Práctica 3

Teclado matricial, puerto serie asíncrono y DMA

3.1. Objetivos de la práctica

En esta práctica seguiremos ampliando el catálogo de dispositivos que vamos a controlar. En primer lugar veremos cómo configurar y utilizar el teclado matricial. Posteriormente, nos centraremos en conocer un dispositivo de comunicaciones serie estándar, la UART. Este dispositivo nos permitirá comunicar la placa del laboratorio con el PC del puesto por medio de un protocolo serie asíncrono. Además, aprenderemos a utilizar el controlador de Acceso Directo a Memoria (DMA) para que los datos recibidos por el puerto sean copiados directamente en memoria sin intervención del procesador. Los principales objetivos de la práctica son:

- Aprender a gestionar el teclado matricial
- Conocer los conceptos básicos de la comunicación serie asíncrona.
- Familiarizarse con la unidad UART del S3C44BOX.
- Comprender los conceptos básicos de acceso directo a memoria (DMA).
- Familiarizarse con el controlador de DMA del S3C44BOX.

3.2. Desarrollo de la práctica

Toda la información necesaria para la configuración del teclado matricial, UART y DMA se encuentra en el documento separado entregado con la documentación de la práctica.

En este guión sólo describiremos el comportamiento esperado por vuestros proyectos, sin indicaciones de cómo llevarlo a cabo. Podéis partir de los códigos que os entregaremos, pero tenéis total libertad para modificar el diseño como consideréis oportuno.

Las tres partes (la última opcional) son similares en su funcionalidad: una versión simplificada del juego *Mastermind*. En el juego *Mastermind* un jugador pone una clave (en nuestro caso de 4 dígitos), que el adversario deberá adivinar (en nuestro caso, no pondremos límite de intentos). En el juego original, tras cada intento se recibe *feedback* por parte del usuario que puso la clave indicando cómo se acercaba nuestro intento a la clave real. En nuestra práctica sólo obtendremos dos posibles respuestas: clave correcta o incorrecta.

En la primera parte usaremos el teclado matricial tanto para introducir la clave como para los sucesivos intentos. En la segunda parte, los intentos se harán desde un PC conectado por puerto serie. Por último, la tercera parte será una modificación de la segunda utilizando DMA.

3.2.1. Parte 1: gestión del teclado

En esta parte usaremos únicamente el teclado (y no el puerto serie) así como el display 8-segmentos y el timer 1. Aunque el teclado no está numerado, supondremos que la tecla de la esquina superior izquierda es el `0`, la tecla de su derecha el `1`... hasta la tecla de la esquina inferior derecha que será la `F`.

La figura 3.1 muestra el flujo de ejecución que debe tener la aplicación. Detallemos paso a paso:

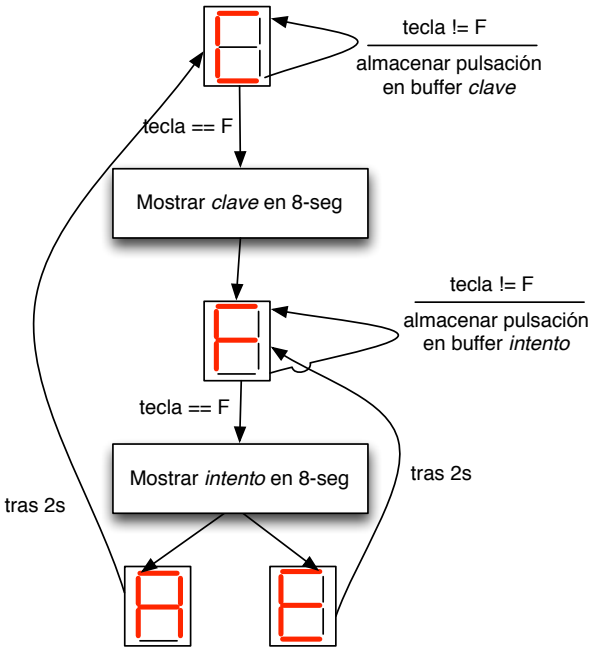

Figura 3.1: Diagrama de flujo de la aplicación

Al comenzar la ejecución, el display 8-segmentos mostrará una letra `C` indicando que está esperando la introducción de la clave.

El usuario comenzará a introducir la clave en el teclado matricial. Podrá hacer tantas pulsaciones desee, pero sólo se tendrán en cuenta las 4 anteriores a pulsar la tecla `F`. Los dígitos introducidos deben almacenarse en un *buffer*[1] denominado `clave` de modo que, al finalizar este paso, la clave de 4 dígitos esté almacenada en dicho buffer.

Cuando el usuario pulse la tecla `F` se mostrará la clave introducida por el display 8-segmentos (hasta este momento, el display únicamente mostraba `C`). Deberá mostrar un **dígito por segundo** y para ello **debe usarse el timer 1** (no se puede usar la función `Delay()`).

Tras el paso anterior, el display 8-segmentos mostrará una `F` para indicar que se puede proceder a tratar de adivinar la clave.

Nuevamente, el usuario realizará pulsaciones (mínimo 4) y acabará pulsando la tecla `F`. Los dígitos introducidos deben almacenarse en un *buffer* denominado `intento` de modo que, al finalizar este paso, el intento de 4 dígitos esté almacenado en dicho buffer.

Cuando el usuario pulse la tecla `F` se mostrará el intento introducido por el display 8-segmentos (hasta este momento, el display mostraba `F`). Deberá mostrar un **dígito por segundo** y para ello **debe usarse el timer 1** (no se puede usar la función `Delay()`).

Tras el paso anterior, si se acertó con la clave se mostrará una `A` y si no, se mostrará una `E`. En cualquiera de los dos casos, se permanecerá así durante 2 segundos. Si se acertó la clave, se volverá al estado inicial. Si no, se volverá a mostrar una `F` en espera de nuevos intentos.

Si bien se deja libertad absoluta para el diseño e implementación de la práctica, os **recomendamos que sigáis los siguientes pasos:**

1. Comenzad completando el código de gestión de teclado matricial entregado. Probadlo de modo que cada pulsación se refleje inmediatamente en el display 8-segmentos (sin otra fuente de interrupción que el propio teclado).

2. En lugar de mostrar el resultado por el display, almacenad las pulsaciones en un *buffer* en memoria (puede ser un simple array o, mejor aún, un tipo de datos que funcione como una cola que, al introducir nuevos elementos, descarte los más antiguos). Comprobad, mediante depuración, que el contenido del *buffer* en memoria es el esperado.

[1] Se puede implementar el *buffer* como se prefiera. Como mínimo, será un *array* que se gestionará de forma que siempre almacene las últimas cuatro pulsaciones, pero se recomienda crear un tipo de datos específico para este cometido

3. Habilitad las interrupciones del *timer 1* y programarlo de forma que muestre el contenido del *buffer* anterior, con una frecuencia de un elemento cada segundo (aprox.)

4. Ya están todos los elementos que forman la aplicación: *buffer*, gestión de teclado y gestión del *timer 1*. Sólo falta pensar cuidadosamente el diseño e integrar los elementos.

3.2.2. Parte 2: gestión del puerto serie

La funcionalidad de esta segunda parte será muy similar a la anterior pero con una salvedad: los intentos por adivinar la clave se realizarán desde el PC conectado por puerto serie en lugar de usar el teclado matricial. Para comunicar la placa con el PC no es necesario hacer ninguna conexión extra, pues el mismo USB utilizado para volcar el código se utilizará como puerto serie virtual.

Utilizaremos el programa de terminal `Termite`, ya instalado en el laboratorio, para establecer la conexión configurándola correctamente de acuerdo a la inicialización que se haya hecho en la placa (baudios, bits de paridad y parada...).

Nuevamente, la figura 3.2 muestra el flujo de ejecución, que detallamos a continuación:

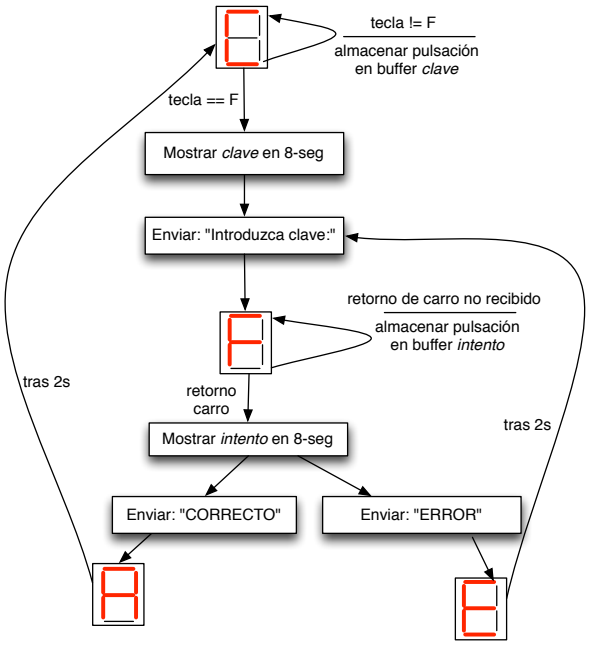

Figura 3.2: Diagrama de flujo de la aplicación

Al comenzar la ejecución, el display 8-segmentos mostrará una letra `C` indicando que está esperando la introducción de la clave.

El usuario comenzará a introducir la clave en el teclado matricial. Podrá hacer tantas pulsaciones desee, pero sólo se tendrán en cuenta las 4 anteriores a pulsar la tecla `F`. Los dígitos introducidos deben almacenarse en un *buffer*[2] denominado `clave` de modo que, al finalizar este paso, la clave de 4 dígitos esté almacenada en dicho buffer.

Cuando el usuario pulse la tecla `F` se mostrará la clave introducida por el display 8-segmentos (hasta este momento, el display únicamente mostraba `C`). Deberá mostrar un **dígito por segundo** y para ello **debe usarse el timer 1** (no se puede usar la función `Delay()`). Además, **se enviará la cadena de caracteres `Introduzca clave:` por el puerto serie** de modo que aparezca en *Termite* en el PC conectado por puerto serie. Asimismo, el display 8-segmentos mostrará una `F` para indicar que se puede proceder a tratar de adivinar la clave.

[2]Se puede implementar el *buffer* como se prefiera. Como mínimo, será un *array* que se gestionará de forma que siempre almacene las últimas cuatro pulsaciones, pero se recomienda crear un tipo de datos específico para este cometido

Ahora, el usuario debe introducir cuatro dígitos a través del terminal *Termite* y pulsar retorno de carro. En recepción, la placa debe ir almacenando cada dígito recibido por el puerto serie de modo que, al finalizar este paso, el intento de 4 dígitos esté almacenado en un buffer. Este paso finalizará cuando detectemos el carácter de *fin de línea* entre los bytes recibidos por el puerto serie. Esta funcionalidad (leer byte a byte del puerto serie hasta encontrar el carácter *fin de línea*) se encapsulará en la función **readline()** que seguirá el siguiente pseudo-código:

```
int readline(char* buf, int maxsize) {

        pos = 0;

        repetir {
                a = leer_byte_de_puerto_serie();
                buf[pos] = a;
                pos++;
        } mientras ( a != '\n' && pos < maxsize)

        return pos;
}
```

La función *readline()* recibirá un puntero a una zona de memoria donde poder almacenar los bytes recibidos por puerto serie (por ejemplo, un array) y la cantidad máxima de bytes que deseamos leer. La función terminará cuando se detecte *fin de línea* o cuando se haya leído el máximo de bytes sin encontrar dicho carácter. Devolverá un entero que indique cuántos bytes se han leído (incluyendo el propio carácter de *fin de línea*).

Tras el paso anterior, se mostrará el intento recibido por el display 8-segmentos. Deberá mostrar un **dígito por segundo** y para ello **debe usarse el timer 1** (no se puede usar la función `Delay()`).

Si la clave recibida es correcta **se enviará la cadena de caracteres `CORRECTO` por el puerto serie** y se mostrará una `A` en el display. Si no era correcta, **se enviará la cadena de caracteres `ERROR` por el puerto serie** y se mostrará una `E` en el display. En cualquiera de los dos casos, se permanecerá así durante 2 segundos. Si se acertó la clave, se volverá al estado inicial. Si no, se volverá a mostrar una `F` en espera de nuevos intentos y se enviará nuevamente la cadena `Introduzca clave:` por el puerto serie.

Nuevamente, tenéis absoluta libertad para el diseño y ejecución de esta parte, pero se recomienda probar cada componente por separado (en este caso, el funcionamiento del puerto serie en recepción y envío) antes de integrar todo.

3.2.3. Parte 3: uso de DMA (opcional)

Esta tercera parte es idéntica a la anterior pero se utilizará el DMA para el envío de cadenas de caracteres (`CORRECTO`, `ERROR` e `Introduzca clave:`) por el puerto serie. Es decir, el ARM sólo deberá indicar al controlador de DMA la dirección de comienzo de la cadena y su longitud, y sólo se le notificará al final del proceso.