuración*** **23**
  - **2.2.3 Determinación de la frecuencia** **25**
- **2.3 Comunicación del resultado** **26**
- **2.4 Diagramas de flujo** **28**
  - **2.4.1 Programa principal** **28**
  - **2.4.2 Rutina de servicio a la interrupción ‘high priority’** **33**
  - **2.4.3 Rutina de servicio a la interrupción ‘low priority’** **35**
    - ***2.4.3.1 Timer 3*** **36**
- **2.5 Alimentación del circuito** **38**

3 PLANOS **39**
- **3.1 Placa PCB** **39**

4 PRESUPUESTO **41**
- **4.1 Introducción** **41**
- **4.2 Lista de precios unitarios** **41**
  - **4.2.1 Componentes** **41**
  - **4.2.2 Mano de obra** **43**
- **4.3 Cuadro de descompuestos** **43**
  - **4.3.1 Componentes** **43**
  - **4.3.2 Mano de obra** **44**
- **4.4 Resumen del presupuesto** **46**
- **4.5 Estudio de precio de venta cara al público del afinador** **47**
  - **4.5.1 Costes fijos** **47**
  - **4.5.2 Costes variables** **47**
  - **4.5.3 Costes totales** **50**
  - **4.5.4 Ingresos** **50**

5 PLIEGO DE CONDICIONES **51**
5.1 Introducción **51**
5.2 Reunidos **51**
5.3 Exponen **51**
5.3.1 Primero **51**
5.3.2 Segundo **51**
5.4 Cláusulas **51**
5.4.1 Objeto de acuerdo **51**
5.4.2 Condiciones de aceptación de proyecto **52**
5.4.3 Demora en el desarrollo **52**
5.4.4 Resolución de conflictos **52**

6 ANEXOS **53**
6.1 Código fuente **53**

1. MEMORIA DESCRIPTIVA

1.1 Objetivo

El objetivo de este proyecto es construir un afinador para bajo eléctrico basado en un microcontrolador (PIC 18F2520). El aparato ha de ser capaz de identificar la cuerda que se está tocando y mostrar al usuario el grado de desviación en caso de estar desafinada.

1.2 Antecedentes

Actualmente, en el mercado, existen una gran variedad de instrumentos de afinación: desde afinadores de 15€, hasta instrumentos que son capaces de mostrar el grado de afinación de las cuatro cuerdas a la vez mientras el músico se encuentra en una actuación en directo.

¿Qué se pretende con el siguiente proyecto? Ante todo, comprender cómo se realiza la tarea de afinar una cuerda adecuándola al rango de frecuencias que le corresponde, todo ello con un microcontrolador de pocos euros.

1.3 Titular del proyecto

Nombre: Universidad Rovira i Virgili

CIF: 44444444-P

Dirección: Av. Països Catalans S/N, Tarragona

Representante: Daniel Díez Díaz-Calonge

Teléfono: 977 222 222

1.4 Introducción al bajo eléctrico

Antes de entrar de lleno en las explicaciones pertinentes al afinador, sería conveniente hacer una pequeña introducción del instrumento que vamos a afinar.

1.4.1 El bajo eléctrico:

Un bajo está dividido en las siguientes partes:

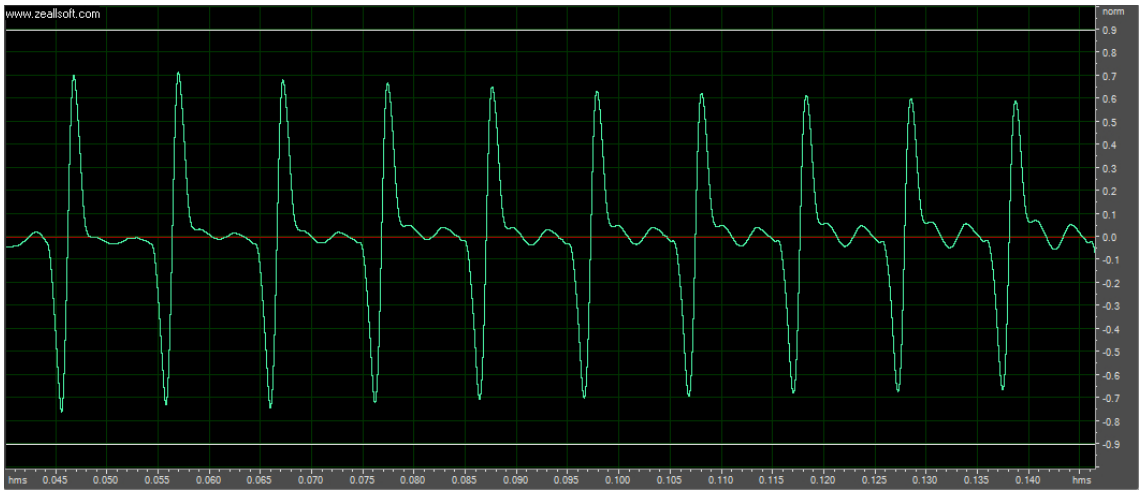
Figura 1

Los bajos pueden oscilar desde las cuatro cuerdas hasta seis o más. Aun y así, lo más común y utilizado son los de cuatro cuerdas, y los de cinco.

Siguiendo con el ejemplo de la figura 1, explicaremos el rango de frecuencias en el que se encontrarían sus cuerdas.

1.4.1.1 Rango de frecuencias

Las cuerdas de los bajos de cuatro cuerdas se encuentran entre los 40 Hz y los 100 Hz. A continuación mostraremos la relación entre cuerda-frecuencia. Cada cuerda tocada al aire (sin pulsar ningún traste) queda representada así:

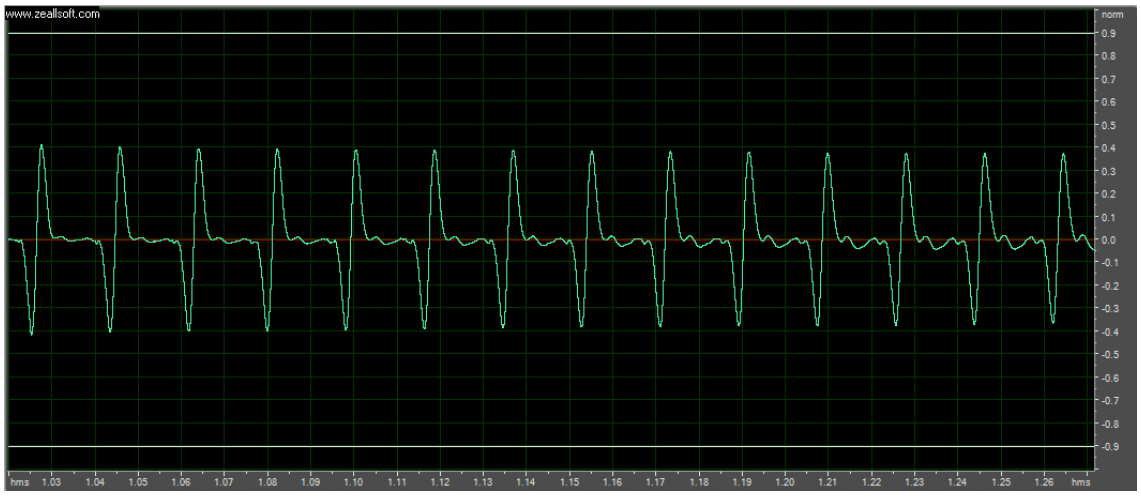


Cuerda 1: Sol

98 Hz

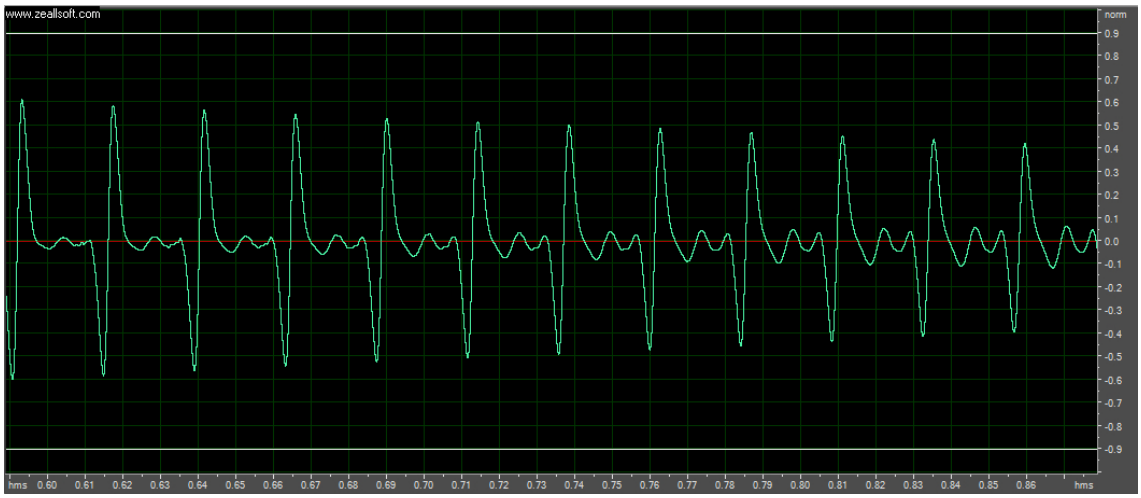
Cuerda 2: Re

73,4 Hz



Cuerda 3: La

55 Hz



Cuerda 4: Mi

41,2 Hz

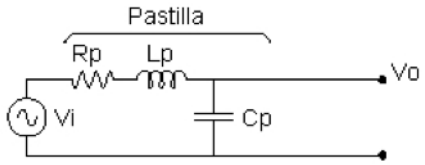
1.4.1.2 Pastillas

Las pastillas están compuestas de un núcleo imantado y de un bobinado a su alrededor. De esta forma, y mediante inducción electromagnética pueden transformar la vibración de la cuerda en impulsos eléctricos con una tensión de pocos centenares de mV (en el apartado *2.1* se hará más hincapié).

Se pueden conseguir tensiones de salida mayores utilizando imanes más potentes, pero como contrapartida el sonido que obtendremos durará menos ya que los imanes atraen más a las cuerdas y se estabilizan más rápidamente.

1.4.1.2.1 Sonido

El bobinado de las pastillas tiene las espiras muy juntas lo que provoca un efecto de condensador. Esto junto a las resistencias del circuito crea un sistema RLC que trabaja como filtro pasa-bajos de segundo orden.



Con una frecuencia de resonancia (*fr*) y un factor de calidad (*Q*):

$$fr = \frac{1}{2\pi\sqrt{LpCp}}$$

$$Q = \frac{1}{2\pi fRpCp}$$

Distintas pastillas tendrán una *fr* distinta también, haciendo que el circuito sea más sensible a frecuencias más bajas o altas, lo que provoca que tengan un sonido singular, resaltando más las frecuencias graves o las agudas.

La frecuencia de resonancia se suele situar entre los 2000 Hz y los 5000 Hz ya que este es el rango en el que el oído humano es más sensible (8ª octava)

1.4.1.2.2 Clasificación

Visto lo anterior, podemos clasificar las distintas configuraciones de pastillas de la siguiente forma:

- Single-coil:

Es la configuración más clásica, que se puede ver en los primeros bajos que vieron la luz allá por los años 50 y 60.

La disposición de las pastillas: una cerca del puente del bajo (de donde nacen las cuerdas) y otra más cerca del mástil. Con eso conseguimos unos medios muy fuertes y unos agudos muy claros

- Humbucker:

La configuración Humbucker no son más que dos pastillas **Single-coil**, pegadas un al lado de la otra, con las bobinas enrolladas en sentido contrario y uno de los dos imanes invertidos. De esta forma conseguimos eliminar los zumbidos que con las Single-coil se producían.

Esta configuración de doble bobinado acarrea una pérdida de agudos ya que el circuito es menos sensible a frecuencias altas, por tanto, al eliminar agudos, los graves cobran mayor importancia en el sonido obtenido y se consigue así una sensación de más profundidad.

- Stacked humbucker:

Son nuevamente dos single-coil, pero a diferencia de las Humbucker, no están una al lado de otra, sino una encima de la otra.

Cómo en el caso de la Humbucker, conseguimos un sonido profundo (un tanto menos pronunciado que en el caso anterior).

- Soapbar:

Más que un tipo de configuración, Soapbar se refiere a una carcasa rectangular bajo la cual se puede colocar cualquier tipo de las pastillas anteriormente explicadas. Así de esta forma podemos hacernos un sonido a nuestra medida, colocando lo que creamos conveniente bajo la “soapbar”.

1.5 Afinador de bajo eléctrico

El objetivo de un afinador es, obviamente, afinar: poner en el tono deseado al instrumento. Para ello se utilizan los sistemas de afinación, los cuales buscan construir una serie de relaciones de frecuencia que dan lugar a las notas de una escala.

1.5.1 La afinación

A modo de pequeña reseña histórica, decir que los patrones de afinación no han sido siempre los mismos y las relaciones nota-frecuencia han ido variando a lo largo del tiempo. Tal era la variedad de afinaciones que se habían ido transmitiendo de generación en generación, que en una misma ciudad podrían haber distintas escuelas de música con su afinación particular usada por cada una de ellas, lo que creaba situaciones caóticas. Es por ello que nació la necesitad de buscar un estándar de afinación, un patrón que hiciese que las notas no variaran de frecuencia según el instrumento o la afinación que hubiese escogido el músico. El estándar actual (ISO 16) se conoce como la_4 a 440 Hz y es a partir del cual se construyen todas las relaciones de frecuencia-nota en las escalas musicales, por tanto el afinador de bajo eléctrico va a afinar siguiendo ese estándar musical.

El afinador de bajo se divide básicamente en tres bloques funcionales:

- Tratamiento y adecuación de la señal de entrada.
- Muestreo y análisis de la onda.
- Comunicación de resultados al usuario.

1.5.2 Tratamiento y adecuación

En esta primera etapa se pretende transformar la señal de entrada que el bajo suministra al circuito. Para ello se ha utilizado un amplificador operacional, el AO741, comúnmente usado en las prácticas de electrónica durante toda la carrera.

Como se ha comentado, el 741 debe convertir la señal de entrada en algo identificable por el microcontrolador. Para ello se ha utilizado el amplificador operacional como comparador, de tal forma que la señal de entrada queda convertida a la salida en pulsos cuadrados periódicos:

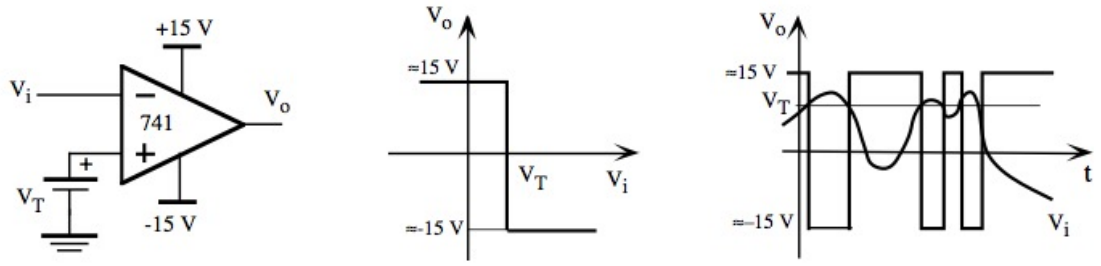


Figura 2

Como podemos ver en la *Figura 2* con esta configuración obtendríamos un tren de pulsos, pero con un semiperiodo negativo. Como al microcontrolador sólo le queremos introducir una señal de amplitud positiva, colocamos un diodo a la salida y eliminamos la parte que no nos interesa.

1.5.3 Muestreo y análisis

Ahora con la señal rectificada, es el turno del microcontrolador, el cual la muestreará para poder determinar en que rango de frecuencias se encuentra. Para ello se ha utilizado un microcontrolador de la casa Microchip, un PIC 18F2520. El muestreo lo llevaremos a cabo utilizando una de las interrupciones externas que posee.

1.5.3.1 Interrupción externa

El PIC 18F2520 tiene, entre otras interrupciones, tres externas: *INT0*, *INT1* e *INT2*, mapeadas cada una de ellas a los pines *RB0*, *RB1* y *RB2* respectivamente. En el caso del afinador utilizaremos la *INT1*.

Las interrupciones externas se pueden configurar para que ejecuten la rutina de servicio a la interrupción bien cuando se detecte un flanco de subida o, por el contrario, un flanco de bajada. En este caso se ha optado por el flanco de subida.

1.5.4 Comunicación de resultados

Una vez el microcontrolador ha realizado todos los cálculos, se ha de poder indicar al usuario el grado de afinación de la cuerda en cuestión. Para ello utilizaremos una serie de diodos LED y una pantalla LCD.

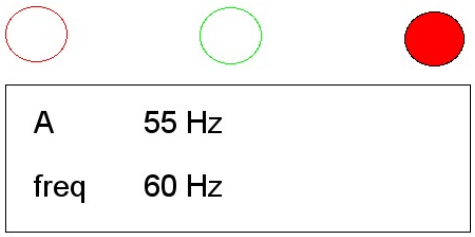


Figura 3

En la *Figura 3* podemos ver una pequeña representación de los diodos y la LCD. La pantalla LCD muestra primero la cuerda que estamos afinando, que debería oscilar a 55 Hz, y en segunda instancia la frecuencia real a la que oscila la nuestra. Como vemos, nuestra cuerda está 5 Hz por encima del valor correcto, por tanto se enciende el LED rojo de la derecha, que indica desafinación por sobrepasar el valor.

Si por el contrario estuviéramos por debajo del valor, se encendería el de la izquierda, mientras que si estuviera bien afinada se encendería el LED verde del medio.

1.6 Posibles soluciones y soluciones adoptadas

A lo largo de este capitulo se explicarán las diferentes opciones que han surgido a lo largo del desarrollo del proyecto y las soluciones adoptadas en cada caso.

1.6.1 Tratamiento de la señal del bajo

En esta apartado de la memoria se hablará de las alternativas que surgieron a la hora del tratamiento de la señal y porque se decidió por la solución que se escogió.

FFT y dsPic:

En un primer momento se pensó en tratar la señal utilizando la FFT, de tal forma que, según fuera llegando al micro la información, éste realizara el tratado y mostrara por pantalla el resultado. Para ello se pensó en utilizar un dsPIC, que es una gama de microcontroladores PIC diseñada por Microchip especialmente para el tratamiento de señales.

Los dsPIC, entre otras características, traen implementados de "fábrica" algunos algoritmos FFT, lo que en nuestro caso sería de gran utilidad.

Transformación de la onda en un tren de pulsos:

Otra solución que surgió fue la de intentar transformar la onda del bajo eléctrico en una onda cuadrada y utilizar el microcontrolador para calcular la frecuencia de la misma.

Esta alternativa, al contrario que la primera, permite trabajar con microcontroladores de gamas inferiores, ya que no necesitamos los algoritmos que un dsPic nos proporciona para el tratamiento de señales.

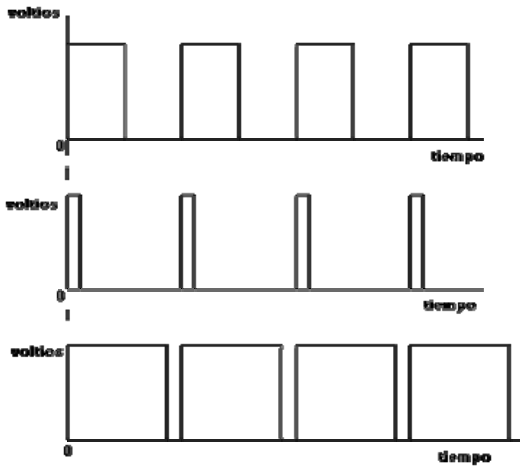


Figura 4

Tres pulsos con la misma frecuencia pero con un ancho de pulso distinto cada uno.

Solución adoptada:

Tras contemplar las dos alternativas, se optó por la segunda y se escogió para ello el PIC 18F2520. Esta alternativa supone las siguientes ventajas:

- Información: A diferencia de los dsPic que llevan en el mercado desde finales de 2004, los PIC de gamas inferiores llevan más años. Eso ha dado lugar a multitud de proyectos realizados con PIC de 8 bits y ejemplos de uso más variados.

- Facilidad de uso: Es menos complejo en cuanto a cálculos se refiere trabajar con señales cuadradas y cálculos de amplitud que con algoritmos de FFT.

- Experiencia previa: En la asignatura de Microcontroladores cursada durante la carrera se nos introduce a los PIC de 8 bits, por lo que se empieza a trabajar sobre una gama conocida.

1.6.2 Entorno de trabajo

Una vez decidido el método a usar para crear el afinador, falta saber bajo que entorno programaremos el microcontrolador.

Se tuvo que escoger entre dos alternativas, programar utilizando el entorno CCS C Compiler para PIC o el entorno MPLAB en unión con el compilador C18 que ofrece el miso *Microchip.*

CCS C Compiler:

El compilador de CCS es la más famosa alternativa al MPLAB de *Microchip*. Está diseñado para trabajar con los PIC mediante una serie de funciones que trae implementadas por defecto que permiten configurar los timers, interrupciones, etc, sin tener que recurrir a tocar directamente los registros involucrados en cada caso.

También incluye un asistente de configuración con el cual podemos crear un proyecto nuevo muy intuitivamente (el asistente te pregunta si quieres o no interrupciones, los timers que se quieren usar, si necesitamos conversión A/D, la frecuencia a la que trabajará el micro…).

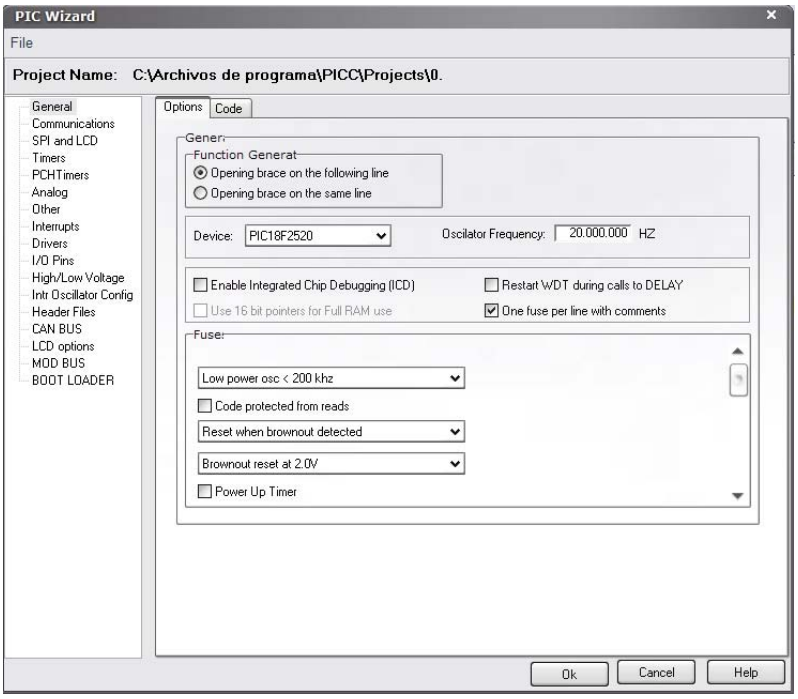


Figura 5

Asistente de configuración

Otro punto a favor es la cantidad de programas ejemplo con los que cuenta el compilador, hay prácticamente un programa ejemplo para cada periférico y sus usos más comunes.

MPLAB IDE:

Si CCS está orientado a hacer el trabajo de programación un tanto más fácil, con una serie de funciones ya implementadas, MPLAB está diseñado para tener un completo control sobre el microcontrolador.

Con CCS, si se tiene que tocar algún registro o un bit en concreto, las direcciones de los mismos al no estar mapeadas, trabajar con registros se convierte en algo tedioso.

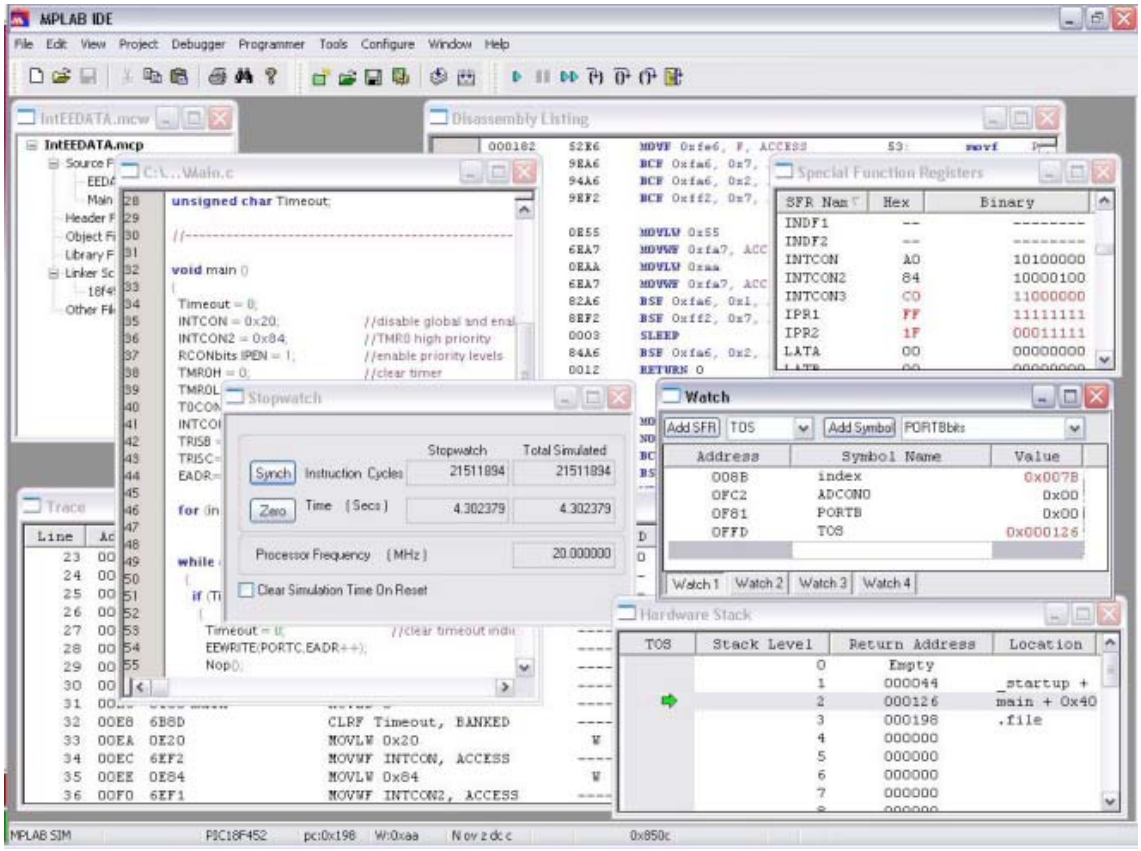


Figura 6

MPLAB IDE

El problema con MPLAB no existe y modificar y trabajar con los registros directamente no es nada complicado, basta con tener a mano el datasheet del microcontrolador para saber el nombre del registro en concreto.

MPLAB cuenta también con un asistente de proyectos, quizás un tanto menos poderoso en cuanto a opciones de configuración que el que proporciona CCS, pero que cumple con su propósito: ayudar al usuario a comenzar.

Solución adoptada:

Por lo expuesto y, aunque CCS gana en sencillez, se optó por el compilador de *Microchip* por las siguientes razones:

- Producto oficial: MPLAB es el entorno que recomienda *Microchip* y cuenta con un foro en Internet muy activo.

- Potencia: Como se ha comentado antes, con MPLAB se pueden controlar todos los registros sin excesiva complejidad.

- Fácilmente accesible: *Microchip* proporciona una versión de MPLAB para estudiantes con la que se puede desarrollar proyectos sin necesidad de adquirir la versión profesional. Sirva de ejemplo el presente afinador.

1.6.3 Entorno de simulación

Debido a las características del proyecto, se hizo necesaria una herramienta que permitiese llevar al día la evolución de éste para solucionar fallos y avanzar. Por ello se utilizó un entorno de simulación: Proteus 7.

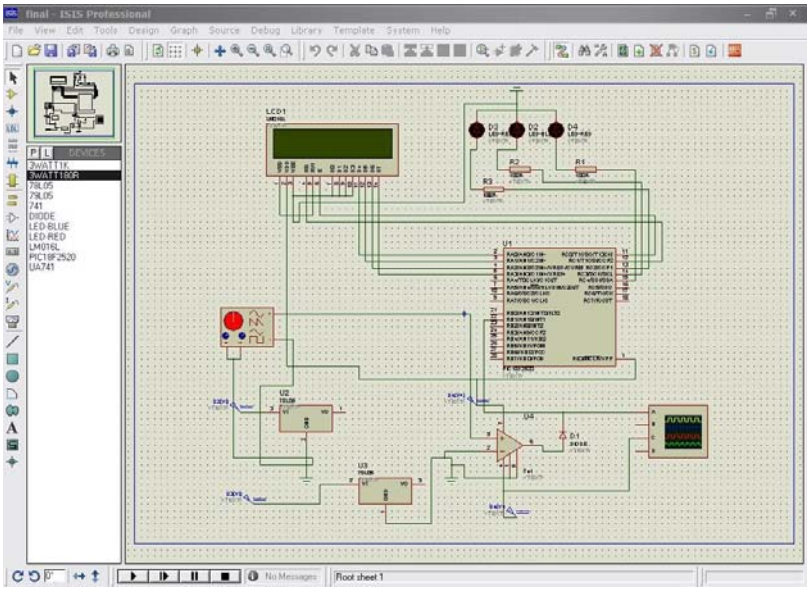


Figura 7

Simulación del afinador

Proteus 7:

Proteus 7 es un simulador de la casa *Labcenter* . Con esta herramienta se pueden construir y simular de una manera muy intuitiva cualquier circuito electrónico, ya que cuenta con una gran cantidad de modelos de dispositivos electrónicos. En el caso concreto del proyecto, ha sido especialmente útil ya que cuenta con la opción de simular el PIC 18F2520, pudiéndole introducir el código generado con el MPLAB.

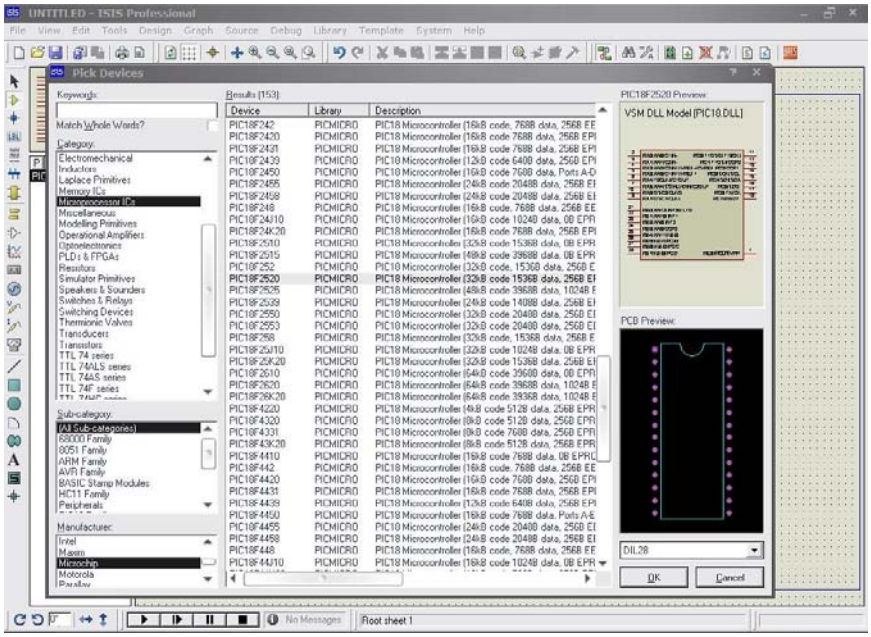


Figura 8

Menú de selección de los componentes electrónicos

2. MEMORIA DE CÁLCULO

2.1 Adecuación de la señal

En este capítulo se reflejarán todos los cálculos realizados para adecuar la onda del bajo eléctrico. También se expondrán los diagramas de flujo del programa que se ha diseñado.

2.1.1 Forma y amplitud

La forma genérica de onda de un bajo es la siguiente:

Figura 1

Forma de onda de la señal de un bajo eléctrico.

En la *Figura 1* podemos apreciar como según pasa el tiempo la onda se va atenuando. También se puede apreciar que en cada periodo aparece un pequeño pico que a lo largo del tiempo se acabará confundiendo con el pico principal de la onda.

En el osciloscopio de uno de los laboratorios de la universidad se midió la amplitud de la señal de la onda, y se extrajeron dos conclusiones:

1. La amplitud varía según cuan de fuerte toquemos la cuerda.
2. La cantidad de pequeños picos que aparecen en la onda y su amplitud varía según toquemos las cuerdas con los dedos o con una púa.
   En el primer caso la onda aparece mucho más nítida mientras que en el segundo caso se producen más picos residuales.

Tras unas sesiones experimentales, probando distintas fuerzas aplicadas a las cuerdas con los dedos, se calculó una amplitud de pico principal de, aproximadamente, **400 mV**.

2.1.2 Circuito de adecuación

El esquema del circuito de adecuación es el siguiente:

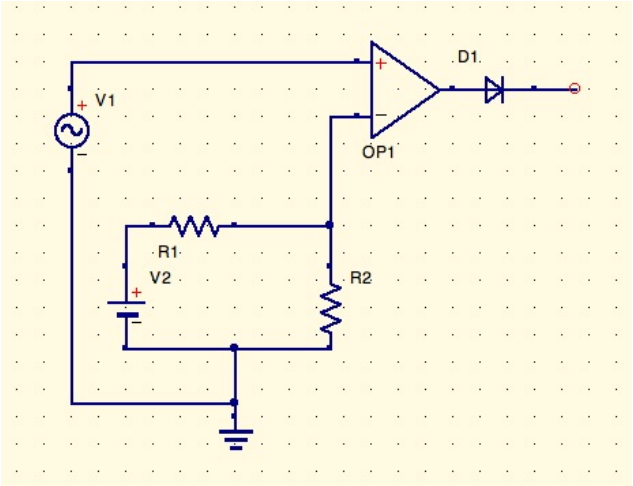


Figura 2

Con esta construcción con el AO 741 funcionando como comparador se pretende convertir la onda del bajo en una onda cuadrada y eliminar los picos que se han comentado en el capítulo *2.1.1*.

La fuente V1 representa la señal del bajo y el divisor de tensión, alimentado por V2, es con el que conseguimos el umbral con el que comparamos la entrada.

2.1.2.1 El AO 741

El amplificador operacional 741 es un operacional muy utilizado en la electrónica y es un componente con el que se ha ido trabajando durante la ingeniería técnica.

A la hora de buscar amplificador operacional para el afinador se necesitaba que fuera capaz de:

- Saturar a ± 5 V para obtener la onda cuadrada.
- Funcionar a bajas frecuencias, de 40 a 100 Hz.
- Aceptar tensiones de unos pocos mV.

Como el 741 cumple las exigencias anteriores es el que se ha utilizado para llevar a cabo el proyecto.

2.1.2.2 Divisor de tensión

Queremos que la tensión umbral de la comparación sea de 300 mV aproximadamente, cualquier pico que supere ese valor no será tomado en cuenta.

Así el divisor de tensión quedaría:

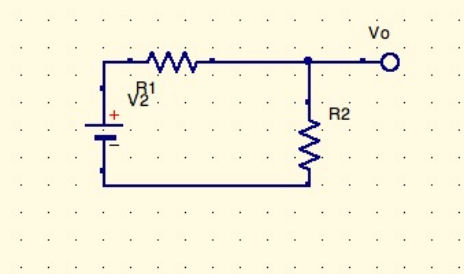


Figura 3

$$Vo = V2 \cdot \frac{R2}{R1 + R2}$$

Si queremos una Vo de 300 mV, forzamos R1=10kΩ y calculamos R2:

V2=5 V
R1=10kΩ

$$300mV = 5V \cdot \frac{R2}{10k\Omega + R2}$$

$$R2 = 638\Omega$$

El valor comercial más cercano es de 680 Ω que es el que se ha utilizado

2.1.2.3 Rectificador de onda

Siguiendo el proceso anterior obtenemos una onda cuadrada de V_{pp}= 10 V, con una parte del periodo V_p= 5 V y la otra V_p= -5 V.

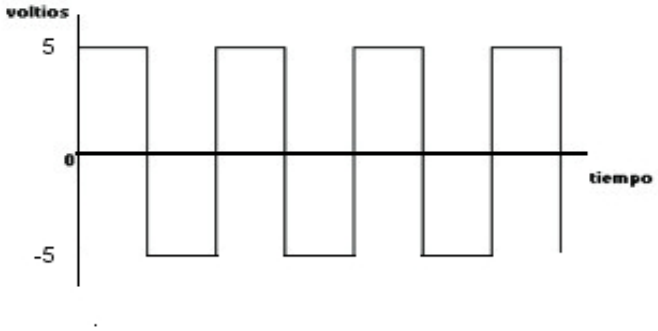


Figura 4

Al microcontrolador sólo le queremos pasar una onda cuadrada de V_{pp}= 5 V, así que colocamos un diodo a la salida del AO (como se aprecia en la *Figura 5)* de tal forma que elimine los semiperiodos negativos que no nos interesan.

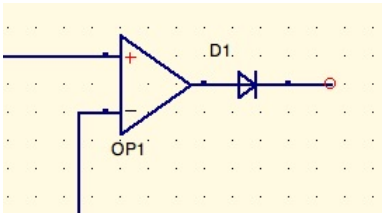


Figura 5

Ahora que ya se ha mostrado como se adecua la onda del bajo eléctrico ya podemos pasar a explicar los cálculos referentes a la afinación.

2.2 Muestreo de la onda

En este apartado se mostrarán los cálculos referentes al microcontrolador y la resolución de la frecuencia a la que oscila la cuerda del bajo.

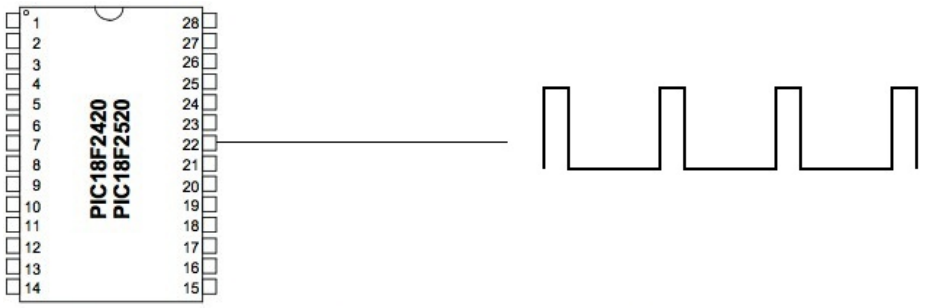
Todos los cálculos y configuraciones están hechas teniendo en cuenta que el microcontrolador trabaja a una frecuencia de 4 MHz mediante el oscilador interno que tiene el PIC.

2.2.1 Adquisición de la onda

La adquisición de la onda se realiza a través de la INT1 configurada para que detecte los flancos de subida. Cada vez que el IF de la INT1 toma valor ‘1’ la rutina de servicio a la interrupción se ejecuta.

El microcontrolador calcula la frecuencia de la onda cada 2 flancos ascendentes, que corresponden a un periodo de onda, y muestra el resultado por la LCD.

Hay que tener la precaución de borrar el IF vía software de la interrupción atendida. Si no lo hacemos así el IF siempre valdrá ‘1’ con lo que no se pedirá servicio a la interrupción por mucho que éstas vayan sucediendo.



REGISTER 9-3: INTCON3: INTERRUPT CONTROL REGISTER 3

R/W-1	R/W-1	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF
X	X		X	1		X	1

Figura 6

Petición de atención a la interrupción INT1IF=1, hay que borrarlo vía software

También hay que prestar atención a las prioridades de las interrupciones. El PIC 18F2520 es capaz de asignar prioridades a las distintas interrupciones: “low priority” o “high priority”. En este caso, la interrupción INT1 se ha asignado como “high priority”. Sin embargo, aunque INT1 sea una interrupción de alta prioridad, la única que tendría máxima prioridad y que saltaría antes que la nuestra sería la INT0 que viene así implementada de fábrica.

Para terminar, mientras no se atiende a ninguna interrupción el microcontrolador entra en modo IDLE, del que se hablará más adelante.

2.2.2 Timer 1

Para calcular la frecuencia de oscilación se ha usado el Timer 1. El Timer cuenta con las siguientes características:

- Se puede configurar mediante software como un timer o contador de 16 bits
- Tiene dos registros de cuenta de 8 bits (el TMR1H y TMR1L) que se pueden leer y escribir.
- Se puede utilizar como oscilador.
- Genera una interrupción al desbordarse su contador.
- Se puede utilizar como reloj a tiempo real.

2.2.2.1 Configuración

Sabemos que el bajo funciona, cuando está afinado, en un rango de entre 40 Hz y 100 Hz aproximadamente. En caso de desafinación el rango se amplía desde los 27,5 Hz hasta los 110 Hz. Entonces:

$$\frac{1}{27{,}5Hz} = 36{,}36ms$$

$$\frac{1}{110Hz} = 9{,}09ms$$

Este es el rango de tiempos que en el que el Timer 1 tiene que funcionar.

Por otro lado sabemos que el tiempo que tarda el Timer 1 en desbordarse es:

$$T = \frac{4}{Fosc} \cdot (preescaler) \cdot (65536 - TMR1)$$

Nos interesa que el timer no llegue a desbordarse, y sabemos por los cálculos anteriores que en el peor de los casos un periodo dura 0,036 s. Este es el tiempo máximo que ha de poder contar sin desbordar

Teniendo en cuenta que:

- Trabajamos con una frecuencia F_{osc}=4 MHz
- Queremos que el timer pueda contar hasta 36,36 ms como mínimo.

Calculamos:

$$T = \frac{4}{4MHz} \cdot (preescaler) \cdot (65536 - TMR)$$

$$36,36ms \geq \frac{4}{4MHz} \cdot (preescaler) \cdot (65536 - TMR)$$

$$36,36ms \cdot \frac{4MHz}{4} \geq (preescaler) \cdot (65536 - TMR)$$

Con un preescaler 1:1 quedaría:

$$TMR \leq 65536 - 36360$$

$$TMR \leq 29176$$

Este es el valor máximo que puede tomar el contador TMR para que sea capaz de calcular frecuencias de 27.5 Hz sin desbordar el Timer 1.

En el programa, el valor del TMR es de 0 puesto que cumple con la condición anterior y facilita los cálculos a la hora de determinar frecuencias.

2.2.3 Determinación de la frecuencia

El afinador trabaja en un rango que va desde los 27,5 Hz hasta los 110 Hz, así que sobre este rango se han hecho cuatro grupos de frecuencias que se corresponderán a las cuatro cuerdas del bajo:

- de 110 a 86 Hz → **sol (G)**
- de 86 a 63 Hz → **re (D)**
- de 63 a 48 Hz → **la (A)**
- de 48 a 27,5 Hz → **mi (E)**

Se ha escogido 110 Hz como el tope máximo por arriba y 27,5 Hz por abajo ya que corresponden con las frecuencias de la siguiente nota después de sol (G) y la anterior a mi (E).

Para calcular en que rango hay que situar la cuerda que estamos afinando primero hay que calcular la frecuencia de ésta. Esto se hace midiendo el tiempo que transcurre entre pulso y pulso. El calculo se realiza así:

```
freq1=TMR1L;
freq2=TMR1H;
freq0=freq2<<8|freq1;
freq3=freq0;
freq3=1/freq3;
freq3=(freq3*1000000);
```

Lo primero que se hace es leer el contenido del TMR en dos lecturas, primero el byte bajo y luego el alto. Cuando ya lo tenemos hay que pasar valor a Hz. Si tenemos presente la ecuación del apartado anterior:

$$T = \frac{4}{Fosc} \cdot (preescaler) \cdot (65536 - TMR1)$$

Para calcular la frecuencia sabiendo el valor del TMR solo hay que hacer:

$$f = \frac{1}{T}$$

Pero como ‘T’ tiene un exponente de 10^{-6} entonces:

$$f = \frac{1}{T} \cdot 1000000$$

Pasemos ahora a ver como se muestra el resultado al usuario. Lo explicamos en el capítulo siguiente.

2.3 Comunicación del resultado

Ya calculada la frecuencia se utiliza una LCD con el controlador HD44780 para comunicar al usuario el resultado. La pantalla es de 16 x 2 (2 filas de 16 caracteres cada una). Juntamente con la LCD se usan también unos diodos LED para facilitar la comprensión del resultado.

La pantalla LCD puede trabajar de dos modos distintos:

- Modo 8 bits
- Modo 4 bits

Para llevar a cabo el proyecto se ha escogido trabajar con el modo 4 bits. Eso significa que de los 8 bits de datos que tiene la pantalla LCD (D0-D7) se usan solamente 4, los bits altos (D4-D7).

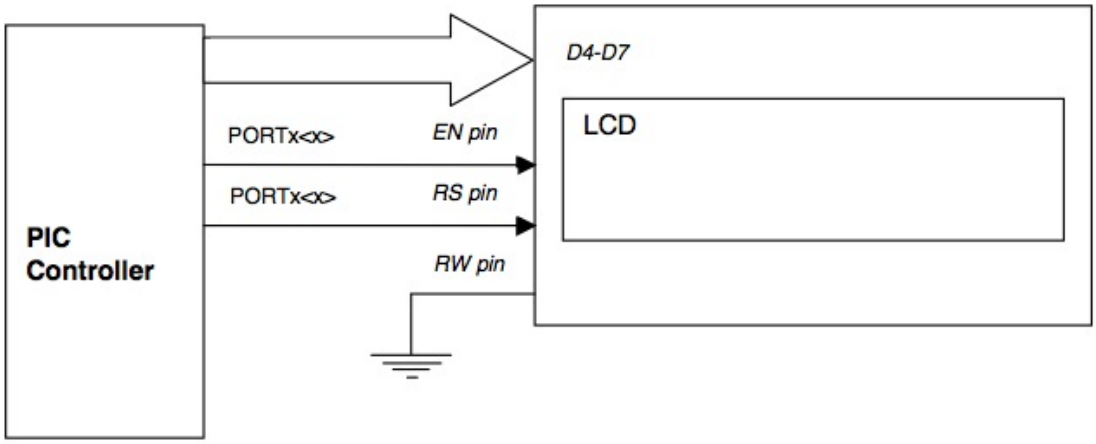


Figura 7

LCD conectada en modo 4 bits

Para poder mostrar el resultado del cálculo de la frecuencia por pantalla, hay que convertir el valor numérico para que el microcontrolador de la LCD lo interprete ya que solamente podemos enviarle caracteres. Así:

```
freqp1 = floor(freq3);
freqp2 = freq3 - freqp1;
freqint1 = (int)freqp1;
freqint2 = (int)100*freqp2;
sprintf(aaa,"%i.%02i", freqint1,freqint2);
```

Este código sirve para pasar el contenido de freq3 al *string* aaa, que es el que más adelante se envía a la LCD para que muestre por pantalla con una precisión de 2 decimales. Además suple una carencia que observé al programar con el compilador C18 para MPLAB, y es que la función *sprintf()* no es capaz de manejar variables con coma flotante, por lo que la idea inicial:

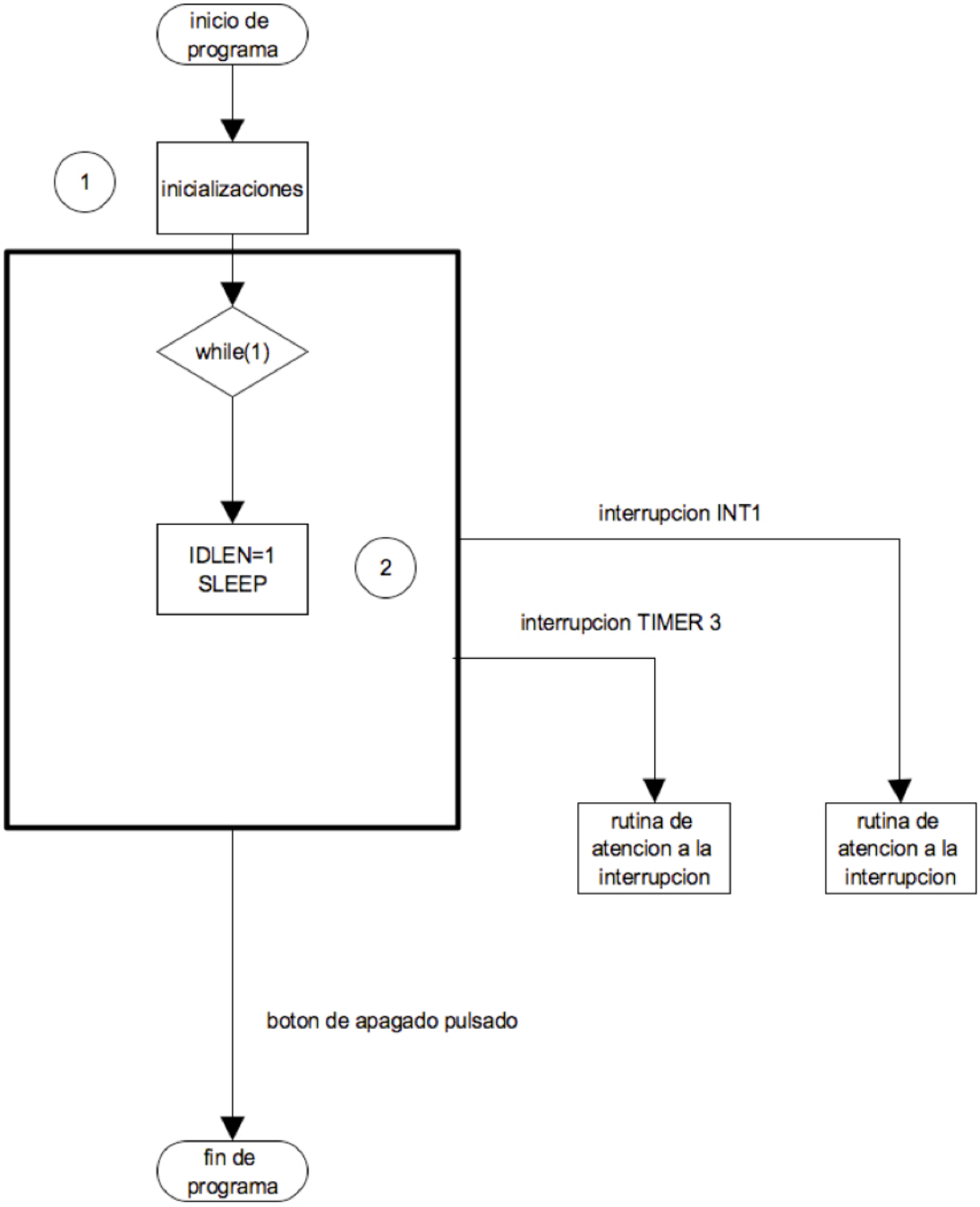
```
sprintf(aaa,"%5.2f”, freq3);
```

no servía, lo que llevó a plantearse una solución. El resultado es el código comentado.

2.4 Diagramas de flujo

Después de haber explicado los cálculos, se dedicará un apartado para explicar los diagramas de flujo del programa que ejecuta el afinador.

2.4.1 Programa principal



1. **Inicializaciones**

Se configuran todos los registros del microcontrolador y se inicializa la pantalla LCD:

- TIMER 1 (inicializado según lo descrito en el apartado 2.2.2)

- TIMER 3:
  - El uso del Timer 3 y su configuración se explcará en el punto 2.5.3

- OSCCON:
  - Desde el OSCCON se controla todo lo relacionado con el funcionamiento del reloj del microcontrolador y como se comportará este ante los diferentes modos de *sleep* que hay.

- ADCON0 y ADCON1:
  - Son los registros de control del conversor A/D. Están inicializados para desactivar el conversor ya que comparte pines con los bits de control de la LCD y habría incompatibilidades.

- PORTA, PORTB y PORTC:
  - Contenido de los puertos A, B y C. Inicializados los tres con valor 0x00.

- TRISA, TRISB y TRISC:
  - Configura los pines I/O. Inicializados todos como output.

- INTCON3:
  - Palabra de control con la que se habilitan y deshabilitan las interrupciones necesarias.

- RCON:
  - Registro que controla las prioridades de las interrupciones. Ya se ha comentado antes que vamos a usar prioridades, así que las habilitamos desde aquí.

- XLCDinit():
  - Rutina de inicialización de la pantalla LCD

- INTCON:
  - Palabra desde la cual se puede modificar los dos GIE (Global Interrupt Enable) que hay, el GIEH para las interrupciones de alto prioridad y el GIEL para las de baja. Esta es la última inicialización y a partir de ahora los GIE valen ‘1’ así que las interrupciones se comenzaran a atender.

2. Microcontrolador en modo *idle*

Mientras el microcontrolador está esperando a que le llegue alguna interrupción se pone la CPU en reposo hasta que algún evento la despierte.

Hay distintos modos de reposo para escoger y dependiendo del tipo de configuración del oscilador que se haya escogido previamente se usará uno u otro.

TABLE 3-1: POWER-MANAGED MODES

Mode	OSCCON<7,1:0> Bits		Module Clocking		Available Clock and Oscillator Source
	IDLEN(1)	SCS1:SCS0	CPU	Peripherals	
Sleep	0	N/A	Off	Off	None – All clocks are disabled
PRI_RUN	N/A	00	Clocked	Clocked	Primary – LP, XT, HS, HSPLL, RC, EC and Internal Oscillator Block(2). This is the normal full power execution mode.
SEC_RUN	N/A	01	Clocked	Clocked	Secondary – Timer1 Oscillator
RC_RUN	N/A	1x	Clocked	Clocked	Internal Oscillator Block(2)
PRI_IDLE	1	00	Off	Clocked	Primary – LP, XT, HS, HSPLL, RC, EC
SEC_IDLE	1	01	Off	Clocked	Secondary – Timer1 Oscillator
RC_IDLE	1	1x	Off	Clocked	Internal Oscillator Block(2)

Note 1: IDLEN reflects its value when the SLEEP instruction is executed.
2: Includes INTOSC and INTOSC postscaler, as well as the INTRC source.

Figura 8

Cuadro de las distintas opciones de configuración

En nuestro caso, teniendo en cuenta que utilizamos el bloque oscilador interno, queremos dormir la CPU pero dejar los periféricos activos, hemos escogido el modo RC_IDLE.

Cuando una interrupción se produzca la CPU saldrá del estado de reposo y realizará la tarea necesaria.

FIGURE 3-7: TRANSITION TIMING FOR ENTRY TO IDLE MODE

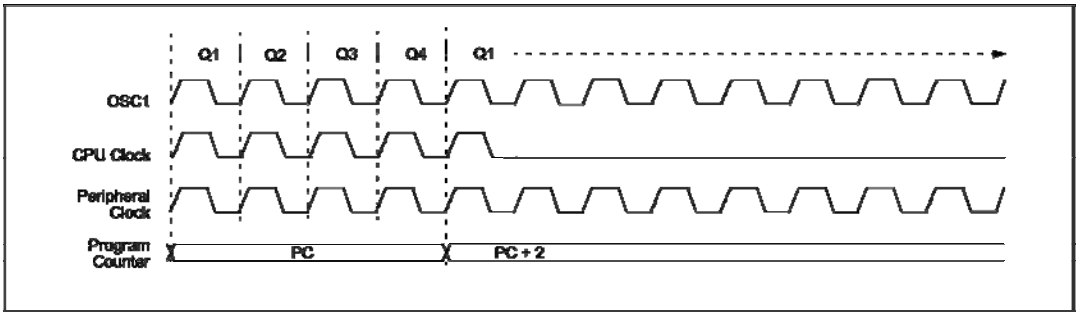


FIGURE 3-8: TRANSITION TIMING FOR WAKE FROM IDLE TO RUN MODE

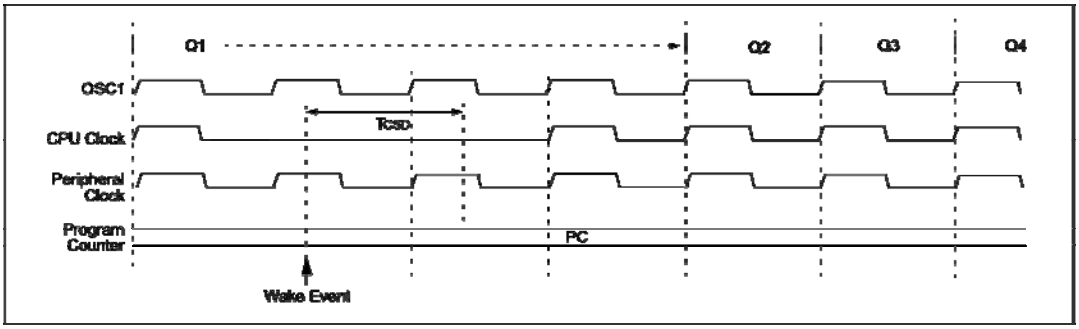


Figura 9

Tiempos de entrada y salida de modo idle

Para saber si trabajar con el microcontrolador en modo *idle* sale a cuenta se realizaron una serie de cálculos.

Por un lado sabemos que el microcontrolador saldrá del modo *idle* siempre que se produzca:

- Interrupción externa INT1:

  El más rápido de los casos es cuando se esté afinando una cuerda que oscile a 110 Hz

$$T = \frac{1}{110Hz} = 9{,}09ms$$

- Interrupción por desbordamiento de Timer 3:

  Se produce una interrupción de Timer 3 cada

$$T = 0{,}52\text{s}$$

  En el apartado 2.5.3 se explica con más detalle los tiempos del Timer 3.

Para que el microcontrolador salga del modo *idle* hace falta un periodo T_{CSD} para que la CPU despierte y empiece a ejecutar código

TABLE 26-10: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, POWER-UP TIMER AND BROWN-OUT RESET REQUIREMENTS

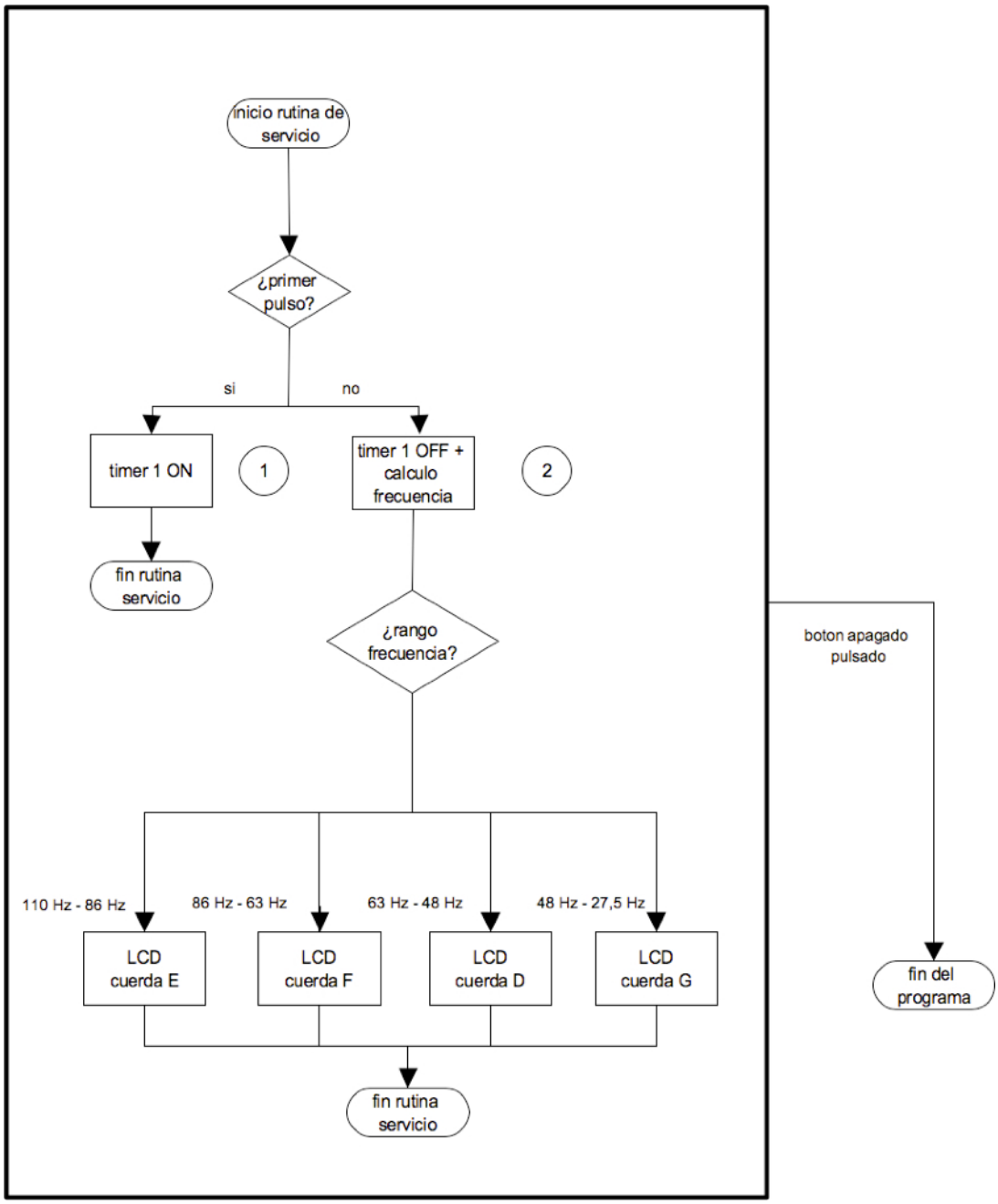
Param. No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions
30	TmcL	$\overline{\text{MCLR}}$ Pulse Width (low)	2	—	—	µs	
31	TWDT	Watchdog Timer Time-out Period (no postscaler)	3.56	4.10	4.82	ms	
32	TOST	Oscillator Start-up Timer Period	1024 TOSC	—	1024 TOSC	—	TOSC = OSC1 period
33	TPWRT	Power-up Timer Period	57.0	65.5	77.1	ms	
34	TIOZ	I/O High-Impedance from $\overline{\text{MCLR}}$ Low or Watchdog Timer Reset	—	2	—	µs	
35	TBOR	Brown-out Reset Pulse Width	200	—	—	µs	VDD ≤ BVDD (see D005)
36	TIVRST	Time for Internal Reference Voltage to become Stable	—	20	50	µs	
37	TLVD	High/Low-Voltage Detect Pulse Width	200	—	—	µs	VDD ≤ VLVD
38	TCSD	CPU Start-up Time	5	—	10	µs	
39	TIOBST	Time for INTOSC to Stabilize	—	1	—	ms	

El cuadro de la *Figura 10* nos muestra que el periodo T_{CSD}, en el peor de los casos, es de 10 µs. Si comparamos este tiempo con el más rápido de los calculados anteriormente:

$$9{,}09ms >> 10\mu s$$

Así que nos da tiempo de sobra para poner en modo *idle* y despertar el microcontrolador.

2.4.2 Rutina de servicio a la interrupción 'high_priority'



Cuando se atiende a una interrupción, hay un contador que determina si es el primer o el segundo flanco el que ha levantado el INT1IF. También se reinicia el contador del bucle que hay en el *main* del programa y se impide que el programa entre en él hasta que no termine de atenderse a la interrupción.

1. **Encender el TIMER 1**

   Cuando es al primer flanco al que se atiende, el programa pone el TIMER 1 en marcha mediante el bit TMR1ON del registro T1CON:

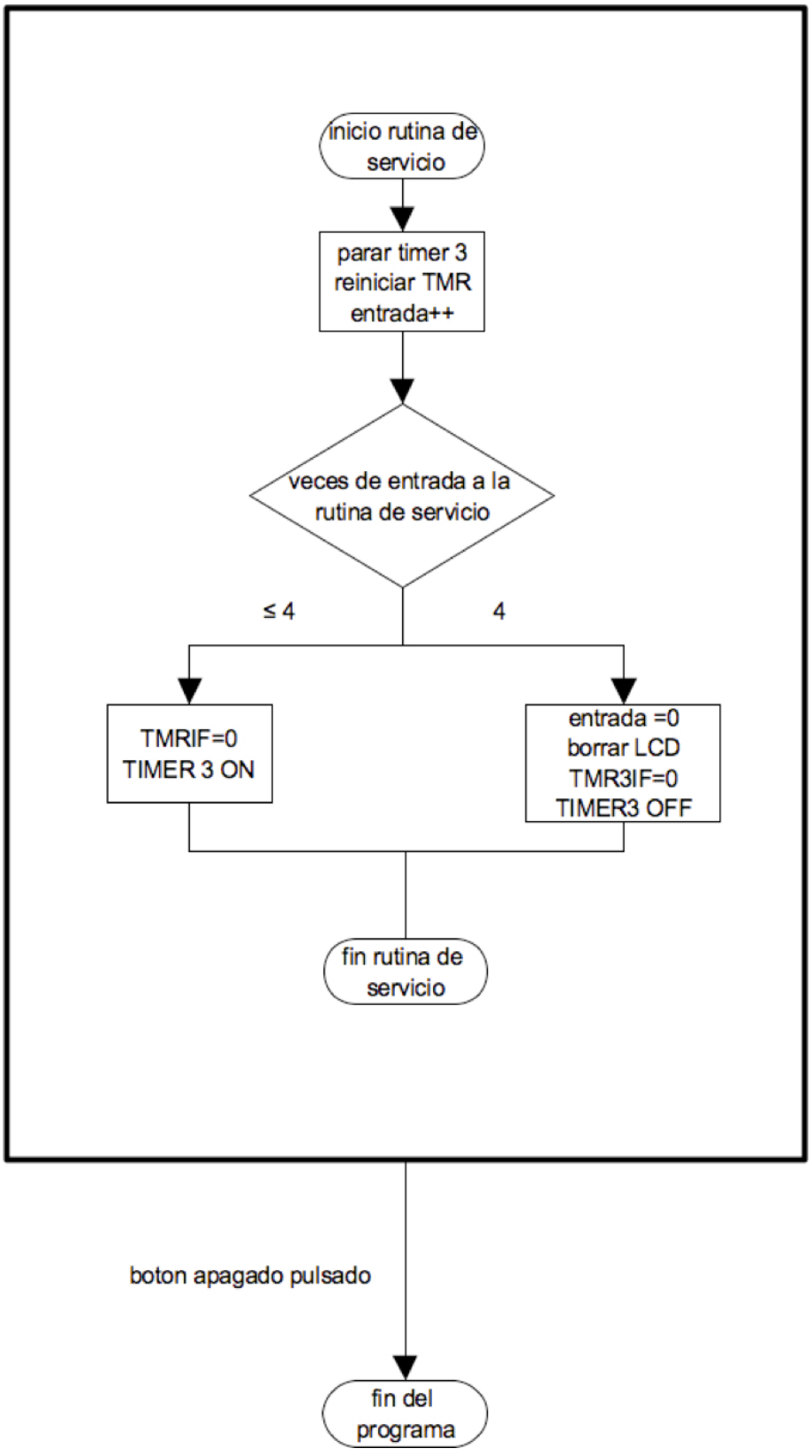
REGISTER 12-1: T1CON: TIMER1 CONTROL REGISTER

R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
X	X	X	X	X	X	X	1

2. **Apagado del TIMER 1 y cuentas**

   En el segundo flanco de subida se para el *timer* y se lee su contenido como se ha descrito en el apartado 2.2.3.
   Tras determinar la frecuencia a la que se oscila la cuerda y clasificarla en uno de los cuatro rangos posibles, se muestra por pantalla.

2.4.3 Rutina de servicio a la interrupción 'Low Priority'



2.4.3.1 Timer 3

Para apagar la pantalla se ha utilizado rutina de servicio a la interrupción basada en el desbordamiento del timer 3. La idea es apagar la LCD 2 segundos después de haber mostrado el último resultado por pantalla.

El Timer 3 incorpora estas características:

- Se puede configurar como timer/contador de 16 bits
- Tiene dos registros de cuentas de 8 bits, TMR3L y TMR3H
- Puede elegirse la fuente de reloj con la que trabajará, tanto externa como interna así como Timer 1 en caso de que este trabaje como reloj.
- Genera interrupción al desbordarse su contador

Como queremos apagar la pantalla a los 2 segundos aprovechando la interrupción que genera el timer 3 y sabemos que:

$$T = \frac{4}{4\,MHz} \cdot (preescaler) \cdot (65536 - TMR3)$$

Calculamos la cuenta que debemos introducirle al TMR3 y el preescaler en caso de que fuera necesario:

$$2s = \frac{4}{4\,MHz} \cdot (preescaler) \cdot (65536 - TMR3)$$

$$2000000 = (preescaler) \cdot (65536 - TMR3)$$

El resultado deja claro que aun dejando el TMR3 a 0 nos hace falta un preescaler. Lo calculamos así:

$$preescaler = \frac{2000000}{65536} = 30{,}52$$

Sabemos que el Timer 3 cuenta con 4 configuraciones distintas de preescaler que son:

REGISTER 14-1: T3CON: TIMER3 CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON
bit 7							bit 0

bit 5-4 **T3CKPS1:T3CKPS0**: Timer3 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

El preescaler más grande que nos deja escoger es de una proporción 1:8 y según los cálculos anteriores necesitaríamos uno de proporción 1:30 aproximadamente. Para solventar el problema se ha creado un contador dentro de la rutina de servicio a la interrupción del timer 3. Cada 4 veces que se atienda a la interrupción del timer 3 se apagará la LCD.

$$T = \frac{4}{4\,MHz} \cdot (1:8) \cdot 65536 \cdot 4$$

$$T = 2{,}1s$$

Una vez apagada la pantalla LCD el microcontrolador quedará en modo IDLE esperando a que le llegue una interrupción.

2.5 Alimentación del circuito

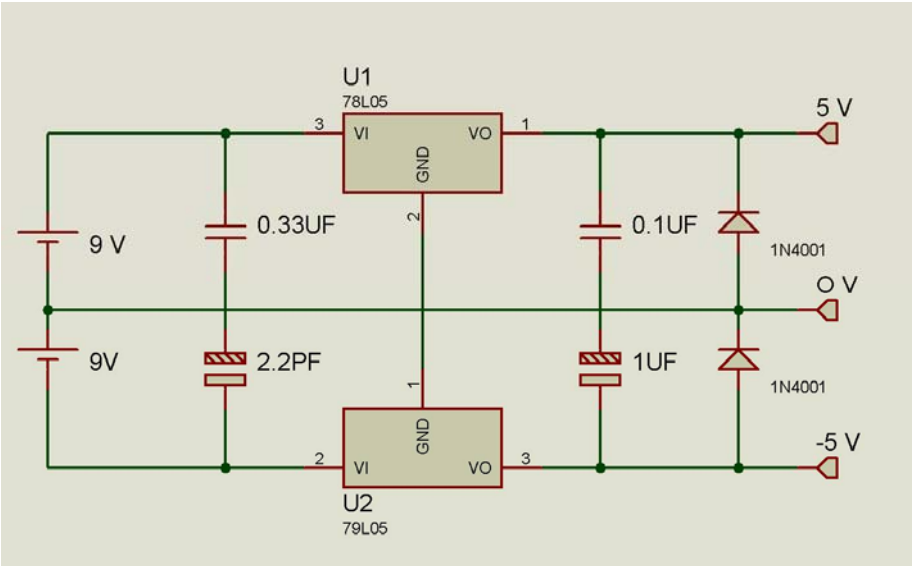
La circuitería del afinador está compuesta de varias partes que necesitan ser alimentadas:

- AO 741:
  - Necesita dos alimentaciones: +5 V y -5 V
- PIC 18F2520:
  - Necesita una alimentación de +5 V y 0 V
- LCD:
  - Necesita como el microcontrolador: +5 V y 0 V

Visto esto, queda claro que necesitamos un sistema que suministre ±5 V y masa. Para ello se ha utilizado una serie de pilas y rectificadores de tensión:

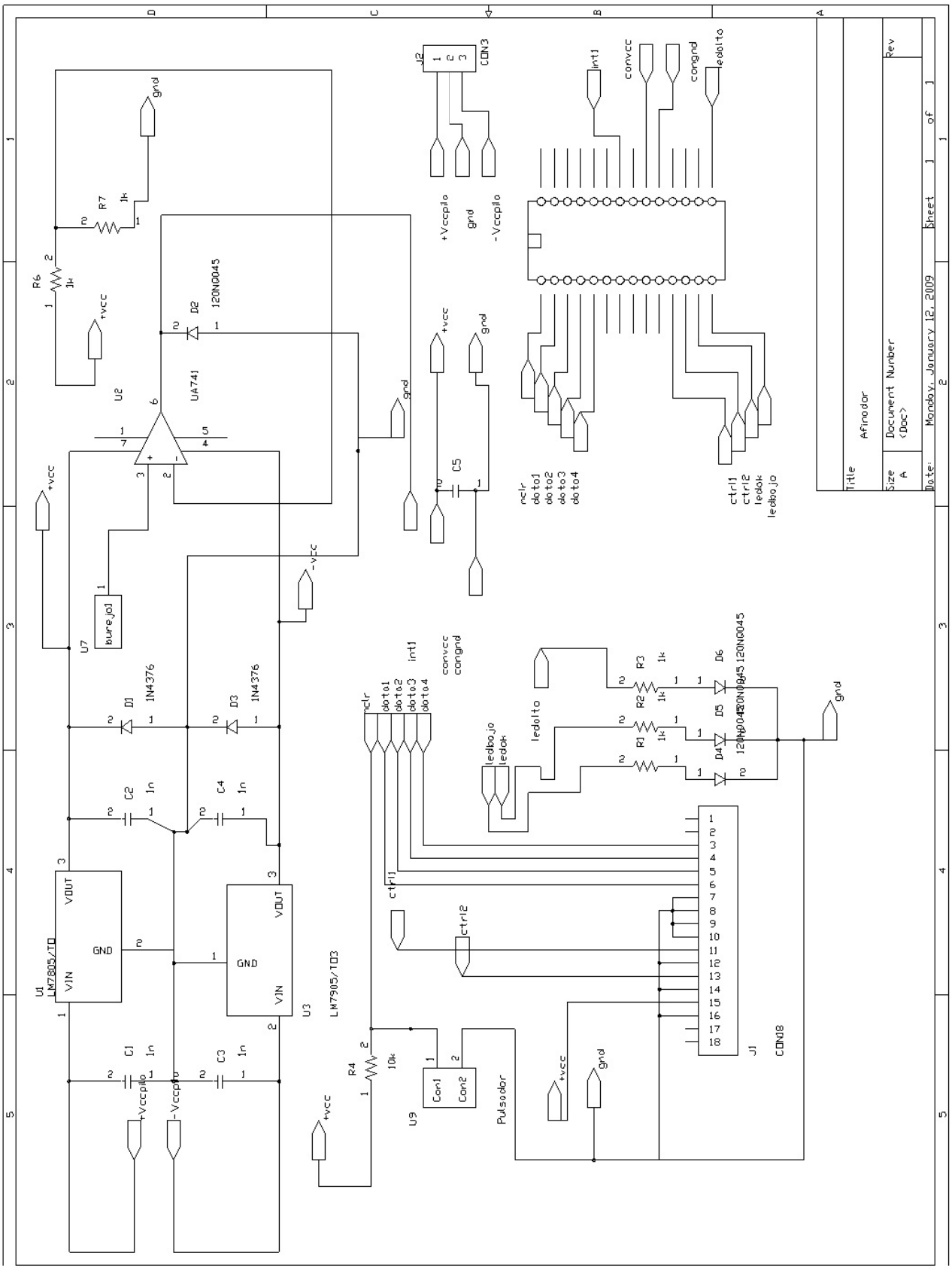
- 2 Pilas de 9 V
- Rectificador de tensión 78L05:
  - Con una entrada de 7 V como mínimo y hasta un tope de 20 V, suministra una tensión constante a la salida de 5 V.
- Rectificador de tensión 79L05:
  - Funciona como el anterior pero con tensiones negativas: se le puede aplicar tensiones de entre -7 V y -20 V y suministra una tensión constante a la salida de -5V

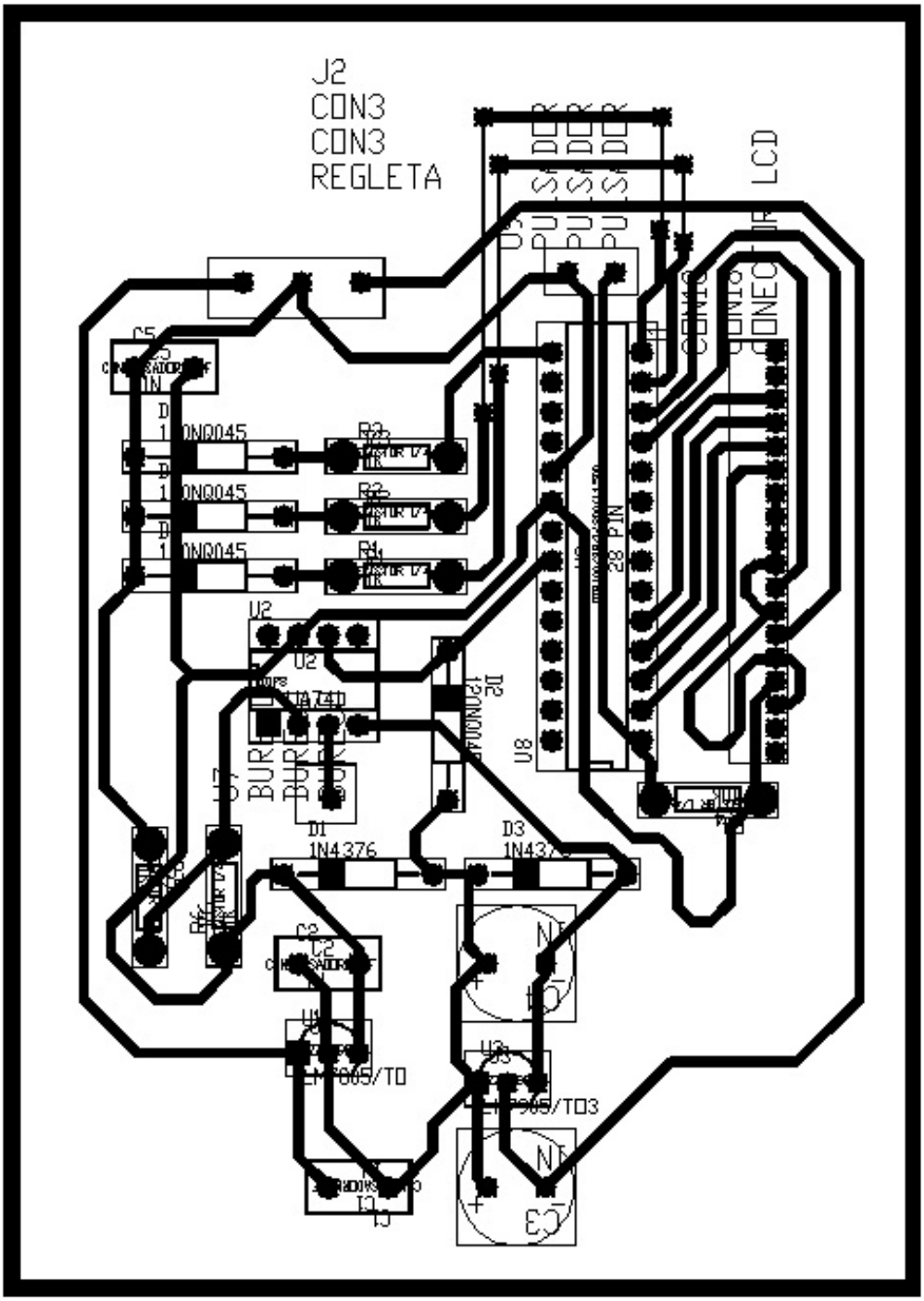
Quedando la alimentación así:



3. PLANOS

3.1 Placa PCB




J2
CON3
CON3
REGLETA
LCD
28 PIN
U8
U2
D1
1N4376
D3
1N437
C2
C3

4. PRESUPUESTO

4.1 Introducción

En este capítulo se reflejará el coste que ha supuesto la realización del afinador.

Para eso se ha dividido los costes en:

- ***Componentes:*** Precios de los componentes que se han utilizado tanto para el primer prototipo en placa de topos como el modelo final en la PCB.
- ***Mano de obra:*** La cuantía de recursos humanos dedicados al proyecto.

4.2 Lista de precios unitarios

4.2.1 Componentes

CÓDIGO	U	DESCRIPCIÓN	PRECIO
C_01	u	Resistencia 100 Ω	CINCO CÉNTIMOS
C_02	u	Resistencia 10 kΩ	CINCO CÉNTIMOS
C_03	u	Resistencia 1 kΩ	CINCO CÉNTIMOS
C_04	u	Resistencia 680 Ω	CINCO CÉNTIMOS
C_05	u	Diodo 1N4001	SIETE CÉNTIMOS
C_06	u	Diodo 1N4007	NUEVE CÉNTIMOS
C_07	u	Condensador 0,33 μF	DOCE CÉNTIMOS
C_08	u	Condensador 0,1 μF	NUEVE CÉNTIMOS
C_09	u	Condensador 2,2 μF	DIECIOCHO CÉNTIMOS
C_10	u	Condensador 1 μF	DIECISEIS CÉNTIMOS
C_11	u	Regulador 78L05	CATORCE CÉNTIMOS
C_12	u	Regulador 79L05	CATORCE CÉNTIMOS

CÓDIGO	U	DESCRIPCIÓN	PRECIO
C_13	u	Diodo LED rojo	TREINTA Y DOS CÉNTIMOS
C_14	u	Diodo LED azul	TREINTA Y DOS CÉNTIMOS
C_15	u	Zócalo DIP 8	QUINCE CÉNTIMOS
C_16	u	Zócalo DIP 28	CINUENTA Y CUATRO CÉNTIMOS
C_17	u	Conector Jack Hembra 6 mm	UN EURO con CINCUENTA Y SIETE CÉNTIMOS
C_18	u	Conector RJ-11	OCHENTA Y NUEVE CÉNTIMOS
C_19	u	Interruptor	VEINTE CÉNTIMOS
C_20	u	AO 741	TREINTA Y NUEVE CÉNTIMOS
C_21	u	Microcontrolador PIC 18F2520	DOS EUROS con DIECISEIS CÉNTIMOS
C_22	u	Pantalla LCD HD44780	OCHO EUROS con NOVENTA Y NUEVE CÉNTIMOS
C_23	u	Placa de topos	CINCO EUROS con TREINTA CÉNTIMOS
C_24	u	Pila de 9 V	UN EURO con CUARENTA CÉNTIMOS
C_25	u	Soporte para pila	NOVENTA CÉNTIMOS
C_26	u	Cable ∅ 0,5 mm^2	CUARENTA Y CUATRO CÉNTIMOS
C_27	u	Tornillo inox. M3x6	DIEZ CÉNTIMOS
C_27	u	Caja de afinador	SIETE EUROS con CUARENTA Y SEIS CÉNTIMOS
C_28	u	Placa PCB	SEIS EUROS

4.2.2 Mano de obra

CÓDIGO	U	DESCRIPCIÓN	PRECIO
M_01	h	Ingeniero Programador	DIECIOCHO EUROS
M_02	h	Técnico de montaje	NUEVE EUROS

4.3 Cuadro de descompuestos

4.3.1 Componentes

CÓDIGO	DESCRIPCIÓN	Uds.	PRECIO	IMPORTE
C_01	**Resistencia 100 Ω** Resistencia de ¼ W	6	0,05€	0,30€
C_02	**Resistencia 10 kΩ** Resistencia de ¼ W	2	0,05€	0,10€
C_03	**Resistencia 1 kΩ** Resistencia de ¼ W	2	0,05€	0,10€
C_04	**Resistencia 680 Ω** Resistencia de ¼ W	2	0,05€	0,10€
C_05	**Diodo 1N4001** Rectificador de silicio de 1A	2	0,07€	0,14€
C_06	**Diodo 1N4007** Rectificador de silicio de 1A	4	0,09€	0,36€
C_07	**Condensador 0,33 µF** Dieléctrico de papel	2	0,12€	0,24€
C_08	**Condensador 0,1 µF** Dieléctrico de papel	2	0,09€	0,18€
C_09	**Condensador 2,2 µF** Dieléctrico electrolítico	2	0,18€	0,36€
C_10	**Condensador 1 µF** Dieléctrico electrolítico	2	0,16€	0,32€

CÓDIGO	DESCRIPCIÓN	Uds.	PRECIO	IMPORTE
C_11	**Regulador 78L05** Regulador de tensión con una tensión de salida de +5 V	2	0,14€	0,28€
C_12	**Regulador 79L05** Regulador de tensión con una tensión de salida de -5 V	2	0,14€	0,28€
C_13	**Diodo LED rojo** LED 5 mm diámetro	4	0,32€	1,28€
C_14	**Diodo LED azul** LED 5 mm diámetro	2	0,32€	0,64€
C_15	**Zócalo DIP 8** Zócalo de 8 pines	2	0,15€	0,30€
C_16	**Zócalo DIP 28** Zócalos de 8 pines	2	0,54€	1,08€
C_17	**Conector Jack Hembra 6 mm** Conector para enchufar el cable del bajo	2	1,57€	3,14€
C_18	**Conector hembra RJ-11** Conector para programar el microcontrolador	2	0,89€	1,78€
C_19	**Interruptor** Interruptor normalmente cerrado	4	0,20€	0,08€
C_20	**AO 741** Amplificador operacional	2	0,39€	0,78€
C_21	**Microcontrolador PIC 18F2520** Microcontrolador de 8 bits de la casa *Microchip*	2	2,16€	4,32€
C_22	**Pantalla LCD HD44780** Pantalla LCD que incorpora el controlador HD44780	1	8,99€	8,99€
C_23	**Placa de topos** Placa de topos para la construcción del primer prototipo	1	5,30€	5,30€
C_24	**Pila de 9 V** Pila para alimentar al afinador	2	1,40€	2,80€
C_25	**Soporte para pila** Soporte para fijar las pilas a la placa	2	0,90€	1,80€

CÓDIGO	DESCRIPCIÓN	Uds.	PRECIO	IMPORTE
C_26	**Cable ∅ 0,5 mm^2** 2 metros de cable de ∅ 0,5 mm^2	1	0,44€	0,44€
C_27	**Tornillo inox. M3x6** Tornillo de acero inoxidable de métrica 3	8	0,10€	0,08€
C_27	**Caja de afinador** Caja que contiene las partes del afinador de bajo	1	7,46€	7,46€
C_28	**Placa PCB** Placa PCB con el diseño final del afinador de bajo	1	6€	6€
	TOTAL			**50,31€**

Asciende a un total de CINCUENTA EUROS con TREINTA Y UN CÉNTIMOS.

4.3.2 Mano de obra

CÓDIGO	DESCRIPCIÓN	Uds.	PRECIO	IMPORTE
M_01	**Ingeniero Programador**	320	16€	5120€
M_02	**Técnico de montaje**	8	7€	56€
	TOTAL			**5176€**

Asciende a un total de CINCO MIL CIENTO SETENTA Y SEIS EUROS.

4.4 Resumen del presupuesto

RESUMEN	IMPORTE
COMPONENTES	50,31€
MANO DE OBRA	5176€
TOTAL EJECUCIÓN MATERIAL	**5226,31€**
5% Gasto General	261,31€
PRESUPUESTO SIN IVA	**5487,62€**
16% IVA	878,02€
PRESUPUESTO TOTAL	**6365,64€**

El presupuesto total asciende a una cantidad de SEIS MIL TRESCIENTOS SESENTA Y CINCO EUROS con SESENTA Y CUATRO CÉNTIMOS.

4.5 Estudio de precio de venta cara al público del afinador

En este apartado se mostrará una aproximación al precio que alcanzaría el afinador en el mercado. Para ello se ha estudiado la relación entre costes e ingresos que supondría realizar 10000 unidades.

Se evaluarán cuatro factores:

- **Costes fijos:** Son los costes que no dependen del número de unidades a producir.
- **Costes variables:** Son los costes de producción que dependen del número de unidades que se vayan a producir.
- **Costes totales:** La suma de costes fijos y costes variables
- **Ingresos:** El beneficio resultante de haber vendido las unidades

4.5.1 Costes fijos

Los costes fijos sería el presupuesto calculado en el apartado anterior, que asciende a:

6387,07 €

4.5.2 Costes variables

Como los costes variables dependen de la cantidad de componentes que se comprarán en relación al número de unidades que vamos a construir, calcularemos otra vez el precio de los componentes en relación a las 10000 unidades.

CÓDIGO	DESCRIPCIÓN	Uds.	PRECIO	IMPORTE
C_01	**Resistencia 100 Ω** Resistencia de ¼ W	30000	0,01€	300€
C_02	**Resistencia 10 kΩ** Resistencia de ¼ W	10000	0,01€	100€
C_03	**Resistencia 1 kΩ** Resistencia de ¼ W	10000	0,01€	100€

CÓDIGO	DESCRIPCIÓN	Uds.	PRECIO	IMPORTE
C_04	**Resistencia 680 Ω** Resistencia de ¼ W	10000	0,01€	100€
C_05	**Diodo 1N4001** Rectificador de silicio de 1A	10000	0,02€	200€
C_06	**Diodo 1N4007** Rectificador de silicio de 1A	20000	0,04€	800€
C_07	**Condensador 0,33 µF** Dieléctrico de papel	10000	0,03€	300€
C_08	**Condensador 0,1 µF** Dieléctrico de papel	10000	0,04€	400€
C_09	**Condensador 2,2 µF** Dieléctrico electrolítico	10000	0,05€	500€
C_10	**Condensador 1 µF** Dieléctrico electrolítico	10000	0,04€	400€
C_11	**Regulador 78L05** Regulador de tensión con una tensión de salida de +5 V	10000	0,03€	300€
C_12	**Regulador 79L05** Regulador de tensión con una tensión de salida de -5 V	10000	0,03€	300€
C_13	**Diodo LED rojo** LED 5 mm diámetro	20000	0,02€	400€
C_14	**Diodo LED azul** LED 5 mm diámetro	10000	0,02€	200€
C_15	**Zócalo DIP 8** Zócalo de 8 pines	10000	0,07€	700€
C_16	**Zócalo DIP 28** Zócalos de 8 pines	10000	0,08€	800€
C_17	**Conector Jack Hembra 6 mm** Conector para enchufar el cable del bajo	10000	0,10€	1000€
C_18	**Interruptor** Interruptor normalmente cerrado	20000	0,06€	1200€
C_19	**AO 741** Amplificador operacional	10000	0,06€	600€

CÓDIGO	DESCRIPCIÓN	Uds.	PRECIO	IMPORTE
C_20	**Microcontrolador PIC 18F2520** Microcontrolador de 8 bits de la casa *Microchip*	10000	1,50€	15000€
C_21	**Pantalla LCD HD44780** Pantalla LCD que incorpora el controlador HD44780	10000	1€	10000€
C_27	**Caja de afinador** Caja que contiene las partes del afinador de bajo	10000	1,10€	11000€
C_28	**Placa PCB** Placa PCB con el diseño final del afinador de bajo	10000	1€	10000€
	TOTAL			**53300€** **5,33€/u**

También sería un gasto variable el número de trabajadores que contrataremos y su sueldo. Supongamos que la producción la llevamos a cabo en China y tenemos contratados 2 empleados. El sueldo medio de un empleado chino se sitúa en los 274€ al mes, por lo que:

$$\frac{2 \cdot 274\,€}{10000u}$$

$$0{,}054\,€/\,u$$

Habría un recargo de 0,05€ por cada unidad de afinador producida.

4.5.3 Costes totales

El coste unitario de cada afinador teniendo en cuenta una producción de 10000 unidades es de:

$$CT = CF + CV$$

$$CT = \frac{6387,07\,€}{10000u} + 5,33€/\ u + \frac{2 \cdot 274\,€}{10000u}$$

$$CT = 6,02\,€/\ u$$

4.5.4 Ingresos

Si vendemos cada unidad producida a las tiendas a un precio de 10,02€ supondría un beneficio de 4€/u, siendo el beneficio total:

$$4\,€/\ u \cdot 10000$$

$$beneficio = 40000\,€$$

5. PLIEGO DE CONDICIONES

5.1 Introducción

En este capítulo se desarrollará un pliego de condiciones que cumple meramente una función didáctica. Todos los datos aquí reflejados son inventados y sólo pretenden dotar al pliego de condiciones de coherencia.

5.2 Reunidos

Sr. José Ortega Munilla, director general de la empresa *Swara S.A*, con domicilio fiscal situado en la Av. Guida d'Arezzo, 7, 42005 Tarragona.

Y

Sr. Daniel Díez Díaz-Calonge, ingeniero técnico industrial de la cooperativa *Proyectos Musicales*, con domicilio fiscal situado en la Calle Miguel Brieva, 44, 43001 Tarragona.

5.3 Exponen

5.3.1 Primero

Que la empresa *Swara S.A*, habiendo demandado proyecto sobre la realización de un afinador de bajo a la cooperativa *Proyectos Musicales*, se compromete a abonar la cantidad acordada.

5.3.2 Segundo

Que la cooperativa *Proyectos Musicales* respetará y cumplirá con el acuerdo de confidencialidad alcanzado por las dos partes respecto al proyecto desarrollado por la mentada cooperativa.

5.4 Cláusulas

5.4.1 Objeto de acuerdo

El acuerdo pactado por las dos partes es el de la realización de un afinador de bajo por parte de la cooperativa *Proyectos Musicales* para ser lanzado próximamente al mercado y que cumpla con las características demandadas por la empresa *Swara S.A.*

5.4.2 Condiciones de aceptación de proyecto

La cooperativa *Proyectos Musicales* acepta las condiciones exigidas por la empresa demandante reflejadas en la memoria descriptiva del proyecto.

5.4.3 Demora en del desarrollo

La cooperativa *Proyectos Musicales* indemnizará a la empresa *Swara S.A.* si las demoras en el desarrollo del producto son debidas a errores en la realización del proyecto presente.

5.4.4 Resolución de conflictos

Tras agotar las vías de acuerdo privado entre las dos partes implicadas y llegados a la imposibilidad de alcanzar un pacto adecuado y aceptable para las dos partes en caso de hipotético conflicto, las dos partes acuerdan que se someterán al arbitraje del Tribunal Arbitral de Barcelona de l'Associació Catalana per l'Arbitratge, cuya resolución será de obligada adopción por parte de los implicados.

Como muestra de conformidad, las dos partes firman el presente documento,

Por parte de *Swara S.A:*

José Ortega Munilla

Por parte de *Proyectos Musicales:*

Daniel Díez Díaz-Calonge

Tarragona, 15 de enero de 2009

6. ANEXOS

6.1 Código Fuente

```
/*-------------------------------------------------------------------------

     AFINADOR 1.0
     AUTOR: Daniel Díez Díaz-Calonge
     FECHA: 15/01/2009
     PROYECTO: Afinador de Bajo Eléctrico controlado por microcontrolador

-------------------------------------------------------------------------*/



#include <stdio.h>
#include <stdlib.h>
#include <delays.h>
#include "C:\Documents and Settings\Administrador\Escritorio\XLCD\xlcd.h" // libreria de
control de la LCD
#include <p18f2520.h>
#include <string.h>
#include <float.h>
#include <math.h>



/*-----------------------------

     CONTROL CURSOR LCD

------------------------------*/

#define XLCDCursorOnBlinkOn()        XLCDCommand(0x0F)
#define XLCDCursorOnBlinkOff()       XLCDCommand(0x0E)
#define XLCDDisplayOnCursorOff()     XLCDCommand(0x0C)
#define XLCDDisplayOff()             XLCDCommand(0x08)
#define XLCDCursorMoveLeft()         XLCDCommand(0x10)
#define XLCDCursorMoveRight()        XLCDCommand(0x14)
#define XLCDDisplayMoveLeft()        XLCDCommand(0x18)
#define XLCDDisplayMoveRight()       XLCDCommand(0x1C)





/*---------------------------------

     CADENAS DE CARÁCTERES EN RAM

-----------------------------------*/

ram const char E[]="E      41.2 Hz";
ram const char A[]="A      55 Hz";
ram const char D[]="D      73.4 Hz";
ram const char G[]="G      98 Hz";
ram const char frecu[]="freq:  ";



/*----------------------------

     DEFINICIÓN DE VARIABLES

-----------------------------*/

unsigned int freq=0;
unsigned int freq0=0;
unsigned char aaa[100];
float f1,f2;
```

```
float freq3=0;
int freq1=0;
int freq2=0;
int sleep=0;
int pols=0;
int j=0;
int buffer_long=4;
int i1,i2;
int ya=0;



/*----------------------------

     DEFINICIÓN DE LAS RSI

------------------------------*/

void int1();
void timer3();

// interrupción de alta prioridad

#pragma code high_vector=0x08
void high_vector (void){
_asm goto int1 _endasm
}

// interrupción de baja prioridad

#pragma code low_vector=0x18
void low_vector (void){
_asm goto timer3 _endasm
}
#pragma code



/*----------------------------

     RSI

------------------------------*/


// interrupcion timer 3

#pragma interruptlow timer3
void timer3(){

T3CONbits.TMR3ON=0;   // se apaga el timer3
ya++;

     if (ya==4){
       T3CONbits.TMR3ON=0;   // se eapaga el timer3
       TMR3H=0x00;
       TMR3L=0x00;
       ya=0;
       XLCDClear();          // borrado de la pantalla LCD
       PORTC=0x00;
       }

PIR2bits.TMR3IF=0;                  //  flag bajado

}


// interrupción INT1 (señal del bajo)

#pragma interrupt int1
void int1 (){

T3CONbits.TMR3ON=0;          // se desconecta el timer3
PIR2bits.TMR3IF=0;
ya=0;
```

```
pols++;
j=0;

     if (pols==1){
       T1CONbits.TMR1ON=1;          // timer 1 encendido
       vuelta=1;
       }

     if (pols==2){
       pols=0;
       T1CONbits.TMR1ON=0;          // timer 1 apagado
       freq1=TMR1L;
       freq2=TMR1H;

       /*----------------------
       cálculo de la frecuencia
       ----------------------*/

       freq0=freq2<<8|freq1;
       freq3=freq0;
       freq3=1/freq3;
       freq3=(freq3*1000000);
       freq=(int)(freq3*100);

       f1=floor(freq3);
       f2=freq3 - f1;
       i1 = (int)f1;
       i2 = (int)100*f2;            // 2 decimales
       sprintf(aaa,"%i.%02i", i1,i2);
       XLCDReturnHome();


       /*------------------------
       determinación de la cuerda
       ------------------------*/

       if(freq>11000){
              PORTC=0x10;
              XLCDPutRamString(G);
              XLCDL2home();
              XLCDPutRamString(frecu);
              XLCDPutRamString(aaa);
              XLCDPut(' ');
              XLCDPut('H');
              XLCDPut('z');
              XLCDPut(' ');
              }


       // CUERDA SOL


       if((freq>=8600)&&(freq<=11000)){

              if ((freq>=9700)&&(freq<=9900)){     // cuerda afinada
                     PORTC=0x08;
                     }

              else if(freq<9700){           // frecuencia cuerda < frecuencia correcta
                     PORTC=0x04;
                     }

              else if (freq>9900){
                     PORTC=0x10;           // frecuencia cuerda > frecuencia correcta
                     }

       XLCDPutRamString(G);
       XLCDL2home();
       XLCDPutRamString(frecu);
       XLCDPutRamString(aaa);
       XLCDPut(' ');                       // mostrar valores por LCD
       XLCDPut('H');
       XLCDPut('z');
       XLCDPut(' ');
       }
```

```
// CUERDA RE


if((freq>=6300)&&(freq<8600)){

        if ((freq>=7240)&&(freq<=7440)){
                PORTC=0x08;
                }

        else if(freq<7440){
                PORTC=0x04;
                }

        else if (freq>7240){
                PORTC=0x10;
                }

XLCDPutRamString(D);
XLCDL2home();
XLCDPutRamString(frecu);
XLCDPutRamString(aaa);
XLCDPut(' ');
XLCDPut('H');
XLCDPut('z');
XLCDPut(' ');
}



//CUERDA LA


if((freq>=4800)&&(freq<6300)){

        if((freq>=5400)&&(freq<=5600)){
                PORTC=0x08;
                }

        else if(freq<5600){
                PORTC=0x04;
                }

        else if (freq>5400){
                PORTC=0x10;
                }

XLCDPutRamString(A);
XLCDL2home();
XLCDPutRamString(frecu);
XLCDPutRamString(aaa);
XLCDPut(' ');
XLCDPut('H');
XLCDPut('z');
XLCDPut(' ');
}

//CUERDA MI


if((freq>=2750)&&(freq<4800)){

        if((freq>=4020)&&(freq<=4220)){
                PORTC=0x08;
                }

        if(freq<4020){
                PORTC=0x04;
                }

        if(freq>4220){
                PORTC=0x10;
                }

XLCDPutRamString(E);
XLCDL2home();
XLCDPutRamString(frecu);
```

```
        XLCDPutRamString(aaa);
        XLCDPut(' ');
        XLCDPut('H');
        XLCDPut('z');
        XLCDPut(' ');
        }


     TMR1H=0x00;             // reinicio del contador del timer 1
     TMR1L=0x00;
     }

TMR3H=0x00;                  // reinicio del contador del timer 3
TMR3L=0x00;
INTCON3bits.INT1IF=0; //flag = 0
T3CONbits.TMR3ON=1;          //encendido timer 3
}
/*----------------------------

     INICIALIZACIONES

-----------------------------*/

void ini_TIMER()
{


// TIMER 1

T1CON=0x00;
     /*

     R16=0            lectura/escritura en 2 tiradas de 8 bits

     T1RUN=0          device clock de otra fuente

     T1CKPS1=0

     T1CKPS0=0 pre-escaler 1:1

     T1OSCEN=0 oscilador apagado

     T1SYNC=0  es indiferente

     TMR1CS=0  reloj interno (Fosc/4)

     TMR1ON=0  timer parado

     */

TMR1H=0x00;
TMR1L=0x00;


// TIMER 3

T3CON=0x30;
     /*

     R16=0            lectura/escritura en 2 tiradas de 8 bits

     T3CCP2=0  indiferente

     T3CKPS1=1

     T3CKPS0=1 pre-escaler 1:8

     T3CCP1=0  indiferente

     !T3SYNC=0 indiferente si TMR3CS=0

     TMR3CS=0  reloj interno (Fosc/4)

     TMR3ON=0  timer parado

     */
```

```
TMR3H=0x00;
TMR3L=0x00;
PIE2bits.TMR3IE=1;                          //  TMR3 interrupción activada
PIR2bits.TMR3IF=0;                          //  FLAG de la interrupción bajado
IPR2bits.TMR3IP=0;                          //  Prioridad=BAJA

}



void ini_INT1(){

PORTA=0x00;
ADCON0=0x00;                                //  deshabilitado el convertidor A/D
ADCON1=0xFF;
TRISB=0x00;
PORTB=0x00;
TRISC=0x00;
PORTC=0x00;
RCONbits.IPEN=1;                    //  Prioridades habilitadas
INTCON3bits.INT1IE=1;               //  INT1 externa activada
INTCON3bits.INT1IF=0;               //  FLAG de la INT1 bajado
INTCON3bits.INT1IP=1;               //  Priorodad=ALTA

XLCDInit();                         // Inicialización de la LCD


INTCONbits.GIEH=1;
INTCONbits.GIEL=1;                  //  Global Interrupt habilitado

}




/*----------------------------

     LCD DELAYS

-----------------------------*/

void XLCDDelay15ms (){

int i;
    for(i=0;i<10000;i++)
    {
    Nop();
    }
}



void XLCDDelay4ms (){

int i;
    for(i=0;i<2500;i++)
    {
    Nop();
    }
}



void XLCD_Delay500ns(){

Nop();
Nop();
Nop();
```

```
}



/*----------------------------

     MAIN

------------------------------*/


void main (){

/*----------------------------
CONFIGURACIÓN OSCILADOR INTERNO
------------------------------*/

OSCCONbits.IRCF2=1;
OSCCONbits.IRCF1=1;
OSCCONbits.IRCF0=0;          // oscilador interno a 4MHz

ini_TIMER();
ini_INT1();
XLCDPut('o');
XLCDPut('k');


     while(1){
       OSCCONbits.IDLEN=1;   // microcontrolador en modo RC_IDLE
       Sleep();

     }

}
```