SISTEMAS BASADOS EN MICROPROCESADOR
TRABAJO FINAL DE PRÁCTICAS

Grado en Ingeniería Electrónica Industrial (3º curso)

Curso 2022/2023

1 INTRODUCCIÓN

El presente documento recoge el trabajo final de prácticas de la asignatura de Sistemas Basados en Microprocesador de los estudios de Grado en Ingeniería Electrónica Industrial. El objetivo de este trabajo es reunir los aspectos tratados durante las prácticas de laboratorio y evaluar los avances de los alumnos. Este trabajo formará parte de la nota conforme se recoge en el apartado 3 y en la normativa de la asignatura.

2 ESTUDIO PRÁCTICO

2.1 Objetivos

El objetivo de este trabajo es el de desarrollar un programa mixto ensamblador y C que realice las siguientes tareas:

- Simular el funcionamiento de un reloj de ajedrez.
- Animación de leds en función del estado.
- Entrada por teclado matricial y micropulsadores.
- Control de la pantalla LCD con atenuación de caracteres.
- Salida de sonidos por zumbador.

2.2 Drivers

El programa tendrá que correr en el *Launchpad* del MSP430 y utilizará los siguientes dispositivos de E/S y sus *drivers* correspondientes:

- Pantalla LCD (`lcd.asm`).
- Leds y pulsadores de usuario del *Launchpad* (`pt.asm`).
- Temporizador del sistema (`st.asm` y `cs.asm`).
- Teclado de 16 teclas del *Boosterpack* (`teclado.asm`).
- Zumbador (`snd.asm`).

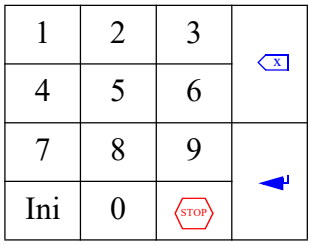
Se realizarán los siguientes cambios para adaptar las librerías:

2.2.1 LCD

Se añadirán funciones que configuren el parpadeo de los segmentos para que se pueda producir una atenuación de determinados dígitos tal como se indica en el punto 2.3.

2.2.2 Teclado

La configuración del teclado será la siguiente:



deberá reconfigurar la tabla de definición de códigos ASCII del teclado para que atiendan a la figura anterior. Observe que las teclas Temp+ y Temp- tendrán la misma salida, `Del` (código ASCII 8). De la misma forma, las teclas Copy y OK devolverán el código `Intro` (código ASCII 13). La tecla `Power` será `Ini` (código ASCII 2) y la tecla `Cool` será `Stop` (código ASCII 3). Deberá retocar el módulo de teclado para que no se emita ningún código cuando se suelte la tecla, ya que entraría el conflicto con la tecla `Stop`.

2.2.3 Sonido

Es necesario reproducir una pequeña melodía en paralelo con el resto del procesamiento de sistema. Puede hacer una subrutina `Play` que lleve a cabo el trabajo por interrupciones o hacerlo como una tarea cooperativa en C, tal como se describe en 2.4.2.1 Reproductor de melodías: `Play`.

2.3 Requisitos funcionales

Para la simulación,

- En este documento un elemento que parpadee a una frecuencia f y periodo T=1/f lo hará mediante una señal cuadrada (ciclo de trabajo del 50%) en la que el elemento estará encendido T/2 y apagado T/2.
- Toda pulsación del teclado debe causar efecto cuando la tecla se pulsa (no cuando se suelte). Y dicho efecto debe mostrarse sin pausa aparente.
- El teclado no tiene autorepetición. Para introducir dos ‘1’ habrá que pulsar y soltar dos veces la tecla.

El sistema tiene los siguientes estados:

- Configuración inicial: Se permitirá al usuario introducir el tiempo de la partida, que no será superior a 9 minutos y 59 segundos. Se podrán usar las teclas numéricas, la tecla de `Del` para borrar los dígitos introducidos y la tecla `Intro` para terminar la edición. Se comprobará que el tiempo introducido es válido al pulsar `Intro`. En ese caso se pasará al estado siguiente. En caso contrario, se emitirá un breve sonido de error, se encenderá el led2 durante un segundo con una señal de 4Hz y se permanecerá en este estado para que el usuario pueda editar el tiempo. Una vez configurado el tiempo de la partida, se mostrará el mensaje “Ini” centrado en la pantalla que parpadeará con una frecuencia de 1Hz y se esperará a que se pulse `Ini` para pasar al siguiente estado y empezar la partida.
- Partida: en la pantalla se visualizarán los tiempos restantes de los dos jugadores, reservando los 3 dígitos de la izquierda para el jugador 1 y los de la derecha para el jugador 2. Se usará el segmento `A1DP` para separar minutos y segundos del jugador 1 y el `A4COL` para los del jugador 2. Es decir, “`A.BC D:EF`”, donde A son los minutos restantes del jugador 1, BC son los segundos, D los minutos restantes del jugador 2 y EF sus segundos.
  - Al iniciar la partida, el jugador activo será el 1.
  - Cuando un jugador esté activo, su cuenta atrás estará en marcha y parpadeará su separador con una frecuencia de 1Hz. El otro jugador tendrá su separador sin parpadear y tanto éste como su tiempo restante se verán atenuados un 50%. La atenuación se implementará haciendo que dichos segmentos estén el 50% del tiempo encendidos y el 50% del tiempo apagados con una frecuencia de al menos 24Hz[1]. En los segmentos de la batería se visualizará el número de movimientos

empleados por el jugador en código binario (si el jugador ha hecho 20 movimientos, se visualizará ▯▯▮▯▮▯)

- Cuando el jugador activo haga su movimiento, actuará sobre su *switch* (el 1 para el jugador 1 y el 2 para el jugador 2). En ese instante, su cuenta atrás se paralizará, se encenderá su separador y se atenuará su tiempo restante. El otro jugador se convertirá en el jugador activo y se reproducirá una escala completa de fusas de la octava 5[2] al empezar su movimiento.

- Pausa: El árbitro de la partida podrá paralizar temporalmente los contadores pulsando la tecla `STOP`. Una partida pausada tendrá los dos contadores atenuados y sus separadores de minutos y segundos encendidos. Sobre los segmentos de la batería se realizará una animación con una frecuencia de 8Hz consistente en empezar encendiendo los segmentos de los extremos y avanzar hacia el centro para terminar de nuevo en los extremos ▮▯▯▯▯▮, ▯▮▯▯▮▯, ▯▯▮▮▯▯, ▯▮▯▯▮▯ y ▮▯▯▯▯▮. El proceso se repetirá a partir de ahí. La partida se reanudará con una nueva pulsación de `STOP`. También se podrá iniciar una nueva partida pulsando `Ini` (ver siguiente estado).
- Reinicio: El árbitro podrá iniciar una nueva partida pulsando `Ini`. En la pantalla aparecerá un mensaje "`Inicia`" que parpadeará al igual que el led2 con una frecuencia de 4Hz y se esperará la pulsación de una tecla. Si se pulsa `Ini` de nuevo, se procederá con una nueva partida desde el estado Configuración inicial. En caso contrario, se retomará la partida por donde fuera.
- Fin de la partida: Si el contador de tiempo del jugador activo llega a 0, terminará la partida dando por vencedor al otro jugador. En la pantalla se mostrará el mensaje "`Gana N`", donde N es el número del ganador. El mensaje y el led2 parpadearán con una frecuencia de 4 Hz. Se reproducirá una melodía de victoria. La única tecla activa será `Ini`. Si se pulsa, se pasará al estado Configuración inicial.

Se recuerda que los tiempos se medirán en TICS de *SystemTimer*. Así, si se elige una partida de 1'00" y el *SystemTimer* tiene una frecuencia de f_{ST}, el tiempo de la partida serán $60*f_{ST}$ TICS. A la hora de representar en pantalla el tiempo restante de los jugadores, se tomará el número de TICS, se dividirán por f_{ST} y se redondearán al entero superior[3].

2.4 Requisitos estructurales

Se deberá entregar un proyecto mixto C/Ensamblador que gestione los requisitos previamente indicados empleando las técnicas de multitarea cooperativa, máquina de estados, configuración de puertos y temporización y lectura de teclas por interrupciones tal y como se ha desarrollado tanto en las clases de teoría como en las prácticas.

En cualquier caso, ninguna tarea debe detener la ejecución de las demás.

---

1. Para implementar esta funcionalidad use las facilidades de parpadeo automático que da el módulo LCD del MSP430 y que tendrá que implementar dentro de `lcd.asm`.
2. Si el jugador activo pasa a ser el 1, la escala será ascendente de Do a Si y si pasa al jugador 2 descendente, de Si a Do.
3. Ejemplo: si f_{ST}=10 y el contador del jugador es 124 TICS, se realizará el cálculo $124/f_{ST}$ = 12'4 y se visualizará 0:13 como tiempo restante.

2.4.1 Sección ensamblador

Respecto de la parte ensamblador: se incluirán, al menos, los módulos desarrollados en prácticas correspondientes a los *drivers* de los dispositivos de E/S.

Podrán añadirse funciones extra de control de dispositivos en la parte correspondiente si se considera necesario.

2.4.2 Sección C

Se escribirá en C el código restante. Es de especial interés la función cooperativa `Play` que permite reproducir una partitura en paralelo con la ejecución del resto de los procesos.

2.4.2.1 Reproductor de melodías: `Play`

```
void Play (void);
```

Su misión será la de reproducir una partitura que se encuentra en la variable global

```
char *partitura;
```

La estructura de una partitura es una secuencia de pares de letra-dígito. En general, la letra es una nota y el dígito es la duración de la nota/silencio. Dicha duración no es un valor absoluto, sino una duración relativa a la duración base T. El dígito '0' representa una *longa*, es decir, el tiempo máximo de una nota o 4 veces T. '1' representa una *cuadrada*, o lo que es lo mismo, una nota de duración mitad de la *longa*. Los demás dígitos siguen la misma relación, dividiendo por 2 la duración de la nota. En resumen:

Dígito	Duración	Tiempo
'0'	Longa	4T
'1'	Cuadrada	2T
'2'	Redonda	T
'3'	Blanca	T/2
'4'	Negra	T/4
'5'	Corchea	T/8
'6'	Semicorchea	T/16
'7'	Fusa	T/32
'8'	Semifusa	T/64
'9'	Cuartifusa	T/128

siendo T el tiempo base de la *redonda* y que estableceremos en 2 segundos (120 bpm).

La letra puede ser:

Letra	Función	Dígito que le sigue
'o'	Establece octava actual. Al iniciar la reproducción, la octava por defecto es la 4	Número de nueva octava. Valores posibles: '4' a '6'

Letra	Función	Dígito que le sigue
‘s’	Hace un silencio en la reproducción	Duración
‘c’	Reproduce la nota *Do* de la octava actual	Duración
‘C’	Reproduce la nota *Do#* de la octava actual	Duración
‘d’	Reproduce la nota *Re* de la octava actual	Duración
‘D’	Reproduce la nota *Re#* de la octava actual	Duración
‘e’	Reproduce la nota *Mi* de la octava actual	Duración
‘f’	Reproduce la nota *Fa* de la octava actual	Duración
‘F’	Reproduce la nota *Fa#* de la octava actual	Duración
‘g’	Reproduce la nota *Sol* de la octava actual	Duración
‘G’	Reproduce la nota *Sol#* de la octava actual	Duración
‘a’	Reproduce la nota *La* de la octava actual	Duración
‘A’	Reproduce la nota *La#* de la octava actual	Duración
‘b’	Reproduce la nota *Si* de la octava actual	Duración

Por ejemplo, la cadena “c5d5e5f5g5a5b5o5c5d5e5f5g5a5b5o6c5d5e5f5g5a5b5” haría una escala de corcheas (duración T/8) por las tres octavas. Por tanto, debería durar 7*3*2/8= 6 segundos).

`Play` debe seguir la arquitectura de las tareas cooperativas para gestionar la reproducción de la melodía en paralelo con el resto del sistema. La variable `ProxEjec` se ajustará para que sea igual al final de la nota actual (y principio de la siguiente). De esta manera, en cada instante de ejecución de `Play` se gestionará un par de letra-dígito. La variable `ProxEjec` será global para que se pueda iniciar la reproducción desde el momento actual. Si se lee un par letra-dígito erróneos o si se alzanza el final de la cadena `partitura`, se apagará el sonido y se desactivará `Play`.

Dado que al inicio no se reproduce sonido, inicializamente se hará `ProxEjec=0xFFFFFFFF` para que no suene la melodía. Para entrar en modo reproducción bastará con cargar el instante actual en `ProxEjec` y la melodía en la variable `partitura`. `Play` gestionará la variable `partitura` para que apunte siempre al siguiente par letra-digito. El cambio de octava en el modo `Play` sólo debe afectar a la octava activa durante la partitura actual. Cada vez que se empiece una melodía nueva, se arranca en la octava 4.

2.4.3 Documentación del trabajo

Se adjuntará al proyecto un documento de texto conteniendo la siguiente documentación:

- Relación alfabética de todas las variables usadas (en C y ensamblador) indicando propósito (indicando la interpretación de los valores), módulo en el que se define, la función que la escribe y la función o funciones que la leen.
- Relación alfabética de funciones utilizadas en el programa (en C y ensamblador) indicando propósito de la misma, el módulo en el que se codifica, parámetros de entrada

(tipo e interpretación de los valores), parámetros de salida (tipo e interpretación de los valores), qué funciones que la llama y funciones que llama (si es el caso).

3 CRITERIOS DE PRESENTACIÓN Y VALORACIÓN

El trabajo final de prácticas se valorará entre 0 y 10 puntos. Sólo podrán realizar este trabajo los alumnos que hayan conseguido este derecho tras la realización del primer parcial de la asignatura, obtniendo la nota mínima de 7. La nota del trabajo, siempre que sea no inferior a 4 puntos, contribuirá al 80% de la nota de la asignatura. El 20% restante se obtendrá de la nota del primer parcial. La asignatura se considerará aprobada para notas no inferiores a 5. A no ser que se indique lo contrario y con carácter general, será de aplicación lo siguiente:

- Los trabajos se realizarán de manera individual. La violación de esta regla supondrá automáticamente la evaluación negativa (suspenso 0) de los alumnos implicados.
- El proyecto CCS ha de tener un nombre diferente para cada alumno. Concretamente, el formato a seguir será: AABB_UVUS, donde “AABB” será sustituido por el curso académico, y “UVUS” será sustituido por el UVUS de cada alumno. Por ejemplo, para un alumno con UVUS (Usuario Virtual de la Universidad de Sevilla y el comienzo del correo universitario, sin @alum.us.es) nomapaapb, matriculado en el curso 25/26, deberá usar como nombre de proyecto 2526_nomapaapb. El **incumplimiento** de este requisito da lugar a la **calificación directa de suspenso con 2’5 puntos**.
- "El fichero de documentación tendrá como nombre el nombre del proyecto, indicado en el punto anterior seguido de "_documentacion.pdf". Así, en el ejemplo dado se llamaría: "2526_nomapaapb_documentacion.pdf".
- Se valorará la presentación en general, el estilo de programación, el uso de comentarios, así como el grado de interés y atención mostrados por el alumno durante el desarrollo de las prácticas.
- Todos los ficheros fuentes deberán incluir una cabecera, a modo de comentario, como en el ejemplo siguiente (para fuentes en C, los comentarios empezarían con “//”):

```
;-------------------------------------------------------------
; Apellido1 Apellido2, Nombre
;-------------------------------------------------------------
```

- Los programas deberán estar libres de errores de compilación/ensamblado. El **incumplimiento** de este punto da lugar a la **calificación directa de suspenso** con 2.5 puntos.
- Es obligatorio seguir la estructura de ficheros, nombres, etc. de las prácticas. La falta de alguna subrutina, el cambio de nombre, o la implementación de una diferente funcionalidad da lugar a la calificación de 0 en la contribución de la subrutina a la nota global del trabajo.
- Los alumnos podrán ser llamados a defender su trabajo si el profesor lo estima necesario.

3.1 Fecha y lugar de entrega

En la página de la plataforma de enseñanza virtual de la asignatura, sección "Contenido", hay una actividad titulada "Entrega trabajo práctico asignatura". En dicha actividad se deberá subir un fichero que contenga lo indicado en el apartado siguiente. Se permitirán entregar tantas versiones como se desee, corrigiéndose exclusivamente la última entregada. Fecha tope entrega: con carácter general, no después del día del examen teórico final de la

asignatura. Cada día hábil de retraso en la entrega penalizará la nota con un punto sobre la calificación del trabajo. Pasados 5 días de la fecha tope de entrega, el trabajo no será evaluado y se considerará entregado para la siguiente convocatoria.

3.2 Contenido de la entrega

Un fichero comprimido (ZIP o RAR) en el que se encuentre la carpeta etiquetada con el convenio de nombres especificado en el apartado 3. Dentro de esa carpeta se incluirán todos los ficheros del proyecto y el de documentación del trabajo.

Control de versiones de este documento:

- v1.0: Versión original.