Fecha			Curso	Calificaciones Parciales			Cal. Final
29	11	2021	2				

Para la realización de este examen dispone de 50 minutos. Descomprima el fichero descargable de *Moodle*, lo que dará lugar a la siguiente estructura de carpetas:

```
MICR
├──L1
│   └──Hitos
│       ├──Hito_1
│       ├──Hito_2
│       └──Hito_3
├──mbed
└──to_7seg
```

Debe trabajar dentro de las carpetas `Hito_1` a `Hito_3`, en la primera de ellas encontrará un proyecto de *Keil*, aunque sus ficheros `main.cpp` están muy incompletos. **Al finalizar el examen debe comprimir la carpeta `Hitos`** (borrando antes las carpetas `~build` y `~listings` que pudieran existir) en un único fichero *7-ZIP* y subirlo al correspondiente enlace en *Moodle*. Este fichero únicamente servirá para su archivo, ya que la nota dependerá únicamente de la evaluación que se realice durante el examen.

Para cada hito verá una lista de objetivos que debe cumplir su programa. Si no logra todos ellos no se podrá obtener la máxima calificación. Cuando considere que tiene uno de los hitos listo debe levantar la mano para que el profesor pueda validarlo. El profesor evaluará cada hito a cada estudiante UNA ÚNICA VEZ. Una vez evaluado un hito no se puede modificar el código evaluado y debe pasar al siguiente hito. En cualquier caso, NO SE QUEDE ESPERANDO A QUE LLEGUE EL PROFESOR, continúe con el siguiente hito. El profesor revisará todos los hitos pendientes.

Para la realización de este examen **no** se permite el uso de la función `wait()` y tampoco de la librería *sw_tick_serial*, además, se desaconseja el uso de FSMs. No se permite la utilización de ningún recurso *software* ajeno a lo disponible en el *Moodle* de la asignatura, ni de cualquier trabajo realizado durante las prácticas o el estudio del examen. No se permite el uso de *pen-drives*, discos USB o cualquier otro medio para el almacenamiento o intercambio de datos.

Hito 1 (35 puntos): PULSADOR

Inicialmente (tras un *reset*) ambos *displays* permanecerán apagados, así como los tres LED. El sistema debe reconocer las pulsaciones del pulsador **central** tratando convenientemente los rebotes y teniendo en cuenta que, si se mantiene pulsado un tiempo largo, se contará solo una pulsación. Además, estas se harán efectivas al pulsar, no al soltar.

El sistema debe contar el número de veces que se acciona el pulsador central, almacenando este valor en una variable **global** (que será del tipo adecuado para contener un número de 8 bits con signo) llamada `g_cnt_sw`. Con cada nueva pulsación, se deberá cambiar el estado del LED central, pasando de apagado a encendido y viceversa. Cuando muestre este hito al profesor, este le pedirá que, mediante el uso de un punto de ruptura (*breakpoint*), detenga la ejecución del programa cuando se accione el pulsador, para así conocer el valor de dicha variable. Por último, debe dormirse el procesador siempre que sea posible.

Los *displays* permanecerán apagados en todo momento. Los demás pulsadores no afectarán al funcionamiento.

CRITERIOS: Vº. Bº:

□ Se llama a `wait()` (-35)

□ La respuesta al pulsador central no es la esperada (-15)
□ Hay rebotes (-15)
□ Las pulsaciones largas cuentan como más de una pulsación o se pierden pulsaciones cortas (-10)
□ Las pulsaciones se hacen efectivas al soltar, no al pulsar (-10)
□ No cambia el estado del LED o lo hace de forma incorrecta (-15)
□ No sabe poner un punto de ruptura o se activa el *breakpoint* en un instante diferente al de pulsación (-10)
□ No sabe ver el valor de una variable (-10)
□ La variable cuenta mal las veces que se acciona el pulsador (-10)
□ La variable no es del tipo adecuado o no es global (-10)
□ Los demás pulsadores influyen en el funcionamiento, o los *display*s se encienden (-10)
□ No se duerme al procesador cuando es posible o se hace incorrectamente (-5)
□ El código dentro de las ISR es complejo (-5)
□ Otros (a valorar):

Hito 2 (35 puntos): SECUENCIA DE LED

Copie en la carpeta `Hito_2` todos los contenidos de la carpeta `Hito_1`. Modifique ahora el código sobre esta carpeta para que, con una frecuencia de 10 Hz, los LED se actualicen, de forma que el LED encendido se «desplace» hacia la derecha. Cuando se llegue hasta el LED derecho, se volverá a empezar desde el izquierdo. En este hito, si se detecta una pulsación, la secuencia dejará de actualizarse, quedando la secuencia «parada» en la posición que estuviese, en lugar de encender el LED siguiente. Si se detecta una pulsación nueva, la secuenciación de los LED volverá a empezar desde ese mismo LED. Para conseguirlo se utilizará una variable **global** de tipo booleana llamada `gb_led_seq_on` que indicará si la secuencia se encuentra «desplazando» (`true`) o «parada» (`false`). En el estado inicial, (tras un *reset*), el LED izquierdo debe comenzar encendido y la secuencia parada.

CRITERIOS: Vº. Bº:
□ Se llama a `wait()` (-35)
□ La respuesta al pulsador central no es la esperada (-15)
□ La secuencia de LED no es correcta (-15)
□ La secuencia no comienza parada (-5)
□ Los demás pulsadores influyen en el funcionamiento, o los *display*s se encienden (-10)
□ Se ha degradado la funcionalidad del hito anterior (gestión de la pulsación) (-10)
□ El código dentro de las ISR es complejo (-5)
□ No se duerme al procesador cuando es posible o se hace incorrectamente (-5)
□ Otros (a valorar):

Hito 3 (30 puntos): MULTIPLEXACIÓN

Copie en la carpeta `Hito_3` todos los contenidos de la carpeta `Hito_2`. Modifique ahora el código sobre esta carpeta para que, cada vez que los LED se actualicen, una variable **global** (que será del tipo adecuado para contener un número de 8 bits con signo) llamada `g_cnt_LED`, aumente su valor en una unidad, empezando tras un *reset* en 0. Si el valor fuese mayor a 99, se volvería a empezar desde 0. Adicionalmente, esta variable se mostrará en los dos *displays* de forma simultánea, por medio de una multiplexación. El comportamiento del pulsador central y de los LED debe ser el mismo que en el hito anterior, por lo que preste especial atención a que la cuenta deje incrementarse cuando los LED dejen de actualizarse.

CRITERIOS: Vº. Bº:
□ Se llama a `wait()` (-30)
□ El valor mostrado en el *display* no es correcto (-15)
□ La multiplexación de los *displays* no es correcta o se aprecian sombras (-15)
□ El pulsador no detiene la cuenta incremental (-10)
□ Tras el «99» no se pasa al «00» (-10)
□ La cuenta no empieza en «00» (-5)
□ Se ha degradado la funcionalidad de los hitos anteriores (-15)
□ El código dentro de las ISR es complejo (-5)
□ No se duerme al procesador cuando es posible o se hace incorrectamente (-5)
□ Otros (a valorar):