Universidad Francisco de Vitoria
UFV Madrid

Arquitectura y Organización de Computadores

Sesiones de Laboratorio curso 2019-2020

Olga Peñalba Rodríguez y Daniel León

Arquitectura y Organización de Computadores

Sesiones de Laboratorio curso 2019-2020

Sesión de laboratorio 5

En esta sesión retomaremos el trabajo con la placa USB-PICSchool. En concreto, aprenderemos a manejar el teclado y el display LCD, lo que nos permitirá incorporar estos dos elementos de E/S (Entrada/Salida) básicos a nuestros proyectos. También aprenderemos a manejar los display de 7 segmentos (numéricos) de la placa.

Aunque podrás avanzar "imitando" el código de ejemplo que te voy a proporcionar en esta sesión, es importante para tu aprendizaje que leas y entiendas los fundamentos teóricos que están incluidos en este documento.

Objetivos

Los principales objetivos que se persiguen en esta sesión son:

- Recordar las características del sistema hardware (placa USB PIC School) que se utilizará para el desarrollo de los ejercicios y proyectos. En concreto, aprenderás a conectar y verás el funcionamiento del teclado, el display LCD y el display de 7 segmentos.
- Recordar el proceso completo de creación de programas y proyectos en ensamblador para este sistema: edición – ensamblado – grabación en la placa – verificación.
- Volver a repasar la estructura de la memoria RAM, qué posiciones están libres para los datos del programa y cómo se puede acceder a posiciones en diferentes bancos.
- Experimentar en qué consiste la E/S programada (ya estudiado en la parte teórica de la asignatura).
- Manejar el teclado como dispositivo de entrada de datos.
- Manejar el display LCD y el de 7 segmentos como dispositivos de salida.

Resultados de aprendizaje

Tras la realización de esta sesión de laboratorio, deberás ser capaz de:

- Conectar adecuadamente el teclado al puerto B del PIC
- Conectar adecuadamente la pantalla LCD a los puertos A y B del PIC
- Conectar adecuadamente el display de 7 segmentos al puerto C del PIC

- Seleccionar y configurar el programa de grabación PICKit-2
- Ensamblar y grabar un programa en el PIC insertado en la placa USB PIC School
- Ejecutar y probar un programa cargado en la memoria del PIC insertado en la placa
- Mostrar cadenas de mensajes en la pantalla LCD
- Mostrar el valor de la tecla pulsada en cualquier posición de la pantalla LCD
- Mostrar números en el display de 7 segmentos.

Teoría

Organización de la memoria RAM de datos

El PIC16F886 dispone de una memoria RAM organizada en 4 bancos de memoria (ver la Figura 2-3 del DataSheet, en la página 15). Cada banco contiene 128 bytes. Las primeras posiciones en cada banco están reservadas para los registros especiales SFR (como STATUS, FSR, INDF, etc.). Algunos de estos registros están replicados en todos los bancos (acceden al mismo dato) y lo mismo sucede para algunas posiciones de memoria de usuario. En total, existen 368 bytes disponibles para datos del programa.

El acceso a una posición de la memoria puede realizarse de modo directo, utilizando la dirección correspondiente, o de modo indirecto, mediante los registros FSR e INDF. A continuación se muestra cómo se lleva a cabo la selección del banco para el modo directo (el indirecto lo veremos más adelante).

Selección de bancos de memoria para acceso directo

Se realiza mediante los bits 6 y 5 del registro STATUS, llamados, respectivamente, RP1 y RP0. La siguiente tabla muestra la relación entre el valor de dichos bits y el banco que se selecciona. Por defecto, el banco seleccionado es el 0.

RP1	RP0	BANCO
0	0	0
0	1	1
1	0	2
1	1	3

Ejemplo 1. Si deseamos acceder a la posición 0A0h, hemos de seleccionar previamente el banco 1, como se muestra en la siguiente secuencia de instrucciones

```
bcf     STATUS, RP1
bsf     STATUS, RP0
movf    0xA0, W
...
```

Opcionalmente se puede utilizar la macro del compilador BANKSEL.

De no seleccionar previamente el banco 1, la instrucción **movf** 0xA0, W no accedería a la posición 0A0h de la memoria, sino a la 020h, que es la posición equivalente en el banco 0.

Recuérdese que las direcciones de la memoria RAM son de 9 bits. Todas las direcciones que tienen los últimos 7 bits iguales son equivalentes entre sí (da igual cuál se utilice en el código): representarán una posición en uno de los cuatro bancos, y la posición concreta dependerá del banco que esté seleccionado en el momento del acceso. La siguiente tabla recoge las 3 direcciones equivalentes a la 020h.

DIRECCIÓN HEX.	DIRECCIÓN EN BINARIO	BANCO AL QUE PERTENECE
0x020	00-0100000	0
0x0A0	01-0100000	1
0x120	10-0100000	2
0x1A0	11-0100000	3

Como se ve en la tabla, las cuatro direcciones tienen los últimos 7 bits iguales, y sólo se diferencian en los dos primeros que, precisamente, representan el número de banco al que pertenece esa posición. Cuando las instrucciones se codifican en binario, al ensamblar, no se utilizan 9 bits para representar la dirección de un dato, sino únicamente 7 (los 7 últimos) de ahí que las cuatro direcciones sean indistinguibles cuando la instrucción se ejecuta. Lo que determina la posición exacta a la que se accede es el valor de los bits RP1 y RP0, como se muestra en la siguiente figura, extraída del datasheet.

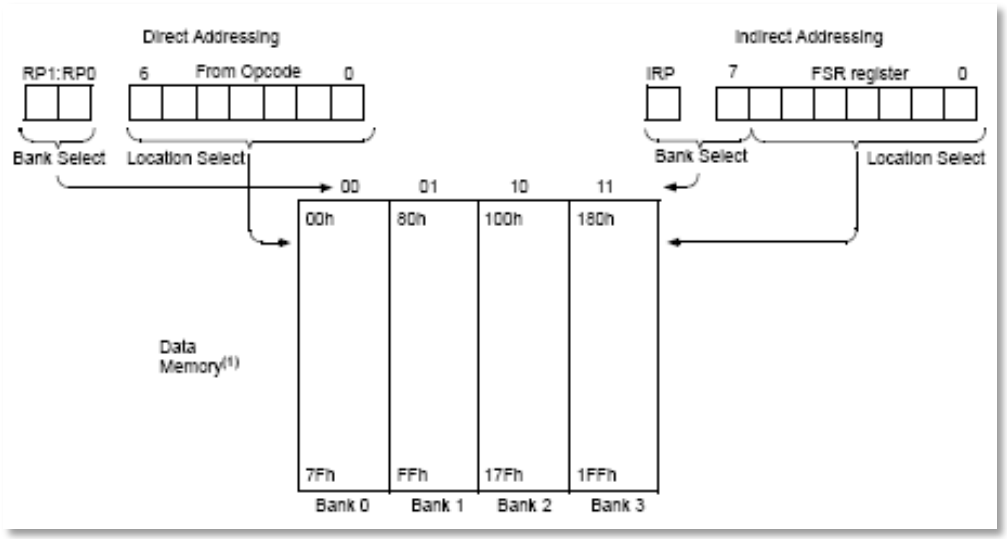


Figura 1. Esquema de selección del banco de memoria según el modo de direccionamiento utilizado

Sin embargo hay una serie de registros especiales, SFR, cuyo acceso es común en todos los bancos, independientemente del estado de RP0 y RP1. A todos los efectos el acceso está replicado y, por tanto, consumen espacio de direccionamiento.

En el PIC16F886, estos registros son **PCL**, **STATUS**, **FSR**, **PCLATH** e **INTCON**. También existen registros de acceso replicado en dos páginas, como **PORTB** o **TRISB**.

Los últimos 16 bytes de cada página, correspondientes a datos de programa que podemos usar, también tienen su acceso replicado en todos los bancos, proporcionando una zona de datos independiente del banco.

Conexión de dispositivos de E/S mediante puertos

En el microcontrolador PIC16F886, los dispositivos de E/S se conectan y se manejan mediante puertos. Existen 3 puertos denominados puerto A, puerto B y puerto C. Cada puerto tiene asociados dos registros especiales principales, llamados **PORT** y **TRIS** (PORTA Y TRISA para el puerto A, PORTB y TRISB para el puerto B, y PORTC Y TRISC para el puerto C). También existe un PORTE que tiene un solo bit de entrada, aunque tiene una función especial (MCLR – Master Clear) y no lo usaremos en nuestras prácticas

El registro **PORT** almacena el dato recibido del dispositivo de E/S o enviado a dicho dispositivo. Este registro se encuentra ubicado en el banco 0 de la memoria RAM.

El registro **TRIS** se utiliza para la configuración de las líneas (bits) conectadas al puerto, eligiendo el sentido de la comunicación: cada bit del registro TRIS con valor 0 configura la línea correspondiente como salida; cada bit con valor 1 configura la línea como entrada. Se encuentra ubicado en el banco 1 de la memoria RAM. En el arranque, todos los puertos se inicializan como entrada (1).

Como hemos visto en sesiones anteriores, los puertos GPIO (General-Purpose Input-Output) son otro periférico más del PIC. En el arranque del microcontrolador, 11 pines no están conectados al periférico GPIO en modo entrada, si no al módulo de conversión Analógico/Digital. Por tanto, aunque configuremos TRIS como entrada, los datos no llegarán al registro PORT no ser que conectemos el periférico a los pines.

En este PIC, eso se realiza desactivando el periférico A/D mediante los registros **ANSEL** y **ANSELH**. Por tanto hay que poner a cero estos dos SFR mediante el siguiente código:

```
BANKSEL ANSEL; Selecciona el banco 3, donde están ANSEL y ANSELH
clrf    ANSEL, f
clrf    ANSELH, f
BANKSEL PORTA ; Selecciona el banco 0, donde está PORTA
```

(Para más información, revisar los apartados 3.2.1 y 3.4.1 del Data Sheet)

Funcionamiento del teclado

La placa USB PICSchool tiene incorporado un teclado matricial de 16 teclas, organizado en cuatro filas y cuatro columnas.

La intersección fila-columna da lugar a una tecla en concreto. Es decir, si se pulsa por ejemplo la tecla 4, supone unir eléctricamente la fila F1 con la columna C0.

Detección de la pulsación de una tecla e identificación de la tecla pulsada.

La rutina software encargada de explorar el teclado (denominada Key_Scan) tiene que determinar qué tecla se ha pulsado. Para ello configura las líneas RB0-RB3 (las columnas) como salidas y RB4-RB7 (las filas) como entradas. Secuencialmente va activando cada una de las columnas al tiempo que lee el estado de las filas. Cuando se detecta que una fila esté activa es porque se pulsó una tecla. Basta conocer qué columna se activó en ese momento para sacar la relación fila-columna que define a cada tecla. Esta tarea conocida como "**barrido del teclado**" ha de repetirse de forma constante y periódica. De esta manera y, a la velocidad de trabajo del PIC, será posible detectar una pulsación en cualquier momento. De hecho, el PIC es tan rápido que es necesario eliminar el rebote de la pulsación.

Uso de la rutina Key_Scan para captura de la tecla pulsada

En primer lugar, para poder utilizar esta rutina, ha de incluirse el archivo "Teclado.inc" en el código fuente, dado que dicho archivo contiene esta y otras subrutinas relacionadas para la identificación de la tecla pulsada, mediante la directiva:

```
INCLUDE "Teclado.inc"
```

Además, y antes de realizar el INCLUDE correspondiente, debe definirse un bloque de etiquetas denominado Key_var. Este bloque define seis etiquetas que son necesarias para el correcto funcionamiento de las rutinas del teclado. Si se declara en la posición 30h, por ejemplo, habría que dejar libres hasta la 36h.

```
Key_var EQU 30h    ; dejar libres hasta las posición 36h
```

La primera de estas etiquetas se denomina Tecla, y es la posición donde la rutina Key_Scan devolverá el valor de la tecla pulsada. Mientras no se detecte ninguna pulsación, el valor que contendrá Tecla será 80h. Así pues, nuestro código debe implementar un bucle de espera a la pulsación de la tecla, que comienza con una llamada a Key_Scan y sigue con la comprobación del valor de Tecla. Si este es 80h, debe repetirse la llamada a Key_Scan; si es distinto de 80h, entonces contendrá el valor de la tecla pulsada (desde el 0 hasta la F).

Este bucle es necesario porque estamos trabajando con un esquema de **Entrada/Salida programada**, es decir, el programa decide en qué momento se captura el valor de cada tecla. Y mientras no se capture, el programa no puede

continuar (el microcontrolador permanece a la espera sin hacer ninguna otra tarea). En un esquema de E/S por interrupciones, el programa continuaría ejecutando otras tareas mientras se produce la pulsación de la tecla y en ese momento sería interrumpido por el teclado para indicar que ya se ha producido la pulsación.

IMPORTANTE: La rutina Key_Scan está diseñada y programada para trabajar en el puerto B, Por ello deberemos conectar el teclado al puerto B tal y como se muestra en la siguiente figura:

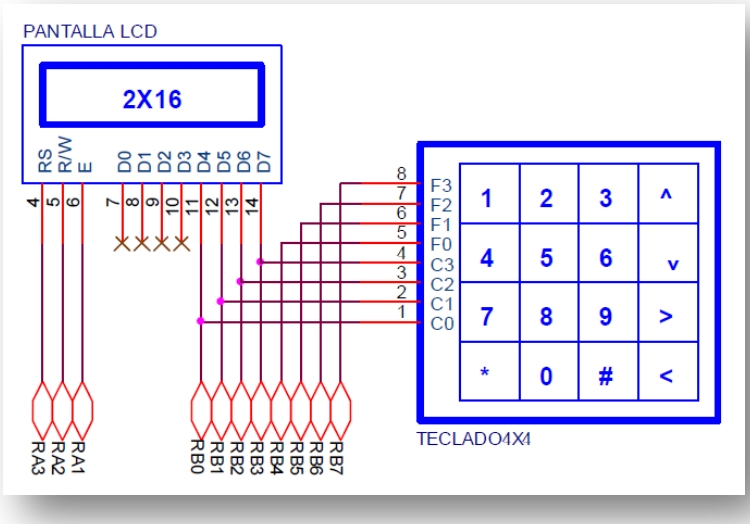


Figura 2. Conexiones del teclado y de la pantalla LCD

Funcionamiento de la pantalla LCD del PIC'Laboratory

La pantalla LCD es el módulo de salida más versátil que incluye la placa PIC'Laboratory.

Nos limitaremos aquí a explicar el modo de uso del dispositivo, sin dar detalles de su funcionamiento interno ni del significado de todos los comandos que se utilizan para iniciarlo, activarlo o configurarlo. El controlador que utiliza es el Hitachi HD44780, por lo que puedes conocer todos los detalles de su funcionamiento en su Data Sheet.

La pantalla LCD contiene dos filas de 16 columnas cada una. Cada posición tiene asignada una dirección, de forma que se puede seleccionar dónde se visualiza cada carácter.

80h	81h	82h	83h	84h	85h	86h	87h	88h	89h	8Ah	8Bh	8Ch	8Dh	8Eh	8Fh
C0h	C1h	C2h	C3h	C4h	C5h	C6h	C7h	C8h	C9h	CAh	CBh	CCh	CDh	CEh	CFh

Cada una de estas 32 posiciones puede visualizar un carácter del juego de caracteres siguiente:

Figura 3. Tabla de símbolos del LCD

Para poder visualizar información sobre el LCD es necesario hacer lo siguiente:

1. Conectar el LCD al PIC según el esquema de conexiones visto en la figura 2.

2. Incluir el archivo "LCD4bitsPIC16.inc" que contiene todas las rutinas de uso del LCD, y declarar previamente las etiquetas Temporal_1, Temporal_2 y Lcd_var (esta necesita 3 posiciones, de forma similar a como ocurría con Key_var).

3. Configurar los puertos A y B como puertos digitales, con las siguientes instrucciones:

```
bsf     STATUS,RP0
bsf     STATUS,RP1              ;Banco 3,
clrf    ANSEL                   ;Puerta A digital
clrf    ANSELH                  ;Puerta B digital
bcf     STATUS,RP1              ;Banco 1
bcf     OPTION_REG,NOT_RBPU ;Habilita cargas pull-Up
bcf     STATUS,RP0              ;Banco 0
```

4. Inicializar el LCD con las siguientes instrucciones:

```
call    UP_LCD      ;Configura puertos para uso del LCD
call    LCD_INI     ;Inicia el LCD
movlw   b'00001100'
call    LCD_REG     ;Enciende LCD
```

5. Para mostrar un carácter cualquier sobre el LCD, basta con situar el cursor en la posición deseada y a continuación hacer una llamada a la rutina LCD_DATO, poniendo primero en W la posición de la tabla de caracteres que se corresponde con dicho carácter. Por ejemplo, si deseamos mostrar una # en la primera posición de la segunda fila, haríamos lo siguiente:

```
movlw   0x01
call    LCD_REG     ;Borra pantalla y sitúa el cursor en
                    ;la primera posición
movlw   0xC0        ;Sitúa el cursor en la segunda fila
call    LCD_REG

movlw   0x23        ;Código del símbolo # en la tabla
call    LCD_DATO
```

6. Para visualizar un número hexadecimal (del 0 a la F) que es variable y que se encuentra almacenado, por ejemplo, en la posición Num, haríamos lo siguiente:

   a. sabemos que los caracteres del 0 al 9 están almacenados en la tabla a partir de la posición 30, de forma que la 30 se corresponde con el 0, la 31 con el 1, la 32 con 2 y así sucesivamente hasta el 39 que se corresponde con el 9. Luego para saber la posición del símbolo de Num basta con calcular 30+Num y llamar la LCD_DATO con el resultado de esa suma. Esto se hace con las siguientes instrucciones

```
movf    Num,W
addlw   0x30
call    LCD_DATO
```

   b. sabemos que los caracteres de la A a la F están almacenados en la tabla a partir de la posición 41, de forma que la 41 se corresponde con la A, la 42 con la B, la 43 con C y así sucesivamente hasta el 46 que se corresponde con la F. Luego en este caso, para saber la posición del símbolo de Num hay que sumarle Num a 37 (37 + A en hexadecimal resulta 41, 37 + B es 42 y así sucesivamente).

```
movf    Num,W
addlw   0x37
call    LCD_DATO
```

c. Luego para mostrar una variable que contiene un dígito hexadecimal, primero tenemos que averiguar si el dígito es menor o igual que 9 o mayor, para calcular adecuadamente su posición en la tabla según lo visto antes. Esto se hace con las siguientes instrucciones (observad que tras mostrar el símbolo en el display se hace un pequeño retardo; eso es especialmente importante cuando se utilizan simultáneamente LCD y Teclado):

```
VisualizarHex   movf    Num,W
                movwf   Dato1
                sublw   .9
                btfss   STATUS,C        ;Es mayor que 9
                goto    MayorQue9       ;Si
                movf    Tecla,W         ;No
                addlw   0x30            ;Calcular posición (0 a 9)
                call    LCD_DATO        ;Visualizar sobre el LCD
MayorQue9       movf    Num,W
                addlw   0x37            ;Calcular posición (A a F)
                call    LCD_DATO        ;Visualiza sobre el LCD
```

Opcionalmente se puede utilizar una tabla de *look-up*, parecida a una tabla de salto, pero rellena de instrucciones **retlw**. Este método, que simula un acceso indirecto, es especialmente útil cuando los datos que hay que retornar no son lineales, como veremos en el caso del display de 7 segmentos después. El PIC16 también dispone de un modo de direccionamiento indirecto que puede aplicarse en estos casos, lo veremos en próximas sesiones.

Si se va a realizar un programa que tiene que mostrar varios números variables en el LCD, lo ideal es convertir el anterior fragmento de código en una rutina, que tome como entrada la variable Num. Con esto se evita repetir el mismo fragmento múltiples veces, incrementando el tamaño del programa.

7. Por último, para simplificar la visualización de mensajes fijos en el programa, podemos crear una tabla con todos los mensajes que se vayan a mostrar por el display, según se muestra en el siguiente ejemplo (los mensajes van entrecomillados y terminan con el carácter 0x00, que indica el fin de la cadena; no pueden tener más de 16 caracteres):

```
Tabla_Mensajes  movwf   PCL
Mens_1          dt      "Mensaje 1",0x00
Mens_2          dt      "Mensaje 2",0x00
Mens_3          dt      "Mensaje 3",0x00
```

Esta tabla de mensajes, junto con la siguiente rutina, permiten visualizar de forma sencilla cadenas de caracteres sobre el display.

```
;Mensaje: Esta rutina visualiza en el LCD el mensaje cuyo inicio está indicado
;en el acumulador. El fin de un mensaje se determina mediante el código 0x00

Mensaje     movwf   Temporal_1      ;Salva posición de la tabla
Mensaje1    movf    Temporal_1,W    ;Recupera posición de la tabla
            call    Tabla_Mensajes  ;Busca caracter de salida
            movwf   Temporal_2      ;Guarda el caracter
            movf    Temporal_2,F
            btfss   STATUS,Z        ;Mira si es el último
            goto    Mensaje2
            return
Mensaje2    call    LCD_DATO        ;Visualiza en el LCD
            incf    Temporal_1,F    ;Siguiente caracter
            goto    Mensaje1
```

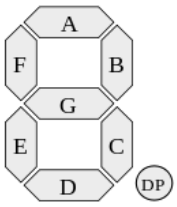
Cada vez que se quiera mostrar un mensaje se llamará a esta rutina de la siguiente forma:

```
            movlw   0x01
            call    LCD_REG         ;Borra pantalla y sitúa el cursor en
                                    ;la primera posición
            movlw   Mensaje1
            call    Mensaje         ;Muestra el mensaje Mensaje1 en el LCD
```

Funcionamiento del display de 7 segmentos del PIC'Laboratory

Los displays de siete segmentos son una solución barata y resistente para mostrar información númerica y, con imaginación, textual. Se trata de 7 leds colocados estratégicamente para formar un "8", más un led de punto decimal. Cada uno de los leds del "8" se denomina segmento, y se denomina con una letra, de la "a" a la "g".

Disposición de los leds en el display de código de 7 segmentos.



	h	g	f	e	d	c	b	a	hex value	
0	0	0	1	1	1	1	1	1	3F	C0
1	0	0	0	0	0	1	1	0	06	F9
2	0	1	0	1	1	0	1	1	5B	A4
3	0	1	0	0	1	1	1	1	4F	B0
4	0	1	1	0	0	1	1	0	66	99
5	0	1	1	0	1	1	0	1	6D	92
6	0	1	1	1	1	1	0	1	7D	82
7	0	0	0	0	0	1	1	1	07	F8
8	0	1	1	1	1	1	1	1	7F	80
9	0	1	1	0	1	1	1	1	6F	90
A	0	1	1	1	0	1	1	1	77	88
b	0	1	1	1	1	1	0	0	7C	83
C	0	0	1	1	1	0	0	1	39	C6
d	0	1	0	1	1	1	1	0	5E	A1
E	0	1	1	1	1	0	0	1	79	86
F	0	1	1	1	0	0	0	1	71	8E

Para mostrar información en el display sólo tenemos que encender los leds adecuados del display. La tabla de la derecha muestra los valores necesarios para los caracteres hexadecimales. Un led, dependiendo de cómo está conectado, puede encenderse con un nivel lógico 1 (como los leds azules de la placa) o con un nivel lógico 0. Este último es el caso de los displays de 7 segmentos que hay en la placa, lo que se denomina "ánodo común". Por tanto, tendremos que elegir la columna azul de la tabla como valor para mostrar el dígito deseado.

Notas de conexión

Si vas a usar el display de 7 segmentos, puedes conectarlo a PORTC, configurado como salida, según la siguente tabla:

PORTC	C7	C6	C5	C4	C3	C2	C1	C0
D7Seg	dp	g	f	e	d	c	b	a

Tendrás también que conectar el ánodo del display que elijas (conector "DEC. UNI." bajo los displays) a VCC. Es recomendable conectarlo al interruptor E3, para poder encender y apagar el display. En este curso no vamos a usar multiplexación de señal del ánodo del display, para usar más de un dígito de forma aparentemente simultánea.

Desarrollo

Vamos a empezar probando el programa TecladoLCD.asm que está disponible en el Aula Virtual. Descárgalo, junto con los archivos P16F886.inc, Teclado.inc y LCD4bitsPIC16.inc.

Para ello, solo tienes que seguir los pasos ya realizados en la Sesión 1.

Ejercicios adicionales

Antes de integrar estos dispositivos de entrada y salida en tu proyecto, realiza algunas pruebas adicionales sobre este código sencillo. Por ejemplo:

- Intenta mostrar las teclas en una posición diferente (en el centro de la segunda fila).
- Muestra un mensaje de bienvenida antes de empezar a visualizar las teclas
- Captura dos teclas y realiza la suma de ambos valores
- Pide que se introduzca un dígito del 0 al 9 y muestra un mensaje de error si se pulsa otra tecla diferente
- Ahora muestra también el dígito introducido por el display de siete segmentos.
- (Extra) Haz un contador de 0-F sobre el display de 7 segmentos