Diseño e implementación de un dispositivo musical para personas con discapacidad motriz

Adrián Cabello Corpas

Resumen— La musicoterapia es uno de los métodos más eficaces en cuanto a expresión se refiere, es importante que este medio de expresión sea asequible para personas con discapacidad motriz. Se presenta el diseño de un sistema musical digital, gobernado por un microcontrolador y adaptado a personas que sufren discapacidad motriz. Se ha diseñado el sistema físico para interaccionar con el usuario, altavoz y pulsadores ergonómicamente adaptados a la movilidad de la persona, que incorporan luces. El hardware y firmware del sistema se ha implementado basado en el microcontrolador PIC 16F84A. Desde el punto de vista funcional se ha desarrollado un programa que permite dos modos de funcionamiento, uno de interacción libre en el que el usuario compone música presionando los pulsadores, y otro más guiado en el que se le propone que relacione un sonido determinado con un pulsador, primero se emite sonido y mediante el led se le indica el pulsador que debe utilizar para parar dicho sonido, con esto se pretende fomentar una motricidad más precisa para interaccionar con el sistema.

Palabras clave—Microcontrolador, PIC 16f84a, antirebote, ensamblador, pulsador, discapacidad motriz, prototipo, diseño.

Abstract— Music therapy is one of the most effective methods in therms of expression. It is important that this medium of expression is affordable for people with motor disabilities. The design of a digital music system is presented, governed by a microcontroller and adapted to people who suffer from motor disabilities. The physical system has been designed to interact with the user, speaker and ergonomically pushbuttons are adapted to the mobility of the person, with incorporated lights. The hardware and firmware system have been implemented based on the 16F84A PIC microcontroller. From the functional point of view a program that allows two operating modes has been developed, one with free interaction in which the user composes music pressing the buttons, and a guided one which the user is proposed to relate a certain sound with a pushbutton, first a sound is emitted and the LED indicates the button that must be used to stop that sound. This is intended to encourage a more precise motor skill to interact with the system.

Index Terms— Microcontroller, PIC 16F84A, anti-kickback, assembler, button, motor disability, prototype, design.

1 Introducción

VIVIMOS en un mundo que se ha construido para el tipo de persona más común, y a veces nos olvidamos de adaptarlo a aquellas personas que no cumplen los estándares marcados por la sociedad.

La musicoterapia es uno de los métodos más eficaces en cuanto a expresión se refiere, pero la inadaptabilidad de los instrumentos provoca que las personas con una discapacidad motriz no puedan interactuar con ellos, reduciendo así el efecto de la musicoterapia [11].

Las adaptaciones de instrumentos que podemos encontrar hoy en día son mecánicas y estas están adaptadas a las necesidades de una persona en concreto y no a las de un colectivo, por lo que las soluciones hoy en día son caras y escasas [10].

Se propone diseñar un dispositivo en el que el usuario puede interactuar con un sistema que dispone de una serie de pulsadores, que permiten detectar la interacción con el sistema mediante presión y actuar emitiendo un sonido e iluminando un led para que haya una doble realimentación. Se propone monitorizar la interacción con el dispositivo, guardarando información de la interacción del usuario con el dispositivo, permitiendo monitorizar la actividad para posteriormente evaluar la evolución de la persona. El sistema dispone de diferentes formas de funcionamiento como pueden ser: un modo que se ha denominado libre, en el que el usuario puede interactuar a su parecer con el sistema, accionando los pulsadores sin un orden o tiempo entre acción preestablecido de modo que lo que podremos monitorizar es: el tiempo de uso del sistema, el patrón que sigue el usuario al interactuar con el sistema, y la variación de tiempo que hay entre interacciones, por ejemplo: un aumento del lapso de tiempo entre cada interacción puede denotar cansancio. El modo denominado guiado, en cambio, se pretende dirigir al usuario, primero se emite el sonido y a su vez se ilumina un led, esperan en este estado hasta que se acciona el pulsador pertinente, en este caso podemos monitorizar una evolución respecto a resultados anteriores o resultados esperados que se puedan llegar a definir.

En este proyecto se tiene muy en cuenta la adaptación a la discapacidad del usuario, la usabilidad, también se ha considerado el coste y el diseño ya que tiene una finali-

- *E-mail de contacto:acabello93@gmail.com*
- *Mención realizada: Ingeniería de Computadores.*
- *Trabajo tutorizado por: Dolores Isabel Rexachs del Rosario (CAOS)*
- *Curso 2015/16*

dad muy específica, tanto a nivel de ocio como a nivel de aprendizaje e interesa que sea económicamente accesible. Para ello se ha colaborado con el centro ASDI de Sant Cugat.

El objetivo principal de proyecto ha sido diseñar e implementar un instrumento musical adaptado a personas con discapacidad motriz que les facilitase la expresión musical, que permita utilizar la música como un elemento más de aprendizaje.
También se ha facilitado la interacción al usuario: diseñando un sistema ergonómico [8] (adaptado al entorno del usuario), con sensores de entrada adecuados, robustos [7] y que proporcionan una realimentación al usuario.
El sistema ha de puede ser usado y/o configurado por una persona sin necesidad de formación alguna.
Funcionalmente cuando el sistema recibe un estímulo, el sistema reacciona emitiendo un sonido y encendiendo el led correspondiente, el sistema tiene tanto una función de uso libre, como un patrón a seguir, indicado mediante los leds y sonidos diferentes.
De forma no funcional el sistema es modular tiene pulsadores (inputs) con leds (outputs) integrados para que la pulsación sea evidente a simple vista, los pulsadores mediante los cuales el sistema recoge la información, tienen formas o colores específicos con la finalidad de aceptación del dispositivo en sí, los "inputs" pueden colocarse en distintas posiciones para una mayor adaptabilidad.

El documento se ha organizado de la siguiente manera: se presentan unos conceptos previos y un estado actual que ayudará a contextualizar el tema. Después podemos ver el diseño del sistema dividido en tres secciones, el del software, el del hardware y el diseño físico del dispositivo seguidos de un apartado en el que se dan detalles de la implementación especificando las herramientas utilizadas en cada caso. A continuación se presentan los resultados obtenidos, el apartado de conclusiones presenta un resumen con los conceptos aprendidos y se proponen unas líneas de futuro.

2 CONCEPTOS RELACIONADOS

La finalidad de este apartado es aportar una serie de conceptos previos para lograr una mejor comprensión del trabajo realizado.

2.1 La discapacidad motriz

La discapacidad motriz es una condición de vida que afecta el control y movimiento del cuerpo, generando alteraciones en el desplazamiento, equilibrio, manipulación, habla y respiración de las personas que la padecen, limitando su desarrollo personal y social [10].
Esta discapacidad se presenta cuando existen alteraciones en los músculos, huesos, articulaciones o medula espinal, así como por alguna afectación del cerebro en el área motriz impactando en la movilidad de la persona [13].
Es importante que entendamos que la discapacidad motriz no es una enfermedad o una discapacidad que afecte el rendimiento intelectual de la persona per se aunque en la mayoría de los casos, esta viene acompañada de otras enfermedades que sí pueden afectar al funcionamiento cerebral. Las dificultades que presenta una persona con Discapacidad Motriz pueden ser muy variadas dependiendo del momento de aparición, los grupos musculares afectados (topografía), el origen (cerebral, espinal o muscular), el grado de afectación (ligera, moderada o grave) y el momento de aparición (antes, después o durante el nacimiento). La Discapacidad Motriz como hasta ahora hemos visto afecta las posibilidades de movimiento y desplazamiento por lo que la accesibilidad y habilitación de medios representan las principales necesidades a las que se enfrentan las personas que se ven afectadas por esta discapacidad, necesitan distintos apoyos y recursos que faciliten su autonomía y favorezcan su comunicación, participación y logro educativo.

2.2 Los microcontroladores

Un microcontrolador es una microcomputadora completa encapsulada en un circuito integrado.
El carácter programable de los microcontroladores simplifica el diseño de circuitos electrónicos. Permiten modularidad y flexibilidad, ya que un mismo circuito se puede utilizar para que realice diferentes funciones con solo cambiar el programa del microcontrolador. Las aplicaciones de los microcontroladores son vastas, se puede decir que solo están limitadas por la imaginación del usuario. Es común encontrar microcontroladores en campos como la robótica y el automatismo, en la industria del entretenimiento, en las telecomunicaciones, en la instrumentación y en el hogar. Para este proyecto el microcontrolador seleccionado es el microcontrolador PIC 16F84A.

2.3 El lenguaje de programación

El PIC16F84A pertenece a la gama media y es de tipo RISC; esto quiere decir que tiene un juego de instrucciones reducido, en concreto de 35 instrucciones que son la base de funcionamiento del PIC [6].
Las instrucciones fundamentalmente se dividen en tres tipos. Esta división viene dada por el tipo de datos con los que trabajan:
Hay que tener en cuenta lo siguiente:

- Instrucciones orientadas a los registros o bytes (byte-oriented operations).
- Instrucciones orientadas a los bits (bit-oriented operations).
- Operaciones con literales y de control (literal and control operations).
- Para comprobar si dos valores son iguales, movemos un valor a un registro y otro en el acumulador, realizamos una resta bit a bit y entonces tenemos que mirar el valor del bit Z del registro de estado, el cual si tiene el valor 1, significa que la resta ha dado 0 por lo que los dos valores son iguales.
- Todos los saltos condicionales se crean a partir de una operación y una instrucción que salta a una etiqueta si un bit concreto es '0' o '1'. Esta operación solo puede ver el valor de un bit, por lo que si queremos compa-

rar el valor del registro con el del acumulador, por ejemplo para ver si hay más de un pulsador presionado a la vez, lo que debemos hacer es realizar una resta, y el bit que ha de ser comprobado es el bit ZERO del registro de estado, si la resta ha resultado cero y por lo tanto ambos valores son iguales, este bit se pondrá a '1'. De esta forma ya podemos crear todas las comparaciones y llamadas a subrutinas necesarias.

- Al modificar un registro de E/S con una operación sobre él mismo (por ejemplo MOVF PORTB,1), el valor utilizado es el que se halle presente en las patillas del PORTB. Por ejemplo, si el biestable tiene un "1" para una patilla configurada como entrada y se pone a nivel bajo desde el exterior, el dato se volverá a escribir como "0". Si se ejecuta esta instrucción sobre el TMR0 y d=1, se borrará el conteo del preescaler, si está asignado al TMR0, pero no se borrará la preescaler asignada en OPTION_REG, que controla Timer0. Si se modifica el Contador de Programa PC o una condición de prueba es verdadera, la instrucción requiere dos ciclos máquina. El segundo ciclo se ejecuta como un NOP. La operación, la sintaxis y el comportamiento del registro STATUS, son imprescindibles para comprender su funcionamiento.

El código fuente sigue una forma muy concreta, primero van las etiquetas, a continuación la instrucción separada por una tabulación de la etiqueta, y por último precedido del carácter ";" están los comentarios.
Debemos empezar especificando lo siguiente:

```
List    p=16F84A        ;Tipo de procesador
include "P16F84A.INC"   ;Definiciones de registros internos
```

A continuación las declaraciones pertinentes indicando mediante la instrucción equ la posición de memoria en la que se alojan.
Tras esto, indicaremos lo siguiente:

```
org     0x00            ;Vector de Reset
goto    Inicio
org     0x05            ;Salva vector de interrupción
```

Donde se muestra que las 5 primeras direcciones están ocupadas.
Esto es demasiado justo ya que aunque las primeras 5 direcciones estén ocupadas, es mejor empezar el programa desde la dirección 200h y no la 5h, cambiando de página, ya que a veces indirectamente, PIC usa esa primera página para guardar sus valores, los cuales no queremos machacar en ningún momento.

2.4 Uso de la memoria EEPROM

El acceso a la EEPROM en el 16F84a es a través de direccionamiento indirecto y se hace mediante 4 registros. EEDAT guarda el dato que habrá de escribirse en la EEPROM, mientras que EEADR es un apuntador hacia la localidad deseada. EECON1 que contiene bits como RD o WD que se ponen a '1' si se va a leer o escribir respectivamente y EECON2 que se emplea como mecanismo de seguridad para la lectura y escritura.

2.5 Dispositivos de entrada: pulsadores

El pulsador va a permitir al usuario interactuar con el sistema, debe estar adaptado en tamaño facilidad de uso y robustez. Otro aspecto importante se da al trabajar con pulsadores y se llama rebote, este es debido a que un pulsador está compuesto de dos partes de metal que entran en contacto (choca una con la otra) al accionarlo. Este choque genera unos rebotes que suceden tan rápido que son imperceptibles para nosotros, Sin embargo, no lo son para el PIC que trabaja a velocidades de 4MHz. Por lo tanto esto puede solucionarse de dos formas, mediante hardware, que es lo más habitual, empleando resistencias y condensadores o en nuestro caso mediante software, guardando cual es el pulsador pulsado y durante un lapso de tiempo pequeño comprobamos si es el mismo pulsador el que está pulsado y por lo tanto no debemos reaccionar dos veces sino una [9].

2.6 dispositivos de salida: altavoz y led.

Como dispositivo de salida para el instrumento musical el altavoz es el elemento clave, se ha querido también incorporar una retroalimentación visual que permita reforzar la interacción.
Para generar sonido digital se utilizan trenes de pulsos a una frecuencia concreta, es decir el periodo que está compuesto por la ½ del tiempo por una señal a '1' y la otra ½ por una señal a '0', repitiendo este periodo un número preciso de veces por segundo, nos marca la frecuencia de una nota musical.

Una forma de crear estos trenes de pulsos es mediante contadores, estos determinan el tiempo que dura ½ periodo, únicamente hay que enviar un '1' a la salida durante este tiempo que hemos calculado, después un '0' obteniendo así un periodo completo (*ver figura 1*)

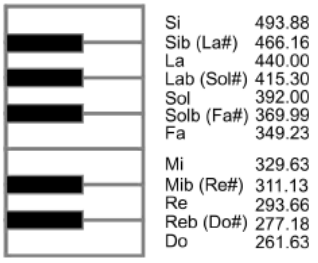

Figura 1: Conjunto de frecuencias equivalentes a las notas de una octava.

Un LED estándar soporta una intensidad de hasta 20mA, se añade una resistencia en serie, para obligar a que la corriente pase por el LED y por la resistencia, con lo que se limita, es decir, se reduce la corriente que circula por el LED [9].
Si elegimos un valor de resistencia un poco más alto del ideal, la intensidad será un poco menor por lo que:
V = I x R → Vfuente – Vled = I x R → Vfuente – Vled = 17mA x R → R = (Vfuente – Vled) / 0,017A

3 Estado del arte

Como ya se ha mencionado, debido a los problemas en el sistema locomotor, es necesaria una adaptación del entorno para satisfacer las necesidades de estas personas, las cuales a veces se obvian. Podemos ver las más ingeniosas cuando nos centramos en el sistema educativo, donde una mesa tiene un corte en forma de media luna o las patas de esta ligeramente más separadas que las de un pupitre convencional, todo ello para favorecer la integración de una silla de ruedas. También existen pupitres con un reborde que sobrepasa el perfil de la mesa, evitando así que las cosas que hay sobre esta caigan al suelo, ya que para una persona con discapacidad motriz sería un problema tener que recoger lo que se ha caído.

Existen también adaptaciones para utensilios como son lápices, tijeras y demás, a estos se les añade un segundo objeto para acomodarlos a las manos [8].

En cuanto a la rama de la informática, podemos ver adaptaciones por ejemplo en los ratones, donde se diferencian dos botones grandes y una bola en el centro con la finalidad de mover esta con la muñeca.

La música en cambio no ha logrado una adaptación tan grande, debemos destacar la importancia de la musicoterapia, tanto como una forma de aprendizaje como una manera de expresarse. El sistema más parecido al que se plantea es uno denominado "La flauta mágica", integrada por una flauta y una pantalla, permite al músico a través de la flauta cambiar el sonido o cambiar la configuración del usuario para satisfacer sus necesidades sin tener que usar las manos.

El sistema cuenta con 128 instrumentos integrados. Esto permite contar con una amplia gama de sonidos para reproducir: flauta, trompeta, saxofón y guitarras. La flauta, además, se puede utilizar con un cable MIDI para controlar fuentes de sonido electrónicos tales como un teclado, sintetizador, módulo de sonido o un equipo de música. La flauta mágica es útil también para realizar ejercicios de respiración, la terapia respiratoria será más amena de esta forma [13].

4 Diseño

El primer paso es escoger una metodología y seleccionar unos elementos previamente para usar a lo largo del proyecto, antes de explicar los diseños realizados.

4.1 Metodología

La metodología que se decidió llevar a cabo fue el desarrollo en cascada. Las razones por las que se escogió esta metodología fueron: por su poca cantidad de etapas, las cuales están bien diferenciadas, porque es una metodología ágil y es sencilla de utilizar. La metodología fue adaptada para poder realizar en base a unos objetivos definidos, un diseño hardware y otro software en paralelo, con sus pertinentes pruebas unitarias antes de realizar una integración, pudiendo volver a la etapa de diseño si no cumplían los requisitos necesarios.

Ha habido una fase de aprendizaje, pero no se ha considerado como una etapa previa en sí, ya que ha sido constante durante todo el proyecto y en cada diseño y en cada prueba, se han aprendido los conocimientos necesarios para llevarlos a cabo (*ver figura 2*).

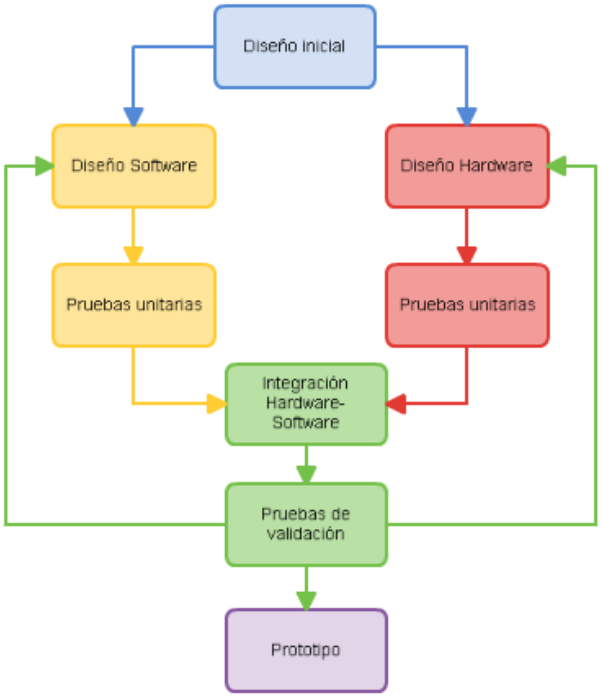

Figura 2: Metodología, utilizada a lo largo del proyecto.

4.2 Elecciones previas

La primera elección ha sido la familia de microcontroladores que se pueden utilizar y en como compararlas entre ellas. En este caso han sido cuatro: Arduino, PIC, freeRTOS y ARM Cortex (*ver tabla 1*). Finalmente se ha escogido PIC debido a la facilidad de programación a bajo nivel (ensamblador), su adaptabilidad a innumerables periféricos, su coste económico y su extensa documentación.

Por otro lado, existen una serie de herramientas de bajo costo por parte de terceros (empresas, profesionales y aficionados), como son programadores, software, etc., que facilitan el uso y programación de estos dispositivos.

TABLA 1
CRITERIOS DE COMPARACIÓN Y SELECCIÓN DE LA FAMILIA DE MICROCONTROLADORES.

	Arduino	PIC	Free-RTOS	ARM Cortex
Coste		X		
Documentación	X	X		
Programación bajo nivel		X		X
Programación alto nivel	X		X	
Familiarizado en clase			X	X

Una vez hemos escogido una familia de microprocesadores, y el sistema de desarrollo escogido para esta familia es: Micro PIC Trainer, es un sistema de desarrollo muy completo y económico de la década de los 90. Permite

diseñar tanto hardware como software que rodea a las aplicaciones basada en los microcontroladores PIC.
El PIC utilizado es el modelo 16F84A, en el cual La memoria de programa contiene 1K words, lo que se traduce en 1024 instrucciones.
La memoria de datos (RAM) contiene 68 bytes.
La memoria EEPROM es de 64 bytes.
También hay 13 pines E/S que son totalmente programables.

Lo siguiente que se decidió, una vez definido el problema, fue el número y tipo de periféricos a utilizar para interactuar con el usuario.
El primer periférico que se pretendía usar para recoger información de los usuarios fue un sonar, mediante un movimiento en el mismo eje en el que el sonar usa para definir una distancia, podemos diferenciar distintos estados, que se traducen en distancias y a su vez lo podemos interpretar como diferentes notas. Esta no es una solución ya que debido a la dificultad e imprecisión del movimiento con el que cuentan los usuarios, es un inconveniente porque dificulta su uso y eso contradice nuestro objetivo principal. La opción por la que se optó fue por usar pulsadores adaptados a las características de los usuarios, otra ventaja que nos aporta el adaptar un pulsador, es que podemos integrar otros componentes como la luz y variar su morfología como veremos más adelante.
Para realimentar al usuario de la información generada por el dispositivo, se utiliza información lumínica en forma de led, uno incorporado en cada pulsador. También contamos con el recorrido que tiene el propio pulsador, con el que podemos identificar si este está siendo accionado o no. Por último y no por ello menos importante, la señal que recibe el usuario es acústica, y esta es reproducida mediante un único altavoz para todo el dispositivo.
El programa que finalmente se ha escogido para realizar el diseño es PROTEUS 8.4, esta decisión se debe a que no solo puede diseñarse el circuito, sino que este puede ser simulado (incluyendo el microcontrolador PIC 16F84A), incluso programado con un código fuente y depurado al máximo detalle.
Para ensamblar el código fuente se ha usado MPASM, recomendado por Microchip y para programar el PIC el programa escogido es WinPIC800.

Diseño del esquemático

Esta herramienta tiene incorporado bloques en los que se ha implementado el funcionamiento del microcontrolador, también tiene los símbolos que identifican a los pulsadores, interruptores y leds. Con esta herramienta es con la que hemos dibujado el circuito que hemos visto anteriormente. A parte también nos proporciona la opción de adjuntar un código fuente y hacer una simulación lo cual es muy útil.

Código fuente

Como ya se ha comentado, existe la opción de adjuntar un código fuente y con las librerías propias del microcontrolador donde se identifica cada pin, registro o cualquier cosa que pueda utilizarse y se le asigna un nombre que lo identifica.
Podemos compilar sin más y este nos informará de errores de sintaxis como un compilador cualquiera.

Depuración

Esta es la etapa más interesante y la razón por la que se eligió este programa, tiene una gran cantidad de funciones para depurar, lo que permite detectar una gran cantidad de errores en la programación (*ver figura 3*).

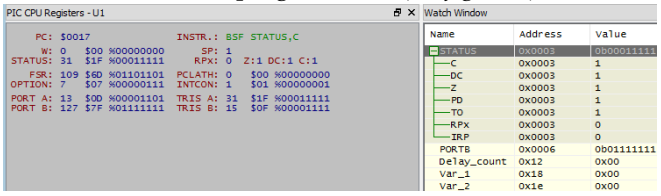

Figura 3: Sección de depuración de la herramienta de desarrollo Proteus 8.4.

En la parte de abajo de la izquierda podemos ver los registros de la CPU del PIC, aquí debemos destacar el registro "W", el cual identifica al acumulador, y este es con el que siempre se realizan las operaciones de cálculo, o hace de auxiliar al mover valores entre registros. También es útil ver la información que se mueve por el PortA o el PortB, estos son los que muestran directamente la información de los pulsadores activos y los leds encendidos.
En Tris A o TrisB podemos ver la configuración de las E/S, nunca está de más revisarlos ya que es algo que se presupone que está bien configurado y puede ser una causa de un problema.
En la parte de abajo de la derecha de la *figura 3*, podemos ver la Watch Window, en esta tenemos registros más concretos, los cuales hemos conseguido añadir mediante la dirección en memoria que conocemos, sea mediante el Datasheet del PIC como es el registro de estado donde por ejemplo nos interesa el bit "Z" el cual una de sus funciones es ponerse a '1' cuando el resultado de una resta es 0, o la variable "Delay_count" la cual hemos decidido ubicar en una posición concreta y por lo tanto sabemos su dirección.

Programa ensamblador

El programa utilizado es MPASM que nos proporciona Arizona Microchip INC, este convierte el código fuente en código ejecutable de una forma muy sencilla, en la primera pasada se chequea la correcta sintaxis (etiquetas duplicadas), asigna valores a los símbolos y borra los comentarios. En la siguiente pasada convierte todas las instrucciones en sus códigos máquina numéricos. A continuación se puede ver una captura de pantalla (*ver figura 4*) del MPASM y se explicarán sus campos.

- Source file: donde escogemos el archivo .ASM del código fuente.
- Processor type: en este campo se escoge el PIC usado, en este caso 16F84A.

- Error file: se crea un fichero en con mismo nombre que el código fuente con posibles errores encontrados durante la compilación.
- Cross reference file: por defecto no se crea pero este contiene un listado con las etiquetas encontradas.
- Listing file: este si se crea por defecto y contiene un listado completo de la compilación.
- Hex dump type: este es el fichero ejecutable que se usará para grabar en el PIC.
- Assamble to object file: por defecto está desactivada y ésta únicamente crea un objeto el cual puede ser enlazado con otros módulos para crear un nuevo objeto mayor.

Finalmente presionamos la tecla F10 para ensamblar el código fuente y MPASM muestra una pantalla de información si ha habido algún error o cualquier tipo de aviso.

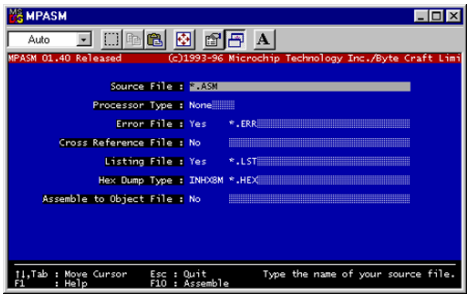

Figura 4: Menú de opciones del programa ensamblador MPASM.

Programa grabador

El programa escogido es WinPIC800, fue el primero en ser probado, tiene la opción de programar el PIC 16F84A y permite la configuración necesaria y se comenta a continuación (*ver figura 5*).

Este es un programa muy sencillo donde cargamos el archivo .HEX y podemos grabar el PIC, leerlo, verificar su estado o borrarlo. Si accedemos a la ventana llamada "Config" podemos escoger entre uno de los 4 tipos de osciladores (cada uno funciona mejor para una frecuencia concreta) y el Watchdog.

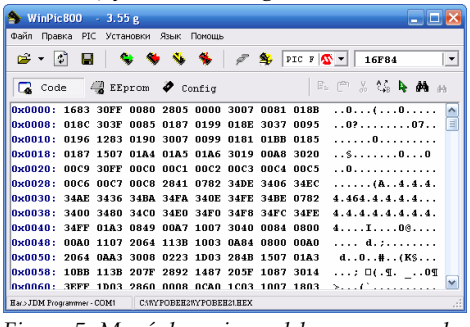

Figura 5: Menú de opciones del programa grabador WinPIC800.

4.3 Diseño hardware

Antes de diseñar un circuito debíamos de contemplar todas aquellas cosas que formarán parte de este, y lo más simple para mostrarlo es un diagrama de bloques sencillo, donde se separan los periféricos necesarios para los inputs y los outputs (*ver figura 6*).

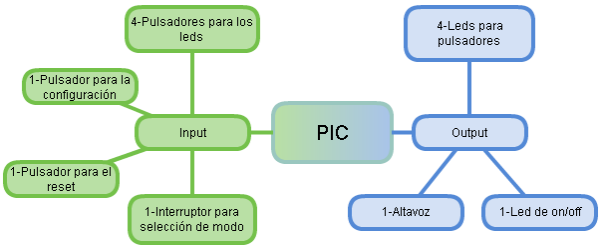

Figura 6: Diagrama de bloques que muestra los periféricos conectados al microcontrolador.

4.4 Diseño software

El primer paso de todos ha sido definir qué es lo que debía hacer y controlar nuestro programa (*ver figura 7*).

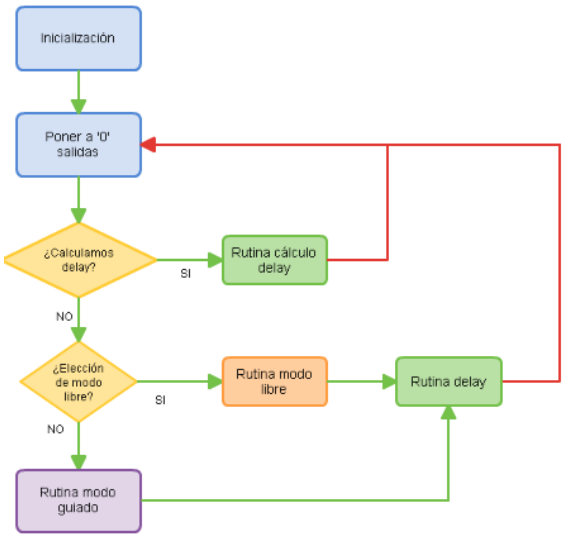

Figura 7: Versión reducida del diagrama de secuencia que explica la lógica del programa.

- Inicialización del sistema.
- Configurar el sistema: el sistema autónomo, pero programable, permite incorporar más de una actividad a realizar por el usuario.
- Identifica el pulsador que acciona el usuario. Se ha tenido en cuenta que los usuarios pueden presionar más de un pulsador por accidente o repetir una pulsación varias veces de forma incontrolable cuando realmente se quería presionar una sola. Por esta razón se ha utilizado la técnica de espera activa leyendo los puertos a los que están conectados los pulsadores siguiente un orden prioritario, la cual cosa ayuda a conseguir que el sistema no detecte que hay dos pulsadores accionados y no reproduzca dos sonidos mezclados.
- Encender el led para dar una realimentación al usuario y emitir el sonido correspondiente.
- También se ha añadido un tiempo de espera al que llamaremos delay, este se usa para marcar el tiempo que pasa desde que hay desde que se ha detectado que hay un pulsador accionado y pasar a comprobarse el estado del siguiente, solucionando así el problema

de las múltiples pulsaciones.
- El tiempo de delay no es una constante programada, sino que mediante el uso de un pulsador puede decidirse de la siguiente forma: si por ejemplo accionas el pulsador de delay durante 5 segundos, el tiempo que pasa entre que se ha detectado que hay un pulsador accionado y se comprueba el estado del siguiente pulsador son 5 segundos.
- Para facilitar la programación de las diferentes actividades o modos de funcionamiento se han diseñado e implementado una serie de módulos que servirán de base.

4.5 Diseño del sistema

El primer paso que fue pensar en cómo el usuario debía comunicarse con el sistema y viceversa. Como ya hemos visto a lo largo del trabajo, la entrada se hace a través de pulsadores y la comunicación de salida no solo es acústica, sino que también es visual, y eso se consigue mediante el uso de luces led. Esa es la primera dificultad añadida, el encontrar un compuesto de pulsador con luz incorporada, además debemos tener en cuenta que el pulsador ha de tener una forma y una resistencia adecuada para el uso que hemos definido en anteriormente, y eso nos lleva a la siguiente conclusión: debemos diseñar y construir unos pulsadores adaptados.
Para esta construcción se ha decidido usar un pulsador pequeño al cual se adapta a una medida concreta empleado metacrilato para crear tanto la base del pulsador que contiene un led, como la superficie con la cual se interactúa, esta está pegada a un pequeño pulsador(*ver figura 8*).
El pulsador se sometió a pruebas de esfuerzo, resistencia y uso por parte de usuarios. No superó esta última prueba ya que la forma en la que se une el pulsador y la superficie que lo amplia, impide que este sea accionado desde una zona diferente a la central, ya que se crea un rozamiento total.

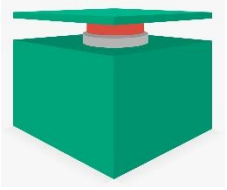

Figura 8: primer diseño de pulsador descartado

Dichos problemas se traducen en volver empezar esta parte de la etapa de diseño con una nueva idea de pulsador. Este tiene una forma bastante parecida, la base incluso se aprovechó, y la superficie de adaptación en vez de estar pegada al pulsador real, lo está mediante una bisagra a la base, lo que reduce el esfuerzo que debe ejercerse sobre ella para realizar un uso efectivo del pulsador. Además en este último diseño se añade pintura opaca en cinco de las seis caras para focalizar la luz que emite el led, este led es distinto al que se emplea para señalizar que el sistema está encendido, para el pulsador se ha empleado un led lumínico, el cual focaliza y emite una luz más intensa y de un color concreto (verde, ámbar, rojo y azul) dependiendo del pulsador [7]. Una espuma entre la base y la superficie de presión para repartir la fuerza que se ejerce sobre el pulsador. En cuanto a tamaño, tiene la altura de un ratón de ordenador, aprovechando su forma ergonómica para que pueda ser adaptado a una mano (*ver figura 9*).
Este último diseño fue sometido a las mismas pruebas que el anterior, superándolas con éxito.

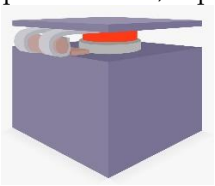

Figura 9: segundo diseño de pulsador aceptado

5 Implementación Software

Para la implementación software, se ha comenzado implementando diferentes componentes que tienen una doble utilidad, estas pruebas tienen la finalidad de familiarizarte tanto con el lenguaje como con los entornos de desarrollo de forma modular.
Todas estas pruebas han sido simuladas mediante la herramienta de Proteus 8.4 ya vista.

5.1 Manejo de los ports E/S

El objetivo de esta prueba es la familiarización con el Manejo de ports de E/S que en este caso se configura PortA como entrada, en el que está conectados 5 interruptores, se leen sus estados y se muestran por PortB el cual anteriormente se han configurado como salida y en el cual están conectados leds.

5.2 Uso de los ports y los timers

El objetivo de este 2º programa es añadir la utilización de timers, uno de los aspectos importante es el control de rebotes, se realiza este control por software mediante la utilización de.
En este caso usamos un pulsador perteneciente a una ampliación de Micro PIC Trainer, llamado PLUS y mostramos el código del pulsador mediante los leds (4 leds para la fila y 4 para la columna). Aquí tenemos en cuenta el rebote de la señal que generan los pulsadores.
En este programa adaptamos un preescaler para generar ciclos de 50 ms de la siguiente manera:
- Se emplea un preescaler de 256 y al TMR0 se le carga con 195 (realmente cargamos el complemento en hexadecimal ya que el TMR0 es ascendente y no descendente).
- La velocidad de trabajo es de 4MHz y por tanto el TMR0 se incrementa cada uS.
- De esta forma, el TMR0 debe contar 195 eventos que, con un preescaler de 256 hace un intervalo total de 50000 uS/50 mS (195 * 256).

Este bucle lo usamos para contar, lo que nos permite programar un delay, el cual usamos para separar las detecciones de los pulsadores (esta programación también se aprovecha para controlar el rebote del pulsador, aunque

se produzca por un fenómeno diferente la solución por software puede ser la misma). Y como antes mostramos el código leído en este caso del teclado (4 bits para la fila y 4 bits para la columna).
La rutina de lectura para barrer el teclado que se ha usado es la más común:

- Por defecto se lee un 1 en las salidas.
- Ponemos un 0 en una de las filas y leemos las columnas por orden, lo que nos da un barrido completo.
- En este caso el teclado tiene 16 teclas por lo que la salida es de 8 bits, los 4 primeros que indicarán la fila y los 4 últimos indicarán la columna con un '0' y tres '1' en ambos casos.

5.3 Tiempo de delay seleccionable

Este programa es una variante del anterior, en este caso mediante los pulsadores escogeremos el tiempo de delay. Si la tecla es la "0" el tiempo serán 0,5 s, y si es cualquier otra, este tiempo serán 2 s. Estos valores han sido introducidos mediante programación como valores constantes.

5.4 Tiempo de delay programable

En este caso podemos ver como el delay del que hemos hablado con anterioridad es programable. Para ello aprovechamos el contador que teníamos para calcular el tiempo en contrarreloj, y hacemos un bucle exactamente igual, usando los 50 ms de cada ciclo mientras haya una tecla pulsada para incrementar una variable, la cual será la que después iremos decrementando cada vez que pulsemos una tecla, haciendo así nuestro delay programable.

5.5 Función dirigida

Este es una programación de la funcionalidad guiada en la cual primero se enciende un led y has de accionar el pulsador adecuado para que pase a encenderse el siguiente (todavía no se contempla el sonido). La programación es sencilla ya que solo varía el orden del segundo programa desarrollado, pero debe de comprobarse que funciona aun cambiando el orden.

5.6 Generación de sonido

Para esta prueba se ha creado una librería que contiene contadores equivalentes a la mitad del periodo de la frecuencia de la nota que se quiere utilizar (en este caso sol, la, si y do). Con estos tiempos y de la forma que hemos visto en el apartado 2.6 se generan los trenes de pulsos adecuados. Este programa lo que finalmente hace es habiendo declarado cuatro entradas, en ellas conectados pulsadores y una salida, donde está conectado el altavoz, detecta que pulsador ha sido presionado y se emite una nota diferente para cada pulsador.

5.7 Guardado de datos

En este caso lo que se ha probado es el guardado de datos, utilizando la memoria EEPROM. Para familiarizarme con las instrucciones que lo permiten, únicamente se guarda un valor de 8 bits que identifica el pulsador que se ha accionado.

5.8 Simulación

En esta etapa es donde podemos comprobar la correcta integración entre el diseño de nuestro circuito y el código fuente que hemos programado. Como podemos ver en la *Figura 10*, estamos simulando la pulsación del primer pulsador conectado a RB0, y esto se ve reflejado en el primero de los leds conectado a RA0. También se puede apreciar en RB6 el led que avisa al usuario que el sistema está conectado y por lo tanto siempre está encendido.

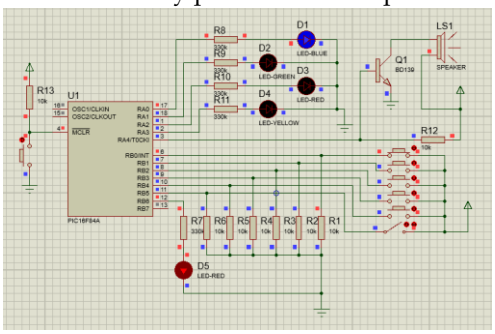

Figura 10: Sección de simulación de la herramienta de desarrollo Proteus 8.4.

6 Implementación Hardware

En este apartado hablaremos sobre la depuración del hardware, igual que en la de software, esta se hace de forma modular. Concretamente para el diseño que se ha realizado a continuación se explica cómo hacer una prueba de un módulo: primero se debe grabar el PIC con un programa que mantenga un led encendido de forma constante y uno que se encienda cada vez que se oprima un pulsador. El segundo paso a realizar es conectar alimentación y tierra del microcontrolador a una fuente de alimentación, y la entrada del oscilador a un generador de onda cuadra de 4MHz (si vemos la salida del oscilador, es decir una vez ha pasado por el microcontrolador, podemos ver como la señal se atenúa un poco y se ve afectada por algo de ruido). A continuación conectamos un pulsador primero directamente a tensión y tierra para poder ver mediante el oscilador que la señal varia correctamente, tras esto ya se puede conectar al microcontrolador con su pertinente resistencia.
Con estas herramientas podemos depurar el comportamiento del hardware según la necesidad.
Es importante listar todos los componentes que forman el sistema y ver su precio para controlar el presupuesto (*ver tabla 2*).

TABLA 2
LISTADO DE COMPONENTES Y PRECIOS UNITARIOS

Componente	Cantidad	Precio unitario
PIC 16f84a	1	4.95€
Altavoz	1	2.75€
Pulsador grande	4	3.12€
Pulsador pequeño	2	1.68€
Interruptor	1	2.00€
Led baja intensidad	1	0.25€
Led alta intensidad	4	0.90€
Transistor bd 193	1	2.10€
Oscilador 4MHz	1	2.25€

Condensador 27 pF	2	0.15€
Resistencia 10kΩ	8	0.10€
Resistencia 300Ω	5	0.10€

Una vez están definidos todos los periféricos que interactuarán pasamos a describir el circuito (*ver figura 11*):

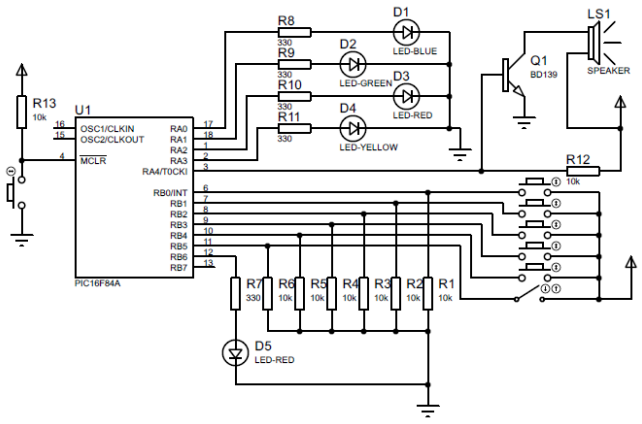

Figura 11: Diseño del circuito eléctrico del dispositivo.

A la izquierda un circuito de reset gobernado por un pulsador, el cual nos permite resetear el sistema entero.
Las 4 primeras E/S del PortA están conectadas a unos leds con sus respectivas resistencias.
La 5 E/S del PortA está conectada a un transistor y una resistencia pull-up, el primero amplifica y permite el correcto funcionamiento del altavoz, mientras que la resistencia conectada a 5V está así ya que esta última E/S es de colector abierto y esa es la forma de conseguir esa carga positiva cuando se configura como salida.
Las cuatro primeras E/S del PortB se configurarán mediante software para que sean entradas y en esta se colocan los pulsadores que interactúan tanto con su respectivo led como con el altavoz.
El último pulsador, conectado a la cuarta E/S del PortB es el que usamos para determinar el tiempo de delay.
En la quinta E/S hay conectado un interruptor, este hace la función de switch entre los diferentes modos de uso, de esta forma se identifica visualmente cuál de los dos modos de uso está en actual funcionamiento. El poder realizar las diferentes configuraciones mediante diferentes componentes es para facilitar el uso del dispositivo al usuario final.
Por último hay conectado un led, el cual siempre estará encendido y sirve para la simple razón de realimentar al usuario informando de que el sistema está en marcha.

Cabe destacar que el valor de las resistencias escogidas tanto para los leds como para los pulsadores o el interruptor ha sido por encima de un valor habitual para asegurar el funcionamiento en simulación. Para las pruebas unitarias de hardware se disminuye este valor con el objeto de disminuir el consumo eléctrico del dispositivo final, para el buen uso de los leds, se aprovecha el tren de pulsos generado mediante código para formar el sonido de las notas correspondientes para iluminar el led de forma intermitente, pero imperceptible al ojo humano y que no se mantenga encendido de forma constante, para asegurar así mayor durabilidad de este

7 RESULTADOS

Como resultado queda un prototipo modular del dispositivo, donde tenemos unos pulsadores adaptados ergonómicamente a la medida de una mano, usando leds de alta intensidad ayudados por la pintura que tiene el pulsador, focalizan la luz en una sola dirección. El pulsador también dispone de una espuma bajo la superficie de presión que ayuda a repartir la fuerza ejercida, como podemos ver a la izquierda de la *figura 12*, a la derecha se ve un pulsador sin esta superficie de presión y la bisagra que guía el movimiento.

Figura 12: Comparación de un pulsador encendido y con superficie adaptada y uno sin esa superficie donde se ve la bisagra guía.

Podemos ver en una protoboard (*ver figura 13*), el microcontrolador de color negro, conectado a alimentación y tierra mediante cables de color rojo y negros respectivamente, justo a su derecha hay conectado un oscilador y sus dos condensadores conectados también a tierra que ayudan a mantener una señal estable. También se pueden ver packs de resistencias naranjas justo encima, unas de 300 Ohmios para los leds, y las otras de 10k Ohmios para los pulsadores, los interruptores y el altavoz, este último lo podemos ver arriba a la izquierda junto al transistor que sirve para amplificar la señal (modelo bd193, recomendado por microchip para este tipo de circuitos). En la parte de abajo a la izquierda vemos un pequeño pulsador que completa el circuito de reset, en el extremo contrario vemos uno de los pulsadores con los que interactúa el usuario y justo debajo el interruptor que sirve para seleccionar el modo de uso. El pequeño led rojo que se ve es el que indica si el sistema está encendido o no.

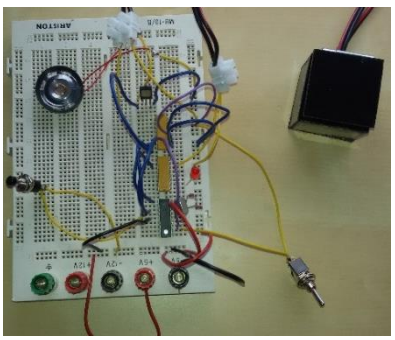

Figura 13: Parte del circuito final montado sobre una protoboard.

Cabe añadir que no se ha añadido el pulsador de configuración del tiempo de antirebote, ni los otros tres pulsadores que faltan para el usuario ya que el exceso de cables impediría mostrar el circuito en una fotografía con claridad.
Desde el punto de vista de software el programa cuenta con dos funcionalidades principales, una denominada "libre" la cual permite al usuario presionar un pulsador cualquiera, y el sistema: identifica el pulsador que se ha accionado, ilumina el led pertinente, genera y reproduce el sonido correspondiente, una vez deja de estar presionado, el sistema vuelve a su estado inicial y comprueba el estado del resto de pulsadores.
La funcionalidad "guiada" en cambio funciona de la siguiente manera: el sistema decide que sonido es el que reproduce (la secuencia está programada, no es aleatoria), ilumina el led correspondiente y se mantiene en este estado hasta que el usuario oprime el pulsador pertinente.
El programa también permite escoger en cualquier momento cuál de las dos funcionalidades es con la que queremos trabajar.

8 Conclusiones y líneas futuras

Se ha realizado el diseño software, hardware y de un sistema físico completo, incluyendo aprender a usar las herramientas pertinentes, lenguajes y métodos de programación adecuados. También se ha aprendido a implementar dichos diseños, además a depurarlos de una forma adecuada dando como resultado la puesta en marcha de un sistema modular funcional, y un conjunto de pulsadores adaptados a las necesidades que se han mostrado a lo largo del artículo.

Una vez finalizado el proyecto, podemos concluir que se ha cumplido el objetivo principal, y los dos objetivos secundarios con mayor prioridad.
La primera que debemos nombrar es la que corresponde al último objetivo, y es la de monitorizar el uso del dispositivo, es decir guardar datos que correspondan al pulsador accionado y al tiempo de respuesta, con lo que se podrá adquirir conocimiento. En el propio microcontrolador existe un bloque de memoria donde podemos escribir y mediante el programa grabador, podemos leer los datos que hemos guardado para un posterior tratamiento.
Durante el ensamblado del dispositivo se pensó en el uso de baterías, para crear una mayor portabilidad, aportando también dos características que son la de cargar la batería tanto ensamblada en el dispositivo o poder cargarla una vez extraída de este. Si nos damos cuenta, el dispositivo precisa de una alimentación de 5V, esta es la misma tensión que nos proporciona una conexión USB, a la cual todos tenemos acceso, incluso en un vehículo, no añadiría únicamente características a la portabilidad, sino que también se podría aprovechar el cable de datos para traspasar directamente esta información de la que hemos hablado antes a otro dispositivo extraíble.
Una tercera línea iría ligada al modo guiado del que hemos hablado al inicio del artículo, un sonido constante puede ser obviado con el tiempo, por lo tanto lo que se propone es mediante un contador generar un time-out, y que este de paso a una señal de intermitencia lumínica, con lo que se conseguiría volver a tener la atención del usuario.
Los pulsadores en cambio podrían mejorar añadiendo una textura del mismo color que el led para mejorar la aceptación.
En un sistema musical es interesante el uso de un potenciómetro para regular el volumen de los sonidos e incluso añadir una salida para poder acoplar un Jack.
Con el sistema programable, se pueden añadir nuevas actividades que ayuden a desarrollar la destreza motriz de los usuarios con la motivación de interactuar con un instrumento musical.

9 Agradecimientos

Quiero agradecer a Lola la ayuda, la confianza y el apoyo para hacer realidad esta idea. Sin olvidar al centro ASDI de Sant Cugat por la colaboración.

10 Bibliografía

[1] Curso práctico con pic. Bilbao: Microsystem Engineering, 1997. Consultado 2/2016.
[2] Manual de usuario "Micro PIC' Trainer". Bilbao: Microsystem Engineering, 1997. Consultado 2/2016.
[3] Manual de usuario "Micro PIC' I/O". Bilbao: Microsystem Engineering, 1997. Consultado 2/2016.
[4] Manual de usuario "Micro PIC' Trainer Plus". Bilbao: Microsystem Engineering, 1997. Consultado 2/2016.
[5] Datasheet 16f84a, 1st ed. MicroChip,1998, Consultado 2/2016. Disponible: http://ww1.microchip.com/downloads/en/DeviceDoc/35007C.pdf
[6] C. Reyes, Microcontroladores PIC, 2nd ed. Ripergraft, 2006. Consultado 2/2016.
[7] P. Ponce, "Sistemas robustos", 2006. Consultado 2/2016. Disponible: http://revistamarina.cl/revistas/2012/6/ponce.pdf
[8] Instituto de Biomecánica de Valencia,"Ergonomía y discapacidad". Consultado 2/2016. Disponible: http://www.uva.es/export/sites/uva/6.vidauniversitaria/6.11.accesibilidadarquitectonica/_documentos/Ergonomia.pdf
[9] J. Usalola Garcia, Apuntes de teoría de circuitos, 1st ed. Madrid: Universidad Carlos III, 2002. Consultado 2/2016. Disponible: http://e-archivo.uc3m.es/bitstream/handle/10016/8973/TCircuitos.pdf?sequence=1
[10] C. Johnson, "Cómo trabajar con niños con capacidades diferentes a través de la música", El How Español, 2016. Consultado 4/2016. Disponible: http://www.ehowenespanol.com/ninos-capacidades-diferentes-traves-musica-como_538435/
[11] "Talleres de educación musical para personas con discapacidad", Asociación de música y discapacidad, 2016. Consultado 4/2016.Disponible: http://www.musicaydiscapacidad.org/IMG/pdf/RESUMEN_PROYECTO_E_MUSICAPARATODOS.pdf
[12] Tododisca"La flauta màgica, instrumento para persones con discapacidad severa". Consultado 2/2016. Disponible: http://www.tododisca.com/la-flauta-magica-instrumento-musical-para-personas-con-discapacidad-fisica-severa-tododisca/
[13] Ministerio de sanidad, Servicios sociales e igualdad, portal informativo. Consultado 3/2016. Disponible: http://www.msssi.gob.es/ssi/discapacidad/home.htm

APÉNDICE

A1. DIAGRAMA DE FLUJO COMPLETO

Por último podemos ver el diagrama que pertenece al programa completo en la *Figura 14*, y en la *tabla 3* la asignación de las siglas. Éste está dividido por colores, primero vemos en azul una inicialización previa y unas asignaciones a las salidas fijas, las cuales se usan para volver al estado inicial a cada iteración. La primera cuestión a la que se enfrenta es a la de ver si el pulsador de configuración de delay está siendo presionado, entrado así al contador que antes hemos comentado (verde). A continuación vemos el estado del interruptor el cual nos bifurca el camino en los dos modos diferentes. Si está a '0' irá al modo libre (naranja) y sino estará a '1', es decir irá al modo guiado (violeta). Las dos subrutinas convergen en el contador de tiempo (verde) ya sea con el valor predeterminado o el marcado por nosotros como he explicado anteriormente. Finalmente el bucle vuelve a iniciarse al devolver las salidas al estado inicial. De esta forma no ha de iniciarse el dispositivo en uno de los modos, sino que este puede ser cambiado durante la ejecución del programa igual que el tiempo de espera.

TABLA 3
ASIGNACIÓN DE LAS SIGLAS USADAS EN EL DIAGRAMA DE FLUJO

Sigla	Significado
CE	Contador interno que se utiliza para calcular el delay.
PA, PB, PC, PD	Pulsadores con los que interactúa el usuario ordenados de forma alfabética.
PE	Pulsador que permite configurar el tiempo de delay.
SA	Interruptor que permite seleccionar el modo de uso (libre o guiado).
AuxCE	Registro auxiliar que permite guardar un valor y con este actualizar el registro contador.
LA, LB, LC, LD	Leds de alta intensidad situados en los pulsadores ordenados de forma alfabética.

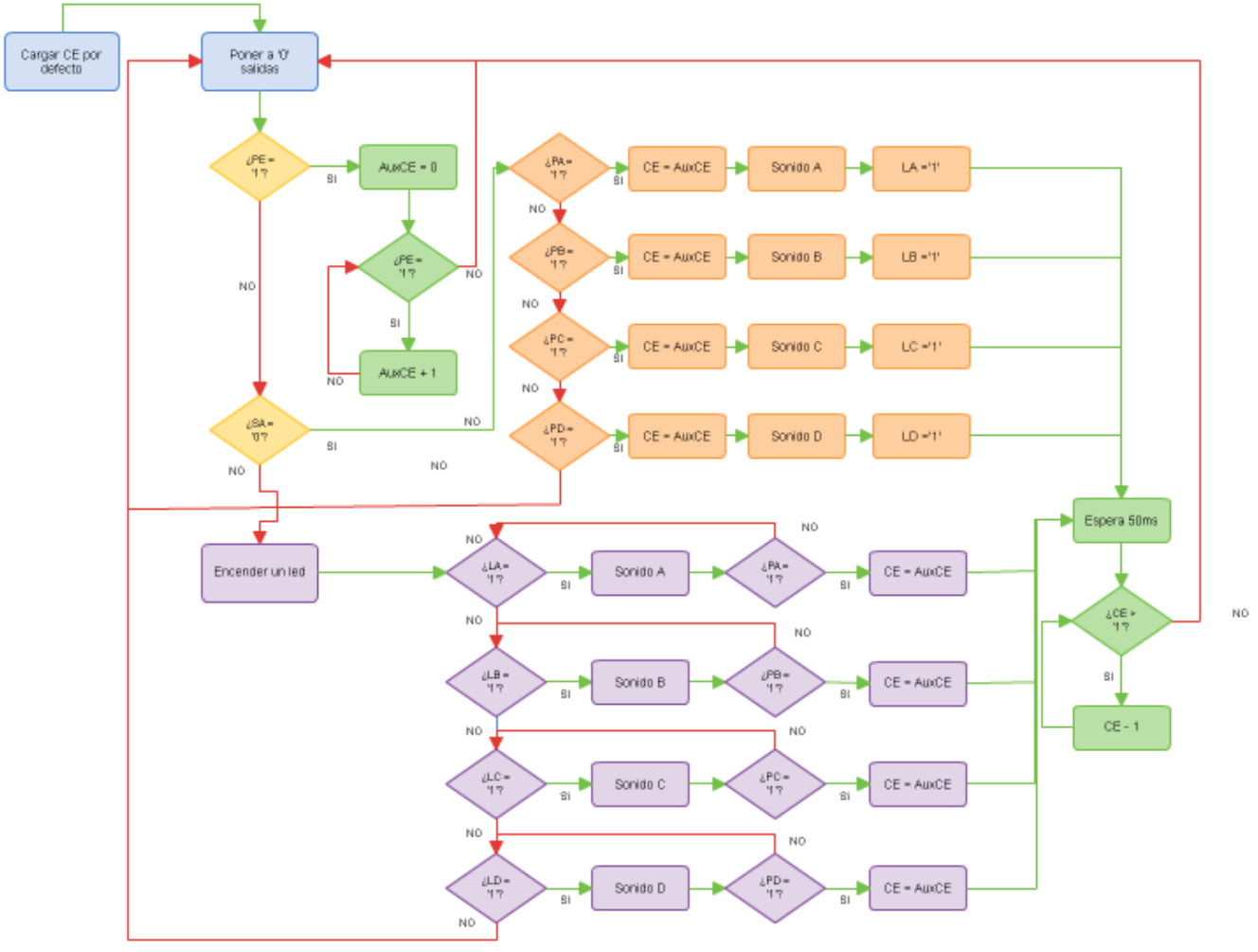

Figura 14: Diagrama de flujo del programa.

A2. CÓDIGO FUENTE

DECLARACIONES PREVIAS

```
;*****************************************************************
        LIST    P=16F84A
        INCLUDE<P16F84A.INC>

        CBLOCK                  0x38
        ENDC
        Var_Aux_Contador        equ     0x12    ;variable donde se guarda el valor del contador a actualizar
        Var_Contador            equ     0x0c    ;contador para calcular el tiempo de delay
        #DEFINE   Salida PORTA,4                ;definición de los puertos
        #DEFINE    Led1 PORTA,0
        #DEFINE    Led2 PORTA,1
        #DEFINE    Led3 PORTA,2
        #DEFINE    Led4 PORTA,3
        #DEFINE    Sol PORTB,0
        #DEFINE    La  PORTB,1
        #DEFINE    Si  PORTB,2
        #DEFINE    DO  PORTB,3
        #DEFINE    Config PORTB,4
        #DEFINE    Switch  PORTB,5
;*****************************************************************
org     0x00                ;Vector de Reset
goto    Inicio
org     0x200               ;Salva vector de interrupción
;*****************************************************************
```

INICIALIZACIÓN

```
;*****************************************************************
Inicio: clrf    PORTA           ;Aseguramos el estado inicial a '0'
        clrf    PORTB
        bsf   STATUS,RP0        ;Acceso al banco 1
        clrf   TRISA            ;Puerto A declarado como salida
        movlw  b'00111111'
        movwf  TRISB            ;Puerto B como entrada excepto el led de on
        movlw    b'00000111'
        movwf    OPTION_REG     ;Preescaler de 256 para el TMR0
        bcf   STATUS,RP0        ;ACCESO AL BANCO 0
        movlw    0x28           ;Ponemos un valor por defecto
        movwf    Var_Aux_Contador
        movlw  b'01000000'      ;Encendemos el led de on
        movwf  PORTB
;*****************************************************************
```

COMPROBACIÓN DE PE Y SA

```
;*****************************************************************
Main:           btfsc   Config              ;Comprobamos si el pulsador de configuración está apretado
                goto    CalculoVarContador  ;Si, saltamos a calcular el valor
                btfsc   Switch              ;No, miramos qué modo hay activo
                goto    modo_guiado
;*****************************************************************
```

COMPROBACIÓN DEL ESTADO DE UN PULSADOR

```
;*****************************************************************
Ver_Sol:            bcf    Salida                  ;Limpiamos la salida
                    btfss  Sol                     ;Esta presionado el boton?
                    goto   Ver_La                  ;No, miramos el siguiente pulsador
                    movf       Var_Aux_Contador,W  ;Reiniciamos el contador
                    movwf      Var_Contador
SueltaSol:          call   SuenaSol
                    btfsc  Sol                     ;Esta presionado el boton?
                    goto   SueltaSol               ;Si, entramos en bucle de sonido, no pues esperamos el delay
DelayContador1:     bcf        INTCON,T0IF         ;Desconecta el flag de rebosamiento
                    movlw      0x3c                ;Complemento hex. de 195
                    movwf      TMR0                ;Carga el TMR0
IntervaloDelay1:    btfss      INTCON,T0IF         ;Rebasamiento del TMR0 ??
                    goto       IntervaloDelay1     ;Todavía no
                    decfsz     Var_Contador,F      ;Decrementa contador de intervalos
                    goto       DelayContador1      ;Repite el intervalo de 50 mS
                    goto   Main
;*****************************************************************
```

ITERACIÓN DEL MODO GUIADO PARA UNA NOTA

```
;*****************************************************************
Primero:  call      SuenaSol
          btfss     Sol             ;Esta presionado el boton?
          goto      Primero         ;No
          clrf      PORTA           ;Si apagamos los leds y el sonido
;*****************************************************************
```

SUBRRUTINA PARA GENERAR TRENES DE PULSOS

```
;*****************************************************************
SuenaSol:           movlw  b'00010001'     ;Ponemos un 1 en el la salida del altavoz y el led correspondiente
                    movwf  PORTA
                    call   RetarSol        ;Llamamos a la subrutina de la libreria de esperas
                    clrf       PORTA
                    Call   RetarSol
                    Return
;*****************************************************************
```

CALCULO DEL DELAY

```
;*****************************************************************
CalculoVarContador:     movlw   0x00                ;Aseguramos que empieza a contar desde 0
                        movwf   Var_Aux_Contador
Sumatorio:              bcf     INTCON,T0IF         ;Desconecta el flag de rebosamiento
                        movlw   0x3c                ;Complemento hex. de 195
                        movwf   TMR0                ;Carga el TMR0
IntervaloCont:          btfss   INTCON,T0IF         ;Rebasamiento del TMR0 ??
                        goto    IntervaloCont       ;Todavía no
                        incf    Var_Aux_Contador
                        btfss   PORTB,4
                        goto    Main                ;No
                        goto    Sumatorio           ;Si, Repite el intervalo de 50 mS
;*****************************************************************
```