industriales
etsii

Escuela Técnica Superior de Ingeniería Industrial

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Industrial

Diseño e implementación de una placa entrenadora/programadora para microcontroladores PIC

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: **José Martínez Álvarez**
Director: Manuel Sánchez Alonso

Cartagena, 6/06/2019

Universidad Politécnica de Cartagena

Índice

1. Introducción

En la titulación del grado de Ingeniería Electrónica Industrial y Automática existe una asignatura contenida en su plan de estudios dedicada al estudio de los microcontroladores, estos dispositivos son empleados para realizar el control sobre dispositivos electrónicos, ya sean digitales o analógicos.

En las titulaciones del grado en Ingeniería Telemática y del grado en Ingeniería en Sistemas de Telecomunicación se incluye también una asignatura dedicada al estudio de los dispositivos antes mencionados, esta asignatura se denomina Sistemas Digitales Basados en Microprocesadores, impartida en el 2º cuatrimestre del 2º curso.

Este proyecto va dirigido al grado de Electrónica, cuya asignatura se llama Sistemas Basados en Microprocesador, la cual es impartida en el 2º cuatrimestre del 3er curso del grado antes mencionado. Tiene 4,5 créditos de los cuales, 3 son teóricos y 1,5 prácticos. Los contenidos teóricos incluyen el estudio de la arquitectura básico de los microcontroladores, el manejo de dispositivos de entrada y salida, así como el funcionamiento y programación de los recursos que incluyen dichos dispositivos.

En cuanto al contenido práctico de la asignatura, desarrollados en sesiones de dos horas, son los siguientes:

- Práctica 1: Introducción al manejo de herramientas de simulación como MPLAB y Virtual BreadBoard.
- Práctica 2: Ejercicios con operaciones aritmético-lógicas, comprobando el funcionamiento del registro de estado.
- Práctica 3: Manejo de los puertos de entrada y salida mediante interruptores y diodos led.
- Práctica 4: Configuración de los microcontroladores como contadores y temporizadores.
- Práctica 5: Gestión de la temporización a través de intermitencias de diodos.
- Práctica 6: Configuración, detección y tratamiento de interrupciones.
- Práctica 7: Lectura y escritura de la memoria EEPROM.
- Práctica 8: Puesta en marcha de una selección de ejercicios resueltos en clase.

Cabe mencionar que la realización de estas prácticas es a través de simuladores, programas que permiten representar el funcionamiento de estos dispositivos sin la necesidad de realizar conexiones físicas.

1.1. Justificación

El uso de los simuladores conlleva muchas ventajas, la posibilidad de realizar cualquier circuito que se desee sin ninguna complicación sumada al ahorro de tiempo que supone no realizar ningún montaje, se puede ver como un claro vencedor frente a las herramientas de hardware. Pero el uso de estas herramientas supone una desventaja: los simuladores pueden no asemejarse a la “realidad”.

El problema de las herramientas de hardware, entrenadoras de microcontroladores en este caso, es que, las más completas, suponen una inversión inicial elevada. Se puede pensar que una vez hecha esta inversión nos olvidamos, pero el uso continuo puede provocar fallos o roturas. A esto hay que sumarle que la conectividad con periféricos en algunos casos puede estar limitada, puesto que las conexiones están preestablecidas, esto implica no poder realizar el montaje de todos los circuitos que se necesiten.

La mejor manera de aprender a manejar los microcontroladores es combinar ambas herramientas, el uso de los simuladores para poder familiarizarse como el software y programación de estos dispositivos y el uso de las herramientas de hardware para comprobar cómo se comporta el mismo programa en la vida real.

1.2. Objetivos

El objetivo es el diseño y fabricación de una placa entrenadora adaptada a las necesidades de las prácticas de la asignatura Sistemas Basados en Microprocesador del Grado de Electrónica Industrial y Automática. Con este proyecto se busca proporcionar equipos de prácticas a un coste razonable, de manera que los futuros estudiantes de dicha asignatura puedan trabajar, tanto a nivel de software como de hardware, con los microcontroladores PIC.

Este es el objetivo general. Se plantean otros objetivos específicos a realizar para llevar a cabo este proyecto:

- Estudio de las placas entrenadoras existentes en el mercado para realizar prácticas con microcontroladores PIC.
- Establecer las necesidades de conectividad de periféricos en las prácticas de la asignatura.
- Estudio de los posibles grabadores de EEPROM.
- Diseño de la placa PCB adaptada a las necesidades de las prácticas.
- Puesta en marcha de un prototipo, realizando todas las pruebas pertinentes para comprobar su funcionamiento.

El resultado final proporcionará el diseño de una placa funcional que pueda fabricarse en masa para proveer a los alumnos equipos de prácticas, las cuales podrán ser usadas tanto para los alumnos del grado de Electrónica, como los de Telemática y Telecomunicaciones.

2. Marco teórico

2.1. Microcontrolador

Un microcontrolador es un circuito integrado el cual es programado para realizar una tarea determinada, y debido a su pequeño tamaño suele incorporarse en el propio dispositivo que gobierna, por eso se puede denominar también como microcomputador empotrado.

Podemos encontrarlos en numerosas aplicaciones debido a su reducido coste y consumo, como en electrodomésticos (televisores, lavadoras, microondas, etc.), sistemas de telecomunicaciones (teléfonos móviles, etc.). También los encontramos en sistemas informáticos, la mayoría de los periféricos de los computadores están gobernados por un microcontrolador (ratones, teclados, impresoras, etc.). Se encuentran también en la industria automovilística, para el control de la climatización, la seguridad y los frenos ABS.

2.1.1. Arquitectura interna

En los microcontroladores se almacenan datos e instrucciones. Según como esté organizada la memoria influye en las prestaciones de los microcontroladores.

La arquitectura de los computadores se dividen en dos tipos: Von Neumann y Harvard. Ambas presentes en los microcontroladores. La arquitectura Von Neumann se caracteriza por tener una memoria donde se almacena instrucciones y datos, es decir, la CPU (unidad central de procesos) se conecta a una memoria única mediante un sistema de buses.

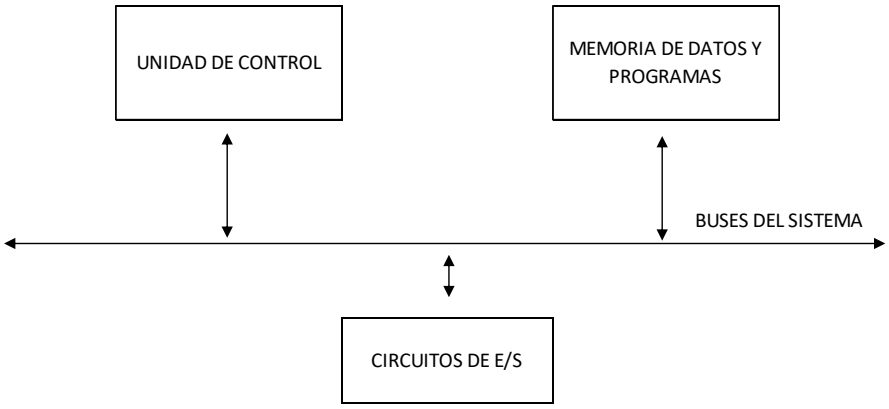


Ilustración 1: Arquitectura Von Neumann

La arquitectura Harvard por el contrario tiene separadas ambas memorias y cada una dispone de sus propios sistemas de buses. Posee una memoria de programa, donde almacena las instrucciones, y otra memoria de datos, donde almacena los datos. Esta memoria es la que emplean los microcontroladores.

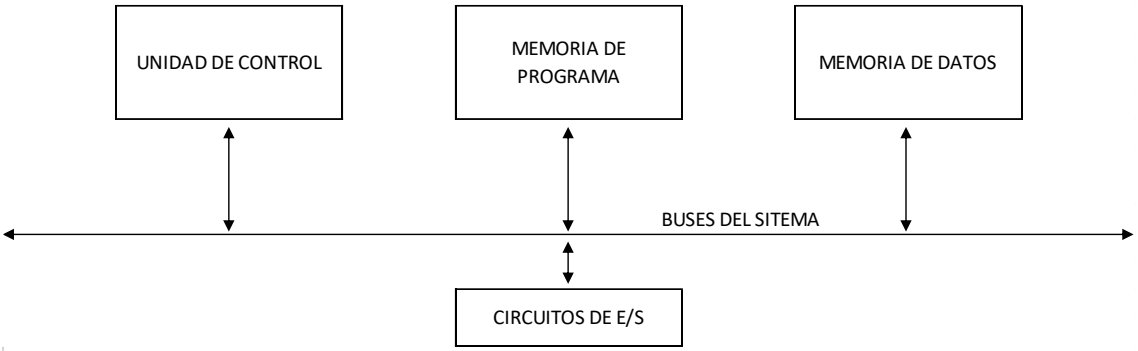


Ilustración 2: Arquitectura Harvard

Los microcontroladores combina en un único integrado todas las partes fundamentales que componen un microcomputador: el procesador, la memoria, líneas de E/S y recursos auxiliares. Estos elementos están conectados a través de buses que pueden ser de direcciones (si transportan direcciones de memoria o de entrada y salida), de datos (si transportan datos o instrucciones) o de control (si transportan señales de control).

2.1.1.1. Procesador

El procesador es el "cerebro" del microcontrolador y se encarga de interpretar las instrucciones del programa almacenado en la memoria y ejecutarlas. También incluye la denominada Unidad Aritmética Lógica, que posibilita realizar operaciones aritméticas y lógicas elementales con los datos binarios.

El procesador de los microcontroladores modernos responde a la arquitectura RISC (*Reduced Instruction Set Computer*), el cual posee un repertorio de instrucciones pequeño y simple, de forma que la mayor parte de las instrucciones se ejecuta en un ciclo de instrucción.

Además, el procesador se encuentra segmentado (pipe-line), es decir, esa descompuesto en etapas para poder procesar una instrucción diferente en cada una de ellas y trabajar con varias a la vez. Esto aumenta el rendimiento del computador.

2.1.1.2. Memoria de programa

El microcontrolador está diseñado de manera que todas las instrucciones del programa se almacenen en su memoria de programa. En este tipo de memoria el acceso a los datos es rápido, sin embargo el proceso de escritura es lento. Como la información contenida siempre será la misma, el programa debe estar grabado de forma permanente. Los tipos de memorias adecuados son los siguientes:

- Máscara ROM:

El programa es grabado durante la fabricación del microcontrolador mediante máscara y no puede ser modificado por los usuarios. Este tipo de memoria es solo recomendable cuando se requieren lotes muy grandes, ya que presenta un alto coste de diseño.

- EPROM (*Electrical Programmable Read Only Memories*):

Este tipo de memoria permite ser grabado mediante un dispositivo conocido como grabador mediante de un ordenador personal. Se puede borrar y volver a grabar el código siempre que se desee. Para borrar el código hay que aplicar al circuito integrado rayos ultravioletas a través de una ventana transparente. Presenta un coste unitario elevado.

Este tipo de memoria ha caído en desuso, debido a la aparición de tecnologías menos costosas y más flexibles, como la memoria EEPROM y FLASH.

- Memoria PROM (*Programmable Read Only Memories*) u OTP (*One Time Programmable*) :

La memoria OTP, como indica su nombre, solo es posible ser grabada una vez. Al igual que en el caso de la memoria EPROM se graba mediante un grabador. La diferencia entre estas dos, es que la memoria PROM no posee una ventana trasparente. Se recomienda su uso en sistemas donde no requiera futuras actualizaciones y para series relativamente pequeñas.

- EEPROM (*Electrically Erasable Programmable Read Only Memories*):

Al igual que las dos anteriores la grabación se realiza con un grabador, sin embargo, el borrado se realiza de la misma forma que la grabación, es decir, eléctricamente. Puede ser programada y borrada tantas veces como se quiera. Se garantizan 1.000.000 ciclos de escritura/borrado.

Una de las características más destacable es que fue en este tipo de microcontroladores donde se empezó a utilizar la programación ICSP, que evitan tener que sacar el microcontrolador del circuito donde está instalado.

- FLASH:

Como en las EEPROM se puede grabar y borrar eléctricamente y suelen disponer de mayor capacidad. Se trata de una memoria de bajo consumo y no volátil. Se garantiza hasta 1.000 ciclos de escritura/borrado. Se emplean en aplicaciones en las que no es necesario modificar el programa a lo largo de la vida útil del producto, como por ejemplo en vehículos. Están sustituyendo a las memorias EEPROM.

Las memorias para almacenar código más empleadas hoy en día son las memorias EEPROM y Flash, ya que podemos grabarlas y borrar su contenido muchas veces eléctricamente sin necesidad de sacar el microcontrolador del zócalo del grabador.

2.1.1.3. Memoria de datos

La memoria de datos se encarga de almacenar la información que el procesador requiere para realizar las operaciones que le indiquemos.

Como los datos que controlan los programas varían continuamente, se necesita que la memoria que almacena los datos sea de escritura y lectura. Para ello se emplea la memoria RAM. Esta permite ser escrita y leída una infinidad de veces.

Hay algunos microcontroladores que disponen de una memoria de datos tipo EEPROM. A diferencia de la memoria RAM, la memoria EEPROM no es volátil, por lo que no se pierde información al producirse un corte en la alimentación. Esto es muy útil para guardar contraseñas o nombres de usuario.

2.1.1.4. Líneas de E/S

Las líneas de E/S se emplean como interfaz entre el mundo exterior y el procesador. Los microcontroladores poseen varias patillas para comunicarse con los periféricos externos que controla. Hay dos formas de transmitir la información entre un periférico y el procesador: en paralelo y en serie.

Los periféricos se dividen en periféricos de entrada o de salida. Los de entrada codifican los mensajes o señales para que el procesador pueda interpretarlas. Como por ejemplo un teclado, sensores, interruptores, pulsadores, etc. Los periféricos de salida observan los resultados obtenidos por el procesador. Como por ejemplo una pantalla, motores, relés, diodos led, etc.

2.1.1.5. Recursos auxiliares

Los microcontroladores, dependiendo del fabricante y el modelo, incluyen una serie de complementos para mejorar la potencia y flexibilidad del dispositivo. Los recursos más comunes son los siguientes:

- Circuito de reloj: mediante un oscilador se generan los pulsos que sincronizan todas las operaciones internas. Pueden de varios tipos, desde un tipo RC, hasta el oscilador de cuarzo que es el más empleado debido a su gran estabilidad de frecuencia. La frecuencia del oscilador influye en la velocidad de ejecución de las instrucciones.
- Temporizadores empleados para controlar tiempos.
- Perro guardián: el perro guardián (WDT: *Watchdog Timer*) se trata de un temporizador de N bits que provoca el reset del microcontrolador cuando se desborda. Una vez iniciado el conteo no se puede parar, solo es posible poner a cero el contador desde el programa. Es un elemento muy importante pues permite detectar a tiempo cualquier fallo.
- Conversores AD y DA para poder enviar y recibir señales analógicas.
- Comparadores analógicos.
- Sistema de protección ante fallos de la alimentación.

2.1.2. Elección del microcontrolador

Las familias de microcontroladores se caracterizan, en general, por tener la misma CPU y ejecutar el mismo repertorio de instrucciones. Esto se conoce como "núcleo" (*core*). Los miembros de una familia, aunque posean el mismo núcleo, tienen diferencias en la entrada y salida y en la memoria.

Existe una gran variedad de microcontroladores en el mercado. Siendo los más importante los de las casas INTEL, MICROCHIP, MOTOROLA y ATMEL. Todos poseen características similares, como por ejemplo, todos poseen una estructura Harvard, además de recursos auxiliares, como puertas de E/S y conversores AD y DA.

Se elegirá el microcontrolador dependiendo de las características de la aplicación que se desee implementar. El tipo de microcontrolador que se estudia en la asignatura Sistemas Basados en Microprocesador y el empleado en este proyecto, como indica el nombre, son los microcontroladores PIC de Microchip.

2.2. Microcontroladores PIC

Los microcontroladores PIC son una familia de microcontrolador desarrollados por la compañía Microchip. Tienen una gran aceptación entre los profesionales y aficionados que trabajan con microcontroladores. Esto se debe a lo siguiente:

- Su coste inferior al de resto de competidores.
- Elevada velocidad de funcionamiento.
- Reducido juego de instrucciones.
- Muchas herramientas de software libres.
- Gran variedad de modelos.

2.2.1. Gamas de PIC

Microchip dispone de cuatro gamas de microcontroladores de 8 bits para adaptarse a las necesidades de la mayoría de los posibles clientes.

2.2.1.1. Gama baja (PIC16X5X):

Se trata de versiones encapsuladas con 18 y 28 patillas que pueden alimentarse a partir de una tensión de 2,5 V lo que les permite trabajar en aplicaciones que funcionan con pilas. Tiene un repertorio de 33 instrucciones de 12 bits cada una. No disponen de interrupciones y la pila es de 2 niveles.

2.2.1.2. Gama media (PIC16XXX):

Encapsulados desde 18 patillas hasta 68. Es la gama más completa de los PIC. El repertorio de instrucciones es de 35 de 14 bits y compatible con la gama baja. Disponen de interrupciones y una Pila de 8 niveles.

Dentro de esta familia se encuentra el famoso PIC16F84A, muy utilizado por los principiantes que se están introduciendo el mundo de los microcontroladores, y será el utilizado mayoritariamente para las aplicaciones desarrolladas con la entrenadora.

2.2.1.3. Gama alta (PIC17XXX):

Estos modelos constan de 58 instrucciones de 16 bits. Disponen de interrupciones vectorizadas. Lo más destacable es su arquitectura abierta, que permite ampliar el microcontrolador con elementos externos a través de buses de datos, direcciones y control, conectándole memorias o controladores de periféricos.

2.2.1.4. Gama mejorada (PIC18XXX):

Esos modelos surgen con la finalidad de soportar las aplicaciones avanzadas de automoción, comunicaciones, ofimática y control industrial. Destacan por su velocidad (40 Mhz) y su gran rendimiento. Las aportaciones más representativas de este modelo son:

- Espacio de direccionamientos para la memoria de programa de 2 MB y 4 KB para la memoria de datos.
- Memoria FLASH para la memoria de programa.
- Juego de instrucciones de 77 de 16 bits.
- Orientación a la programación en lenguaje C.
- Nuevas herramientas para la simulación.

2.2.1.5. PIC miniatura (PIC12XXX):

Su principal característica es su tamaño ya que disponen de un encapsulado de 8 patillas. Su juego de instrucciones puede ser de 33 o 35 instrucciones de 12 o 14 bits respectivamente.

2.2.2. Características generales

Los PIC poseen una estructura RISC y tipo Harvard. Esto unido a la segmentación del procesador (pipe-line) hace que sea posible ejecutar un ciclo de instrucción en 1 microsegundo si funciona a una frecuencia de 4 MHz. Todas las instrucciones tardaran en ejecutarse ese tiempo, excepto las instrucciones de salto que duran el doble.

La memoria de programa de los PIC es mucho mayor que la de memoria, como ocurre con la mayoría de los microcontroladores. Está organizada en palabras de 12, 14 o 16 bits. La memoria de datos, sin embargo, está compuesta en registros de 8 bits.

Otra característica común es como esta implementada la pila. La pila no forma parte de las memorias de datos ni de programas. Se trata de una zona aislada que tiene una profundidad limitada, dependiendo del modelo de PIC.

Todos los PIC poseen un temporizador que trabaja como perro guardián. Se puede configurar durante el proceso de grabación de los PIC mediante la palabra de configuración.

La palabra de configuración permite configurar los PIC, solo accesible durante el proceso de grabación. Permite adaptar mejor el PIC a las necesidades de la aplicación. Consiste en una serie de bits, según el modelo, permitiendo configurar lo siguiente:

- El tipo de oscilador.
- Habilitación del perro guardián.
- Protección de la memoria de programa.
- Protección de la memoria EEPROM.
- Características del reset y alimentación del dispositivo.

2.3. Descripción del PIC16f84A

En la entrenadora se va incluir zócalos para instalar los microcontroladores PIC de 18 y 28 pines y como en "Sistemas Basados en Microprocesador" se estudia la estructura y programación del PIC16F84A y las prácticas, que posteriormente van a ser resueltas y simuladas con la placa, se utiliza este dispositivo, se va a realizar una breve descripción de la estructura y arquitectura de este.

Como todos los microcontroladores posee una arquitectura Harvard, con una memoria flash de programa de 1 KB, una memoria RAM dividida en dos áreas: 22 registros de propósito específico (SFR), encargados del funcionamiento del microcontrolador y sus recursos, y 68 de propósito general (GPR), disponibles por el usuario para almacenar valores, además de una memoria EEPROM de 64 bytes.

Posee un procesador segmentado o Pipeline, que permite ejecutar una instrucción en un solo ciclo pues mientras ejecuta una instrucción busca la siguiente, excepto cuando la instrucción es de salto de programa. Cuenta con un procesador RISC, donde las instrucciones son muy simples y solo tiene 35.

Contiene una ALU de 8 bits, para la realización de operaciones aritmético-lógicas, así como un registro de trabajo, denominado acumulador (W), además de un registro de estado (STATUS), para las operaciones realizadas, indicar el estado del RESET o SLEEP y seleccionar el banco de trabajo. Un contador de programa (PCL) de 13 bits (lo que en teoría permitiría direccionar 4 KB de memoria, aunque el PIC16F84A solo tiene 1 KB.

La memoria está divida en tres bloques:

- Memoria de programa: contiene el programa con las instrucciones. Como es de tipo no volátil, el programa se mantiene aunque se pierda la alimentación.
- Memoria de datos RAM: almacena los datos y variables. Es de tipo volátil por lo que la información se pierde antes fallos de alimentación.
- Memoria EEPROM: pequeña memoria de escritura y lectura y, al igual que la memoria de programas, es de tipo no volátil.

Podemos ver la estructura en la siguiente imagen:

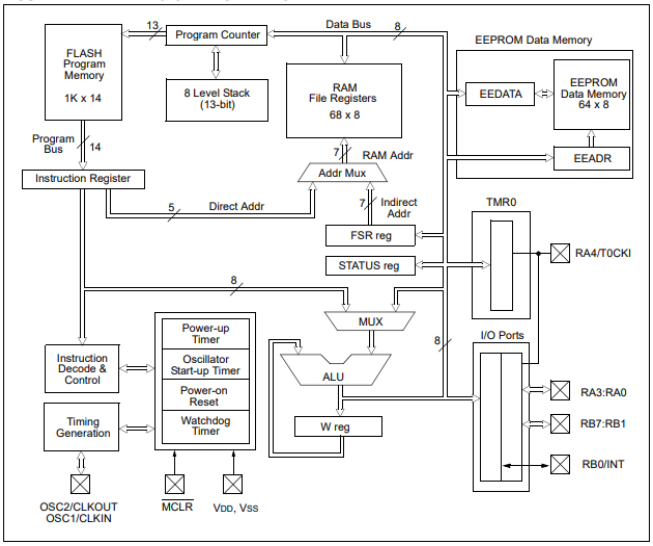


Ilustración 3: Estructura PIC16F84A

Como recursos fundamentales cuenta con dos puertas para comunicarse con el exterior, una de 5 líneas (PORTA) y otra de 8 (PORTB), un dispositivo temporizador/contador (TMR0). Además es capaz de gestionar cuatro tipos de interrupciones con un solo "vector de interrupción". Estas causas de interrupción son: desbordamiento del TMR0, fin del proceso de escritura de la memoria EEPROM, cuando se introduce un flanco por la patilla RB0 y por último, una que indica el cambio de valor de entrada en las patillas RB4:RB7 de la puerta B. Cuando sucede una de estas interrupciones el contador de programa carga la dirección del vector de interrupción (004h) donde se coloca la rutina de servicio de interrupción (RSI) diseñada.

Estas puertas pueden ser configuradas como entradas o como salidas según se desee. Esta configuración puede cambiarse a lo largo del programa a criterio del usuario y en función de las necesidades del sistema. El registro TMR0 se puede incrementar de forma síncrona, pudiéndose ser utilizar para la temporización, o de forma asíncrona cuando se estimula a través de la patilla RA4 de la puerta A. El PIC incorpora un divisor de frecuencias (DF), que recoge los estímulos y cada cierto número, introducido por los usuarios, de estos se produce un incremento. Esto permite el tiempo hasta que se produce el desbordamiento del TMR0 o aumentar la cantidad de eventos externos recibidos.

En las siguientes tablas se ve un resumen de las funciones de los pines de las puertas A y B (PORTA y PORTB):

Nombre	Bit	Función
RA0	Bit 0	Entrada/Salida
RA1	Bit 1	Entrada/Salida
RA2	Bit 2	Entrada/Salida
RA3	Bit 3	Entrada/Salida
RA4	Bit 4	Entrada/Salida y señal de reloj externa para el TMR0

Tabla 1: Funciones PORTA

Nombre	Bit	Función
RB0	Bit 0	Entrada/Salida, interrupción externa
RB1	Bit 1	Entrada/Salida
RB2	Bit 2	Entrada/Salida
RB3	Bit 3	Entrada/Salida
RB4	Bit 4	Entrada/Salida, interrupción por cambio de estado
RB5	Bit 5	Entrada/Salida, interrupción por cambio de estado
RB5	Bit 6	Entrada/Salida, interrupción por cambio de estado
RB7	Bit 7	Entrada/Salida, interrupción por cambio de estado

Tabla 2: Funciones PORTB

Por último la memoria EEPROM de datos se utiliza para almacenar los datos deseados sin que se pierda ninguna información aunque se pierda la alimentación. Emplea dos registros para el manejo de esta, uno de escritura y otro para la lectura, además de un registro para indicar la dirección y otro para la información que se lee o escribe.

2.4. Herramientas para trabajar con PIC

Para el estudio de circuitos integrados programables es necesario emplear una serie de herramientas con las que poder trabajar con el software y el hardware de estos.

Con respecto al software lo más habitual son los compiladores. Son programas informáticos que traducen los lenguajes de programación empleados, estos son indispensables para comprobar si en el código implementado existe algún error. Pero para simular la ejecución de instrucciones y poder comprobar el comportamiento del procesador y el estado de las líneas de E/S se emplean los denominados simuladores.

La empresa Microchip pode a disposición de los usuarios un entorno de simulación y programación denominado MPLAB X-IDE, el cual es compatible con todos los modelos de PIC fabricados. Para la simulación y desarrollo de software ofrece interesantes opciones, una de ellas es que permite la programación en varios lenguajes, como ensamblador o C. Además permite la simulación de varios tipos de placas entrenadoras sin necesidad de programar físicamente el PIC.

Existen otros softwares de simulación, algunos gratuitos, tales como “Real PIC Simulator” y “Multisim”, y otros de pago como “Proteus” o “Virtual Breadboard”. El

software con el que se trabaja en Sistemas basados en Microprocesador es este último, que aunque tenga un precio de 50 dólares, proporciona licencia para tres equipos. Esta aplicación permite diseñar una placa entrenadora de forma virtual hecha a medida colocando el PIC sobre una placa y conectar distintos tipos de periféricos a las patillas de E/S. Esta herramienta proporciona varios plugins que permite utilizarla con MPLAB X-IDE.

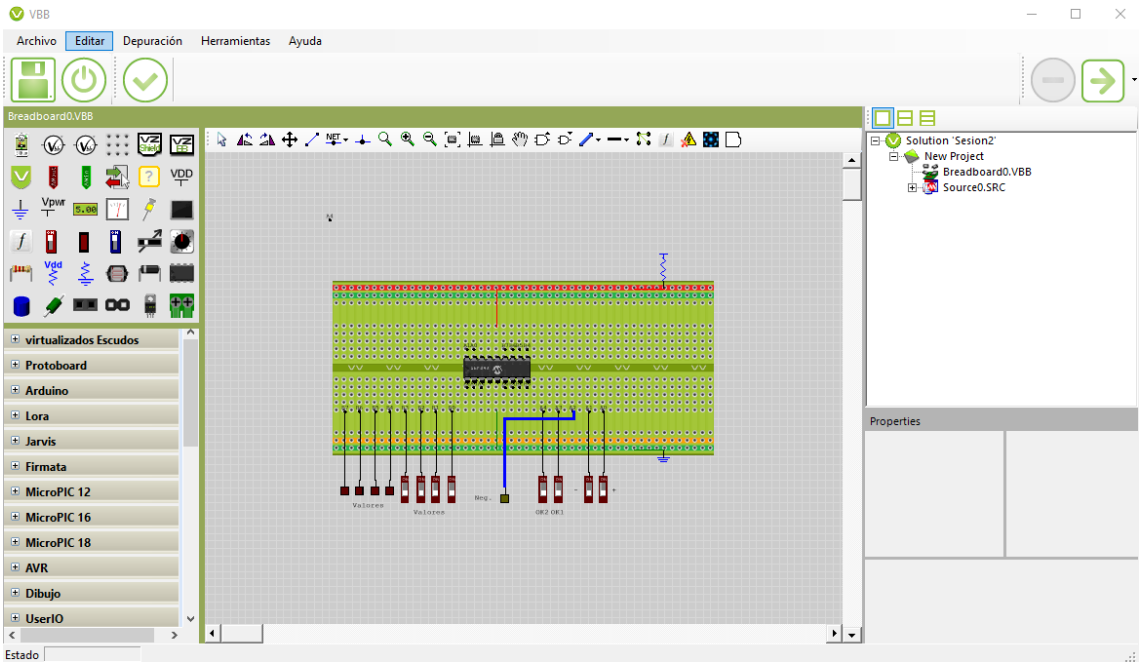


Ilustración 4: Vitual BreadBoard

En cuanto al hardware se puede emplear una placa de desarrollos para montar los circuitos que se deseen. Pero realizar cada montaje conlleva demasiado tiempo, por lo que no es una opción demasiado viable. Lo que sí es indispensable es el uso de un grabador. Este permite grabar el programa en la memoria de los microcontroladores.

Otra herramienta de hardware muy útil es lo que se denomina una entrenadora. Podemos definirla como una placa predefinida que contiene los elementos básicos que componen los circuitos de los microcontroladores, además de una serie de periféricos para poder trabajar con ellos.

2.4.1. Entrenadoras

Básicamente una entrenadora es una tarjeta electrónica, que incorpora un microcontrolador y todos los elementos necesarios para su funcionamiento (oscilador, reset, alimentación, etc.), las cuales pueden incluir o no otros periféricos de E/S. Incluye los medios necesarios, ya sea un conector de ICSP o un grabador integrado, que permite programar una y otra vez el PIC instalado, de esta manera es posible realizar cualquier tarea que se le programe.

En el mercado existen varias herramientas que permiten trabajar con este tipo de microcontroladores. Se buscan las compatibles con los microcontroladores PIC16F84A. Algunas de ellas son:

- USB-PIC'School:

Se trata de un laboratorio de carácter didáctico que permite trabajar con los dispositivos PIC de las familias 12F, 16F y 18F. Dispone de una gran cantidad de periféricos utilizados en aplicaciones reales. Algunas de sus características más relevantes son:

- Alimentación mediante fuente de alimentación de 9 a 15VDC. Con circuitos de filtrado, estabilización, piloto e interruptor ON/OFF.
- Oscilador integrado de cuarzo encapsulado en DIP8 para generar la frecuencia de trabajo e intercambiable por otros osciladores.
- Integra el hardware necesario para la depuración y grabación compatible con el PICKIT2 de Microchip.
- No hay conexiones predeterminadas.
- Gran variedad de periféricos de E/S, como: diodos led, displays 7 segmentos, interruptores y pulsadores, pantalla LCD, teclado matricial, potenciómetros, generador lógico de onda cuadrada, módulo board, conector de expansión PIC-BUS 2.

Ilustración 5: USB PIC'School

Se trata de una herramienta muy completa, que permite realizar una gran variedad de proyectos. La pega es el alto elevado que tiene, el cual es de 160€ por unidad. Esto es demasiado para 20 puestos de trabajos (el estándar en las titulaciones técnicas).

- LAB-X3 Experimenter Board:

Es una placa desarrollada por ME Labs compatibles con los PIC de 18 patillas. Contiene los siguientes elementos:

- Entradas: 1 botón y 1 potenciómetro.
- Salidas: 2 diodos led, 1 conector a un servomotor y una pantalla LCD 2x20.
- Oscilador de 4MHz, alimentación a 5 V, botón de reset.
- Conector ICSP.
- Área de prototipos para realizar circuitos adicionales.

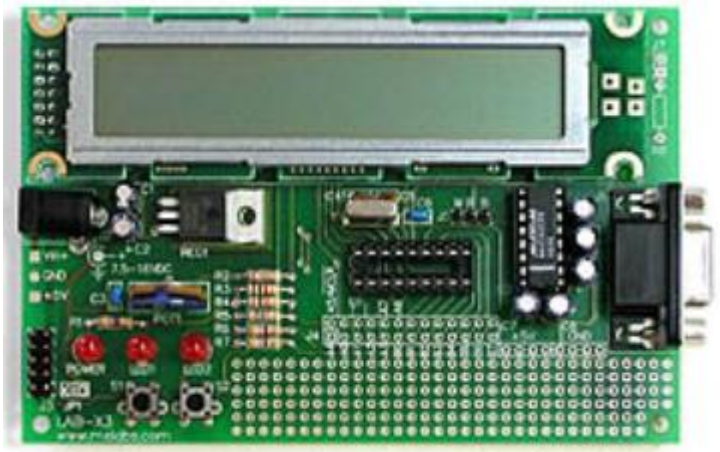


Ilustración 6: LAB-X3 Experimenter Board

Es una herramienta poco versátil ya que dispone de conexiones preestablecidas y contiene pocos periféricos, y para el precio que tiene, 119,95$, no es una buena opción frente otras herramientas más completas.

- EXPLORER 8 DEVELOPMENT KIT:

Se trata de un kit de desarrollo para trabajar con toda la gama de 8 bits de los microcontroladores PIC de 8 a 80 pines desarrollada por Microchip. Tiene las siguientes características:

- Compatible con placa MikroElektronika Click.
- Compatible con placa Diligent Pmods™.
- Conectores de expansión PICtail y PICtail Plus.
- Conector de placa complementaria de 20 pines, para placas complementarias integradas personalizadas.
- Opciones de depuración y programación flexibles: incluido el emulador en circuito PICkit™ 3, ICD 3 y MPLAB® REAL ICE™.
- Se integra con el entorno de desarrollo integrado (IDE) MPLAB® X.
- Conectores macho de programación y conexiones de alimentación para todos los conectores hembra de MCU.
- Tres fuentes de alimentación individuales: 5 V, 3,3 V y variable (1,5 -4,5 V).
- Display LCD de 16 x 2 caracteres.
- Conexiones externas para interfaces de expansión y comunicaciones estándar de la industria.
- Interruptores de botón pulsador para restablecer el dispositivo y entradas definidas por el usuario.

Ilustración 7: EXPLORER 8 DEVELOPMENT KIT

No incluye periféricos con los que poder trabajar, por lo que es necesario disponer de algunas de las placas mencionadas en las características. El precio de esta placa es de 82,36€, a esto habría que sumarle el precio de las placas auxiliares. No es una buena opción si se buscan equipos de prácticas económicos.

En general no se han encontrado muchas placas destinadas al estudio de microcontroladores PIC que sean compatibles con el PIC16F84A, para otras versiones superiores a este existen en mayor cantidad. Las encontradas compatibles resultan no se demasiado económicas y no dispones de los periféricos suficientes para poder realizar las prácticas, excepto la USB PIC'School que es una herramienta muy completa pero que presenta el mismo problemas que las demás: el precio.

2.4.2. Grabación de PIC

El proceso de grabación consiste en grabar el programa de control en la memoria de programa de los microcontroladores. Esto se realiza mediante el uso de un grabador o programador.

Cuando se desarrollan aplicaciones con microcontroladores es muy útil tener la posibilidad de reprogramarlos cuando se desee sin necesidad de extraer el PIC del circuito. Para ello se emplea la programación mediante ICSP (*In Circuit Serial Programming*, programación en serie en circuito). Esta tecnología está presente en todos los PIC más recientes.

La programación ICSP usa 5 señales para realizar la escritura, lectura y verificación de los programas. Estas señales son las siguientes:

- VDD (voltaje de alimentación): para programar un PIC es necesario que esté alimentado a 5/3,3 V dependiendo del modelo del PIC. Este voltaje se introduce por el pin VDD.
- VPP (voltaje de programación): voltaje necesario para entrar en modo programación. Se trata de un voltaje comprendido entre 12 y 13 V aplicado al pin MCLR.
- PGC (*Program Clock*): se utiliza el pin RA6 como señal de reloj.

- PGD (*Program Data*): se utiliza el pin RA7 como señal de datos.
- GND: conexión del pin o pines VSS a masa.

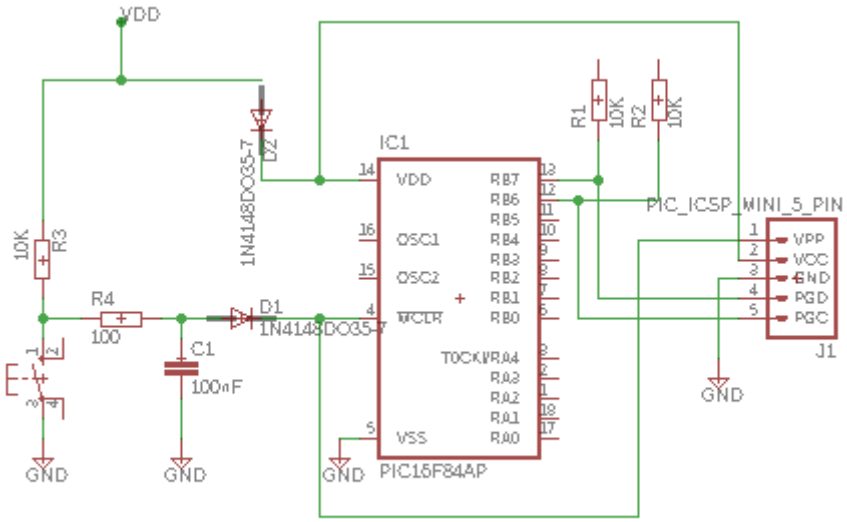


Ilustración 8: Ejemplo conexión programación ICSP

En la Ilustración 2 vemos un ejemplo de cómo se realizaría la conexión básica de la programación ICSP. Las 5 señales necesarias se introducen a través del conector J1 mediante un grabador.

El grabador se conecta a un ordenador a través de uno de los puertos disponibles (serie, paralelo o USB). Uno de los más conocidos, sencillos y económicos es el programador JDM que emplea el puerto serie RS-232. Compatible con el software ICPROG.

Sin embargo, los ordenadores modernos ya no suelen disponer de puertos series ni paralelos. Lo habitual hoy en día es el uso de grabadores que se conectan mediante USB. Estos programadores emplean un microcontrolador, grabado con un programa (firmware), para controlar el intercambio de datos entre el USB del ordenador y el PIC a grabar.

Probablemente los grabadores mediante USB más famosos sea la familia PICkit de Microchip. Siendo el PICkit 4 la última versión. El software empleado es MPLAB X IPE desarrollado por la misma compañía.

Ilustración 9: PICkit 4 de Microchip

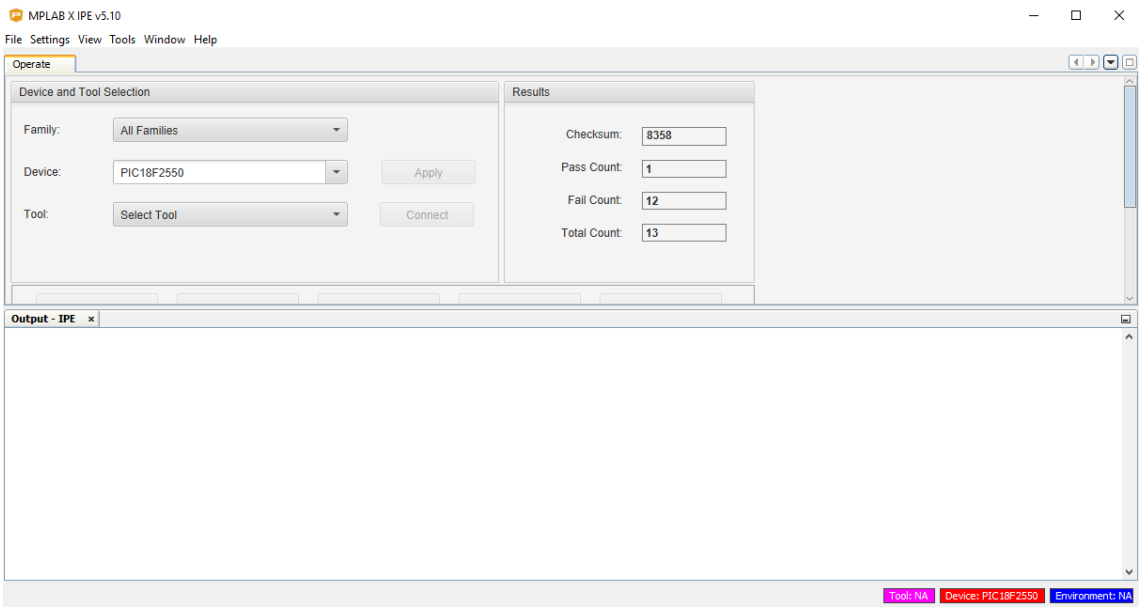


Ilustración 10: MPLAB X IPE

2.4.2.1. USBPICPROG

En este proyecto se implementará un grabador a través de conexión USB. Se trata de un grabador basado en el proyecto "free open source" llamado USBPICPROG. "Free open source" quiere decir que el diseño del hardware, firmware y software están totalmente disponibles sin ningún coste. Es compatible con una gran variedad de PIC desde las familias PIC10 hasta PIC30.

Está divido en tres partes:

- Hardware: contiene los componentes necesarios para conectar el puerto USB con el PIC a grabar a través de la conexión ICSP. Controlado mediante un PIC18F2550.
- Firmware: programa grabado en el PIC18F2550 que contiene los algoritmos de programación para todos los dispositivos PIC implementados.
- Software: aplicación para comunicarse con el hardware y el firmware.

USBPICPROG también comercializaba este producto a un precio muy razonable. Ahora solo está disponible para hacerlo uno mismo. Hay dos diseños diferentes, uno empleando componentes SMD (montaje superficial) y otro con componentes de agujeros pasantes.

(a) (b)

Ilustración 11: (a) USBPICPROG SMD, (b) USBPICPROG agujero pasante

Más adelante se hablara más en profundidad acerca del hardware y el software. Así como del proceso de grabación de los PIC utilizando este grabador. Se puede encontrar toda la información a través de su página web (www.usbpicprog.org).

3. Metodología de diseño

3.1. Periféricos empleados

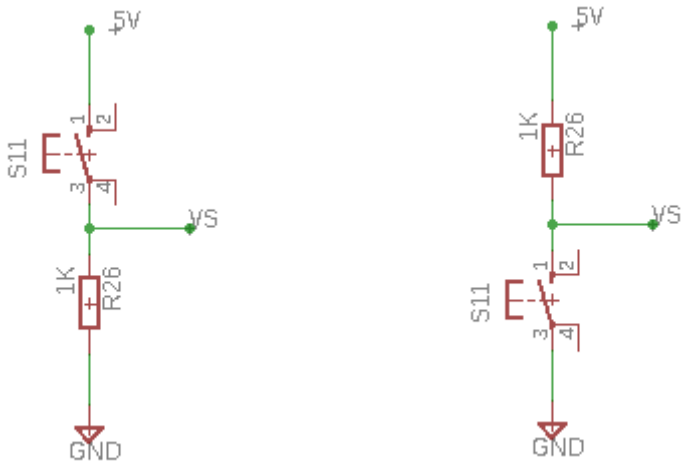
La entrenadora tiene que estar adaptada a las prácticas de la asignatura, por lo que es necesario establecer los periféricos mínimos que garantice que puedan llevarse a cabo todas las prácticas. Para ello se realiza un estudio de las necesidades de conectividad de dispositivos. Se emplearán los siguientes periféricos de entrada/salida:

- Interruptores y pulsadores:

Para la simulación de entradas digitales se emplearán varios interruptores y pulsadores. Estos permiten introducir un nivel lógico de "0" o "1" según la posición en que se encuentren. Hay dos maneras de realizar la lectura del estado de interruptores y pulsadores: mediante resistencias pull-up y pull-down.

Antes de nada, cabe mencionar que estas resistencias son resistencias normales, se llaman así por cómo están conectadas.

Las resistencias pull-up se conectan a la alimentación, de esta manera cuando el pulsador esté en reposo (no pulsado), VS tendrá un valor lógico alto "1". Cuando se pulse, VS se conectara a masa pasándose a un valor lógico bajo "0".



(a) (b)

Ilustración 12: (a) Resistencias pull-up, (b) Resistencias pull-down

Por el contrario, mediantes las resistencias pull-down, VS tendrá un valor lógico bajo cuando el sistema esté en reposo y un valor lógico alto cuando el pulsador está activado.

- Diodos led:

Los diodos led son perfectos para monitorizar el estado de los circuitos a los que son conectados mediante la emisión de luz. Son económicos y muy fáciles de conectar a los microcontroladores. Hay dos maneras:

Conectar el ánodo del led a la salida del microcontrolador y el cátodo a masa. De esta manera, al aplicar un "1" lógico en la salida del micro, en led se encenderá.

Conectar el ánodo a la alimentación y el cátodo a la salida del PIC. Habrá que aplicar un "0" lógico en la salida para que el led se ilumine.

De ambas maneras es necesario conectar los led a resistencias limitadoras para no quemarlos.

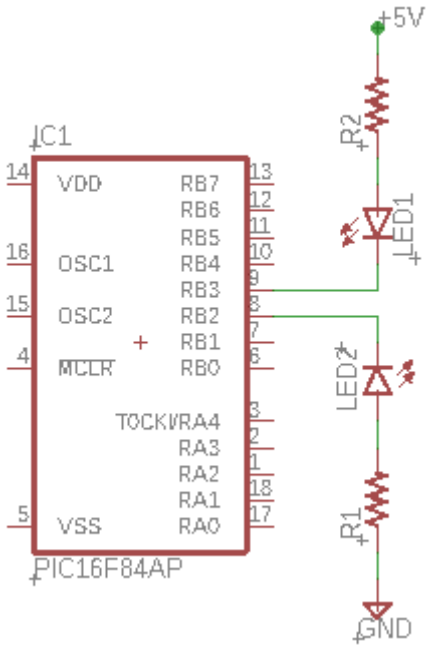


Ilustración 13: Maneras de conectar un led

- Display 7 segmentos:

Se trata de un periférico que permite representar valores numéricos. Cada display consta de 7 segmentos y punto decimal. Todos ellos se tratan de diodos led. Se pueden encontrar en dos configuraciones posibles según estén conectados los led: ánodo común o cátodo común.

Si son de ánodo común para encender un segmento hay que aplicar un valor lógico de "0" en la salida del PIC conectada en dicho segmento. En caso de que sean de cátodo común hay que introducir un "1".

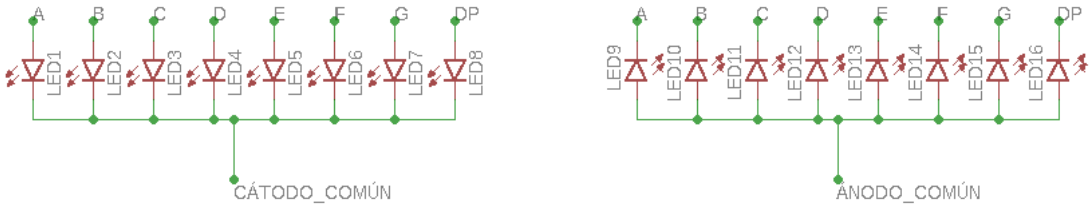


Ilustración 14: Constitución displays 7 segmentos

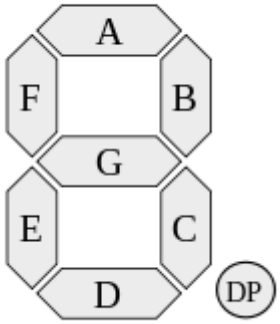


Ilustración 15: Identificación segmentos

- Potenciómetros:

Los periféricos antes mencionados son los que se usan en la asignatura. Pero ya que algunos modelos de PIC disponen de entradas analógicas se decide incluir también periféricos para poder simular estas entradas. El elemento más sencillo son los potenciómetros. Se trata de una resistencia que permite variar su valor desde 0 hasta el valor máximo nominal que posea. De todos los tipos de potenciómetros que existen usaremos los denominados rotatorios, los cuales son empleados en circuitos de pequeñas corrientes.

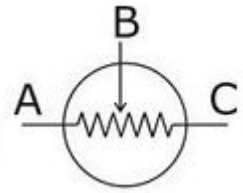


Ilustración 16: Patillas potenciómetro

Para que la resistencia sea variable siempre hay que conectar la patilla de en medio y la A o la C. Si conectásemos solo las patillas A y C, sin la B, el valor de la resistencia sería fijo y equivalente al valor nominal de la misma. Tienen una rosca que permite variarse, variando el valor de la resistencia.

3.2. Herramienta de diseño

En esta sección se va a explicar el software empleado para obtener el diseño final, es decir, el diseño PCB de la placa entrenadora. Existen multitud de softwares para el diseño de PCB ("*Printed Circuit Board*", Placa de Circuito Impreso), cada uno tienes sus ventajas e inconvenientes. El que se ha utilizado para la elaboración de este trabajo es la herramienta EAGLE ("*Easily Applicable Graphical Layout Editor*") desarrollado por Autodesk.

Es un programa que contiene un editor de diagramas esquemático donde los componentes pueden colocarse en múltiples hojas y conectarse entre sí mediantes puertos, además de

permitir la simulación de circuitos mediante métodos SPICE. Incluye un editor de placas de circuito impreso (PCB), al cual se puede acceder tras completar el diseño esquemático pulsando solamente un botón. Permite el enrutamiento automático para conectar todas las trazas de cobre, según las conexiones implementadas en el esquemático, de manera automática.

Ofrece la posibilidad de generar los archivos Gerber, archivos de taladores Excellon y PostScript (formatos de archivos estándar para la fabricación de PCB). Muchos fabricantes aceptan directamente los archivos .BRD de EAGLE.

Cuenta con una licencia gratuita pero está limitada: el tamaño máximo de la placa es de 100x80 mm, solo dos capas de trabajo (Bottom y Top) y el plano de esquemático solo puede ser de una hoja. Para estudiantes y profesorado, dispone de una licencia (para uso no comercial) que ofrece todos los beneficios de la licencia Premium: 16 capas de trabajo, 999 hojas para esquemáticos y el tamaño de las placas pueden ser hasta 4 m^2.

Se ha elegido este software porque es intuitivo y fácil de aprender y manejar, además de incluir una gran cantidad de librerías y componentes, y en caso de no incluir un componente deseado, puede crearse. Muchas empresas como Spark Fun ponen a disposición sus propias librerías. Cuenta con una comunidad muy extensa, donde se puede encontrar ayuda y tutoriales.

En la Ilustración 17: Editor de esquemáticos EAGLE vemos el editor de esquemáticos y en la Ilustración 18: Editor de PCB EAGLE el editor de PCB.

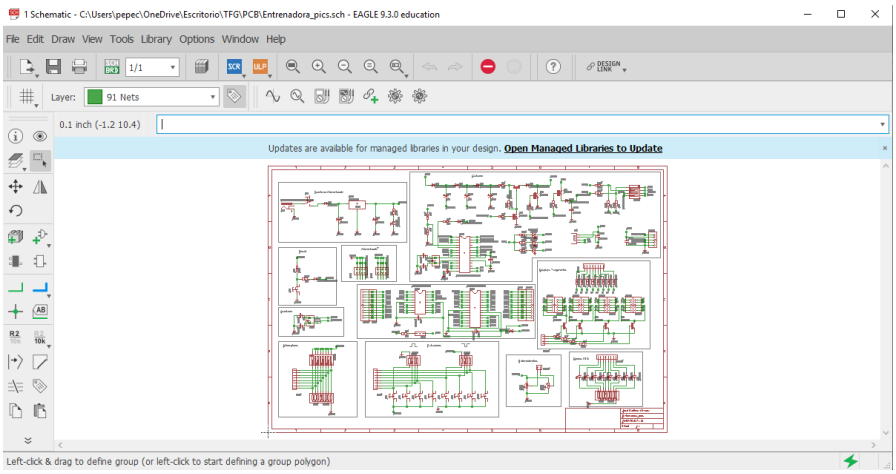


Ilustración 17: Editor de esquemáticos EAGLE

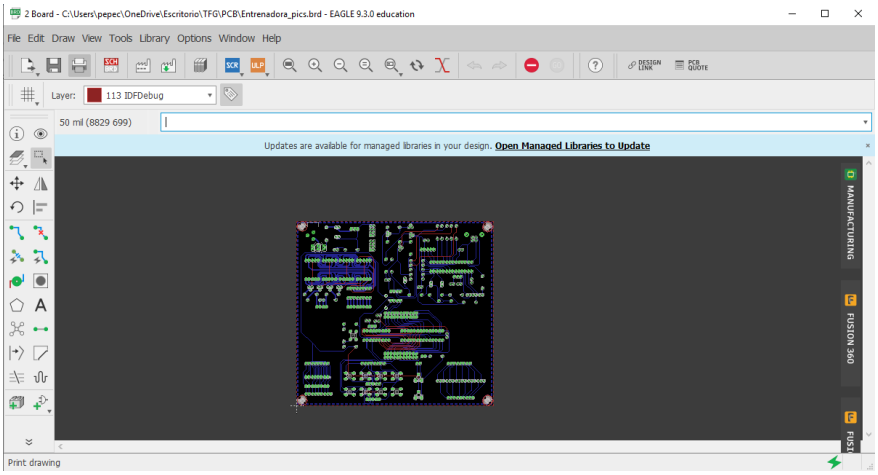


Ilustración 18: Editor de PCB EAGLE

3.3. Diseño esquemático

La entrenadora tiene una estructura modular, es decir, que está divida en módulos y todos ellos perfectamente identificados para que los estudiantes puedan localizar todas las partes. La entrenadora incluye los siguientes elementos:

- Fuente de alimentación.
- Conectores para alimentación y masa.
- Zócalos para integrados de 18 y 28 pines.
- Pines para colocar el oscilador.
- Pulsador de reset.
- 1 interruptor DIP de 8 posiciones, activo por transición 0-1-0.
- 8 pulsadores, 4 de ellos activos por transición 0-1-0 y los otros 4 activos por transición 1-0-1.
- 8 diodos led.
- 4 displays de 1 multiplexados.
- 2 potenciómetros para simular entradas analógicas.
- Un cargador de EEPROM (grabador) mediante USB compatible con una gran variedad de microprocesadores PIC.

Se puede acceder a todas las líneas de entrada/salida de los microprocesadores instalados, por lo que es posible conectar cualquier periférico a cualquier pin de E/S. El plano puede encontrarse en el *Anexo I. Planos*.

A continuación se pasa a explicar con más detalles todos los módulos que componen la entrenadora:

3.3.1. Fuente de alimentación

Se decide emplear una fuente de alimentación externa de 12V y 1A para alimentar toda la placa (excepto el grabador que es a través del puerto USB). Los 12 voltios son introducidos por un conector Jack y mediante un regulador de tensión LM7805 se

obtienen los 5 voltios necesarios para alimentar los microprocesadores y el resto de elementos.

Además mediante un interruptor On/Off podemos controlar la conexión de la alimentación. También es posible visualizar el estado de la fuente a través de un diodo led. El diodo D2 protege la fuente en caso de polaridad inversa.

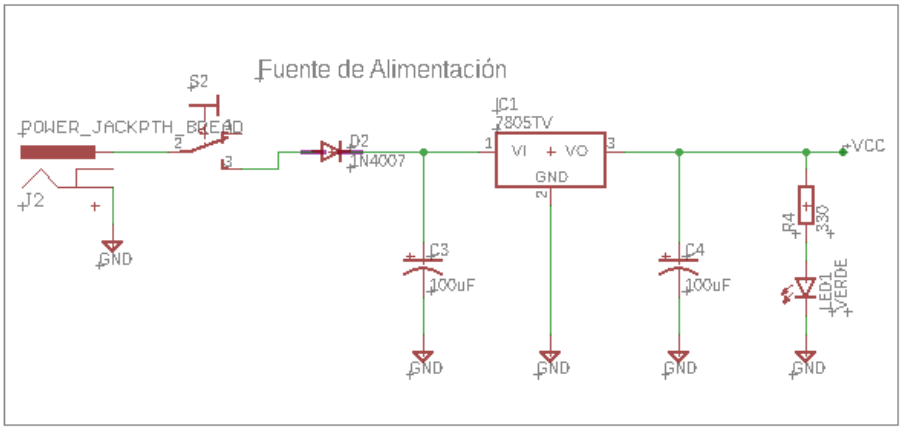


Ilustración 19: Fuente de alimentación

Dado que con 1A es suficiente para alimentar la entrenadora y si fuese necesario para alimentar otros módulos ajenos a la placa, es posible acceder a los 5 voltios y a masa mediante los conectores JP3 y JP6.

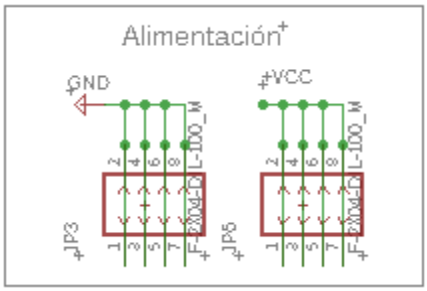


Ilustración 20: Conectores para alimentación

3.3.2. Zócalos para microcontroladores

La parte más importante de la entrenadora son los zócalos de 18 y 28 pines donde se instalan los microprocesadores PIC de las familias 16F y 18F. Estos están directamente conectados a la alimentación, a masa, a la señal de reloj del oscilador y al pulsador de reset. Para no limitar la entrenadora, el resto de patillas, que corresponden a los puertos de entrada/salida, no estarán conectados a nada, por lo que es posible realizar cualquier conexión que se desee. Para ello se puede acceder a estas patillas a través de los conectores JP4 y JP9.

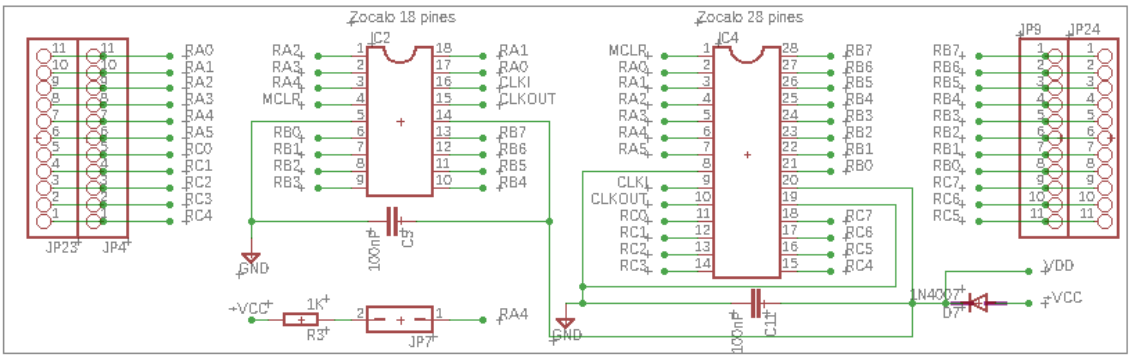
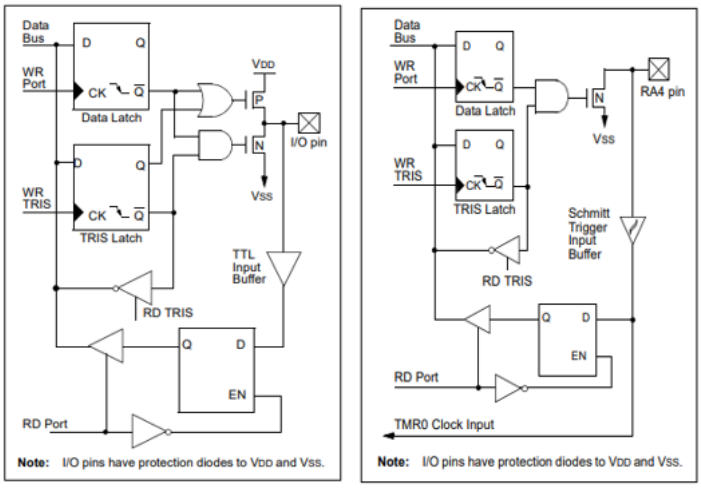


Ilustración 21: Conexiones de los Microcontroladores

El diodo D7 está para proteger la fuente de alimentación durante el proceso de grabación en caso de que se esté usando la fuente de alimentación externa y la alimentación del cargador de EEPROM a la vez. El cargador de EEPROM también posee su propio diodo, de esta manera una fuente no interfiere con la otra.

3.3.2.1. RA4/TOCKI



(a) (b)

Ilustración 22: (a) Estructura Puerta A: RA0:RA3 PIC16F84A, (b) Estructura Puerta A: RA4 PIC16F84A

En algunos PIC, como el PIC16F84A, la patilla RA4/TOCKI es diferente a las demás de la puerta A. Esto puede verse en la Ilustración 22: (a) Estructura Puerta A: RA0:RA3 PIC16F84A, (b) Estructura Puerta A: RA4 PIC16F84A. En las patillas RA0:RA3 cuando el bit de TRIS sea “1” la salida del biestable TRIS será “1” y estarán configuradas como entradas.

Ahora bien, cuando el bit de TRIS sea “0”, estas patillas estarán configuradas como salidas. Cuando esto ocurra, si la salida del biestable de datos es “0”, el mosfet de canal P no conducirá y el mosfet de canal N si lo hará, conectando los pines a masas, siendo la salida de los pines “0”. En caso de que la salida del biestable de datos sea “1”, será el mosfet de canal P el que conducirá, conectando los pines a la alimentación.

Lo mismo ocurre para la patilla RA4 en caso de que sea configurada como entrada. La cosa cambia cuando queremos que actúe como salida. En este caso, cuando la salida del biestable de datos sea “0”, el mosfet conducirá y conectara el pin a masa. El problema está cuando la salida del biestable es “1”, ya que el mosfet no conducirá y el pin queda flotante.

Para solucionar esto hay que conectar a la patilla RA4 una resistencia pull-up, de esta manera es posible acceder a los 5 V.

Sin embargo, si dejásemos esta resistencia siempre conectada, cuando deseemos utilizar esta patilla como entrada, está siempre se encontrará a “1”. Esto puede no interesarnos siempre y además no todos los PIC cuentan con esta peculiaridad. Así que se decide añadir el jumper JP7 para poder elegir la configuración que nos interese. Si el jumper esta quitado se empleara como entrada, y si esta puesto como salida.

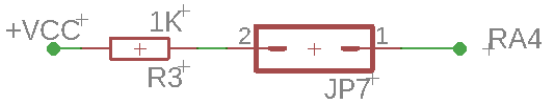


Ilustración 23: Jumper Salida/Entrada RA4

3.3.3. Oscilador

El circuito es muy simple. Se empleará un cristal de cuarzo conectado a los pines OSC1/CLKIN y OSC2/CLKOUT de ambos zócalos.

Además es posible trabajar con cristales de diferentes frecuencias ya que estos no van conectados directamente a la placa, sino que gracias a unos pines torneados es posible seleccionar el cristal que más nos conviene según la aplicación.

Es muy importante que oscilador esté los más cerca del PIC posible, ya que si se coloca demasiado alejado será una importante fuente de ruido.

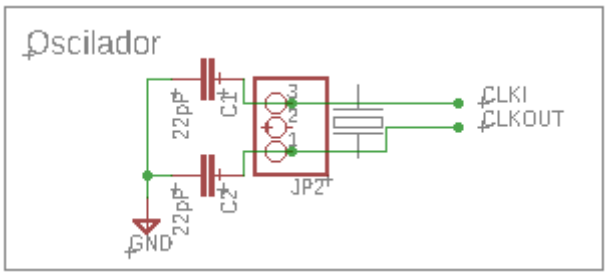


Ilustración 24: Diagrama de conexiones Oscilador

3.3.4. Reset

El reset no es más que un pulsador conectado a una resistencia pull-up. Conectado a la patilla de MCLR de ambos zócalos. Consta también de un diodo para evitar que el voltaje de programación proveniente del cargador de EEPROM afecte a la fuente de alimentación.

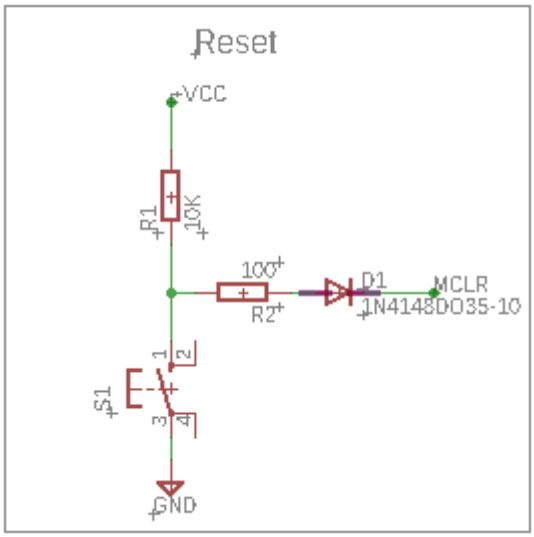


Ilustración 25: Diagrama de conexiones Reset

3.3.5. Entradas digitales

3.3.5.1. Interruptores

La placa consta de 8 interruptores deslizantes en formato DIP. Inicialmente tienen un valor lógico de 0, ya que están conectados a resistencias pull-down. Cuando son accionados pasan a un estado lógico de 1. Se puede acceder a estos interruptores a través del conector JP1.

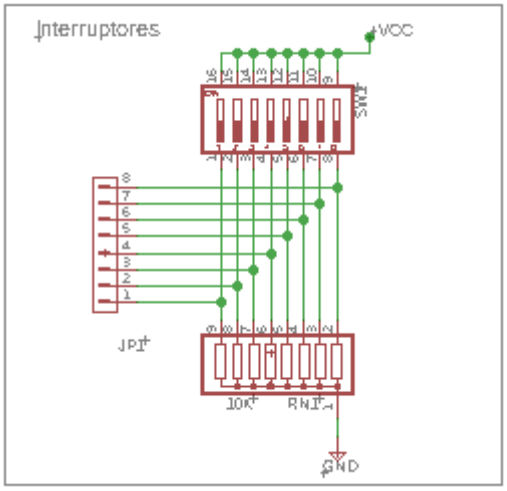


Ilustración 26: Diagrama de conexiones Interruptores

3.3.5.2. Pulsadores

8 pulsadores, de los cuales 4 están conectados a resistencias pull-down, por lo que tienen un valor lógico de 0 en reposo, cuando son pulsados pasan a un valor lógico de 1". Los otros 4 son lo contrario, están conectados a resistencias pull-up, es decir que pasan a un valor lógico de 0 al ser pulsados. Accesibles mediante el conector JP5.

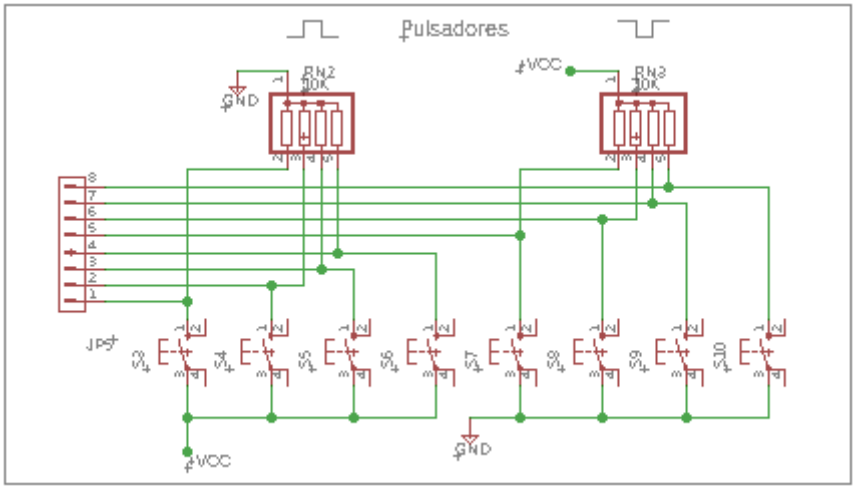


Ilustración 27: Diagrama de conexiones Pulsadores

3.3.5.3. Rebotes

Los interruptores y pulsadores son mecánicos, esto ocasiona que pueda aparecer un efecto muy problemático y que hay que tener cuenta su tratamiento si se da el caso. Este efecto es conocido como "rebote".

La conexión y desconexión depende de un muelle, el cual controla la apertura y el cierre del contacto. Pero este proceso no se realiza inmediatamente, sino que tenemos un tren de impulsos de conexión/desconexión hasta que se estabiliza. Para otras aplicaciones fuera de los microcontroladores no hay problema, pero dada la velocidad de las señales que tratan esto puede suponer un problema.

Se puede solucionar tanto como por hardware o por software. En el primer caso basta con añadir un condensador conectado a masa en paralelo con los interruptores o pulsadores. Por software, tras leer el estado del pulsador hay que aplicar un retraso de unos 20 milisegundos para que la salida se estabilice y volver a comprobar el estado del pulsador, para asegurarse de que este pulsado y no sea un rebote.

Este es un problema que no aparece en los simuladores, por eso es tan importante no limitarse solo al uso de estos.

3.3.6. Entradas analógicas

Se incluyen dos potenciómetros conectando sus extremos a masa y a la alimentación. De esta manera se puede obtener un rango de tensión que va desde los 0 voltios hasta los 5, según la variación de la resistencia. Actuando así como un divisor de tensión. Entre sus usos se puede citar la conversión analógica-digital. Podemos acceder a ellos a través de JP10.

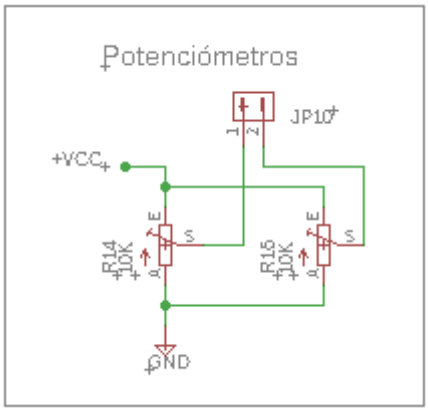


Ilustración 28: Diagrama de conexiones Potenciómetros

3.3.7. Salidas digitales

3.3.7.1. Diodos led

Se disponen de 8 diodos led de color rojo idóneos para representar el estado lógico de diversas aplicaciones, así como, de simular el encendido y apagado de dispositivos más complejos.

Las salidas del PIC se conectarán a los ánodos de los diodos led y los cátodos están conectados a masa, por lo que para encenderlos habrá que introducir un valor lógico de 1 y para apagarlos un 0. Se puede acceder a los diodos led mediante el conector JP13.

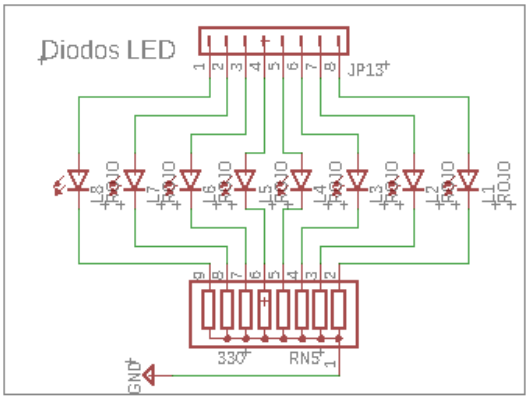


Ilustración 29: Diagrama de conexiones Diodos LED

3.3.7.2. Displays 7 segmentos

La entrenadora incluye 4 displays de 7 segmentos de cátodo común multiplexados. No van conectados a la placa, sino que se podrán colocar y retirar gracias a unos pines torneados, por lo que es posible cambiarlos en caso de que fallen sin tener que desoldarlos. Mediante el conector JP12 tenemos acceso a los segmentos, que al estar multiplexados, estarán conectados a todos los displays, y a través del conector JP11 se accede a los transistores, pudiéndose controlar los displays secuencialmente.

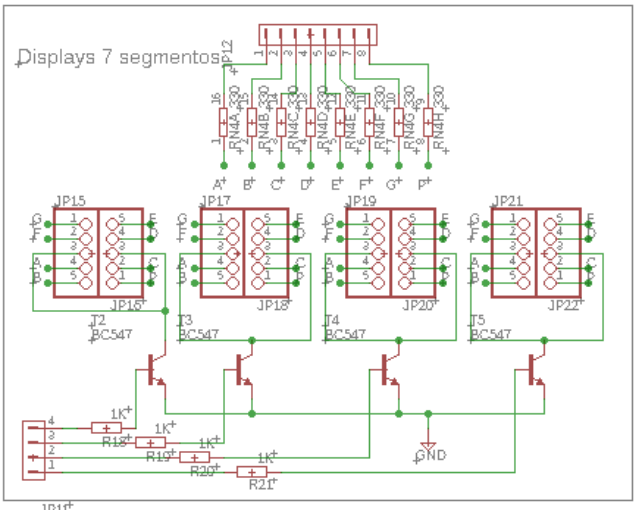


Ilustración 30: Diagrama de conexiones Displays 7 Segmentos

3.3.7.1. Multiplexación

El número de puertos de entrada/salida de los PIC son limitados, si quisiéramos conectar 4 displays necesitaríamos emplear 8 patillas, si incluimos el punto, por cada display, es decir, 32 patillas. Esto es imposible. Para ello se recurre a la técnica de la multiplexación. Con la multiplexación se conectan los 4 displays en paralelo y un transistor en la base de cada display. Entonces con 12 líneas es suficiente para controlarlos. Esta técnica nos permite reducir el hardware necesario, sin embargo, conlleva una complicación en el software.

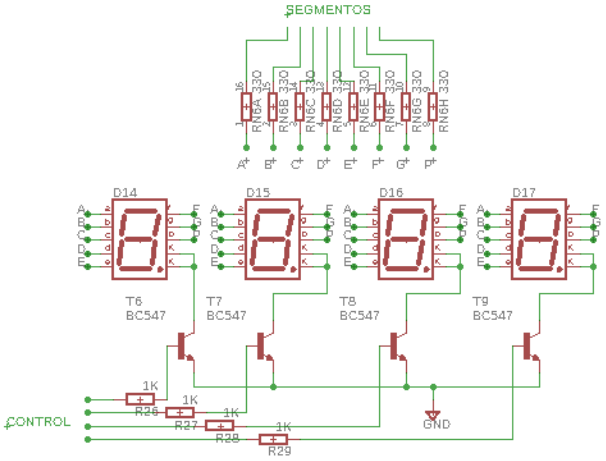


Ilustración 31: 4 displays multiplexados

En la imagen de arriba vemos un ejemplo de 4 displays multiplexados. Esta técnica consiste en controlar secuencialmente los displays mediante los transistores, esto lo maneja el puerto al que va conectado mediante 4 bits (“control” en la imagen), mientras se envía por el puerto conectado a los segmentos los datos correspondientes (“segmentos” en la imagen). Esto hay que realizarse con la velocidad suficiente para engañar al ojo

humano, de manera que parezca que los displays están mostrando los valores a la vez. Se ha calculado experimentalmente que mostrar cada digito durante 5 milisegundos es suficiente.

Los transistores en este caso actúan como si se tratase de un interruptor, inicialmente están abiertos, es decir, no circula la corriente entre el colector y el emisor. Para activar los transistores (interruptor cerrado) es necesario aplicar un "1" lógico en la base del mismo, de esta manera entrará en saturación, permitiendo el paso de la corriente entre colector y emisor, conectando el común del display a masa. Como los displays son de cátodo común, y al final los displays no son más que diodos led, para activar cada segmento hay que aplicar un valor de "1", de esta manera circulará la corriente desde el ánodo hasta el cátodo, iluminándose dicho segmento.

En el proceso de multiplexación encontramos otra diferencia entre el uso de los simuladores, como el Virtual Breadboard, y las herramientas de hardware, puesto que en el primer caso no es necesario dejar un tiempo encendido cada display, basta con energizar cada display secuencialmente, ni es necesario el uso de transistores. Esto implica que la forma de programarlos es diferente en ambos casos.

3.3.8. Cargador de EEPROM

El cargador de EEPROM es una de las características más atractivas que posee la entrenadora. Ya que al estar implementado en la propia entrenadora, no es necesario disponer de un cargador externo para cada puesto de trabajo, eso supone un importante ahorro económico.

El hardware es muy sencillo, básicamente consta de las siguientes partes:

- Un microcontrolador PIC18F2550, precargado con el firmware, encargado de controlar el proceso de grabación.
- Un multiplicador de voltaje, el cual convierte los 5 voltios provenientes del puerto USB a los 12 voltios que son necesarios para la programación de los PIC.
- Un conector para programar el propio PIC18F2550 en caso de que se desprograme o para grabar por primera vez el firmware.
- Un interruptor DIP de 4 posiciones para permitir el paso de las señales ICSP para programar los PIC. Estas señales son:
    - PGC: señal de reloj.
    - PGD: señal de datos.
    - VDD: voltaje de alimentación del PIC a programar.
    - VPP: voltaje de programación.
    - La masa no hace falta ya que ya van conectadas.
- 3 diodos led con diferente color y funcionalidad:
    - Led1 (rojo): indica el proceso de escritura.
    - Led2 (amarillo): indica el proceso de lectura.
    - Led3 (verde): indica que el programador está conectado.

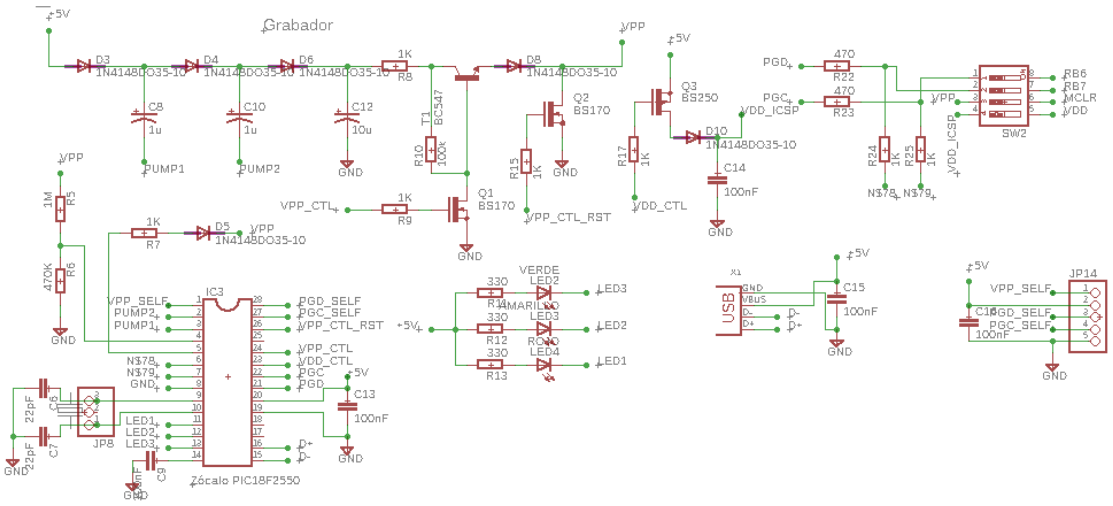


Ilustración 32: Hardware grabador

3.4. Diseño PCB

Una vez realizado el esquemático se pasa a la realización del circuito impreso. Se busca que todos los componentes estén bien identificados y que presente la información necesaria para que sea lo suficientemente intuitiva, de manera que los estudiantes no encuentren problemas a la hora de realizar las conexiones pertinentes. Se diseña de manera que este divida en módulos delimitados por líneas. La distribución de los componentes en la placa puede verse en la siguiente imagen:

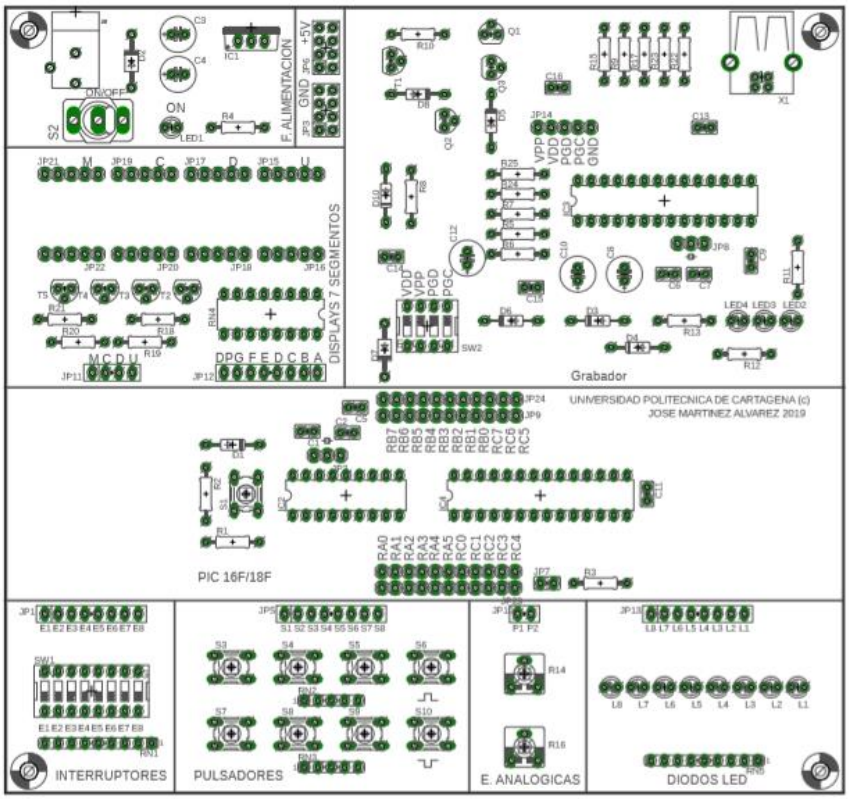


Ilustración 33: Distribución de componentes

Se ha intentado diseñarla con el menor tamaño posible, ya que en muchos fabricantes el tamaño de las placas a fabricar es limitado, pero siempre dentro de la comodidad a la hora de trabajar con ella. El resultado final es una PCB de 160x150 mm.

A la hora de enrutar, aunque EAGLE ofrezca la posibilidad de hacerlo de manera automática, se ha decido hacerlo manualmente, puesto que el proceso automático puede llevar demasiado tiempo y una vez realizado no quedar satisfecho con el resultado. La anchura de las trazas de cobre es de 0,508 mm, esto permite que pueda pasar a través de los pines de los componentes. Se ha empleado tanto la cara TOP (cara superior) como la BOTTOM (capa inferior) a la hora de realizar las trazas. Ambas caras poseen un plano de masa, esto puede ayudar a reducir la impedancia y por tanto el ruido.

Tras realizar el diseño PCB, el siguiente paso es generar los archivos Gerber. Este formato contiene la información necesaria para la fabricación de circuitos impresos. Esta información es la capa inferior y superior de cobre, la máscara de soldadura, la serigrafía y los taladros a realizar. Antes de enviar los Gerber a un fabricante, es recomendable acudir a un visor de archivos Gerber para verificar si se han generado correctamente. Muchos de los fabricantes de PCB disponen en sus respectivas páginas web de visores. Una empresa conocida de creación de prototipado rápido y producción de bajo volumen es “pcbway”, si acudimos a su página podemos visualizar los archivos generados:

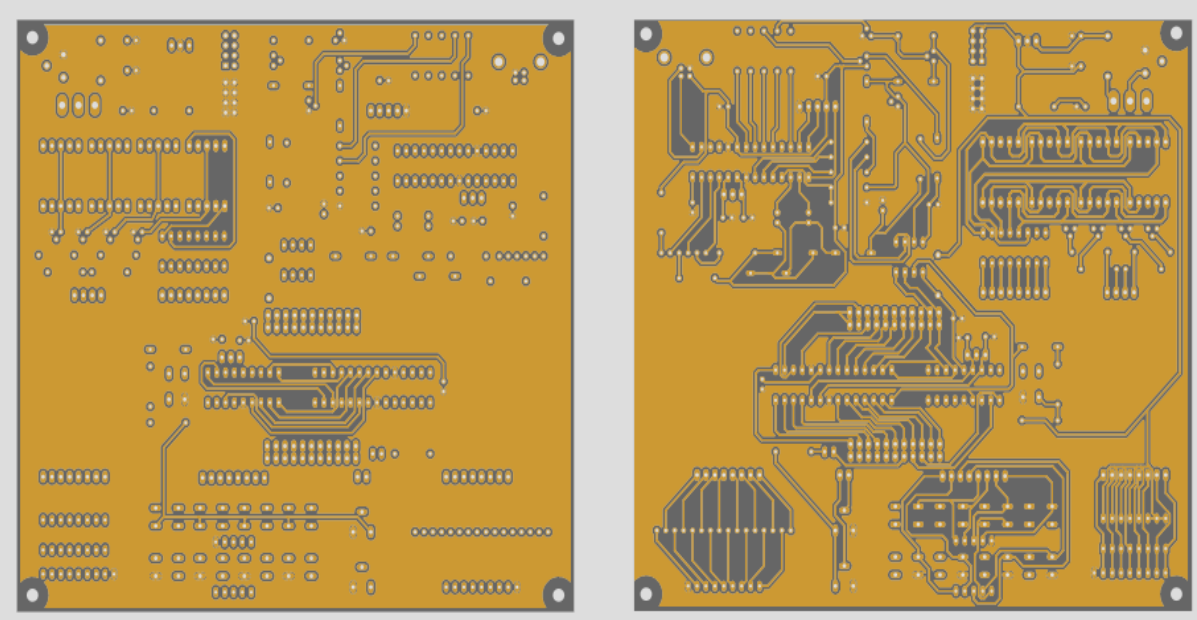


Ilustración 34: Placa en el visor Gerber

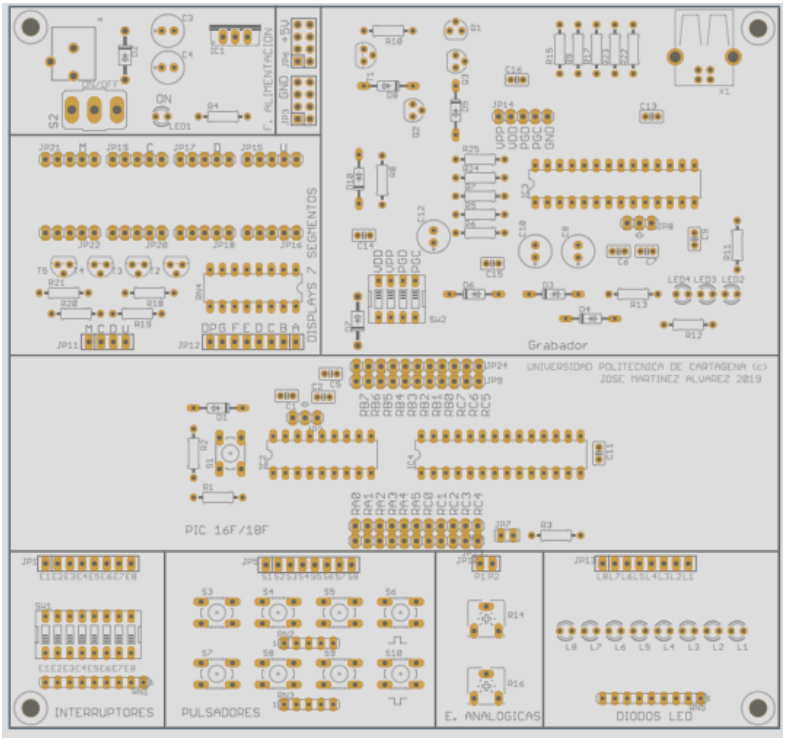


Ilustración 35: Serigrafía en el visor Gerber

Tras comprobar los archivos Gerber ya estaría listo para enviárselos al fabricante seleccionado. Los planos del diseño PCB puede verse con más detalle en el *Anexo I.II: PCB TOP* y en el *Anexo I.III: PCB BOTTOM*.

3.5. Prototipo y problemas de diseño

Inicialmente se realizó un primer diseño PCB, cuyo esquemático es prácticamente igual que el explicado anteriormente, los cambios con respecto al diseño final se comentan más adelante.

En la Ilustración 36: Diseño inicial PCB podemos ver la distribución de componentes del diseño inicial:

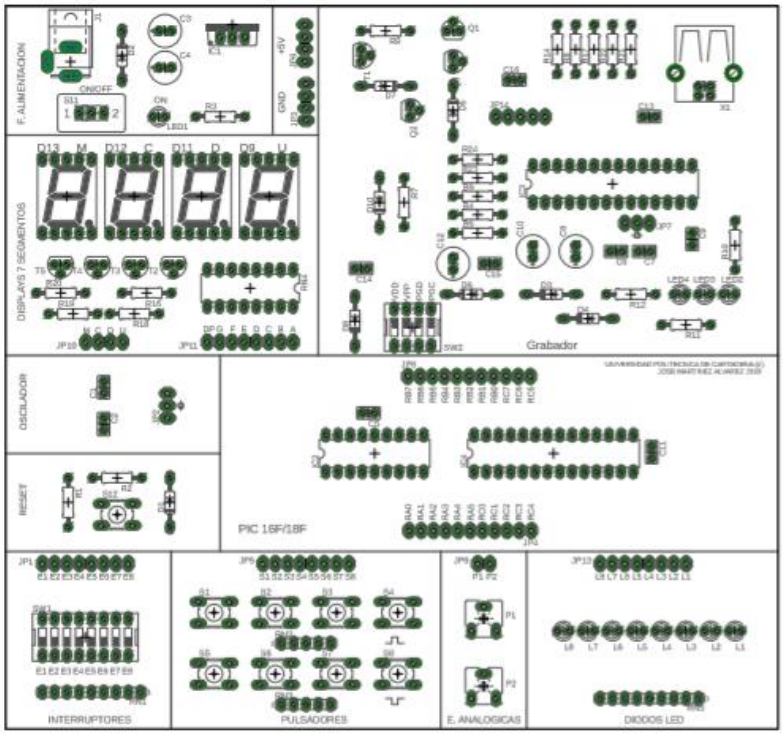


Ilustración 36: Diseño inicial PCB

Tras completar el diseño se pasó a implementar un prototipo realizado por el personal de la UPCT en los laboratorios de electrónica del edificio SAIT (Servicio de Apoyo a la Investigación Tecnológica). El prototipo se fabricó empleando una Microfresadora LPKF ProtoLaser S, esta herramienta elimina de forma selectiva las capas de cobre en las placas de circuito impreso de una y dos caras, creando canales aislantes que delinean con precisión las trazas y pads conductores. Además de realizar también los taladros necesarios.

Ilustración 37: Microfresadora LPKF ProtoLaser S

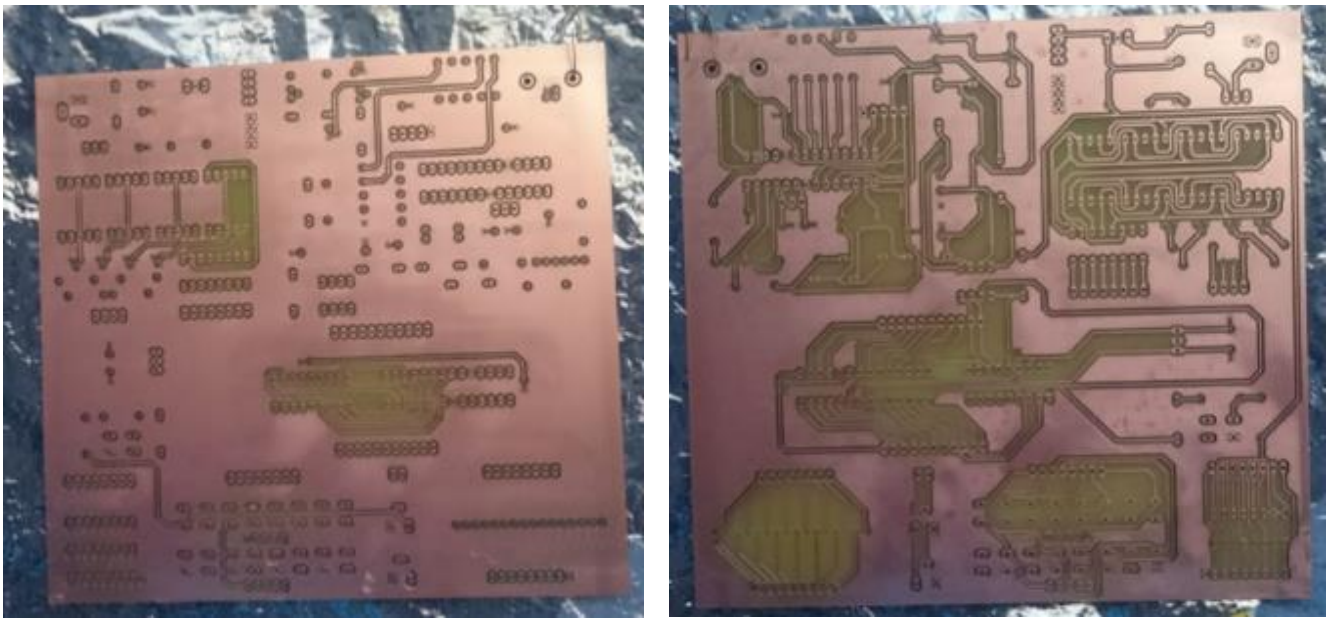


Ilustración 38: PCB realizada en el laboratorio

Tras soldar los componentes, el resultado final del prototipo puede verse en la Ilustración 39: Prototipo.

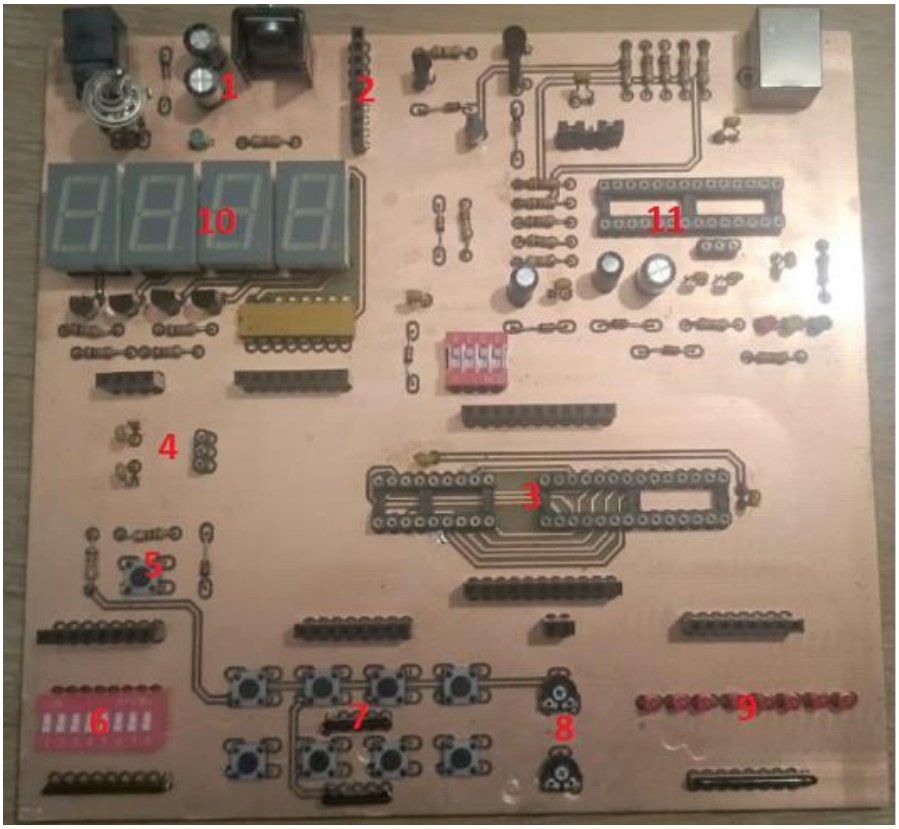


Ilustración 39: Prototipo

Donde:

IDENTIFICACIÓN	ELEMENTO
1	Fuente de alimentación
2	Conectores para alimentación
3	Zócalos para PIC de 18 y 28 pines
4	Pines torneados para el oscilador
5	Pulsador de reset
6	Interruptores DIP
7	Pulsadores
8	Potenciómetros
9	Diodos led
10	Displays 7 segmentos
11	Grabador

Tabla 3: Módulos entrenadora

Tras la realización se pasa a realizar las pruebas pertinentes para comprobar su funcionalidad, comprobando la continuidad de los componentes, así como la resolución de las prácticas (esto puede verse en la siguiente sección). Aunque el prototipo cumple su cometido de forma satisfactoria, se han realizado una serie de cambios para solucionar problemas encontrados, algunos de los cuales pueden afectar al correcto funcionamiento de la entrenadora, y otros que simplemente son para aumentar la funcionalidad. Los cambios realizados son los siguientes:

- Como la entrenadora tiene fines didácticos, para que los estudiantes pudieran identificar de forma clara los componentes, se puso en primera instancia el oscilador demasiado alejado de los zócalos. Aunque una vez realizado todas las conexiones pertinentes no hay problema alguno, si se desconecta algún cable, la aplicación que se esté probando se verá afectada por el ruido. Por lo que se ha colocado el oscilador lo más pegado posible al zócalo de 18 pines.
- Como se ha mencionado es necesario añadir una resistencia pull-up a la patilla RA4 para poder emplearla como salida. Pues bien, esto es uno de los errores más comunes en el diseño de aplicaciones con microcontroladores, y no se tuvo en cuenta en el diseño del prototipo.
- Se ha sustituido el diodo de la fuente de alimentación por un diodo 1N4007, ya que el diodo 1N4148 que se puso en primer lugar falló al poco tiempo de haberse completado el prototipo.
- El programa empleado (EAGLE) para el diseño de la PCB no realizó bien los taladros del conector Jack de alimentación, eran demasiado pequeños para poder introducir el conector. Con este software no es posible aumentar el tamaño de los taladros, por lo que se ha optado por descargar una librería de Spark Fun con diversos conectores con el tamaño de los taladros correctos y en la lista de componentes seleccionar específicamente el conector asociado. Otra opción era emplear la capa “Milling” que es donde EAGLE genera los taladros de este conector y con una nota avisar al fabricante para que tengan en cuenta esta capa a la hora de realizar el PCB. Pero se

ha decidido la primera opción porque podemos asegurarnos de que tienen el tamaño necesario.

- Los headers asociados a los puertos de entrada/salida colocados son de 1x11, es decir 1 fila y 11 posiciones. Esto puede ser un poco escaso si es necesario colocar dos periféricos en un mismo puerto, así que se ha decido incluir otro header de 1x11, otra posibilidad era cambiarlo por dos de 2x11 pero este es un componente raro no encontrado en la distribuidora de componentes escogida. Lo mismo ocurre con los conectadores de alimentación y masa, inicialmente eran de 1x4 y se han cambiado por 2x4 para poder alimentar un mayor número de módulos.
- El interruptor On/Off pedido en la distribuidora de componente no se encontraba en stock por lo que se optó por colocar uno disponible en el laboratorio, debido al buen funcionamiento de este y ante la incertidumbre del si el seleccionado en un principio funcionaria correctamente, se ha cambiado en el diseño este interruptor por el empleado en el prototipo.

El diseño final es el que ha podido observarse a lo largo de este trabajo y que se puede encontrar en el *Anexo I: "Planos".*

3.6. Software programador

Como se ha visto en la parte del marco teórico, el software empleado para la programación es usbpicprog. Aquí se hará una descripción tanto del firmware como del software utilizado en la grabación de los PIC.

3.6.1. Firmware

A la hora de escribir el código para el PIC18F2550 se ha utilizado Microchip C18, el cual es una aplicación totalmente integrada en MPLAB IDE que permite el desarrollo y depuración de programas, basado en el PicDem de Microchip.

El firmware es el encargado, a través de diversas señales, de permitir el paso de las señales de ICSP controlando la saturación de los mosfet Q1, Q2 y Q3. Así como de cargar y descargar los condensadores, alternando las señales PUMP1 y PUMP2, obteniendo así los 12 voltios necesarios para la programación.

El firmware no va precargado en el microprocesador PIC18F2550, sino que es necesario cargarlo mediante otro cargador de EEPROM, ya sea a través del conector "JP14", o retirando el PIC del zócalo.

3.6.2. Software

La aplicación se comunica con el hardware para grabar el archivo .hex en dispositivo deseado. El software está disponible para Linux, Windows y Mac. Además de controlar el hardware, ha sido implementado el bootloader PICDem FS USB de Microchip, de esta manera es posible actualizar el propio firmware de usbpicprog.

Con el software abierto y la placa conectada al PC a través del cable USB saldrá la siguiente pantalla:

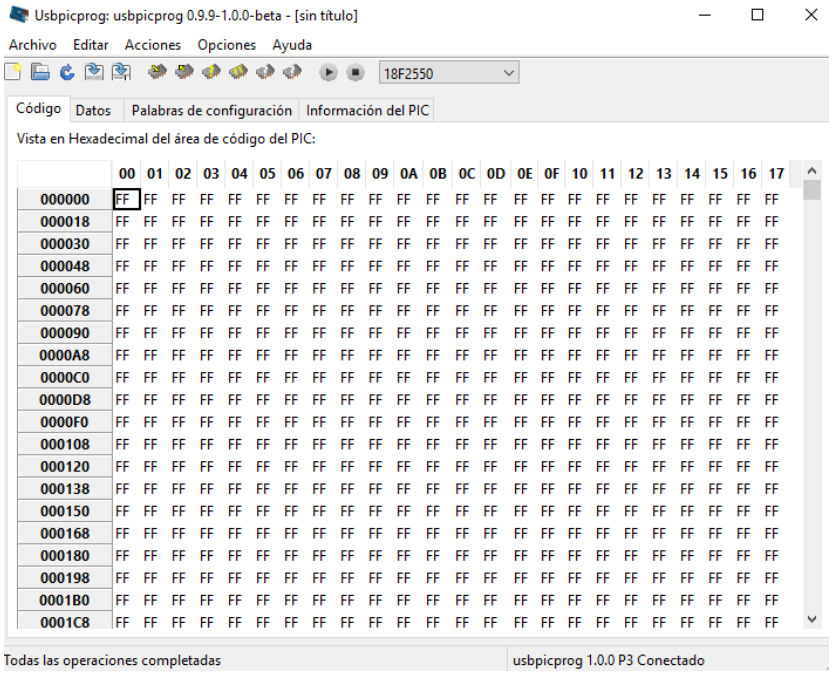


Ilustración 40: Ventana principal del software usbpicprog

Se puede ver que la aplicación ha detectado la placa en la parte inferior. Si en los zócalos se encuentra algún PIC y el interruptor DIP está activado, al pulsar “auto detectar” o “F5” el software detectará automáticamente el PIC instalado.

En la ventana “Palabras de configuración” podemos cambiarla según nos interese. Esta ventana será diferente según la versión del PIC que queramos grabar. Por ejemplo, para el PIC16F84A, esta ventana mostraría lo siguiente:

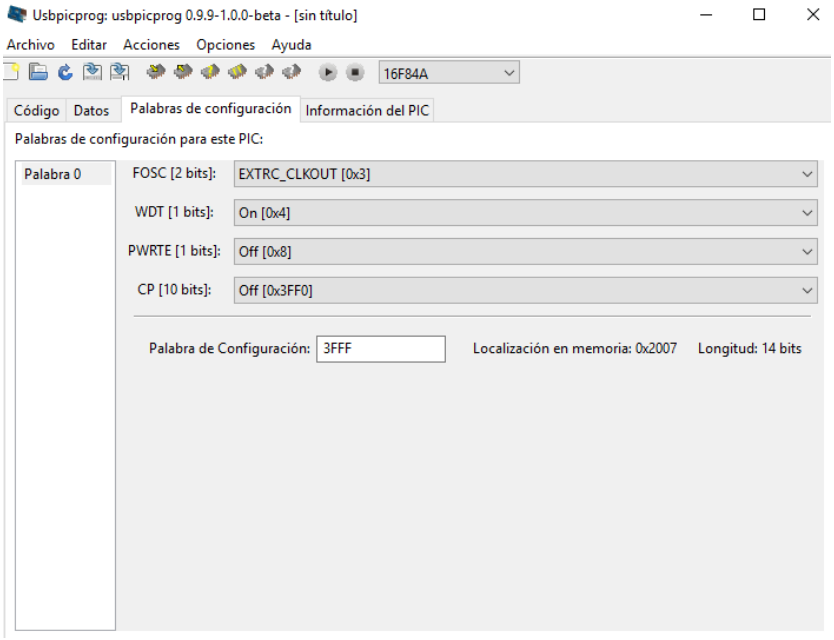


Ilustración 41: Ventana "Palabras de configuración" del software usbpicprog

Donde:

FOSC: selección del oscilador:

- Oscilador LP [0x0]
- Oscilador XT [0x1]
- Oscilador HS [0x2]
- Oscilador RC [0x3]

WDT: habilitar el perro guardián:

- Off [0x0]: WDT habilitado.
- On [0x4]: WDT deshabilitado.

PWRTE: habilitar el Power- Up Timer:

- On [0x0]: Power-Up Timer habilitado.
- Off [0x8]: Power-Up Timer deshabilitado.

CP: protección del código:

- All [0x0]: protección del código habilitado.
- Off [0x3FF0]: memoria del código protegido.

De la misma manera la ventana "Información del PIC" mostrara la información más importante del PIC seleccionado. Para el mismo caso anterior:

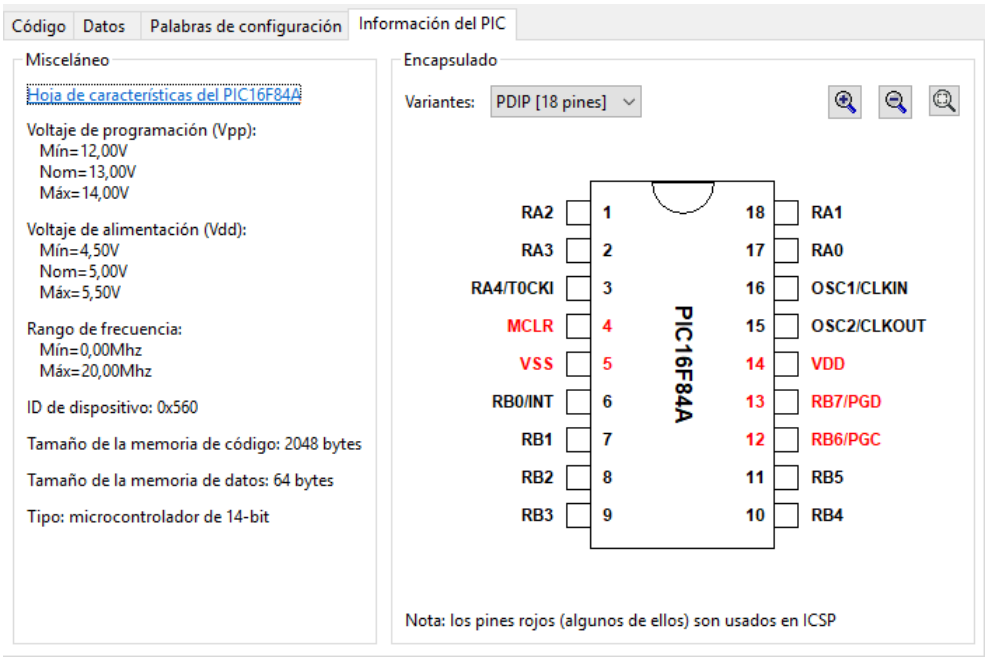


Ilustración 42: Ventana "Información del PIC" del software usbpicprog

3.6.3. Proceso de grabación

El primer paso es conectar la placa al ordenador mediante el conector USB:

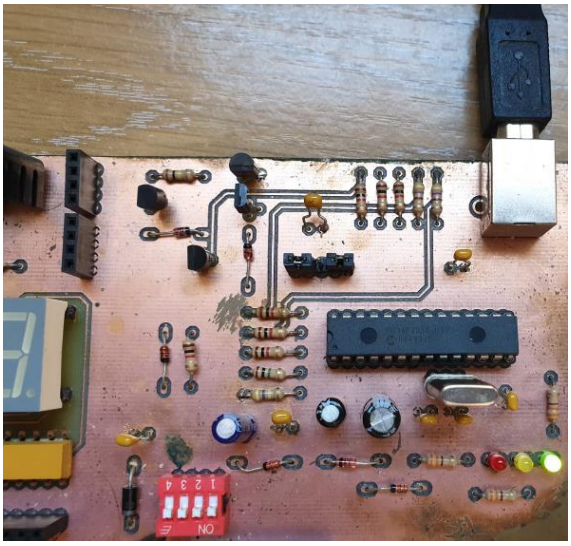


Ilustración 43: Programador conectado al USB

El led verde indica que el ordenador ha detectado el firmware y el programador está conectado. Los cuatro interruptores del interruptor DIP tienen que estar a ON.

Lo siguiente es abrir el software de USBPICPROG mostrándose las ventanas enseñadas anteriormente. Tras esto hay que abrir el archivo .hex que se quiera grabar en el PIC deseado, para ello hay que seleccionar en el menú “Archivo” >> “Abrir” y seleccionar el nombre del archivo deseado, o usando la barra de herramientas, pinchando en el segundo icono.

Y por último programar “Acciones” >> “Programar”, mediante “F7” o pulsando sobre el icono “programar” de la barra de herramientas. El proceso de programación puede verse en las siguientes imágenes:

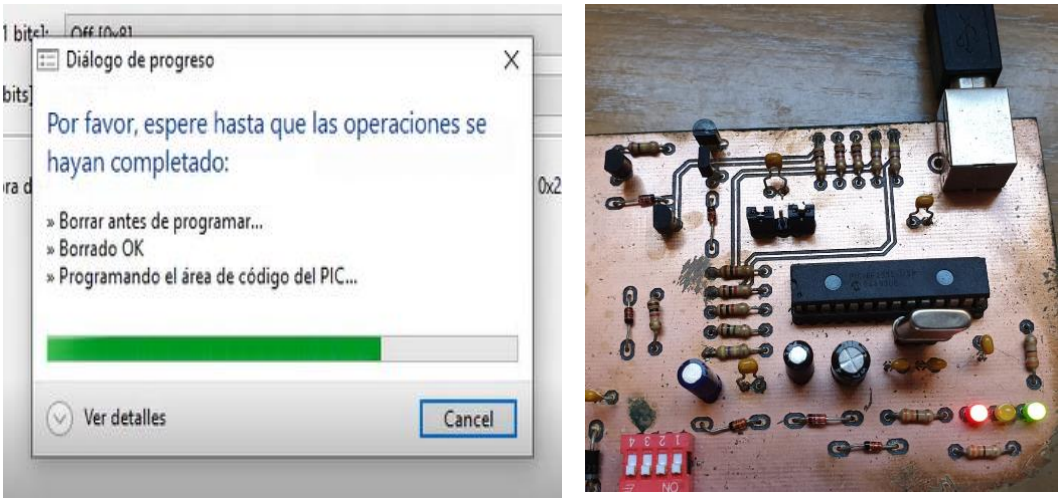


Ilustración 44: Proceso de grabación

Ilustración 45: Proceso de grabación

Durante la escritura y programación el led rojo se encendera, del mismo modo lo hará el led amarillo cuando lea la memoria del PIC a grabar o durante el proceso de verificación. Si el proceso de grabación ha sido satisfactorio aparecerá la siguiente ventana:

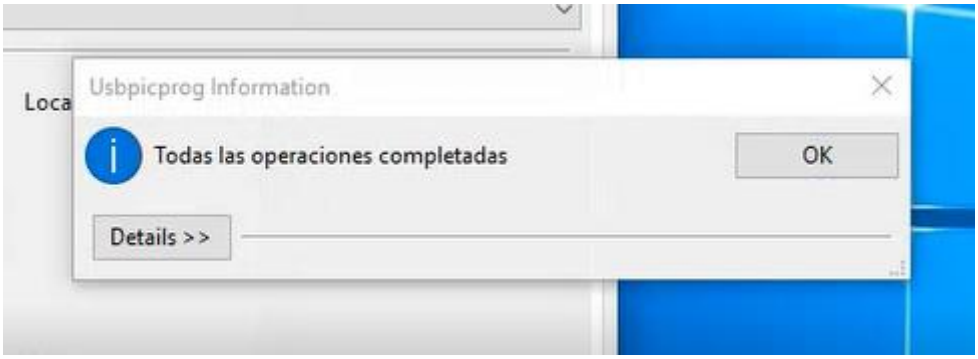


Ilustración 46: Programación realizado con éxito

Tras finalizar el proceso de grabación es importante que todas las posiciones del interruptor DIP vuelvan a la posición inicial, sino el PIC instalado no funcionará.

4. Casos prácticos

En este capítulo se va a resolver las prácticas de la asignatura Sistemas basados en microprocesador, así como otros ejercicios propuestos en la asignatura para comprobar la funcionalidad de la entrenadora y detectar posibles problemas a la hora de realizar las prácticas. Se van a resolver utilizando el lenguaje ensamblador.

La metodología a seguir será la siguiente:

- Enunciado de cada ejercicio.
- Diagrama de conexiones y conexiones empleadas en la entrenadora.
- Enlace a YouTube para visualizar el funcionamiento de la placa.
- Explicación del funcionamiento mostrado en el video anterior.

Todos los enunciados están disponibles en el *Anexo II. Prácticas de la asignatura* y los códigos empleados en ensamblador en el *Anexo III. Códigos ensamblador*.

4.1. Sesión 2

Se empieza por el segundo ejercicio de la sesión 2 de prácticas, puesto que la primera sesión es una introducción al manejo de Virtual BreadBoard y MPLAB. El primer ejercicio de la segunda sesión consiste en familiarizarse con los bits Z y C (cero y acarreo en el bit de más peso) del registro de estado mediante la suma y resta de dos números.

- Enunciado ejercicio 2:

Se trata de implementar un sumador/restador de 4 bits de la siguiente manera: mediante los 4 interruptores se introduce el primer operando y se pulsará el pulsador “OK1”, a continuación se introduce el segundo operando por el interruptor y se acciona el pulsador “OK2”. Mediante los dos pulsadores restantes seleccionaremos la operación que queremos realizar (suma o resta) y visualizaremos el resultado de la operación en los diodos led (“Resultado”). Si la resta es negativa se encenderá el diodo “Negativo”.

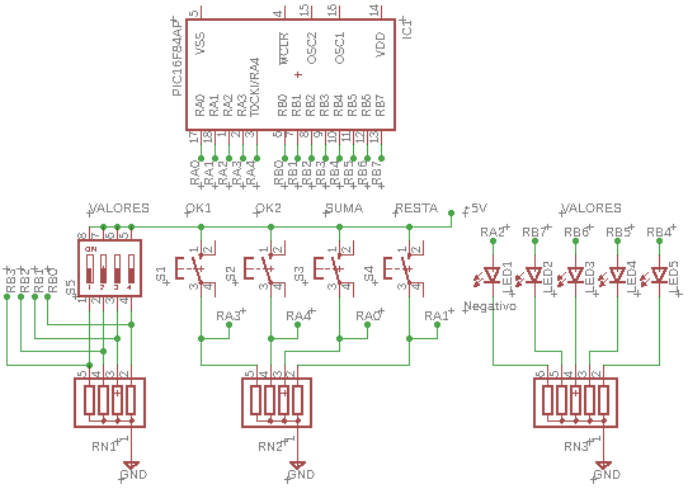


Ilustración 47: Diagrama de conexiones ejercicio 2 sesión 2

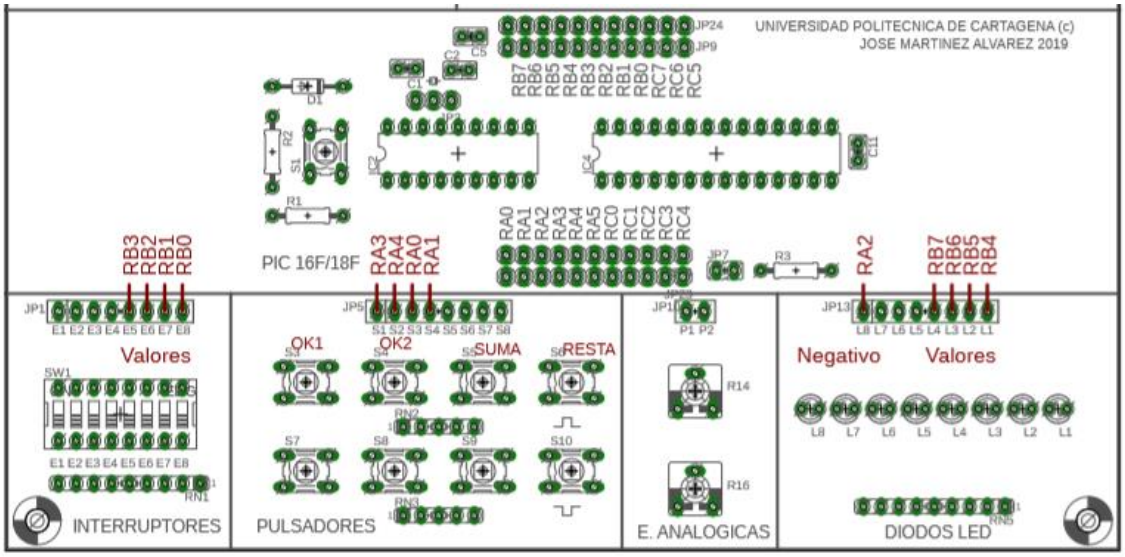


Ilustración 48: Conexiones Entrenadora Ejercicio 2 Sesión 2

Enlace: https://youtu.be/sShRXDxyk7Q

Se introduce el número “0001” por los interruptores que corresponde al número 1, a continuación se pulsa “OK1” y “OK2” y se selecciona la operación de suma. El resultado se muestra por los primero cuatro diodos, siendo “0010”, 2 en decimal. Posteriormente se almacena en el primer operando el número 2 (0010) y en el segundo el 1 (0001), se selecciona la operación de resta, dando el resultado “0001”, es decir, 1 en decimal. Por último, para probar la resta negativa, se introduce un 1 (0001) en el primer operando y un 4 (0100) en el segundo, encendiéndose el led conectado a RA2 y mostrando por los led el resultado de la operación, que es 3 (0011).

4.2. Sesión 3

- Enunciado ejercicio 1:

Realizar un programa que encienda y apague todos los led con un parpadeo de un segundo (medio segundo encendido y medio apagado) mientras la patilla RA0 esté estimulada.

- Enunciado ejercicio 2:

Realizar un programa que cuando se estimule RA0, encienda el diodo conectado a RB0 durante medio segundo, transcurrido este tiempo se apague durante medio segundo y acto seguido se encienda el diodo conectado a RB1, y así sucesivamente hasta llegar al led conectado a la patilla RB7. Cuando se apague este último el proceso volverá a empezar de nuevo.

- Enunciado ejercicio 3:

Realizar un programa que ejecute el proceso inverso al descrito en el ejercicio anterior.

- Enunciado ejercicio 4:

Realizar un programa que cuando se estimule RA0 se encenderán y apagaran los diodos de izquierda a derecha, y al llegar al led RB7 encienda y apague los leds de derecha a izquierda. Todos estos procesos lo harán indefinidamente mientras haya un "1" lógico por la patilla RA0, en caso contrario el proceso se detiene.

Vamos a resolver el ejercicio 4, ya que es el más completo y comprende los 3 ejercicios anteriores:

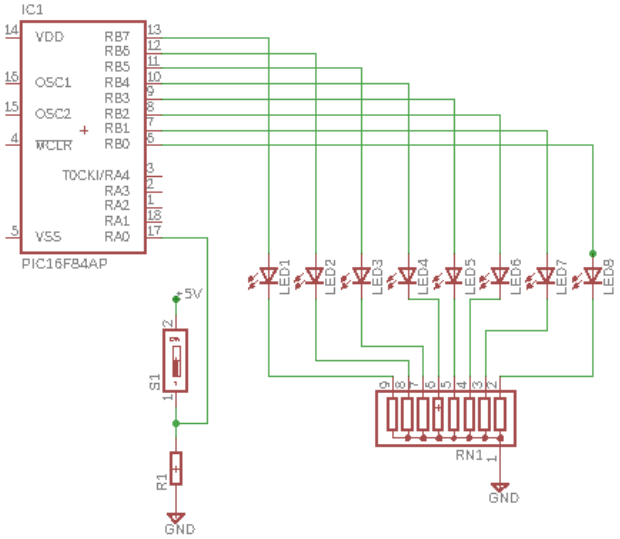


Ilustración 49: Diagrama de conexiones Ejercicio 4 Sesión 3

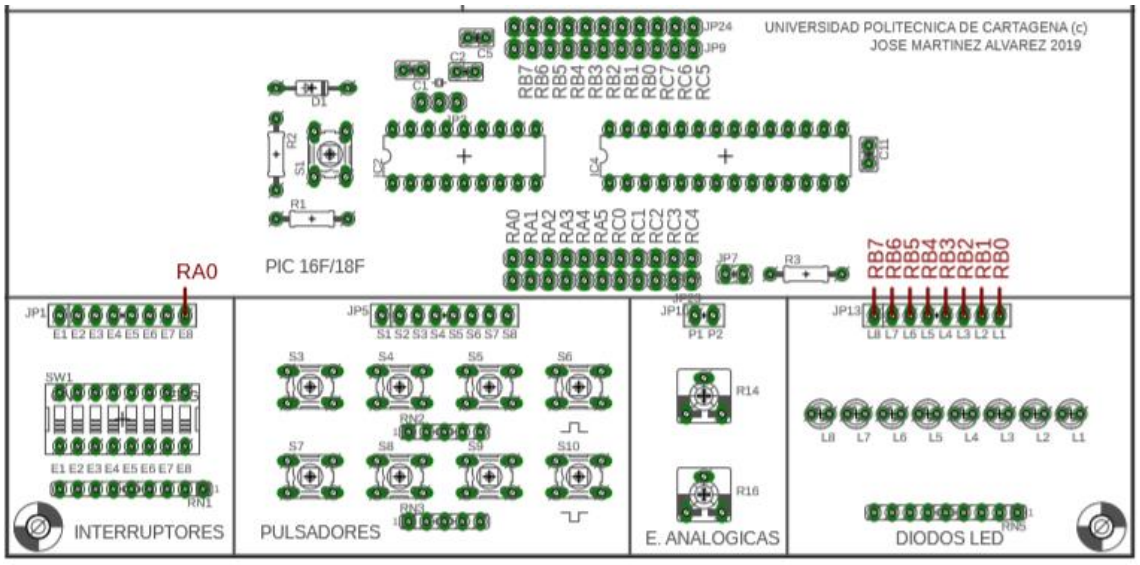


Ilustración 50: Conexiones Entrenadora Ejercicio 4 Sesión 3

Enlace: https://youtu.be/1y3GtU9YoKk

Se ve como al activar el interruptor comienza el proceso, encendiendo y apagando los led de izquierda a derecha y viceversa. Más adelante se desactiva RA0 parándose el proceso y cuando es activado de nuevo se reanuda desde donde se había parado.

4.3. Sesión 4

- Enunciado ejercicio 1:

El primer ejercicio consiste en implementar un contador ascendente/descendente que vaya desde 0 hasta F y vuelva hasta 0 si la patilla RA0 es estimulada. El valor de cada incremento/decremento se mostrará por el display de forma intermitente (medio segundo encendido y medio segundo apagado). El proceso será infinito mientras se mantenga a "1" el interruptor conectado a RA0 ("OK").

- Enunciado ejercicio 2:

Ahora usando el contador anterior, modificaremos el programa para poder variar el valor del Divisor de Frecuencias durante el transcurso del proceso, esto nos permitirá cambiar la velocidad en la que se mostrará el valor por el display. Para ello usaremos los interruptores conectados a RA1-RA3 ("PS0", "PS1" y "PS2" en el diagrama esquemático) y activando la patilla RA4 ("OK1").

Puesto que ambos ejercicios consisten en hacer un contador ascendente/descendente, desde 0 hasta F y desde F hasta 0 se va a resolver el segundo, ya que presenta una variación interesante de ver:

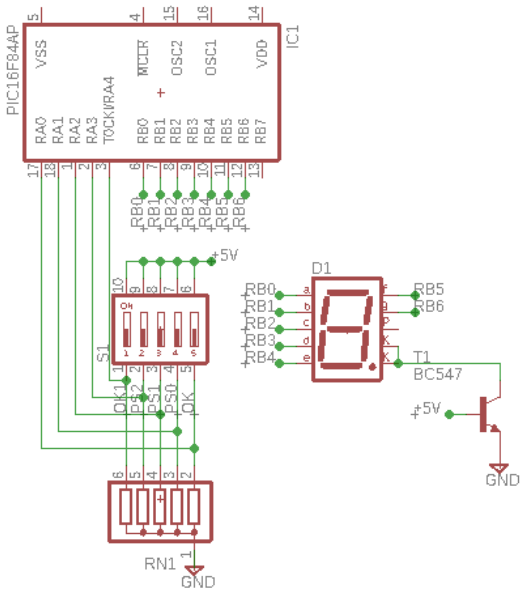


Ilustración 51: Diagrama de conexiones Ejercicio 2 Sesión 4

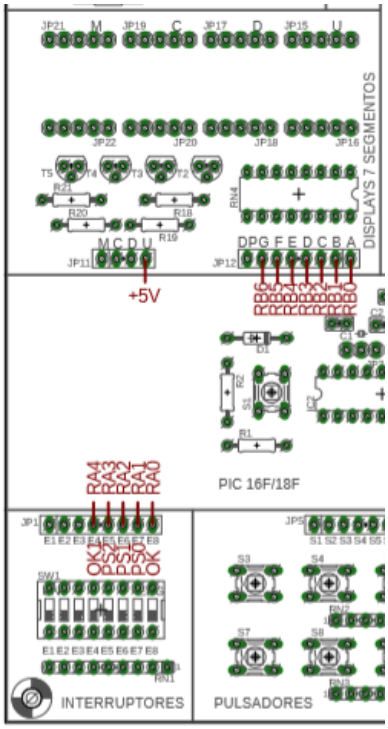


Ilustración 52: Conexiones Entrenadora Ejercicio 2 Sesión 4

Enlace: https://youtu.be/pgupZuc_eqY

Cuando activamos RA0 el contador se pone en marcha, primero introducimos por el interruptor el valor "100" que corresponde a un divisor de frecuencias de 1:32 pero hasta que RA4 no es activado sigue a la misma velocidad. Cuando esto ocurre inmediatamente se vuelve más rápido el proceso. Si RA4 sigue activo y vamos variando el divisor de frecuencias vemos cómo va variando la velocidad.

4.4. Sesión 5

En esta sesión vamos a estudiar el uso de las interrupciones.

- Enunciado ejercicio 1:

Este ejercicio es similar al ejercicio 2 de la sesión 4, con la diferencia de que ahora se usa la interrupción por estimulo en la patilla RB0 para, como en el caso del ejercicio anterior, variar la velocidad de parpadeo. Este ejercicio no va a ser resuelto ya que es prácticamente igual que el anterior y en el ejercicio 3 de esta sesión podremos comprobar el funcionamiento de la interrupción en RB0.

- Enunciado ejercicio 2:

Realizar un programa que compruebe el funcionamiento de las interrupciones producidas por el cambio de valor de las patillas RB4-RB7. Para ello se programará un bucle infinito a la espera de que se produzca alguna interrupción. Cuando esta suceda la rutina de servicio la detectará y mostrará su número por el display hasta que se modifique otra

distinta. El display estará apagado si todos los interruptores están a OFF o si hay más de un interruptor a ON.

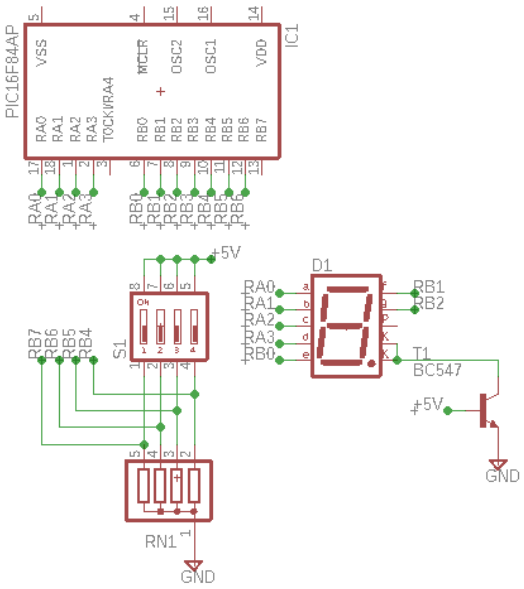


Ilustración 53: Diagrama de conexiones Ejercicio 2 Sesión 5

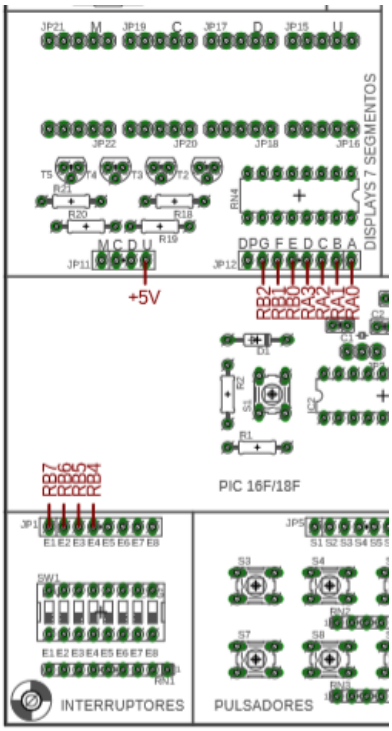


Ilustración 54: Conexiones Entrenadora Ejercicio 2 Sesión 5

Enlace: https://youtu.be/nDj_5X1ogwE

Se ve como cada vez que hay un flanco ascendente por uno de los interruptores se muestra por el display el número de la interrupción que ha sido activada. Cuando se llega a la última interrupción y se activa también la interrupción RB6 el display se apaga, puesto

que hay más de un interruptor ON. En cuanto se baje uno se muestra el que sigue activo. Al bajar todos el display se mantiene apagado.

- Enunciado ejercicio 3:

Realizar un programa que compruebe el tratamiento de las interrupciones en RB4-RB7 y RB0. De tal manera que cuando se produzca la primera se encienda el led “INTB7-4” y con la segunda el led “INTB0”, quedando encendido hasta la siguiente interrupción.

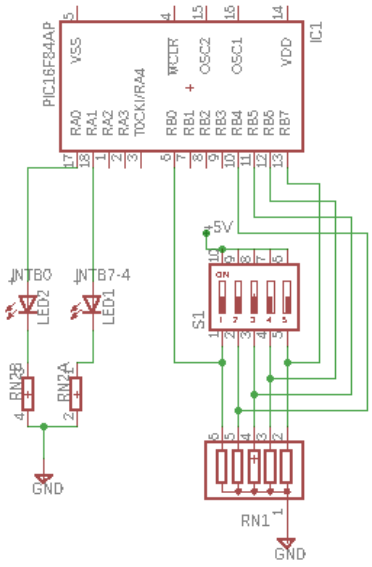


Ilustración 55: Diagrama de conexiones Ejercicio 3 Sesión 5

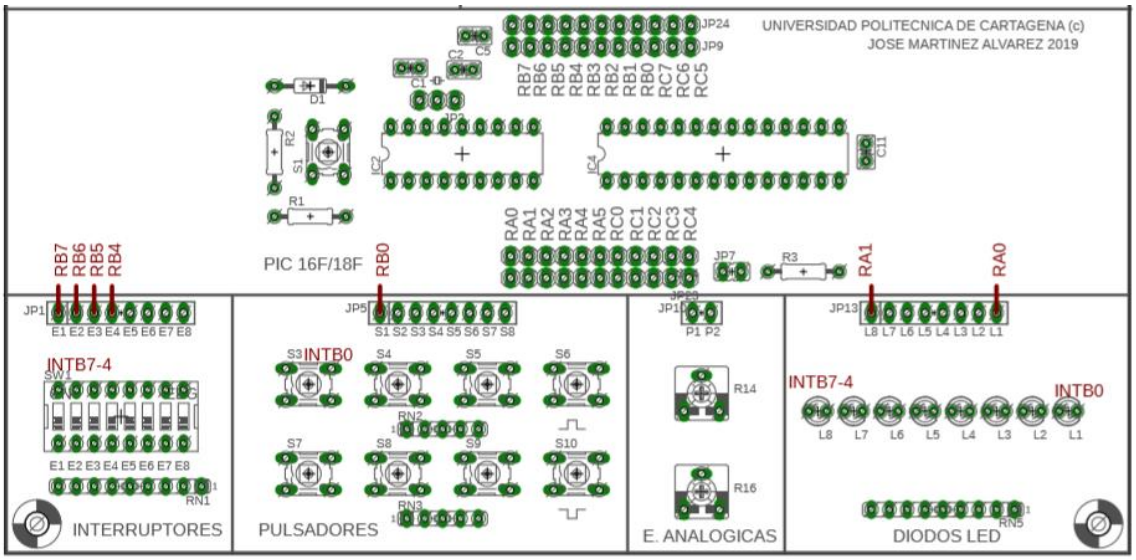


Ilustración 56: Conexiones Entrenadora Ejercicio 3 Sesión 5

En este ejercicio se encuentra uno de los primero problemas. Al principio con el código realizado en las prácticas, la puesta en marcha en la entrenadora no se comporta como era esperado, al realizarse una interrupción por las patillas RB4-RB7 no había problema, sin embargo, en el caso de la patilla RB0, al producirse una interrupción no se comportaba

como era debido. El posible problema es el caso de los rebotes. Para solucionarlo al entrar en la rutina de servicio de interrupciones y comprobar si ha sido por flanco descendente de RB0, se ha esperado 20 milisegundos para que se estabilizara, posteriormente se ha preguntado si el pulsador RB0 tenía un valor de "0". Si esto se cumple, entonces enciende el led. Esto puede comprobarse en el siguiente enlace:

Enlace: https://youtu.be/6mzaKluokV0

Cada vez que cambia de estado una de las patillas RB4-RB7 se enciende el led de la izquierda y se apaga el de la derecha y cuando ocurre la interrupción de RB0 sucede el efecto contrario.

4.5. Sesión 6

Es esta sesión se estudia el uso de la memoria EEPROM de datos. Esta sesión es muy interesante, ya que veremos cómo al trabajar con esta memoria no se pierde información al producirse algún tipo de reset o cuando se produce un corte en la alimentación.

- Enunciado ejercicio 1:

El primer ejercicio consiste en escribir los valores de la secuencia 2, 4, 6,…, 32 desde la posición 0x10 hasta la 0x1F, esto lo hará cuando se active la patilla RA0. Cuando termine el proceso de escritura se encenderá un led indicando el fin de este proceso. Cuando se introduzca un "1" por RA4 se realizara el proceso de lectura de las posiciones 0x10 hasta la 0x1F mostrando los valores por los diodos led de forma intermitente.

La conexión inicial de este ejercicio implica la utilización de 9 diodos led pero solo disponemos de 8. Afortunadamente gracias a los displays, contamos con 8 diodos más en forma de segmentos, así que se empleará uno de ellos para representa el fin del proceso de escritura.

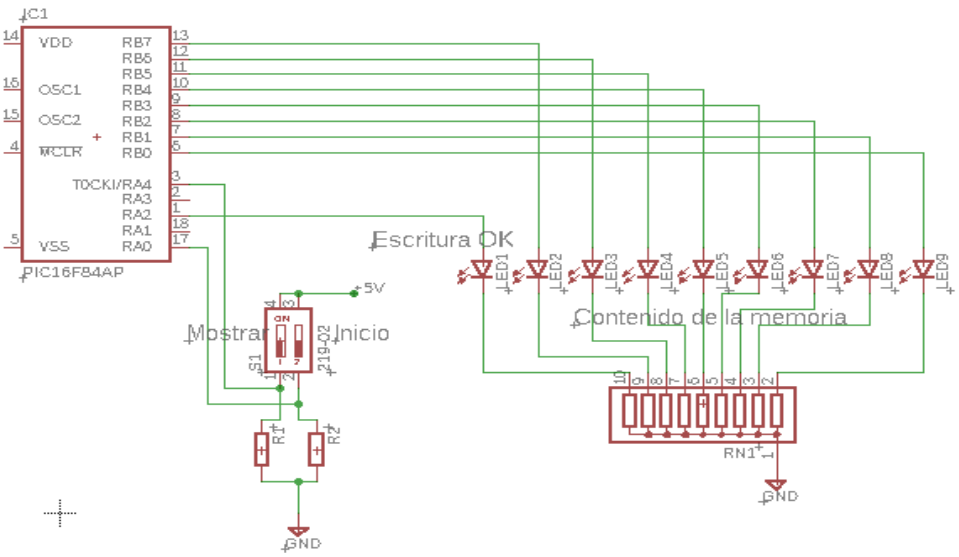


Ilustración 57: Diagrama de conexiones Ejercicio 1 Sesión 6

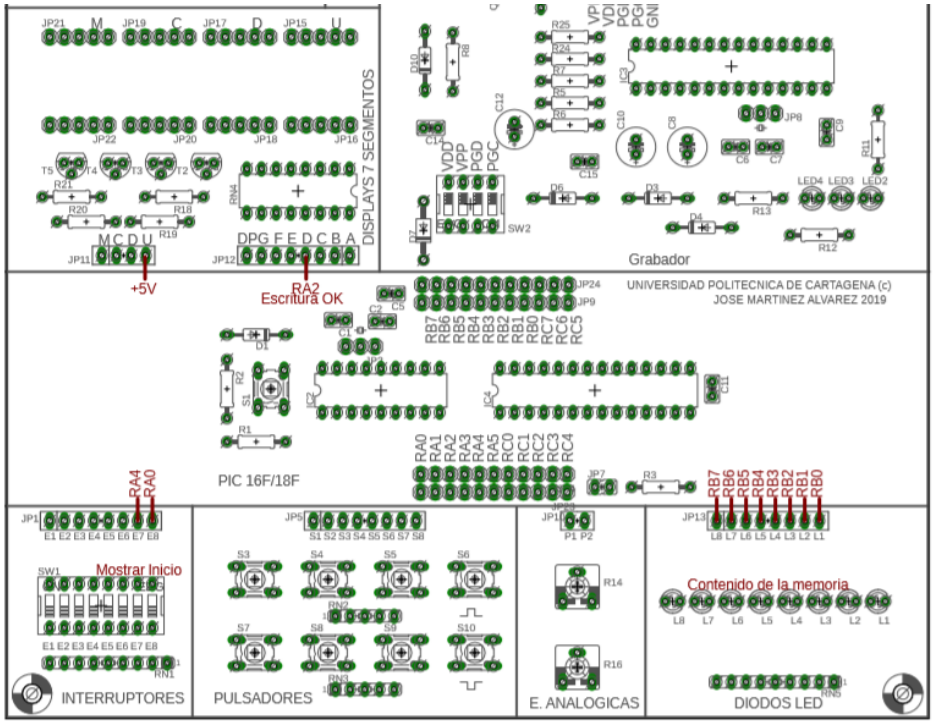


Ilustración 58: Conexiones Entrenadora Ejercicio 1 Sesión 6

Enlace: https://youtu.be/VN75Jptgwts

En cuanto “Inicio” es activado se enciende “Escritura OK” dando lugar a la finalización del proceso de escritura, este proceso es inmediato puesto que tarda aproximadamente 10 milisegundos. Inmediatamente al activar “Mostrar” se representa por los diodos los valores de la secuencia 2 (“00000010”), 4, 6,…, 32 (“00100000”) y vuelve a empezar, realizándose un bucle infinito.

- Enunciado ejercicio 2:

En el segundo lo que haremos será un programa que escriba y lea en la memoria EEPROM de datos utilizando las posiciones de memoria desde la 0x00 hasta la 0x07. En primer lugar el programa comenzará desde la posición 0x00 esperando a que introduzcamos un valor por las patillas RA0-RA2 y confirmando su valor por la patilla RA3. Posteriormente y de forma automática se incrementará la dirección de memoria e iremos introduciendo números hasta alcanzar la posición 0x07. Una vez introducidos todos los números se encenderá el led conectado a RA4, que indica que estamos en “modo lectura”. Ahora podemos mostrar el contenido de una dirección de memoria que indiquemos mediante las patillas RA0-RA2 y su confirmación por RA3. En los diodos, los primero 4 bits se representaran el valor, y en los otros cuatro la dirección de memoria.

Como en el prototipo no se implementó la resistencia pull-up necesaria para que RA4 funcionara como salida y al igual que en el ejercicio anterior requiere un led más, vamos a volver a usar un segmento de uno de los displays para indicar que estamos en “modo lectura” y vamos a utilizar la patilla RA3 para ello y la patilla RA4 como confirmación.

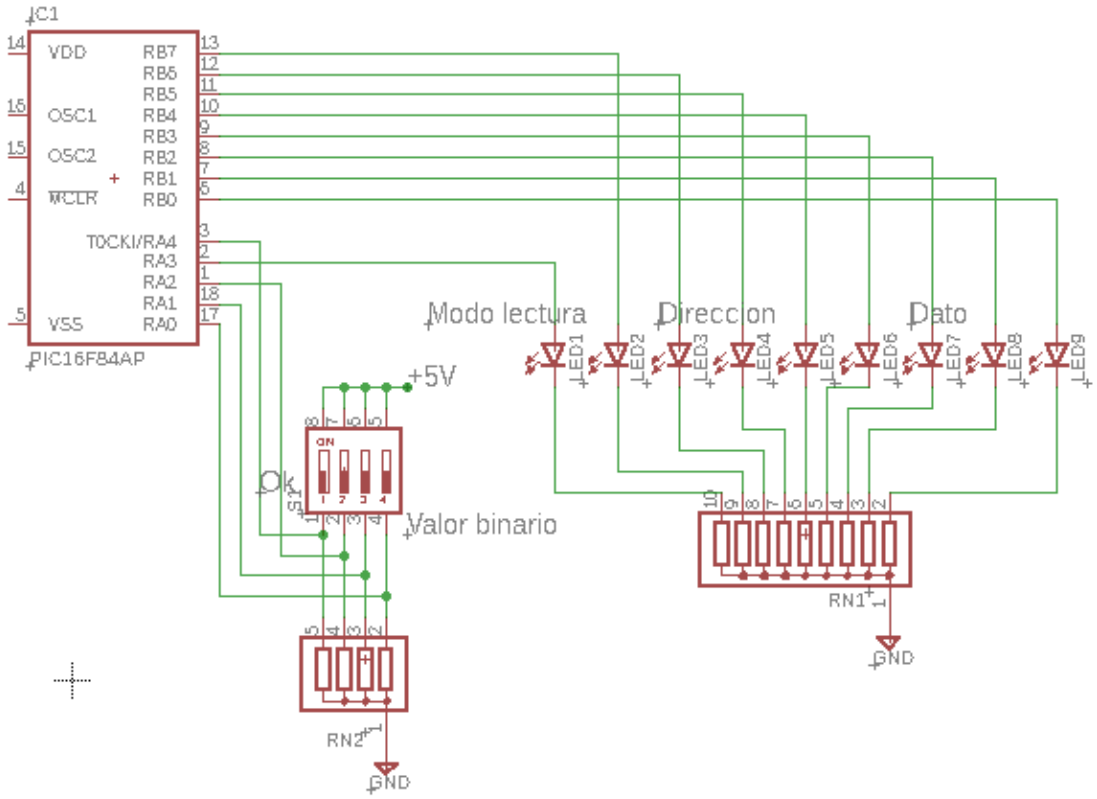


Ilustración 59: Diagrama de conexiones Ejercicio 2 Sesión 6

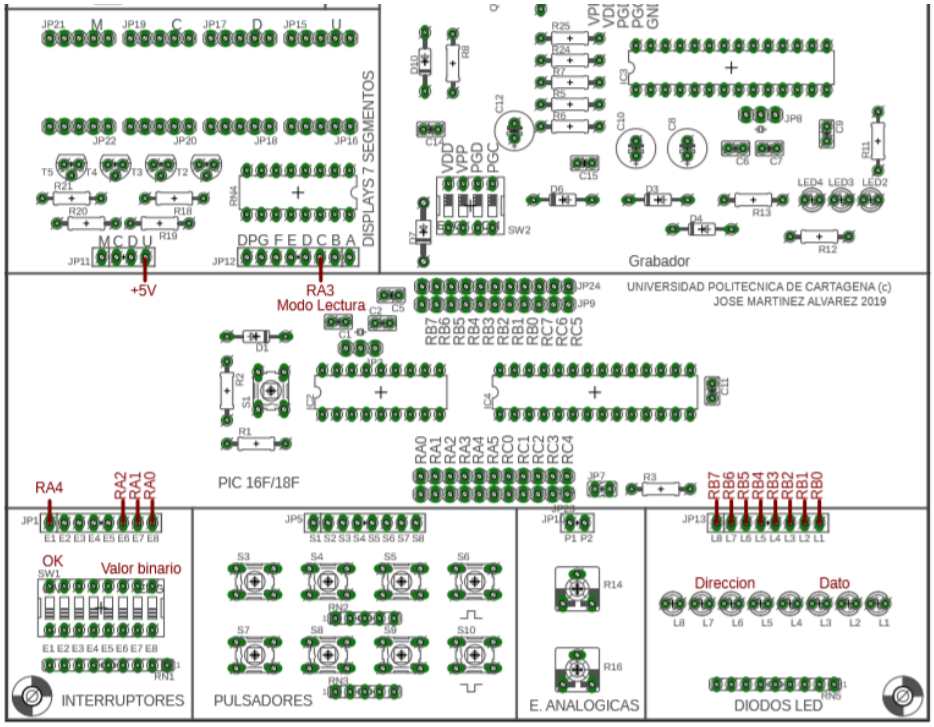


Ilustración 60: Conexiones Entrenadora Ejercicio 2 Sesión 6

Enlace: https://youtu.be/aRp30bUbPBM

Al empezar el programa no se enciende ningún led ya que está esperando a que se introduzca un valor en la dirección 0x00. Se introduce el número 0 e inmediatamente se incrementa el valor de la dirección, mostrando su valor por 4 diodos de la izquierda. Vamos introduciendo valores que se van mostrando por los diodos de la derecha hasta llegar a la dirección 0x07. Tras esto se enciende el segmento del display indicando que ahora estamos en modo lectura. Leemos primero la dirección 0x01, donde está almacenado el valor 1, el valor de la dirección se muestra en los led de la izquierda y el valor en los de la derecha, y posteriormente la dirección 0x06. Como estamos trabajando con la memoria EEPROM, al desconectar y volver a conectar la placa a la alimentación, sigue conservando en su memoria los valores introducidos en el proceso de escritura.

4.6. Dado electrónico

Se trata de realizar un programa que al pulsar un botón se muestre por uno de los displays un número pseudoaleatorio comprendido entre el uno y el seis, como si de un dado se tratara.

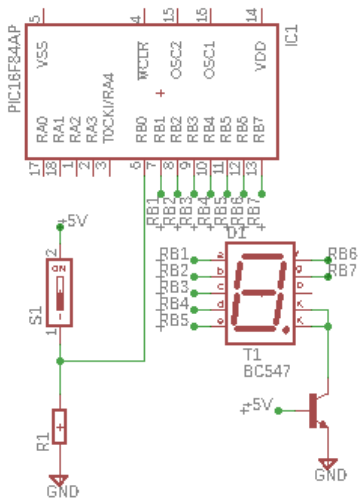


Ilustración 61-: Diagrama de conexiones Dado Electrónico

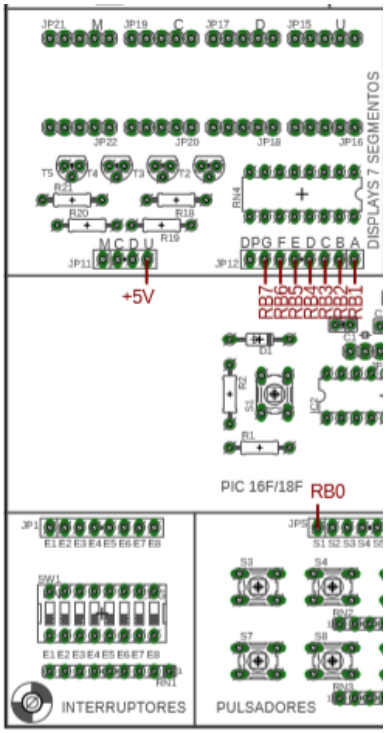


Ilustración 62: Conexiones Entrenadora Dado Electrónico

Enlace: https://youtu.be/QYEkBly7t74

Vemos que cada vez que se pulsa el botón se muestra por el display un número comprendido entre 1 y 6.

4.7. Ejemplo multiplexación

Esto no es un ejercicio de clase pero ya que antes se ha hablado del proceso de multiplexación, se va a realizar un ejemplo: para mostrar el número “1234” el proceso a seguir sería el siguiente:

- Activar el transistor de las milésimas, activando el bit del puerto correspondiente ha dicho transistor.
- Enviar el número 1 por el puerto que controla los segmentos.
- Esperar 5 milisegundos y desactivar el transistor de las milésimas.

Esto hay que hacerlo con los 4 transistores. El esquema de conexiones es:

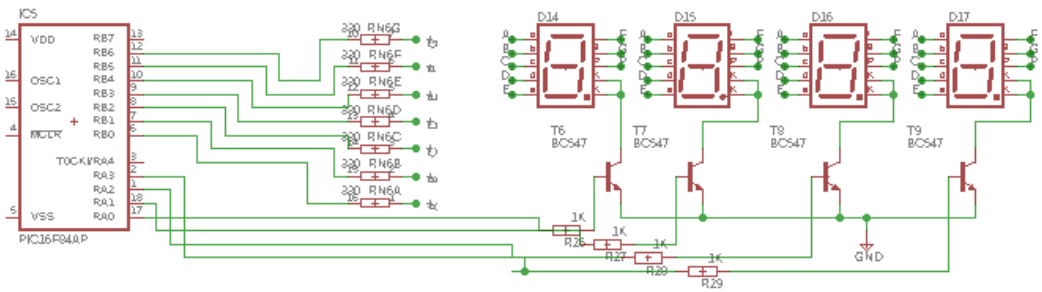


Ilustración 63: Diagrama de conexiones ejemplo displays "1234"

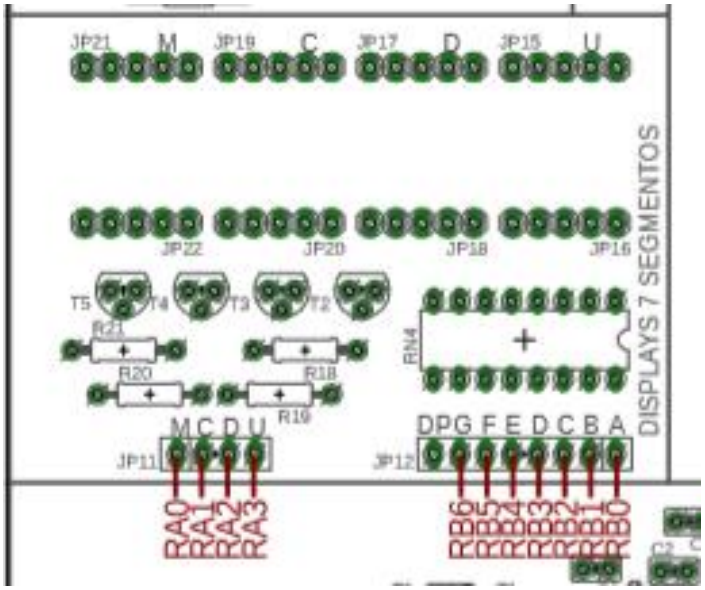


Ilustración 64: Conexiones Entrenadora Ejemplo "1234"

El resultado podemos verlo en la siguiente imagen:

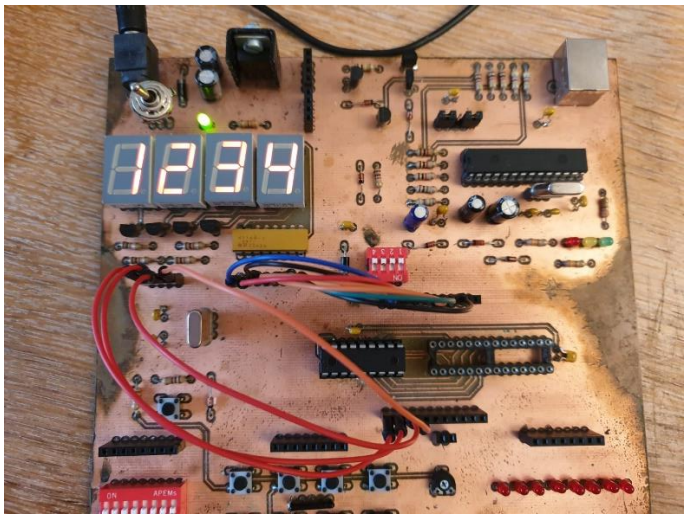


Ilustración 65: Ejemplo Multiplexación "1234"

Se observa los cuatro displays encendidos a la vez, pero esto realmente no es así. Gracias a la multiplexación es posible conectar varios displays a un mismo microcontrolador.

4.8. Contador ascendente 8 bits

Se trata de implementar un programa que cuando se introduzca un "1" por la partilla RA4 se inicie una cuenta desde 0 hasta 255, mostrando el valor actual del contador por tres displays. Se irá incrementado cada dígito con una frecuencia de 0,25 segundos. En caso de que se deje de alimentar RA4 el contador se parará.

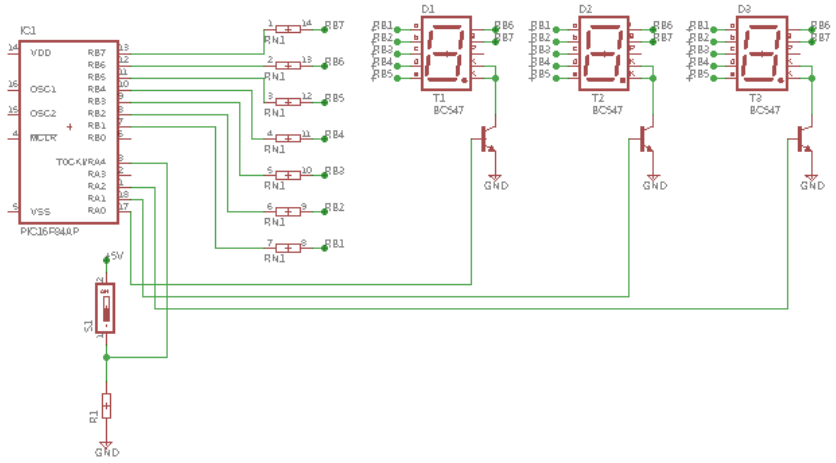


Ilustración 66: Diagrama de conexiones Contador ascendente

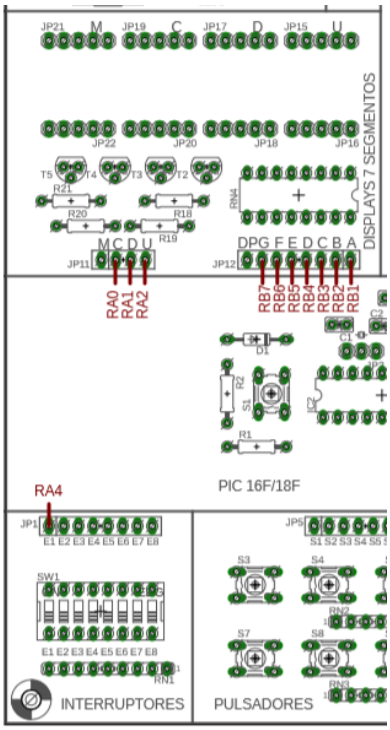


Ilustración 67: Conexiones entrenadora Contador ascendente

La solución de este ejercicio frente a la simulación hecha en clase es diferente. El código realizado en el simulador no se comportaba igual que en la realidad, en este último cada vez que se incrementaba el número, los displays se apagan y se volvían a encender al mostrarlos, cosa que no sucedía en la simulación. La realidad es que como estaba implementado el programa en este último era imposible que funcionase como se quería, que era mostrar la cuenta sin intermitencias, ya que mientras se ejecutaba la rutina que cuenta 0,25 milisegundos se dejaba de mostrar continuamente el número. La solución a este problema es muy sencilla, simplemente basta con llamar a la rutina que muestra el número por los displays en la rutina del tiempo.

Enlace: https://youtu.be/kUf92WSmarU

Se ve como al activar el interruptor inmediatamente se empieza a contar desde 0. Cuando llega a 255 empieza de nuevo. Si en algún momento ponemos a "0" RA4, la cuenta se detiene mostrando por los displays el valor que tiene el contador en el momento de la parada, tras volver a activar el interruptor la cuenta sigue desde donde estaba.

4.9. Sensor marcha atrás

Se desea desarrollar un sistema de aviso de aparcamiento para vehículos, de manera que informe al conductor de lo lejos o cerca que esta de un obstáculo cuando da marcha atrás. El funcionamiento es el siguiente: mediante un led simularemos un zumbador, de manera que cuanto más se acerque al objeto se encenderá y apagara de forma intermitente más rápido. La intermitencia del led dependerá de la distancia, medido en centímetros, según la siguiente tabla:

Distancia (cm)	Intermitencia
Despejado	Nunca
5-20	0,50seg
21-40	0,75seg
41-60	1,00seg
>61	Nunca

Tabla 4: Intermitencia Sensor Marcha Atrás

El sensor proporciona 6 valores binarios al PIC que indican la distancia. Utilizaremos un interruptor conectado a RA0 que indique cuando se ha activado la marcha atrás. Para simplificar el problema suponemos que nunca nos acercamos a menos de 5 cm.

Utilizaremos 5 de los interruptores para simular el sensor que irán conectados a los pines RB0-RB7. Otro de los interruptores como la marcha atrás y el led irá conectado a la patilla RA0, podemos ver las conexiones en las siguientes imágenes:

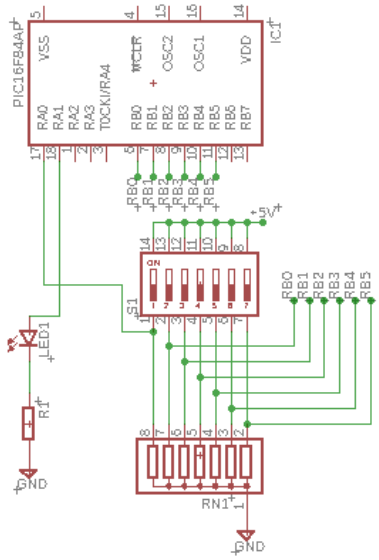


Ilustración 68: Diagrama de conexiones Sensor Marcha Atrás

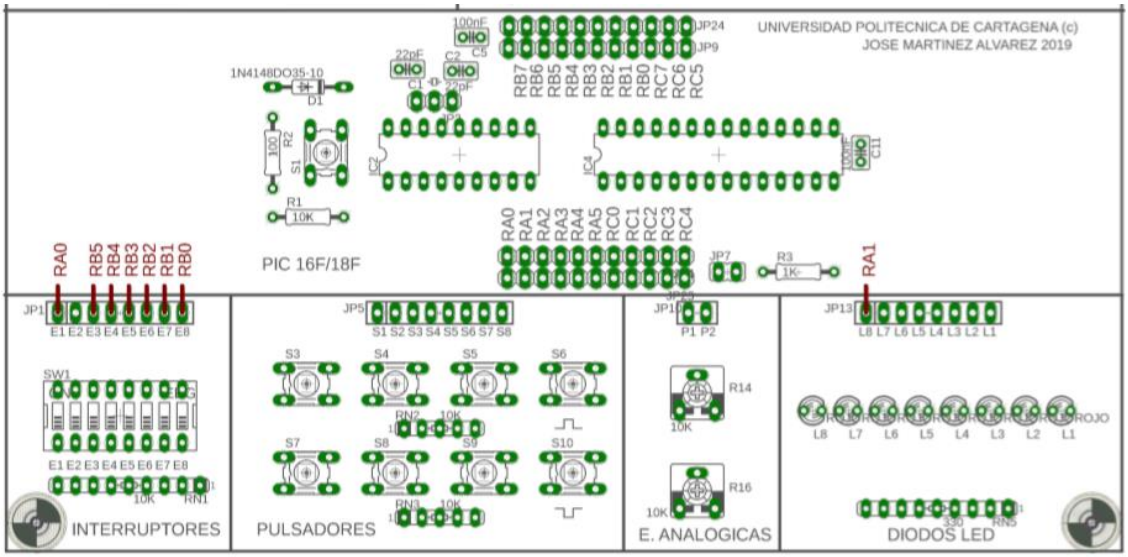


Ilustración 69: Conexiones entrenadora Sensor Marcha Atrás

Enlace: https://youtu.be/92Ycst8-5Sk

Al iniciar el programa y una vez activado el sensor (energizando RA0), el led se encuentra apagado, esto se debe a que por los interruptores estamos introduciendo un 0 y hemos establecido que la distancia mínima será de 5. Vemos como al introducir un 4 (000100) sigue sin encenderse. A continuación se introduce un 5 (000101) y el led empieza a encenderse y apagarse con una intermitencia de 0,5 segundos, esto se mantiene hasta el valor 20 (010100). Posteriormente se mete el 21 (010101), donde se ve como ahora la intermitencia es más lenta, de 0,75 segundos, hasta el valor 40. Más adelante se introduce el 44 (101101) pasándose a 1 segundo de intermitencia, y el valor 60 (111100). Para finalizar se ve como al llegar a 61 (111101) y posterior, 63 (111111) el led ya no se enciende, pues está despejado.

5. Presupuesto

Una de las cosas más importantes que hay que tener en cuenta a la hora de realizar un proyecto es el coste necesario para llevarlo a cabo. Este proyecto surge con la idea de proporcionar equipos de prácticas a un coste razonable. En esta sección se estudiará si el diseño realizado es más rentable frente a otras placas entrenadoras existentes el mercado.

En la siguiente tabla encontramos el precio de los componentes, así como el precio de la fuente de alimentación externa utilizada y el cable USB:

Descripción	Valor	Cantidad	Precio Unitario	Precio Total
Interruptor DIP 8 Posiciones		1	1,77 €	1,77 €
Interruptor DIP 4 Posiciones		1	1,04 €	1,04 €
Interruptor táctil		9	0,22 €	1,98 €
Resistencia	1K	12	0,09 €	1,08 €
Resistencia	330	4	0,09 €	0,36 €
Resistencia	470	2	0,09 €	0,18 €
Resistencia	100	1	0,09 €	0,09 €
Resistencia	100K	1	0,09 €	0,09 €
Resistencia	10K	1	0,09 €	0,09 €
Resistencia	470K	1	0,09 €	0,09 €
Resistencia	1M	1	0,09 €	0,09 €
Red de resistencias 9 PIN	330	1	0,39 €	0,39 €
Red de resistencias 9 PIN	10K	1	0,43 €	0,43 €
Red de resistencias 5 PIN	10K	2	0,29 €	0,58 €
Red de resistencias 16 PIN	330	1	0,95 €	0,95 €
Resistencia ajustable	10k	2	0,49 €	0,98 €
Diodo	1N4007	2	0,17 €	0,34 €
Diodo	1N4148	7	0,09 €	0,63 €
LED Rojo 3mm		9	0,15 €	1,35 €
LED Verde 3mm		2	0,15 €	0,30 €
LED Amarillo 3mm		1	0,15 €	0,15 €
Display 7 Segmentos		4	1,58 €	6,32 €
Zócalo 18 PIN		1	0,75 €	0,75 €
Zócalo 28 PIN		2	1,15 €	2,30 €
USB tipo B Hembra		1	1,20 €	1,20 €
Condensador	0,1uF	7	0,08 €	0,56 €
Condensador	22pF	4	0,08 €	0,32 €
Condensador electrolítico	100uF	2	0,18 €	0,36 €
Condensador electrolítico	10uF	1	0,15 €	0,15 €

Condensador electrolítico	1uF	1	0,18 €	0,18 €
Mosfet canal N	BS170	2	0,36 €	0,72 €
Transistor bipolar	BC547B-AP	5	0,37 €	1,85 €
Mosfet canal P	BS250P	1	0,68 €	0,68 €
Regulador de voltaje	MC7805CTG	1	0,38 €	0,38 €
Conector Jack 2,1mm		1	0,83 €	0,83 €
Header hembra 8 PIN		4	0,90 €	3,60 €
Header hembra 2 PIN		1	0,58 €	0,58 €
Header hembra 4 PIN		1	0,64 €	0,64 €
Header hembra 11 PIN		4	0,95 €	3,80 €
Header hembra 3 PIN		2	0,29 €	0,58 €
Header hembra 2x4 PIN		2	1,18 €	2,36 €
Header hembra 5 PIN		8	0,31 €	2,48 €
Header macho 2 PIN		1	0,09 €	0,09 €
Header macho 5 PIN		1	0,22 €	0,22 €
Jumper		3	0,16 €	0,48 €
Disipador de calor		1	0,55 €	0,55 €
Microprocesador PIC18F2550		1	4,23 €	4,23 €
Microprocesador PIC16F84A		1	3,80 €	3,80 €
Conmutador de palanca		1	1,89 €	1,89 €
Cristal de cuarzo	4MHz	1	0,49 €	0,49 €
Cristal de cuarzo	8MHz	1	0,49 €	0,49 €
Fuente de alimentación			10,68€	10,68€
Cable USB tipo A-B		1	2,40€	2,40€
			TOTAL	68,76€

Tabla 5: Coste de los componentes

Como necesitamos una placa por puesto, para 20 en este caso, es necesario acudir a un fabricante de PCB. Si acudimos a pcbway, una fabricante de prototipos, podemos hacer una estimación de cuánto costaría la fabricación de 20 placas, el cual sería de 142$, un poco más de 7$ por placa. Por lo que el precio final sería de:

Descripción	Precio
Componentes	1370,4€
PCB	125,41€
TOTAL	1495.81€

Tabla 6: Coste total

El precio de cada placa, incluyendo fabricación y componentes, es de 74,8€ aproximadamente. Si lo comparamos con la entrenadora USB-PIC'School, la cual tiene

un precio unitario de 160€, 3200€ las 20 placas necesarias, supone un ahorro económico del 46%.

Hay que tener en cuenta que este es el precio de los componentes seleccionados en la distribuidora de componentes Mouser. Por lo que el precio puede variar según la distribuidora donde se decida adquirir los componentes.

6. Conclusiones

Lo que se buscaba con este proyecto era proporcionar equipos para realizar prácticas en la asignatura Sistemas basados en Microprocesador del grado de Ingeniería Electrónica Industrial y Automática. Para ello se ha realizado un estudio de diversas entrenadoras existentes en el mercado para trabajar con los microcontroladores PIC, observándose que el principal problema que poseen estas es el precio, puesto que es necesario una por cada puesto de trabajo, el precio total resulta demasiado alto. Otro de los problemas encontrados es que presentan conexiones preestablecidas, no cuentan con periféricos con los que trabajar o incluyen demasiado pocos que no permiten la resolución de las prácticas.

Posteriormente es necesario establecer las necesidades de conectividad mínimas, concluyendo que los periféricos de E/S que se requieren son salidas digitales a diodos led y displays de 7 segmentos, y entradas digitales mediante interruptores y pulsadores. Pero para no limitar la entrenadora y puesto que existen microcontroladores PIC que incluyen entradas analógicas se decide incluir potenciómetros para poder hacer pruebas con señales analógicas. También, por lo mismo que antes, y aunque en la asignatura nos centremos en el manejo del PIC16F84A, gracias a unos zócalos es posible instalar no solo el PIC mencionado sino cualquier PIC de 18 y 28 pines. A parte es posible conectar otros elementos que no se incluya en la entrenadora puesto que no existen conexiones predeterminadas, sino que es posible conectar cualquier periférico a cualquier patilla del PIC instalado. Además se puede alimentar otros módulos gracias a unos conectores que proveen 5 voltios y conexión a masa. Esto hace que aumente su flexibilidad, ya que es no está limitada a los periféricos incluidos sino que es plausible conectar y alimentar cualquier otro módulo que se desee, como puede ser una pantalla LCD o un teclado matricial.

Una de las características más importantes es que posee su propio grabador. De las diferentes alternativas que existen, se decide incluir un grabador mediante USB basado en “usbpicprog”. Se ha escogido este grabador puesto que es “open source” el cual tanto el hardware, software y firmware está disponible de forma gratuita, es sencillo de utilizar, rápido y es posible programar una cantidad de microcontroladores PIC. Si hay que destacar algo negativo, es que el proyecto ya no está en desarrollo, por lo que a medida que vayan saliendo PIC más modernos no va a ser posible la grabación de los mismos.

La placa está divida en diferentes módulos y todo bien identificado, para que los estudiantes que trabajen con estos equipos puedan identificar cada elemento de la forma más rápida posible y les ayude a conocer un poco más el hardware que rodea este tipo de componentes electrónicos. Tras el diseño, se pasa a realizar un prototipo, pues esto es totalmente necesario para verificar si el diseño cumple con su cometido de forma satisfactoria. Se encuentran varios errores, no muy graves y fáciles de corregir, muchos de ellos culpa de ser principiante con el uso de estos dispositivos. Por ejemplo, se puso el oscilador demasiado lejos de los zócalos, lo que ocasionaba que hubiera demasiado ruido. Otros de los fallos encontrados, y también una de las diferencias que se encuentra en el

uso de los simuladores frente a las herramientas de hardware, es que la patilla RA4 de algunos PIC es diferente a las demás patillas y requiere ser tratada, el problema fue solucionado en el diseño puesto que requiere de más circuitería. Otros problemas incluyen errores en los tamaños de los taladros que el software utilizado no ha generado bien y fallas de componentes.

Tras solucionar los problemas encontrados se resuelven las prácticas de la asignatura, así como otros ejercicios propuestos en el transcurso de la asignatura, con el objetivo de si es posible llevarlas a cabo correctamente. Hasta el ejercicio 3 de la sesión 5 no encontramos ningún problema. En este se encuentra un problema asociado a los rebotes, pero fácilmente solucionable mediante software. Posteriormente, en la sesión 6, encontramos que se necesita un led más, pero esto no es realmente un problema, se puede utilizar uno de los segmentos de un display como si de un led se tratase. Lo que si hay que adaptar para el prototipo (para la solución final los alumnos no tendrán ese problema) es utilizar otra patilla como salida en vez de la RA4, esto tampoco supone mucho problema. Por lo demás no se ha encontrado nada más, pudiéndose completar todas las prácticas de forma correcta.

En cuanto al apartado económico, el precio final de la fabricación de la placa y los componentes, suponiendo que serán montadas por el personal de la universidad, sube a 74,8€ aproximadamente por placa, un total de 1495,81€. Esto puedo parecer un coste inicial alto pero si lo comparamos con otras entrenadoras, como la USB-PIC'School, la cual ha sido utilizada en el laboratorio y es la más completa, notamos que el ahorro que supone realizar una entrenadora propia y adaptada, supone un 46%, un gran ahorro a tener en cuenta.

El resultado final que se buscaba era el de proporcionar un diseño funcional que se pueda fabricar en masa a un coste razonable. Tras la finalización del trabajo se puede asegurar que el diseño realizado cumple con su cometido, a un coste que a priori puede parecer elevado, que si es comparado con otras existentes en el mercado, vemos que es rentable.

7. Vías futuras

La entrenadora está adaptada a las prácticas de la asignatura Sistemas Basados en Microprocesador, por lo que solo se incluye los periféricos utilizados en el trascurso de estas. Así que las posibles vías futuras son:

Añadir periféricos de E/S digitales más complejos, como puede ser una pantalla LCD, un teclado matricial, pines para conectar servomotores o leds RGB.

No centrarse solamente en el estudio de componentes digitales, incluyéndose elementos de entrada analógicos tales como sensores de temperatura, luz, presión, humedad, aceleración, etc.

Estos periféricos pueden incluirse en la propia placa o bien crear otros módulos independientes con estos dispositivos, creándose así lo que podríamos denominar módulos de expansión.

Además, una posible modificación muy interesante, sería añadir un conector compatible con los grabadores/depuradores PICkit. De esta manera será posible grabar los PIC instalados en los zócalos en caso de que el propio grabador falle.

Pero lo más interesante es que sería posible realizar la depuración de los PIC. El software MPLAB X-IDE junto al PICkit permite la depuración, este proceso consiste en controlar la ejecución del programa en tiempo real grabado en el PIC, con BreakPoints (puntos de ruptura) y paso a paso. Permitiéndose observar y modificar los registros especiales y las posiciones de memoria. Tener esta opción sería valioso ya que permitiría localizar fallos en la programación y poder corregirlos antes de grabar el PIC.

Este proceso no es compatible con todos los PIC, sobre todo los más antiguos, de manera que podría ser beneficioso para los estudiantes el estudio de otros microcontroladores de esta familia más modernos compatibles con el proceso de depuración.

8. Bibliografía

8.1. Referencias

"PIC16F84 Datasheet", 2001 Microchip Technology Inc,

“PICkitTM 3 Programmer/Debugger User’s Guide”, 2010 Microchip Technology Inc,

Angulo, J, y Angulo, I. (2003). *Microcontroladores PIC. Diseño practico de aplicaciones. Primera parte. El PIC16F84 Lenguajes PBASIC y Ensamblador*. Madrid: McGRAW-HILL.

Mandado, E. (2007). *Microcontroladores PIC: sistema integrado para el autoaprendizaje*. Barcelona: Marcombo.

Palacios, E., Remiro, F. y López, L. (2009). *Microcontrolador PIC16F84. Desarrollo de Proyectos*. Madrid: RAMA.

Sánchez, M., Fernández, J. M. y García, J. M. (2018). *Una propuesta para la docencia de dispositivos microcontroladores*. Universidad Politécnica de Cartagena.

8.2. Referencias web

Distribuidora de componentes: https://www.mouser.es/

Edificio SAIT: http://www.upct.es/sait/es/inicio/

http://picmania.garcia-cuervo.net/proyectos_aux_botones.php

http://tecbolivia.com/index.php/articulos-y-tutoriales-microcontroladores/19-icsp-como-usar-qprogramacion-serial-en-circuitoq-con-microcontroladores-pic

http://www.todopic.com.ar/foros/index.php?topic=15129.0

https://www.areatecnologia.com/electronica/potenciometro.html

https://www.lpkf.com/en/industries-technologies/research-in-house-pcb-prototyping/pcb-structuring/

https://www.zonamaker.com/electronica/intro-electronica/teoria/resistencias-de-pull-up-y-pull-down

Manual de la placa *Explorer 8 Development Kit:* *https://www.microchip.com/Developmenttools/ProductDetails/DM160228*

Manual de la placa *LAB-X3 Experimenter Board:* *http://store.melabs.com/prod/boards/LABX3A.html*

Manual de la placa *USB-Pic'School*: https://mkelectronica.com/producto/usb-picschool/

Prototipos de PCB: https://www.pcbway.com/

Web de la empresa Microchip INC. www.microchip.com.

Web de usbpicprog: http://usbpicprog.org/

Wikipedia. Microcontrolador PIC. https://es.wikipedia.org/wiki/Microcontrolador_PIC, 2019.

Wikipedia. Microcontrolador. https://es.wikipedia.org/wiki/Microcontrolador. 2019.

Wikipedia. Visualizador de siete segmentos. https://es.wikipedia.org/wiki/Visualizador_de_siete_segmentos. 2019.

Anexos

Anexo I. Planos

Anexo I.I. Esquemático

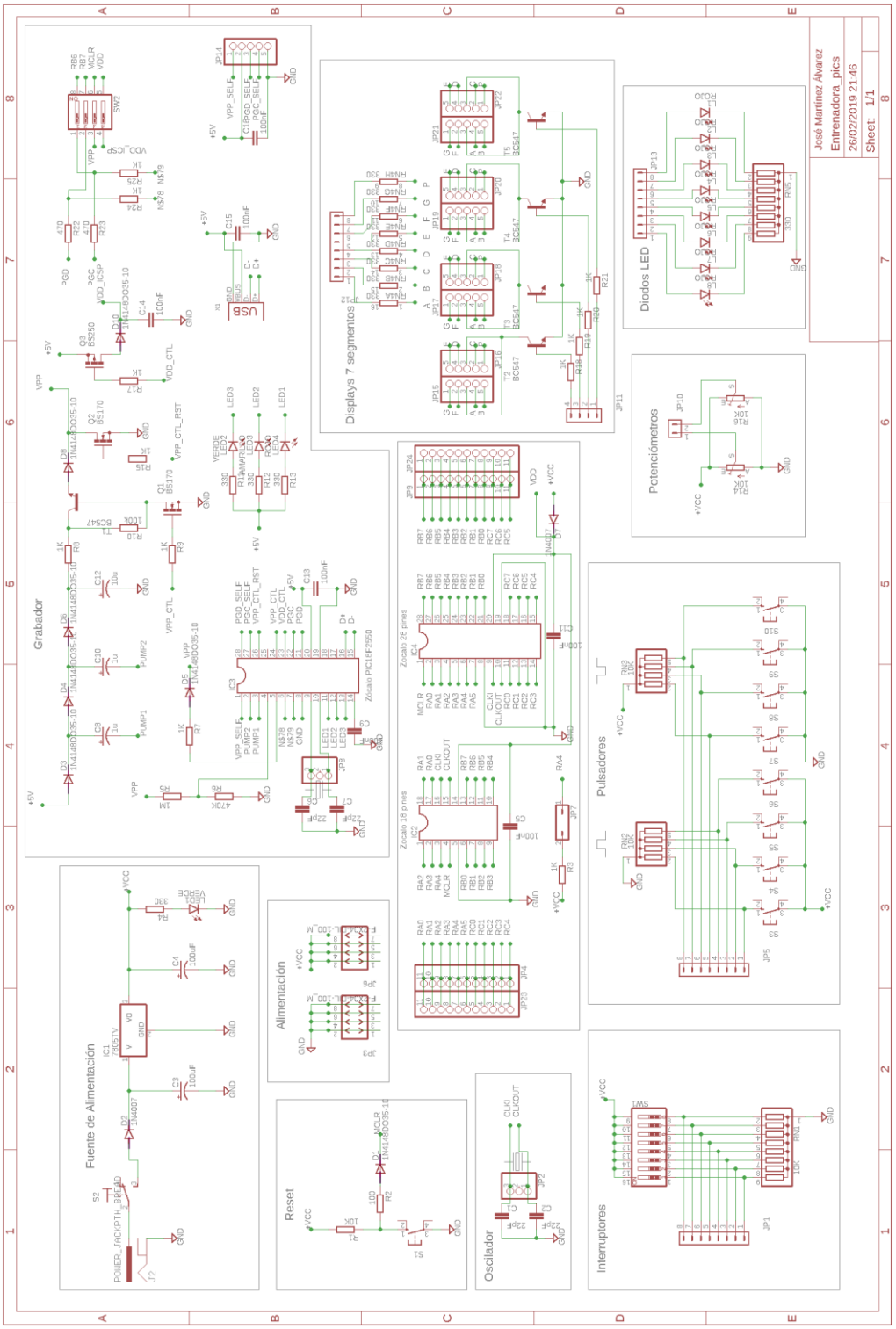


Ilustración 70: Esquemático

Anexo I.II. PCB TOP

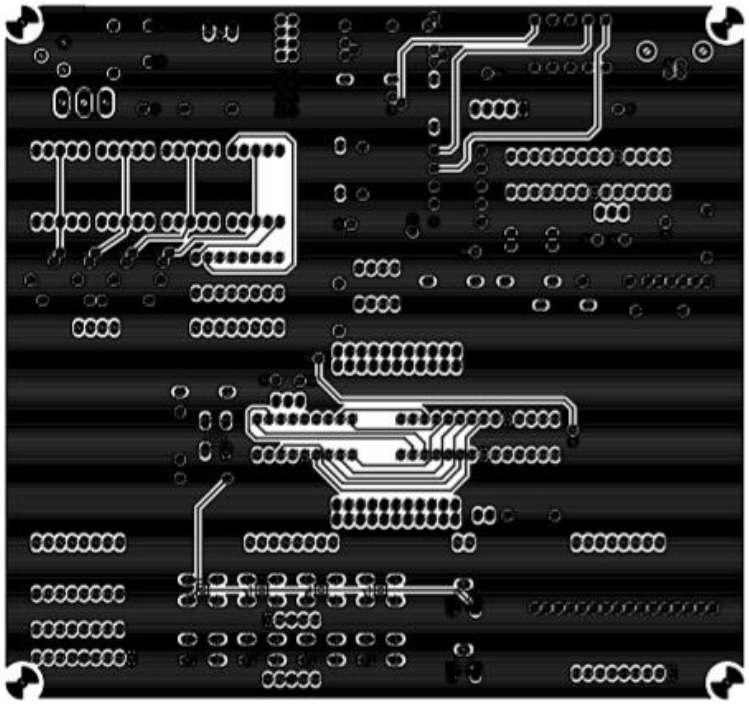


Ilustración 71: PCB TOP

Anexo I.III. PCB BOTTOM

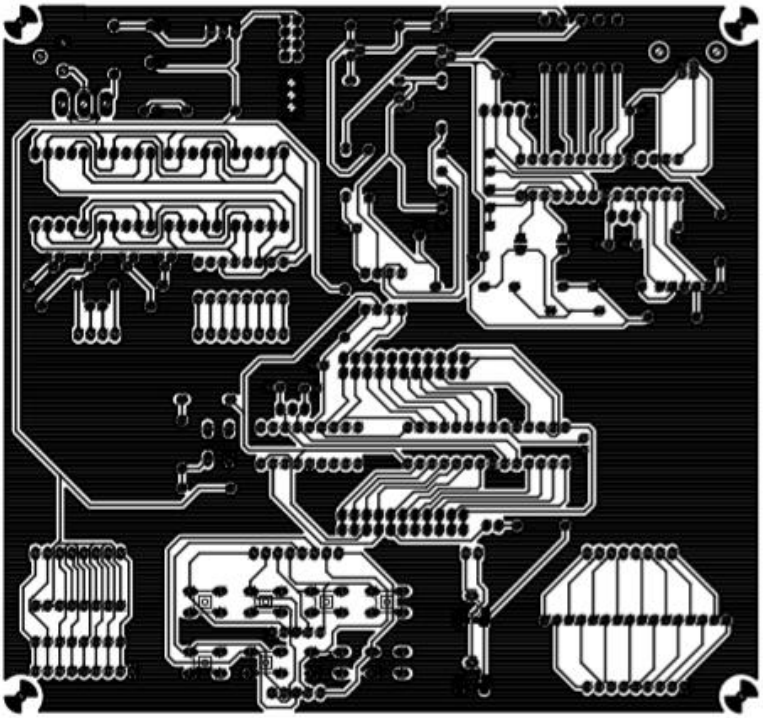


Ilustración 72: PCB BOTTOM

Anexo II. Prácticas de la asignatura

Sesión 2:

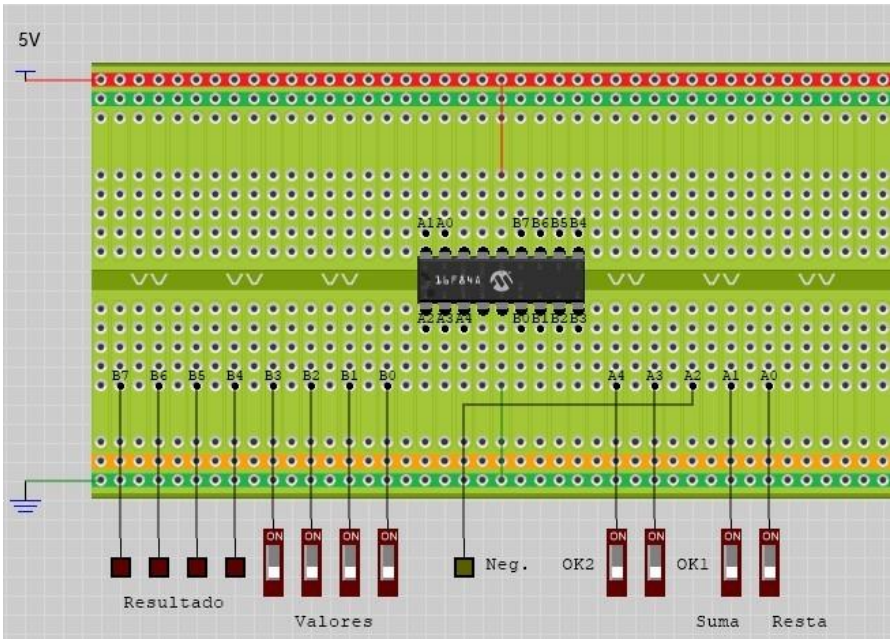
Ejercicio 1.

Utilizando el programa SIM84 comprobaremos el funcionamiento y los resultados de operaciones de suma y resta de los valores 158 y 125 así como de los bits del registro de Estado (dirección 03). Usaremos como variables intermedias cualquiera desde la dirección 1C hasta la 2F. El mismo fichero se usará haciendo en cada caso los modificaciones necesarias para conseguir cada uno de los siguientes casos:

- Sume los valores decimales 158 y 125. Comprobar el resultado y el valor del bit C.
- Reste los valores decimales 158 y 125. Comprobar el resultado y el valor del bit C.
- Reste los valores decimales 125 y 158 realizando el complemento a 2 del valor obtenido. Comprobar el resultado y el valor del bit C.
- Reste los valores decimales 158 y 158. Comprobar el resultado y los valores de los bits Z y C.

Ejercicio 2.

Utilizando el programa Virtualbreadboard construir un sumador/restador de 4 bits, usando el circuito de la figura.

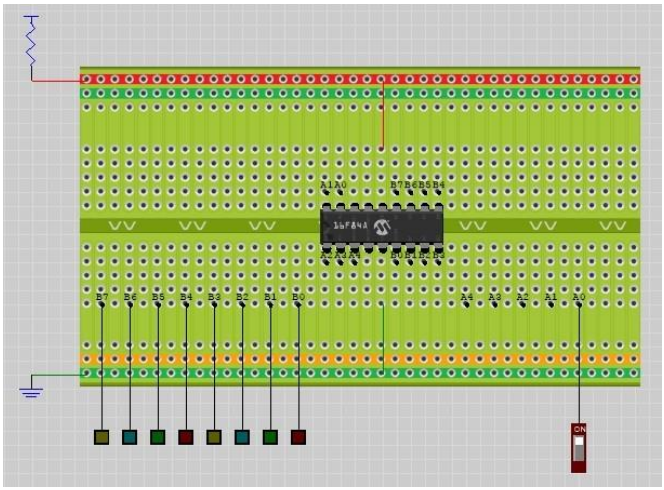


Por los cuatro interruptores se selecciona el valor del primer operando y se pulsará OK1 (pasando de 0 a 1 y de 1 a 0) a continuación, se selecciona el valor del segundo operando y se pulsará OK2. Elegimos la opción de suma o de resta pulsando sus interruptores, mostrando el resultado de la operación en los diodos. ATENCIÓN. La resta se realiza

como: operando1 menos operando 2. Si el resultado es negativo se encenderá el diodo amarillo, y al valor del resultado se le aplicará el "complemento a 2" para obtener el módulo del resultado.

Sesión 3:

Con la aplicación Virtualbreadboard preparar el diseño de conexiones y componentes que se indica en la figura.



Ejercicio 1.

Realizar un programa que mientras se estimule la patilla A0 encienda y apague los Led's de la puerta B, con un parpadeo de 1 segundo (medio segundo encendido y medio apagado). Modificar los valores del bucle de tiempo para acelerar o retrasar el parpadeo.

Ejercicio 2.

Realizar un programa que mientras se estimule la patilla A0 encienda el Led conectado a B0, tras medio segundo lo apague medio segundo y encienda el Led B1, lo apague y encienda el Led B2, y sucesivamente hasta el Led B7, al finalizar que vuelva a realizar el mismo proceso indefinidamente.

Ejercicio 3.

Realizar un programa que mientras se estimule la patilla A0 encienda el Led conectado a B7, tras medio segundo lo apague medio segundo y encienda el Led B6, lo apague y encienda el Led B5, y sucesivamente hasta el Led B0, al finalizar que vuelva a realizar el mismo proceso indefinidamente.

Ejercicio 4.

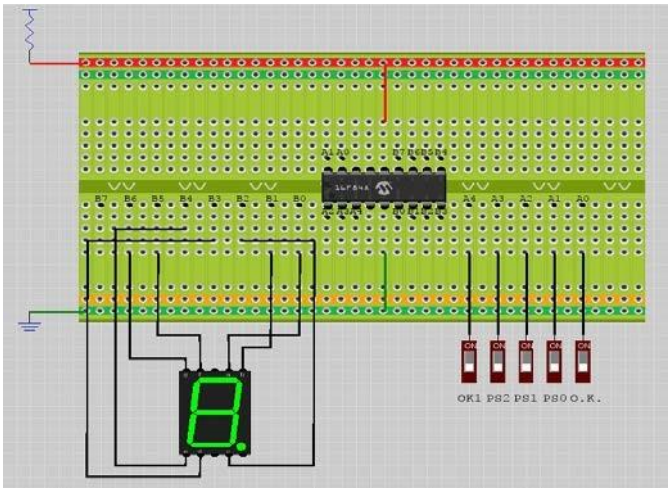
Realizar un programa que mientras se estimule la patilla A0 encienda y apague los Led's hacia la izquierda y al llegar al B7 los encienda y apague hacia la derecha hasta el B0, repitiendo el proceso indefinidamente (efecto "el coche fantástico").

Consideraciones.

Las puertas se configurarán de forma que RB0-7 sean salidas y RA0-4 entradas. Siempre que se apague la patilla A0 el proceso se detiene. Para configurar la temporización, utilizar el conjunto de bucles diseñado en clase de teoría con los valores VAR3=4, VAR2=163 y VAR1=254 para obtener medio segundo de no hacer nada.

Sesión 4:

Con la aplicación Virtualbreadboard preparar el diseño de conexiones y componentes que se indica en la figura.



Ejercicio 1.

Realizar un programa que cada vez que se estimula la patilla A0 (O.K.) comience una cuenta ascendente de 0 hasta F y a continuación descendente de F hasta 0, si se mantiene a uno el interruptor el proceso se repite indefinidamente. El valor de cada incremento/decremento se mostrará en un *display* 8 segmentos de forma intermitente, 0'5 segundos encendido y 0'5 apagado.

Ejercicio 2.

Usando el programa realizado en el ejercicio1, incluir una modificación para variar, mientras se hace la simulación, el valor del Divisor de Frecuencias. Para ello usaremos los interruptores PS0, PS1, PS2 para introducir la combinación de la división de frecuencias deseada y activaremos OK1. Manteniendo a 1 O.K. y OK1 a la vez podremos variar la frecuencia en todo instante.

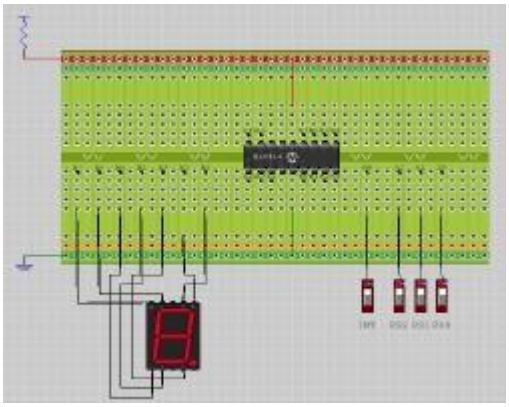
Consideraciones.

Las puertas se configurarán de forma que RB0-7 sean salidas y RA0-4 entradas. Para configurar la temporización, Configurar el TMR0 como *TIMER* con *DF 1:256*. El valor a cargar en TMR0 para la cuenta es 61 en decimal. Cuando se use el SIM84 se debe desactivar el Perro Guardián WDT (Opciones de simulación/Parámetros de simulación).

Sesión 5:

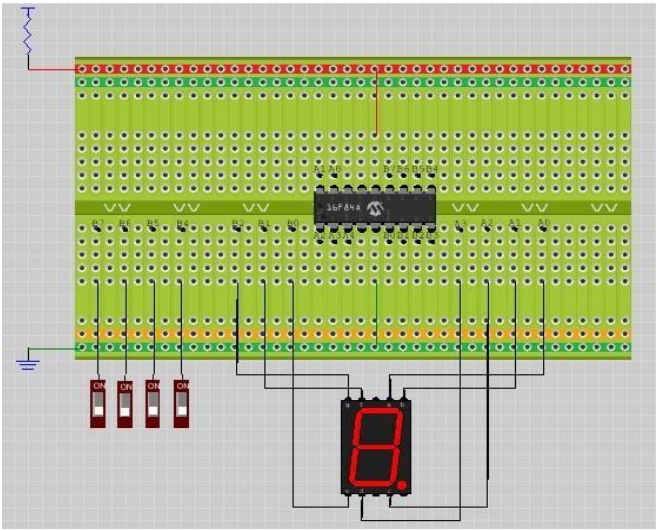
Ejercicio1.

Realizar un programa que comience una cuenta ascendente de 0 hasta F y a continuación descendente de F hasta 0, repitiendo el proceso indefinidamente. El valor de cada incremento/decremento se mostrará en un display 8 segmentos de forma intermitente, 0'5 segundos encendido y 0'5 apagado. Con objeto de comprobar el funcionamiento de la interrupción por estímulo en la patilla RB0, se deberá incluir una rutina de servicio de interrupción que lea los valores PS2, PS1 y PS0 y los cambie en el registro OPTION asociado al TMR0, variando pues la velocidad del parpadeo. Obsérvense bien las conexiones del Display y de los Interruptores.



Ejercicio 2.

Realizar un programa que compruebe el funcionamiento de la interrupción producida por el cambio en el valor en las patillas RB7 a RB4. Para ello se programará un bucle infinito y se esperará a que se produzca la interrupción, la rutina de servicio detectará la patilla que ha cambiado a ON y mostrará su número por el display hasta que se modifique otra distinta. Hay que tener en cuenta que cuando las cuatro patillas están a OFF el display estará apagado, igual que si hay más de una patilla a ON.



Ejercicio 3.

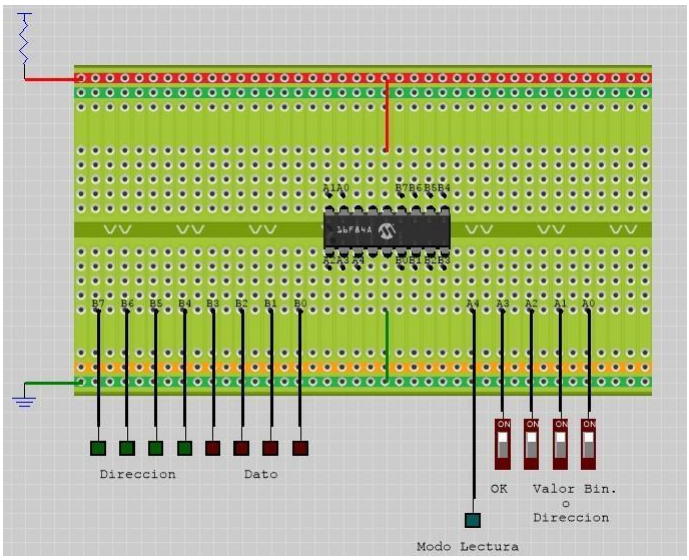
Realizar un programa que compruebe el tratamiento de las interrupciones por cambio en las patillas RB7 a RB4, y la de estímulo por la patilla RB0. De forma que cuando se produzca la primera se encienda el led rojo y con la segunda el led verde, quedando encendido hasta la siguiente interrupción.

Sesión 6:

Ejercicio 1.

Usando la aplicación Virtualbreadboard, realizar un programa para leer y escribir en la memoria EEPROM de datos. Cuando se ponga a uno la patilla A0, el programa escribirá los valores de la secuencia 2, 4, 6, 8…..32 desde la posición 0x10 hasta la posición 0x1F,al finalizar se encenderá un led verde indicando el fin de la escritura. Cuando introduzcamos un 1 por la patilla A4 se leerán las posiciones de memoria desde la 0x10 hasta la 0x1F mostrando su valor por 8 diodos conectados a la puerta B cada 0,5 segundos de forma intermitente como un bucle infinito.

Ejercicio 2.



Con el diseño de la imagen, realizar un programa que escriba y lea valores de la memoria EEPROM de datos sin realizar tratamiento de interrupciones, solo observando el valor del bit EEIF. Usaremos las posiciones 0x00 hasta la 0x07 de la memoria EEPROM. En un primer paso el programa comenzará con la posición 0x00 esperando a introducir un valor por los interruptores RA0-RA2 confirmándolo con el interruptor RA3, de forma automática se incrementarán las direcciones introduciendo valores hasta llegar a la 0x07 y se encenderá el led “Modo Lectura”. A partir de ese momento podemos pedir que muestre el contenido de una dirección que introduzcamos por RA0-RA3 y su confirmación por RA4. En los diodos de la Puerta B se muestran la dirección en verde y el contenido en rojo.

Sensor marcha atrás:

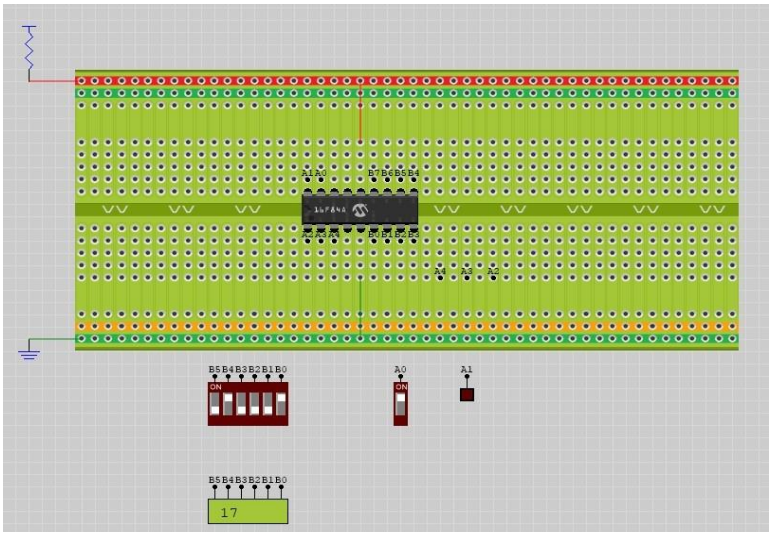
SISTEMAS BASADOS EN MICROPROCESADOR

Ejercicio

Se desea desarrollar un sistema de aviso para aparcamiento de vehículos, de manera que informe al conductor de lo cerca o lejos que está de un obstáculo cuando da marcha atrás. Para ello usaremos un PIC 16F84A a 4 MHz, un sensor de proximidad y un zumbador que generará las señales sonoras. El funcionamiento del sistema es el siguiente. Cuando el conductor activa la marcha atrás, automáticamente se pone en marcha el dispositivo, y en función de si hay un obstáculo suena el zumbador o no. El sensor proporciona al PIC 6 valores binarios que indican la distancia en cm. al posible obstáculo. El zumbador genera un pitido intermitente cuya frecuencia depende de la distancia a la que esté el obstáculo, según la siguiente tabla.

Distancia (cm)	Suena cada
Despejado	Nunca
5-20	0,50 seg
21-40	0,75 seg
41-60	1,00 seg
61-∞	Nunca

En el caso de que el obstáculo se encuentre más lejos de 60 cm, el sistema entenderá que está despejado. Conforme vamos dando marcha atrás y nos aproximemos al obstáculo, se ira incrementado la frecuencia de los pitidos, por lo tanto, el zumbador sonará en función de la lectura del sensor. El zumbador funciona mediante una señal que puesta a 1 indicará que suene y puesta a 0 se desconecta. Se necesitará una entrada para detectar cuando se pone la marcha atrás y cuando deja de estar puesta, ya que en el momento que se cambie de velocidad deja de funcionar el sistema. Para simplificar el problema, supondremos que nunca nos acercamos a menos de 5 cm del obstáculo.



Anexo III. Códigos ensamblador

Ejercicio 2 Sesión 2:

```
                list        P=16F84A
                radix       HEX
                #include    <P16F84A.INC>
OP1             equ         0C      ;Variable donde almacenaremos el primero
                                    ; operando
OP2             equ         0D      ;Variable donde almacenaremos el segundo
                                    ; operando
                org         00      ;Inicio del programa
Inicio          bsf         STATUS,RP0          ;Accedemos al banco 1
                movlw       b'00011011'   ;RA0-RA1 y RA3-RA4 como entradas y
                                          ; RA2 como salida
                movwf       PORTA
                movlw       b'00001111'   ;RB0-RB3 como entradas y RB4-RB7 como
                                          ; salidas
                Movwf       PORTB
                bcf         STATUS,RP0  ;Volvemos al banco 0
Atras           btfss       PORTA,3     ;Comprobamos si hay un uno en RA3
                goto        Atras       ;Si hay un cero volvemos a comprobar RA3
                clrf        PORTA       ;Si hay un uno borramos el contenido de la
                                        ; puerta A
                clrf        PORTB        ;Borramos el contenido de la puerta B
                movf        PORTB,W     ;Movemos el valor del primer operando
                                        ; introducido por la puerta B al acumulador
                andlw       b'00001111' ;Hacemos un and para quedarnos con los
                                        ; valores que nos interesa
                Movwf       OP1         ;Guardamos su valor en OP1
Atrás  1        btfsc       PORTA,3     ;Esperamos a que haya un cero en RA3
                goto        Atras1
Atras2          btfss       PORTA,4     ;Hacemos lo mismo con el segundo
                                        ; operando
                goto        Atras2
                movf        PORTB,W
                andlw       b'00001111'
                movwf       OP2
Atras3          btfsc       PORTA,4
                goto        Atras3
Atras4          btfsc       PORTA,0      ;Comprobamos si hay un cero en RA0
                goto        Suma         ;Si hay un uno vamos a Suma
                btfsc       PORTA,1     ;Si hay un cero comprobamos si hay un cero
en RA1
```

```
            goto    Resta       ;Si hay un uno vamos a Resta
            goto    Atras4      ;Si hay un cero esperamos
                                ; a que haya un uno en RA0 o RA1
Suma        btfsc   PORTA,0     ;Esperamos a que hay un cero en RA0
            Goto    Suma
            movf    OP2,W       ;Movemos el valor de OP2 al acumulador
            addwf   OP1,1       ;Sumamos el valor del acumulador a OP1
            goto    Mostrar     ;Vamos a mostrar
Resta       btfsc   PORTA,1     ;Esperamos a que hay un cero en RA1
            goto    Resta
            movf    OP2,W       ;Movemos el valor de OP2 al acumulador
            subwf   OP1,1       ;Restamos el valor anterior a OP1
            btfsc   STATUS,C    ;Comprobamos si la resta ha sido positiva
            goto    Mostrar     ;Si lo ha sido vamos a mostrar
            comf    OP1,1       ;Si ha sido negativo hacemos el
                                ; complemento a1
            incf    OP1
            bsf     PORTA,2     ;Encendemos el led conectado a RA2
Mostrar     swapf   OP1,W       ;Cambiamos los nibbles de OP1 y movemos
                                ; su valor al acumulador
            movwf   PORTB       ;Mostramos el valor por la puerta A
            goto    Atrás   ;Volvemos al principio
            end
```

- **Ejercicio 4 Sesión 3**

```
            list    P=16F84A
            radix   HEX
            #include <P16F84A.INC>
TMP1        equ     1C
TMP2        equ     1D
TMP3        equ     1E
DES         equ     1F
            org     00
            bsf     STATUS,RP0  ;Cambiamos al banco 1
            clrf    PORTB       ;Puerta B como salidas
            movlw   0xFF        ;Puerta A
            movwf   PORTA       ;como entradas
            bcf     STATUS,RP0  ;Cambiamos al banco 0
            clrf    PORTA       ;Ponemos a cero Puerta A
            clrf    PORTB       ;Ponemos a cero Puerta B
            clrf    DES         ;Ponemos a cero la variable Desplazamiento
            bsf     STATUS,C    ;Ponemos a uno el bit de acarreo
Atras       btfss   PORTA,0     ;Comprobamos el valor de A0
```

```
        goto    Atras   ;Si A0=0 volvemos atrás
        rlf     DES,1   ;Desplazamos hacia la izquierda
        movf    DES,0   ;Movemos el valor al acumulador
        movwf   PORTB   ;Lo movemos a la puerta B
        call    MedioS  ;Llamamos a la rutina MedioS
        clrf    PORTB   ;Ponemos a cero la puerta B
        call    MedioS
        btfss   DES,7   ;Comprobamos el valor del bit 7 de DES
        goto    Atras   ;Si es 0 seguimos desplazando
        clrf    DES     ;Si es 1 Borramos DES
        bsf     DES,7   ;Ponemos a 1 el bit 7
Atras1  btfss   PORTA,0
        goto    Atras1  ;Hacemos lo mismo con secuencia hacia la
                        ; derecha
Bucle2  rrf     DES,1   ;Desplazamos un bit hacia la derecha
        movf    DES,0
        movwf   PORTB
        call    MedioS
        clrf    PORTB
        call    MedioS
        btfss   DES,0   ;Comprobamos el valor del bit 0 de DES
        goto    Atras1  ;Si es 0  seguimos desplazando
        clrf    DES     ;Si es 1 borramos DES
        bsf     DES,0   ;Ponemos a 1 el bit 0
        goto    Atras   ;Volvemos a desplazar a la izquierda
                        ;Rutina para contar medio segundo
MedioS  movlw   d'2'
        Movwf   TMP3
Loop2   movlw   d'163'
        movwf   TMP2
Loop1   movlw   d'254'
        Movwf   TMP1
Loop    decfsz  TMP1,1
        goto    Loop
        decfsz  TMP2,1
        goto    Loop1
        decfsz  TMP3,1
        goto    Loop2
        return
        end
```

- **Ejercicio 2 Sesión 4:**

```
            list        P=16F84A
            radix       HEX
            #include    <P16F84A.INC>
TMP         equ         1C
VAR         equ         1D
FREC        equ         1E

            org         00
            bsf         STATUS,RP0  ;Cambiamos al banco 1
            movlw       b'00000111' ;Valor del registro OPTION
            movwf       TMR0        ;con DF=1:256
            clrf        PORTB       ;Puerta B como salidas
            movlw       0x1F        ;Puerta A como entradas
            movwf       PORTA
            bcf         STATUS,RP0  ;Cambiamos al banco 0
            clrf        PORTB       ;Ponemos a cero la puerta B
            clrf        VAR         ;Ponemos a cero VAR
Sube        btfss       PORTA,0     ;Comprobamos el valor de RA0
            goto        Sube        ;Si RA0 = 0 volvemos atrás
            call        Display     ;Si RA1 = 1 llamamos a la rutina Display
            movwf       PORTB       ;Mostramos el valor a representar por el
                                    ; display
            call        MedioS      ;Llamamos la rutina MedioS
            clrf        PORTB       ;Ponemos a cero la puerta B
            call        MedioS
            incf        VAR,1       ;Incrementamos el contador
            movlw       0x0F        ;Límite superior digito F
            xorwf       VAR,W       ;Comprobamos si ha llegado al digito F
            btfss       STATUS,Z    ;Si ha llegado vamos a Baja
            goto        Sube        ;Si no ha llegado hacemos el proceso
                                    ; contrario
Baja        btfss       PORTA,0
            goto        Baja
            call        Display
            movwf       PORTB
            call        MedioS
            clrf        PORTB
            call        MedioS
            decf        VAR,1
            movlw       0x00        ;Límite inferior digito 0
            xorwf       VAR,W
            btfss       STATUS,Z
```

```
                goto        Baja
                goto        Sube

;Rutina para contar medio segundo
MedioS          movlw       d'10'           ;Numero de desbordamientos
                Movwf       TMP             ;Valor a cargar para 50 mseg.
Jump1           movlw       d'61'
                movwf       TMR0
Jump            btfss       INTCON,T0IF;Comprobamos el desbordamiento del
                                        ; TMR0
                goto        Frecu
                bcf         INTCON,T0IF
                decfsz      TMP,1
                goto        Jump1
                return
;Trozo de código para leer el nuevo valor del divisor de frecuencias
Frecu           btfss       PORTA,4
                goto        Jump
                movf        PORTA,0     ;Leemos el valor de la Puerta A
                movwf       FREC        ;Lo copiamos en la variable FREC
                bsf         STATUS,RP0  ;Cambiamos al banco 1
                rrf         FREC,1              ;Desplazamos a la derecha un bit
                movlw       b'00000111' ;Filtramos los valores que nos interesan
                andwf       FREC,W              ;copiandolos al acumculador
                movwf       TMR0        ;copiamos al registro OPTION
                bcf         STATUS,RP0  ;Cambiamos al banco 0
                goto        Jump

;Rutina para devolver el valor a mostrar por el display
Display         movf        VAR,W
                addwf       PCL,1
                retlw       0x3F        ; Retorna con el codigo del 0
                retlw       0x06        ; Retorna con el código del 1
                retlw       0x5B        ; Retorna con el código del 2
                retlw       0x4F        ; Retorna con el código del 3
                retlw       0x66        ; Retorna con el código del 4
                retlw       0x6D        ; Retorna con el código del 5
                retlw       0x7D        ; Retorna con el código del 6
                retlw       0x07        ; Retorna con el código del 7
                retlw       0x7F        ; Retorna con el código del 8
                retlw       0x6F        ; Retrona con el código del 9
                retlw       0x77        ; Retorna con el código del A
                retlw       0x7C        ; Retorna con el código del B
                retlw       0x39        ; Retorna con el código del C
```

```
        retlw   0x5E        ; Retorna con el código del D
        retlw   0x79        ; Retorna con el código del E
        retlw   0x71        ; Retorna con el código del F
        return
        end
```

- **Ejercicio 2 Sesión 5:**

```
        list    P=16F84A
        radix   HEX
        #include <P16F84A.INC>
VAR     equ     1C
VAR1    equ     1D
        org     00
        goto    Inicio
        org     04          ;Rutina de tratamiento de interrupciones
        goto    Rsi
Inicio  bsf     STATUS,RP0  ;Cambio al banco 1
        movlw   0xF0   ;RB0-RB3 como salidas RB4-RB7 como entradas
        movwf   PORTB
        clrf    PORTA       ;Puerta A como salidas
        bcf     STATUS,RP0  ;Cambio al banco 0
        movlw   b'10001000' ;Habilitamos las interrupciones por cambio
                            ; de estado RB4-RB7
        Movwf   INTCON
        clrf    PORTB       ;Ponemos a cero la puerta B
        clrf    PORTA       ;Ponemos a cero la puerta A
Bucle   goto    Bucle       ;Bucle infinito a la espera de que se produzca
                            ; alguna interrupción
Rsi     bcf     INTCON,RBIF;Ponemos a cero el flag RBIF
        swapf   PORTB,W     ;Intercambiamos los nibbles de la puerta B
        movwf   VAR         ;Copiamos si valor en VAR
        movlw   0x0F        ;Hacemos un and para quedarnos con los
                            ; valores que nos interesa
        andwf   VAR,1
        movf    VAR,1       ;Comprobamos si la VAR = 0
        btfsc   STATUS,Z
        goto    Mostr0
        movlw   d'1'        ;Comprobamos si la interrupcion ha sido en
                            ; RB4
        xorwf   VAR,W
        btfsc   STATUS,Z
        goto    Mostr4
        movlw   d'2'        ;Comprobamos si la interrupcion ha sido en
```

```
                                    ; RB5
        xorwf       VAR,W
        btfsc       STATUS,Z
        goto        Mostr5
        movlw       d'4'            ;Comprobamos si la interrupcion ha sido en
                                    ; RB6
        xorwf       VAR,W
        btfsc       STATUS,Z
        goto        Mostr6
        movlw       d'8'            ;Comprobamos si la interrupcion ha sido en
                                    ; RB7
        xorwf       VAR,W
        btfsc       STATUS,Z
        goto        Mostr7
        clrf        PORTA
        clrf        PORTB
        retfie
Mostr0  clrf        PORTB           ;Ponemos a cero las puertas A y B
        clrf        PORTA
        retfie
Mostr4  movlw       0x06            ;Mostramos el valor 4
        Movwf       PORTB
        movwf       PORTA
        retfie
Mostr5  movlw       0x06            ;Mostramos el valor 5
        Movwf       PORTB
        movlw       0x0D
        movwf       PORTA
        retfie
Mostr6  movlw       0x07            ;Mostramos el valor 6
        movwf       PORTB
        movlw       0x0D
        movwf       PORTA
        retfie
Mostr7  clrf        PORTB           ;Mostramos el valor 7
        Movlw       0x07
        movwf       PORTA
        retfie
        end
```

- **Ejercicio 3 Sesión 5:**

```
        list    p=16F84
        radix   hex

TMROPT  equ     01
estado  equ     03
pata    equ     05
pbtb    equ     06
INTCON  equ     0B
TMP     equ     1C
VAR     equ     1D
VAR1    equ     1E
TMP2    equ     10
TMP1    equ     11
        org     00
        goto    Inicio
        org     04
        goto    RSI

Inicio  bsf     estado,5
        movlw   0xFF
        movwf   pbtb
        clrf    pata
        clrf    TMROPT
        movlw   b'10011000'
        movwf   INTCON
        bcf     estado,5
        clrf    pata
loop    goto    loop

RSI     btfss   INTCON,1
        goto    RB4
        call    Delay
        btfss   pbtb,0
        goto    finint
        bcf     INTCON,1
        bsf     pata,0
        bcf     pata,1
finint  retfie

RB4     bcf     INTCON,0
        bcf     pata,0
        bsf     pata,1
        retfie
```

```
Delay   movlw   0xff
        movwf   TMP2
Jump1   movlw   0x19
        Movwf   TMP1
Jump    decfsz  TMP1,1
        goto    Jump
        decfsz  TMP2,1
        goto    Jump1
        return
        end
```

- **Ejercicio 1 Sesión 6:**

```
        list    P=16F84A
        radix   HEX
        #include <P16F84A.INC>
VALOR   equ     0C
TMP     equ     0D
        org     00
        bsf     STATUS,RP0 ;Cambio al banco 1
        movlw   b'00000111'  ;Divisor de frecuencias 1:256
        movwf   TMR0
        clrf    PORTB      ;Puerta B como salidas
        movlw   b'00010001'  ;RA0 y RA4 como entradas, el resto como
                           ; saludas
        movwf   PORTA
        bcf     STATUS,RP0 ;Cambio al banco 0
Inicio  clrf    PORTA
        clrf    PORTB
        movlw   0x10       ;Direccion 0x10
        movwf   EEADR
        movlw   d'2'       ;Valor 2
        movwf   EEDATA
Atras   btfss   PORTA,0    ;Comprobamos el valor de RA0
        goto    Atras      ;Si RA0 = 0 volvemos atras
Loop    call    Escribe    ;Si RA0 = 1 escribimos el valor
        movlw   0x1F       ;Comprobamos si ha llegado a la ultima
                           ; dirección a escribir
        xorwf   EEADR,W
        btfsc   STATUS,Z   ;Si ha llegado vamos a Sigue
        goto    Sigue      ;Si no incrementamos el valor de la dirección
                           ; y el dato a escribir
        incf    EEADR,1
```

```
            incf    EEDATA,1
            incf    EEDATA,1
            goto    Loop
Sigue       bsf     PORTA,2     ;Encendemos el led conectado a RA2
Sigue1      btfss   PORTA,4     ;Comprobamos el valor de RA4
            goto    Sigue       ;Si RA4 = 0 volvemos atras
            bcf     PORTA,2     ;Si RA4 = 1 apagamos el led
            movlw   0x10        ;Movemos la primera direccion de memoria
                                ; a mostrar
            movwf   EEADR
Loop1       call    Leer        ;Leemos la memoria EEPROM
            movf    EEDATA,W    ;Mostramos el valor por la puerta B
            movwf   PORTB
            call    MedioS      ;Medio segundo
            clrf    PORTB       ;Apagamos los led
            call    MedioS
            movlw   0x1F        ;Comprobamos si ha llegado a la ultima
                                ; direccion
            xorwf   EEADR,W
            btfsc   STATUS,Z
            goto    Sigue1      ;Si ha llegado vamos a Sigue1 y volvemos a
                                ; realizar la lectura
            incf    EEADR,1     ;Si no ha llegado icrementamos la
                                ; posicion de memoria y seguimos
                                ; mostrando los valores
            goto    Loop1

;Rutina de medio segundo
MedioS      movlw   d'10'
            movwf   TMP
Jump1       movlw   d'61'
            Movwf   TMR0
Jump        btfss   INTCON,T0IF
            goto    Jump
            bcf     INTCON,T0IF
            decfsz  TMP,1
            goto    Jump1
            return

;Rutina para escribir en la memoria EEPROM
Escribe     bsf     STATUS,RP0
            bsf     EEDATA,WREN
            movlw   0x55
            movwf   EEADR
```

```
            movlw   0xAA
            movwf   EEADR
            bsf     EEDATA,WR
Jump2       btfsc   EEDATA,WR
            goto    Jump2
            bcf     STATUS,RP0
            return

;Rutina para leer en la memoria EEPROM
Leer        bsf     STATUS,RP0
            bsf     EEDATA,RD
            bcf     STATUS,RP0
            return
            end
```

- **Ejercicio 2 Sesión 6:**

```
            list    P=16F84A
            radix   HEX
            #include <P16F84A.INC>

DATO        equ     1C
ADRESS      equ     1D
VAR         equ     0E
            org     00
            bsf     STATUS,RP0 ;Cambio al banco 1
            clrf    PORTB      ;Puerta B como salidas
            movlw   0x0F       ;RA0-RA3 como entradas RA4 como salida
            movwf   PORTA
            bcf     STATUS,RP0 ;Cambio al banco 0
            clrf    PORTA      ;Ponemos a cero la puerta A
            clrf    PORTB      ;Ponemos a cero la puerta B
            clrf    ADRESS     ;Ponemos a cero la variable ADRESS
            clrf    DATO       ;Ponemos a cero la variable DATO
Loop        swapf   ADRESS,W   ;Intercambiamos los nibbles de ADRESS
            movwf   PORTB      ;Mostramos la direccon por la puerta B
            btfss   PORTA,3    ;Comprobamos el valor de RA3
            goto    Loop       ;Si RA3 = 0 volvemos atras
            movf    PORTA,W    ;Si RA3 = 1 movemos el valor de DATO al
                               ; acumulador
            movwf   DATO
            movlw   b'00000111' ;Nos quedamos con los valores que nos
                               ; interesa
            Andwf   DATO,1
```

```
            movf    DATO,W
            iorwf   PORTB,1

;Rutina para escribir
Escribe     movf    DATO,W
            movwf   EEDATA
            movf    ADRESS,W
            movwf   EEADR
            bsf     STATUS,RP0
            bsf     EEDATA,WREN
            movlw   0x55
            movwf   EEADR
            movlw   0xAA
            movwf   EEADR
            bsf     EEDATA,WR
Wait        btfss   EEDATA,EEIF
            goto    Wait
            bcf     EEDATA,EEIF
            bcf     STATUS,RP0
Loop1       btfsc   PORTA,3
            goto    Loop1
            incf    ADRESS,1
            movlw   0x08    ;       Limite de escritura
            xorwf   ADRESS,W
            btfss   STATUS,Z
            goto    Loop
Loop4       bsf     PORTA,4
            clrf    PORTB
Loop2       btfss   PORTA,3
            goto    Loop2
            movf    PORTA,W
            movwf   ADRESS
            movlw   b'00000111'
            andwf   ADRESS,1
Leer        movf    ADRESS,W
            movwf   EEADR
            bsf     STATUS,RP0
            bsf     EEDATA,RD
            bcf     STATUS,RP0
            swapf   ADRESS,W
            iorwf   EEDATA,W
            movwf   PORTB
Loop3       btfsc   PORTA,3
            goto    Loop3
```

```
        goto    Loop4
        end
```

- **Dado Electrónico:**

```
        list    P=16F84A
        radix   HEX
        #include <P16F84A.INC>
VAR     equ     0C
VAR1    equ     0D
VAR2    equ     0E
        org     00
        goto    Inicio
        org     04              ;Rutina de tratamiento de interrupciones
        goto    Rsi
Inicio  bsf     STATUS,RP0      ;Cambio al banco 1
        Movlw   b'00000001'     ;RA0 como entrada, RA1-RA7 como salidas
        Movwf   PORTB
        movlw   b'11111111'     ;Puerta A como entradas
        movwf   PORTA
        bcf     STATUS,RP0      ;Cambio al banco 0
        movlw   b'10010000'     ;Permitimos la interrupción por activación
                                ; de RB0
        movwf   INTCON
        clrf    PORTA           ;Ponemos a cero las puertas
        clrf    PORTB
        movlw   d'0'            ;Cargamos un cero en el TMR0
        movwf   TMR0
Loop    incf    VAR,1           ;Hacemos un bucle infinito donde vamos
        goto Loop               ; aumentando la variable a la espera de que
                                ; se produzca una interrupción en RB0
Rsi     bcf     INTCON,1        ;Ponemos a cero el flag INTF
Loop1   movf    VAR,W           ;Movemos el valor de VAR a VAR1 para no
                                ; perder su valor
        Movwf   VAR1
        Movlw   d'6'            ;Restamos 6 a VAR hasta llegar a un valor
                                ; negativo, entonces
        subwf   VAR,1           ;mostraremos el valor anterior, el cual será
        btfsc   STATUS,C        ; un numero comprendido entre 1 y 6
        goto    Loop1
        movf    VAR1,W
        call    Display
        movwf   PORTB
        retfie
```

```
Display     addwf     PCL,1
            retlw     b'00001100'   ;Uno
            retlw     b'10110110'   ;Dos
            retlw     b'10011110'   ;Tres
            retlw     b'11001100'   ;Cuatro
            retlw     b'11011010'   ;Cinco
            retlw     b'11111010'   ;Seis
            end
```

- **Ejemplo “1234”:**

```
            list      p=16f84a
            radix     hex
            #include  <P16F84A.INC>
TMP1        equ       0C
TMP2        equ       0D
            org       00
            bsf       STATUS,RP0;Banco 1
            clrf      PORTB         ;Puerta B como salidas
            clrf      PORTA         ;Puerta A como salidas
            bcf       STATUS,RP0;Banco 0
            clrf      PORTB         ;Ponemos a cero la puerta B
            clrf      PORTA         ;Ponemos a cero la puerta A
Bucle       movlw     b'00000001'   ;Activamos el display de las milésimas
            movwf     PORTA
            movlw     b'00000110'   ;Mostramos el número "1"
            movwf     PORTB
            call      Delay         ;Retraso de 5 milisegundos
            movlw     b'00000010'   ;Activamos el display de las centenas
            movwf     PORTA
            movlw     b'01011011'   ;Mostramos el número "2"
            movwf     PORTB
            call      Delay         ;Retraso de 5 milisegundos
            movlw     b'00000100    ';Activamos el display de las decenas
            movwf     PORTA
            movlw     b'01001111'   ;Mostramos el número "3"
            movwf     PORTB
            call      Delay         ;Retraso de 5 milisegundos
            movlw     b'00001000'   ;Activamos el display de las unidades
            movwf     PORTA
            movlw     b'01100110    ';Mostramos el número "4"
            movwf     PORTB
            call      Delay         ;Retraso de 5 milisegundos
```

```
            goto        Bucle       ;Bucle infinito

;rutina para contar 5 milisegundos
Delay       movlw       d'20'
            movwf       TMP2
Loop1       movlw       d'70'
            movwf       TMP1
Loop        decfsz      TMP1,1
            goto        Loop
            decfsz      TMP2,1
            goto        Loop1
            return
            end
```

- **Contador ascendente:**

```
            list        P=16F84A
            radix       HEX
TMR0        equ         01
PCL         equ         02
ESTADO      equ         03
PATA        equ         05
PBTB        equ         06
INTCON      equ         0B
VAR         equ         0C
VAR1        equ         0D
CENT        equ         0E
DECE        equ         0F
UNID        equ         10
TMP         equ         11
TMP1        equ         12
TMP2        equ         13
            org         00
            bsf         ESTADO,5    ;Cambio al banco 1
            movlw       b'00011000' ;RA3-RA4 como entradas y RA0-RA2 como
                                    ; salidas
            movwf       PATA
            clrf        PBTB        ;Puerta B como saludas
            movlw       b'00000111' ;Divisor de frecuencias 1:256
            movwf       TMR0
            bcf         ESTADO,5    ;Cambio al banco 0
            clrf        VAR         ;Ponemos a cero todas las puertas y variables
            clrf        PBTB
            clrf        PATA
```

```
            clrf    CENT
            clrf    DECE
            clrf    UNID
Loop        btfss   PATA,4      ;Comprobamos el valor de RA4
            goto    Loop        ;Si RA4 = 0 volvemos atras
            call    R0.25s
            incf    VAR,1       ;Incrementamos el contador una unidad
            clrf    DECE
            clrf    UNID
            clrf    CENT
            movf    VAR,0       ;Descomponemos el valor actual del
                                ; contador en unidades, decenas y centenas
            movwf   VAR1        ;para poder representar cada digito por los
                                ; displays
            movwf   UNID
Centena     movf    UNID,0
            movwf   VAR1
            movlw   d'100'
            subwf   VAR1,1
            btfss   ESTADO,0
            goto    Decena
            incf    CENT,1
            movf    VAR1,0
            movwf   UNID
            goto    Centena
Decena      movf    UNID,0
            movwf   VAR1
            movlw   d'10'
            subwf   VAR1,1
            btfss   ESTADO,0
            goto    Loop1
            incf    DECE,1
            movf    VAR1,0
            movwf   UNID
            goto    Decena
Loop1       call    Mostrar     ;Mostramos el numero por los displays
            btfss   PATA,4      ;Comprobamos la patilla ra4
            goto    Loop1       ;Si es un 0 se muestra el ultimo valor
            goto    Loop        ;Si es un 1 se incrementa

;Rutina para mostrar los valores por los displays
Mostrar     movf    CENT,0
            call    Display
            movwf   PBTB
```

```
                bsf     PATA,0
                call    Delay
                bcf     PATA,0
                movf    DECE,0
                call    Display
                movwf   PBTB
                bsf     PATA,1
                call    Delay
                bcf     PATA,1
                movf    UNID,0
                call    Display
                movwf   PBTB
                bsf     PATA,2
                call    Delay
                bcf     PATA,2
                return

;Rutina para contar 0.25 segundos
R0.25s          movlw   d'5'
                movwf   TMP
R0.025s         movlw   d'61'
                movwf   TMR0
Bucle           btfsc   INTCON,2    ;Comprobamos si desborda el TMR0
                goto    Seguir      ;Si desborda vamos a Seguir
                call    Mostrar      ;Si no desborda se muestra el numero
                goto    Bucle
Seguir          bcf     INTCON,2
                decfsz  TMP,1
                goto    R0.025s
                return

;Rutina para contar 5 milisegundos necesarios para la multiplexacion
Delay           movlw   d'20'
                movwf   TMP2
Jump1           movlw   d'70'
                Movwf   TMP1
Jump            decfsz  TMP1,1
                goto    Jump
                decfsz  TMP2,1
                goto    Jump1
                return

;Rutina para obtener el valor binario de los digitos a mostrar
Display         addwf   PCL,1
```

```
            Retlw       b'01111110'   ;Cero
            retlw       b'00001100'   ;Uno
            retlw       b'10110110'   ;Dos
            retlw       b'10011110'   ;Tres
            retlw       b'11001100'   ;Cuatro
            retlw       b'11011010'   ;Cinco
            retlw       b'11111010'   ;Seis
            retlw       b'00001110'   ;Siete
            retlw       b'11111110'   ;Ocho
            retlw       b'11001110'   ;Nueve
            retlw       b'10000000'   ;Guión
            end
```

- **Sensor marcha atrás:**

```
            list        P=16F84A
            radix       HEX
TMR0        equ         01
ESTADO      equ         03
PATA        equ         05
PBTB        equ         06
INTCON      equ         0B
VAR         equ         0C
VECES       equ         0D
TMP         equ         0E

            org         00
            bsf         ESTADO,5
            movlw       b'00000110'
            movwf       TMR0
            movlw       0xFF
            movwf       PBTB
            movlw       b'00000001'
            movwf       PATA
            bcf         ESTADO,5
            clrf        PBTB
            clrf        PATA
            clrf        INTCON
            clrf        VECES
            movlw       d'64'
            movwf       PBTB
Loop        bcf         PATA,1
            btfss       PATA,0
            goto        Loop
```

```
Loop1       call        Compara
            movwf       VECES
            movf        VECES,1
            btfsc       ESTADO,2
            goto        Loop
            movlw       b'000111110'
            movwf       PATA
            call        Rutina
            clrf        PATA
            call        Rutina
            goto        Loop

Compara     movf        PBTB,0
            Andlw       b'00111111'
            Movwf       VAR
            Movlw       d'61'
            subwf       VAR,0
            btfsc       ESTADO,0
            goto        Cero
            movlw       d'41'
            subwf       VAR,0
            btfsc       ESTADO,0
            goto        Cuatro
            movlw       d'21
            subwf       VAR,0
            btfsc       ESTADO,0
            goto        Tres
            movlw       d'5'
            subwf       VAR,0
            btfsc       ESTADO,0
            goto        Dos
Cero        retlw       d'0'
Dos         retlw       d'20'   ;d'2'
Tres        retlw       d'40'   ;d'3'
Cuatro      retlw       d'70'   ;d'4'

Rutina      movf        VECES,0
            movwf       TMP
Loop3       movlw       d'61'
            movwf       TMR0
Loop2       btfss       INTCON,2
            goto        Loop2
            bcf         INTCON,2
            decfsz      TMP,1
```

```
        goto    Loop3
        return
        end
```