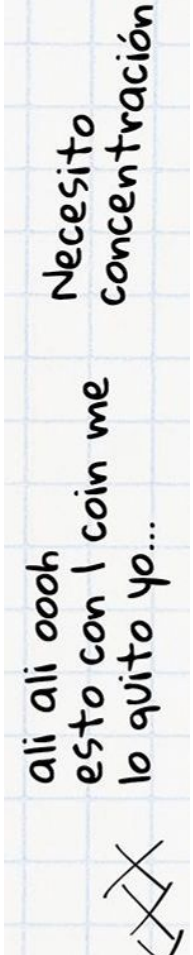
EXPSBMB22020SOL.pdf

Anónimo

Sistemas Basados en Microprocesador

3º Grado en Ingeniería Electrónica de Comunicaciones

Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación
Universidad Politécnica de Madrid



UNIVERSIDAD POLITÉCNICA DE MADRID *E.T.S.I.S.Telecomunicación*			**APELLIDOS:**	
			NOMBRE: **DNI:**	
			SISTEMAS BASADOS EN MICROPROCESADOR **Examen Práctico Bloque 2**	
			Grupo	
Fecha		**Curso**	**Calificaciones Parciales**	**Cal. Final**
17	**01**	**2020**		

CUANDO COMPLETE CADA UNO DE LOS SIGUIENTES APARTADOS LLAME A SU PROFESOR PARA QUE VERIFIQUE SU FUNCIONALIDAD. UNA VEZ TERMINE TODOS LOS APARTADOS O BIEN CUANDO SU PROFESOR SE LO DIGA, COMPRIMA CADA UNO DE LOS PROYECTOS OBTENIDOS EN UN ÚNICO FICHERO ZIP Y SÚBALO A LA TAREA DE MOODLE CORRESPONDIENTE.

Partiendo de las prácticas realizadas anteriormente y utilizando las tarjetas mbed NXP LPC1768, y la mbed Application Board, conteste a los siguientes apartados. Deberá utilizar para su implementación CMSIS-RTOS RTX.

APARTADO A (5 puntos)

Implemente una función denominada PintaPelota cuyo prototipo sea:

void PintaPelota (int posicion);

La función debe dibujar en el display un rectángulo de 12 pixeles de altura y 8 de ancho. Donde posición se corresponde con la columna donde se comienza a dibujar el rectángulo. El rectángulo se deberá representar en la parte del display correspondiente a las dos primeras páginas de memoria del LCD.

Para probar la función anterior, Implemente un timer virtual en CMSIS-RTOS RTX de forma que partiendo de un rectángulo en la posición cero, cada 600 ms el rectángulo anterior se desplace horizontalmente 8 pixeles. Una vez que el rectángulo llegué a la derecha del display, la siguiente posición será de nuevo la correspondiente a la columna cero. Este ciclo debe repetirse indefinidamente de forma cíclica.

APARTADO B (3 puntos)

Con la funcionalidad antes implementada se va a realizar un sencillo juego, consistente en golpear una pelota que se desplaza por el display. Las dos primeras páginas de la memoria del display se utilizarán para el desplazamiento de la pelota, las dos últimas se utilizarán para mostrar la puntuación del juego. La pelota será un rectángulo de 12x8 pixeles, y la puntuación se visualizará en la parte inferior del display con el siguiente texto: "Puntuacion x", donde x se corresponderá con la puntuación del juego.

La funcionalidad del juego será la siguiente:

- Inicialmente, tras un RESET, la pelota comenzará de forma cíclica a desplazarse de izquierda a derecha según se realizo en el apartado anterior.
- Si el usuario pulsa "CENTRAL" en el joystick, y la pulsación coincide cuando la pelota está en la derecha de la pantalla (última posición), se contabilizará como punto y se actualizará en el display la puntuación. En caso contrario no se actualizará la puntuación.

La lógica del juego, control de las pulsaciones, se debe realizar en un thread. Deben eliminarse los posibles rebotes del joystick. Para compartir la información entre el thread y el timer virtual puede utilizar una variable global denominada posición.

APARTADO D (2 puntos)

Modifique la aplicación anterior para en el caso de que se obtenga un punto, la pelota aumenté su velocidad de desplazamiento, disminuyendo el tiempo del timer virtual en 50 ms, hasta que éste alcance el valor de 100ms.

```
// THREAD principal.


extern osThreadId tid_TBase;
extern osTimerId id2;
extern      int posicion;

void TPrincipal (void const *argument);                             // thread function
osThreadId tid_TPrincipal;                                          // thread id
osThreadDef (TPrincipal, osPriorityNormal, 1, 0);                   // thread object

int Init_Thread (void) {

  tid_TPrincipal = osThreadCreate (osThread(TPrincipal), NULL);
  if (!tid_TPrincipal) return(-1);

  return(0);
}

void TPrincipal (void const *argument) {

    osEvent senal;
    int puntos=0;
    char texto[20];

    pintaPelota (0);
    EscribeLinea_2(" Puntuacion: 0");
    copy_to_lcd();

    osTimerStart (id2, 600);
    while(1){

           osSignalWait(S_JENTER,osWaitForever);
           if (posicion==15){
                  puntos++;
                  sprintf(texto," Puntuacion: %i",puntos);
                  EscribeLinea_2(texto);
                  copy_to_lcd();
           }
    }
  osThreadYield ();                                             // suspend thread

}
```

```
// Timer One-shot
//Replace Timer1 por nombre timer

extern      int posicion;
extern osThreadId tid_TPrincipal;

static void Timer1_Callback (void const *arg);                  // prototype for timer callback function

osTimerId id1;                                                  // timer id
static uint32_t  exec1;                                         // argument for the timer call back function
static osTimerDef (Timer1, Timer1_Callback);                    // define timers

// One-Shoot Timer Function
static void Timer1_Callback (void const *arg) {
  // add user code here
}



//Timer periodico
//Replace Timer2 por nombre timer
// utilizar     status = osTimerStart (id1, 100);




static void Timer2_Callback (void const *arg);                  // prototype for timer callback function

osTimerId id2;                                                  // timer id
static uint32_t  exec2;                                         // argument for the timer call back function
static osTimerDef (Timer2, Timer2_Callback);







// Periodic Timer Example
static void Timer2_Callback (void const *arg) {

    borraPelota((8*posicion));
    copy_to_lcd();
    posicion++;
    posicion &= 0x0F;
    pintaPelota((8*posicion));
    copy_to_lcd();

}




// Creamos los timers

void Init_Timers (void) {

  // Timer one-shoot
  exec1 = 1;
  id1 = osTimerCreate (osTimer(Timer1), osTimerOnce, &exec1);
  if (id1 == NULL) {    // One-shot timer created
           EscribeLinea_2("Error Crear Timer 1");
           copy_to_lcd();
    }

  // Timer Periodico
  exec2 = 2;
  id2 = osTimerCreate (osTimer(Timer2), osTimerPeriodic, &exec2);
  if (id2 == NULL) {    // Periodic timer created
```


```
            EscribeLinea_2("Error Crear Timer 2");
            copy_to_lcd();
        }
    }
```