Diseño Basado en Microprocesadores
Práctica 12
Aplicación con GPIO, timers, ADC e interrupciones

Índice

---

1. Objetivos

En esta práctica desarrollaremos una aplicación que use los periféricos del microcontrolador estudiados hasta ahora.

2. Uso de una resistencia LDR

En esta práctica usaremos una resistencia dependiente de la luz o LDR. La LDR se suministra en un pequeño módulo que incluye también una resistencia fija de 10 kΩ que con la LDR completa un divisor de tensión (ver la figura 1).

La tensión recogida en el divisor de tensión será

$$V_S = \frac{R_{LDR}}{R_1 + R_{LDR}} V_{CC} \quad (V)$$

Resolviendo para R_{LDR}

$$R_{LDR} = \frac{R_1 V_S}{V_{CC} - V_S} \quad (\Omega)$$

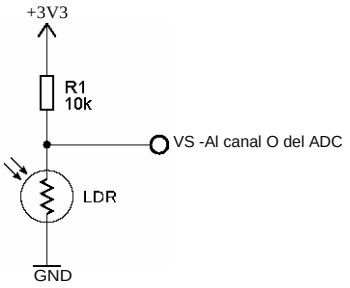

Figura 1: Esquema del módulo sensor con LDR.

Supondremos que, para la LDR usada, la relación entre su resistencia y el nivel de iluminación o iluminancia en luxes, *Ev,* es

$$Ev = 1.25 \times 10^7 (R_{LDR})^{-1.41} \quad (lx)$$

Para calcular la iluminancia recibida por la LDR usaremos la función

- float32_t ldr_traducir_a_iluminancia (float32_t V_s)

que recibe como argumento la tensión de salida del módulo sensor. Esta función pertenece al fichero fuente ldr.c.

Conectaremos la salida del sensor al canal O del ADC del microcontrolador. El cuadro siguiente muestra las conexiones que hay que realizar entre el módulo LDR y el conector J5 de la tarjeta del microcontrolador. Se trata de las mismas conexiones se hicieron con el módulo NTC en una práctica anterior.

Cuadro 1: Conexiones del módulo sensor LDR al conector J5.

Terminal del sensor	**Señal del** sensor	**Pin de J5**
Negro	GND	1
Rojo	+3.3V	2
Blanco	Salida (Vs)	23

3. Generación de sonido

En la tarjeta LPC4088 Developer's Kit hay un pequeño altavoz que está conectado al pin P0[26] del microcontrolador a través de un amplificador (figura 2). Esto permite generar sonidos de forma sencilla.

Hay dos formas de usar el pin P0[26] para manejar el altavoz. Un método consiste en usar el pin con su función normal de GPIO configurándolo como salida. Para que el altavoz produzca un sonido audible, se genera una señal cuadrada en el pin cambiando rápidamente su estado. La frecuencia de la señal cuadrada debe estar dentro del rango de frecuencias audibles, que está entre 20 Hz y 20 kHz, aunque no todas las personas pueden oír los tonos más graves entre 20 Hz y 50 Hz o los más agudos entre 16 kHz y 20kHz . Funcionando como salida digital, el pin no puede generar directamente una forma de onda concreta sino sólo ondas cuadradas o rectangulares, así que no es fácil modificar el timbre del sonido. Por ello, esta forma de manejar al altavoz suele usarse para generar pitidos o melodías sencillas.

Otro método consiste en configurar el pin P0[26] en su función de salida del convertidor digital/analógico (DAC) del LPC4088. Usando el DAC pueden enviarse

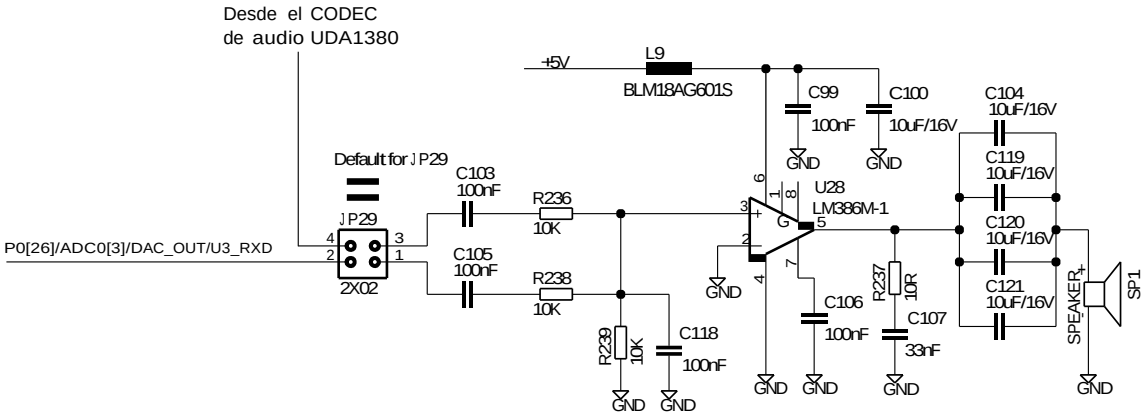

Figura 2: Esquema del amplificador de audio y altavoz de la tarjeta.

al altavoz formas de onda arbitrarias para generar sonidos más complejos. (La tarjeta LPC4088 Developer's Kit también cuenta con un codec de audio UDA1380 conectado al interfaz I2S del microcontrolador. Esto permite generar y capturar sonido estéreo de buena calidad a través de los conectores jack de audio de la tarjeta.)

Para generar pitidos a través del altavoz usaremos la siguiente función:

- void sonido_emitir_pitido(uint32_t frecuencia, uint32_t duracion_ms)

donde el argumento frecuencia es la frecuencia en Hz del pitido y el argumento duracion_ms es el tiempo en milisegundos que debe durar.

4. Manejo de un teclado matricial

Cuando se desea conectar a un microcontrolador un número considerable de pulsadores o interruptores es común hacerlo en forma de matriz para reducir el número de pines GPIO necesarios.

En la figura 3 se muestra el aspecto del teclado que usaremos en la práctica.

Como puede verse en la figura 4, el teclado consiste internamente en cuatro líneas de fila y cuatro líneas de columna con cada pulsador conectado a una fila y a una columna. Así, en lugar de los dieciséis pines GPIO que serían necesarios si los

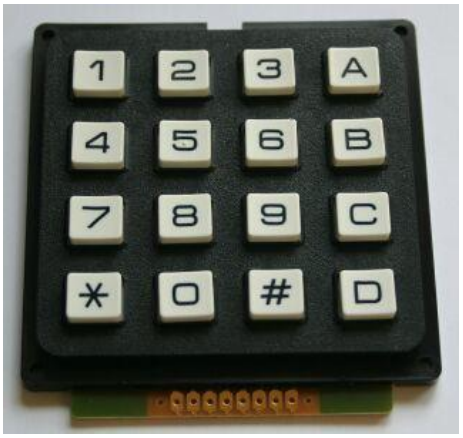

Figura 3: Teclado usado en la práctica.

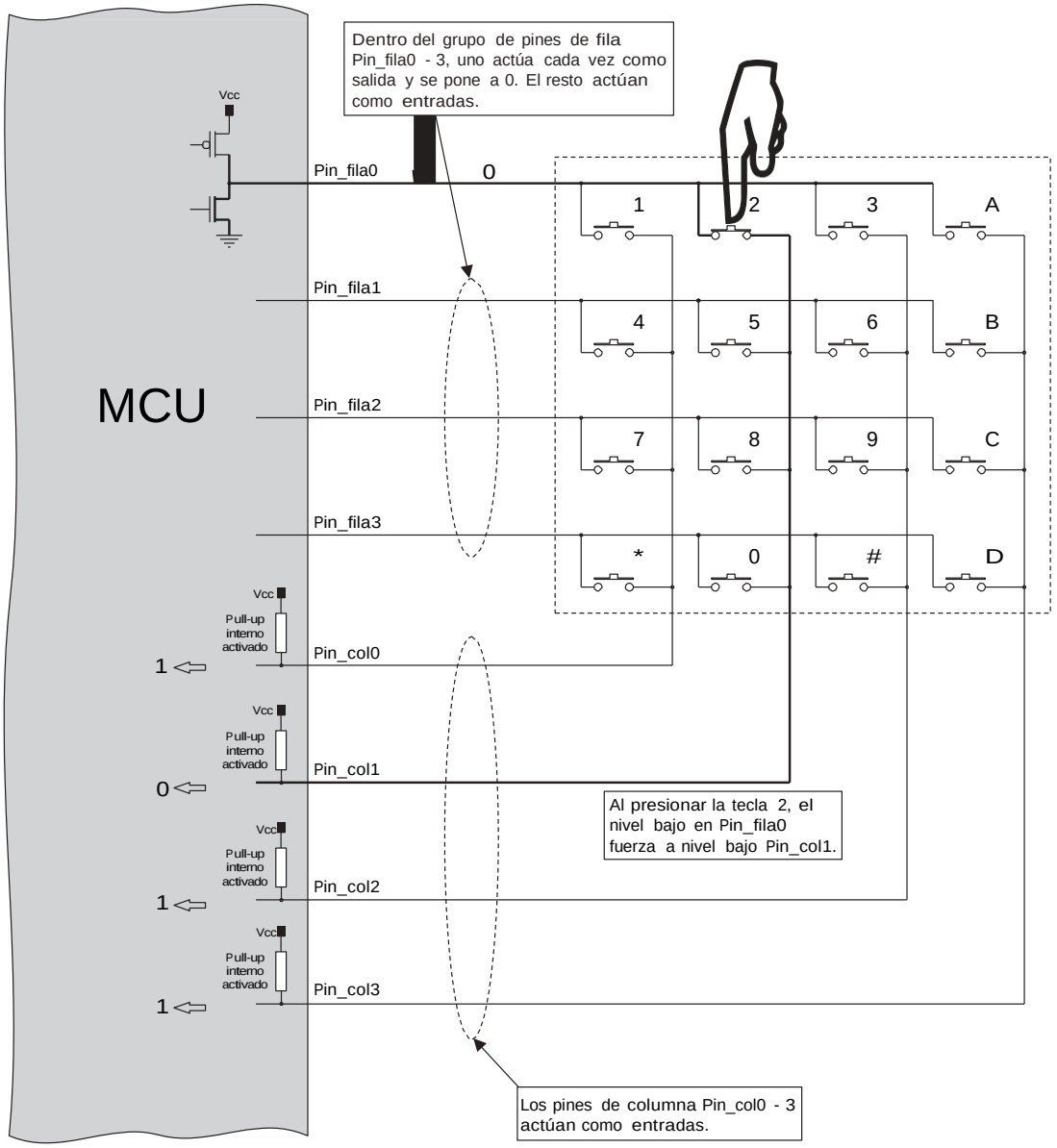

Figura 4: Método de barrido por filas.

pulsadores se conectasen de forma independiente, solamente se requerirán ocho. Sin embargo, esto conllevará complicar el software de lectura del teclado.

Existen fundamentalmente dos métodos para identificar una pulsación en el teclado. El primero consiste en activar sucesivamente cada uno de los pines de fila como salida y leer a continuación los pines de columna (también pueden intercambiarse los papeles de filas y columnas pero esto es básicamente lo mismo). Otro método consiste en activar primero todos los pines de fila como salidas y leer los pines de columna y, seguidamente, activar todos los pines de columna como salidas y leer las filas. En esta práctica se usará el primer método, que denominaremos método de “barrido por filas”.

4.1. Barrido por filas

Como se ha dicho, este método consiste en realizar un barrido fila a fila, leyendo cada vez las columnas. En la figura 4 vemos que las filas y columnas del teclado están conectadas a ocho pines del microcontrolador que se han designado de forma genérica como Pin_fila0-3 y Pin_col0-3, respectivamente. Los pines de columna tienen activadas sus resistencias de pull-up internas. En caso de que los pines conectados a las columnas no tengan la posibilidad de activar pull-ups internos deberán conectarse resistencias de pull-up en el exterior. Debido a la presencia de las resistencias de pull-up, todos los pines de columna se encontrarán a nivel alto mientras no haya una acción externa que los fuerce a nivel bajo.

El proceso de barrido comienza con la configuración como entradas de todos los pines excepto el pin Pin_fila0, que se configura como salida y además se pone a nivel bajo. A continuación, se leerán los pines de columna. Si alguna de las teclas de la fila 0 está presionada, establecerá una conexión entre el Pin_fila0 y el correspondiente pin de columna, con lo que dicho pin de columna será forzado a nivel bajo. Por tanto, al leer los pines de columna encontraremos un cero en uno de ellos y habremos encontrado una pulsación en la fila 0. En la figura 4 vemos el ejemplo de pulsación de la tecla '2', que "trasladará" el nivel bajo de la línea Pin_fila0 al pin Pin_col1. Seguidamente, el pin Pin_fila0 se configurará como entrada.

Si no se está pulsando ninguna tecla de la fila 0 no se encontrará ningún pin de columna a 0. Si es así, se procederá a configurar el pin Pin_fila1 como salida y a ponerlo a cero, repitiendo el proceso de lectura de los pines de columna en busca de algún cero. Mientras no se encuentre una pulsación, el proceso se repite igualmente con las filas 2 y 3. Si en ningún momento se encuentra un 0 en las líneas de columna concluiremos que no se está pulsando ninguna tecla.

4.2. Rebotes de pulsadores e interruptores

Aunque un pulsador o interruptor es en apariencia un dispositivo muy sencillo, su lectura desde un microcontrolador puede complicarse debido a los rebotes de sus contactos cada vez que el pulsador se pulsa o se suelta o, en el caso de un interruptor, pasa de una posición a otra.

En la figura 5 vemos un oscilograma que muestra el efecto de la pulsación de un pulsador sujeto en reposo a nivel alto por una resistencia de pull-up. La traza superior

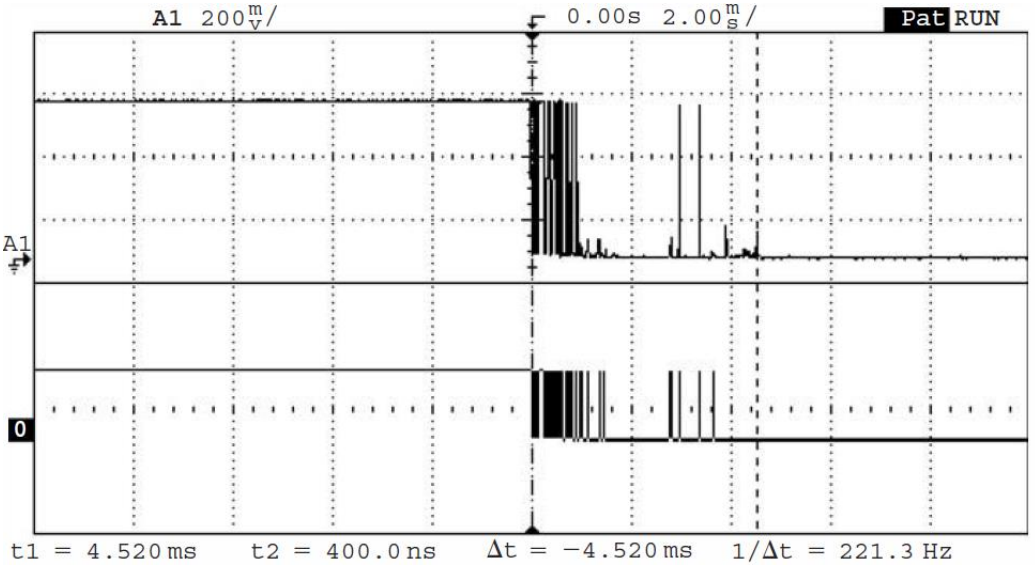

Figura 5: Rebotes de un pulsador.

representa la tensión en el terminal del pulsador unido al pull-up. Como vemos, cuando el pulsador se pulsa la tensión no cambia de forma "limpia" de nivel alto a bajo sino que se producen múltiples transiciones entre niveles así como pulsos de ruido de pequeña amplitud. Esto se debe a que los contactos del pulsador tardan un tiempo en asentarse y crear un camino de baja resistencia a su través. En la traza inferior se muestra como sería percibido el nivel lógico del pin desde el interior del microcontrolador. El problema es que los múltiples cambios que se perciben en la entrada pueden hacer que una única pulsación se interprete desde el programa como una sucesión de acciones de pulsación y liberación del pulsador.

Para reducir el efecto de los rebotes de un pulsador puede añadirse un condensador en paralelo con él, aunque esto no suele bastar para eliminarlos totalmente. Existen otras soluciones hardware más eficaces pero también más complejas y costosas por lo que, en general, se recurre a realizar un filtrado de rebotes en el software.

4.3. Pines usados para conectar el teclado

Los pines del microcontrolador que usaremos para conectar el teclado se muestran en el cuadro 2. Se han elegido pines de los puertos 0 y 2 por su capacidad de generar interrupciones GPIO, lo cual puede usarse para reducir el tiempo que la CPU debe dedicar a atender el teclado. La razón de no haber elegido pines consecutivos es que la mayoría de los pines del microcontrolador ya se usan en la tarjeta de desarrollo para otras funciones incompatibles con la conexión del teclado.

Cuadro 2: Conexiones entre el teclado 4x4 y el microcontrolador.

Teclado	Microcontrolador	Pin conectores	Color cable
Fila 0	P0[5]	J5-16	Negro
Fila 1	P0[7]	J5-17	Blanco
Fila 2	P0[9]	J5-18	Gris
Fila 3	P0[21]	J5-22	Morado
Columna 0	P2[0]	J4-5	Azul
Columna 1	P2[1]	J4-7	Verde
Columna 2	P2[14]	J5-45	Amarillo
Columna 3	P2[19]	J5-46	Naranja

4.4. Funciones para manejar el teclado

Para manejar el teclado matricial partiremos de un conjunto de funciones ya desarrolladas. Con el fin de analizar distintas formas de concretar el método de barrido por filas y el filtrado de rebotes, tendremos disponibles tres implementaciones de las funciones de lectura en los ficheros:

- teclado_4x4_sondeo_basico.c
- teclado_4x4_interrupcion_timer.c
- teclado_4x4_interrupciones_gpio_y_timer.c

Las tres versiones comparten el mismo fichero de cabecera teclado_4x4.h. Las funciones básicas para acceder al teclado son:

- void tec4x4_inicializar(void): Inicializa el sistema de lectura del teclado. Esta función debe ser llamada antes de usar cualquier otra función del teclado.
- char tec4x4_leer(void): retorna el código ASCII de la tecla que se esté pulsando o 0 si no se está pulsando ninguna.
- char tec4x4_esperar_pulsacion(void): espera hasta que no se esté pulsando ninguna tecla y, a continuación, espera a que se pulse una tecla. La función retorna el código ASCII de la tecla pulsada.

En los ficheros teclado_4x4_interrupcion_timer.c y teclado_4x4_interrupciones_gpio_y_timer.c se ha añadido un buffer de teclado que permite almacenar varias pulsaciones (diez por defecto) hasta que la aplicación las requiera. De esta forma se evita perder pulsaciones en caso de que la aplicación no atienda el teclado durante un tiempo. En estos ficheros, las funciones tec4x4_leer y tec4x4_esperar_pulsacion extraen las pulsaciones desde el buffer del teclado. Además, se tienen las siguientes funciones adicionales:

- void tec4x4_vaciar_buffer(void): Vacía el buffer del teclado.
- uint32_t tec4x4_caracteres_en_buffer(void): retorna el número de caracteres almace- nados actualmente en el buffer.
- void tec4x4_leer_cadena(char *ptr_buffer, uint32_t tamano_buffer): lee una cadena de caracteres del teclado. La tecla '#' se usa como ENTER. El argumento ptr_buffer señala al buffer donde se almacenará la cadena. La cadena se asegura de terminar la cadena con un carácter nulo. El argumento tamano_buffer indica longitud del buffer en bytes. La función sólo almacenará en el buffer los (tamano_buffer – 1) primeros caracteres que se pulsen. El tamaño del buffer debe ser como mínimo 1 para tener espacio al menos para el terminador.
- void tec4x4_leer_cadena_numeros(char *ptr_buffer, uint32_t tamano_buffer): igual que la anterior pero restringe la entrada a caracteres de números.

5. Ejercicios

5.1. Ejercicio 1

Conecta el módulo con la LDR según se indica en el cuadro 1 (igual que conectaste la NTC en una práctica anterior). La salida del módulo quedará conectada al canal 0 del ADC.

Completa la función ldr_traducir_a_iluminancia del fichero fuente ldr.c.

Escribe una función main que muestre en la pantalla LCD la medida de iluminancia obtenida. Refresca la lectura una vez por segundo. Ten cuidado con el "rastro" que puede quedar en pantalla cuando el valor de iluminancia pase a tener menos dígitos. Esto puede provocar una presentación incoherente en la pantalla. Para evitar este problema, usa un especificador de ancho fijo en la función glcd_xprintf (por ejemplo "%9.2f") o añade espacios en blanco a la derecha del valor.

Tapa la LDR con la mano para observar los cambios en la medida de iluminancia.

5.2. Ejercicio 2

Analiza la función sonido_emitir_pitido en el fichero sonido.c.

Amplia el ejercicio anterior para que se emita un pitido corto una vez por segundo siempre que la iluminancia captada por la LDR sea inferior a 100 lux. Prueba con pitidos de una frecuencia de 1000 Hz y 100 ms de duración.

Tapa la LDR con la mano para hacer que se reduzca la iluminancia medida y suene el aviso.

5.3. Ejercicio 3

Analiza los ficheros con las tres versiones de las funciones de acceso al teclado matricial. Después, deja vinculado al proyecto únicamente la versión teclado_4x4_interrupciones_gpio_y_timer.c.

Cambia el ejercicio 2 para que el umbral de iluminación pueda ser introducido por el usuario. Para que sea sencillo, pide al usuario el valor del umbral solamente una vez al comienzo de la ejecución del programa. Si se requiere volver a introducir el valor, será necesario pulsar el botón de reset (el de la barra de herramientas de depuración de μVision o el botón físico de reset de la tarjeta de desarrollo).

Para la lectura de la cadena con el valor umbral usa la función tec4x4_leer_cadena_numeros. Para convertir la cadena de caracteres a su valor numérico puedes usar la función atoi, que tiene su prototipo en el fichero de cabecera estándar stdlib.h.