UNIVERSIDAD DE ALCALÁ ESCUELA POLITÉCNICA SUPERIOR DEPARTAMENTO DE ELECTRÓNICA		GRADO EN INGENIERÍA ELECTRÓNICA DE COMUNICACIONES	
ASIGNATURA	SISTEMAS ELECTRÓNICOS DIGITALES AVANZADOS	FECHA	MARZO 2018
APELLIDOS, NOMBRE	SOLUCIÓN	GRUPO	

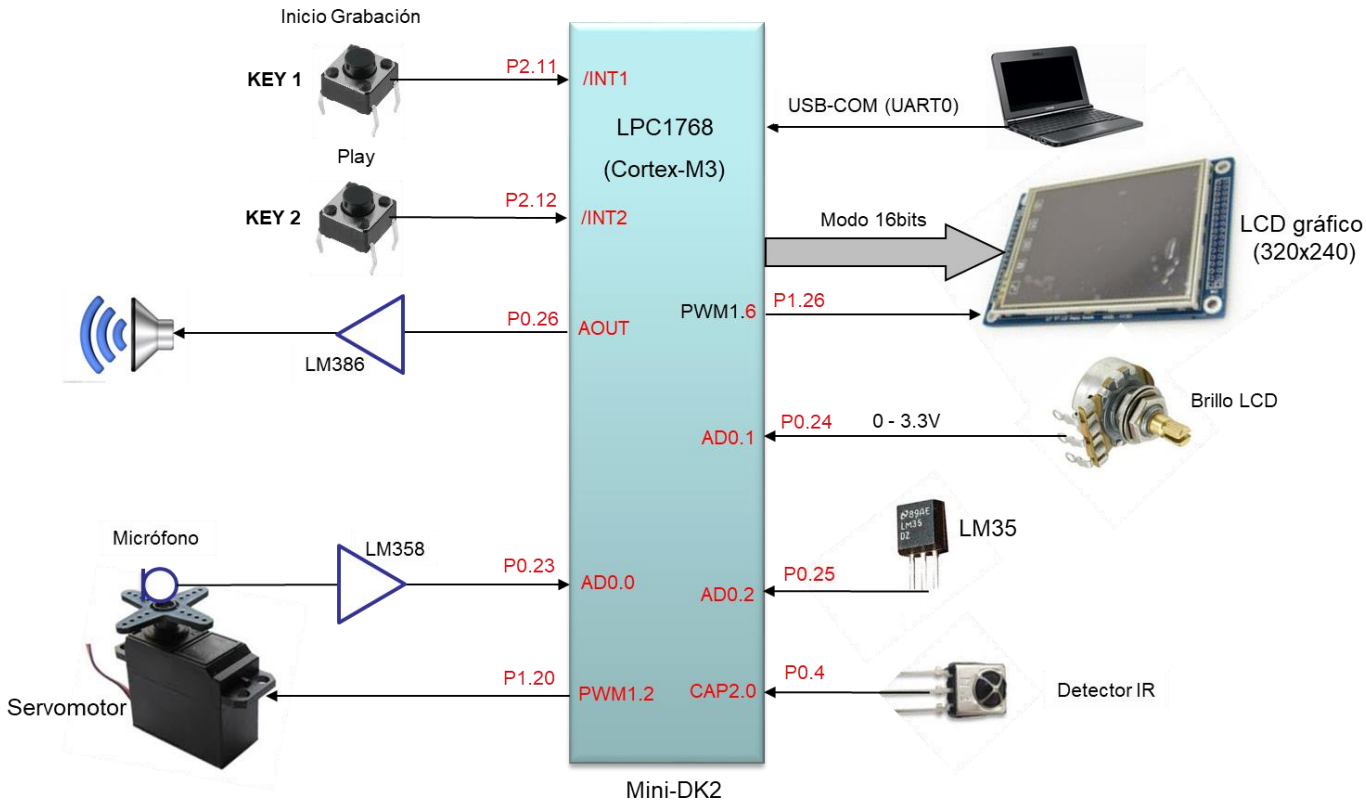
PRUEBA DE EVALUACIÓN INTERMEDIA 1

CUESTIÓN 1

El diagrama de la figura muestra un sistema empotrado basado en el LPC1768 para posicionar un micrófono direccional de manera que se oriente automáticamente hacia un locutor presente sobre un escenario. Un sistema de control buscará la máxima señal adquirida y actuará automáticamente sobre el servomotor que lleva acoplado mecánicamente el micrófono.

El sistema incorpora un detector de infrarrojos que permite ajustar la ganancia de la señal de salida audio (señal de alarma) a partir de la duración de la portadora de 38kHz generada con un dispositivo externo ajeno al sistema.

El sistema también es capaz de medir la temperatura exterior a partir de un sensor analógico y controlar el brillo del LCD de forma manual. El LCD muestra información y permite monitorizar las variables.



Condiciones de diseño:

El valor de la tensión analógica que proporciona el potenciómetro y el sensor de temperatura se ha de tomar **periódicamente cada segundo**.
Un menú de configuración permitirá seleccionar una opción para grabar en RAM el mensaje de alarma utilizando un micrófono con el correspondiente circuito acondicionador. La pulsación de KEY 1 iniciará el proceso de grabación a una frecuencia de muestreo de 8 kHz durante 2 segundos.
Considere que el SysTick está habilitado y que interrumpe periódicamente cada 50ms, entre otras cosas, para para hacer medidas de tiempos grandes.

a) Complete sobre el diagrama de la figura1, el nombre del pin (**Pn.x**) sobre la línea de conexión y el nombre del recurso utilizado dentro del bloque que representa la Mini-DK2 (ej. **MAT1.0**) excepto el LCD.

b) Complete la función de configuración del ADC (incluyendo los comentarios) para la entrada a la que se conecta el micrófono e indique el **tiempo de conversión** y la configuración del Timer (sin escribir el código) que proporciona el muestreo (Fs=8kHz a 8bits). Considere que funciona por DMA tanto para grabar el audio del mensaje de alarma (canal 0), como para procesar en tiempo real el audio del locutor (canal 1).

```
void init_ADC_microfono(void)
{
LPC_ADC->ADCR= (    1<<0)|    // ADC input P0.23 (AD0.0)
               ( 0x01<<8)|    // Fclk=25e6/2= 12,5MHz
                  (1<<21)|    // PDN=1
                   (6<<24);   // MAT1.0 -> frecuencia de muestreo
LPC_ADC->ADINTEN=(1<<0);      // Hab. canal 0 para DMA (no interrumpe)
NVIC_DisableIRQ(ADC_IRQn);    // Ojo! ADC no interrumpe!!
}
```

Tconv=65*1/12,5MHz = 5,2µs

Configuración Timer 1:

```
Power ON Timer1
Reset on Match 0
MR0=25e6/8000/2-1; // Cada 2 Match inicio de conversión (Fs=8KHz)
Sin interrupción
Stop Timer
```

c) Complete la función de configuración del ADC (incluyendo los comentarios) para las entradas del sensor de temperatura y del potenciómetro de brillo del LCD. Considere la frecuencia mínima de muestreo y calcule su valor.

```
void init_ADC_sensores(void)
{
LPC_ADC->ADCR=     (1<<1)|(1<<2)|     // Canales 1 y 2
                (0xFF<<8)|           // Fclk=25e6/256= 97,656kHz
                  (1<<21)|           // PDN=1
                   (1<<16);          // Modo BURST
NVIC_DisableIRQ(ADC_IRQn);           // No interrumpe
}
```

Fs=25e6/2*65*(CLKDIV+1)= 751,2Hz

NOTA: Considere ya escrita la siguiente función y añada **sólo** los comentarios:

```
void init_ADC_pines(void)
{
LPC_SC->PCONP|= (1<<12);                    // Power ON ADC
LPC_PINCON->PINSEL   |=(       )        ;   // P0.23,P0.24,P0.25 ADC inputs
LPC_PINCON->PINMODE  |=(       )        ;   // Sin pull-up ni pull-down
}
```

d) Indique la configuración del DMA (**canal 1**) para el modo de **procesado de la señal de audio** considerando que se han de almacenar en memoria las muestras del ADC de **8 bits** (2000 muestras)

Power ON DMA;
LPC_GPDMACH1->DMACCDestAddr = (uint32_t) &buffer_procesado[0]; // Destino Memoria
LPC_GPDMACH1->DMACCSrcAddr = (uint32_t) &(LPC_ADC->ADDR0) +1; // Origen Periférico (ojo 8 bits)
Transfer Size=2000; // Tamaño del bloque
Ancho transferencia= 8 **bits**;
Incrementa Destino;
No incremento Fuente;
Hab. Interrup DMA canal 1

uint8_t buffer_procesado[2000];

e) Escriba en pseudocódigo la función de interrupción del DMA que activa el flag ***fin_buffer_procesado=1***, en el modo procesado de la señal de audio, o el flag ***fin_grabacion_alarma=1***, en el modo de grabación de la señal de alarma.

```
DMA_IRQHandler:

Si ha interrumpido canal 0: // Necesario que interrumpa 4 veces
                                (4*4000=16000 muestras → 2seg. Grabación a fs=8kHz)
   Borrar flag interrupción canal 0
   contador++;
   Si contador==4 fin_grabacion_alarma=1;
   Si No, init_DMA_canal_0(&buffer_alarma[0]+4000*contador); //Dir. Destino

Si ha interrumpido canal 1:
   Borrar flag interrupción canal 1
   fin_buffer_procesado=1;
```

NOTA: Elegimos para el canal 0 Transfer Size=4000; // Tamaño del bloque

f) ¿Qué condiciones y/o parámetros limitan la duración máxima del mensaje de alarma?

- La memoria RAM interna (32KB + 32KB en segundo espacio) que considerando una Fs=8kHz tendríamos una duración de 7 segundos. NOTA: Tenemos que tener en cuenta que hay que restar el tamaño de la PILA, variables, y buffer para adquirir la señal a procesar (2000 bytes).
- La frecuencia de muestreo.

g) Escriba la función de interrupción del Timer 3 que saca las muestras hacia el DAC para generar la señal de alarma, y el valor del registro **MRx** correspondiente para obtener la señal de salida correctamente.

```
void TIMER3_IRQHandler(void)
{
static uint16_t indice_muestras;
LPC_TIM3->IR|=(1<<0);                              // Clear flag MR0
LPC_ADC_DACR=buffer_alarma[indice_muestras++]<<8; // DAC 8 bits
if(indice_muestras==16000){
                          indice_muestras=0;
                          LPC_TIM3->TC=0x02;       // Stop Timer
                          }
}
```

T3MR0=Fpclk/Fs -1= 25e6/8000 -1= 3124

NOTA: Ha de estar declarado, ***uint8_t buffer_alarma[16000];***

h) Escriba el código o pseudocódigo de las funciones de interrupción asociadas a los pulsadores Key 1 y Key 2.

```
void EINT1_IRQHandler(void) {
Borrar flag;
init_ADC_microfono();
init_DMA_canal_0(&buffer_alarma[0]);
Start Timer 1;
fin_grabacion_alarma=0;
}

void EINT2_IRQHandler(void) {
Borrar flag;
Start Timer 3;
}
```

i) Explique qué recurso utilizaría y como lo configuraría para reducir la carga de CPU durante la **generación de la señal de alarma**.

Evitaríamos la ejecución de la interrupción del Timer3 empleando el DMA en modo Linked (4 transferencias de 4000 muestras) con el propio canal (ej. Canal 2), para que sólo interrumpa en la última transferencia.

LPC_GPDMACH2->DMACCSrcAddr = (uint32_t) &buffer_alarma[0]; // Fuente Memoria
LPC_GPDMACH2->DMACCDestAddr = (uint32_t) &(LPC_DAC->DACR) **+ 1**; //Destino Periférico (ojo 8bits)
Transfer Size=**4000**; Ancho transferencia= **8 bits**;
Incrementa Fuente; No incremento Destino;
LPC_GPDMACH2->DMACCLLI= array_linked1;

uint32_t *array_linked1[4]*={ &(LPC_DAC->DACR) **+ 1,**
&buffer_alarma[4000],
array_linked2,
DMAcontrol };

NOTA: ídem, ***array_linked2*** y ***array_linked3***, excepto que este último y en DMAcontrol hay que hab. Interrup TC .

LPC_DAC->DACCNTVAL = (F_pclk/8000) -1 = 3124; // Frecuencia transf. DMA hacia DAC

j) Complete en **pseudocódigo** la función de configuración de la señal PWM (**incluyendo los comentarios**). Considere que el periodo sea de **15ms**, y el tiempo a nivel alto varíe entre **0.8-2.4ms**, para un movimiento de su posición entre 0º y 180º.

```
void config_pwm(void)
{
LPC_PINCON->PINSEL7|=(3<<18); // PWM1.2
LPC_SC->PCONP|=(1<<6);  //Power PMW module
MR0=(Fpclk*15e-3 -1); // periodo PWM
Reset on MR0;
Single Edge PWM2 y Hab. PWM1.2 (ENA2=1)
Start Timer;
}
```

k) Complete la función de actualización de la posición del servo.

```
void set_servo(char grados)
{
LPC_PWM1->MR2=Fpclk*0.8e-3 + Fpclk*1.6e-3*grados/180;
LPC_PWM1->LER|= (1<<2)|(1<<0);
}
```

l) Escriba la función de interrupción asociada al Timer que mide la duración del pulso procedente del detector de IR y modifica la variable global ***ganancia*** (entre 0 y 100) proporcionalmente a la duración de la portadora de 38kHz (entre 1 y 20ms). Indique la configuración del Timer sin escribir el código.

```
void TIMER2_IRQHandler(void)
{
static uint32_t temp;
Borrar flag;

if (flanco_bajada){
                    temp=LPC_TIM2->CR0; // Cuentas primer flanco (bajada)
                    config. Flanco subida;
                    flanco_bajada=0;
}

else {
         N=LPC_TIM2->CR0-temp;           // cuentas entre flancos (= µseg. *)
         if ((N>1000)&&(N<20000)) ganancia=(N-1000)/190;
         config. Flanco bajada;
         flanco_bajada=1;
}
```

```
Timer 2:
      Modo Captura (CAP2.0)
      *PR=24 → Ftick= 1MHz
      Flanco_bajada=1 (inicio captura en flanco bajada)
      Hab. Interrupción (máxima prioridad)
      Start Timer
```

m) Escriba el código de la función de interrupción del **SysTick**, encargada de procesar el audio en tiempo real para actuar sobre el servomotor para posicionar el micrófono, así como de **mostrar por el LCD** la temperatura, y modificar el brillo del LCD **cada segundo**.
Considere ya escrita la función ***procesa_audio()*** que a procesa las muestras de audio adquiridas vía DMA y posiciona el servo.

```
void SysTick_IRQHandler(void)
{
   cont++;
   if(cont%20==0){
              init_ADC_sensores(void);
              Start Timer 1;
               codigo_brillo=(LPC_ADC->ADDR1>>8)&0xFF); // 8 bits
              LPC_PWM1->MR2=(uint32_t) (LPC_PWM1->MR0*codigo_brillo/255);
              LPC_PWM1->LER|=(1<<2)|(1<<0);
              temperatura=((LPC_ADC->ADDR2>>4)&0xFFF)*330/4095;//grados=mV/10
              sprintf(cadena,"temperatura= %2.1f grados",temperatura);
              GUI_Text(10,10, cadena, WHITE, BLACK);
              init_DMA_canal_0();
              init_ADC_microfono(void);
        }
   else if (fin_buffer_procesado){
                    Stop Timer 1;
                    fin_buffer_procesado=0;
                    procesa_audio();
                    Start Timer 1;
              }
}
```

CUESTIÓN 2

Se desea especificar el sistema de control de un ascensor de un edificio para lo que se dispone de las siguientes señales de señalización y control:

- En cada piso hay un sensor que se activa cuando el ascensor está en el entorno del piso. La activación de este sensor modifica el valor de la variable "Actual" indicando el piso en que se encuentra el ascensor en ese momento. Entre pisos, la variable Actual valdrá cero.
- En todos los pisos hay un sensor que se activa justo cuando el ascensor en la posición donde debe detenerse. Todos estos sensores están cableados de tal forma que cuando se activa uno de ellos, la variable "Stop" vale uno, y cuando ninguno está activo, la variable vale cero.
- En el interior del ascensor hay una botonera con tantos botones como pisos más un botón de reapertura de puertas. En cada piso hay un botón de llamada del ascensor.
- Cuando se pulsa un botón de piso en el interior del ascensor o en un botón de llamada, estando el ascensor parado, se almacena en la variable "Fin" el piso correspondiente al botón pulsado. Esta funcionalidad la hace un sistema externo.
- El motor tiene tres señales de control que corresponde con tres variables:

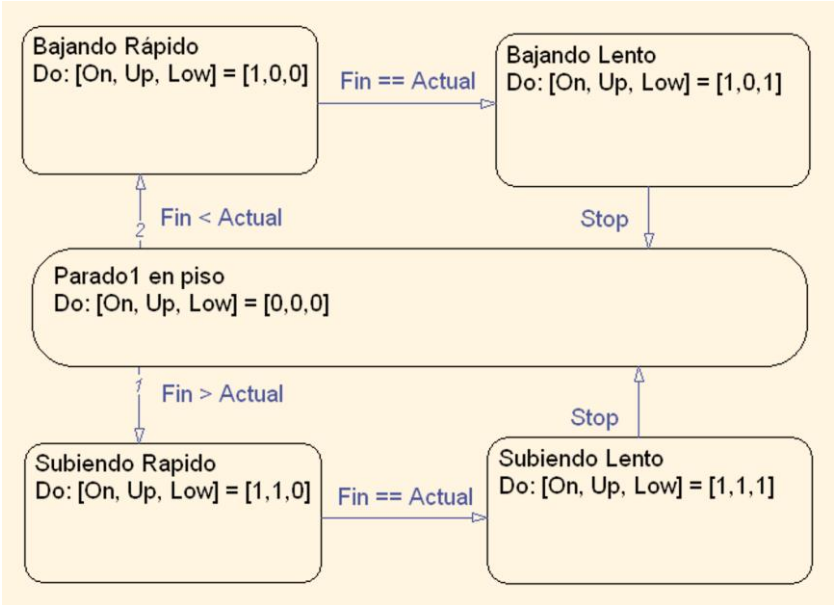
On	*Up*	*Low*	
0	x	x	Parado
1	0	0	Baja rápido
1	0	1	Baja despacio
1	1	0	Sube rápido
1	1	1	Sube despacio

- Las puertas del ascensor tienen las siguientes señales de control:

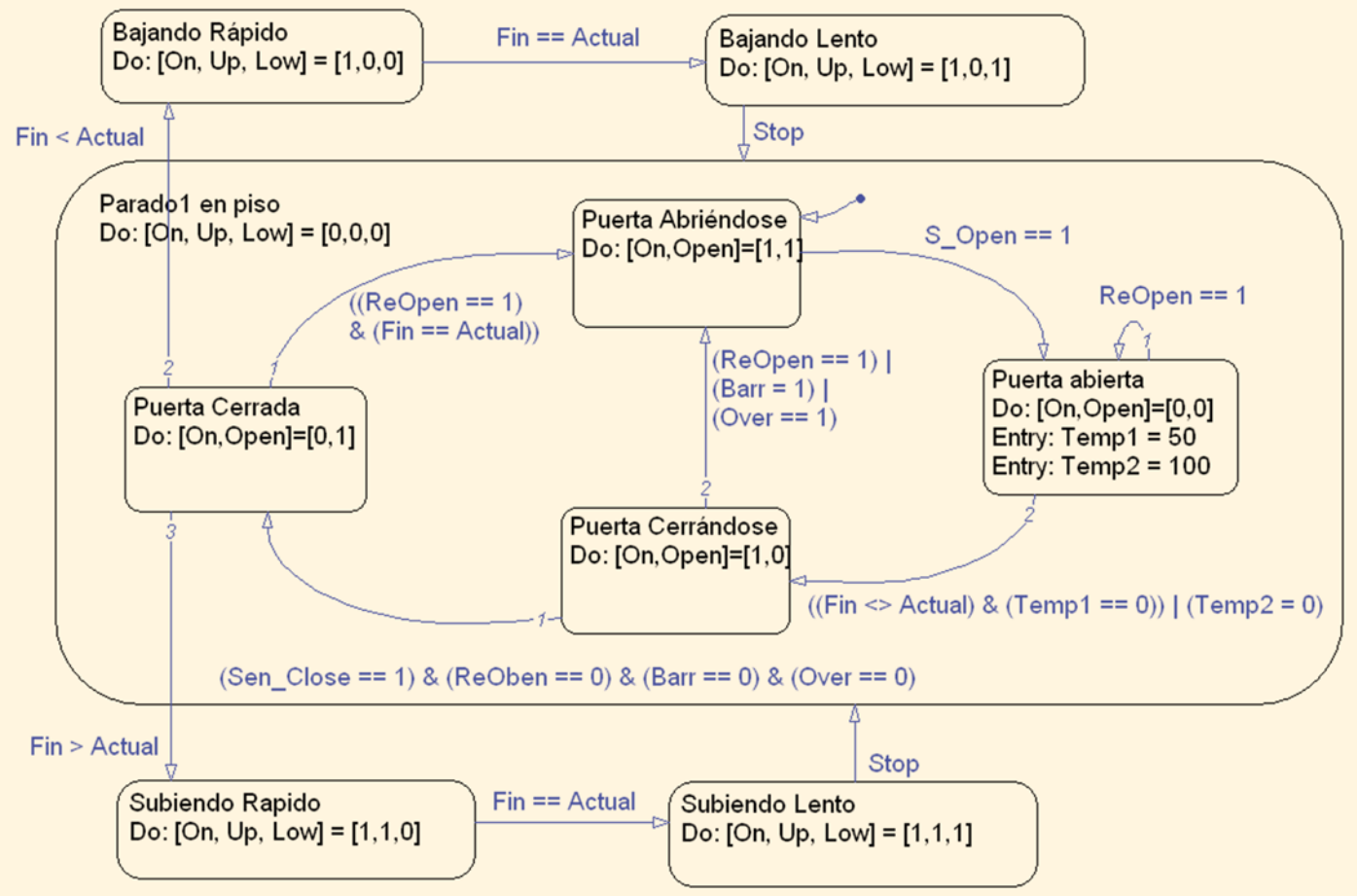
On	*Open*	
0	x	Paradas
1	0	Cerrándose
1	1	Abriéndose

- Hay un sensor que indica que las puertas del ascensor están cerradas (S_Close) y otro que indica que están abiertas (S_Open).
- El ascensor dispone de un sensor de barrera (Barr) que detecta si alguien está pasando y un sensor de exceso de corriente (Over) que indica que el motor está haciendo una fuerza anormal. Ambos sensores son activos a nivel alto.

Rehaga el StateChart de la figura para incluir el control de apertura de las puertas, teniendo en cuenta que las puertas deben permanecer abiertas al menos 5 segundos y deben cerrarse pasados 10 segundos si no se activan los botones o el sensor de barrera. Si las puertas se están cerrando y se detecta el sensor de barrera, el de sobre corriente o se pulsa el botón de reapertura deberán abrirse inmediatamente.



Solución:



Nota: Por error, en el enunciado no se indica cómo se notifica al sistema de control cuando el ascensor está en reposo un piso determinado con las puertas cerradas y se pulsa el botón de llamada del mismo piso. La solución propuesta tampoco lo contempla.

Se supone que hay una interrupción periódica que se ejecuta cada (por ejemplo, 100ms) donde se van decrementando las variables Temp1 y Temp2 si son mayores que cero. Cuando se escribe en el StateChart Temp1=5s es equivalente a poner Temp1 = 50 y con Temp2=10s correspondería con Temp2=100.

CUESTIÓN 3

Se dispone de un sistema de alimentación ininterrumpida (UPS) que alimenta a un servidor. El sistema debe presentar la información del estado de la UPS mediante una página web como la indicada en la figura adjunta, monitorizando si hay alimentación de entra de la red eléctrica, si está activa la salida de alimentación al servidor, el nivel de carga de la batería debe proporcionar la posibilidad de apagar el servidor de forma temporizada.

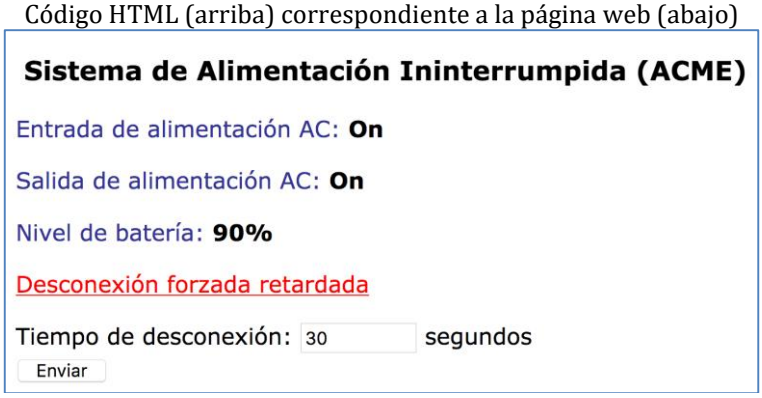
Suponiendo que el sistema está implementado con LPC1768 y que la información se encuentra en las siguientes variables, escriba el contenido del fichero **ups.cgi**, y de las funciones **cgi_func(...)**, **cgi_process_var(...)** y/o **cgi_process_data(...)** que sea necesario.

- LineOn – Vale 1 si hay alimentación externa de entrada y 0 si no hay alimentación externa
- Out_On - Vale 1 si la salida de alimentación está activa.
- Battery – Indica el porcentaje de la batería disponible.
- void powerOff(unsigned int time) – Función que, al ser llamada, envía una señal al ordenador para que se apague pasados tantos segundos como se indique en el parámetro 'time'.

Al pulsar el botón de Enviar la url a la que accede es http://www.seda.uah.es/ups.cgi?time=30

```
<!DOCTYPE html>
<html>
  <head>
    <meta content="text/html; charset=UTF-8" http-equiv="content-type">
    <title>index.htm</title>
  </head>
  <body>
    <h3 style="text-align: center;">Sistema de Alimentaci&oacute;n Ininterrumpida (ACME)</h3>
    <p><span style="color: #333399;">Entrada de alimentaci&oacute;n AC:</span> <strong>On</strong></p>
    <p><span style="color: #333399;">Salida de alimentaci&oacute;n AC:</span> <strong>On</strong></p>
    <p><span style="color: #333399;">Nivel de bater&iacute;a:</span> <strong>90%</strong></p>
    <p><span style="text-decoration: underline;"><span style="color: #ff0000; text-decoration: underline;">
        Desconexi&oacute;n forzada retardada</span></span></p>
    <form action="ups.cgi" method="GET">
      <p>Tiempo de desconexi&oacute;n: <input name="time" size="8" value="30" type="text" /> segundos<br>
      <input value="Enviar" type="submit"></p>
    </form>
  </body>
</html>
```

Código HTML (arriba) correspondiente a la página web (abajo)




```
t  <!DOCTYPE html>
t  <html>
t    <head>
t      <meta content="text/html; charset=UTF-8" http-equiv="content-type">
t      <title>index.htm</title>
t    </head>
t    <body>
t      <h3 style="text-align: center;">Sistema de Alimentaci&oacute;n Ininterrumpida (ACME)</h3>
c 1    <p><span style="color: #333399;">Entrada de alimentaci&oacute;n AC:</span> <strong>%s</strong></p>
c 2    <p><span style="color: #333399;">Salida de alimentaci&oacute;n AC:</span> <strong>%s</strong></p>
c 3    <p><span style="color: #333399;">Nivel de bater&iacute;a:</span> <strong>%d%</strong></p>
t      <p><span style="text-decoration: underline;"><span style="color: #ff0000; text-decoration: underline;">
t          Desconexi&oacute;n forzada retardada</span></span></p>
t      <form action="ups.cgi" method="GET">
t        <p>Tiempo de desconexi&oacute;n: <input name="time" size="8" value="  " type="text" /> segundos<br>
t        <input value="Enviar" type="submit"></p>
t      </form>
t    </body>
t  </html>
.
```

```
#include <Net_Config.h>
#include <stdio.h>
#include <string.h>


U16 cgi_func (U8 *env, U8 *buf, U16 buflen, U32 *pcgi) {
  U32 len = 0;
  U32 indice = 0;

  switch (env[0]) {
    case '1' :
       len = sprintf((char *)buf,(const char *)&env[2],(LineOn == 1) ? "ON" : "OFF"));
       break;
    case '2' :
       len = sprintf((char *)buf,(const char *)&env[2],(OutOn == 1) ? "ON" : "OFF"));
       break;
    case '3' :
       len = sprintf((char *)buf,(const char *)&env[2], Battery);
      break;
    default:
      break;
  }
  return ((U16)len);
}


void cgi_process_var (U8 *qs) {
  U8 *var;
  int time;

  var = (U8 *)alloc_mem (40);
  do {
    qs = http_get_var (qs, var, 40);
    if (var[0] != 0) {
      if (str_scomp (var, "time=") == __TRUE) {
         time = atoi(&var[5]);
         if (time > 0)
            powerOff(time);
      }
    }
  }while (qs);
  free_mem ((OS_FRAME *)var);
}
```

ANEXO I (Mapa de memoria)

Table 3. LPC176x/5x memory usage and details

Address range	General Use	Address range details and description	
0x0000 0000 to 0x1FFF FFFF	On-chip non-volatile memory	0x0000 0000 - 0x0007 FFFF	For devices with 512 kB of flash memory.
		0x0000 0000 - 0x0003 FFFF	For devices with 256 kB of flash memory.
		0x0000 0000 - 0x0001 FFFF	For devices with 128 kB of flash memory.
		0x0000 0000 - 0x0000 FFFF	For devices with 64 kB of flash memory.
		0x0000 0000 - 0x0000 7FFF	For devices with 32 kB of flash memory.
	On-chip SRAM	0x1000 0000 - 0x1000 7FFF	For devices with 32 kB of local SRAM.
		0x1000 0000 - 0x1000 3FFF	For devices with 16 kB of local SRAM.
		0x1000 0000 - 0x1000 1FFF	For devices with 8 kB of local SRAM.
	Boot ROM	0x1FFF 0000 - 0x1FFF 1FFF	8 kB Boot ROM with flash services.
0x2000 0000 to 0x3FFF FFFF	On-chip SRAM (typically used for peripheral data)	0x2007 C000 - 0x2007 FFFF	AHB SRAM - bank 0 (16 kB), present on devices with 32 kB or 64 kB of total SRAM.
		0x2008 0000 - 0x2008 3FFF	AHB SRAM - bank 1 (16 kB), present on devices with 64 kB of total SRAM.
	GPIO	0x2009 C000 - 0x2009 FFFF	GPIO.
0x4000 0000 to 0x5FFF FFFF	APB Peripherals	0x4000 0000 - 0x4007 FFFF	APB0 Peripherals, up to 32 peripheral blocks, 16 kB each.
		0x4008 0000 - 0x400F FFFF	APB1 Peripherals, up to 32 peripheral blocks, 16 kB each.
	AHB peripherals	0x5000 0000 - 0x501F FFFF	DMA Controller, Ethernet interface, and USB interface.
0xE000 0000 to 0xE00F FFFF	Cortex-M3 Private Peripheral Bus	0xE000 0000 - 0xE00F FFFF	Cortex-M3 related functions, includes the NVIC and System Tick Timer.

Table 530. ADC registers

Generic Name	Description	Access	Reset value[1]	AD0 Name & Address
ADCR	A/D Control Register. The ADCR register must be written to select the operating mode before A/D conversion can occur.	R/W	1	AD0CR - 0x4003 4000
ADGDR	A/D Global Data Register. This register contains the ADC's DONE bit and the result of the most recent A/D conversion.	R/W	NA	AD0GDR - 0x4003 4004
ADINTEN	A/D Interrupt Enable Register. This register contains enable bits that allow the DONE flag of each A/D channel to be included or excluded from contributing to the generation of an A/D interrupt.	R/W	0x100	AD0INTEN - 0x4003 400C
ADDR0	A/D Channel 0 Data Register. This register contains the result of the most recent conversion completed on channel 0.	RO	NA	AD0DR0 - 0x4003 4010
ADDR1	A/D Channel 1 Data Register. This register contains the result of the most recent conversion completed on channel 1.	RO	NA	AD0DR1 - 0x4003 4014
ADDR2	A/D Channel 2 Data Register. This register contains the result of the most recent conversion completed on channel 2.	RO	NA	AD0DR2 - 0x4003 4018
ADDR3	A/D Channel 3 Data Register. This register contains the result of the most recent conversion completed on channel 3.	RO	NA	AD0DR3 - 0x4003 401C
ADDR4	A/D Channel 4 Data Register. This register contains the result of the most recent conversion completed on channel 4.	RO	NA	AD0DR4 - 0x4003 4020
ADDR5	A/D Channel 5 Data Register. This register contains the result of the most recent conversion completed on channel 5.	RO	NA	AD0DR5 - 0x4003 4024
ADDR6	A/D Channel 6 Data Register. This register contains the result of the most recent conversion completed on channel 6.	RO	NA	AD0DR6 - 0x4003 4028
ADDR7	A/D Channel 7 Data Register. This register contains the result of the most recent conversion completed on channel 7.	RO	NA	AD0DR7 - 0x4003 402C
ADSTAT	A/D Status Register. This register contains DONE and OVERRUN flags for all of the A/D channels, as well as the A/D interrupt/DMA flag.	RO	0	AD0STAT - 0x4003 4030
ADTRM	ADC trim register.	R/W	0x0000 0F00	AD0TRM - 0x4003 4034