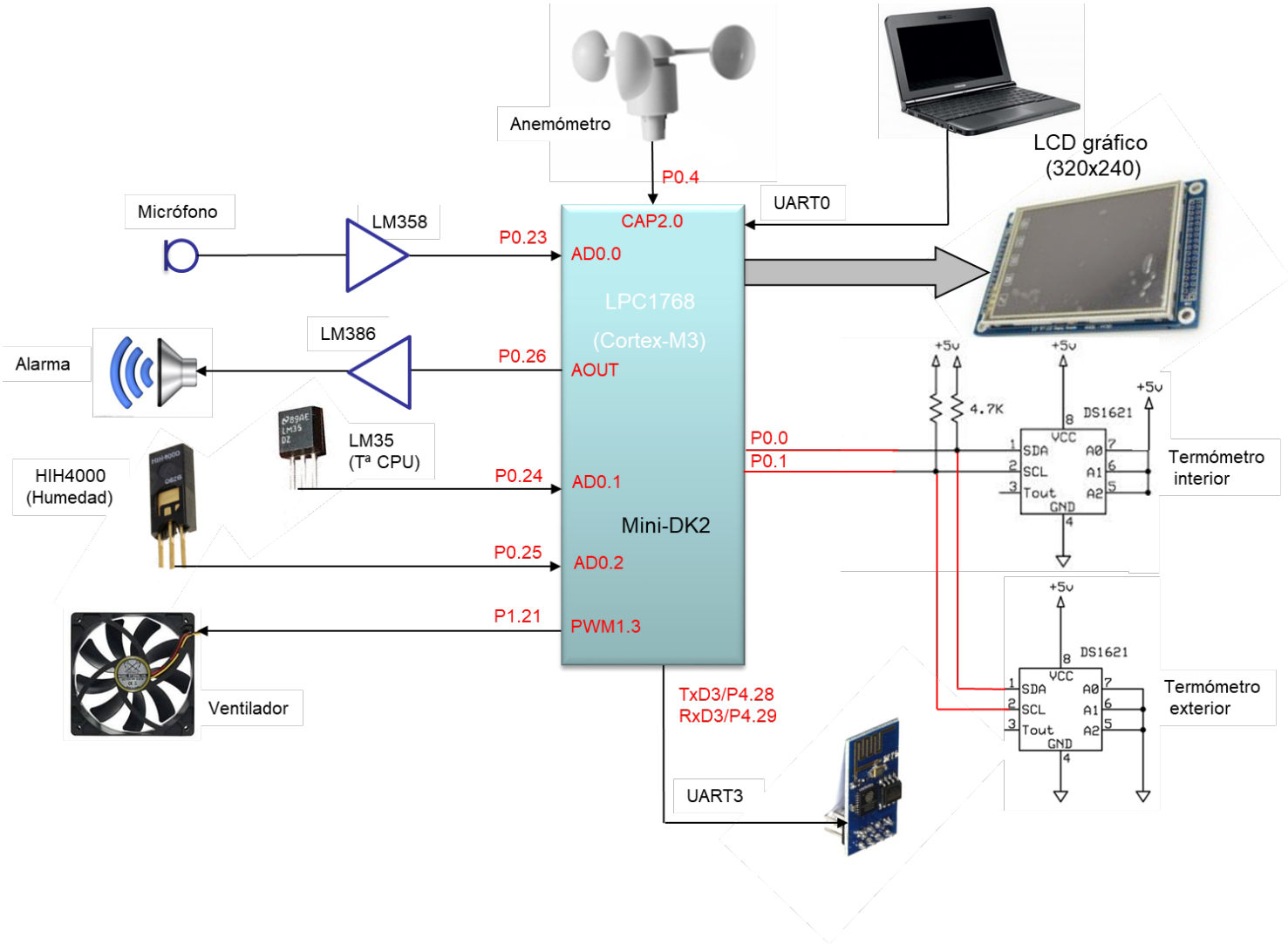
UNIVERSIDAD DE ALCALÁ ESCUELA POLITÉCNICA SUPERIOR DEPARTAMENTO DE ELECTRÓNICA		GRADO EN INGENIERÍA ELECTRÓNICA DE COMUNICACIONES GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN	
ASIGNATURA	SISTEMAS ELECTRÓNICOS DIGITALES AVANZADOS	FECHA	JUNIO 2018
APELLIDOS, NOMBRE	SOLUCIÓN	GRUPO	

PRUEBA DE EVALUACIÓN EXTRAORDINARIA

CUESTIÓN 1.

Se propone diseñar un SE (sistema empotrado) basado en el microcontrolador LPC1768 (Cortex-M3) con objeto de implementar una estación meteorológica, que tenga la posibilidad de ser monitorizada de forma remota desde un ordenador mediante una interfaz serie asíncrona, una conexión WiFi, o desde un entorno WEB. El sistema permitirá mostrar sobre un display la magnitud medida por los distintos sensores: velocidad del viento, temperatura exterior, temperatura interior, humedad del ambiente y temperatura de la CPU. **NOTA:** F_pclk=25Mhz, y F_cpu=100Mhz.

El sistema incorporará un sistema de control de temperatura del equipo mediante ventilación forzada en caso de un emplazamiento externo para garantizar unas condiciones óptimas de funcionamiento.



Funcionamiento:

Tendrá un modo de funcionamiento en el que periódicamente (tiempo configurable) se registrarán todas estas magnitudes en memoria con objeto de poder analizar su evolución temporal a lo largo de un día, una semana, etc. Mediante un PC conectado al sistema se podrán volcar todos estos datos a través de un puerto serie sobre un fichero para un posterior análisis. Las distintas magnitudes medidas podrán ser consultadas en tiempo real a través una conexión de internet mediante la implementación de un servidor web empotrado, pudiendo interactuar de forma remota con el sistema. Se propone que la estación tenga un sistema de alarma o aviso programable sobre cualquier magnitud, de manera que se genere un mensaje de voz grabado previamente por el usuario.

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. La pulsación de KEY 2 inicia la reproducción para comprobar en mensaje.

a) Complete sobre el diagrama de la figura, a medida que responde a las cuestiones, 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 en lo referente al LCD.

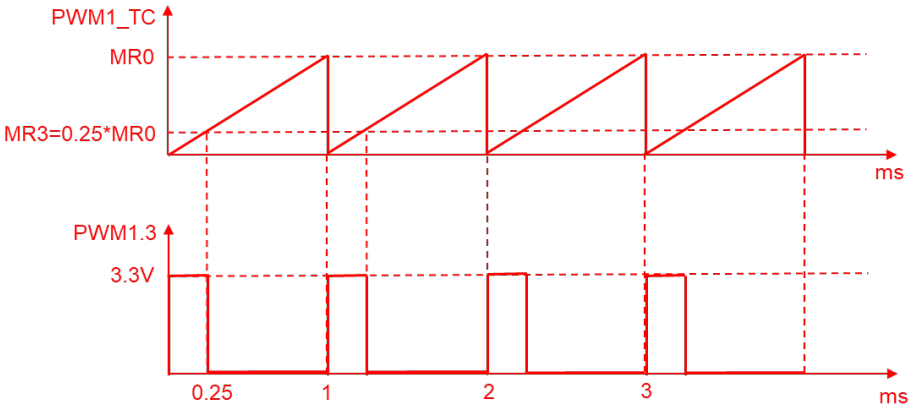
b) Se desea leer la temperatura y humedad del sensor LM35 y del HIH4000 empleando el **ADC por interrupción**. Complete la función de configuración del ADC y de interrupción, así como los comentarios del código e indique la frecuencia de muestreo de cada canal

```
void init_ADC_TIMER_sensores(void)
{
LPC_SC->PCONP|= (1<<12);                    // Power ON
LPC_PINCON->PINSEL1|=sólo comentarios     ;// P0.24 es AD0.1 y P0.25 es AD0.2
LPC_PINCON->PINMODE1|=solo comentarios    ;// Sin Pull-Up ni Pull-Down
LPC_ADC->ADCR= (    1<<1 )|                 // Canal 1 (ojo!! sólo un canal)
                (    1<<8)|                 // CLKDIV=1 -> Fclk=25e6/2= 12.5
MHz
                   (1<<21)|                 // PDN=1
                   (6<<24);                 // MAT1.0 -> frecuencia de muestreo
LPC_ADC->ADINTEN=(3<<1);                   // Hab. Interrup. canal 1 y 2
NVIC_EnableIRQ(ADC_IRQn);                  // Ojo! ADC no interrumpe!!
NVIC_SetPriority(ADC_IRQn,2);
LPC_TIM1->MR0=6.250e6;   // MR0=Fpclk/N*2*Fs →Fs=1Hz ojo N=2 (N° de canales)
LPC_TIM1->MCR= 0x02;     // Reset on Match
LPC_TIM1->TCR=0x01;      // Start Timer
}


void ADC_IRQ_Handler(void)
{                              //Interrupciones periódicas cada 1/N*Fs= 0.5 seg.
LPC_ADC->ADCR&=0xFFFFFF00;     // Borramos canales
if(LPC_ADC->STAT&(1<<1)){      // Canal 1?
                LM35_temp=((LPC_ADC->ADDR1>>4)&0xFFF)*330/4095; //grados=mV/10
                 LPC_ADC->ADCR|=(1<<2);// seleccionamos canal 2
                }
else{
               volt_HIH=((LPC_ADC->ADDR2>>4)&0xFFF)*3.3/4095;
               HIH4000_hum=(volt_HIH/3.1 - 80/3.1) ;      //Humedad
               LPC_ADC->ADCR|=(1<<1);// seleccionamos canal 1
}
```

c) Complete la función de configuración de la señal PWM que actúa sobre el ventilador. Considere una frecuencia de la señal PWM de 1Khz. **Dibuje** la **evolución del Timer** y la salida PWM1.3 para un ciclo de trabajo de 0.25 y escriba todos los parámetros sobre el gráfico.

```
void config_pwm_motor(void)
{
```

```
LPC_PINCON->PINSEL3|=(2 <<10);   // P1.21 salida PWM (PWM1.3)
LPC_SC->PCONP|=(1<<6);           //Power PMW module
LPC_PWM1->MR0= (F_pclk/F_pwm)-1; //Fpwm=1kHz
LPC_PWM1->PCR|=(1<<11);          //Habilita ENA3 (PWM1.3)
LPC_PWM1->MCR|=(1<<1);           //Reset on Match
LPC_PWM1->TCR|=(1<<0);           // Start Timer
}
```

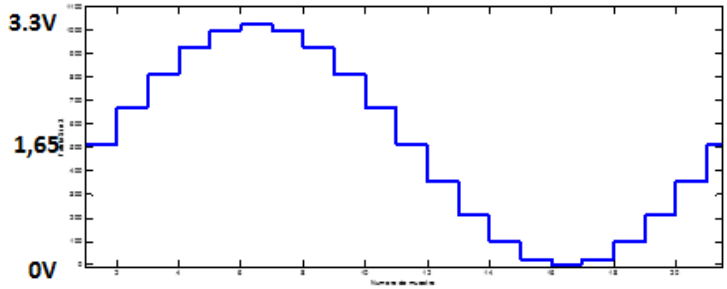


d) Explique el recurso interno utilizado para conectar el anemómetro, y escriba la función de interrupción que permite actualizar la variable global ***velocidad***, a partir de la frecuencia entregada por el sensor del anemómetro.
NOTA: *velocidad=k*frecuencia*

Para medir la frecuencia de la señal utilizamos el timer 2 en modo captura (subida o bajada) habilitando la interrupción. Tan sólo es necesario determinar las cuentas entre el flanco actual y el anterior.

```
void TIMER2_IRQHandler(void)
{
static uint32_t temp;
Borrar flag;
N=LPC_TIM2->CR0-temp;
velocidad=K*(Fpclk/N);
temp=LPC_TIM2->CR0; // Cuentas primer flanco
}
```

e) Se desea también tener un tono de 1Khz como señal de alarma, almacenando las muestras en memoria. Si se conectara un osciloscopio al pin P0.26 en reproducción aparece la señal de la figura:



Complete la función:

```
void genera_muestras_seno(void)
{
char i;
for(i=0;i< 20;i++) muestras[i]=(uint8_t)( 127 + 127*sin(2*pi*i/20  );
}
```

f) Escriba la función de interrupción del **Timer 3** que saca las muestras hacia el DAC para generar la señal senoidal e indique el periodo de interrupción.

```
void TIMER3_IRQHandler(void)
{
static uint16_t indice_muestras;
LPC_TIM3->IR|=(1<<0);                              // Clear flag MR0
LPC_ADC_DACR=muestras[indice_muestras++]<<8;       // DAC 8 bits
if(indice_muestras==20) indice_muestras=0;
}
```

Ts = 1/20*Fout= 50us.

g) Explique en detalle, sin escribir código, qué recurso utilizaría y cómo lo configuraría para **reducir al máximo** la carga de CPU durante la generación de la señal senoidal.

Evitaríamos la ejecución de la interrupción del Timer3 empleando el DMA en modo Linked con el propio canal (ej. Canal 0) sin que se produzcan interrupciones.

LPC_GPDMACH0->DMACCSrcAddr = (uint32_t) &muestras[0]; // Fuente Memoria
LPC_GPDMACH0->DMACCDestAddr = (uint32_t) &(LPC_DAC->DACR) **+ 1**; //Destino Periférico (**ojo 8bits**)
LPC_GPDMACH0->DMACCLLI= &LLI0; // dir. estructura de datos:

LLI0.source = (uint32_t) &muestras[0];
LLI0.destination = (uint32_t) &(LPC_DAC->DACR+1);
LLI0.next = (uint32_t) &LLI0;
LLI0.control = *parámetros de configuración:

Transfer Size=2**0**;
Ancho transferencia= **8 bits**;
Incrementa Fuente;
No incremento Destino;
Sin interrupción

LPC_DAC->DACCNTVAL = (F_pclk/1000/20) -1 = **1249**; // Frecuencia transf. DMA hacia DAC

h) Escriba el código de la función de interrupción del **SysTick**, encargada de **mostrar por el LCD** y de **enviar por el puerto serie** (UART0) los valores de la temperatura de la CPU y humedad **cada 30 segundos**, y la velocidad del viento **cada segundo**. Considere ejecutada la función ***SysTick_Config(5000000)*.**

```
void SysTick_IRQHandler(void)
{
   cont++;
   if(cont%600==0){
                    sprintf(cadena1,"temperatura_CPU= %2.1f grados",LM35_temp);
                    GUI_Text(10,10, cadena1, WHITE, BLACK);
                    sprintf(cadena2,"Humedad= %2.1f %",HIH4000_hum);
                    GUI_Text(10,50, cadena2, WHITE, BLACK);
                    tx_cadena_uart0(cadena1); while(tx_completa==0);
                    tx_cadena_uart0(cadena2); while(tx_completa==0);
```

```
                }

    if(cont%20==0){
                    sprintf(cadena,"velocidad= %3.1f m/s",velocidad);
                    GUI_Text(10,100, cadena, WHITE, BLACK);
                    tx_cadena_uart0(cadena); while(tx_completa==0);
                }
}
```

i) Complete la conexión de los sensores de temperatura interior y exterior (DS1621) y escriba la función que obtiene dichos valores con una **resolución de 0.5 grados**, modificando las variables globales ***Temp_exterior***, y ***Temp_interior*** a partir de la información del **Anexo IV**. **NOTA: Considere ya configurados los sensores.**

```
void DS1621(void)
{

I2CSendAddr(0x48,0);
I2CSendByte(0xAA);
I2CSendByte(0x48,1);
```

```
void I2CSendByte(unsigned char byte);
void I2CSendAddr(unsigned char addr, unsigned char rw);
unsigned char I2CGetByte(unsigned char ACK);
void I2CSendStop(void);
```

```
#include <LPC17xx.h>
#include "i2c_lpc17xx.h"
#define SDA 0 //pin 0
#define SCL 1 //pin 1
```

```
void I2CSendByte(unsigned char byte)
{
 unsigned char i;
 for(i=0;i<8;i++){

   if (byte &0x80)  LPC_GPIO0->FIOSET=(1<<SDA);
   else  LPC_GPIO0->FIOCLR=(1<<SDA);
   byte = byte <<1;    // siguiente bit
   pulso_SCL();
  }
```

```
Temp_exterior=(float)(I2CGetByte(0)+(I2CGetByte(1)>>7)*0.5);
I2CSendStop();
I2CSendAddr(0x4F,0);
I2CSendByte(0xAA);
I2CSendByte(0x4F,1);
Temp_interior=(float)(I2CGetByte(0)+(I2CGetByte(1)>>7)*0.5);
I2CSendStop();
}
```

j) Complete la función **simplificada** de configuración de la velocidad del puerto serie (considerando FDR=ResetValue). Dibuje la forma de la señal de salida de la uart3 al ejecutarse *tx_cadena_uart0(**"1. Temperatura"**)*, durante la transmisión del primer carácter, **e indique sobre ella los parámetros temporales.**

NOTA: Considere para los cálculos**, minimizar el error en la velocidad real obtenida,** si se llama a *uart0_set_baudrate(19200)* en la configuración.

```
void uart0_set_baudrate(int baudrate)
{
LPC_UART0->LCR |= DLAB_ENABLE;
LPC_UART0->DLM = 0;
LPC_UART0->DLL = 81;
LPC_UART0->LCR &= ~DLAB_ENABLE;
}
```

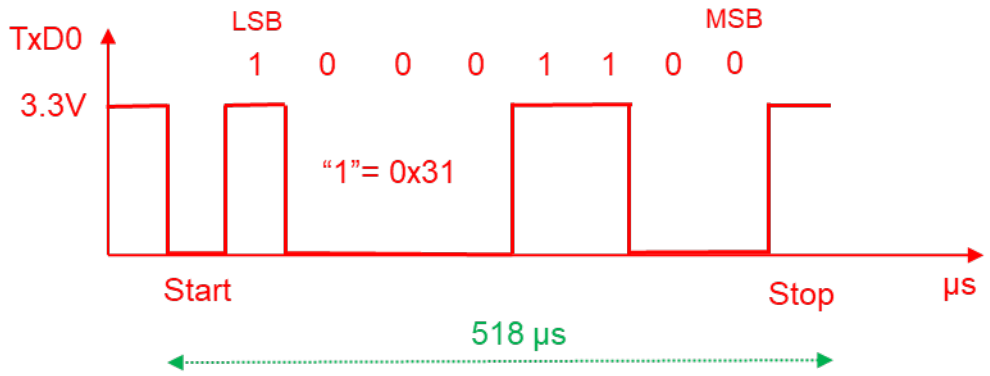
$$UARTn_{baudrate} = \frac{PCLK}{16 \times (256 \times UnDLM + UnDLL) \times \left(1 + \frac{DivAddVal}{MulVal}\right)}$$

Consideramos FR=1

DL_{16}= 25e6/(16*19200)*FR = 81,38 → **81**

Vt= 25e6/(16*81) = **19290 baudios**

Tcarácter= (1/Vt) * 10 bits = **0,518 ms**



CUESTIÓN 2

Se desea realizar el control de las barreras de un parking universitario con tres zonas de acceso. Cada una de las zonas de acceso tiene dos barreras (una de entrada y otra de salida) con los siguientes elementos:

- Un sensor que detecta la presencia de un vehículo delante de la barrera. Hay uno antes de la barrera de entrada (**COCHE_IN**) y otro antes de la barrera de salida (**COCHE_OUT**). Los sensores entregan un nivel alto cuando se detecta que hay un coche.
- Un sensor de infrarrojos debajo de cada barrera detecta si hay un vehículo justo debajo de la barrera (**S_IR_IN** y **S_IR_OUT**).
- Un sensor final de carrera en cada barrera que detecta cuando la barrera está bajada (**ABAJO_IN** y **ABAJO_OUT**) y otro detecta que la barrera está completamente subida (**ARRIBA_IN, ARRIBA_OUT**)
- Cada una de las barreras tiene un motor que mueve la barrera con dos señales de control **ABRE_IN**, **CIERRA_IN**, **ABRE_OUT** y **CIERRA_OUT** con el siguiente comportamiento:
-

Salida. ABRE_x.	Estado CIERRA_x	
0.	0	Barrera parada
0.	1	Cerrando barrera
1.	0.	Abriendo barrera
1	1.	Barrera parada

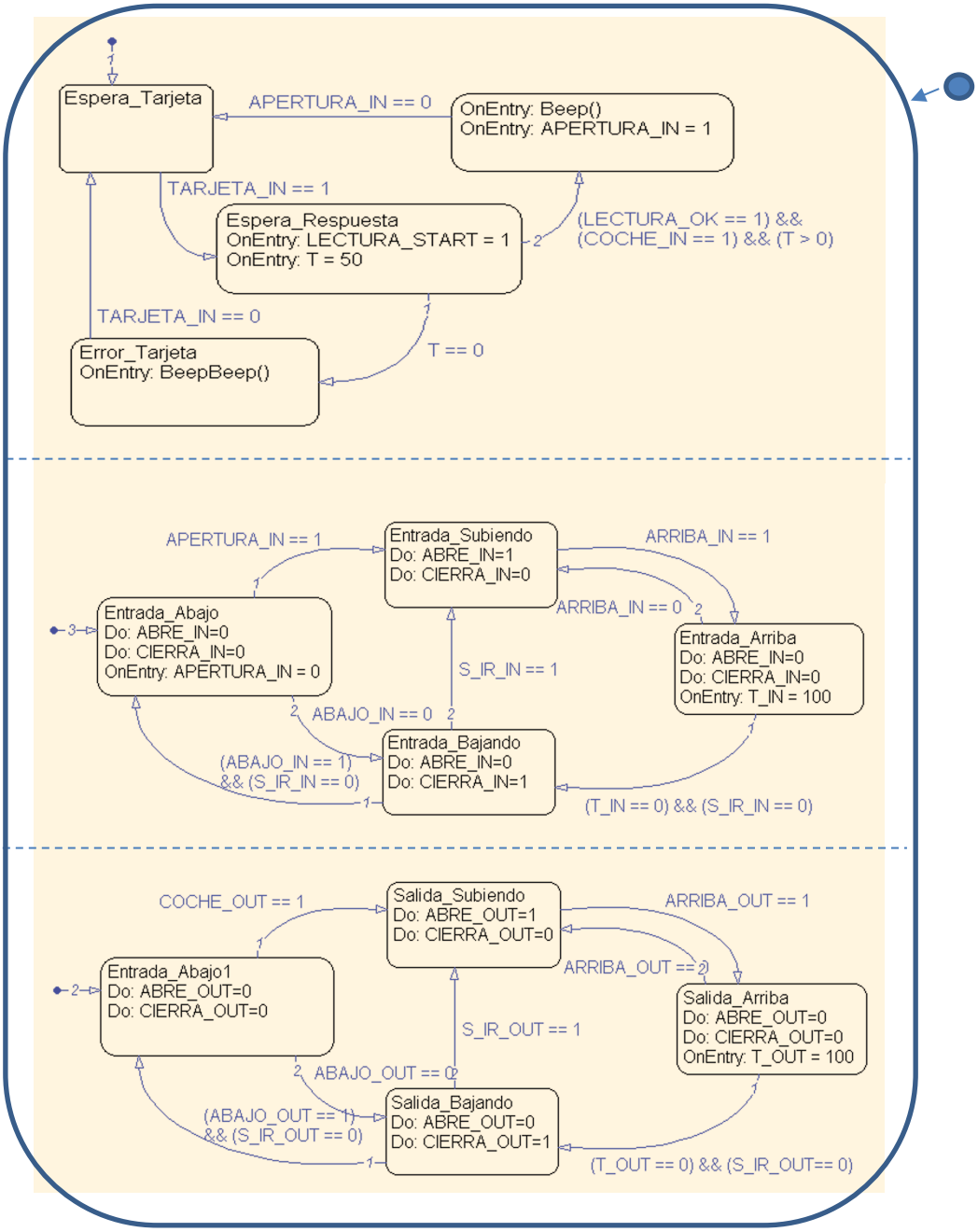
- Situada junto a la barrera de entrada hay una ranura donde se debe introducir la tarjeta universitaria del usuario que tiene asociado un subsistema con las siguientes **entradas y salidas**:
    - **TARJETA_IN**. Se activa a nivel alto mientras la tarjeta está introducida.
    - **LECTURA_START**: Orden que el sistema de control debe dar al sistema de lectura de la tarjeta para que éste realice una lectura de la información de la tarjeta y compruebe si es una tarjeta universitaria válida.
    - **LECTURA_OK**. El módulo activa con un nivel alto cuando se detecta una lectura de una tarjeta universitaria válida. Esta señal pasa de nuevo a nivel bajo al extraer la tarjeta.

El sistema tendrá la siguiente funcionalidad

- Si se introduce la tarjeta en la barrera de entrada y se recibe una señal de LECTURA_OK **antes de 5 segundos** se genera un pitido llamando a la función ***Beep()*** y se abre la barrera. **Si en 5 segundos no hay respuesta** se genera un **doble pitido** llamando a la función ***BeepBeep()*** y no se levanta la barrera, siendo necesario retirar la tarjeta y volver a insertarla para intentar leerla de nuevo.
- La barrera de entrada debe subirse cuando se reconoce una tarjeta universitaria válida pero sólo si se detecta un coche en la entrada. Si no se detecta coche no debe subirse.
- La barrera de salida debe abrirse cuando se detecta presencia de un vehículo delante de la barrera de salida.
- Las barreras deben permanecer abiertas al menos **10 segundos** y no deben bajar mientras detecte un objeto por el sensor de infrarrojos correspondiente. Si la detección se produce mientras está bajando, debería volver a subir reiniciándose la cuenta.
- Si estando la barrera cerrada, se detecta que se está abriendo manualmente (ABAJO_x == 0) ésta debe mantenerse cerrada. Lo mismo sucede cuando la barrera está abierta y se detecta que se está cerrando (ARRIBA_X == 0)

Se pide:

a) Realice una tabla donde indique las señales de entrada y de salida del sistema de control de una zona de acceso.
b) Realice el SatateChart que gestiona el funcionamiento de una zona de acceso.
c) Explique detalladamente la forma de realizar las temporizaciones.



Entradas:

ARRIBA_IN, ARRIBA_OUT, ABAJO_IN, ABAJO_OUT, S_IR_IN, S_IR_OUT, COCHE_IN, COCHE_OUT, LECTURA_OK, TARJETA_IN

Salidas:

ABRE_IN, ABRE_OUT, CIERRA_IN, CIERRA_OUT, LECTURA_START,

Como elemento de comunicación interna también se utiliza como lectura y escritura la variable APERTURA_IN y como variables de temporización T, T_IN, y T_OUT

La temporización se podría realizar utilizando una interrupción periódica que se ejecute cada 100ms donde se incluya estas tres instrucciones: if (T>0) T--; if (T_OUT>0) T_OUT--; if (T_IN>0) T_IN--;

CUESTIÓN 3

Se dispone de un sistema de control de la activación remota de la iluminación de un escaparate y el hilo musical de un comercio. El sistema está basado en el LPC1768 que, utilizando las librerías de comunicaciones de Keil, presenta dos páginas web para monitorizar y actuar sobre el sistema (**iluminacion.cgi** y **musica.cgi**) cuya apariencia y código HTML puede verse en la información adjunta.

Para actuar sobre el Hardware están implementadas las siguientes funciones:

- ActivaLuz(n): Enciende la luz del escaparate de la zona 1 (n=1) o de la zona 2 (n= 2).
- ApagaLuz(n): Apaga la luz del escaparate de la zona 1 (n=1) o de la zona 2 (n= 2).
- EnciendeMusica(): Activa el hilo musical.
- ApagaMusica(): Apaga el hilo musical.

Escriba el contenido de los ficheros **iluminacion.cgi**, **musica.cgi** y de las funciones **cgi_func(...)**, **cgi_process_var(...)** y/o **cgi_process_data(...)** que sea necesario.

iluminación.cgi

Iluminación Hilo musical

Escaparate 1: Apagado Encender

Escaparate 2: Encendido Apagar

```
<html>
<body>
<table style="height: 23px;" width="300">
<tbody>
<tr>
<td style="width: 139.533px; text-align: center;"><strong>Iluminaci&oacute;n</strong></td>
<td style="width: 144.467px; text-align: center;"><a href="musica.cgi">Hilo musical</a></td>
</tr>
</tbody>
</table>
<form action="iluminacion.cgi" method="get">
<p><strong>Escaparate 1:</strong> Apagado <button name="Escaparate_1" type="submit" value="ON">Encender</button></p>
<p><strong>Escaparate 2:</strong> Encendido <button name="Escaparate_2" type="submit" value="OFF">Apagar</button></p>
</form>
 </body>
</html>
```

musica .cgi

Iluminación **Hilo Musical**

Hilo musical: Apagado Encender

```
<html>
<body>
<table style="height: 23px;" width="300">
<tbody>
<tr>
<td style="width: 139.533px; text-align: center;"><a href="iluminacion.cgi">Iluminaci&oacute;n</a></td>
<td style="width: 144.467px; text-align: center;"><strong>Hilo Musical</strong></td>
</tr>
</tbody>
</table>
<form action="musica.cgi" method="post">
<p><strong>Hilo musical:</strong> Apagado <button name="Musica" type="submit" value="ON">Encender</button></p>
</form>
 </body>
</html>
```

Fichero iluminación.cgi

```
<html>
<body>
<table style="height: 23px;" width="300">
<tbody>
<tr>
<td style="width: 139.533px; text-align: center;"><strong>Iluminaci&oacute;n</strong></td>
<td style="width: 144.467px; text-align: center;"><a href="musica.cgi">Hilo musical</a></td>
</tr>
</tbody>
</table>
<form action="iluminacion.cgi" method="get">
<p><strong>Escaparate 1:</strong>  %s  <button name="Escaparate_1" type="submit" value="%"> %s </button></p>
<p><strong>Escaparate 2:</strong>  %s  <button name="Escaparate_2" type="submit" value="%s"> %s </button></p>
</form>
 </body>
</html>
```

Fichero música.cgi

```
<html>
<body>
<table style="height: 23px;" width="300">
<tbody>
<tr>
<td style="width: 139.533px; text-align: center;"><a href="iluminacion.cgi">Iluminaci&oacute;n</a></td>
<td style="width: 144.467px; text-align: center;"><strong>Hilo Musical</strong></td>
</tr>
</tbody>
</table>
<form action="musica.cgi" method="post">
<p><strong>Hilo musical:</strong>  %s  <button name="Musica" type="submit" value="%"> %s </button></p>
</form>
 </body>
</html>
```

Suponemos que existen las siguientes funciones:

- Escaparate(n) que devuelve un 0 si el escaparate correspondiente está apagado y un 1 si está encendido.
- Musica() que devuelve un 0 si el hilo musical está apagado y un 1 si está encendido.

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

const char * mensaje1[] = {"Apagado","Encendido"};
const char * mensaje2[] = {"ON","OFF"};
const char * mensaje3[] = {"Encender","Apagar"};


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

  switch (env[0]) {
    case 'a' :
       switch (env[2]) {
          case '1' :
          case '2' :
             i = Escaparate(env[2]-'0');
             len = sprintf((char *)buf,(const char *)&env[4],
                           mensaje1[i],mensaje2[i],mensaje3[i]);
             break;
          default:
             break;
       }
       break;

    case 'b' :
       switch (env[2]) {
          case '1' :
             i = Musica();
             len = sprintf((char *)buf,(const char *)&env[4],
                           mensaje1[i],mensaje2[i],mensaje3[i]);
             break;
          default:
             break;
       }
       break;
    default:
      break;
  }
  return ((U16)len);
}
```

```
void cgi_process_var (U8 *qs) {
  U8 *var;

  var = (U8 *)alloc_mem (40);
  do {
    qs = http_get_var (qs, var, 40);
    if (var[0] != 0) {
      if (str_scomp (var, "Escaparate_1=ON") == __TRUE) {
         ActivaLuz(1);
      }
      else if (str_scomp (var, "Escaparate_1=OFF") == __TRUE) {
         ApagaLuz(1));
      }
      else if (str_scomp (var, "Escaparate_2=ON") == __TRUE) {
         ActivaLuz(2);
      }
      else if (str_scomp (var, "Escaparate_2=OFF") == __TRUE) {
         ApagaLuz(2));
      }
    }
  }while (qs);
  free_mem ((OS_FRAME *)var);
}

void cgi_process_data (U8 code, U8 *dat, U16 len) {
  U8 passw[12],retyped[12];
  U8 *var,stpassw;

  var = (U8 *)alloc_mem (40);
  do {
    dat = http_get_var (dat, var, 40);
    if (var[0] != 0) {

      if (str_scomp (var, "Musica=ON") == __TRUE) {
         EnciendeMusica();
      }
      else if (str_scomp (var, "Musica=OFF") == __TRUE) {
         ApagaMusica());
      }
  }while (dat);
  free_mem ((OS_FRAME *)var);
}
```

Nota: Para que el programa funcione será necessrio cambiar el código HTML correspondiente a las líneas del comando C de los ficheros CGI donde aparecen “ por \”

CUESTIÓN 4.

Suponiendo que un sistema basado en el LPC1768 tiene cinco tareas con los parámetros que se indican en la tabla, y teniendo en cuenta que existe una **región crítica** de **5ms** en la Tarea B, **2ms** en la Tarea C y en el programa principal de **3ms**.

Tarea	Prioridad	Subprioridad	C	T	D
Tarea A	0	0	1	10	10
Tarea B	1	0	10	50	30
Tarea C	1	1	15	100	40
Tarea D	1	2	20	100	80
Tarea E	2	1	30	100	100

Nota: Unidades en ms

Se pide:

a) Analice los bloqueos que hay que tener en cuenta en la ejecutabidad del sistema.
b) Analice la ejecutabilidad de la **Tarea D**

Bloqueos:

Tarea A: B_A=max{RC_B, RC_C, RC_{PP}} = max{5, 2, 3} = 5 ms
Tarea B: B_B=max{C_c, C_D, RC_{PP}} = max{15, 20, 3} = 20 ms
Tarea C: B_C=max{ C_D, RC_{PP}} = max{20, 3} = 20 ms
Tarea D: B_D= RC_{PP} = 3 ms
Tarea E: B_E= RC_{PP} = 3 ms

$$R_D = C_D + B_D + I_D = C_D + B_D + C_A.\left\lceil\frac{R_D}{T_A}\right\rceil + C_B.\left\lceil\frac{R_D - C_D}{T_B}\right\rceil + C_C.\left\lceil\frac{R_D - C_D}{T_C}\right\rceil$$
$$= 20 + 3 + 1.\left\lceil\frac{R_D}{10}\right\rceil + 10.\left\lceil\frac{R_D - 20}{50}\right\rceil + 15.\left\lceil\frac{R_D - 20}{100}\right\rceil$$

$$w^0 = 20 + 3 + 1 + 10 + 15 = 49$$

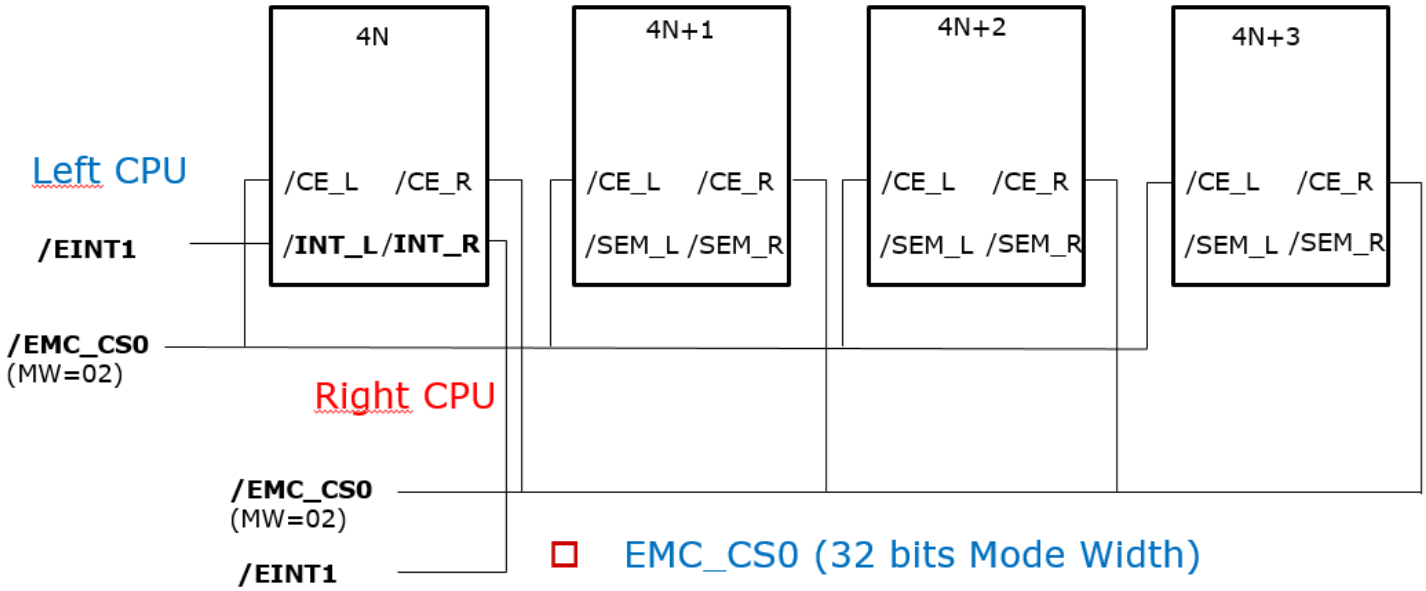
$$w^1 = 20 + 3 + 1.\left\lceil\frac{w^0}{10}\right\rceil + 10.\left\lceil\frac{w^0 - 20}{50}\right\rceil + 15.\left\lceil\frac{w^0 - 20}{100}\right\rceil = 20 + 3 + 1.\left\lceil\frac{49}{10}\right\rceil + 10.\left\lceil\frac{49 - 20}{50}\right\rceil + 15.\left\lceil\frac{49 - 20}{100}\right\rceil$$
$$= 20 + 3 + 5 + 10 + 15 = 53$$
$$w^2 = 20 + 3 + 1.\left\lceil\frac{w^1}{10}\right\rceil + 10.\left\lceil\frac{w^1 - 20}{50}\right\rceil + 15.\left\lceil\frac{w^1 - 20}{100}\right\rceil = 20 + 3 + 1.\left\lceil\frac{53}{10}\right\rceil + 10.\left\lceil\frac{53 - 20}{50}\right\rceil + 15.\left\lceil\frac{53 - 20}{100}\right\rceil$$
$$= 20 + 3 + 6 + 10 + 15 = 54$$

$$w^3 = 20 + 3 + 1.\left\lceil\frac{w^2}{10}\right\rceil + 10.\left\lceil\frac{w^2 - 20}{50}\right\rceil + 15.\left\lceil\frac{w^2 - 20}{100}\right\rceil = 20 + 3 + 1.\left\lceil\frac{54}{10}\right\rceil + 10.\left\lceil\frac{54 - 20}{50}\right\rceil + 15.\left\lceil\frac{54 - 20}{100}\right\rceil$$
$$= 20 + 3 + 6 + 10 + 15 = 54$$

$$R_D = 54ms \leq D_D = 80ms. \longrightarrow Es\ ejecutable$$

CUESTIÓN 5.

Explique en detalle el método de arbitración por interrupción de las memorias DUAL-PORT y justifícalo sobre el esquema de la figura. Considere que el bloque de memoria está mapeado en la dirección 0x8000.0000 y que cada chip tiene una capacidad de **32Kbytes**.



El espacio de direccionamiento del bloque de memoria de la Dual-Port de la figura, para la CPU de la izquierda y derecha es de 128kBytes (0x8000.0000 – 0x8001.FFFF).

La arbitración por interrupción del bloque de memoria de la DUAL-PORT al utilizar las señales de interrupción del chip **4N,** utiliza las posiciones **0x8001.FFF8** y **0x8001.FFFC** (dos últimas posiciones del chip).

La CPU_L **escribe** en 0x8001.FFFC e interrumpe a la CPU_R activando $/INT_R$. La CPU_R lee dicha posición y desactiva o deja en reposo (nivel alto) la señal de interrupción ($/INT_R$)**.**

La CPU_R **escribe** en 0x8001.FFF8 e interrumpe a la CPU_L activando $/INT_L$. La CPU_L lee dicha posición y desactiva o deja en reposo (nivel alto) la señal de interrupción ($/INT_L$)**.**

CUESTIÓN 6.

Explique qué es un DSP y sus características fundamentales.

Ver solución

ANEXO I (Mapa de memoria LPC1768 y Registros ADC)

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

ANEXO II (Funciones de control del puerto serie y LCD)

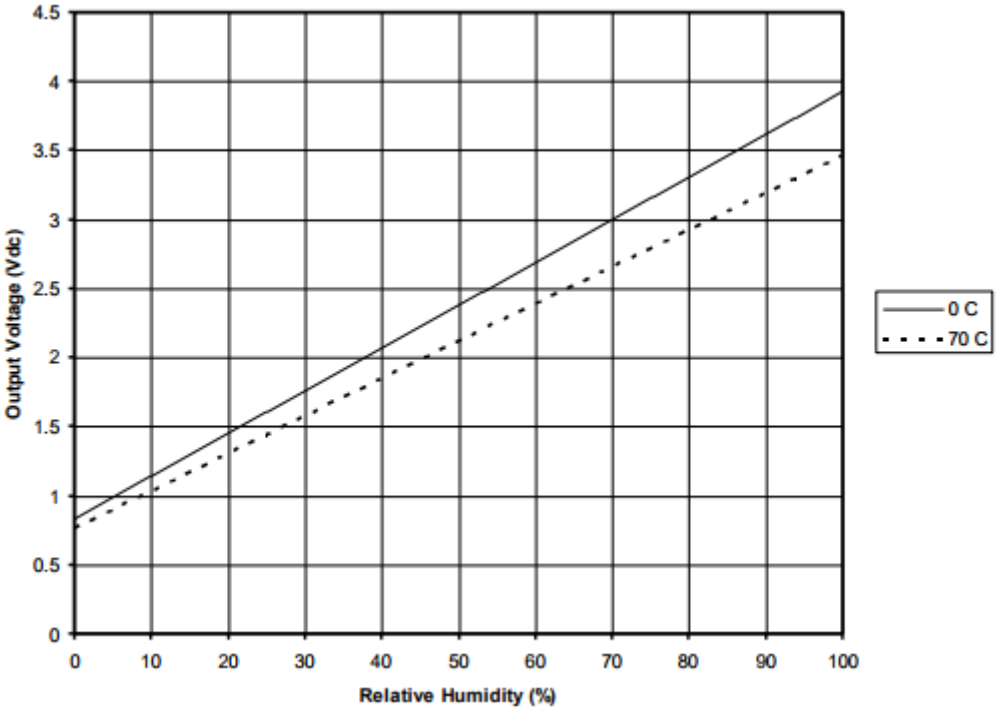
```
void UART0_IRQHandler(void) {

   switch(LPC_UART0->IIR&0x0E) {

   case 0x04:                  /* RBR, Receiver Buffer Ready */
     *ptr_rx=LPC_UART0->RBR;   /* lee el dato recibido y lo almacena */
     if(*ptr_rx++ ==13){       /* Caracter return --> Cadena completa */
       *ptr_rx=0;              /* Añadimos el caracter null para tratar los datos recibidos como una cadena*/
       rx_completa = 1;        /* rx completa */
       ptr_rx=buffer;          /* puntero al inicio del buffer para nueva recepción */
     }
     break;

   case 0x02:                      /* THRE, Transmit Holding Register empty */
     if(*ptr_tx!=0)
       LPC_UART0->THR = *ptr_tx++; /* carga un nuevo dato para ser transmitido */
     else
       tx_completa=1;
     break;
   }
}
```

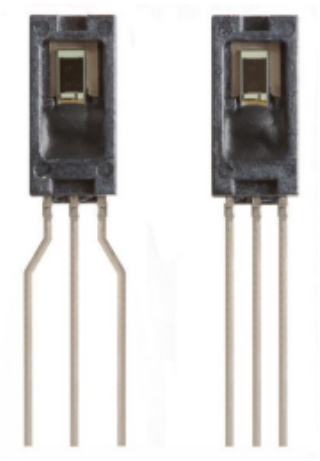
```
void uart0_init(int baudrate) {

    LPC_PINCON->PINSEL0 = (1 << 4) | (1 << 6);     // Change P0.2 and P0.3 mode to TXD0 and RXD0

    // Set 8N1 mode (8 bits/dato, sin paridad, y 1 bit de stop)
    LPC_UART0->LCR |= CHAR_8_BIT | STOP_1_BIT | PARITY_NONE;

    uart0_set_baudrate(baudrate);                 // Set the baud rate

    LPC_UART0->IER = THRE_IRQ_ENABLE | RBR_IRQ_ENABLE; // Enable UART TX and RX interrupt (for LPC17xx UART)
    NVIC_EnableIRQ(UART0_IRQn);                        // Enable the UART interrupt (for Cortex-CM3 NVIC)

}
```

```
void tx_cadena_UART0(char *cadena)
{
   ptr_tx=cadena;
   tx_completa=0;
   LPC_UART0->THR = *ptr_tx;   // IMPORTANTE: Introducir un carácter al comienzo para iniciar TX o
}                              // activar flag interrupción por registro transmisor vacio
```

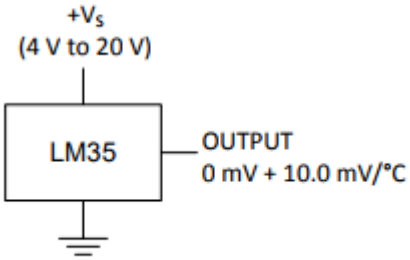
```
/* Private function prototypes -----------------------------------------------*/
void LCD_Initializtion(void);
void LCD_Clear(uint16_t Color);
uint16_t LCD_GetPoint(uint16_t Xpos,uint16_t Ypos);
void LCD_SetPoint(uint16_t Xpos,uint16_t Ypos,uint16_t point);
void LCD_DrawLine( uint16_t x0, uint16_t y0, uint16_t x1, uint16_t y1 , uint16_t color );
void PutChar( uint16_t Xpos, uint16_t Ypos, uint8_t ASCI, uint16_t charColor, uint16_t bkColor );
void GUI_Text(uint16_t Xpos, uint16_t Ypos, uint8_t *str,uint16_t Color, uint16_t bkColor);
void PutChinese(uint16_t Xpos,uint16_t Ypos,uint8_t *str,uint16_t Color,uint16_t bkColor);
void GUI_Chinese(uint16_t Xpos, uint16_t Ypos, uint8_t *str,uint16_t Color, uint16_t bkColor);
```

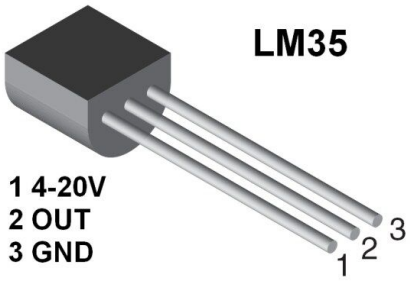
ANEXO III (Características sensores HIH4000 y LM35, Vcc=+5V)





Basic Centigrade Temperature Sensor (2°C to 150°C)





ANEXO IV (DS1621)

Table 3. DS1621 COMMAND SET

INSTRUCTION	DESCRIPTION	PROTOCOL	2-WIRE BUS DATA AFTER ISSUING PROTOCOL	NOTES
TEMPERATURE CONVERSION COMMANDS				
Read Temperature	Read last converted temperature value from temperature register.	AAh	<read 2 bytes data>	
Read Counter	Reads value of Count_Remain	A8h	<read data>	
Read Slope	Reads value of the Count_Per_C	A9h	<read data>	
Start Convert T	Initiates temperature conversion.	EEh	idle	1
Stop Convert T	Halts temperature conversion.	22h	idle	1
THERMOSTAT COMMANDS				
Access TH	Reads or writes high temperature limit value into TH register.	A1h	<write data>	2
Access TL	Reads or writes low temperature limit value into TL register.	A2h	<write data>	2
Access Config	Reads or writes configuration data to configuration register.	ACh	<write data>	2