Fecha			Curso	Calificaciones Parciales				Cal. Final
11	11	2019	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
├──EX_L1
│   └──Hitos
│       ├──Hito_1
│       ├──Hito_2
│       └──Hito_3
├──mbed
├──sw_tick_serial
└──to_7seg
```

Debe trabajar dentro de las carpetas `Hito_1` a `Hito_3`, en las que encontrará sendos proyectos 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*.

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()`. No se permite la utilización de ningún recurso *software* ajeno a lo disponible en el *Moodle* de la asignatura, en el que sus entregas de las anteriores prácticas no están accesibles. No se permite el uso de *pen-drives*, discos USB o cualquier otro medio para el almacenamiento de datos.

Hito 1 (45 puntos): DECREMENTO

El sistema debe emplear el pulsador central. Inicialmente (tras un *reset*) en el *display* de la *izquierda* se mostrará el dígito «4». Dicho dígito se *decrementará en dos unidades* con cada pulsación del pulsador central. Al llegar a cero, una nueva pulsación pondrá el *display* a 8. A la vez de todo lo anterior, el LED *central* lucirá de forma intermitente a una frecuencia de 0.5 Hz (1 s encendido, otro segundo apagado). Los LED restantes, así como el *display* derecho, permanecerán apagados en todo momento. Los demás pulsadores no afectarán al funcionamiento.

El sistema debe, además, contar el número de veces que se enciende el LED en una variable **global** (que será del tipo adecuado para contener un número de 16 bits con signo) llamada `cnt_led`. 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 apague el LED para, de esta forma, conocer el valor de dicha variable.

CRITERIOS: Vº. Bº:

□ Se llama a `wait()` (-45)
□ Los mensajes mostrados en el *display* no son los indicados (-15)
□ La respuesta al pulsador central no es la esperada (-15)
□ El funcionamiento del LED central no es correcto (-15)

□ No sabe poner un punto de ruptura o se activa el *breakpoint* en un instante diferente al apagado (-15)
□ No sabe ver el valor de una variable (-15)
□ La variable cuenta mal las veces que se enciende el LED (-10)
□ La variable no es del tipo adecuado o no es global (-10)
□ Los demás pulsadores influyen en el funcionamiento o los demás LED o el *display* derecho se encienden (-10)
□ El brillo del *display* varía en el tiempo o se aprecian parpadeos (-10)
□ Otros (a valorar):

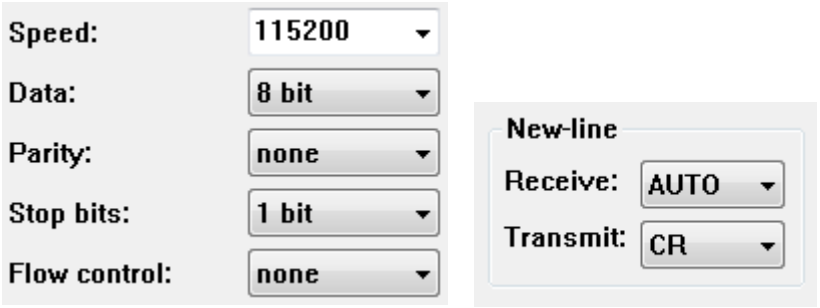
Hito 2 (30 puntos): MULTIPLEXACIÓN

Modifique el código del hito anterior para que la cuenta descendente mostrada en el *display* empiece en 96 y se muestren dos cifras. Cuando enseñe este hito al profesor, este le pedirá que, empleando los recursos de depuración de la herramienta (sin alterar el programa) modifique el valor de la cuenta del *display*, de modo que se represente «04» sin necesidad de realizar las 46 pulsaciones que serían necesarias para conseguirlo y que, de esta forma, sea cómodo verificar que tras el estado «00» se pasa al «98». La restante funcionalidad del hito anterior debe permanecer inalterada.

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)
□ No sabe modificar el valor de la cuenta para forzar la visualización del «04» (-15)
□ Tras el «00» no se pasa al «98» (-10)
□ La restante funcionalidad del hito 1 se ha modificado o degradado (-15)
□ Otros (a valorar):

Hito 3 (25 puntos): `PRINTF()`

Modifique el código del hito anterior para que, además, cada 10 s, el valor de la cuenta se incremente automáticamente en 4 unidades. Si tras el incremento resultase un valor de cuenta superior a 98, el valor de la cuenta se fijará en 98. También, cada vez que pasen esos 10 s, se enviará hacia el PC —mediante `printf()`, para ser visualizado en *Tera-Term*— el mensaje, sin incluir las comillas: «10 s MAS (*x*)», donde *x* es el valor de la cuenta (tras el incremento automático) y cada mensaje aparece en una línea nueva. La restante funcionalidad de los hitos anteriores debe conservarse inalterada. Se recuerdan los valores a los que debe ajustarse, en *Tera-Term*, el puerto de comunicaciones y el terminal cuando se usa la librería *sw_tick_serial*:



CRITERIOS: Vº. Bº:
□ Se llama a `wait()` (-25)
□ No ocurre el incremento automático de la cuenta (-15)
□ El incremento ocurre a una frecuencia o en una cantidad distinta a las esperadas (-10)
□ No sabe configurar adecuadamente *Tera-Term* (-15)
□ No ocurriendo lo anterior, los datos visualizados en *Tera-Term* no son los esperados (-15)
□ La restante funcionalidad de los hitos anteriores se ha modificado o degradado (-10)
□ Otros (a valorar):