Anexo II – Código fuente.

Header.h

```
/**-------------------------------------------------------------------------------------------------------------//
//      @filename     header.h                                                                                  //
//      @version      0.00                                                                                      //
//      @author       Alberto Palomo Alonso                                                                     //
//                                                                                                              //
//      @brief      Cabezera del código fuente, agrupa todos los archivos.                                      //
//                                                                                                              //
//      @category     Principal.                                                                                //
//                                                                                                              //
//      @map          @include                                                                                  //
//                  @end                                                                                        //
//                                                                                                              //
//                                                                                                              //
//-------------------------------------------------------------------------------------------------------------//
//                                                                                                              //
                                                                                                                //
//      @include      Incluye todos los archivos necesarios.                                                    //
//                                                                                                              //
//-------------------------------------------------------------------------------------------------------------**/
//  Librería de registros.
#ifndef LPC17XX
#define LPC17XX
#include "LPC17XX.H"
#endif
//  Símbolos del sistema.
#ifndef SYSTEMSYMBOLS
#define SYSTEMSYMBOLS
#include "Systemsymbols.h"
#endif
//  Pwm.
#ifndef     PWM
#define     PWM
#include "PWM.h"
#endif
//  Configura
#ifndef CONFIGURA
#define     CONFIGURA
#include "configura.h"
#endif
#ifndef STATECHART
#define     STATECHART
#include "Statechart.h"
#endif
#ifndef GLCD
#define     GLCD
#include    "GLCD.h"
#include    "TouchPanel.h"
#include    "menu.h"
#include    "leds.h"
#endif
#ifndef STATECHART
#define     STATECHART
#include    "Statechart.h"
#endif
#ifndef HTTPSOURCE
#define     HTTPSOURCE
#include "HTTP_SOURCE.h"
#endif
#ifdef  DEBUG
#include    "DEBUG.h"
```

```
#endif
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
//                                                                                                                   //
//      @end        ENDFILE.                                                                                         //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
```

Systemsymbols.h

```
/**-------------------------------------------------------------------------------------------------------------------//
//      @filename       Systemsymbols.h                                                                              //
//      @version        0.00                                                                                         //
//      @author         Alberto Palomo Alonso                                                                        //
//                                                                                                                   //
//      @brief      Este es el archivo donde son guardados los símbolos del sistema utilizados para los             //
//                  diferentes archivos que necesiten una llamada a este tipo de definiciones.                      //
//                                                                                                                   //
//      @category       Esencial.                                                                                    //
//                                                                                                                   //
//      @map            @none                                                                                       //
//                  @types                                                                                          //
//                  @end                                                                                            //
//-------------------------------------------------------------------------------------------------------------------**/
#ifndef LPC17XX
#define     LPC17XX
#include    "LPC17XX.H"
#endif
//  Acopladores de código.
#ifndef         null
#define         null                0
#endif
#ifndef     NULL
#define         NULL                0
#endif
#define         none                0
#define         NONE                0
#define         VOID                void
//  Símbolos prácticos de valores instantáneos.
#define         TODO_1_8            0xFFFF
#define         TODO_1_16               0xFFFFFFFF
#define         TODO_1_32               0xFFFFFFFFFFFFFFFF
//  Símbolos correspondientes a definiciones propias para facilitar lectura de código.
#define         SUBIDA                  1
#define         BAJADA                  0
#define         ALTO                    1
#define         BAJO                    0
#define         FLANCO                  1
#define         NIVEL               0
#define         INICIALMENTE_ACTIVO    1
#define         INICIALMENTE_INACTIVO0
#define         FUNC0               0x0
#define         FUNC1               0x1
#define         FUNC2               0x2
#define         FUNC3               0x3
#define         NOPIN                   0x00000000
#define         PIN_00              0x00000001
#define         PIN_01                  0x00000002
#define         PIN_02                  0x00000004
#define         PIN_03                  0x00000008
#define         PIN_04                  0x00000010
#define         PIN_05                  0x00000020
#define         PIN_06                  0x00000040
```

```
#define		PIN_07			0x00000080
#define		PIN_08			0x00000100
#define		PIN_09			0x00000200
#define		PIN_10			0x00000400
#define		PIN_11			0x00000800
#define		PIN_12			0x00001000
#define		PIN_13			0x00002000
#define		PIN_14			0x00004000
#define		PIN_15			0x00008000
#define		PIN_16			0x00010000
#define		PIN_17			0x00020000
#define		PIN_18			0x00040000
#define		PIN_19			0x00080000
#define		PIN_20			0x00100000
#define		PIN_21			0x00200000
#define		PIN_22			0x00400000
#define		PIN_23			0x00800000
#define		PIN_24			0x01000000
#define		PIN_25			0x02000000
#define		PIN_26			0x04000000
#define		PIN_27			0x08000000
#define		PIN_28			0x10000000
#define		PIN_29			0x20000000
#define		PIN_30			0x40000000
#define		PIN_31			0x80000000
//	Símbolos correspondientes a relojes del sistema.
#define		Fcpu		100000000			//	100MHz velocidad de la cpu.
#define		APBvalue	4				//	Valor del primer prescaler, 4 afer-reset.
#define		Prescaler	0				//	Valor del prescaler, 0 after-reset.
#define		Fclk		(float)Fcpu/(float)APBvalue	//	Valor del reloj prescalado por APB.
#define		Ftick		Fclk/(float)(Prescaler+1)	//	Valor del reloj asociado a los contadores.

#define		Ts0		0.5			//	Tiempo de muestreo en segundos sin prescaler. (Muestras)
#define		Fs0		(float)1/(float)Ts0		//	Frecuencia de muestreo en Hz. (Muestras)
#define		CsADC		Fs0			//	Frecuencia de muestreo del LDR.
#define		CsCAP		10*Fs0			//	Frecuencia de muestreo del UVA.

#define		FsAudio		8000			//	3kHz de audio, Nyquist *= 2,  Yo *= 8khz.
#define		TsAudio		(float)1/(float)FsAudio		//	Periodo de muestreo del audio.
#define		DURACION_AUDIO	2			//	2 segundos de audio.
#define		MUESTRAS_AUDIO	DURACION_AUDIO*FsAudio		//	Muestras en los 2 segundos de audio.
#define		MUESTRAS_SENO	32
#define		LECTURA_AUDIO	25		//	Pin que señaliza lectura de audio.
#define		ESCRITURA_AUDIO	26		//	Pin que señaliza escritura de audio.

#define	MAX_PRES	MODIFICABLES.Max_servo_p
#define	MAX_TEMP	MODIFICABLES.Max_servo_t
#define	MIN_PRES	MODIFICABLES.Min_servo_p
#define	MIN_TEMP	MODIFICABLES.Min_servo_t

//	Constantes universales.
#define		PI		3.141592
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
//                                                                                                                   //
//	@types		Tipos utilizados para el programa.                                                             //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
typedef signed char		int8_t;
typedef signed short int	int16_t;
typedef signed int		int32_t;
typedef signed long long	int64_t;

typedef unsigned char		uint8_t;
typedef unsigned short int	uint16_t;
typedef unsigned		int	uint32_t;
```

```
typedef unsigned long long       uint64_t;

typedef    struct {          //  Contadores de 8, 16 y 32 bits.
    __IO   uint8_ti;
    __IO   uint8_tj;
    __IO   uint16_t   k;
    __IO   uint32_t   Audio;
    __IO uint32_t   Segundos;
    __IO   uint32_t   RITicks;
}Counters_t;

typedef struct {
    uint8_tCHART;
}State_t;

typedef struct {
        float   Longitud;
        float   Latitud;
        float   Altura;
}locat_t;

typedef struct {
        float   Temperatura; //  En grados celsius.
        float   Presion;        //  En pascales.
        float   Humedad;     //  En  %.
        float   IndiceUV;       //  En  UVs.
    locat_t Lugar;      //  Sitio donde el GPS nos posiciona.
        float   VelViento; //  En      m/s.
        float   Brillo;     //  En      LUX.
}misDatos_t;

typedef struct {
    __IO uint8_t   Anemometro:1;
    __IO uint8_t   AnemometroRev:1;
    __IO uint8_t   LDR:1;
    __IO   uint8_tLDRrev:1;
    __IO   uint8_tUVA:1;
    __IO   uint8_tUVArev:1;
    __IO   uint8_tAudio:1;
    __IO   uint8_tAudiorev:1;
    __IO uint8_t   TempRev:1;
}actualizador_t;

typedef struct {
   uint32_t   source;          // Start of source area
   uint32_t   destination;// Start of destination area
   uint32_t   next;            // Address of next strLLI in chain
   uint32_t   control;      // DMACCxControl register
} DMA_t;

typedef struct {
    float   Max_servo_t;     //  Done
    float   Min_servo_t;     //  Done
    float   Max_servo_p;     //  Done
    float   Min_servo_p;     //  Done

    uint8_tVar_medida;       //  Done
    uint32_t   TiempoBrillo;      //  Done
}modificables_t;

/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
//      @end      ENDFILE.                                                                                           //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
```

Main.c

```
/**-------------------------------------------------------------------------------------------------------------------//
//      @filename       main.c                                                                                       //
//      @version        0.00                                                                                    //
//      @author         Alberto Palomo Alonso                                                                        //
//                                                                                                                   //
//      @brief      Código fuente del programa principal.                                                            //
//                                                                                                                   //
//      @category       Principal.                                                                                   //
//                                                                                                                   //
//      @map            @include                                                                                //
//                  @global                                                                                          //
//                  @main                                                                                        //
//                  @end                                                                                             //
//                                                                                                                   //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
                                                                                                                     //
//      @include        Estos son los archivos utilizados con el código fuente.                                      //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
#ifndef HEADER
#define     HEADER
#include    "header.h"
#endif
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
                                                                                                                     //
//      @global         Programa principal, variables globales.                                                      //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
misDatos_t          objDATOS;               //  Objeto.
misDatos_t      *   DATOS = &objDATOS;          //  Mis datos almacenados en la variable objDATOS.
State_t             objESTADO;              //  Objeto.
State_t     *   ESTADO = &objESTADO;        //  Declarar como extern. (Hey, compilador, creeme que hay una variable por ahí que se
llama ESTADO)
Counters_t          objCOUNTERS;            //  Objeto.
Counters_t      *   COUNTERS = &objCOUNTERS;    //  Declarar como extern. (Hey, compilador, creeme que hay una variable por ahí
que se llama COUNTERS)
actualizador_t      objACTUALIZADOR;            //  Objeto.
actualizador_t *    ACTUALIZADOR = &objACTUALIZADOR; //   Declarar como extern. (Hey, compilador, creeme que hay una variable por
ahí que se llama ACTUALIZADOR)
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
                                                                                                                     //
//      @main       Programa principal, inicio after-reset.                                                          //
//                                                                                                                   //
//      @ref            __configuraPrograma__   ->  configura.h                                                      //
//                  __mainLoop__                ->  statechart.h                                                     //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
int main    ()
{
    __configuraPrograma__();
    while ( 1   )
    {
        __mainLoop__();
        __mantenerTCP__();
    }
}
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
                                                                                                                     //
```

```
//      @end       ENDFILE.                                                                    //
//                                                                                             //
//-----------------------------------------------------------------------------------------------**/
```

Statechart.h

```
/**-----------------------------------------------------------------------------------------------//
//      @filename     Statechart.c                                                              //
//      @version      0.00                                                                      //
//      @author       Alberto Palomo Alonso                                                        //
//                                                                                              //
//      @brief     Cabecera del código fuente de Statechart.c                                      //
//                                                                                              //
//      @category     Principal.                                                                //
//                                                                                              //
//      @map          @include                                                                  //
//                 @private                                                                     //
//                 @types                                                                       //
//                 @funcdef                                                                     //
//                 @end                                                                         //
//                                                                                              //
//                                                                                              //
//-----------------------------------------------------------------------------------------------//
//                                                                                              //
                                                                                                    //
//      @include      Estos son los archivos utilizados con el statechart.                          //
//                                                                                              //
//-----------------------------------------------------------------------------------------------**/
//  LCD
#ifndef GLCD
#define    GLCD
#include   "GLCD.h"
#include   "TouchPanel.h"
#include   "menu.h"
#endif
#ifndef SYSTEMSYMBOLS
#define    SYSTEMSYMBOLS
#include   "Systemsymbols.h"
#endif
#ifndef STRING
#define    STRING
#include   <string.h>
#endif
#ifndef WTD
#define    WDT
#include "WDT.h"
#endif
#ifndef PWM
#define    PWM
#include   "PWM.h"
#endif
#ifndef STDIO
#define    STDIO
#include   <stdio.h>
#endif
#ifndef MIGLOBAL
#define    MIGLOBAL
#include   "miGlobal.h"
#endif
#ifndef TIMERS
#define    TIMERS
#include   "Timers.h"
```

```
#endif
#ifndef RTL
#define     RTL
#include    "RTL.h"
#endif
#ifndef HTTPSOURCE
#define     HTTPSOURCE
#include    "HTTP_SOURCE.h"
#endif
#ifndef LUT
#define     LUT
#include "LUT.h"
#endif
#ifndef DAC
#define     DAC
#include    "DAC.h"
#endif
#ifndef UFONO
#define     UFONO
#include    "uFono.h"
#endif
#ifndef ONEWIRE
#define     ONEWIRE
#include    "OneWire.h"
#endif
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
//                                                                                                             //
//      @private        Estos son los símbolos correspondientes al statechart.                                 //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
#define     PANTALLA_INICIO 0
#define     PANTALLA_MEDIDAS1     1
#define     PANTALLA_MEDIDAS2     2
#define     PANTALLA_AJUSTES   3
#define     PANTALLA_LOADING  4
#define     PANTALLA_VALORES  5

#define     MAXIMOX           240
#define     MAXIMOY           320
#define     CLEAR_BUFFER        "                                   "
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
//                                                                                                             //
//      @types          Tipos utilizados en el statechart.  COPIADOS DE menu.c                                 //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
typedef struct {
    uint16_t x;
    uint16_t y;
    uint16_t size_x;
    uint16_t size_y;
    uint8_t    pressed;
}screenZone_t;
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
//                                                                                                             //
//      @funcdef       Estas son las funciones correspondientes al statechart.                                 //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
void    __mainLoop__          (   void    );
void __configuraLCD__    (   void    );
void __pintaInicio__   (   void    );
void __pintaAjustes__     (   void    );
void    __pintaMedidas1__   (   void    );
void    __pintaMedidas2__   (   void    );
```

```
void squareButton         ( screenZone_t *   zone        , char *          text     , uint16_t  textColor   , uint16_t  lineColor);
void checkTouchPanel      (    void     );
int8_t zoneNewPressed     (screenZone_t * zone      );
void squareBox            ( screenZone_t *   zone        ,    uint16_t    color);
void __pintaCargandoSeno__     (    void     );
void __pintaCargandoConexion__(   void     );
void __pintaCargandoDone__     (    void     );
void __pintaCargandoInicio__   (    void     );
void __pintaCargandoIniciando__    (    void     );
void __pintaValores__          (    void     );
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
//                                                                                                                   //
//      @end       ENDFILE.                                                                                          //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
```

Statechart.c

```
/**-------------------------------------------------------------------------------------------------------------------//
//      @filename     Statechart.c                                                                                    //
//      @version      0.00                                                                                            //
//      @author       Alberto Palomo Alonso                                                                           //
//                                                                                                                    //
//      @brief     Código fuente correspondiente a la máquina de estados que compone el menú.                         //
//                                                                                                                    //
//      @category     Principal.                                                                                      //
//                                                                                                                    //
//      @map          @include                                                                                        //
//                 @global                                                                                            //
//                 @function                                                                                          //
//                 @end                                                                                               //
//                                                                                                                    //
//                                                                                                                    //
//-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                    //
//                                                                                                                    //
//      @include      Estos son los archivos utilizados en el statechart.                                             //
//                                                                                                                    //
//-------------------------------------------------------------------------------------------------------------------**/
#ifndef STATECHART
#define    STATECHART
#include   "Statechart.h"
#endif
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                    //
//                                                                                                                    //
//      @global       Estas son las variables globales pertenecientes al statechart.                                  //
//                                                                                                                    //
//-------------------------------------------------------------------------------------------------------------------**/
extern uint8_t          Clock[23];
extern State_t     *    ESTADO;
extern misDatos_t  *    DATOS;
extern actualizador_t * ACTUALIZADOR;
extern uint8_t          OWEjecutameExterno;
extern uint16_t         contadorLUZ;
modificables_t          MODIFICABLES;
char buffer[23];
```

```
uint8_tModo_energetico=0;
uint8_tModo_brillo=3;
uint8_tpressedTouchPanel;
uint8_t__brilloFade = 0;
uint8_t__brilloAuto = 0;
uint8_tAux8;
// ZONA DE PANTALLA DE INICIO.
screenZone_t zona_0= { 0 , 0 , MAXIMOX , MAXIMOY*0.2 , 0 }; // Marco del reloj y botones de alante y atrás.
screenZone_t zona_1= { MAXIMOX*0.15 , 0 , MAXIMOX*0.7 , MAXIMOY*0.2 , 0 }; // Reloj.
screenZone_t zona_2= { MAXIMOX*0.85 , 0 , MAXIMOX*0.15 , MAXIMOY*0.2 , 0 }; // Derecha.
screenZone_t zona_3= { MAXIMOX*0 , 0 , MAXIMOX*0.15 , MAXIMOY*0.2 , 0 }; // Izquierda.
screenZone_t zona_4= { MAXIMOX*0 , MAXIMOY*0.2 , MAXIMOX*0.5 , MAXIMOY*0.3 , 0 }; // Primer botón.
screenZone_t zona_5= { MAXIMOX*0.5 , MAXIMOY*0.2 , MAXIMOX*0.5 , MAXIMOY*0.3 , 0 }; // Segundo botón.
screenZone_t zona_6= { MAXIMOX*0 , MAXIMOY*0.5 , MAXIMOX , MAXIMOY*0.15 , 0 }; // Brillo info.
screenZone_t zona_7= { MAXIMOX*0 , MAXIMOY*0.65 , MAXIMOX*0.2 , MAXIMOY*0.15 , 0 }; // Primer botón de brillo.
screenZone_t zona_8= { MAXIMOX*0.2 , MAXIMOY*0.65 , MAXIMOX*0.2 , MAXIMOY*0.15 , 0 }; // Segundo botón de brillo.
screenZone_t zona_9= { MAXIMOX*0.4 , MAXIMOY*0.65 , MAXIMOX*0.2 , MAXIMOY*0.15 , 0 }; // Tercer botón de brillo.
screenZone_t zona_10 = { MAXIMOX*0.6 , MAXIMOY*0.65 , MAXIMOX*0.2 , MAXIMOY*0.15 , 0 }; // Cuarto botón de brillo.
screenZone_t zona_11 = { MAXIMOX*0.8 , MAXIMOY*0.65 , MAXIMOX*0.2 , MAXIMOY*0.15 , 0 }; // Brillo automático.
screenZone_t zona_12 = { MAXIMOX*0 , MAXIMOY*0.8 , MAXIMOX*0.2 , MAXIMOY*0.2 , 0 }; // Botón de audio.
screenZone_t zona_13 = { MAXIMOX*0.2 , MAXIMOY*0.8 , MAXIMOX*0.2 , MAXIMOY*0.2 , 0 }; // Volumen = 1.
screenZone_t zona_14 = { MAXIMOX*0.4 , MAXIMOY*0.8 , MAXIMOX*0.2 , MAXIMOY*0.2 , 0 }; // Volumen = 2.
screenZone_t zona_15 = { MAXIMOX*0.6 , MAXIMOY*0.8 , MAXIMOX*0.2 , MAXIMOY*0.2 , 0 }; // Volumen = 3.
screenZone_t zona_16 = { MAXIMOX*0.8 , MAXIMOY*0.8 , MAXIMOX*0.2 , MAXIMOY*0.2 , 0 }; // Botón de load.
// ZONA DE MEDIDAS.
screenZone_t zona_17 = { MAXIMOX*0 , MAXIMOY*0.2 , MAXIMOX , MAXIMOY*0.1 , 0 }; // Información de página, medidas.
screenZone_t zona_18 = { MAXIMOX*0 , MAXIMOY*0.3 , MAXIMOX , MAXIMOY*0.1 , 0 }; // Localización.
screenZone_t zona_19 = { MAXIMOX*0 , MAXIMOY*0.4 , MAXIMOX*0.5 , MAXIMOY*0.15 , 0 }; // Temperatura.
screenZone_t zona_20 = { MAXIMOX*0 , MAXIMOY*0.55 , MAXIMOX*0.5 , MAXIMOY*0.15 , 0 }; // Humedad.
screenZone_t zona_21 = { MAXIMOX*0 , MAXIMOY*0.70 , MAXIMOX*0.5 , MAXIMOY*0.15 , 0 }; // Presión.
screenZone_t zona_22 = { MAXIMOX*0 , MAXIMOY*0.85 , MAXIMOX*0.5 , MAXIMOY*0.15 , 0 }; // IndiceUV.
// ZONAS DE DATOS.
screenZone_t zona_23 = { MAXIMOX*0.5 , MAXIMOY*0.4 , MAXIMOX*0.5 , MAXIMOY*0.15 , 0 }; // Valor de temperatura.
screenZone_t zona_24 = { MAXIMOX*0.5 , MAXIMOY*0.55 , MAXIMOX*0.5 , MAXIMOY*0.15 , 0 }; // Valor de humedad.
screenZone_t zona_25 = { MAXIMOX*0.5 , MAXIMOY*0.70 , MAXIMOX*0.5 , MAXIMOY*0.15 , 0 }; // Valor de presión.
screenZone_t zona_26 = { MAXIMOX*0.5 , MAXIMOY*0.85 , MAXIMOX*0.5 , MAXIMOY*0.15 , 0 }; // Valor de IndiceUV.
// ZONAS DE AJUSTES.
screenZone_t zona_27 = { MAXIMOX*0 , MAXIMOY*0.2 , MAXIMOX*0.5 , MAXIMOY*0.15 , 0 }; // Horas.
screenZone_t zona_28 = { MAXIMOX*0 , MAXIMOY*0.35 , MAXIMOX*0.5 , MAXIMOY*0.15 , 0 }; // Minutos.
```

```
screenZone_t     zona_29  = {  MAXIMOX*0     ,  MAXIMOY*0.5   ,  MAXIMOX*0.5   ,  MAXIMOY*0.15  ,  0  }; //
    Segundos.
screenZone_t     zona_30  = {  MAXIMOX*0     ,  MAXIMOY*0.65  ,  MAXIMOX*0.5   ,  MAXIMOY*0.15  ,  0  }; //
    Dia.
screenZone_t zona_31  = {  MAXIMOX*0     ,  MAXIMOY*0.8   ,  MAXIMOX       ,  MAXIMOY*0.2   ,  0  }; // Slot
libre.
screenZone_t     zona_27m = {  MAXIMOX*0.5   ,  MAXIMOY*0.2   ,  MAXIMOX*0.25  ,  MAXIMOY*0.15  ,  0  }; //
    Resta horas.
screenZone_t     zona_28m = {  MAXIMOX*0.5   ,  MAXIMOY*0.35  ,  MAXIMOX*0.25  ,  MAXIMOY*0.15  ,  0  }; //
    Resta minutos.
screenZone_t     zona_29m = {  MAXIMOX*0.5   ,  MAXIMOY*0.5   ,  MAXIMOX*0.25  ,  MAXIMOY*0.15  ,  0  }; //
    Resta segundos.
screenZone_t     zona_30m = {  MAXIMOX*0.5   ,  MAXIMOY*0.65  ,  MAXIMOX*0.25  ,  MAXIMOY*0.15  ,  0  }; //
    Resta dia.
screenZone_t     zona_27M = {  MAXIMOX*0.75  ,  MAXIMOY*0.2   ,  MAXIMOX*0.25  ,  MAXIMOY*0.15  ,  0  }; //
    Suma horas.
screenZone_t     zona_28M = {  MAXIMOX*0.75  ,  MAXIMOY*0.35  ,  MAXIMOX*0.25  ,  MAXIMOY*0.15  ,  0  }; //
    Suma minutos.
screenZone_t     zona_29M = {  MAXIMOX*0.75  ,  MAXIMOY*0.5   ,  MAXIMOX*0.25  ,  MAXIMOY*0.15  ,  0  }; //
    Suma segundos.
screenZone_t     zona_30M = {  MAXIMOX*0.75  ,  MAXIMOY*0.65  ,  MAXIMOX*0.25  ,  MAXIMOY*0.15  ,  0  }; //
    Suma dia.
//  ZONAS DE MEDIDAS 2 (VIENTO)
screenZone_t zona_32  = {  MAXIMOX*0     ,  MAXIMOY*0.2   ,  MAXIMOX*0.5   ,  MAXIMOY*0.2   ,  0  }; //
    Velocidad del viento.
screenZone_t zona_32n = {  MAXIMOX*0.5   ,  MAXIMOY*0.2   ,  MAXIMOX*0.5   ,  MAXIMOY*0.2   ,  0  }; //
    Velocidad del viento.
screenZone_t zona_33  = {  MAXIMOX*0     ,  MAXIMOY*0.4   ,  MAXIMOX       ,  MAXIMOY*0.2   ,  0  }; //
    Velocidad del viento.
screenZone_t zona_34  = {  MAXIMOX*0     ,  MAXIMOY*0.6   ,  MAXIMOX*0.5   ,  MAXIMOY*0.2   ,  0  }; //
    Cantidad de brillo.
screenZone_t zona_34n = {  MAXIMOX*0.5   ,  MAXIMOY*0.6   ,  MAXIMOX*0.5   ,  MAXIMOY*0.2   ,  0  }; //
    Cantidad de brillo.
screenZone_t zona_35  = {  MAXIMOX*0     ,  MAXIMOY*0.8   ,  MAXIMOX       ,  MAXIMOY*0.2   ,  0  }; //
    Cantidad de brillo.
//  Display de barras.
screenZone_t zona_350 = {  MAXIMOX*0     ,  MAXIMOY*0.8   ,  MAXIMOX*0.1   ,  MAXIMOY*0.2   ,  0  };
screenZone_t zona_351 = {  MAXIMOX*0.1   ,  MAXIMOY*0.8   ,  MAXIMOX*0.1   ,  MAXIMOY*0.2   ,  0  };
screenZone_t zona_352 = {  MAXIMOX*0.2   ,  MAXIMOY*0.8   ,  MAXIMOX*0.1   ,  MAXIMOY*0.2   ,  0  };
screenZone_t zona_353 = {  MAXIMOX*0.3   ,  MAXIMOY*0.8   ,  MAXIMOX*0.1   ,  MAXIMOY*0.2   ,  0  };
screenZone_t zona_354 = {  MAXIMOX*0.4   ,  MAXIMOY*0.8   ,  MAXIMOX*0.1   ,  MAXIMOY*0.2   ,  0  };
screenZone_t zona_355 = {  MAXIMOX*0.5   ,  MAXIMOY*0.8   ,  MAXIMOX*0.1   ,  MAXIMOY*0.2   ,  0  };
screenZone_t zona_356 = {  MAXIMOX*0.6   ,  MAXIMOY*0.8   ,  MAXIMOX*0.1   ,  MAXIMOY*0.2   ,  0  };
screenZone_t zona_357 = {  MAXIMOX*0.7   ,  MAXIMOY*0.8   ,  MAXIMOX*0.1   ,  MAXIMOY*0.2   ,  0  };
screenZone_t zona_358 = {  MAXIMOX*0.8   ,  MAXIMOY*0.8   ,  MAXIMOX*0.1   ,  MAXIMOY*0.2   ,  0  };
screenZone_t zona_359 = {  MAXIMOX*0.9   ,  MAXIMOY*0.8   ,  MAXIMOX*0.1   ,  MAXIMOY*0.2   ,  0  };
//  Display de barras.
screenZone_t zona_330 = {  MAXIMOX*0.0   ,  MAXIMOY*0.4   ,  MAXIMOX*0.1   ,  MAXIMOY*0.2   ,  0  };
screenZone_t zona_331 = {  MAXIMOX*0.1   ,  MAXIMOY*0.4   ,  MAXIMOX*0.1   ,  MAXIMOY*0.2   ,  0  };
screenZone_t zona_332 = {  MAXIMOX*0.2   ,  MAXIMOY*0.4   ,  MAXIMOX*0.1   ,  MAXIMOY*0.2   ,  0  };
screenZone_t zona_333 = {  MAXIMOX*0.3   ,  MAXIMOY*0.4   ,  MAXIMOX*0.1   ,  MAXIMOY*0.2   ,  0  };
screenZone_t zona_334 = {  MAXIMOX*0.4   ,  MAXIMOY*0.4   ,  MAXIMOX*0.1   ,  MAXIMOY*0.2   ,  0  };
screenZone_t zona_335 = {  MAXIMOX*0.5   ,  MAXIMOY*0.4   ,  MAXIMOX*0.1   ,  MAXIMOY*0.2   ,  0  };
screenZone_t zona_336 = {  MAXIMOX*0.6   ,  MAXIMOY*0.4   ,  MAXIMOX*0.1   ,  MAXIMOY*0.2   ,  0  };
screenZone_t zona_337 = {  MAXIMOX*0.7   ,  MAXIMOY*0.4   ,  MAXIMOX*0.1   ,  MAXIMOY*0.2   ,  0  };
screenZone_t zona_338 = {  MAXIMOX*0.8   ,  MAXIMOY*0.4   ,  MAXIMOX*0.1   ,  MAXIMOY*0.2   ,  0  };
screenZone_t zona_339 = {  MAXIMOX*0.9   ,  MAXIMOY*0.4   ,  MAXIMOX*0.1   ,  MAXIMOY*0.2   ,  0  };
//  Menú de carga.
screenZone_t zona_lo0  = {  MAXIMOX*0.2   ,  MAXIMOY*0.2   ,  MAXIMOX*0.6   ,  MAXIMOY*0.2   ,  0  };
screenZone_t zona_lo1  = {  MAXIMOX*0.2   ,  MAXIMOY*0.5   ,  MAXIMOX*0.6   ,  MAXIMOY*0.2   ,  0  };
screenZone_t zona_lo20 = {  MAXIMOX*0.2   ,  MAXIMOY*0.7   ,  MAXIMOX*0.15  ,  MAXIMOY*0.2   ,  0  };
screenZone_t zona_lo21 = {  MAXIMOX*0.35  ,  MAXIMOY*0.7   ,  MAXIMOX*0.15  ,  MAXIMOY*0.2   ,  0  };
screenZone_t zona_lo22 = {  MAXIMOX*0.5   ,  MAXIMOY*0.7   ,  MAXIMOX*0.15  ,  MAXIMOY*0.2   ,  0  };
screenZone_t zona_lo23 = {  MAXIMOX*0.65  ,  MAXIMOY*0.7   ,  MAXIMOX*0.15  ,  MAXIMOY*0.2   ,  0  };
/**-------------------------------------------------------------------------------------------------//
```

```
//                                                                                                                  //
//                                                                                                                                  //
//      @function       __mainLoop__()                                                                              //
//                                                                                                                  //
//      @brief      Esta función es la que se ejecuta en el bucle principal del main. Debe contener                 //
//                  todo el código ejecutable por el loop principal.                                                //
//                                                                                                                  //
//------------------------------------------------------------------------------------------------------------------**/
void    __mainLoop__(   void    )
{
    /** @LOOP:    Primera parte del programa.  */
    alimentaWDT();
    checkTouchPanel();
    if ( __brilloAuto && (SysTick->CTRL & 0x10000))    //  Cada 100 ms si el brillo auto está activado.
    {
        goto_LUT( DATOS->Brillo, BRILLO2CICLO_LDR , none , &Aux8 , none , none);
        modificaPulso(    PWM6 ,    MODO_CICLO ,    Aux8    ,    none    ,    none    ,    none    );
    }
    /** @LOOP:    Máquina de estados LCD.      */
    switch(    ESTADO->CHART   )
    {
        case    PANTALLA_INICIO:
            __pintaInicio__();
            if (zoneNewPressed(  &zona_2))
            {
                ESTADO->CHART = PANTALLA_MEDIDAS1;
                LCD_Clear(Black);
            }
            if (zoneNewPressed(  &zona_3))
            {
                ESTADO->CHART = PANTALLA_MEDIDAS2;
                LCD_Clear(Black);
            }
            if (zoneNewPressed(  &zona_5))
            {
                ESTADO->CHART = PANTALLA_AJUSTES;
                LCD_Clear(Black);
            }
            if (zoneNewPressed(  &zona_4))
            {
                ESTADO->CHART = PANTALLA_VALORES;
                LCD_Clear(Black);
            }
            if (zoneNewPressed(  &zona_7))
            {
                __brilloAuto = 0;
                __brilloFade = 0;
                modificaPulso(    PWM6 ,    MODO_CICLO ,    1    ,    none    ,    none    ,    none    );
                Modo_brillo = 0;
                Modo_energetico =    0;  //  HP.
            }
            if (zoneNewPressed(  &zona_8))
            {
                __brilloAuto = 0;
                __brilloFade = 0;
                modificaPulso(    PWM6 ,    MODO_CICLO ,    20 ,    none    ,    none    ,    none    );
                Modo_brillo = 1;
                Modo_energetico =    0;  //  HP.
            }
            if (zoneNewPressed(  &zona_9))
            {
                __brilloAuto = 0;
                __brilloFade = 0;
                modificaPulso(    PWM6 ,    MODO_CICLO ,    40 ,    none    ,    none    ,    none    );
                Modo_brillo = 2;
                Modo_energetico =    0;  //  HP.
```

```
        }
        if (zoneNewPressed( &zona_10))
        {
            modificaPulso(  PWM6 ,  MODO_CICLO ,  60 ,  none  ,  none  ,  none  );
            __brilloAuto = 0;
            __brilloFade = 0;
            Modo_brillo = 3;
            Modo_energetico =  0; // HP.
        }
        if (zoneNewPressed( &zona_11))
        {
            __brilloAuto = 1;
            __brilloFade = 0;
            Modo_brillo = 4;
            Modo_energetico =  1; // LP.
        }
        if (zoneNewPressed( &zona_12))
        {
            if  (  ACTUALIZADOR->Audiorev  )
            {
                ACTUALIZADOR->Audiorev = 0;
                __configuraTono__();
                activarDac();
            }
        }
        if (zoneNewPressed( &zona_16))
        {
            if  (  ACTUALIZADOR->Audiorev  )
            {
                ACTUALIZADOR->Audiorev = 0;
                lanzaUFONO();
            }
        }


        if (zoneNewPressed( &zona_13))
        {
            Modo_energetico =  0; // HP.
            __brilloAuto    =  0; // No hay brillo auto.
            Modo_brillo     =  3; // Brillo a tope.
            modificaPulso(  PWM6 ,  MODO_CICLO ,  60 ,  none  ,  none  ,  none  );
            __brilloFade    =  0;
        }
        if (zoneNewPressed( &zona_14))
        {
            Modo_energetico =  1; // LP.
            Modo_brillo     =  4; // Brillo auto.
            __brilloAuto    =  1; // Activo el brillo automático.
            __brilloFade    =  0; // No pueda apagarse la pantalla.
        }
        if (zoneNewPressed( &zona_15))
        {
            Modo_energetico =  2; // ULP.
            Modo_brillo     =  4; // Brillo auto.
            __brilloAuto    =  1; // Activo el brillo automático.
            __brilloFade    =  1; // Que pueda apagarse la pantalla.

        }

        break;

    case  PANTALLA_MEDIDAS1:
        __pintaMedidas1__();
        if (zoneNewPressed( &zona_2))
        {
            ESTADO->CHART = PANTALLA_MEDIDAS2;
```

```
            LCD_Clear(Black);
        }
        if (zoneNewPressed(  &zona_3))
        {
            ESTADO->CHART = PANTALLA_INICIO;
            LCD_Clear(Black);
        }
        break;

    case    PANTALLA_MEDIDAS2:
        __pintaMedidas2__();
        if (zoneNewPressed(  &zona_2))
        {
            ESTADO->CHART = PANTALLA_INICIO;
            LCD_Clear(Black);
        }
        if (zoneNewPressed(  &zona_3))
        {
            ESTADO->CHART = PANTALLA_MEDIDAS1;
            LCD_Clear(Black);
        }
        break;

    case    PANTALLA_AJUSTES:
        __pintaAjustes__();
        if (zoneNewPressed(  &zona_2))
        {
            ESTADO->CHART = PANTALLA_INICIO;
            LCD_Clear(Black);
        }
        if (zoneNewPressed(  &zona_3))
        {
            ESTADO->CHART = PANTALLA_INICIO;
            LCD_Clear(Black);
        }
        if (zoneNewPressed(  &zona_27m))
        {
            LPC_RTC->HOUR--;
        }
        if (zoneNewPressed(  &zona_28m))
        {
            LPC_RTC->MIN--;
        }
        if (zoneNewPressed(  &zona_29m))
        {
            LPC_RTC->SEC--;
        }
        if (zoneNewPressed(  &zona_30m))
        {
            LPC_RTC->DOM--;
        }
        if (zoneNewPressed(  &zona_27M))
        {
            LPC_RTC->HOUR++;
        }
        if (zoneNewPressed(  &zona_28M))
        {
            LPC_RTC->MIN++;
        }
        if (zoneNewPressed(  &zona_29M))
        {
            LPC_RTC->SEC++;
        }
        if (zoneNewPressed(  &zona_30M))
        {
            LPC_RTC->DOM++;
```

```
            }
            break;

        case    PANTALLA_LOADING:
            break;
        case PANTALLA_VALORES:
            __pintaValores__();
            if (zoneNewPressed(  &zona_2))
            {
                ESTADO->CHART = PANTALLA_INICIO;
                LCD_Clear(Black);
            }
            if (zoneNewPressed(  &zona_3))
            {
                ESTADO->CHART = PANTALLA_INICIO;
                LCD_Clear(Black);
            }
            if (zoneNewPressed(  &zona_27m))
            {
                MODIFICABLES.Min_servo_t--;
            }
            if (zoneNewPressed(  &zona_28m))
            {
                MODIFICABLES.Max_servo_t--;
            }
            if (zoneNewPressed(  &zona_29m))
            {
                MODIFICABLES.Min_servo_p -= 10;
            }
            if (zoneNewPressed(  &zona_30m))
            {
                MODIFICABLES.Max_servo_p -= 10;
            }
            if (zoneNewPressed(  &zona_27M))
            {
                MODIFICABLES.Min_servo_t++;
            }
            if (zoneNewPressed(  &zona_28M))
            {
                MODIFICABLES.Max_servo_t++;
            }
            if (zoneNewPressed(  &zona_29M))
            {
                MODIFICABLES.Min_servo_p += 10;
            }
            if (zoneNewPressed(  &zona_30M))
            {
                MODIFICABLES.Max_servo_p += 10;
            }
            if (zoneNewPressed(  &zona_31))
            {
                MODIFICABLES.Var_medida = 1 - MODIFICABLES.Var_medida;
            }
        default:
            break;
    };
}
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
//      @function       __configuraLCD__()                                                                     //
//                                                                                                             //
//      @brief      Esta función configura el TFT HY32B conectado al driver ILI9325C                           //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
void    __configuraLCD__(   void    )
```

```
{
    TP_Init();
    LCD_Initializtion();
}
/**-----------------------------------------------------------------------------------------------------------------//
//                                                                                                                  //
//      @function      __pintaInicio__()                                                                            //
//                                                                                                                  //
//      @brief      Esta función pinta la pantalla de inicio.                                                       //
//                                                                                                                  //
//-----------------------------------------------------------------------------------------------------------------**/
void    __pintaInicio__(   void    )
{
    squareButton(&zona_1   ,   (char *)Clock      ,   Yellow ,   Green  );
    squareButton(&zona_2   ,   "->"               ,   Yellow ,   Green  );
    squareButton(&zona_3   ,   "<-"               ,   Yellow ,   Green  );
    squareButton(&zona_4   ,   "Valores"          ,   Yellow ,   Green  );
    squareButton(&zona_5   ,   "Ajustes"          ,   Yellow ,   Green  );
    squareButton(&zona_6   ,   "Nivel de brillo:" ,   Yellow ,   Green  );
    switch (    Modo_brillo    )
    {
        case 0:
            squareButton(&zona_7   ,   "1"            ,   White ,   White  );
            squareButton(&zona_8   ,   "2"            ,   Yellow ,   Green  );
            squareButton(&zona_9   ,   "3"            ,   Yellow ,   Green  );
            squareButton(&zona_10 ,   "4"            ,   Yellow ,   Green  );
            squareButton(&zona_11 ,   "A"            ,   Yellow ,   Green  );
            break;
        case 1:
            squareButton(&zona_7   ,   "1"            ,   Yellow ,   Green  );
            squareButton(&zona_8   ,   "2"            ,   White ,   White  );
            squareButton(&zona_9   ,   "3"            ,   Yellow ,   Green  );
            squareButton(&zona_10 ,   "4"            ,   Yellow ,   Green  );
            squareButton(&zona_11 ,   "A"            ,   Yellow ,   Green  );
            break;
        case 2:
            squareButton(&zona_7   ,   "1"            ,   Yellow ,   Green  );
            squareButton(&zona_8   ,   "2"            ,   Yellow ,   Green  );
            squareButton(&zona_9   ,   "3"            ,   White ,   White  );
            squareButton(&zona_10 ,   "4"            ,   Yellow ,   Green  );
            squareButton(&zona_11 ,   "A"            ,   Yellow ,   Green  );
            break;
        case 3:
            squareButton(&zona_7   ,   "1"            ,   Yellow ,   Green  );
            squareButton(&zona_8   ,   "2"            ,   Yellow ,   Green  );
            squareButton(&zona_9   ,   "3"            ,   Yellow ,   Green  );
            squareButton(&zona_10 ,   "4"            ,   White ,   White  );
            squareButton(&zona_11 ,   "A"            ,   Yellow ,   Green  );
            break;
        case 4:
            squareButton(&zona_7   ,   "1"            ,   Yellow ,   Green  );
            squareButton(&zona_8   ,   "2"            ,   Yellow ,   Green  );
            squareButton(&zona_9   ,   "3"            ,   Yellow ,   Green  );
            squareButton(&zona_10 ,   "4"            ,   Yellow ,   Green  );
            squareButton(&zona_11 ,   "A"            ,   White ,   White  );
            break;
    }
    squareButton(&zona_12 ,   "Play"         ,   Yellow ,   Green  );
    switch (    Modo_energetico )
    {
        case 0:
            squareButton(&zona_13 ,   "HP"           ,   Red       ,   Red       );
            squareButton(&zona_14 ,   "LP"           ,   Yellow ,   Green  );
            squareButton(&zona_15 ,   "ULP"          ,   Yellow ,   Green  );
            break;
```

```
        case 1:
            squareButton( &zona_13 ,   "HP"            ,   Yellow ,   Green  );
            squareButton( &zona_14 ,   "LP"            ,   Blue   ,   Blue   );
            squareButton( &zona_15 ,   "ULP"           ,   Yellow ,   Green  );
            break;
        case 2:
            squareButton( &zona_13 ,   "HP"            ,   Yellow ,   Green  );
            squareButton( &zona_14 ,   "LP"            ,   Yellow ,   Green  );
            squareButton( &zona_15 ,   "ULP"           ,   White  ,   White  );
            break;
    }
    squareButton( &zona_16 ,   "Load"          ,   Yellow ,   Green  );
}
/**-------------------------------------------------------------------------------------------------------//
//                                                                                                       //
//      @function      __pintaAjustes__()                                                                //
//                                                                                                       //
//      @brief      Esta función pinta la pantalla de ajustes.                                           //
//                                                                                                       //
//-------------------------------------------------------------------------------------------------------**/
void    __pintaAjustes__(void    )
{
    squareButton( &zona_1   ,   (char *)Clock      ,   Yellow ,   Green  );
    squareButton( &zona_2   ,   "->"               ,   Yellow ,   Green  );
    squareButton( &zona_3   ,   "<-"               ,   Yellow ,   Green  );
    squareButton( &zona_27 ,   "Horas"             ,   Yellow ,   Green  );
    squareButton( &zona_28 ,   "Minutos"           ,   Yellow ,   Green  );
    squareButton( &zona_29 ,   "Segundos"          ,   Yellow ,   Green  );
    squareButton( &zona_30 ,   "Dias"          ,   Yellow ,   Green  );
    sprintf((char *)buffer ,   "IP:%d.%d.%d.%d"    ,   __IP1B ,   __IP2B ,   __IP3B ,   __IP4B );
    squareButton( &zona_31 ,   (char*)buffer       ,   Yellow ,   Green  );
    squareButton( &zona_27m   ,   "-"              ,   Yellow ,   Green  );
    squareButton( &zona_28m   ,   "-"              ,   Yellow ,   Green  );
    squareButton( &zona_29m   ,   "-"              ,   Yellow ,   Green  );
    squareButton( &zona_30m   ,   "-"              ,   Yellow ,   Green  );
    squareButton( &zona_27M   ,   "+"              ,   Yellow ,   Green  );
    squareButton( &zona_28M   ,   "+"              ,   Yellow ,   Green  );
    squareButton( &zona_29M   ,   "+"              ,   Yellow ,   Green  );
    squareButton( &zona_30M   ,   "+"              ,   Yellow ,   Green  );
}
/**-------------------------------------------------------------------------------------------------------//
//                                                                                                       //
//      @function      __pintaValores__()                                                                //
//                                                                                                       //
//      @brief      Esta función pinta la pantalla de valores.                                           //
//                                                                                                       //
//-------------------------------------------------------------------------------------------------------**/
void    __pintaValores__(void    )
{
    squareButton( &zona_1   ,   (char *)Clock      ,   Yellow ,   Green  );
    squareButton( &zona_2   ,   "->"               ,   Yellow ,   Green  );
    squareButton( &zona_3   ,   "<-"               ,   Yellow ,   Green  );
    squareButton( &zona_27 ,   "Temp.min."         ,   Yellow ,   Green  );
    squareButton( &zona_28 ,   "Temp.max."         ,   Yellow ,   Green  );
    squareButton( &zona_29 ,   "Pres.min."         ,   Yellow ,   Green  );
    squareButton( &zona_30 ,   "Pres.max."         ,   Yellow ,   Green  );
    sprintf((char *)buffer ,   "Pres [%d,%d] Temp [%d,%d]"    ,   (int)MODIFICABLES.Min_servo_p , (int)MODIFICABLES.Max_servo_p,
(int)MODIFICABLES.Min_servo_t, (int)MODIFICABLES.Max_servo_t);
    if ( MODIFICABLES.Var_medida )
    {
        squareButton( &zona_31 ,   (char*)buffer       ,   Green ,   Green  );
    }
    else
    {
```

```
        squareButton( &zona_31 ,   (char*)buffer     ,   Red       ,   Green );
    }
    squareButton( &zona_27m   ,   "-"            ,   Yellow ,   Green );
    squareButton( &zona_28m   ,   "-"            ,   Yellow ,   Green );
    squareButton( &zona_29m   ,   "-"            ,   Yellow ,   Green );
    squareButton( &zona_30m   ,   "-"            ,   Yellow ,   Green );
    squareButton( &zona_27M   ,   "+"            ,   Yellow ,   Green );
    squareButton( &zona_28M   ,   "+"            ,   Yellow ,   Green );
    squareButton( &zona_29M   ,   "+"            ,   Yellow ,   Green );
    squareButton( &zona_30M   ,   "+"            ,   Yellow ,   Green );
}
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                    //
//      @function     __pintaCargandoInicio__()                                                                       //
//                                                                                                                    //
//      @brief      Esta función pinta la pantalla cargando incio.                                                    //
//                                                                                                                    //
//-------------------------------------------------------------------------------------------------------------------**/
void    __pintaCargandoInicio__(    void    )
{
    squareButton( &zona_lo0,   "CARGANDO ..."          ,   Blue   ,   Green );
    squareButton( &zona_lo1,   "Preparando el sistema..."   ,   Blue   ,   Green );
}
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                    //
//      @function     __pintaCargandoSeno__()                                                                         //
//                                                                                                                    //
//      @brief      Esta función pinta la pantalla cargando seno.                                                     //
//                                                                                                                    //
//-------------------------------------------------------------------------------------------------------------------**/
void    __pintaCargandoSeno__(void    )
{
    squareButton( &zona_lo0,   "CARGANDO ..."          ,   Blue   ,   Green );
    squareButton( &zona_lo1,   "Creando muestras..."    ,   Blue   ,   Green );
    squareBox(    &zona_lo20,  White );
}
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                    //
//      @function     __pintaCargandoConexion__()                                                                     //
//                                                                                                                    //
//      @brief      Esta función pinta la pantalla cargando conexión.                                                 //
//                                                                                                                    //
//-------------------------------------------------------------------------------------------------------------------**/
void    __pintaCargandoConexion__(    void    )
{
    squareButton( &zona_lo0,   "CARGANDO ..."          ,   Blue   ,   Green );
    squareButton( &zona_lo1,   "Buscando conexion TCP..."  ,   Blue   ,   Green );
    squareBox(    &zona_lo20,  White );
    squareBox(    &zona_lo21,  White );
}
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                    //
//      @function     __pintaCargandoIniciando__()                                                                    //
//                                                                                                                    //
//      @brief      Esta función pinta la pantalla cargando iniciando.                                                //
//                                                                                                                    //
//-------------------------------------------------------------------------------------------------------------------**/
void    __pintaCargandoIniciando__(    void    )
{
    squareButton( &zona_lo0,   "CARGANDO ..."          ,   Blue   ,   Green );
    squareButton( &zona_lo1,   "Iniciando modulos..."       ,   Blue   ,   Green );
    squareBox(    &zona_lo20,  White );
```

```
    squareBox(    &zona_lo21,   White  );
    squareBox(    &zona_lo22,   White  );
}
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                              //
//      @function      __pintaCargandoDone__()                                                                  //
//                                                                                                              //
//      @brief      Esta función pinta la pantalla cargando hecho.                                              //
//                                                                                                              //
//-------------------------------------------------------------------------------------------------------------**/
void    __pintaCargandoDone__(    void    )
{
    squareButton( &zona_lo0,    "CARGADO"              ,    Blue    ,    Green  );
    squareButton( &zona_lo1,    "100%"                 ,    Blue    ,    Green  );
    squareBox(    &zona_lo20,   White  );
    squareBox(    &zona_lo21,   White  );
    squareBox(    &zona_lo22,   White  );
    squareBox(    &zona_lo22,   White  );
}
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                              //
//      @function      __pintaMedidas1__()                                                                      //
//                                                                                                              //
//      @brief      Esta función pinta la primera pantalla de medidas.                                          //
//                                                                                                              //
//-------------------------------------------------------------------------------------------------------------**/
void    __pintaMedidas1__(  void    )
{
    squareButton( &zona_1   ,    (char *)Clock      ,    Yellow ,    Green  );
    squareButton( &zona_2   ,    "->"               ,    Yellow ,    Green  );
    squareButton( &zona_3   ,    "<-"               ,    Yellow ,    Green  );
    squareButton( &zona_17 ,    "MEDIDAS ACTUALES"    ,    Yellow ,    Green  );
    if  (    ACTUALIZADOR->TempRev    )
    {
        sprintf((char*)buffer,"Altura: %.02f m.", DATOS->Lugar.Altura);
        squareButton( &zona_18 ,    (char *)buffer     ,    Yellow ,    Green  );
    }
    squareButton( &zona_19 ,    "Vel. v.:"         ,    Yellow ,    Green  );
    squareButton( &zona_20 ,    "Humedad:"         ,    Yellow ,    Green  );
    squareButton( &zona_21 ,    "Claridad:"        ,    Yellow ,    Green  );
    squareButton( &zona_22 ,    "Incide UV:"       ,    Yellow ,    Green  );
    if  (    ACTUALIZADOR->Anemometro )
    {
        sprintf((char*)buffer,"%.02f mps",    DATOS->VelViento);
        squareButton( &zona_23 ,    (char *)buffer     ,    Yellow ,    Green  );
        ACTUALIZADOR->Anemometro = 0;
    }

    if  (    ACTUALIZADOR->TempRev    )
    {
        sprintf((char*)buffer,"%.02f %%",     DATOS->Humedad);
        squareButton( &zona_24 ,    (char *)buffer     ,    Yellow ,    Green  );
        ACTUALIZADOR->TempRev = 0;
    }
    sprintf((char*)buffer,"%.02f LUX",DATOS->Brillo);
    squareButton( &zona_25 ,    (char *)buffer     ,    Yellow ,    Green  );
    sprintf((char*)buffer,"%.02f UVs",    DATOS->IndiceUV);
    squareButton( &zona_26 ,    (char *)buffer     ,    Yellow ,    Green  );
}
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                              //
//      @function      __pintaMedidas2__()                                                                      //
//                                                                                                              //
```

```
//      @brief      Esta función pinta la segunda pantalla de medidas.                                  //
//                                                                                                      //
//------------------------------------------------------------------------------------------------------**/
void    __pintaMedidas2__(  void    )
{
    squareButton( &zona_1   ,   (char *)Clock       ,   Yellow ,    Green  );
    squareButton( &zona_2   ,   "->"                ,   Yellow ,    Green  );
    squareButton( &zona_3   ,   "<-"                ,   Yellow ,    Green  );
    squareButton( &zona_32 ,    "Temperatura:"      ,   Yellow ,    Green  );
    if  (   ACTUALIZADOR->TempRev   )
    {
        sprintf((char*)buffer,"%.02f dC", DATOS->Temperatura);
        squareButton( &zona_32n     ,   (char *)buffer      ,   Yellow ,    Green  );
        sprintf((char*)buffer,"%.02f mBar.", DATOS->Presion);
        squareButton( &zona_34n     ,   (char *)buffer      ,   Yellow ,    Green  );
        ACTUALIZADOR->TempRev = 0;
    }                       //  Digo que toca medir.
    switch (    (int)(10*(DATOS->Temperatura - MIN_TEMP)/(MAX_TEMP - MIN_TEMP))     )
    {
        case 0:
            squareBox( &zona_330 , Black);
            squareBox( &zona_331 , Black);
            squareBox( &zona_332 , Black);
            squareBox( &zona_333 , Black);
            squareBox( &zona_334 , Black);
            squareBox( &zona_335 , Black);
            squareBox( &zona_336 , Black);
            squareBox( &zona_337 , Black);
            squareBox( &zona_338 , Black);
            squareBox( &zona_339 , Black);
            break;
        case 1:
            squareBox( &zona_330 , White);
            squareBox( &zona_331 , Black);
            squareBox( &zona_332 , Black);
            squareBox( &zona_333 , Black);
            squareBox( &zona_334 , Black);
            squareBox( &zona_335 , Black);
            squareBox( &zona_336 , Black);
            squareBox( &zona_337 , Black);
            squareBox( &zona_338 , Black);
            squareBox( &zona_339 , Black);
            break;
        case 2:
            squareBox( &zona_330 , White);
            squareBox( &zona_331 , White);
            squareBox( &zona_332 , Black);
            squareBox( &zona_333 , Black);
            squareBox( &zona_334 , Black);
            squareBox( &zona_335 , Black);
            squareBox( &zona_336 , Black);
            squareBox( &zona_337 , Black);
            squareBox( &zona_338 , Black);
            squareBox( &zona_339 , Black);
            break;
        case 3:
            squareBox( &zona_330 , Yellow);
            squareBox( &zona_331 , Yellow);
            squareBox( &zona_332 , Yellow);
            squareBox( &zona_333 , Black);
            squareBox( &zona_334 , Black);
            squareBox( &zona_335 , Black);
            squareBox( &zona_336 , Black);
            squareBox( &zona_337 , Black);
            squareBox( &zona_338 , Black);
            squareBox( &zona_339 , Black);
```

```
            break;
        case 4:
            squareBox( &zona_330 , Yellow);
            squareBox( &zona_331 , Yellow);
            squareBox( &zona_332 , Yellow);
            squareBox( &zona_333 , Yellow);
            squareBox( &zona_334 , Black);
            squareBox( &zona_335 , Black);
            squareBox( &zona_336 , Black);
            squareBox( &zona_337 , Black);
            squareBox( &zona_338 , Black);
            squareBox( &zona_339 , Black);
            break;
        case 5:
            squareBox( &zona_330 , Blue);
            squareBox( &zona_331 , Blue);
            squareBox( &zona_332 , Blue);
            squareBox( &zona_333 , Blue);
            squareBox( &zona_334 , Blue);
            squareBox( &zona_335 , Black);
            squareBox( &zona_336 , Black);
            squareBox( &zona_337 , Black);
            squareBox( &zona_338 , Black);
            squareBox( &zona_339 , Black);
            break;
        case 6:
            squareBox( &zona_330 , Blue);
            squareBox( &zona_331 , Blue);
            squareBox( &zona_332 , Blue);
            squareBox( &zona_333 , Blue);
            squareBox( &zona_334 , Blue);
            squareBox( &zona_335 , Blue);
            squareBox( &zona_336 , Black);
            squareBox( &zona_337 , Black);
            squareBox( &zona_338 , Black);
            squareBox( &zona_339 , Black);
            break;
        case 7:
            squareBox( &zona_330 , Green);
            squareBox( &zona_331 , Green);
            squareBox( &zona_332 , Green);
            squareBox( &zona_333 , Green);
            squareBox( &zona_334 , Green);
            squareBox( &zona_335 , Green);
            squareBox( &zona_336 , Green);
            squareBox( &zona_337 , Black);
            squareBox( &zona_338 , Black);
            squareBox( &zona_339 , Black);
            break;
        case 8:
            squareBox( &zona_330 , Green);
            squareBox( &zona_331 , Green);
            squareBox( &zona_332 , Green);
            squareBox( &zona_333 , Green);
            squareBox( &zona_334 , Green);
            squareBox( &zona_335 , Green);
            squareBox( &zona_336 , Green);
            squareBox( &zona_337 , Green);
            squareBox( &zona_338 , Black);
            squareBox( &zona_339 , Black);
            break;
        case 9:
            squareBox( &zona_330 , Red);
            squareBox( &zona_331 , Red);
            squareBox( &zona_332 , Red);
            squareBox( &zona_333 , Red);
```

```
            squareBox( &zona_334 , Red);
            squareBox( &zona_335 , Red);
            squareBox( &zona_336 , Red);
            squareBox( &zona_337 , Red);
            squareBox( &zona_338 , Red);
            squareBox( &zona_339 , Black);
            break;
        case 10:
            squareBox( &zona_330 , Red);
            squareBox( &zona_331 , Red);
            squareBox( &zona_332 , Red);
            squareBox( &zona_333 , Red);
            squareBox( &zona_334 , Red);
            squareBox( &zona_335 , Red);
            squareBox( &zona_336 , Red);
            squareBox( &zona_337 , Red);
            squareBox( &zona_338 , Red);
            squareBox( &zona_339 , Red);
            break;
        default:
            if ( DATOS->Temperatura > MIN_TEMP)
            {
                squareBox( &zona_330 , Red);
                squareBox( &zona_331 , Red);
                squareBox( &zona_332 , Red);
                squareBox( &zona_333 , Red);
                squareBox( &zona_334 , Red);
                squareBox( &zona_335 , Red);
                squareBox( &zona_336 , Red);
                squareBox( &zona_337 , Red);
                squareBox( &zona_338 , Red);
                squareBox( &zona_339 , Red);
            }
            if ( DATOS->Temperatura < MIN_TEMP)
            {
                squareBox( &zona_330 , Black);
                squareBox( &zona_331 , Black);
                squareBox( &zona_332 , Black);
                squareBox( &zona_333 , Black);
                squareBox( &zona_334 , Black);
                squareBox( &zona_335 , Black);
                squareBox( &zona_336 , Black);
                squareBox( &zona_337 , Black);
                squareBox( &zona_338 , Black);
                squareBox( &zona_339 , Black);
            }
    };

    squareButton( &zona_34 ,    "Presion:"      ,    Yellow ,    Green  );
    switch (    (int)(10*(DATOS->Presion - MIN_PRES)/(MAX_PRES - MIN_PRES)) )
    {
        case 0:
            squareBox( &zona_350 , Black);
            squareBox( &zona_351 , Black);
            squareBox( &zona_352 , Black);
            squareBox( &zona_353 , Black);
            squareBox( &zona_354 , Black);
            squareBox( &zona_355 , Black);
            squareBox( &zona_356 , Black);
            squareBox( &zona_357 , Black);
            squareBox( &zona_358 , Black);
            squareBox( &zona_359 , Black);
            break;
        case 1:
            squareBox( &zona_350 , White);
            squareBox( &zona_351 , Black);
```

```
            squareBox( &zona_352 , Black);
            squareBox( &zona_353 , Black);
            squareBox( &zona_354 , Black);
            squareBox( &zona_355 , Black);
            squareBox( &zona_356 , Black);
            squareBox( &zona_357 , Black);
            squareBox( &zona_358 , Black);
            squareBox( &zona_359 , Black);
            break;
        case 2:
            squareBox( &zona_350 , White);
            squareBox( &zona_351 , White);
            squareBox( &zona_352 , Black);
            squareBox( &zona_353 , Black);
            squareBox( &zona_354 , Black);
            squareBox( &zona_355 , Black);
            squareBox( &zona_356 , Black);
            squareBox( &zona_357 , Black);
            squareBox( &zona_358 , Black);
            squareBox( &zona_359 , Black);
            break;
        case 3:
            squareBox( &zona_350 , Yellow);
            squareBox( &zona_351 , Yellow);
            squareBox( &zona_352 , Yellow);
            squareBox( &zona_353 , Black);
            squareBox( &zona_354 , Black);
            squareBox( &zona_355 , Black);
            squareBox( &zona_356 , Black);
            squareBox( &zona_357 , Black);
            squareBox( &zona_358 , Black);
            squareBox( &zona_359 , Black);
            break;
        case 4:
            squareBox( &zona_350 , Yellow);
            squareBox( &zona_351 , Yellow);
            squareBox( &zona_352 , Yellow);
            squareBox( &zona_353 , Yellow);
            squareBox( &zona_354 , Black);
            squareBox( &zona_355 , Black);
            squareBox( &zona_356 , Black);
            squareBox( &zona_357 , Black);
            squareBox( &zona_358 , Black);
            squareBox( &zona_359 , Black);
            break;
        case 5:
            squareBox( &zona_350 , Blue);
            squareBox( &zona_351 , Blue);
            squareBox( &zona_352 , Blue);
            squareBox( &zona_353 , Blue);
            squareBox( &zona_354 , Blue);
            squareBox( &zona_355 , Black);
            squareBox( &zona_356 , Black);
            squareBox( &zona_357 , Black);
            squareBox( &zona_358 , Black);
            squareBox( &zona_359 , Black);
            break;
        case 6:
            squareBox( &zona_350 , Blue);
            squareBox( &zona_351 , Blue);
            squareBox( &zona_352 , Blue);
            squareBox( &zona_353 , Blue);
            squareBox( &zona_354 , Blue);
            squareBox( &zona_355 , Blue);
            squareBox( &zona_356 , Black);
            squareBox( &zona_357 , Black);
```

```
            squareBox( &zona_358 , Black);
            squareBox( &zona_359 , Black);
            break;
        case 7:
            squareBox( &zona_350 , Green);
            squareBox( &zona_351 , Green);
            squareBox( &zona_352 , Green);
            squareBox( &zona_353 , Green);
            squareBox( &zona_354 , Green);
            squareBox( &zona_355 , Green);
            squareBox( &zona_356 , Green);
            squareBox( &zona_357 , Black);
            squareBox( &zona_358 , Black);
            squareBox( &zona_359 , Black);
            break;
        case 8:
            squareBox( &zona_350 , Green);
            squareBox( &zona_351 , Green);
            squareBox( &zona_352 , Green);
            squareBox( &zona_353 , Green);
            squareBox( &zona_354 , Green);
            squareBox( &zona_355 , Green);
            squareBox( &zona_356 , Green);
            squareBox( &zona_357 , Green);
            squareBox( &zona_358 , Black);
            squareBox( &zona_359 , Black);
            break;
        case 9:
            squareBox( &zona_350 , Red);
            squareBox( &zona_351 , Red);
            squareBox( &zona_352 , Red);
            squareBox( &zona_353 , Red);
            squareBox( &zona_354 , Red);
            squareBox( &zona_355 , Red);
            squareBox( &zona_356 , Red);
            squareBox( &zona_357 , Red);
            squareBox( &zona_358 , Red);
            squareBox( &zona_359 , Black);
            break;
        case 10:
            squareBox( &zona_350 , Red);
            squareBox( &zona_351 , Red);
            squareBox( &zona_352 , Red);
            squareBox( &zona_353 , Red);
            squareBox( &zona_354 , Red);
            squareBox( &zona_355 , Red);
            squareBox( &zona_356 , Red);
            squareBox( &zona_357 , Red);
            squareBox( &zona_358 , Red);
            squareBox( &zona_359 , Red);
            break;
        default:
            if ( DATOS->Presion > MAX_PRES)
            {
                squareBox( &zona_350 , Red);
                squareBox( &zona_351 , Red);
                squareBox( &zona_352 , Red);
                squareBox( &zona_353 , Red);
                squareBox( &zona_354 , Red);
                squareBox( &zona_355 , Red);
                squareBox( &zona_356 , Red);
                squareBox( &zona_357 , Red);
                squareBox( &zona_358 , Red);
                squareBox( &zona_359 , Red);
            }
            if ( DATOS->Presion < MIN_PRES)
```

```
            {
                squareBox( &zona_350 , Black);
                squareBox( &zona_351 , Black);
                squareBox( &zona_352 , Black);
                squareBox( &zona_353 , Black);
                squareBox( &zona_354 , Black);
                squareBox( &zona_355 , Black);
                squareBox( &zona_356 , Black);
                squareBox( &zona_357 , Black);
                squareBox( &zona_358 , Black);
                squareBox( &zona_359 , Black);
            }
    }
}
/**-----------------------------------------------------------------------------------------------------//
//                                                                                                     //
//      @function       squareButton()                                                                 //
//                                                                                                     //
//      @brief      Dibuja un botón cuadrado, con texto y colores.                                     //
//                                                                                                     //
//-----------------------------------------------------------------------------------------------------**/
void squareButton(screenZone_t* zone, char * text, uint16_t textColor, uint16_t lineColor)
{
    LCD_DrawLine( zone->x, zone->y, zone->x + zone->size_x - 1, zone->y, lineColor);
    LCD_DrawLine( zone->x, zone->y, zone->x, zone->y + zone->size_y - 1, lineColor);
    LCD_DrawLine( zone->x, zone->y + zone->size_y - 1, zone->x + zone->size_x - 1, zone->y + zone->size_y - 1, lineColor);
    LCD_DrawLine( zone->x + zone->size_x - 1, zone->y, zone->x + zone->size_x - 1, zone->y + zone->size_y - 1, lineColor);
    GUI_Text(zone->x + zone->size_x/2 - (strlen(text)/2)*8, zone->y + zone->size_y/2 - 8, (uint8_t*) text, textColor, Black);
}
/**-----------------------------------------------------------------------------------------------------//
//                                                                                                     //
//      @function       squareBox()                                                                    //
//                                                                                                     //
//      @brief      Dibuja un cuadrado de un color.                                                    //
//                                                                                                     //
//-----------------------------------------------------------------------------------------------------**/
void squareBox(screenZone_t* zone, uint16_t color)
{
    int i;
    for (i = 0; i < (zone->size_x - 4) ; i++)
    {
        LCD_DrawLine( zone->x + i + 2, zone->y + 2, zone->x + i + 2, zone->y + zone->size_y - 2, color);
    }
}
/**-----------------------------------------------------------------------------------------------------//
//                                                                                                     //
//      @function       checkTouchPanel()                                                              //
//                                                                                                     //
//      @brief      Verifica se si ha tocado la pantalla.                                              //
//                                                                                                     //
//-----------------------------------------------------------------------------------------------------**/
void checkTouchPanel(void)
{
    Coordinate* coord;

    coord = Read_Ads7846();

    if (coord > 0) {
       getDisplayPoint(&display, coord, &matrix );
       pressedTouchPanel = 1;
    }
    else
       pressedTouchPanel = 0;
```

```
}
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
//                                                                                                             //
//      @function       zoneNewPressed()                                                                       //
//                                                                                                             //
//      @brief      Verifica si se ha presionado una cierta zona de la pantalla.                               //
//                                                                                                             //
//              @zone Zona a comprobar.                                                                        //
//                                                                                                             //
//              @return     0 - Si no se ha producido un toque.                                                //
//                          1 - Si se ha producido un toque.                                                   //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
int8_t zoneNewPressed(screenZone_t * zone)
{
    if (pressedTouchPanel == 1) {

        if ((display.x > zone->x) && (display.x < zone->x + zone->size_x) &&
                (display.y > zone->y) && (display.y < zone->y + zone->size_y))
        {
            if (zone->pressed == 0)
            {
                zone->pressed = 1;
                return 1;
            }
            return 0;
        }
     /**    @MOD:    Esto lo he añadido yo  */
      if (contadorLUZ  >=  (FREQ_OVERFLOW_SYSTICK * MODIFICABLES.TiempoBrillo))   //  Si se ha activado el apagar pantalla...
      {
        modificaPulso     (    PWM6,     MODO_CICLO ,    60  ,    none    ,    none           ,    none           );  //  La enciendo como
si hubiese habido un reset.
        Modo_brillo = 3;
        if   (    Modo_energetico > 1 )
        {
            __brilloAuto = 1;
            Modo_brillo = 4;
            if   (    Modo_energetico == 2 )
            {
                __brilloFade = 1;
            }
        }

    }
      contadorLUZ = 0;          //  Reseteo el contador de apagar la pantalla.
    }

    zone->pressed = 0;
    return 0;
}
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
//                                                                                                             //
//      @end        ENDFILE.                                                                                   //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
```

Configura.h

```
/**-------------------------------------------------------------------------------------------------------------------//
//      @filename     configura.h                                                                                     //
//      @version      0.00                                                                                            //
//      @author       Alberto Palomo Alonso                                                                           //
//                                                                                                                    //
//      @brief      Cabecera para el código de configuración.c                                                        //
//                                                                                                                    //
//      @category     Principal.                                                                                      //
//                                                                                                                    //
//      @map          @include                                                                                        //
//                  @private                                                                                          //
//                  @funcdef                                                                                          //
//                  @end                                                                                              //
//                                                                                                                    //
//                                                                                                                    //
//-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                    //
                                                                                                                      //
//      @include      Estos son los archivos utilizados con el código de configuración.                               //
//                                                                                                                    //
//-------------------------------------------------------------------------------------------------------------------**/
// PWM
#ifndef PWM
#define     PWM
#include    "PWM.h"
#endif
#ifndef GLCD
#define     GLDC
#include    "GLCD.h"
#include    "TouchPanel.h"
#include    "menu.h"
#endif
#ifndef STATECHART
#define     STATECHART
#include    "Statechart.h"
#endif
#ifndef RTC
#define     RTC
#include    "RTC.h"
#endif
#ifndef SYSTEMSYMBOLS
#define     SYSTEMSYMBOLS
#include    "Systemsymbols.h"
#endif
#ifndef WDT
#define     WDT
#include    "WDT.h"
#endif
#ifndef HTTP_SOURCE
#define     HTTP_SOURCE
#include "HTTP_SOURCE.h"
#endif
#ifndef TIMERS
#define     TIMERS
#include    "Timers.h"
#endif
#ifndef ANEMOMETRO
#define     ANEMOMETRO
```

```
#include    "Anemometro.h"
#endif
#ifndef DAC
#define     DAC
#include    "DAC.h"
#endif
#ifndef UVA30A
#define     UVA30A
#include    "UVA30A.h"
#endif
#ifndef UFONO
#define     UFONO
#include    "uFono.h"
#endif
#ifndef LUT
#define     LUT
#include    "LUT.h"
#endif
#ifndef DMA
#define     DMA
#include    "DMA.h"
#endif
#ifndef I2C
#define     I2C
#include    "I2C.h"
#endif
#ifndef UART0
#define     UART0
#include    "UART0.h"
#endif
#ifndef UART3
#define     UART3
#include    "UART3.h"
#endif
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
//                                                                                                             //
//      @private        Estos son los símbolos correspondientes a la configuración.                            //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
#define     ACTIVOS_TODOS2      0x003F
#define     ACTIVOS_1_2         0x0001
#define     ACTIVOS_2_2         0x0002
#define     ACTIVOS_1_1         0x0100
#define     ACTIVOS_2_1         0x0200
#define     ACTIVOS_6_2         0x0020
#define     ACTIVOS_6_1         0x2000

#define     MAXIMO_PRESION          1500
#define     MINIMO_PRESION          500
#define     MAXIMO_TEMPERATURA      50
#define     MINIMO_TEMPERATURA -10
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
//                                                                                                             //
//      @funcdef        Estas son las funciones correspondientes a la configuración.                           //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
void __configuraPrograma__ (    void    );
void __iniciaVariables__    (    void    );
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
//                                                                                                             //
//      @end        ENDFILE.                                                                                   //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
```

Configura.c

```
/**-------------------------------------------------------------------------------------------------------------------//
//      @filename     configura.c                                                                                     //
//      @version      0.00                                                                                            //
//      @author       Alberto Palomo Alonso                                                                           //
//                                                                                                                    //
//      @brief      Código que configura y llama a las funciones de configuración para hacer un Setup del programa.//
//                                                                                                                    //
//      @category     Principal.                                                                                      //
//                                                                                                                    //
//      @map          @include                                                                                        //
//                  @funcion                                                                                          //
//                  @end                                                                                              //
//                                                                                                                    //
//                                                                                                                    //
//-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                    //
                                                                                                                      //
//      @include      Estos son los archivos utilizados en el código de configuración.                                //
//                                                                                                                    //
//-------------------------------------------------------------------------------------------------------------------**/
#ifndef     CONFIGURA
#define     CONFIGURA
#include "configura.h"
#endif
extern State_t      *   ESTADO;
extern misDatos_t   *   DATOS;
extern modificables_t     MODIFICABLES;
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                    //
                                                                                                                      //
//      @funcion      Esta función configura el programa entero.                                                      //
//                                                                                                                    //
//      @ref            __configuraPWM__        ->  PWM.h                                                           //
//                  modificaPulso           ->  PWM.h                                                               //
//                  __configuraLCD__      ->  Statechart.h                                                          //
//                                                                                                                    //
//-------------------------------------------------------------------------------------------------------------------**/
void __configuraPrograma__(     void    )
{
    __configuraLCD__      ();
    LCD_Clear(Black);
    __pintaCargandoInicio__ ();
    __iniciaVariables__         ();
    LCD_Clear(Black);
    __pintaCargandoSeno__  ();
    //  Añadir generación de variables de alto costo computacional.
    LCD_Clear(Black);
    __pintaCargandoConexion__();
    __configuraWEB__          ();
    LCD_Clear(Black);
    __pintaCargandoIniciando__();
    __configuraSysTick__ ();
    __configuraTimer0__       ();
    __configuraLDR__      ();
    __configuraUVA30A__         ();
```

```
    __configuraUFONO__     ();
    __configuraRTC__       ();
    __configuraPWM__   (   Fpwm  ,   ACTIVOS_2_1 | ACTIVOS_6_1    );
    modificaPulso      (   PWM2,    MODO_SERVO    ,   none  ,   90  ,   MINIMO_SERVO  ,   MAXIMO_SERVO  );
    modificaPulso      (   PWM6,    MODO_CICLO ,   50  ,   none  ,   none          ,   none          );
    __configuraWDT__       ();
    __configuraDAC__       ();
    __configuraDMA__       ();
    __configuraOW__    ();
    __configuraAnemometro__  ();
    __configuraUART0__     ();
//  __configuraUART3__     ();
    __configuraI2C__       ();
#ifndef DEBUG
//  TouchPanel_Calibrate();
#endif
    LCD_Clear(Black);
    __pintaCargandoDone__ ();
    LCD_Clear(Black);
    ESTADO->CHART = PANTALLA_INICIO;
}
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
//                                                                                                                   //
//      @funcion      __iniciaVariables__()                                                                          //
//                                                                                                                   //
//      @brief     Esta función inicia las variables del sistema para que tengan un momento inicial.                 //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
void __iniciaVariables__()
{
    ESTADO->CHART = PANTALLA_LOADING;
    DATOS->Temperatura          = 0;
    DATOS->Humedad          = 0;
    DATOS->Presion       = 0;
    DATOS->VelViento        = 0;
    DATOS->IndiceUV         = 0;
    DATOS->Lugar.Altura     = 0;
    DATOS->Lugar.Longitud   = 0;
    DATOS->Lugar.Latitud    = 0;

    MODIFICABLES.Max_servo_t =   (float)MAXIMO_TEMPERATURA;
    MODIFICABLES.Min_servo_t =   (float)MINIMO_TEMPERATURA;
    MODIFICABLES.Max_servo_p=    (float)MAXIMO_PRESION;
    MODIFICABLES.Min_servo_p =   (float)MINIMO_PRESION;
    MODIFICABLES.TiempoBrillo    =   10;
    MODIFICABLES.Var_medida      =   0;      //  0 la temperatura, 1 la presión.
}
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
//                                                                                                                   //
//      @end      ENDFILE.                                                                                           //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
```

I2C.h

```
/**-------------------------------------------------------------------------------------------------------------------//
```

```
//      @filename     I2C.h                                                                   //
//      @version      0.00                                                                        //
//      @author       Alberto Palomo Alonso                                                           //
//                                                                                        //
//      @brief      Esta es la cabecera que recoge la comunicación por I2C.                           //
//                                                                                        //
//      @category     Opcional.                                                               //
//                                                                                        //
//      @map          @include                                                                //
//                  @function                                                             //
//                  @end                                                                  //
//                                                                                        //
//------------------------------------------------------------------------------------------------//
//                                                                                        //
                                                                                                    //
//      @include      Includes pertenecientes a la comunicación por I2C.                        //
//                                                                                        //
//------------------------------------------------------------------------------------------------**/
#ifndef LPC17XX
#define     LPC17XX
#include    "LPC17XX.H"
#endif
#ifndef SYSTEMSYMBOLS
#define     SYSTEMSYMBOLS
#include    "Systemsymbols.h"
#endif
#ifndef MATH
#define     MATH
#include    <math.h>
#endif

typedef struct
{
    short ac1;
    short ac2;
    short ac3;
    unsigned short    ac4;
    unsigned short    ac5;
    unsigned short    ac6;
    short b1;
    short b2;
    short mb;
    short mc;
    short md;
}BMP_t;
/**------------------------------------------------------------------------------------------------//
//                                                                                        //
                                                                                                    //
//      @private      Estos son los símbolos correspondientes al protocolo I2C.                   //
//                                                                                        //
//------------------------------------------------------------------------------------------------**/
#define     SDA                 0       //  Pin 0
#define     SCL                 1       //  Pin 1
#define     DELAYTOTAL          100         //  Ajustar.

#define     BMP_ADD             119         //  Cambia en función del sensor.
#define     REGISTRO_PRESION        0xF6        //  Acceso al registro de presión.
#define     REGISTRO_TEMPERATURA    0x

#define     READ                1
#define     WRITE           0
#define     SACK                0
#define     NACK                1
#define     PRESION_BMP         10
#define     TEMPERATURA_BMP 11
```

```
#define    AC1              0xAA
#define    AC2              0xAC
#define    AC3              0xAE
#define    AC4              0xB0
#define    AC5              0xB2
#define    AC6              0xB4
#define    B1           0xB6
#define    B2           0xB8
#define    MB           0xBA
#define    MC           0xBC
#define    MD           0xBE
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
//      @funcdef      Estas son las funciones correspondientes al protocolo I2C.                                //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
void    __configuraI2C__ (   void    )       ;
void    irRegistro      (   uint8_tREG)   ;
void    pedirDato           (   void    )       ;
void pedirDatoReg           (   uint8_tREG)   ;
void __calibraBMP           (   void    )       ;
int32_t calculaTemperatura( void    )       ;
int32_t calculaPresion      (   void    )       ;
void    medirBMP            (   void    )       ;
uint16_t obtenerDato(   uint8_tREG    );
void    mandaDato               (   uint8_tREG    ,   uint8_tDATA);

void I2Cdelay       (   void    )           ;
void I2CSendAddr (   uint8_taddr   ,   uint8_trw );
void    I2CSendByte   (   uint8_tbyte)   ;
uint8_t I2CGetByte    (   uint8_tACK)       ;
void I2CSendStop (   void    )           ;
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
//      @end      ENDFILE.                                                                                     //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
```

I2C.c

```
/**-------------------------------------------------------------------------------------------------------------//
//      @filename     I2C.c                                                                                    //
//      @version      0.00                                                                                     //
//      @author       Alberto Palomo Alonso                                                                    //
//                                                                                                             //
```

```
//      @brief      Este es el programa que recoge la comunicación por I2C.                    //
//                                                                                             //
//      @category     Opcional.                                                                //
//                                                                                             //
//      @map          @include                                                                 //
//                  @variables                                                                 //
//                  @function                                                                  //
//                  @end                                                                       //
//                                                                                             //
//-------------------------------------------------------------------------------------------//
//                                                                                             //
                                                                                                //
//      @include      Includes pertenecientes a la comunicación por I2C.                       //
//                                                                                             //
//-------------------------------------------------------------------------------------------**/
#ifndef I2C
#define     I2C
#include    "I2C.h"
#endif
/**-------------------------------------------------------------------------------------------//
//                                                                                             //
                                                                                                //
//      @variables          Variables del fichero.                                             //
//                                                                                             //
//-------------------------------------------------------------------------------------------**/
//  Variables globales y externas.
BMP_t COEF              ;
extern      misDatos_t    *    DATOS;
uint8_tTemperaturaConBmp =    0;
uint8_tLecturaBMP0  ;
uint8_tLecturaBMP1  ;
uint16_t    LecturaBMP    ;
float temperatura;
float presion;
/**-------------------------------------------------------------------------------------------//
//                                                                                             //
                                                                                                //
//      @function       __configuraI2C__()                                                     //
//                                                                                             //
//      @brief      Esta función es la que configura el protocolo I2C.                         //
//                                                                                             //
//-------------------------------------------------------------------------------------------**/
void    __configuraI2C__ (    void    )
{
    __calibraBMP();
}
void __calibraBMP()
{
    I2CSendAddr( BMP_ADD ,    WRITE );
    I2CSendByte( AC1    );
    I2CSendAddr( BMP_ADD ,    READ   );
    COEF.ac1  =   I2CGetByte(   SACK   )   << 8;
    COEF.ac1  |=  I2CGetByte(   NACK   );
    I2CSendStop();

    I2CSendAddr( BMP_ADD ,    WRITE );
    I2CSendByte( AC2    );
    I2CSendAddr( BMP_ADD ,    READ   );
    COEF.ac2  =   I2CGetByte(   SACK   )   << 8;
    COEF.ac2  |=  I2CGetByte(   NACK   );
    I2CSendStop();

    I2CSendAddr( BMP_ADD ,    WRITE );
    I2CSendByte( AC3    );
    I2CSendAddr( BMP_ADD ,    READ   );
    COEF.ac3  =   I2CGetByte(   SACK   )   << 8;
```

```
    COEF.ac3  |= I2CGetByte(   NACK  );
    I2CSendStop();

    I2CSendAddr( BMP_ADD ,   WRITE );
    I2CSendByte( AC4    );
    I2CSendAddr( BMP_ADD ,   READ  );
    COEF.ac4  =  I2CGetByte(   SACK  )   << 8;
    COEF.ac4  |= I2CGetByte(   NACK  );
    I2CSendStop();

    I2CSendAddr( BMP_ADD ,   WRITE );
    I2CSendByte( AC5    );
    I2CSendAddr( BMP_ADD ,   READ  );
    COEF.ac5  =  I2CGetByte(   SACK  )   << 8;
    COEF.ac5  |= I2CGetByte(   NACK  );
    I2CSendStop();

        I2CSendAddr( BMP_ADD ,    WRITE );
    I2CSendByte( AC6    );
    I2CSendAddr( BMP_ADD ,   READ  );
    COEF.ac6  =  I2CGetByte(   SACK  )   << 8;
    COEF.ac6  |= I2CGetByte(   NACK  );
    I2CSendStop();

    I2CSendAddr( BMP_ADD ,   WRITE );
    I2CSendByte( B1 );
    I2CSendAddr( BMP_ADD ,   READ  );
    COEF.b1   =  I2CGetByte(   SACK  )   << 8;
    COEF.b2   |= I2CGetByte(   NACK  );
    I2CSendStop();

    I2CSendAddr( BMP_ADD ,   WRITE );
    I2CSendByte( B2 );
    I2CSendAddr( BMP_ADD ,   READ  );
    COEF.b2   =  I2CGetByte(   SACK  )   << 8;
    COEF.b2   |= I2CGetByte(   NACK  );
    I2CSendStop();

    I2CSendAddr( BMP_ADD ,   WRITE );
    I2CSendByte( MB);
    I2CSendAddr( BMP_ADD ,   READ  );
    COEF.mb   =  I2CGetByte(   SACK  )   << 8;
    COEF.mb   |= I2CGetByte(   NACK  );
    I2CSendStop();

    I2CSendAddr( BMP_ADD ,   WRITE );
    I2CSendByte( MC);
    I2CSendAddr( BMP_ADD ,   READ  );
    COEF.mc   =  I2CGetByte(   SACK  )   << 8;
    COEF.mc   |= I2CGetByte(   NACK  );
    I2CSendStop();

    I2CSendAddr( BMP_ADD ,   WRITE );
    I2CSendByte( MD);
    I2CSendAddr( BMP_ADD ,   READ  );
    COEF.md   =  I2CGetByte(   SACK  )   << 8;
    COEF.md   |= I2CGetByte(   NACK  );
    I2CSendStop();
}
/**-----------------------------------------------------------------------------------------------------------------------//
//                                                                                                                      //
//      @function       procesarDato()                                                                                  //
//                                                                                                                      //
//      @brief      Procesa el dato guardado en la variable LecturaBMP.                                                 //
//                                                                                                                      //
```

```
//-------------------------------------------------------------------------------------------------------**/
void    procesarDato (    uint8_tTipo    )
{
    DATOS->Presion       =   presion*0.01;
    DATOS->Temperatura    =   temperatura;
    DATOS->Lugar.Altura =       (float)44330*( 1 - pow((presion / 101325 ),(1/5.255)));
}
/**-------------------------------------------------------------------------------------------------------//
//                                                                                                       //
//                                                                                                       //
//      @function      pedirDatoReg()                                                                    //
//                                                                                                       //
//      @brief      Pide un dato al sensor I2C de 16 bits especificando el registro.                     //
//                                                                                                       //
//-------------------------------------------------------------------------------------------------------**/
void    pedirDatoReg (    uint8_tREG    )
{
    irRegistro( REG    );
    pedirDato ();
    procesarDato( PRESION_BMP     );
}
/**-------------------------------------------------------------------------------------------------------//
//                                                                                                       //
//                                                                                                       //
//      @function      pedirDato()                                                                       //
//                                                                                                       //
//      @brief      Pide un dato al sensor I2C de 16 bits.                                               //
//                                                                                                       //
//-------------------------------------------------------------------------------------------------------**/
void    pedirDato (    void    )
{
    I2CSendAddr( BMP_ADD ,    READ    )        ;        // Mandar la dirección en modo lectura.
    LecturaBMP0  =   I2CGetByte (   SACK    )    ;        // Leo el primer byte.
    LecturaBMP1  =   I2CGetByte (   NACK ) ;        // Leo el segundo byte.
    I2CSendStop()                          ;        // Mando el fin de la comunicación.
    LecturaBMP    =   (LecturaBMP0 << 8) | LecturaBMP1; // Todo al buffer de 16 bits.
}
/**-------------------------------------------------------------------------------------------------------//
//                                                                                                       //
//                                                                                                       //
//      @function      irRegistro()                                                                      //
//                                                                                                       //
//      @brief      Accede al registro REG de la memoria EPROM.                                          //
//                                                                                                       //
//-------------------------------------------------------------------------------------------------------**/
void irRegistro(    uint8_tREG    )   // Acceso al registro REG del sensor.
{
    I2CSendAddr( BMP_ADD ,    WRITE);    // Selecciono BMP.
    I2CSendByte(       REG          );        // Selecciono el registro de presión.
}
/**-------------------------------------------------------------------------------------------------------//
//                                                                                                       //
//                                                                                                       //
//      @function      bmp180_get_pressure()                                                             //
//                                                                                                       //
//      @brief      No hablo el mismo idioma que el creador del driver, pero parece que devuelve la presión.  //
//                                                                                                       //
//      @ref            Extraido de https://github.com/BoschSensortec/BMP180_driver; referido por la datasheet.  //
//                                                                                                       //
//-------------------------------------------------------------------------------------------------------**/
uint16_t obtenerDato(    uint8_tREG    )
{
    uint16_t    RETVAL;
    I2CSendAddr( BMP_ADD ,    WRITE );
    I2CSendByte( REG    );
    I2CSendAddr( BMP_ADD ,    READ   );
```

```
    RETVAL  =  I2CGetByte(  SACK  )  << 8;
    RETVAL  |= I2CGetByte(  NACK  );
    I2CSendStop();
    return RETVAL;
}
void  mandaDato  (  uint8_tREG  ,  uint8_tDATA)
{
    I2CSendAddr( BMP_ADD ,  WRITE );
    I2CSendByte( REG  );
    I2CSendByte( DATA  );
    I2CSendStop();
}

void medirBMP()
{
    int i;
    long  UT, UP , X1 , X2 , X3 , B3 , B5 , B6 , T , p;
    unsigned long B4, B7;
    mandaDato  (  0xF4  ,  0x2E  );
    //Espera 4.7ms.
    for ( i = 0; i < 1000; i++)
    {
        I2Cdelay();
    }
    //Espera activa corta!
    UT = obtenerDato (  0xF6);
    mandaDato  (  0xF4  ,  0x34  );
    //Esperar 4.7ms.
    for ( i = 0; i < 1000; i++)
    {
        I2Cdelay();
    }
    //Espera activa corta!
    UP = obtenerDato (  0xF6);
    X1 = (UT - COEF.ac6) * COEF.ac5 / 32768;
    X2 = COEF.mc * 2048 / (X1 + COEF.md);
    B5 = X1 + X2;
    T  = ((B5 + 8) >> 4);

    B6 = B5 - 4000;
    X1 = (COEF.b2 * ((B6 * B6) >> 12)) >> 11;
    X2 = (COEF.ac2 * B6) >> 11;
    X3 = X1 + X2;
    B3 = ((COEF.ac1 * 4 + X3) + 2) / 4;
    X1 = (COEF.ac3 * B6) >> 13;
    X2 = (COEF.b1 * ((B6 * B6) >> 12)) >> 16;
    X3 = (X1 + X2 + 2) >> 2;
    B4 = COEF.ac4 * (unsigned long)(X3 + 32768) >> 15;
    B7 = ((unsigned long)UP - B3)*(50000);

    if  (B7 < 0x80000000)
    {
        p = (B7*2) / B4;
    }
    else
    {
        p = (B7 / B4) * 2;
    }

    X1 = (p >> 8)*(p >> 8);
    X1 = (X1 * 3038 >> 16);
    X2 = (-7357 * p) >> 16;
    p = p + ((X1 + X2 + 3791) >> 4);
//  temperatura  = (float)(28.0/107.0)*((float)T)/10;
//  presion      = (float)(936.0/1150.0)*(float)p;
    temperatura  = ((float)T)/10;
```

```
    presion         = (float)p;
    procesarDato(0);
}
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
//      @end        ENDFILE.                                                                                   //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
```

Anemometro.h

```
/**-------------------------------------------------------------------------------------------------------------//
//      @filename       Anemometro.h                                                                           //
//      @version        0.00                                                                                   //
//      @author         Alberto Palomo Alonso                                                                  //
//                                                                                                             //
//      @brief      Esta es la cabecera donde se declara todo lo utilizado en el anemómetro.                   //
//                                                                                                             //
//      @category       Opcional.                                                                              //
//                                                                                                             //
//      @map            @include                                                                               //
//                  @private                                                                                   //
//                  @funcdef                                                                                   //
//                  @end                                                                                       //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
                                                                                                               //
//      @include        Includes pertenecientes al módulo del PWM.                                             //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
#ifndef LPC17XX
#define     LPC17XX
#include    "LPC17XX.H"
#endif
#ifndef SYSTEMSYMBOLS
#define     SYSTEMSYMBOLS
#include "Systemsymbols.h"
#endif
#ifndef TIMERS
#define     TIMERS
#include    "Timers.h"
#endif
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
                                                                                                               //
//      @private        Estos son los símbolos correspondientes al anemómetro.                                 //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
#define     PULSOS_VUELTA       2
#define     DIAMETRO_ANEMOMETRO   14  // En centímetros.
#define     PULSOS_CENTIMETRO     (float)PULSOS_VUELTA/((float)PI*(float)DIAMETRO_ANEMOMETRO)
#define     CTCR_MASCARA        0x0//   Dejar al TC contando.
#define     CCR_MASCARA_EN          0x5//   Generar interrupción.
#define     CCR_MASCARA_DIS   0x4//   Desactivar interrupción.
#define     WARMUP_CICLOS     4   //   Ciclos de calentamiento.
```

```
#define     CAPTURE_FUNCION   0x3//  Capture 1.0.
#define     PULL_UP           0x0//  El pull.
#define     CAP10_IR          0x10   //  El IR de CAP1.0
#define     MR1_IR            0x00000002   //  El IR del MR1.
#define     MR2_IR            0x00000005   //  El IR del MR2 QUE SE JUNTA CON LA DEL 0.
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                    //
//       @funcdef       Estas son las funciones correspondientes al anemómetro.                                       //
//                                                                                                                    //
//-------------------------------------------------------------------------------------------------------------------**/
void __configuraAnemometro__(    void    );
void    mideAnemometro           (    void    );
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                    //
//       @end       ENDFILE.                                                                                         //
//                                                                                                                    //
//-------------------------------------------------------------------------------------------------------------------**/
```

Anemometro.c

```
/**-------------------------------------------------------------------------------------------------------------------//
//       @filename     Anemometro.c                                                                                   //
//       @version      2.00                                                                                           //
//       @author       Alberto Palomo Alonso                                                                          //
//                                                                                                                    //
//       @brief        Este es el programa donde se encuentran las funciones correspondientes al                      //
//                     anemómetro de la estación.                                                                     //
//                                                                                                                    //
//                                                                                                                    //
//       @category     Medida.                                                                                        //
//                                                                                                                    //
//       @map          @include                                                                                       //
//                     @variables                                                                                     //
//                     @function                                                                                      //
//                     @end                                                                                           //
//                                                                                                                    //
//-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                    //
//       @include      Includes pertenecientes al módulo del anemómetro.                                              //
//                                                                                                                    //
//-------------------------------------------------------------------------------------------------------------------**/
#ifndef ANEMOMETRO
#define     ANEMOMETRO
#include    "Anemometro.h"
#endif
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                    //
//       @variables          Variables del fichero.                                                                   //
//                                                                                                                    //
//-------------------------------------------------------------------------------------------------------------------**/
uint8_tCAPcont        =   2*PULSOS_VUELTA;
uint8_tSLAYERcont     =   0;
uint32_t    CLKbuff[]       =   {0 , 0};
extern misDatos_t     *     DATOS;
```

```
extern actualizador_t *   ACTUALIZADOR;
float   aux_viento    =   0;
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
//                                                                                                                   //
//      @function      __configuraAnemometro__()                                                                     //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
void __configuraAnemometro__()
{
    LPC_PINCON->PINMODE3        &=     ~(PULL_UP << (2*2));       //  PULL_UP
    LPC_PINCON->PINMODE3    |=  PULL_UP << (2*2);              //  PULL_UP
    LPC_PINCON->PINSEL3           &=     ~((CAPTURE_FUNCION) << (2*2));//  CAPTURE1.0
    LPC_PINCON->PINSEL3       |=  (CAPTURE_FUNCION) << (2*2);   //  CAPTURE1.0
    LPC_TIM1->CTCR          |=  CTCR_MASCARA;               //  Flanco de subida.
    LPC_TIM1->CCR           |=  CCR_MASCARA_EN;             //  Inicio con interrupción.
    LPC_SC->PCONP           |=  TIMER1_BIT;                 //  Activo el módulo del timer 1.
    LPC_TIM1->PR        =   0;                          //  Sin prescaler.
    LPC_TIM1->TCR           |=  RESET_TIMER_TCR;            //  Reseteo el contador.
    LPC_TIM1->TCR           &=     ~RESET_TIMER_TCR;            //  Reseteo el contador.
    LPC_TIM1->TCR           |=  0x1;                        //      Que cuente.
    LPC_PINCON->PINSEL3       |=  0x3<<  4;                 //  Entrada como CAP1.0.
    NVIC_EnableIRQ(  TIMER1_IRQn );
}
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
//                                                                                                                   //
//      @HANDLER        mideAnemometro()                                                                             //
//                                                                                                                   //
//      @brief      Es la medidas del anemómetro, se suicida al acabar y es reanimada por el reanimador cada 5s.  //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
void mideAnemometro()
{
                /** @WARNING:   Esto me parece un poco sucio, pero es la única manera de que no se generen
                            interrupciones espúrias utilizando únicamente recursos sorftware. Idealmente
                            no debería usar delays y mucho menos en interrupciones. Si no uso esto se generan
                            varias interrupciones por flanco debido al ruido. Esto se arregla con un condensador
                            a masa en la entrada, pero es perder recursos hardware y he decidido perderlos
                            mediante software. */
                int i;
                for(i = 0; i < 30000; i++) {}
                /** @TODO Poner condensadores entre Vin y masa del capture 1.0 para evitar doble int en flancos ascendentes. */


    LPC_TIM1->IR = 1 << 4;                      //  Desactivo la interrupción.
    CAPcont++;                              //  Incremento el contador de pulsos.
    if  (   CAPcont   >=  PULSOS_VUELTA   )   /** @WARNING: Se generan __UN__ interrupciones por pulso y 2*PULSOS_VUELTA por
vuelta. */
    {
        CLKbuff[1] = CLKbuff[0];                //  Almaceno el valor anterior.
        CLKbuff[0] = LPC_TIM1->CR0;                 //  Cargo el valor actual.
        CAPcont = 0;                            //  Reseteo el contador de pulsos.
        aux_viento    =   Ftick*(float)PI*(float)DIAMETRO_ANEMOMETRO/((float)100*(float)((uint32_t)CLKbuff[0] -
(uint32_t)CLKbuff[1]));   // Metros / segundo.
        SLAYERcont++;                           //  Hay warmup, aumento el slayer.
        if (  SLAYERcont == WARMUP_CICLOS )
        {
            LPC_TIM1->CCR   |=  ~CCR_MASCARA_DIS;//  Slay capture. OJO: QUE HAY QUE REVIVIRLO.
            SLAYERcont = 0;                     //  Reseteo el slayer.
            DATOS->VelViento = aux_viento;      //  Guardo el valor calentado.
            ACTUALIZADOR->AnemometroRev = 0;        //  Digo que he medido al timer.
            ACTUALIZADOR->Anemometro = 1;           //  Digo que he medido al statechart.
        }
        else
        {
```

```
            LPC_TIM1->CCR   |= CCR_MASCARA_EN;  //  Si no, está activado.
        }
    }
}
/**--------------------------------------------------------------------------------------------------------------//
//                                                                                                              //
//                                                                                                              //
//      @end        ENDFILE.                                                                                    //
//                                                                                                              //
//--------------------------------------------------------------------------------------------------------------**/
```

LDR.h

```
/**--------------------------------------------------------------------------------------------------------------//
//      @filename       LDR.h                                                                                   //
//      @version                                                                                                //
//      @author         Alberto Palomo Alonso                                                                   //
//                                                                                                              //
//      @brief      Cabecera para el código de LDR.c                                                            //
//                                                                                                              //
//      @category       Periférico.                                                                             //
//                                                                                                              //
//      @map            @include                                                                                //
//                  @private                                                                                    //
//                  @funcdef                                                                                    //
//                  @end                                                                                        //
//                                                                                                              //
//                                                                                                              //
//--------------------------------------------------------------------------------------------------------------//
//                                                                                                              //
//                                                                                                              //
//      @include        Estos son los archivos utilizados con el código de configuración.                       //
//                                                                                                              //
//--------------------------------------------------------------------------------------------------------------**/
#ifndef LPC17xx
#define     LPC17XX
#include    "LPC17XX.H"
#endif
#ifndef SYSTEMSYMBOLS
#define     SYSTEMSYMBOLS
#include    "Systemsymbols.h"
#endif
#ifndef LUT
#define     LUT
#include    "LUT.h"
#endif
#ifndef DMA
#define     DMA
#include    "DMA.h"
#endif
#ifndef UFONO
#define     UFONO
#include    "uFono.h"
#endif
/**--------------------------------------------------------------------------------------------------------------//
//                                                                                                              //
//                                                                                                              //
//      @private        Estos son los símbolos correspondientes a la configuración.                             //
```

```
//                                                                                                                  //
//------------------------------------------------------------------------------------------------------------------**/
#define     PCONP_ADC_ON  (1 << 12)
#define     PINSEL_ADC01      (1 << 16)
#define     PINMODE_ADC01 (3 << 16)
#define     BRUST_PIN         (1 << 16)
#define     SEL_CANAL1    (1 << 1)
#define     SEL_CANAL_GLOBAL  (1 << 8)
#define     ADC_POWER         (1 << 21)
#define     ADC_START         (0x6 << 24)
#define     CLK_DIV_MAX (0xFF << 8)

#define     RESISTENCIA_PULL     70.00
#define     LDRRESISTENCIA_MAX     100
#define     LDRRESISTENCIA_MIN     1
#define     BRILLO_MAX        100
#define     BRILLO_MIN        1

#define     VREF              1.2
#define     VINDICE           10
/**----------------------------------------------------------------------------------------------------------------//
//                                                                                                                  //
//                                                                                                                        //
//      @funcdef        Estas son las funciones correspondientes a la configuración.                        //
//                                                                                                                  //
//------------------------------------------------------------------------------------------------------------------**/
void __configuraLDR__     (     void      );
void ponAudioDMA          (     void      );
/**----------------------------------------------------------------------------------------------------------------//
//                                                                                                                  //
//                                                                                                                     //
//      @end        ENDFILE.                                                                                  //
//                                                                                                                  //
//------------------------------------------------------------------------------------------------------------------**/
```

LDR.c

```
/**----------------------------------------------------------------------------------------------------------------//
//      @filename       LDR.c                                                                                 //
//      @version        0.00                                                                                      //
//      @author         Alberto Palomo Alonso                                                                        //
//                                                                                                                  //
//      @brief      Código fuente que contiene las funciones para LDR (ADC).                                  //
```

```
//                                                                              //
//      @category       Periférico.                                                //
//                                                                              //
//      @map            @include                                                     //
//                      @VARIABLES                                                 //
//                      @funcion                                                   //
//                      @end                                                       //
//                                                                              //
//                                                                              //
//------------------------------------------------------------------------------------//
//                                                                              //
                                                                                                //
//      @include        Estos son los archivos utilizados en el código de LDR.                          //
//                                                                              //
//------------------------------------------------------------------------------------**/
#ifndef LDR
#define     LDR
#include    "LDR.h"
#endif
/**---------------------------------------------------------------------------------//
//                                                                              //
                                                                                                //
//      @variables          Variables del fichero.                                      //
//                                                                              //
//------------------------------------------------------------------------------------**/
extern misDatos_t     *   DATOS;
float   BUFFER_BRILLO = 0;
float   BUFFER_UVA        = 0;
extern      actualizador_t *    ACTUALIZADOR;
extern      uint8_tYaPuedesMedir;
uint32_t    contador;
uint8_tAUDIO[MUESTRAS_AUDIO];
/**---------------------------------------------------------------------------------//
//                                                                              //
                                                                                                //
//      @funcion  __configuraLDR__()                                                    //
//                                                                              //
//      @brief Esta función configura el ADC y el LDR                                  //
//                                                                              //
//------------------------------------------------------------------------------------**/
void    __configuraLDR__()
{
    LPC_SC->PCONP               |= PCONP_ADC_ON;        //  Enciendo el ADClock.
    LPC_PINCON->PINSEL1          |= PINSEL_ADC01;        //  AD0.1
    LPC_PINCON->PINMODE1      &= ~PINMODE_ADC01;    //  AD0.1
    LPC_ADC->ADCR               |= BRUST_PIN            //  Modo ráfaga.
                            |   SEL_CANAL1        //  AD0.1 activado.
                            |   ADC_POWER            //  Empiezo ENCENDIENDO el ADC.
                            |   CLK_DIV_MAX;         //  Clkdiv hace que Fadc = Fclk/256, inferior al umbral de 13MHz. (Ojo: clkdiv = 0 implica
que no funcione en placa).
    LPC_ADC->ADINTEN            |= SEL_CANAL1;       //  Genera interrupción el canal 1.    (Debería ser el penúltimo)
    ACTUALIZADOR->LDRrev =     1;                   //  Inicia para activar.
    LPC_ADC->ADINTEN            &= ~(SEL_CANAL_GLOBAL);//    Apago la interrupción global.
    NVIC_EnableIRQ(  ADC_IRQn );
}
/**---------------------------------------------------------------------------------//
//                                                                              //
                                                                                                //
//      @HANDLER      ADC_IRQHandler()                                                   //
//                                                                              //
//      @brief Esta función gestiona la interrupción del ADC.                          //
//                                                                              //
//------------------------------------------------------------------------------------**/
void    ADC_IRQHandler()
{
    switch( YaPuedesMedir )
```

```
    {
        case 1:
            LPC_ADC->ADCR           &= ~BRUST_PIN;                                          // Mato el BURST.
            BUFFER_BRILLO           =  (float)((LPC_ADC->ADDR1&(0xFFF0)) >> 4);             // Empieza a partir del
bit 4.
            BUFFER_BRILLO           /= (float)0xFFF;  //*rel                                // Relacción de código.
(Código/Código máximo)
            BUFFER_BRILLO           =  RESISTENCIA_PULL*(BUFFER_BRILLO)/(1.00 - BUFFER_BRILLO);     // Leo el ADC.
(Ressitencia del LDR en kOhms)
            goto_LUT( BUFFER_BRILLO   ,   BRILLO_LDR_NOLUT, (float *)&DATOS->Brillo   ,   none   ,   none   ,   none   ); //
    Traduzco resistencias a LUX.
            BUFFER_UVA          =  (float)((LPC_ADC->ADDR2&(0xFFF0)) >> 4);             // Empieza a partir del bit 4.
            DATOS->IndiceUV     =  (float)VINDICE*VREF*BUFFER_UVA/(float)(0xFFF);       // Traducción del código
al índice.
            ACTUALIZADOR->LDRrev =  1;                                                  // Digo que el LDR ha sido leido.
            //ACTUALIZADOR->LDR       =  1;                                             // Señal al LCD para que
muestre por pantalla.
            break;
        case 0:
            AUDIO[contador] =  (uint8_t)((0xFF) & LPC_ADC->ADDR0 >> (4+4));     // El ADC es de 12 bits y las muestras de 8 bits, por
lo que hay que reducir los 4 LSB.
            if (contador++ >= MUESTRAS_AUDIO - 1)
            {
                contador            =  0;       // Reseteo el contador.
                LPC_TIM1->MCR       =  0;       // No interrumpe el MR0.
                ACTUALIZADOR->Audiorev = 1;     // Señalizo el fin del audio.
                recuperaContexto();             // Recupero el contexto del ADC.
            }
            break;
    }
}
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
//                                                                                                             //
//      @end        ENDFILE.                                                                                   //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
```

uFono.h

```
/**-------------------------------------------------------------------------------------------------------------//
//      @filename       uFono.h                                                                                //
//      @version        0.00                                                                                   //
//      @author         Alberto Palomo Alonso                                                                  //
//                                                                                                             //
```

```
//      @brief      Cabecera para configurar el audio del micrófono.                                    //
//                                                                                                      //
//      @category     Opcional.                                                                         //
//                                                                                                      //
//      @map          @include                                                                          //
//                  @funcdef                                                                          //
//                  @end                                                                              //
//                                                                                                      //
//                                                                                                      //
//--------------------------------------------------------------------------------------------------------//
//                                                                                                      //
                                                                                                        //
//      @include      Estos son los archivos utilizados para el audio del micrófono.                    //
//                                                                                                      //
//--------------------------------------------------------------------------------------------------------**/
#ifndef SYSTEMSYMBOLS
#define     SYSTEMSYMBOLS
#include    "Systemsymbols.h"
#endif
#ifndef LPC17XX
#define     LPC17XX
#include    "LPC17XX.H"
#endif
#ifndef TIMERS
#define     TIMERS
#include    "Timers.h"
#endif
/**------------------------------------------------------------------------------------------------------//
//                                                                                                      //
                                                                                                        //
//      @private      Estos son los símbolos correspondientes al audio del micrófono.                   //
//                                                                                                      //
//--------------------------------------------------------------------------------------------------------**/
#define     FUNC_ADC        0x1
#define     PIN_UFONO       (23-16)
#define     CANAL_ADC_UF    0x1
/**------------------------------------------------------------------------------------------------------//
//                                                                                                      //
                                                                                                        //
//      @funcdef      Estas son las funciones correspondientes al audio del micrófono.                  //
//                                                                                                      //
//--------------------------------------------------------------------------------------------------------**/
void __configuraUFONO__    (    void    );  //  Configuramos el micrófono.
void    lanzaUFONO         (    void    );  //  Lanzamos la carga de audio.
void recuperaContexto      (    void    );  //  Recupera el ADC para lanzar las muestras.
/**------------------------------------------------------------------------------------------------------//
//                                                                                                      //
                                                                                                        //
//      @end      ENDFILE.                                                                            //
//                                                                                                      //
//--------------------------------------------------------------------------------------------------------**/
```

uFono.c

```
/**------------------------------------------------------------------------------------------------------//
//      @filename     uFono.c                                                                           //
//      @version      1.00                                                                              //
```

```
//      @author       Alberto Palomo Alonso                                                    //
//                                                                                             //
//      @brief      Código que contiene todo lo relaccionado con el micrófono.                 //
//                                                                                             //
//      @category     Opcional.                                                                //
//                                                                                             //
//      @map          @include                                                                 //
//                  @variables                                                                 //
//                  @funcion                                                                   //
//                  @end                                                                       //
//                                                                                             //
//                                                                                             //
//-------------------------------------------------------------------------------------------//
//                                                                                             //
                                                                                                //
//      @include      Estos son los archivos utilizados para el auido del micrófono.           //
//                                                                                             //
//-------------------------------------------------------------------------------------------**/
#ifndef UFONO
#define    UFONO
#include   "uFono.h"
#endif
/**-------------------------------------------------------------------------------------------//
//                                                                                             //
                                                                                                //
//      @variables        Variables del fichero.                                               //
//                                                                                             //
//-------------------------------------------------------------------------------------------**/
extern uint8_t Timer2_MODO;
uint8_t YaPuedesMedir = 1;
uint32_t   ADC_ConfigBuffer;
uint32_t   ADC_IntenBuffer;
/**-------------------------------------------------------------------------------------------//
//                                                                                             //
                                                                                                //
//      @funcion  __configuraUFONO__()                                                         //
//                                                                                             //
//      @ref        Configura todo lo necesario para la lectura de audio.                      //
//                                                                                             //
//      @WARNING    Utiliza variables de bloqueo.                                              //
//                                                                                             //
//-------------------------------------------------------------------------------------------**/
void __configuraUFONO__()
{
//  LPC_PINCON->PINSEL1 |= ~(0x3 << (2*PIN_UFONO));
    LPC_PINCON->PINSEL1 |= (FUNC_ADC << (2*PIN_UFONO));

    //  NEW:
    LPC_TIM1->MR0   =   Fclk*TsAudio - 1;  //  Cada MR0 se genera una interrupción de leer el audio.
    LPC_TIM1->TCR   =   0x2;               //  Reset al contador.
    LPC_TIM1->TCR   =   0x1;               //  Activo contador.
    LPC_TIM1->MCR   =       0x0;           //  MR0 que NO genera la interrupción.
    NVIC_EnableIRQ( TIMER1_IRQn );  //  Activo interrupción.
}
/**-------------------------------------------------------------------------------------------//
//                                                                                             //
                                                                                                //
//      @funcion  lanzaUFONO()                                                                 //
//                                                                                             //
//      @ref        Lanza la lectura de audio.                                                 //
//                                                                                             //
//-------------------------------------------------------------------------------------------**/
void    lanzaUFONO()
{
    //  Preparo el contexto.
    YaPuedesMedir       =   0;             //  Bloqueo el ADC para el audio.
```

```
    LPC_GPIO3->FIOSET  =  (   1   <<  LECTURA_AUDIO);    // Señalizo lectura de audio.
    Timer2_MODO        =  MODO_ENTRADA;            // Indico que el audio está siendo grabado.
    ADC_ConfigBuffer   =  LPC_ADC->ADCR;          // Guardo el contexto de la configuración.
    ADC_IntenBuffer  =  LPC_ADC->ADINTEN;      // Guardo el contexto de la configuración de interrupciones.
    // Configurar ADC.
    LPC_ADC->ADINTEN  =  1;                  // No quiero interrupciones por conversión excepto en AD0.0.
    LPC_ADC->ADCR      &= ~0xFF;             // Borro el sel entero, sólo voy a usar un canal.
    LPC_ADC->ADCR      |=  CANAL_ADC_UF;          // Canal para el audio.
    //
    LPC_ADC->ADCR      &= ~(0xFF << 8);          // Borro el clkdiv.
    LPC_ADC->ADCR      |=  (0x1 << 8);         // CLKDiv a 1.
    // Empiezo con la conversión.
    LPC_ADC->ADCR  &= ~BRUST_PIN;              // QUITO EL MODO BURST.          // Reanimo el timer.
    LPC_ADC->ADCR  &= ~(0x7 << 24);            // Configuro el start.
    LPC_ADC->ADCR  |=  ADC_START;              // Configuro el start.
    // Activo el timer.
    LPC_TIM1->MCR  =  0x2;                     // Reset on match.
    LPC_TIM1->TCR  =  0x2;                     // Que resetee.
    LPC_TIM1->TCR  =  0x1;                     // Que cuente.
    LPC_TIM1->EMR  =  0x31;                  // Activo el Match0 en modo toogle.
}
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
                                                                                                                //
//      @funcion  recuperaContexto()                                                                           //
//                                                                                                             //
//      @ref      Desbloquea los recursos utilizados para la lectura de audio.                                 //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
void recuperaContexto()
{
    // Recupero el contexto.
    if ( Timer2_MODO == MODO_ENTRADA)          // Si toca recuperar...
    {
        LPC_ADC->ADCR      =  ADC_ConfigBuffer;    // Cargo el contexto de la configuración.
        LPC_ADC->ADINTEN   =  ADC_IntenBuffer; // Cargo el contexto de la configuración de interrupciones.
        LPC_ADC->ADCR      &= ~(0x7 << 24);       // Borro el START del ADC.
        LPC_ADC->ADCR      |= (0xFF << 8);        // CLKDIV max.
        YaPuedesMedir      =  1;               // Desbloqueo el ADC.
        LPC_GPIO3->FIOCLR  =  (   1   <<  LECTURA_AUDIO);    // Señalizo fin de lectura.
        Timer2_MODO        =  MODO_SALIDA;       // Default modo salida.
    }
}
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
                                                                                                                //
//      @end      ENDFILE.                                                                                     //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
```

UVA30A.h

```
/**-------------------------------------------------------------------------------------------------------------//
```

```
//      @filename     UVA30A.h                                                    //
//      @version      0.00                                                        //
//      @author       Alberto Palomo Alonso                                           //
//                                                                          //
//      @brief      Cabecera para el código de UVA30A.c                             //
//                                                                          //
//      @category     Periférico.                                           //
//                                                                          //
//      @map          @include                                                  //
//                  @private                                                  //
//                  @funcdef                                                  //
//                  @end                                                      //
//                                                                          //
//                                                                          //
//--------------------------------------------------------------------------------------------------//
//                                                                          //
                                                                                                    //
//      @include      Estos son los archivos utilizados con el código de configuración.                 //
//                                                                          //
//--------------------------------------------------------------------------------------------------**/
#ifndef LPC17xx
#define     LPC17XX
#include    "LPC17XX.H"
#endif
#ifndef LDR
#define     LDR
#include    "LDR.h"
#endif
/**------------------------------------------------------------------------------------------------//
//                                                                          //
                                                                                                    //
//      @private      Estos son los símbolos correspondientes a la configuración.                       //
//                                                                          //
//--------------------------------------------------------------------------------------------------**/
#define     LDR_primero   1
#define     PINSEL_ADC02      (1 << 18)
#define     PINMODE_ADC02 (3 << 18)
#define     SEL_CANAL2    (1 << 2)
/**------------------------------------------------------------------------------------------------//
//                                                                          //
                                                                                                    //
//      @funcdef      Estas son las funciones correspondientes a la configuración.                      //
//                                                                          //
//--------------------------------------------------------------------------------------------------**/
void    __configuraUVA30A__(    void    );

/**------------------------------------------------------------------------------------------------//
//                                                                          //
                                                                                                        //
//      @end      ENDFILE.                                                    //
//                                                                          //
//--------------------------------------------------------------------------------------------------**/
```

UVA30A.c

```
/**------------------------------------------------------------------------------------------------//
//      @filename     UVA30A.c                                                    //
//      @version      0.00                                                        //
//      @author       Alberto Palomo Alonso                                           //
```

```
//                                                                                                      //
//      @brief      Código fuente que contiene las funciones para UVA30A (ADC).                         //
//                                                                                                      //
//      @category     Periférico.                                                                       //
//                                                                                                      //
//      @map          @include                                                                          //
//                  @funcion                                                                            //
//                  @end                                                                                //
//                                                                                                      //
//                                                                                                      //
//------------------------------------------------------------------------------------------------------//
//                                                                                                      //
                                                                                                        //
//      @include      Estos son los archivos utilizados en el código de configuración del UVA.          //
//                                                                                                      //
//------------------------------------------------------------------------------------------------------**/
#ifndef UVA30A
#define    UVA30A
#include   "UVA30A.h"
#endif
/**--------------------------------------------------------------------------------------------------------//
//                                                                                                      //
                                                                                                        //
//      @funcion  __configuraUVA30A__()                                                                 //
//                                                                                                      //
//      @ref        Configura todo lo necesario para que el UVA30A lea el índice UV.                    //
//                                                                                                      //
//------------------------------------------------------------------------------------------------------**/
void    __configuraUVA30A__()
{
    if ( !LDR_primero )
    {
        __configuraLDR__();
    }
    LPC_PINCON->PINSEL1         |=  PINSEL_ADC02;        //  AD0.2


    LPC_PINCON->PINMODE1     &= ~PINMODE_ADC02;   //  AD0.2
    LPC_ADC->ADCR            |=  SEL_CANAL2;      //  AD0.2
}
/**--------------------------------------------------------------------------------------------------------//
//                                                                                                      //
                                                                                                        //
//      @end      ENDFILE.                                                                              //
//                                                                                                      //
//------------------------------------------------------------------------------------------------------**/
```

OneWire.h

```
/**--------------------------------------------------------------------------------------------------------//
```

```
//      @filename     OneWire.h                                                    //
//      @version      0.00                                                         //
//      @author       Alberto Palomo Alonso                                            //
//                                                                                 //
//      @brief      Cabecera para configurar el protocolo OneWire.                       //
//                                                                                 //
//      @category     Opcional.                                                    //
//                                                                                 //
//      @map          @include                                                     //
//                  @funcdef                                                     //
//                  @end                                                         //
//                                                                                 //
//                                                                                 //
//-------------------------------------------------------------------------------------------------------------//
//                                                                                 //
                                                                                                                //
//      @include      Estos son los archivos utilizados para el protocolo OneWire.                  //
//                                                                                 //
//-------------------------------------------------------------------------------------------------------------**/
#ifndef SYSTEMSYMBOLS
#define   SYSTEMSYMBOLS
#include  "Systemsymbols.h"
#endif
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                 //
                                                                                                                //
//      @private      Estos son los símbolos correspondientes al protocolo OneWire.                  //
//                                                                                 //
//-------------------------------------------------------------------------------------------------------------**/
#define   SIZEOF_TRAMA          40    //  Tamaño del array como buffer de 8 bits.
#define   SIZEOF_SLOT         8     //  Los últimos 8 bits son el checksum.

#define   OW_RESET_BAJO         0.018 //  Tiempo a nivel bajo de la señal de start. (MCU -> Sensor)
                /** @CHANGE: 0.04*/
#define   OW_RESET_ALTO         0.02      //  Tiempo a nivel anto de la señal de start. (MCU -> Sensor)
#define   OW_RESPUESTA_BAJO       0.08        //  Señal respuesta del sensor a nivel bajo.
#define   OW_RESPUESTA_ALTO       0.08        //  Señal respuesta del sensor a nivel alto.
                /** @CHANGE: 0.078*/
#define   OW_MANDADO0         0.076 //  Duración del tiempo de un bit = 0.
#define   OW_MANDADO1         0.120 //  Duración del tiempo de un bit = 1.

#define   PERMISO_DIFERENCIAL     -0.005 //  Error que le permito tener al sensor.
#define   PERMISO_PROPORCIONAL  0.005 //  Error que le permito tener al sensor.

#define   OW_PULL_UP          0x0   //  Función de pull up.
#define   OW_PULL_DOWN          0x3   //  Función de pull down.
#define   OW_CAPTURE_FUNC     0x3   //  Función capture.
#define   OW_CTCR_MASCARA     0x0   //  Dejar TC contando.
#define   OW_CCR_MASCARA_EN     0x30      //  Activo por flanco de bajada.
#define   OW_CCR_MASCARA_DIS    0x20      //  Desactivo por flanco.
#define   CAP11_IR              0x00000020// El IR de CAP1.1
#define   MR0_IR                0x00000001// El IR de MR0.

#define   OWDEEPSLEEP             0
#define   OWINICIO              1
#define   OWESPERANDO             2
#define   OWESPERANDO_BIT       3
#define   OWCHECKSUM              4
#define   OWGENERA                5
#define   OWESPERANDO_SEQ       6

#define   LIMITE_FALLOS           5
#define   BITOW             19

#define   US_AHORA        (LPC_TIM3->TC)
#define   PIN_OWp     19
```

```
#define     PIN_OW        (1  <<  19)
#define     CONFIG_OUT  (LPC_GPIO1->FIODIR |= PIN_OW)
#define     CONFIG_IN         (LPC_GPIO1->FIODIR &= ~(PIN_OW))
#define     CLEAR_PIN         (LPC_GPIO1->FIOCLR = PIN_OW)
#define     SET_PIN         (LPC_GPIO1->FIOSET = PIN_OW)
#define     ENTRADA       ((LPC_GPIO1->FIOPIN >> PIN_OWp) & 1)
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
//       @funcdef       Estas son las funciones correspondientes al protocolo OneWire.                               //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
void    __configuraOW__(    void    );  //  Configuración del protocolo OneWire.
void    mideTemperatura(    void    );  //  Código de medición de temperatura.
void    activaMedidaOW (    void    );  //  Lanza el activador del one wire.

void    StateChartOneWire   (   uint32_t   DeltaCap   );
void OWSetPin         (   uint8_tNivel)  ;
void OWConfiguraEntrada    (   void    )       ;
void OWConfiguraSalida  (   void    )       ;
void    ErrorRx             (   void    )       ;
void ErrorTx            (   void    )       ;
void InvalidChecksum (   void    )       ;
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
//       @end       ENDFILE.                                                                                         //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
```

OneWirev2.c

```
/**-------------------------------------------------------------------------------------------------------------------//
//       @filename     OneWire.c                                                                                     //
```

```
//      @version        4.01                                                    //
//      @author         Alberto Palomo Alonso                                   //
//                                                                              //
//      @brief      Código que configura el protocolo monohilo del sensor de temperatura y humedad.     //
//                                                                              //
//      @category       Opcional.                                               //
//                                                                              //
//      @map            @include                                                //
//                  @variables
//                  @funcion                                                    //
//                  @end                                                        //
//                                                                              //
//                                                                              //
//------------------------------------------------------------------------------//
//                                                                              //
                                                                                //
//      @include        Estos son los archivos utilizados para el protocolo OneWire.        //
//                                                                              //
//------------------------------------------------------------------------------**/
#ifndef ONEWIRE
#define     ONEWIRE
#include    "OneWire.h"
#endif
/**------------------------------------------------------------------------------//
//                                                                              //
                                                                                //
//      @variables          Variables del fichero.                              //
//                                                                              //
//------------------------------------------------------------------------------**/
uint32_t reiniciaCuenta         (   void    );
void     inicializaT3           (   void    );
void _delayUS                   (   uint16_t usegundos  );
uint8_t      compruebaRespuesta (   void    );
uint8_tleerByte                 (   void    );
//  Externo:
extern misDatos_t           *   DATOS;
uint32_t    TRAZA [100];
uint32_t    HOLD      [100];
int p;
uint8_t Checksum;
/**------------------------------------------------------------------------------//
//                                                                              //
                                                                                //
//      @function       __configuraOW__()                                       //
//                                                                              //
//      @brief      Configura los pines y los recursos utilizados para el protocolo OneWire.    //
//                                                                              //
//------------------------------------------------------------------------------**/
void __configuraOW__()
{
    inicializaT3();
    reiniciaCuenta();
}
/**------------------------------------------------------------------------------//
//                                                                              //
                                                                                //
//      @function       mideTemperatura()                                       //
//                                                                              //
//      @brief      Configura los pines y los recursos utilizados para el protocolo OneWire.    //
//                                                                              //
//------------------------------------------------------------------------------**/
void    mideTemperatura(    void    )
{
    int i;
    uint8_t Check[4] = {0,0,0,0};
    uint32_t Rx                 =   0;
```

```
    uint8_t Checksum_Recibido  =   0;
    p = 0;
    Checksum = 0;
    /** @state:     Estado en el que mandamos la señal de petición.     */
    CONFIG_OUT;
    CLEAR_PIN;
    _delayUS(18000);
    CONFIG_IN;
    /** @state:     Esperamos la respuesta.                             */
    if (  compruebaRespuesta()    )
    {
        //  ERROR A AL ESPERAR LA RESPUESTA DEL SENSOR...
        return;
    }
    /** @state:     Leemos los 5 bytes...                               */
    for(i = 0; i < 4; i++)
    {
        Rx |= (leerByte() << (3-i)*8);
    }
    Checksum_Recibido = leerByte();
    /** @state:     Procesamos Rx.          */
    Check[0]   =   ((Rx & (0xFF000000)) >> 6*4);
    Check[1]   =   ((Rx & (0x00FF0000)) >> 4*4);
    Check[2]   =   ((Rx & (0x0000FF00)) >> 2*4);
    Check[3]   =   ((Rx & (0x000000FF)) >> 0*4);
    Checksum = Check[0] + Check[1] + Check[2] + Check[3] ;

    if( Checksum == Checksum_Recibido )
    {
        DATOS->Humedad     = (float)(((Rx >> 16) & 0xFFFF)/10.0);
        DATOS->Temperatura     = (float)(((Rx >> 00) & 0xFFFF)/10.0);
    }
}
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
//      @function       reiniciaCuenta()                                                                             //
//                                                                                                                   //
//      @brief      Reinicia el contador del timer 3.                                                                //
//                                                                                                                   //
//      @ret            Retorna el valor de la cuenta antes de reiniciarla.                                          //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
uint32_t reiniciaCuenta()
{
    uint32_t retval =  US_AHORA;
    LPC_TIM3->TCR   =   2;  //  Reinicia timer.
    LPC_TIM3->TCR   =   1;  //  Activa cuentas.
    return retval;
}
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
//      @function       inicializaT3()                                                                               //
//                                                                                                                   //
//      @brief      Configura el timer 3 para utilizarlo para el OneWire.                                            //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
void inicializaT3()
{
    LPC_SC->PCONP   |=  (1  <<  23);    //  Activo el timer 3.
    LPC_TIM3->CTCR  =   0;          //  Contar por prescaler.
    LPC_TIM3->PR=   24;         //  Cuentas cada 1us.
}
/**-------------------------------------------------------------------------------------------------------------------//
```

```
//                                                                                              //
                                                                                                                                    //
//      @function       _delayUS()                                                                  //
//                                                                                              //
//      @brief      Espera activa de [ usegundos ] microsegundos.                                           //
//                                                                                              //
//----------------------------------------------------------------------------------------------**/
void    _delayUS( uint16_t    usegundos )
{
    reiniciaCuenta();
    while   (US_AHORA < usegundos) {}
}
/**---------------------------------------------------------------------------------------------//
//                                                                                              //
                                                                                                                                    //
//      @function       compruebaRespuesta()                                                                //
//                                                                                              //
//      @brief      Comprueba si el sensor ha respondido apropiadamente.                                            //
//                                                                                              //
//      @ret            Retorna 0 si todo ha ido bien, 1 si no.                                             //
//                                                                                              //
//----------------------------------------------------------------------------------------------**/
uint8_tcompruebaRespuesta()
{
    uint32_t    Tiempo = 0;
    /** @state:     Esperamos que el sensor responda con un pull down...   */
    reiniciaCuenta();
    while   (   ENTRADA   && US_AHORA < 45)           //  Si la entrada está a nivel alto y no han pasado 45 us...
    {
                                                //  Mantente en espera.
    }
    Tiempo = reiniciaCuenta();                          //  Me quedo con cuentos us han pasado.
    TRAZA[p++] = Tiempo;    //!!!!!
    if ( Tiempo < 5 || Tiempo > 45)                     //  Si el márgen de pull down del sensor no es el adecuado.
    {
        return 1;                                   //  Exit error code.
    }
    /** @state:     Esperamos la respuesta del sensor...                */
    reiniciaCuenta();
    while   (   ENTRADA == 0 && US_AHORA < 100)//  Tiempo nivel bajo...
    {

    }
    Tiempo = reiniciaCuenta();
    TRAZA[p++] = Tiempo;    //!!!!!
    if ( Tiempo < 70 || Tiempo > 90)                    //  Si el tiempo de pull down no es adecuado...
    {
        return 1;                                   //  Exit error code.
    }
    reiniciaCuenta();
    while   (   ENTRADA       && US_AHORA < 100)//  Tiempo nivel alto...
    {

    }
    Tiempo = reiniciaCuenta();
    TRAZA[p++] = Tiempo;    //!!!!!
    if ( Tiempo < 70 || Tiempo > 90)                    //  Si el tiempo de pull down no es adecuado...
    {
        return 1;                                   //  Exit error code.
    }
    return 0;                                       //  Respuesta correcta.
}
/**---------------------------------------------------------------------------------------------//
//                                                                                              //
                                                                                                                                    //
//      @function       leerByte()                                                                  //
```

```
//                                                                                                      //
//      @brief      Lee 8 bits en ráfaga del sensor.                                                    //
//                                                                                                      //
//------------------------------------------------------------------------------------------------------**/
uint8_t leerByte(   void    )
{
    int i;
    uint8_t Rx=0;
    uint32_t Tiempo = 0;
    for (i = 0; i < 8; i++)
    {
        reiniciaCuenta();
        while ( ENTRADA == 0 && US_AHORA < 100)     //  Mientras la entrada valga 0...
        {
                                                    //  Mantenernos esperando.
        }
        Tiempo = reiniciaCuenta();
        TRAZA[p++] = Tiempo;    //!!!!!
        if (Tiempo < 40 || Tiempo > 67)             //  Si el tiempo está fuera del margen.
        {
            return 0;                               //  Error al leer el bit, retorna 0.
        }
        Tiempo = 0;
        reiniciaCuenta();
        while   (   ENTRADA  && US_AHORA    <   100)        //  Mientras la entrada valga 1...
        {
                                                    //  Mantenernos esperando.
        }
        Tiempo = reiniciaCuenta();
        TRAZA[p++] = Tiempo;    //!!!!!
        if (  Tiempo > 60 && Tiempo < 80)           //  Si entra en el márgen del 1...
        {
            Rx  |=  1 << (7-i);                     //  Añadimos un 1.
        }
        else
        {
            if ( Tiempo < 10 || Tiempo >    100)    //  Si se ha salido del márgen...
            {
                return 0;                           //  Error al leer el bit, retorna 0.
            }
        }

    }
    return Rx;
}
/**------------------------------------------------------------------------------------------------------//
//                                                                                                      //
                                                                                                        //
//      @end        ENDFILE.                                                                            //
//                                                                                                      //
//------------------------------------------------------------------------------------------------------**/
```

PWM.h

```
/**-------------------------------------------------------------------------------------------------------------------//
//      @filename     PWM.h                                                                                          //
//      @version      0.00                                                                                           //
//      @author       Alberto Palomo Alonso                                                                          //
//                                                                                                                   //
//      @brief        Este es el programa donde están definidas las funciones a utilizar en el módulo                //
//                    PWM dedicado al proyecto de Sistemas electrónicos digitales avanzados (UAH - EPS).             //
//                                                                                                                   //
//                                                                                                                   //
//      @category     Opcional.                                                                                      //
//                                                                                                                   //
//      @map          @include                                                                                       //
//                    @private                                                                                       //
//                    @funcdef                                                                                       //
//                    @end                                                                                           //
//                                                                                                                   //
//                                                                                                                   //
//      |---|    |       |    |\    /|                                                                              //
//      |    |   |       |      | \   / |                                                                            //
//      |---|      |         |       |   \/    |                                                                    //
//      |           |   | |     |      |                                                                             //
//      |             | |       |      |                                                                             //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
                                                                                                                     //
//      @include      Estos son los archivos utilizados con el código PWM.                                           //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
#ifndef LPC17XX
#define LPC17XX
#include "LPC17XX.H"
#endif

#ifndef SYSTEMSYMBOLS
#define SYSTEMSYMBOLS
#include "Systemsymbols.h"
#endif
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
                                                                                                                     //
//      @private      Estos son los símbolos correspondientes al módulo PWM.                                         //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
#define       Fpwm          50              // Velocidad de 20Hz.
#define       Tpwm          0.02                // Periodo de 20ms.

#define       TCR_MASK          0x9             // Activo el contador y el PWM.
#define       MCR_MASK          0x4             // Para el set del PWM.
#define       MCR_MASK_RESET    0x5                 // Para resetear el MCR.
#define       PCONP_MASK 0x1 << 6           // Para encender el PWM.

#define       MODO_CICLO 1                // Para la función modificaPulso().
#define       MODO_SERVO     0                // Para la función modificaPulso().

#define       PWM1          1
#define       PWM2          2
#define       PWM3          3
#define       PWM4          4
#define       PWM5          5
#define       PWM6          6

#define       OPEN_DRAIN  0x2
#define       PULL_UP       0x0
#define       PULL_DOWN         0x3
#define       UNUSED        0x1
```

```
#define    KMX              1.3          //  Constante de corrección máxima.
#define    KMN              0.6          //  Constante de corrección mínima.
#define    MINIMO_SERVO     0.001
#define    MAXIMO_SERVO     0.002

#define    TEMP_MAX         32           //  Visionado de temperatura.
#define    TEMP_MIN         17           //  Visionado de temperatura.
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
//                                                                                                             //
//      @funcdef      Estas son las funciones correspondientes al módulo PWM.                                  //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
void    __configuraPWM__  (  float   FrecuenciaPWM     ,  uint16_t   CualesPWM   );  //  Default = 0x3F3F -> Todo activado.
void    modificaPulso     (  uint32_t   PWMn           ,  uint8_tModo     ,  uint8_tCiclo      ,  uint8_t Grados    ,  float
Minimo    ,  float Maximo  );
void softMod          (  uint8_tGradosObjetivo      ,  uint8_tGradosActuales,  float Minimo  ,  float   Maximo   ,  uint32_t
    PWMn);
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
//                                                                                                             //
//      @end      ENDFILE.                                                                                     //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
```

PWM.c

```
/**-------------------------------------------------------------------------------------------------------------------//
//      @filename     PWM.c                                                                                          //
//      @version      0.00                                                                                           //
//      @author       Alberto Palomo Alonso                                                                          //
//                                                                                                                   //
//      @brief      Este es el programa donde están definidas las funciones a utilizar en el módulo                  //
//                  PWM dedicado al proyecto de Sistemas electrónicos digitales avanzados (UAH - EPS).               //
//                                                                                                                   //
//                                                                                                                   //
//      @category     Opcional.                                                                                      //
//                                                                                                                   //
//      @map          @include                                                                                       //
//                  @variables                                                                                       //
//                  @function                                                                                        //
//                  @end                                                                                             //
//                                                                                                                   //
//      |---|    |       |    |\    /|                                                                              //
//      |    |   |       |      | \   / |                                                                            //
//      |---|      |         |       |    \/    |                                                                   //
//      |           |    | |      |      |                                                                          //
//      |             | |        |      |                                                                          //
//                                                                                                                   //
//--------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
                                                                                                                     //
//      @include      Includes pertenecientes al módulo del PWM.                                                     //
//                                                                                                                   //
//--------------------------------------------------------------------------------------------------------------**/
#ifndef PWM
#define     PWM
#include "PWM.h"
#endif
#ifndef SYSTEMSYMBOLS
#define     SYSTEMSYMBOLS
#include    "Systemsymbols.h"
#endif
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
                                                                                                                     //
//      @variables        Variables del fichero.                                                                     //
//                                                                                                                   //
//--------------------------------------------------------------------------------------------------------------**/
extern Counters_t     *    COUNTERS;
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
                                                                                                                     //
//      @function     __configuraPWM__()                                                                             //
//                                                                                                                   //
//      @brief      Configura el PWM en función de la frecuencia a utilizar y que PWM se quieren usar.               //
//                  El primer byte activa las del puerto 2 y las del segundo las del otro puerto en orden            //
//                  ascendente.                                                                                      //
//                                                                                                                   //
//          @FrecuenciaPWM          Frecuencia a la que se desea ajustar el ciclo PWM. En Hz.                        //
//          @CualesPWM      Máscara que define los PWM a configurar, los 6 primeros bits corresponden al             //
//                          puerto 2. Del 9º bit al 14º bit corresponden al puerto 1. En orden ascendente            //
//                          desde del PWM1.1 al PWM1.6                                                               //
//                                                                                                                   //
//                                                                                                                   //
//--------------------------------------------------------------------------------------------------------------**/
void __configuraPWM__(float    FrecuenciaPWM   ,    uint16_t    CualesPWM    )
{
    LPC_SC->PCONP       |=  PCONP_MASK;                              // Enciendo el PWM.
```

```
    LPC_PWM1->MR0        =   ((uint32_t)(Ftick/FrecuenciaPWM) - 1);          // Configuro la frecuencia.
    LPC_PWM1->TCR        |=  TCR_MASK;                                       // Activo el PWM.
    LPC_PWM1->MCR        &= ~TODO_1_32;                                      // Reset al MCR.
    LPC_PWM1->MCR          |=    0x2;                                        // Ahora el MR0 resetea el contador.
    LPC_PINCON->PINSEL4 &=    ~0xFFF;                                       // Reset en pines PWM puerto 2.
    LPC_PINCON->PINSEL3 &=    ~0x33CF30;                                    // Reset en pines PWM puerto 1.

    for (COUNTERS->i = 0; COUNTERS->i < 6; (COUNTERS->i)++)                 // Para el puerto 2: seleccionados.
    {
        if (  (CualesPWM >> COUNTERS->i) & ~0xFFFE)                         // Miro si está seleccionado el iésimo.
        {
            LPC_PINCON->PINSEL4    |= (FUNC1      << (2*COUNTERS->i));// Pongo la función 1 en los pines PWM.
            LPC_PWM1->PCR |= (0x1 << (COUNTERS->i + 0x9));                  // Pongo la función de enable output en el PWM.
        }
    }
    /** @REMARK:    Esto se configuraría como open drain sobre todo por no perder potencia, pero prefiero aseugrar con pull.*/
    for (COUNTERS->i = 6; COUNTERS->i < 12; (COUNTERS->i)++)                // Para el otro puerto: seleccionados.
    {
        if (  (CualesPWM >> (COUNTERS->i + 2)) & ~0xFFFE)                   // Miro si está seleccionado el iésimo.
        {
            LPC_PWM1->PCR |= (0x1 << (COUNTERS->i - 0x6 + 0x9));            // Pongo la función de enable output en el PWM.
            switch (COUNTERS->i)                                        // Pongo la función 2 en los pines PWM.
            {
                case 6:
                    LPC_PINCON->PINSEL3    |=  FUNC2 <<  2*2;
//                  LPC_PINCON->PINMODE3    |=  OPEN_DRAIN <<   2*2;
                    break;
                case 7:
                    LPC_PINCON->PINSEL3    |=  FUNC2 <<  2*4;
//                  LPC_PINCON->PINMODE3    |=  OPEN_DRAIN <<   2*4;
                    break;
                case 8:
                    LPC_PINCON->PINSEL3    |=  FUNC2 <<  2*5;
//                  LPC_PINCON->PINMODE3    |=  OPEN_DRAIN <<   2*5;
                    break;
                case 9:
                    LPC_PINCON->PINSEL3    |=  FUNC2 <<  2*7;
//                  LPC_PINCON->PINMODE3    |=  OPEN_DRAIN <<   2*7;
                    break;
                case 10:
                    LPC_PINCON->PINSEL3    |=  FUNC2 <<  2*8;
//                  LPC_PINCON->PINMODE3    |=  OPEN_DRAIN <<   2*8;
                    break;
                case 11:
                    LPC_PINCON->PINSEL3    |=  FUNC2 <<  2*10;
//                  LPC_PINCON->PINMODE3    |=  OPEN_DRAIN <<   2*8;
                    break;
                default:
                    break;
            }
        }
    }
    COUNTERS->i = 0;                                                    // Dejo el contador a 0.
}
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
//      @function       modificaPulso()                                                                              //
//                                                                                                                   //
//      @brief      Configura el    pulso de PWM en función del ciclo de trabajo o de valores de oscilación          //
//                  si se encuentra en modo servo.                                                                   //
//                                                                                                                   //
//              @PWMn   Selecciona el PWM1.n a modificar.                                                            //
//              @Modo   Selecciona si modo servo (valores)o modo ciclo (en porcentaje).                              //
//              @Ciclo Selecciona el ciclo de trabajo para el modo ciclo.                                            //
//              @Grados   Selecciona los grados a rotar el servo en modo servo.                                      //
```

```
//              @Minimo  Selecciona el mínimo valor de Ton del ciclo PWM. En segundos.                          //
//              @Maximo Selecciona el máximo valor de Ton del ciclo PWM. En segundos.                           //
//                                                                                                              //
//                                                                                                              //
//                                                                                                              //
//                                                                                                              //
//                                                                                                              //
//------------------------------------------------------------------------------------------------------------**/
void    modificaPulso(    uint32_t    PWMn ,    uint8_t Modo ,    uint8_t Ciclo   ,    uint8_tGrados,    float Minimo   ,    float
    Maximo    )
{
    if (PWMn > 3)
    {
        PWMn += 6;   //  Debido a la asimétrica distribución de los registros.
    }

    Minimo *= KMN;          //  Definitivamente había algo mal.
    Maximo *= KMX;          //  Estos servos utilizan la potencia del pulso y no precisamente su duración.

    /** @REMARK:    La potencia entregada no es la debida. En la datasheet especifica pulsos del rango de 5V, se ofrece uno
                de 3.3V, se ha podido usar un transistor, pero deduzco que estos servos utilizan la potencia de la señal
                PWM para obtener el ángulo y modificando los tiempos podemos entregar más potencia de señal.*/

    switch(     Modo  )
    {
        case MODO_CICLO:   // Escribo en LPC_PWM1->MRn el valor correspondiente al porcentaje de MR0; < 1.
            *(__IO uint32_t *)((uint32_t)&(LPC_PWM1->MR0) + (uint32_t)(0x4*PWMn)) =
(uint32_t)((float)(LPC_PWM1->MR0)*((float)Ciclo/(float)100));
            break;
        case MODO_SERVO:  // Escribo en LPC_PWM1->MRn el valor correspondiente al tiempo Ton en función del grado.
            *(__IO uint32_t *)((uint32_t)&(LPC_PWM1->MR0) + (uint32_t)(0x4*PWMn)) = (uint32_t)((Ftick*(Minimo + (Maximo -
Minimo)*(float)(Grados/(float)(180)))- (float)1));
            break;
        default:
            break;
    }
        if (PWMn > 3)
    {
        PWMn -= 6;   //  Devolvemos PWMn a su estado oriegen.
    }
    LPC_PWM1->LER |= 0x1 << PWMn | 0x1;        // Activo el load de los MR0 y MRn.
}
/**------------------------------------------------------------------------------------------------------------//
//                                                                                                              //
                                                                                                                //
//      @end       ENDFILE.                                                                        //
//                                                                                                              //
//------------------------------------------------------------------------------------------------------------**/
```

DAC.h

```
/**-------------------------------------------------------------------------------------------------------------//
//      @filename     DAC.h                                                                                    //
//      @version                                                                                               //
//      @author       Alberto Palomo Alonso                                                                    //
//                                                                                                             //
//      @brief      Cabecera para el código de DAC.c                                                           //
//                                                                                                             //
//      @category     Periférico.                                                                              //
//                                                                                                             //
//      @map          @include                                                                                 //
//                    @private                                                                                 //
//                    @funcdef                                                                                 //
//                    @end                                                                                     //
//                                                                                                             //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
                                                                                                               //
//      @include      Estos son los archivos utilizados con el código de configuración.                        //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
#ifndef LPC17xx
#define     LPC17XX
#include    "LPC17XX.H"
#endif
#ifndef SYSTEMSYMBOLS
#define     SYSTEMSYMBOLS
#include    "Systemsymbols.h"
#endif
#ifndef LUT
#define     LUT
#include    "LUT.h"
#endif
#ifndef TIMERS
#define     TIMERS
#include    "Timers.h"
#endif
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
                                                                                                               //
//      @private      Estos son los símbolos correspondientes a la configuración.                              //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
#define     DAC_PIN         (28 - 16)           //  Pin del DAC.
#define     DAC_FUNC          0x2               //  Función 2 del pin 28.
#define     DAC_BIAS        (1 << 16)           //  2.5 us de upstream.
#define     BORRAR_DAC  (0x3FF << 6)            //  10 bits de DAC.
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
                                                                                                               //
//      @funcdef      Estas son las funciones correspondientes a la configuración.                             //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
void __configuraDAC__   (   void    );
void    activarDac      (   void    );
void desactivarDAC      (   void    );
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
                                                                                                                 //
//      @end      ENDFILE.                                                                                     //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
```

DAC.c

```
/**-------------------------------------------------------------------------------------------------------------------//
//      @filename       DAC.c                                                                                       //
//      @version        6.01                                                                                        //
//      @author         Alberto Palomo Alonso                                                                       //
//                                                                                                                  //
//      @brief      Código fuente que contiene las funciones para audio (DAC).                                      //
//                                                                                                                  //
//      @category       Periférico.                                                                                 //
//                                                                                                                  //
//      @map            @include                                                                                    //
//                  @variables                                                                                      //
//                  @funcion                                                                                        //
//                  @end                                                                                            //
//                                                                                                                  //
//                                                                                                                  //
//-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                  //
//                                                                                                                  //
//      @include        Estos son los archivos utilizados en el código de LDR.                                      //
//                                                                                                                  //
//-------------------------------------------------------------------------------------------------------------------**/
#ifndef DAC
#define     DAC
#include    "DAC.h"
#endif
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                  //
//                                                                                                                  //
//      @variables          Variables del fichero.                                                                  //
//                                                                                                                  //
//-------------------------------------------------------------------------------------------------------------------**/
extern actualizador_t *    ACTUALIZADOR;
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                  //
//                                                                                                                  //
//      @funcion        __configuraDAC__()                                                                          //
//                                                                                                                  //
//      @brief      Función de configuración del DAC y su timer (Timer2).                                           //
//                                                                                                                  //
//      @REMARK:            Para activar un periodo del DAC (2 segundos)                                            //
                        //
//                                                                                                                  //
//-------------------------------------------------------------------------------------------------------------------**/
void __configuraDAC__()
{
    LPC_GPIO3->FIODIR   |= (1  <<  LECTURA_AUDIO  ) | (1   <<  ESCRITURA_AUDIO);       // Leds de lectura / escritura de audio.
    LPC_GPIO3->FIOCLR   =       (1  <<  LECTURA_AUDIO);                 //Turn ON LED1
    LPC_GPIO3->FIOCLR   =       (1  <<  ESCRITURA_AUDIO);               //Turn ON LED2
}
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                  //
//                                                                                                                  //
//      @funcion        activarDac()                                                                                //
//                                                                                                                  //
//      @brief      Señal de activar el timer del DAC.                                                              //
//                                                                                                                  //
//      @REMARK:            Utiliza DMA.                                                                            //
//                                                                                                                  //
//-------------------------------------------------------------------------------------------------------------------**/
void activarDac()
{
    /**@TODO:   DMA*/
    LPC_GPDMACH0->DMACCConfig|=  1;                             //  Activo el DMA.
```

```
    LPC_TIM1->MCR              =  (1 <<3    )  |  (1 << 4);     // Activo la interrupción por MR1 y reset por MR1.
    LPC_TIM1->MR1              =  (Fclk*DURACION_AUDIO) - 1;          // Valor de MR1.
    LPC_TIM1->TCR              =  0x2;                            // Reset del timer.
    LPC_TIM1->TCR              =  0x1;                            // El timer cuenta.
    ACTUALIZADOR->Audiorev     =  0;                              // Señalizo el bloqueo de audio.
    LPC_GPIO3->FIOSET          =  (  1  << ESCRITURA_AUDIO);      // Señaizo escritura de audio.
}
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
//                                                                                                             //
//      @funcion       activarDac()                                                                            //
//                                                                                                             //
//      @brief     Señal de activar el timer del DAC.                                                          //
//                                                                                                             //
//      @REMARK:         Activador del DAC.                                                                    //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
void desactivarDAC()
{
    LPC_GPDMACH0->DMACCConfig&= ~0x1;                             // Desactivo el DMA.
    ACTUALIZADOR->Audiorev     =  1;                              // Señalizo el fin del DAC.
    LPC_GPIO3->FIOCLR          =  (  1  << ESCRITURA_AUDIO);      // Señaizo escritura de audio.
    LPC_TIM1->MCR              &= ~(7 << 3);                      // Desactivo la interrupción por MR1 y reset tras MR1.
    LPC_DAC->DACR              =  0;                              // No hay señal de salida.
}
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
//                                                                                                             //
//      @end       ENDFILE.                                                                                    //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
```

LUT.h

```
/**-------------------------------------------------------------------------------------------------------------------//
//      @filename     LUT.g                                                                                          //
//      @version      0.00                                                                                               //
//      @author       Alberto Palomo Alonso                                                                                  //
//                                                                                                                   //
//      @brief      Este es el programa donde se encuentra la cabecera de LUT.c                                                 //
//                                                                                                                   //
//      @category     Opcional.                                                                                          //
//                                                                                                                   //
//      @map          @include                                                                                           //
//                  @private                                                                                           //
//                  @funcdef                                                                                           //
//                  @end                                                                                               //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
                                                                                                                            //
//      @include      Includes pertenecientes al módulo del anemómetro.                                                        //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
#ifndef SYSTEMSYMBOLS
#define    SYSTEMSYMBOLS
#include   "Systemsymbols.h"
#endif
#ifndef LDR
#define    LDR
#include   "LDR.h"
#endif
#ifndef DMA
#define    DMA
#include   "DMA.h"
#endif
#ifndef MATH
#define    MATH
#include   <math.h>
#endif

#define    MUESTRAS_SENO     32
#define    BRILLO_LDR        0
#define    BRILLO2CICLO_LDR  1
#define    INDICE_UVA        2
#define    BRILLO_LDR_NOLUT  3

void goto_LUT(    float dato , uint8_t LUTn , float * ret_f , uint8_t * ret_8 , uint16_t * ret_16 , uint32_t * ret_32);
void crearSeno(   void    );
void ponTonoDMA(  void    );
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
                                                                                                                            //
//      @end      ENDFILE.                                                                                               //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
/**-------------------------------------------------------------------------------------------------------------------//
//      @filename     LUT.g                                                                                          //
//      @version      0.00                                                                                               //
//      @author       Alberto Palomo Alonso                                                                                  //
//                                                                                                                   //
//      @brief      Este es el programa donde se encuentra la cabecera de LUT.c                                                 //
//                                                                                                                   //
//      @category     Opcional.                                                                                          //
//                                                                                                                   //
//      @map          @include                                                                                           //
//                  @private                                                                                           //
//                  @funcdef                                                                                           //
```

```
//                  @end                                                                                    //
//                                                                                                          //
//----------------------------------------------------------------------------------------------------------//
//                                                                                                          //
                                                                                                                //
//      @include        Includes pertenecientes al módulo del anemómetro.                                   //
//                                                                                                          //
//----------------------------------------------------------------------------------------------------------**/
#ifndef SYSTEMSYMBOLS
#define     SYSTEMSYMBOLS
#include    "Systemsymbols.h"
#endif
#ifndef LDR
#define     LDR
#include    "LDR.h"
#endif
#ifndef DMA
#define     DMA
#include    "DMA.h"
#endif
#ifndef MATH
#define     MATH
#include    <math.h>
#endif

#define     MUESTRAS_SENO       32
#define     BRILLO_LDR          0
#define     BRILLO2CICLO_LDR    1
#define     INDICE_UVA          2
#define     BRILLO_LDR_NOLUT    3

void goto_LUT(      float dato , uint8_t LUTn , float * ret_f , uint8_t * ret_8 , uint16_t * ret_16 , uint32_t * ret_32);
void crearSeno(     void    );
void ponTonoDMA(    void    );
/**--------------------------------------------------------------------------------------------------------//
//                                                                                                          //
                                                                                                                //
//      @end        ENDFILE.                                                                                //
//                                                                                                          //
//----------------------------------------------------------------------------------------------------------**/
```

LUT.c

```
/**-------------------------------------------------------------------------------------------------------------//
//      @filename     LUT.c                                                                                    //
//      @version      0.00                                                                                     //
//      @author       Alberto Palomo Alonso                                                                    //
//                                                                                                             //
//      @brief        Este es el programa donde se encuentran las look up tables que optimizan                 //
//                    el uso de cpu sacrificando memoria, pero como uso una SD la memoria no es                //
//                    un problema grande.                                                                      //
//                                                                                                             //
//      @category     Opcional.                                                                                //
//                                                                                                             //
//      @map          @include                                                                                 //
//                    @variables                                                                               //
//                    @LUT                                                                                     //
//                    @function                                                                                //
//                    @end                                                                                     //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
                                                                                                               //
//      @include      Includes pertenecientes al módulo del anemómetro.                                        //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
#ifndef LUT
#define    LUT
#include   "LUT.h"
#endif
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
                                                                                                               //
//      @variables        Variables del fichero.                                                               //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
uint8_tPREGRABADA[MUESTRAS_SENO];
extern uint8_t*   AUDIO;
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
                                                                                                               //
//      @LUT          LookUpTables                                                                             //
//                                                                                                             //
//      @brief        Bases de datos.                                                                          //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
uint8_tBrillo2ciclo_LDR[] =
{
   13 ,   20 ,   25 ,   30 ,   35 ,   40 ,   45 ,   50 ,   55 ,   60 ,
   65 ,   70 ,   75 ,   80
                                                                        };
uint8_tBrillo_LDR[]   =
{
   21 ,   1  ,   5  ,   10 ,   15 ,   20 ,   25 ,   30 ,   35 ,   40 ,
   45 ,   50 ,   65 ,   60 ,   65 ,   70 ,   75 ,   80 ,   85 ,   90 ,
   95 ,   100
                                                                        };
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
                                                                                                               //
//      @function     goto_LUT()                                                                               //
//                                                                                                             //
//      @brief        Esta función es la que mira las LUT y obtiene el dato que queremos de la base            //
//                    de datos.                                                                                //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
void goto_LUT( float variable , uint8_t LUTn , float * ret_flotante , uint8_t * ret_int8 , uint16_t * ret_int16 , uint32_t * ret_int32)
```

```
{
    switch( LUTn )
    {
        case BRILLO_LDR:
            *ret_flotante = Brillo_LDR[         (uint8_t)(((variable - LDRRESISTENCIA_MIN) /(LDRRESISTENCIA_MAX -
LDRRESISTENCIA_MIN))    *Brillo_LDR[0]) + 1];
            break;
        case    BRILLO2CICLO_LDR:
            *ret_int8   = Brillo2ciclo_LDR[         (uint8_t)(((variable - BRILLO_MIN)              /(BRILLO_MAX - BRILLO_MIN))
    *Brillo2ciclo_LDR[0]) + 1];
            break;
        case    INDICE_UVA:
            *ret_flotante = variable;  //  El output DC corresponde al índice, es muy sencillo traducirlo, se recomienda no llamar a esta
función en este modo.
            break;
        case BRILLO_LDR_NOLUT:
            *ret_flotante = -(1.0102)*variable + 102.0204;
            if (*ret_flotante < 0)
            {
                *ret_flotante = 0;
            }
        default:
            break;
    }
}
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
                                                                                                                        //
//      @end       ENDFILE.                                                                                          //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
```

DMA.h

```
/**-------------------------------------------------------------------------------------------------------------//
//      @filename     DMA.h                                                                                    //
//      @version      0.00                                                                                        //
//      @author       Alberto Palomo Alonso                                                                          //
//                                                                                                          //
//      @brief      Cabecera del configurado del DMA.                                                              //
//                                                                                                          //
//      @category     Opcional.                                                                                 //
//                                                                                                          //
//      @map          @include                                                                                  //
//                  @private                                                                                    //
//                  @funcdef                                                                                    //
//                  @end                                                                                        //
//                                                                                                          //
//                                                                                                          //
//-------------------------------------------------------------------------------------------------------------//
//                                                                                                          //
                                                                                                                    //
//      @include      Estos son los archivos utilizados con el código del DMA.                                        //
//                                                                                                          //
//-------------------------------------------------------------------------------------------------------------**/
#ifndef SYSTEMSYMBOLS
#define     SYSTEMSYMBOLS
#include    "Systemsymbols.h"
#endif
#ifndef LDR
#define     LDR
#include    "LDR.h"
#endif
#ifndef LUT
#define     LUT
#include    "LUT.h"
#endif
#include <LPC17xx.H>
#include <math.h>
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                          //
                                                                                                                    //
//      @private      Estos son los símbolos correspondientes al DMA.                                                 //
//                                                                                                          //
//-------------------------------------------------------------------------------------------------------------**/
#define     N_samples_wave          32    // Nº de muestras por ciclo
#define     Ftono             400
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                          //
                                                                                                                    //
//      @funcdef  Estas son las funciones que se usan en el DMA.                                                  //
//                                                                                                          //
//-------------------------------------------------------------------------------------------------------------**/
void __configuraDMA__  (    void    );  //  Configurador del DMA.
void __configuraTono__  (    void    );  //  Configurador del tono.
void __configuraAudio__ (    void    );  //  Configurador del audio.
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                          //
                                                                                                                  //
//      @end      ENDFILE.                                                                                      //
//                                                                                                          //
//-------------------------------------------------------------------------------------------------------------**/
```

DMA.c

```
/**-------------------------------------------------------------------------------------------------------------//
//      @filename       DMA.c                                                                                  //
//      @version        0.00                                                                                       //
//      @author         Alberto Palomo Alonso                                                                          //
//                                                                                                             //
//      @brief      Código fuente que contiene las funciones para audio (DMA).                                         //
//                                                                                                             //
//      @category       Periférico.                                                                            //
//                                                                                                             //
//      @map            @include                                                                                   //
//                      @variables                                                                             //
//                      @funcion                                                                               //
//                      @end                                                                                   //
//                                                                                                             //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
                                                                                                                       //
//      @include        Estos son los archivos utilizados en el código de LDR.                                         //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
#ifndef DMA
#define     DMA
#include    "DMA.h"
#endif
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
                                                                                                                       //
//      @variables          Variables del fichero.                                                                 //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
extern actualizador_t *    ACTUALIZADOR;
extern uint8_t AUDIO[MUESTRAS_AUDIO];
DMA_t LLI0;
uint8_t sinusoide[N_samples_wave];
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
                                                                                                                       //
//      @funcion        __configuraDMA__()                                                                             //
//                                                                                                             //
//      @brief      Función de configuración del DMA.                                                              //
//                                                                                                             //
//      @REMARK:            Para activar un periodo del DAC (2 segundos)                                               //
                        //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
void __configuraDMA__ (void)
{
    int i;
    for(i=0; i < N_samples_wave; i++)
    {
        sinusoide[i] = (int)(127 + 127*sin(2*PI*i/N_samples_wave)); // DACR bit 6-15 VALUE (valor ya desplazado!!!)
    }
    LPC_PINCON->PINSEL1 |= (2<<20);                     // enable AOUT (P0.26) pin
    __configuraTono__();
}
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
                                                                                                                       //
//      @funcion        __configuraTono__()                                                                            //
//                                                                                                             //
//      @brief      Función de configuración del tono.                                                             //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
```

```
void __configuraTono__()
{
    // Linked CH0
    LLI0.source      = (uint32_t) &sinusoide[0];
    LLI0.destination = (uint32_t) &(LPC_DAC->DACR) + 1;
    LLI0.next        = (uint32_t) &LLI0;
    LLI0.control     = 1<<26 | 0<<21 | 0<<18 | N_samples_wave; //Transfersize=  N_samples_wave, SWidth=8bits, DWidth=8bits,
Source Increment



    LPC_SC->PCONP   |= (1<<29);                                  // Power DMA
    LPC_GPDMA->DMACConfig = 1;                                   // enable the GPDMA controller
    LPC_GPDMA->DMACSync   = (1<<6);                              // enable synchro logic for all reqs

    LPC_GPDMACH0->DMACCSrcAddr  = (uint32_t) &sinusoide[0];
    LPC_GPDMACH0->DMACCDestAddr = (uint32_t) &(LPC_DAC->DACR) + 1;
    LPC_GPDMACH0->DMACCLLI      = (uint32_t) &LLI0; // linked lists for ch0
    LPC_GPDMACH0->DMACCControl  = N_samples_wave     // transfer size (0 - 11) = 32 muestras /ciclo
                                | (0 << 12)          // source burst size (12 - 14) = 1
                                | (0 << 15)          // destination burst size (15 - 17) = 1
                                | (0 << 18)          // source width (18 - 20) = 32 bit          CAMBIADO
                                | (0 << 21)          // destination width (21 - 23) = 32 bit          NO CAMBIADO
                                | (0 << 24)          // source AHB select (24) = AHB 0
                                | (0 << 25)          // destination AHB select (25) = AHB 0
                                | (1 << 26)          // source increment (26) = increment
                                | (0 << 27)          // destination increment (27) = no increment
                                | (0 << 28)          // mode select (28) = access in user mode
                                | (0 << 29)          // (29) = access not bufferable
                                | (0 << 30)          // (30) = access not cacheable
                                | (0 << 31);         // terminal count interrupt disabled
    LPC_GPDMACH0->DMACCConfig   =  0                                 // channel enabled (0)
                | (0 << 1)                  // source peripheral (1 - 5) = none
                | (7 << 6)                  // destination peripheral (6 - 10) = DAC
                | (1 << 11)                 // flow control (11 - 13) = MEM to PERF
                | (0 << 14)                 // (14) = mask out error interrupt
                | (0 << 15)                 // (15) = mask out terminal count interrupt
                | (0 << 16)                 // (16) = no locked transfers
                | (0 << 18);                // (27) = no HALT

    //F_out (salida del DAC)
    LPC_DAC->DACCNTVAL = Fclk/N_samples_wave/Ftono -1;   // (Ts DAC = F_out/N_samples   < Tsetup DAC = 1useg. !!!!)

    /* DMA, timer running, dbuff */
    LPC_DAC->DACCTRL    = 1<<3 | 1<<2 | 1<<1;
    ACTUALIZADOR->Audiorev = 1;
}
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                    //
//      @funcion      __configuraAudio__()                                                                            //
//                                                                                                                    //
//      @brief      Función de configuración del audio.                                                               //
//                                                                                                                    //
//-------------------------------------------------------------------------------------------------------------------**/
void __configuraAudio__()
{
    // Linked CH0
    LLI0.source      = (uint32_t) AUDIO;
    LLI0.destination = (uint32_t) &(LPC_DAC->DACR) + 1;
    LLI0.next        = (uint32_t) &LLI0;
    LLI0.control     = 1<<26 | 0<<21 | 0<<18 | MUESTRAS_AUDIO;     //  Incremento origen, MUESTRAS_AUDIO muestras, 8 bits
todo.

    LPC_SC->PCONP          |=  (1<<29);                  //  Enciendo el DMA.
    LPC_GPDMA->DMACConfig   =   1;                       //  Activo el controlador del DMA.
```

```
    LPC_GPDMA->DMACSync     =   (1<<6);                     //   Sincronización de registros.

    LPC_GPDMACH0->DMACCSrcAddr    = (uint32_t) AUDIO;            //   Empieza en AUDIO.
    LPC_GPDMACH0->DMACCDestAddr = (uint32_t) &(LPC_DAC->DACR) + 1; //   Ve a DACR + 2.
    LPC_GPDMACH0->DMACCLLI          = (uint32_t) &LLI0; // linked lists for ch0
    LPC_GPDMACH0->DMACCControl    = MUESTRAS_AUDIO      // transfer size (0 - 11) = 32 muestras /ciclo
                                  | (0 << 12)                 // source burst size (12 - 14) = 1
                                  | (0 << 15)                 // destination burst size (15 - 17) = 1
                                  | (0 << 18)                 // source width (18 - 20) = 32 bit
                                  | (2 << 21)                 // destination width (21 - 23) = 32 bit
                                  | (0 << 24)                 // source AHB select (24) = AHB 0
                                  | (0 << 25)                 // destination AHB select (25) = AHB 0
                                  | (1 << 26)                 // source increment (26) = increment
                                  | (0 << 27)                 // destination increment (27) = no increment
                                  | (0 << 28)                 // mode select (28) = access in user mode
                                  | (0 << 29)                 // (29) = access not bufferable
                                  | (0 << 30)                 // (30) = access not cacheable
                                  | (0 << 31);                // terminal count interrupt disabled
    LPC_GPDMACH0->DMACCConfig     =   0                               // channel enabled (0)
                  | (0 << 1)                    // source peripheral (1 - 5) = none
                  | (7 << 6)                    // destination peripheral (6 - 10) = DAC
                  | (1 << 11)                   // flow control (11 - 13) = MEM to PERF
                  | (0 << 14)                   // (14) = mask out error interrupt
                  | (0 << 15)                   // (15) = mask out terminal count interrupt
                  | (0 << 16)                   // (16) = no locked transfers
                  | (0 << 18);                  // (27) = no HALT

    //F_out (salida del DAC)
    LPC_DAC->DACCNTVAL = (Fclk/4000) - 1;    // (Ts DAC = TsAudio    < Tsetup DAC = 1useg. !!!!)

    /* DMA, timer running, dbuff */
    LPC_DAC->DACCTRL      = 1<<3 | 1<<2 | 1<<1;
    ACTUALIZADOR->Audiorev = 1;
}
/**-----------------------------------------------------------------------------------------------------------------//
//                                                                                                                  //
                                                                                                                                //
//        @end        ENDFILE.                                                                                             //
//                                                                                                                  //
//------------------------------------------------------------------------------------------------------------------**/
```

WDT.h

```
/**-------------------------------------------------------------------------------------------------------------//
//      @filename       WDT.h                                                                                  //
//      @version        0.00                                                                                       //
//      @author         Alberto Palomo Alonso                                                                          //
//                                                                                                          //
//      @brief      Cabecera del configurado del WDT.                                                              //
//                                                                                                          //
//      @category       Opcional.                                                                              //
//                                                                                                          //
//      @map            @include                                                                               //
//                  @private                                                                                   //
//                  @funcdef                                                                                   //
//                  @end                                                                                       //
//                                                                                                          //
//                                                                                                          //
//-------------------------------------------------------------------------------------------------------------//
//                                                                                                          //
                                                                                                                    //
//      @include        Estos son los archivos utilizados con el código del WDT.                                //
//                                                                                                          //
//-------------------------------------------------------------------------------------------------------------**/
#ifndef SYSTEMSYMBOLS
#define     SYSTEMSYMBOLS
#include    "Systemsymbols.h"
#endif
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                          //
                                                                                                                    //
//      @private        Estos son los símbolos correspondientes al WDT.                                             //
//                                                                                                          //
//-------------------------------------------------------------------------------------------------------------**/
#define     WATCHDOG_TIMEOUT    10          //  En segundos.
#define     Fwdt                Fclk/(float)4   //  Reloj seleccionado del WDT.
#define     WDMOD_MASK          0x03            //  Se activa y resetea el programa tras el timeout.
#define     WDCLKSEL_MASK       0x01            //  Se ha escogido el reloj Fclk.
#define     WDT_CODIGO1         0xAA            //  Primer código del WDT.
#define     WDT_CODIGO2         0x55            //  Segundo código del WDT.
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                          //
                                                                                                                    //
//      @funcdef  Estas son las funciones que se usan en el WDT.                                                //
//                                                                                                          //
//-------------------------------------------------------------------------------------------------------------**/
void    __configuraWDT__    (   void    );
void alimentaWDT            (   void    );
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                          //
                                                                                                                        //
//      @end        ENDFILE.                                                                                    //
//                                                                                                          //
//-------------------------------------------------------------------------------------------------------------**/
```

WDT.c

```
/**-------------------------------------------------------------------------------------------------------------------//
//      @filename     WDT.c                                                                                          //
//      @version      0.00                                                                                           //
//      @author       Alberto Palomo Alonso                                                                          //
//                                                                                                                   //
//      @brief      Código fuente del configurado del WDT.                                                           //
//                                                                                                                   //
//      @category     Interno.                                                                                       //
//                                                                                                                   //
//      @map          @include                                                                                       //
//                  @function                                                                                        //
//                  @end                                                                                             //
//                                                                                                                   //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
                                                                                                                     //
//      @include      Estos son los archivos utilizados con el código del WDT.                                       //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
#ifndef WDT
#define    WDT
#include   "WDT.h"
#endif
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
                                                                                                                     //
//      @function __configuraWDT__()                                                                                //
//                                                                                                                   //
//      @brief Función que configura el WDT como un contador que observa si se ha bloqueado el programa.            //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
void   __configuraWDT__()
{
    LPC_WDT->WDTC     =   Fwdt*WATCHDOG_TIMEOUT     ;   //  Timeout de WATCHDOG_TIMEOUT segundos.
    LPC_WDT->WDCLKSEL   =   WDCLKSEL_MASK         ;   //  Se selecciona el reloj que se desea para el WDT.
    LPC_WDT->WDMOD      =   WDMOD_MASK            ;   //  Se selecciona la acción a realizar si WDT llega a cero.
    alimentaWDT();
}
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
                                                                                                                     //
//      @function alimentaWDT()                                                                                      //
//                                                                                                                   //
//      @brief Función que evita que el contador del WatchDogTimer llegue a 0.                                       //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
void   alimentaWDT()
{
    LPC_WDT->WDFEED  =   WDT_CODIGO1;
    LPC_WDT->WDFEED  =   WDT_CODIGO2;
}
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
                                                                                                                     //
//      @end       ENDFILE.                                                                                          //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
```

UART0.h

```
/**-------------------------------------------------------------------------------------------------------------------//
//      @filename       UART0.h                                                                                      //
//      @version        0.00                                                                                         //
//      @author         Alberto Palomo Alonso                                                                        //
//                                                                                                                   //
//      @brief      Este es el programa que recoge la transmisión por UART0.                                         //
//                                                                                                                   //
//      @category       Opcional.                                                                                    //
//                                                                                                                   //
//      @map            @include                                                                                     //
//                  @function                                                                                        //
//                  @end                                                                                             //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
                                                                                                                     //
//      @include        Includes pertenecientes a e la transmisión asíncrona.                                        //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
#ifndef LPC17XX
#define     LPC17XX
#include    "LPC17XX.H"
#endif
#ifndef SYSTEMSYMBOLS
#define     SYSTEMSYMBOLS
#include    "Systemsymbols.h"
#endif
#ifndef STDIO
#define     STDIO
#include    <stdio.h>
#endif
#ifndef STRING
#define     STRING
#include    <string.h>
#endif
#ifndef PWM
#define     PWM
#include    "PWM.h"
#endif
#ifndef UART
#define     UART
#include    "uart.h"
#endif
#ifndef MIGLOBAL
#define     MIGLOBAL
#include    "miGlobal.h"
#endif
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
                                                                                                                     //
//      @private        Estos son los símbolos correspondientes al protocolo UART0.                                  //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
#define     DLP                 7
#define     UART0_MTX           (1  <<  1)
#define     UART0_MRX           (1  <<  2)
#define     RetornoDeCarro      13
#define     CADMAX              120

#define     COM10               "GIVE SUGAR\r"
#define     COM11               "GIVE IP\r"
#define     COM12               "GIVE TEMPERATURA\r"
#define     COM13               "GIVE PRESION\r"
#define     COM14               "GIVE VIENTO\r"
```

```
#define	COM15		"GIVE LUGAR\r"
#define	COM16		"GIVE INDICEUV\r"
#define	COM17		"GIVE HORA\r"
#define	COM18		"GIVE HUMEDAD\r"
#define	COM19		"GIVE BRILLO\r"

#define	COM20		"SET BRILLO\r"
#define	COM21		"SET HORA\r"
#define	COM22		"SET MIN TEMP\r"
#define	COM23		"SET MAX TEMP\r"
#define	COM24		"SET MIN PRES\r"
#define	COM25		"SET MAX PRES\r"
#define	COM26		"SET TEMPERATURA\r"
#define	COM27		"SET PRESION\r"

#define	COM3		"KILL\r"

#define	COM0		"ABOUT\r"
#define	COM4		"HELP\r"
#define	COM41		"HELP GIVE\r"
#define	COM42		"HELP SET\r"

#define	UART_TX		0
#define	UART_RX_BRILLO	1
#define	UART_RX_MINT	2
#define	UART_RX_MAXT	3
#define	UART_RX_MINP	4
#define	UART_RX_MAXP	5
#define	UART_RX_HORA	6
#define	UART_RX_VARM	7
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
//	@funcdef	Estas son las funciones correspondientes al protocolo UART0.                                         //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
void	__configuraUART0__ (	void	);
uint8_tprocesarComando(	char *	);
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
//	@end	ENDFILE.                                                                                                //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
```

UART0.c

```
/**-------------------------------------------------------------------------------------------------------------------//
//      @filename     UART0.c                                                                                        //
//      @version      2.01                                                                                           //
//      @author       Alberto Palomo Alonso                                                                          //
//                                                                                                                   //
//      @brief      Este es el programa que recoge la transmisión por UART0.                                         //
//                                                                                                                   //
//      @category     Opcional.                                                                                      //
//                                                                                                                   //
//      @map          @include                                                                                       //
//                    @variables                                                                                     //
//                    @function                                                                                      //
//                    @end                                                                                           //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
                                                                                                                     //
//      @include      Includes pertenecientes a la transmisión asíncrona.                                            //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
#ifndef UART0
#define     UART0
#include    "UART0.h"
#endif
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
                                                                                                                     //
//      @variables          Variables del fichero.                                                                   //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
//  Variables globales y externas.
char UART0_BUFFER_TX[CADMAX + 1];
extern char bufferx[30];
extern misDatos_t * DATOS;
extern uint8_t Clock[23];
uint8_t EstadoUART0 = UART_TX;
extern modificables_t MODIFICABLES;
uint8_t Inmortal = 0;
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
                                                                                                                     //
//      @function     __configuraUART0__()                                                                           //
//                                                                                                                   //
//      @brief      Esta función es la que configura el UART0 para transmisión y recepción.                          //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
void    __configuraUART0__(void    )                //  Configurado a 9600 baudios.
{
    uart0_init(9600);
    tx_cadena_UART0(    "Hola.\n"   );
}
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
                                                                                                                     //
//      @function     procesarComando()                                                                              //
//                                                                                                                   //
//      @brief      Esta función manda el UART0_BUFFER_TX a la salida TX del UART0.                                  //
//                                                                                                                   //
//      @input          char * Buff:    El buffer donde está contenido el comando.                                   //
//                                                                                                                   //
//      @ret            Devuelve 1 si ha sido exitoso y 0 si no se ha obtenido el comando.                           //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
uint8_tprocesarComando(    char    *    Buff    )
```

```
{
    uint8_t     retval = 0;
    switch( EstadoUART0 )
    {
        case UART_TX:
            /**SECCIÓN PARA LOS COMANDOS DE TIPO 0:  ABOUT*/
            if  (   !strcmp(    Buff    ,   COM0 )   )
            {
                retval = 1;
                strcpy( UART0_BUFFER_TX    ,    "\n Autor: \t Alberto Palomo Alonso \n Version: \t 2.1.0 \n Sistemas Electronicos
Digitales Avanzados \t UAH \n"   );
            }
            /**SECCIÓN PARA LOS COMANDOS DE TIPO 1:  GIVE*/
            if  (   !strcmp(    Buff    ,   COM10)   )
            {
                if ( !Inmortal )
                {
                    strcpy  (   UART0_BUFFER_TX    ,    "\nSUGAR n.n\n");
                    Inmortal = 1;
                    retval = 1;
                }
            }
            if  (   !strcmp(    Buff    ,   COM11)   )
            {
                sprintf (   UART0_BUFFER_TX    ,    "\nIP: %d.%d.%d.%d \n", __IP1B, __IP2B, __IP3B, __IP4B);
                retval = 1;
            }
            if  (   !strcmp(    Buff    ,   COM12)   )
            {
                sprintf (   UART0_BUFFER_TX    ,    "\nTEMPERATURA: %f ºC\n", DATOS->Temperatura);

                retval = 1;
            }
            if  (   !strcmp(    Buff    ,   COM13)   )
            {
                sprintf (   UART0_BUFFER_TX    ,    "\nPRESION: %f mBar.\n", DATOS->Presion);
                retval = 1;
            }
            if  (   !strcmp(    Buff    ,   COM14)   )
            {
                sprintf (   UART0_BUFFER_TX    ,    "\nVELOCIDAD DEL VIENTO: %f m./s.\n", DATOS->VelViento);
                retval = 1;
            }
            if  (   !strcmp(    Buff    ,   COM15)   )
            {
                sprintf (   UART0_BUFFER_TX    ,    "\nX: NA     \nY: NA    \nZ: %f m.\n", DATOS->Lugar.Altura);
                retval = 1;
            }
            if  (   !strcmp(    Buff    ,   COM16)   )
            {
                sprintf (   UART0_BUFFER_TX    ,    "\nINDICE UV: %f\n", DATOS->IndiceUV);
                retval = 1;
            }
            if  (   !strcmp(    Buff    ,   COM17)   )
            {
                strcpy  (   UART0_BUFFER_TX    ,    (const char *)Clock);
                retval = 1;
            }
            if  (   !strcmp(    Buff    ,   COM18)   )
            {
                sprintf (   UART0_BUFFER_TX    ,    "\nHUMEDAD: %f \n", 0.01*DATOS->Humedad);
                retval = 1;
            }
            if  (   !strcmp(    Buff    ,   COM19)   )
            {
                sprintf (   UART0_BUFFER_TX    ,    "\nBRILLO: %f LUX.\n", DATOS->Brillo);
```

```
            retval = 1;
        }
        /**SECCIÓN PARA LOS COMANDOS DE TIPO 3:  KILL*/
        if  (   !strcmp(   Buff    ,   COM3 )   )
        {
            while ( !Inmortal );
            strcpy( UART0_BUFFER_TX   ,   "Demasiado dulce como para matarlo, mejor para otra ocasion...\n");
            retval = 1;
        }
        /**SECCIÓN PARA LOS COMANDOS DE TIPO 4:  HELP   */
        if  (   !strcmp(   Buff    ,   COM4 )   )
        {
            strcpy( UART0_BUFFER_TX   ,   "Informacion:\n\n ABOUT: Muestra info. del sistema.\n GIVE: Proporciona el dato
deseado.\n KILL: Cuelga el programa.\n SET: Modifica variables.\n"    );
            retval = 1;
        }
        if  (   !strcmp(   Buff    ,   COM41    )   )
        {
            strcpy( UART0_BUFFER_TX   ,   "\nGIVE + [IP, TEMPERATURA, PRESION, BRILLO, LUGAR, VIENTO, INDICEUV, HORA,
HUMEDAD]\n"    );
            retval = 1;
        }
        if  (   !strcmp(   Buff    ,   COM42    )   )
        {
            strcpy( UART0_BUFFER_TX   ,   "\nSET + [BRILLO, HORA, MIN TEMP, MAX TEMP, MIN PRES, MAX PRES, TEMPERATURA,
PRESION]\n"  );
            retval = 1;
        }
        /**CONTROL DE ERROR:  */
        if ( !retval )
        {
            strcpy( UART0_BUFFER_TX,   "Error: comando no definido, escriba 'HELP' para ver la lista.\n");
        }
        /** SECCIÓN PARA LOS COMANDOS DE TIPO 2: SET*/
        if  (   !strcmp(   Buff    ,   COM20    )   )
        {
            strcpy( UART0_BUFFER_TX   ,   "Introduzca los segundos de brillo: \n"   );
            retval = 1;
            EstadoUART0 = UART_RX_BRILLO;
        }
        if  (   !strcmp(   Buff    ,   COM21    )   )
        {
            strcpy( UART0_BUFFER_TX   ,   "Introduzca la fecha separada por espacios: \n");
            retval = 1;
            EstadoUART0 = UART_RX_HORA;
        }
        if  (   !strcmp(   Buff    ,   COM22    )   )
        {
            strcpy( UART0_BUFFER_TX   ,   "Introduzca el valor minimo de temperatura: \n"    );
            retval = 1;
            EstadoUART0 = UART_RX_MINT;
        }
        if  (   !strcmp(   Buff    ,   COM23    )   )
        {
            strcpy( UART0_BUFFER_TX   ,   "Introduzca el valor maximo de temperatura: \n"    );
            retval = 1;
            EstadoUART0 = UART_RX_MAXT;
        }
        if  (   !strcmp(   Buff    ,   COM24    )   )
        {
            strcpy( UART0_BUFFER_TX   ,   "Introduzca el valor minimo de presion: \n" );
            retval = 1;
            EstadoUART0 = UART_RX_MINP;
        }
        if  (   !strcmp(   Buff    ,   COM25    )   )
        {
```

```
            strcpy( UART0_BUFFER_TX   ,   "Introducza el valor maximo de presion: \n");
            retval = 1;
            EstadoUART0 = UART_RX_MAXP;
        }
        if  (   !strcmp(   Buff   ,   COM27   )   )
        {
            strcpy( UART0_BUFFER_TX   ,   "Ahora medimos presion... \n"   );
            retval = 1;
            MODIFICABLES.Var_medida = 1;
        }
        if  (   !strcmp(   Buff   ,   COM26   )   )
        {
            strcpy( UART0_BUFFER_TX   ,   "Ahora medimos temperatura... \n" );
            retval = 1;
            MODIFICABLES.Var_medida = 0;
        }
        break;

    case UART_RX_BRILLO:
        sscanf( bufferx,   "%d"   ,   &MODIFICABLES.TiempoBrillo);
        strcpy( UART0_BUFFER_TX       ,   "Tiempo de hold cambiado.\n"   );
        EstadoUART0 = UART_TX;
        break;
    case UART_RX_MINT:
        sscanf( bufferx,   "%f"   ,   &MODIFICABLES.Min_servo_t);
        strcpy( UART0_BUFFER_TX       ,   "Cota minima de temperatura cambiada.\n"    );
        EstadoUART0 = UART_TX;
        break;
    case UART_RX_MAXT:
        sscanf( bufferx,   "%f"   ,   &MODIFICABLES.Max_servo_t);
        strcpy( UART0_BUFFER_TX       ,   "Cota maxima de temperatura cambiada.\n"    );
        EstadoUART0 = UART_TX;
        break;
    case UART_RX_MINP:
        sscanf( bufferx,   "%f"   ,   &MODIFICABLES.Min_servo_p);
        strcpy( UART0_BUFFER_TX       ,   "Cota minima de presion cambiada.\n"  );
        EstadoUART0 = UART_TX;
        break;
    case UART_RX_MAXP:
        sscanf( bufferx,   "%f"   ,   &MODIFICABLES.Max_servo_p);
        strcpy( UART0_BUFFER_TX       ,   "Cota maxima de presion cambiada.\n" );
        EstadoUART0 = UART_TX;
        break;
    case UART_RX_HORA:
        sscanf( bufferx,   "%d %d %d %d %d %d"   ,   (int *)&LPC_RTC->DOM , (int *)&LPC_RTC->MONTH, (int *)&LPC_RTC->YEAR,
(int *)&LPC_RTC->HOUR, (int *)&LPC_RTC->MIN, (int *)&LPC_RTC->SEC);
        strcpy( UART0_BUFFER_TX       ,   "Hora cambiada...\n" );
        EstadoUART0 = UART_TX;
        break;

  }
  tx_cadena_UART0(UART0_BUFFER_TX);
  /**ZONA RETURN*/
  return retval;
}
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
                                                                                                               //
//      @end      ENDFILE.                                                                                     //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
```

HTTP_SOURCE.h

```
/**-------------------------------------------------------------------------------------------------------------//
//      @filename     HTTP_SOURCE.h                                                                           //
//      @version      0.00                                                                                    //
//      @author       Alberto Palomo Alonso                                                                   //
//                                                                                                            //
//      @brief      Cabecera que configura la página WEB.                                                     //
//                                                                                                            //
//      @category     Opcional.                                                                               //
//                                                                                                            //
//      @map          @include                                                                                //
//                  @funcdef                                                                                  //
//                  @end                                                                                      //
//                                                                                                            //
//                                                                                                            //
//-------------------------------------------------------------------------------------------------------------//
//                                                                                                            //
                                                                                                              //
//      @include      Estos son los archivos utilizados en el código de configuración.                        //
//                                                                                                            //
//-------------------------------------------------------------------------------------------------------------**/
#ifndef NETCONFIG
#define     NETCONFIG
#include    <Net_Config.h>
#endif
#ifndef STDIO
#define     STDIO
#include    <stdio.h>
#endif
#ifndef LPC17XX
#define     LPC17XX
#include <LPC17XX.H>
#endif
#ifndef RTL
#define RTL
#include <RTL.h>
#endif

void    __configuraWEB__    (   void    );
void        __mantenerTCP__     (   void    );


/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                            //
                                                                                                              //
//      @end        ENDFILE.                                                                                  //
//                                                                                                            //
//-------------------------------------------------------------------------------------------------------------**/
```

HTTP_SOURCE.c

```
/**-------------------------------------------------------------------------------------------------------------//
//      @filename     HTTP_SOURCE.c                                                                            //
//      @version      0.00                                                                                   //
//      @author       Alberto Palomo Alonso                                                                    //
//                                                                                                           //
//      @brief      Código que configura la página WEB.                                                        //
//                                                                                                           //
//      @category     Opcional.                                                                              //
//                                                                                                           //
//      @map          @include                                                                               //
//                  @funcion                                                                                 //
//                  @end                                                                                     //
//                                                                                                           //
//                                                                                                           //
//-------------------------------------------------------------------------------------------------------------//
//                                                                                                           //
                                                                                                             //
//      @include      Estos son los archivos utilizados en el código de configuración.                       //
//                                                                                                           //
//-------------------------------------------------------------------------------------------------------------**/
#ifndef HTTPSOURCE
#define     HTTPSOURCE
#include    "HTTP_SOURCE.h"
#endif

void __configuraWEB__()
{
    init_TcpNet(); //  Inicializamos TcpNet(RTL.h).
}

void __mantenerTCP__()
{
    main_TcpNet();
}
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                           //
                                                                                                             //
//      @end      ENDFILE.                                                                                   //
//                                                                                                           //
//-------------------------------------------------------------------------------------------------------------**/
```

NET_CONFIG.h

```
/*----------------------------------------------------------------------------
 *      RL-ARM - TCPnet
 *----------------------------------------------------------------------------
 *      Name:        NET_CONFIG.C
 *      Purpose: Configuration of RL TCPnet by user
 *      Rev.:        V4.72
 *----------------------------------------------------------------------------
 *      This code is part of the RealView Run-Time Library.
 *      Copyright (c) 2004-2013 KEIL - An ARM Company. All rights reserved.
 *---------------------------------------------------------------------------*/

#include <Net_Config.h>
#include    "miGlobal.h"

//-------- <<< Use Configuration Wizard in Context Menu >>> ------------------
//
// <h>System Definitions
// =====================
// <i> Global TCPnet System definitions
//     <s.15>Local Host Name
//     <i> This is the name under which embedded host
//     <i> can be accessed on a local area network.
//     <i> Default: "mcb2300"
#define LHOST_NAME          "MiniDK2"

//     <o>Memory Pool size <1536-262144:4><#/4>
//     <i> This is the size of a memory pool in bytes. Buffers for
//     <i> TCPnet packets are allocated from this memory pool.
//     <i> Default: 8000 bytes
#define MEM_SIZE            3000

//     <o>Tick Timer interval <10=> 10 ms <20=> 20 ms <25=> 25 ms
//                            <40=> 40 ms <50=> 50 ms <100=> 100 ms
//                            <200=> 200 ms
//     <i> System Tick Timer interval for software timers
//     <i> Default: 100 ms
#define TICK_INTERVAL    100

// </h>
// <e>Ethernet Network Interface
// =============================
// <i> Enable or disable Ethernet Network Interface
#define ETH_ENABLE          1

//     <h>MAC Address
//     ==============
//     <i> Local Ethernet MAC Address
//     <i> Value FF:FF:FF:FF:FF:FF is not allowed.
//     <i> It is an ethernet Broadcast MAC address.
//       <o>Address byte 1 <0x00-0xff:2>
//       <i> LSB is an ethernet Multicast bit.
//       <i> Must be 0 for local MAC address.
//       <i> Default: 0x00
#define _MAC1               0xE0

//       <o>Address byte 2 <0x00-0xff>
//       <i> Default: 0x30
#define _MAC2               0xF8

//       <o>Address byte 3 <0x00-0xff>
//       <i> Default: 0x6C
#define _MAC3               0x46

//       <o>Address byte 4 <0x00-0xff>
```

```
//      <i> Default: 0x00
#define _MAC4               0x35

//      <o>Address byte 5 <0x00-0xff>
//      <i> Default: 0x00
#define _MAC5               0x45

//      <o>Address byte 6 <0x00-0xff>
//      <i> Default: 0x01
#define _MAC6               0xBC

//    </h>
//    <h>IP Address
//    =============
//    <i> Local Static IP Address
//    <i> Value 255.255.255.255 is not allowed.
//    <i> It is a Broadcast IP address.
//      <o>Address byte 1 <0-255>
//      <i> Default: 192
#define _IP1                __IP1B

//      <o>Address byte 2 <0-255>
//      <i> Default: 168
#define _IP2                __IP2B

//      <o>Address byte 3 <0-255>
//      <i> Default: 0
#define _IP3                __IP3B

//      <o>Address byte 4 <0-255>
//      <i> Default: 100
#define _IP4                __IP4B

//    </h>
//    <h>Subnet mask
//    ==============
//    <i> Local Subnet mask
//      <o>Mask byte 1 <0-255>
//      <i> Default: 255
#define _MSK1               __MASK1B

//      <o>Mask byte 2 <0-255>
//      <i> Default: 255
#define _MSK2               __MASK2B

//      <o>Mask byte 3 <0-255>
//      <i> Default: 255
#define _MSK3               __MASK3B

//      <o>Mask byte 4 <0-255>
//      <i> Default: 0
#define _MSK4               __MASK4B

//    </h>
//    <h>Default Gateway
//    ==================
//    <i> Default Gateway IP Address
//      <o>Address byte 1 <0-255>
//      <i> Default: 192
#define _GW1                __GW1B

//      <o>Address byte 2 <0-255>
//      <i> Default: 168
#define _GW2                __GW2B

//      <o>Address byte 3 <0-255>
```

```
//     <i> Default: 0
#define _GW3                __GW3B

//     <o>Address byte 4 <0-255>
//     <i> Default: 254
#define _GW4                __GW4B

//   </h>
//   <h>Primary DNS Server
//   ======================
//   <i> Primary DNS Server IP Address
//     <o>Address byte 1 <0-255>
//     <i> Default: 194
#define _pDNS1              192

//     <o>Address byte 2 <0-255>
//     <i> Default: 25
#define _pDNS2              168

//     <o>Address byte 3 <0-255>
//     <i> Default: 2
#define _pDNS3              5

//     <o>Address byte 4 <0-255>
//     <i> Default: 129
#define _pDNS4              1

//   </h>
//   <h>Secondary DNS Server
//   ========================
//   <i> Secondary DNS Server IP Address
//     <o>Address byte 1 <0-255>
//     <i> Default: 194
#define _sDNS1              194

//     <o>Address byte 2 <0-255>
//     <i> Default: 25
#define _sDNS2              25

//     <o>Address byte 3 <0-255>
//     <i> Default: 2
#define _sDNS3              2

//     <o>Address byte 4 <0-255>
//     <i> Default: 130
#define _sDNS4              130

//   </h>
//   <h>ARP Definitions
//   ==================
//   <i> Address Resolution Protocol Definitions
//     <o>Cache Table size <5-100>
//     <i> Number of cached hardware/IP addresses
//     <i> Default: 10
#define ARP_TABSIZE       10

//     <o>Cache Timeout in seconds <5-255>
//     <i> A timeout for a cached hardware/IP addresses
//     <i> Default: 150
#define ARP_TIMEOUT       150

//     <o>Number of Retries <0-20>
//     <i> Number of Retries to resolve an IP address
//     <i> before ARP module gives up
//     <i> Default: 4
#define ARP_MAXRETRY      4
```

```
//         <o>Resend Timeout in seconds <1-10>
//         <i> A timeout to resend the ARP Request
//         <i> Default: 2
#define ARP_RESEND         2

//         <q>Send Notification on Address changes
//         <i> When this option is enabled, the embedded host
//         <i> will send a Gratuitous ARP notification at startup,
//         <i> or when the device IP address has changed.
//         <i> Default: Disabled
#define ARP_NOTIFY         0

//     </h>
//     <e>IGMP Group Management
//     ========================
//     <i> Enable or disable Internet Group Management Protocol
#define IGMP_ENABLE        0

//         <o>Membership Table size <2-50>
//         <i> Number of Groups this host can join
//         <i> Default: 5
#define IGMP_TABSIZE       5

//     </e>
//     <q>NetBIOS Name Service
//     =======================
//     <i> When this option is enabled, the embedded host can be
//     <i> accessed by his name on the local LAN using NBNS protocol.
#define NBNS_ENABLE        0

//     <e>Dynamic Host Configuration
//     =============================
//     <i> When this option is enabled, local IP address, Net Mask
//     <i> and Default Gateway are obtained automatically from
//     <i> the DHCP Server on local LAN.
#define DHCP_ENABLE        0

//         <s.40>Vendor Class Identifier
//         <i> This value is optional. If specified, it is added
//         <i> to DHCP request message, identifying vendor type.
//         <i> Default: ""
#define DHCP_VCID          ""

//         <q>Bootfile Name
//         <i> This value is optional. If enabled, the Bootfile Name
//         <i> (option 67) is also requested from DHCP server.
//         <i> Default: disabled
#define DHCP_BOOTF         0

//         <q>NTP Servers
//         <i> This value is optional. If enabled, a list of NTP Servers
//         <i> (option 42) is also requested from DHCP server.
//         <i> Default: disabled
#define DHCP_NTPSRV        0

//     </e>
// </e>

// <e>PPP Network Interface
// ========================
// <i> Enable or disable PPP Network Interface
#define PPP_ENABLE         0

//     <h>IP Address
//     =============
```

```
//     <i> Local Static IP Address
//         <o>Address byte 1 <0-255>
//         <i> Default: 192
#define _IP1P               192

//         <o>Address byte 2 <0-255>
//         <i> Default: 168
#define _IP2P               168

//         <o>Address byte 3 <0-255>
//         <i> Default: 125
#define _IP3P               125

//         <o>Address byte 4 <0-255>
//         <i> Default: 1
#define _IP4P               1

//     </h>
//     <h>Subnet mask
//     ==============
//     <i> Local Subnet mask
//         <o>Mask byte 1 <0-255>
//         <i> Default: 255
#define _MSK1P              255

//         <o>Mask byte 2 <0-255>
//         <i> Default: 255
#define _MSK2P              255

//         <o>Mask byte 3 <0-255>
//         <i> Default: 255
#define _MSK3P              255

//         <o>Mask byte 4 <0-255>
//         <i> Default: 0
#define _MSK4P              0

//     </h>
//     <h>Primary DNS Server
//     =====================
//     <i> Primary DNS Server IP Address
//         <o>Address byte 1 <0-255>
//         <i> Default: 194
#define _pDNS1P             194

//         <o>Address byte 2 <0-255>
//         <i> Default: 25
#define _pDNS2P             25

//         <o>Address byte 3 <0-255>
//         <i> Default: 2
#define _pDNS3P             2

//         <o>Address byte 4 <0-255>
//         <i> Default: 129
#define _pDNS4P             129

//     </h>
//     <h>Secondary DNS Server
//     =======================
//     <i> Secondary DNS Server IP Address
//         <o>Address byte 1 <0-255>
//         <i> Default: 194
#define _sDNS1P             194

//         <o>Address byte 2 <0-255>
```

```
//      <i> Default: 25
#define _sDNS2P             25

//      <o>Address byte 3 <0-255>
//      <i> Default: 2
#define _sDNS3P             2

//      <o>Address byte 4 <0-255>
//      <i> Default: 130
#define _sDNS4P             130

//    </h>
//    <e>Logon Authentication
//    ======================
//    <i> Enable or disable user authentication
#define PPP_AUTHEN          1

//      <q>Unsecured password (PAP)
//      <i>Allow or use Password Authentication Protocol.
#define PPP_PAPEN           1

//      <q>Secured password (CHAP-MD5)
//      <i>Request or use Challenge Handshake Authentication
//      <i>Protocol with MD5 digest algorithm.
#define PPP_CHAPEN          1

//    </e>
//    <q>Obtain Client IP address automatically
//    ========================================
//    <i> This option only applies when PPP Dial-up is used to dial
//    <i> to remote PPP Server. If checked, network connection
//    <i> dynamically obtains an IP address from remote PPP Server.
#define PPP_GETIP           1

//    <q>Use Default Gateway on remote Network
//    ======================================
//    <i> This option only applies when both Ethernet and PPP Dial-up
//    <i> are used. If checked, data that cannot be sent to local LAN
//    <i> is forwarded to Dial-up network instead.
#define PPP_DEFGW           1

//    <o>Async Control Character Map <0x0-0xffffffff>
//    <i> A bit-map of control characters 0-31, which are
//    <i> transmitted escaped as a 2 byte sequence.
//    <i> For XON/XOFF set this value to: 0x000A 0000
//    <i> Default: 0x00000000
#define PPP_ACCM            0x00000000

//    <o>LCP Echo Interval in seconds <0-3600>
//    <i> If no frames are received within this interval, PPP sends an
//    <i> Echo Request and expects an Echo Response from the peer.
//    <i> If the response is not received, the link is terminated.
//    <i> A value of 0 disables the LCP Echo test.
//    <i> Default: 30
#define PPP_ECHOTOUT        30

//    <o>Number of Retries <0-20>
//    <i> How many times PPP will try to retransmit data
//    <i> before giving up. Increase this value for links
//    <i> with low baud rates or high latency.
//    <i> Default: 3
#define PPP_MAXRETRY        3

//    <o>Retry Timeout in seconds <1-10>
//    <i> If no response received within this time frame,
//    <i> PPP module will try to resend the data again.
```

```
//     <i> Default: 2
#define PPP_RETRYTOUT    2

// </e>
// <e>SLIP Network Interface
// =========================
// <i> Enable or disable SLIP Network Interface
#define SLIP_ENABLE      0

//     <h>IP Address
//     =============
//     <i> Local Static IP Address
//         <o>Address byte 1 <0-255>
//         <i> Default: 192
#define _IP1S                192

//         <o>Address byte 2 <0-255>
//         <i> Default: 168
#define _IP2S                168

//         <o>Address byte 3 <0-255>
//         <i> Default: 225
#define _IP3S                225

//         <o>Address byte 4 <0-255>
//         <i> Default: 1
#define _IP4S                1

//     </h>
//     <h>Subnet mask
//     ==============
//     <i> Local Subnet mask
//         <o>Mask byte 1 <0-255>
//         <i> Default: 255
#define _MSK1S               255

//         <o>Mask byte 2 <0-255>
//         <i> Default: 255
#define _MSK2S               255

//         <o>Mask byte 3 <0-255>
//         <i> Default: 255
#define _MSK3S               255

//         <o>Mask byte 4 <0-255>
//         <i> Default: 0
#define _MSK4S               0

//     </h>
//     <h>Primary DNS Server
//     =====================
//     <i> Primary DNS Server IP Address
//         <o>Address byte 1 <0-255>
//         <i> Default: 194
#define _pDNS1S              194

//         <o>Address byte 2 <0-255>
//         <i> Default: 25
#define _pDNS2S              25

//         <o>Address byte 3 <0-255>
//         <i> Default: 2
#define _pDNS3S              2

//         <o>Address byte 4 <0-255>
//         <i> Default: 129
```

```
#define _pDNS4S             129

//     </h>
//     <h>Secondary DNS Server
//     ======================
//     <i> Secondary DNS Server IP Address
//       <o>Address byte 1 <0-255>
//       <i> Default: 194
#define _sDNS1S             194

//       <o>Address byte 2 <0-255>
//       <i> Default: 25
#define _sDNS2S             25

//       <o>Address byte 3 <0-255>
//       <i> Default: 2
#define _sDNS3S             2

//       <o>Address byte 4 <0-255>
//       <i> Default: 130
#define _sDNS4S             130

//     </h>
//     <q>Use Default Gateway on remote Network
//     ========================================
//     <i> This option only applies when both Ethernet and SLIP Dial-up
//     <i> are used. If checked, data that cannot be sent to local LAN
//     <i> is forwarded to Dial-up network instead.
#define SLIP_DEFGW          1

// </e>
// <e>UDP Sockets
// ==============
// <i> Enable or disable UDP Sockets
#define UDP_ENABLE          1

//     <o>Number of UDP Sockets <1-20>
//     <i> Number of available UDP sockets
//     <i> Default: 5
#define UDP_NUMSOCKS        2

// </e>
// <e>TCP Sockets
// ==============
// <i> Enable or disable TCP Sockets
#define TCP_ENABLE          1

//     <o>Number of TCP Sockets <1-20>
//     <i> Number of available TCP sockets
//     <i> Default: 5
#define TCP_NUMSOCKS        12

//     <o>Number of Retries <0-20>
//     <i> How many times TCP module will try to retransmit data
//     <i> before giving up. Increase this value for high-latency
//     <i> and low_throughput networks.
//     <i> Default: 5
#define TCP_MAXRETRY        5

//     <o>Retry Timeout in seconds <1-10>
//     <i> If data frame not acknowledged within this time frame,
//     <i> TCP module will try to resend the data again.
//     <i> Default: 4
#define TCP_RETRYTOUT       4

//     <o>Default Connect Timeout in seconds <1-600>
```

```
//     <i> Default TCP Socket Keep Alive timeout. When it expires
//     <i> with no TCP data frame send, TCP Connection is closed.
//     <i> Default: 120
#define TCP_DEFTOUT       120

//     <o>Maximum Segment Size <536-1460>
//     <i> The Maximum Segment Size specifies the maximum
//     <i> number of bytes in the TCP segment's Data field.
//     <i> Default: 1460
#define TCP_MAXSEGSZ      1460

//     <o>Receive Window Size <536-65535>
//     <i> Receive Window Size specifies the size of data,
//     <i> that the socket is able to buffer in flow-control mode.
//     <i> Default: 4380
#define TCP_RECWINSZ      4380

/* TCP fixed timeouts */
#define TCP_INIT_RETRY_TOUT 1         /* TCP initial Retransmit period in sec.      */
#define TCP_SYN_RETRY_TOUT    2         /* TCP SYN frame retransmit period in sec. */
#define TCP_CONRETRY              7         /* Number of retries to establish a conn.    */

// </e>
// <e>HTTP Server
// ==============
// <i> Enable or disable HTTP Server
#define HTTP_ENABLE       1

//     <o>Number of HTTP Sessions <1-10>
//     <i> Number of simultaneously active HTTP Sessions.
//     <i> Default: 3
#define HTTP_NUMSESS      6

//     <o>Port Number <1-65535>
//     <i> Listening port number.
//     <i> Default: 80
#define HTTP_PORTNUM      80

//     <s.50>Server-Id header
//     <i> This value is optional. If specified, it overrides
//     <i> the default HTTP Server header from the library.
//     <i> Default: ""
#define HTTP_SRVID          ""

//     <e>Enable User Authentication
//       <i> When enabled, the user will have to authenticate
//       <i> himself by username and password before accessing
//       <i> any page on this Embedded WEB server.
#define HTTP_ENAUTH       1

//       <s.20>Authentication Realm
//       <i> Default: "Embedded WEB Server"
#define HTTP_AUTHREALM "Embedded WEB Server"

//       <s.15>Authentication Username
//       <i> Default: "admin"
#define HTTP_AUTHUSER    "user"

//       <s.15>Authentication Password
//       <i> Default: ""
#define HTTP_AUTHPASSW "Alver"

//     </e>
// </e>
// <e>Telnet Server
// ================
```

```
// <i> Enable or disable Telnet Server
#define TNET_ENABLE       0

//     <o>Number of Telnet Connections <1-10>
//     <i> Number of simultaneously active Telnet Connections.
//     <i> Default: 1
#define TNET_NUMSESS      2

//     <o>Port Number <1-65535>
//     <i> Listening port number.
//     <i> Default: 23
#define TNET_PORTNUM      23

//     <o>Idle Connection Timeout in seconds <0-3600>
//     <i> When timeout expires, the connection is closed.
//     <i> A value of 0 disables disconnection on timeout.
//     <i> Default: 120
#define TNET_IDLETOUT     120

//     <q>Disable Echo
//     <i> When disabled, the server will not echo
//     <i> characters it receives.
//     <i> Default: Not disabled
#define TNET_NOECHO       0

//     <e>Enable User Authentication
//     <i> When enabled, the user will have to authenticate
//     <i> himself by username and password before access
//     <i> to the system is allowed.
#define TNET_ENAUTH       1

//       <s.15>Authentication Username
//       <i> Default: "admin"
#define TNET_AUTHUSER     "admin"

//       <s.15>Authentication Password
//       <i> Default: ""
#define TNET_AUTHPASSW ""

//     </e>
// </e>
// <e>TFTP Server
// ==============
// <i> Enable or disable TFTP Server
#define TFTP_ENABLE       0

//     <o>Number of TFTP Sessions <1-10>
//     <i> Number of simultaneously active TFTP Sessions
//     <i> Default: 1
#define TFTP_NUMSESS      1

//     <o>Port Number <1-65535>
//     <i> Listening port number.
//     <i> Default: 69
#define TFTP_PORTNUM      69

//     <q>Enable Firewall Support
//     <i> Use the same Port Number to receive
//     <i> requests and send answers to clients.
//     <i> Default: Not Enabled
#define TFTP_ENFWALL      0

//     <o>Inactive Session Timeout in seconds <5-120>
//     <i> When timeout expires TFTP Session is closed.
//     <i> Default: 15
#define TFTP_DEFTOUT      15
```

```
//     <o>Number of Retries <1-10>
//     <i> How many times TFTP Server will try to
//     <i> retransmit the data before giving up.
//     <i> Default: 4
#define TFTP_MAXRETRY    4

// </e>
// <e>TFTP Client
// ==============
// <i> Enable or disable TFTP Client
#define TFTPC_ENABLE     0

//     <o>Block Size <128=>128     <256=>256     <512=>512
//                          <1024=>1024 <1428=>1428
//     <i> Size of transfer block in bytes.
//     <i> Default: 512
#define TFTPC_BLOCKSZ    512

//     <o>Number of Retries <1-10>
//     <i> How many times TFTP Client will try to
//     <i> retransmit the data before giving up.
//     <i> Default: 4
#define TFTPC_MAXRETRY 4

//     <o>Retry Timeout <2=>200 ms <5=>500 ms <10=>1 sec
//                              <20=>2 sec <50=>5 sec <100=>10 sec
//     <i> If data frame not acknowledged within this time frame,
//     <i> TFTP Client will try to resend the data again.
//     <i> Default: 500 ms
#define TFTPC_RETRYTO    5

// </e>
// <e>FTP Server
// ==============
// <i> Enable or disable FTP Server
#define FTP_ENABLE         0

//     <o>Number of FTP Sessions <1-10>
//     <i> Number of simultaneously active FTP Sessions
//     <i> Default: 1
#define FTP_NUMSESS        3

//     <o>Port Number <1-65535>
//     <i> Listening port number.
//     <i> Default: 21
#define FTP_PORTNUM        21

//     <s.50>Welcome Message
//     <i> This value is optional. If specified,
//     <i> it overrides the default welcome message.
//     <i> Default: ""
#define FTP_WELMSG         ""

//     <o>Idle Session Timeout in seconds <0-3600>
//     <i> When timeout expires, the connection is closed.
//     <i> A value of 0 disables disconnection on timeout.
//     <i> Default: 120
#define FTP_IDLETOUT       120

//     <e>Enable User Authentication
//     <i> When enabled, the user will have to authenticate
//     <i> himself by username and password before access
//     <i> to the system is allowed.
#define FTP_ENAUTH         1
```

```
//         <s.15>Authentication Username
//         <i> Default: "admin"
#define FTP_AUTHUSER      "admin"

//         <s.15>Authentication Password
//         <i> Default: ""
#define FTP_AUTHPASSW     ""

//     </e>
// </e>
// <e>FTP Client
// =============
// <i> Enable or disable FTP Client
#define FTPC_ENABLE       0

//         <o>Response Timeout in seconds <1-120>
//         <i> This is a time for FTP Client to wait for a response from
//         <i> the Server. If timeout expires, Client aborts operation.
//         <i> Default: 10
#define FTPC_DEFTOUT      10

//         <q>Passive mode (PASV)
//         <i> The client initiates a data connection to the server.
//         <i> Default: Not passive (Active)
#define FTPC_PASVMODE     0

// </e>
// <e>DNS Client
// =============
// <i> Enable or disable DNS Client
#define DNS_ENABLE        0

//         <o>Cache Table size <5-100>
//         <i> Number of cached DNS host names/IP addresses
//         <i> Default: 20
#define DNS_TABSIZE       20

// </e>
// <e>SMTP Client
// ==============
// <i> Enable or disable SMTP Client
#define SMTP_ENABLE       0

//         <o>Response Timeout in seconds <5-120>
//         <i> This is a time for SMTP Client to wait for a response from
//         <i> SMTP Server. If timeout expires, Client aborts operation.
//         <i> Default: 20
#define SMTP_DEFTOUT      20

// </e>
// <e>SNMP Agent
// =============
// <i> Enable or disable SNMP Agent
#define SNMP_ENABLE       0

//     <s.15>Community Name
//     <i> Defines where an SNMP message is destined for.
//     <i> Default: "public"
#define SNMP_COMMUNITY "public"

//     <o>Port Number <1-65535>
//     <i> Listening port number.
//     <i> Default: 161
#define SNMP_PORTNUM      161

//     <o>Trap Port Number <1-65535>
```

```
//     <i> Port number for Trap operations.
//     <i> Default: 162
#define SNMP_TRAPPORT    162

//     <h>Trap Server
//     ==============
//     <i> Trap Server IP Address
//     <o>Address byte 1 <0-255>
//     <i> Default: 192
#define SNMP_TRAPIP1     192

//     <o>Address byte 2 <0-255>
//     <i> Default: 168
#define SNMP_TRAPIP2     168

//     <o>Address byte 3 <0-255>
//     <i> Default: 0
#define SNMP_TRAPIP3     0

//     <o>Address byte 4 <0-255>
//     <i> Default: 100
#define SNMP_TRAPIP4     1

//     </h>
// </e>
// <e>SNTP Client
// ==============
// <i> Enable or disable SNTP Client
#define SNTP_ENABLE      0

//     <q>Broadcast Mode
//     =================
//     <i> Enable this option, if you have NTP/SNTP server
//     <i> on LAN, which is broadcasting NTP time messages.
//     <i> Disable this option to access public NTP server.
//     <i> Default: disabled
#define SNTP_BCMODE      0

//     <h>NTP Server
//     =============
//     <i> Server IP Address
//     <o>Address byte 1 <0-255>
//     <i> Default: 217
#define SNTP_SRVIP1      217

//     <o>Address byte 2 <0-255>
//     <i> Default: 79
#define SNTP_SRVIP2      79

//     <o>Address byte 3 <0-255>
//     <i> Default: 179
#define SNTP_SRVIP3      179

//     <o>Address byte 4 <0-255>
//     <i> Default: 106
#define SNTP_SRVIP4      106

//     </h>
// </e>
// <e>BSD Socket Interface
// =======================
// <i> Enable or disable Berkeley Socket Programming Interface
#define BSD_ENABLE       0

//     <o>Number of BSD Sockets <1-20>
//     <i> Number of available Berkeley Sockets
```

```
//     <i> Default: 2
#define BSD_NUMSOCKS     2

//     <o>Number of Streaming Server Sockets <0-20>
//     <i> Defines a number of Streaming (TCP) Server sockets,
//     <i> that listen for an incoming connection from the client.
//     <i> Default: 1
#define BSD_SRVSOCKS     1

//     <o>Receive Timeout in seconds <0-600>
//     <i> A timeout for socket receive in blocking mode.
//     <i> Timeout value of 0 means indefinite timeout.
//     <i> Default: 20
#define BSD_RCVTOUT      20

//     <q>Hostname Resolver
//     <i> Enable or disable Berkeley style hostname resolver.
#define BSD_GETHOSTEN    0

// </e>
//------------- <<< end of configuration section >>> ----------------------

/*----------------------------------------------------------------------------
  *          Fatal Error Handler
  *---------------------------------------------------------------------------*/

void sys_error (ERROR_CODE code) {
   /* This function is called when a fatal error is encountered. The normal */
   /* program execution is not possible anymore. Add your crytical error      .*/
   /* handler code here.                                                                  */

   switch (code) {
      case ERR_MEM_ALLOC:
         /* Out of memory. */
         break;

      case ERR_MEM_FREE:
         /* Trying to release non existing memory block. */
         break;

      case ERR_MEM_CORRUPT:
         /* Memory Link pointer is Corrupted. */
         /* More data written than the size of allocated mem block. */
         break;

      case ERR_MEM_LOCK:
         /* Locked Memory management function (alloc/free) re-entered. */
         /* RTX multithread protection malfunctioning, not implemented */
         /* or interrupt disable is not functioning correctly. */
         break;

      case ERR_UDP_ALLOC:
         /* Out of UDP Sockets. */
         break;

      case ERR_TCP_ALLOC:
         /* Out of TCP Sockets. */
         break;

      case ERR_TCP_STATE:
         /* TCP State machine in undefined state. */
         break;
   }
   /* End-less loop */
   while (1);
}
```

```
/*----------------------------------------------------------------------------
 *          TCPnet Config Functions
 *----------------------------------------------------------------------------*/

#define   __NET_CONFIG__

#include <Net_lib.c>

/*----------------------------------------------------------------------------
 * end of file
 *----------------------------------------------------------------------------*/
```

miGlobal.h

```
#define    DEFAULT

#ifdef  DEFAULT
#define    __IP1B 192
#define    __IP2B 168
#define    __IP3B 1
#define    __IP4B 120

#define    __GW1B  192
#define    __GW2B  168
#define    __GW3B  1
#define    __GW4B  20

#define    __MASK1B255
#define    __MASK2B255
#define    __MASK3B255
#define    __MASK4B0
#endif
```

HTTP_CGI.h

```
/**-------------------------------------------------------------------------------------------------------------//
//      @filename     HTTP_CGI.h                                                                                //
//      @version      0.00                                                                                      //
//      @author       Alberto Palomo Alonso                                                                     //
//                                                                                                              //
//      @brief      Cabecera que configura los callback la página WEB.                                          //
//                                                                                                              //
//      @category     Opcional.                                                                                 //
//                                                                                                              //
//      @map          @include                                                                                  //
//                  @private                                                                                    //
//                  @funcdef                                                                                    //
//                  @end                                                                                        //
//                                                                                                              //
//                                                                                                              //
//-------------------------------------------------------------------------------------------------------------//
//                                                                                                              //
                                                                                                                //
//      @include      Estos son los archivos utilizados en el código de configuración.                          //
//                                                                                                              //
//-------------------------------------------------------------------------------------------------------------**/
#ifndef NETCONFIG
#define     NETCONFIG
#include    <Net_Config.h>
#endif
#ifndef STDIO
#define     STDIO
#include    <stdio.h>
#endif
#ifndef STDLIB
#define     STDLIB
#include <stdlib.h>
#endif
#ifndef STDINT
#define     STDINT
#include <stdint.h>
#endif
#ifndef SYSTEMSYMBOLS
#define     SYSTEMSYMBOLS
#include <Systemsymbols.h>
#endif
#ifndef LPC17XX
#define      LPC17XX
#include "LPC17XX.H"
#endif
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                              //
                                                                                                                //
//      @private      Estos son los símbolos correspondientes al cgi.                                           //
//                                                                                                              //
//-------------------------------------------------------------------------------------------------------------**/
#define     TEMPERATURA     't'
#define     VELOCIDAD       'v'
#define     PRESION         'p'
#define     HUMEDAD         'h'
#define     INDICEUV        'i'
#define     BRILLO      'b'
#define     ALTITUD         'a'
#define     LONGITUD        'x'
#define     LATITUD         'y'
#define     ANYO            'A'
#define     MES             'M'
#define     DIA         'D'
#define     HORAS       'H'
```

```
#define     MINUTOS       'T'
#define     SEGUNDOS      'S'
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
//                                                                                                                   //
//       @funcdef       Estas son las funciones correspondientes al cgi.                                             //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
void cgi_process_var  (   U8*    qs);//       NO SE USA.
void cgi_process_data    (   U8        tipo    ,   U8 *   qs  ,   U16        longitud);  //  NO SE USA.
U16 cgi_func          (   U8 *   env,   U8 *   buff    ,   U16       bufflen,   U32    *   pcgi);
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
//                                                                                                                   //
//       @end       ENDFILE.                                                                                         //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
```

HTTP_CGI.c

```
/**-------------------------------------------------------------------------------------------------------------//
//      @filename       HTTP_CGI.c                                                                              //
//      @version        3.00                                                                                    //
//      @author         Alberto Palomo Alonso                                                                   //
//                                                                                                              //
//      @brief      Código que contiene las llamadas a las funciones de CGI.                                    //
//                                                                                                              //
//      @category       Opcional.                                                                               //
//                                                                                                              //
//      @map            @include                                                                                //
//                  @extern                                                                                     //
//                  @funcion                                                                                    //
//                  @end                                                                                        //
//                                                                                                              //
//                                                                                                              //
//-------------------------------------------------------------------------------------------------------------//
//                                                                                                              //
                                                                                                                //
//      @include        Estos son los archivos utilizados en el código de configuración.                        //
//                                                                                                              //
//-------------------------------------------------------------------------------------------------------------**/
#ifndef HTTPCGI
#define     HTTPCGI
#include    "HTTP_CGI.h"
#endif
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                              //
                                                                                                                //
//      @extern         misDatos_t * DATOS -> main.c                                                            //
//                                                                                                              //
//-------------------------------------------------------------------------------------------------------------**/
extern misDatos_t     *    DATOS;
extern modificables_t      MODIFICABLES;
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                              //
                                                                                                                //
//      @function       cgi_process_var                                                                         //
//                                                                                                              //
//      @brief      Utilizado para el método GET.                                                               //
//                                                                                                              //
//-------------------------------------------------------------------------------------------------------------**/
void cgi_process_var  (    U8*     qs)
{
    U8 * var;
    var = (U8 *)alloc_mem(40);

    do
    {
        qs = http_get_var(qs , var, 40);
        if( var[0] )
        {
            if (str_scomp( var , (U8 *)"tmin="))
            {
                sscanf( (const char *)&var[5] , "%f" , &MODIFICABLES.Min_servo_t);
            }
            if (str_scomp( var , (U8 *)"tmax="))
            {
                sscanf( (const char *)&var[5] , "%f" , &MODIFICABLES.Max_servo_t);
            }
            if (str_scomp( var , (U8 *)"pmin="))
            {
                sscanf( (const char *)&var[5] , "%f" , &MODIFICABLES.Min_servo_p);
            }
            if (str_scomp( var , (U8 *)"pmax="))
```

```
            {
                sscanf( (const char *)&var[5] , "%f" , &MODIFICABLES.Max_servo_p);
            }
            if (str_scomp( var , (U8 *)"bsec="))
            {
                sscanf( (const char *)&var[5] , "%d" , &MODIFICABLES.TiempoBrillo);
            }
            if (str_scomp( var , (U8 *)"vart="))
            {
                MODIFICABLES.Var_medida = 0;
            }
            if (str_scomp( var , (U8 *)"varp="))
            {
                MODIFICABLES.Var_medida = 1;
            }
        }
    }while(qs);

    free_mem( (OS_FRAME *)var );
}
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
//                                                                                                             //
//      @function       cgi_process_var                                                                        //
//                                                                                                             //
//      @brief      NO UTILIZADO.                                                                              //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
void cgi_process_data    (    U8          tipo    ,    U8 *    qs  ,    U16         longitud)
{
    //  NO UTILIZADO,    NO HAY PETICIONES EN ESTA VERSIÓN.
}
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
//                                                                                                             //
//      @function       cgi_func()                                                                             //
//                                                                                                             //
//      @brief      Función que es llamada por el CGI cada vez que se socilicita una callback.                 //
//                  Obtiene una cadena de caracteres como parámetro de CGI y actúa en consecuencia.            //
//                  En nuestro caso sólo llama a los datos y los exporta a html.                               //
//                                                                                                             //
//          @env            Cadena de caracteres de entrada.                                               //
//          @buff           Salida de datos.                                                               //
//          @bufflen        (No utilizado)Tamaño del buffer.                                               //
//          @pcgui          (No utilizado)                                                                 //
//                                                                                                             //
//              @return     Tamaño de la cadena de salida en bytes.                                        //
//                                                                                                             //
//                                                                                                             //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
U16 cgi_func         (    U8 *    env,    U8 *    buff    ,    U16         bufflen,    U32    *    pcgi)
{
    U32 longitud;

    switch(     env[0] )
    {
        case    TEMPERATURA:
            longitud = sprintf  (    (char*)buff    ,    (const char *)&env[4]    ,    DATOS->Temperatura    );
            break;
        case    PRESION:
            longitud = sprintf  (    (char*)buff    ,    (const char *)&env[4]    ,    DATOS->Presion        );
            break;
        case    HUMEDAD:
            longitud = sprintf  (    (char*)buff    ,    (const char *)&env[4]    ,    DATOS->Humedad        );
            break;
```

```
        case    BRILLO:
            longitud = sprintf (    (char*)buff     ,    (const char *)&env[4]     ,    DATOS->Brillo      );
            break;
        case    ALTITUD:
            longitud = sprintf (    (char*)buff     ,    (const char *)&env[4]     ,    DATOS->Lugar.Altura );
            break;
        case    LATITUD:
            longitud = sprintf (    (char*)buff     ,    (const char *)&env[4]     ,    DATOS->Lugar.Latitud);
            break;
        case    LONGITUD:
            longitud = sprintf (    (char*)buff     ,    (const char *)&env[4]     ,    DATOS->Lugar.Longitud);
            break;
        case    INDICEUV:
            longitud = sprintf (    (char*)buff     ,    (const char *)&env[4]     ,    DATOS->IndiceUV );
            break;
        case    VELOCIDAD:
            longitud = sprintf (    (char*)buff     ,    (const char *)&env[4]     ,    DATOS->VelViento     );
            break;
        case ANYO:
            longitud = sprintf (    (char*)buff     ,    (const char *)&env[4]     ,    LPC_RTC->YEAR        );
            break;
        case MES:
            longitud = sprintf (    (char*)buff     ,    (const char *)&env[4]     ,    LPC_RTC->MONTH           );
            break;
        case DIA:
            longitud = sprintf (    (char*)buff     ,    (const char *)&env[4]     ,    LPC_RTC->DOM         );
            break;
        case HORAS:
            longitud = sprintf (    (char*)buff     ,    (const char *)&env[4]     ,    LPC_RTC->HOUR        );
            break;
        case MINUTOS:
            longitud = sprintf (    (char*)buff     ,    (const char *)&env[4]     ,    LPC_RTC->MIN         );
            break;
        case SEGUNDOS:
            longitud = sprintf (    (char*)buff     ,    (const char *)&env[4]     ,    LPC_RTC->SEC     );
            break;
        default:
            longitud = sprintf (    (char*)buff     ,    "<p>Y... que quieres que ponga aqui? :v</p>");
            break;
    }
    return (    (U16)longitud );
}
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
//                                                                                                                 //
//      @end        ENDFILE.                                                                                       //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
```

Index.cgi

```
t <!DOCTYPE html>
t   <html>
t     <head>
t       <meta content="text/html; charset=UTF-8" http-equiv="content-type">
#         ----------------------------------------------------------------------------> OJO QUE HAY QUE PONER LA URL CORRECTA:
t       <meta http-equiv="refresh" content="20; url=http://192.168.1.120/index.cgi">
t         <title>ESTACION METEOROLOGICA</title>
t           </head>
t             <body style="background-color:rgb(200,200,200);">
t               <h1 align="center">Estacion meteorologica</h1>
t                 <br />
t                 <table align="center" border="1">
t                   <caption>Datos medios actuales:</caption>
t                     <tr>
t                       <td>Temperatura:</td>
t                       <td>
c t                      " %f
t                       </td>
t                       <td>Velocidad del viento:</td>
t                       <td>
c v                      " %f
t                       </td>
t                     </tr>
t                     <tr>
t                       <td>Humedad:</td>
t                       <td>
c h                      " %f
t                       </td>
t                       <td>Indice UV:</td>
t                       <td>
c i                      " %f
t                       </td>
t                     </tr>
t                     <tr>
t                       <td>Presion:</td>
t                       <td>
c p                      " %f
t                       </td>
t                       <td>Brillo:</td>
t                       <td>
c b                      " %f
t                       </td>
t                     </tr>
t                 </table>
t                 <table align="center" border="1">
t                   <tr>
t                   <td>Altitud:</td>
t                   <td>
c a                  " %f
t                   </td>
t                   <td>Longitud:</td>
t                   <td>
c x                  " %f
t                   </td>
t                   <td>Latitud:</td>
t                   <td>
c y                  " %f
t                   </td>
t                   </tr>
t                 </table>
t                 <br></br>
t                 <table align="center" border="1">
t                   <caption>Hora de la ultima muestra:</caption>
t                   <tr>
```

```
t                <td>Anyo:</td>
t                <td>
c A              " %d
t                </td>
t                <td>Mes:</td>
t                <td>
c M              " %d
t                </td>
t                <td>Dia:</td>
t                <td>
c D              " %d
t                </td>
t               <tr>
t                <td>Hora:</td>
t                <td>
c H              " %d
t                </td>
t                <td>Minuto:</td>
t                <td>
c T              " %d
t                </td>
t                <td>Segundos:</td>
t                <td>
c S              " %d
t                </td>
t               </tr>
t              </table>
t              <br></br>
t              <br></br>
t              <table stile="width: 100%" border="1" align="center">
t               <tbody>
t                 <tr>
t                  <td>
t                   <h1 style=" text-align: center;"> Magnitudes modificables: </h1>
t                   <form method="GET" action="index.cgi">
t                    <br> Temperatura min. :
t                    <input size="10" value="-10" name="tmin" type="text">
t                    <br> Temperatura max. :
t                    <input size="10" value="50" name="tmax" type="text">
t                    <br> Presion min. :
t                    <input size="10" value="500" name="pmin" type="text">
t                    <br> Presion max. :
t                    <input size="10" value="1500" name="pmax" type="text">
t                    <br> Segundos encendido :
t                    <input size="10" value="10" name="bsec" type="text">
t                    <br> <input value="si" type="radio" name="vart"> Temperatura<br>
t                    <input value="si" type="radio" name="varp"> Presion<br>
t                    <br> <input value="Enviar" type="submit">
t                   </form>
t                  </td>
t                 </tr>
t               </tbody>
t              </table>
t              <br></br>
t              <br></br>
t              <br></br>
t              <br></br>
t              <table align="center">
t               <tr>
t                 <td>Autor:  Alberto Palomo Alonso.</td>
t                 <td>Sistemas Electronicos Digitales Avanzados.</td>
t                 <td>Universidad de Alcala - Escuela politecnica superior.</td>
t               </tr>
t              </table>
t         </body>
t    </html>
```

RTC.h

```
/**-------------------------------------------------------------------------------------------------------------//
//      @filename       RTC.h                                                                                  //
//      @version        0.00                                                                                       //
//      @author         Alberto Palomo Alonso                                                                          //
//                                                                                                             //
//      @brief      Cabecera del código RTC.                                                                       //
//                                                                                                             //
//      @category       Opcional.                                                                                  //
//                                                                                                             //
//      @map            @include                                                                               //
//                  @private                                                                                       //
//                  @funcdef                                                                                       //
//                  @end                                                                                           //
//                                                                                                             //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
                                                                                                                    //
//      @include        Estos son los archivos utilizados con el código fuente.                                        //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
#ifndef SYSTEMSYMBOLS
#define     SYSTEMSYMBOLS
#include "Systemsymbols.h"
#endif
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
                                                                                                                    //
//      @private        Estos son los símbolos privados a RTC.                                                         //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
#define     MITIEMPO(time_t *)(LPC_RTC_BASE)
#define     RTCMASK             1 << 9
#define     CALIBRACION_RTC     (1 << 4) | (0x1 << 1)
#define     CALIBRATION_VALUE 0x00
#define     INT_SEGUNDOS        1 << 0
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
                                                                                                                    //
//      @funcdef        Funciones a definir en el RTC.                                                             //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
void __configuraRTC__(   void    );
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
                                                                                                                        //
//      @end        ENDFILE.                                                                                       //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
```

RTC.c

```
/**-------------------------------------------------------------------------------------------------------------//
//      @filename     RTC.c                                                                                    //
//      @version      0.00                                                                                     //
//      @author       Alberto Palomo Alonso                                                                    //
//                                                                                                             //
//      @brief      Código fuente del configurado y manejador del RTC.                                         //
//                                                                                                             //
//      @category     Opcional.                                                                                //
//                                                                                                             //
//      @map          @include                                                                                 //
//                    @variables                                                                               //
//                    @function                                                                                //
//                    @HANDLER                                                                                 //
//                    @end                                                                                     //
//                                                                                                             //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
                                                                                                               //
//      @include      Estos son los archivos utilizados con el código del RTC.                                 //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
#ifndef RTC
#define     RTC
#include "RTC.h"
#endif
#ifndef STRING
#define     STRING
#include <string.h>
#endif
#ifndef STDIO
#define     STDIO
#include    <stdio.h>
#endif
#ifndef SYSTEMSYMBOLS
#define     SYSTEMSYMBOLS
#include    "Systemsymbols.h"
#endif
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
                                                                                                               //
//      @variables          Variables del fichero.                                                           //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
uint8_t Clock[23];
extern Counters_t     *    COUNTERS;
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
                                                                                                               //
//      @function __configuraRTC__()                                                                           //
//                                                                                                             //
//      @brief Función que configura el RTC como un contador que interrumpe cada segundo.                      //
//                                                                                                             //
//            @Tiempo  Puntero a variable donde se almacena el tiempo actual.                                  //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
void __configuraRTC__( void )
{
    LPC_SC->PCONP       |= RTCMASK;               // Activo el RTC.
    LPC_RTC->CCR        = CALIBRACION_RTC;        // Calibro el RTC.
    LPC_RTC->CALIBRATION = CALIBRATION_VALUE;
    LPC_RTC->CCR     = 0x1;
    LPC_RTC->CIIR    |= INT_SEGUNDOS;           // Interrupción del RTC cada segundo.
    LPC_RTC->YEAR       =   2020;               // Configuro el registro
```

```
    LPC_RTC->MONTH      =  1;               //  que tiene en cuenta
    LPC_RTC->DOM        =  1;               //  los dias, meses,
    LPC_RTC->HOUR       =  0;               //  minutos y segundos
    LPC_RTC->MIN        =  0;               //  del RTC.
    LPC_RTC->SEC    =  0;               //
    NVIC_EnableIRQ( RTC_IRQn);              // Habilito la interrumpción del RTC.
    NVIC_SetPriority( RTC_IRQn ,   0   );   // Se le asigna prioridad alta.
}
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
//      @HANDLER    RTC_IRQHandler()                                                                                 //
//                                                                                                                   //
//      @brief Manejador de la interrupción RTC.                                                                     //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
void RTC_IRQHandler( void )
{
    COUNTERS->Segundos      += 0x1;// Incrementa el contador.
    LPC_RTC->ILR            |= 1;   // Borra el flag de interrupción.
    sprintf((char*)Clock,"%02d/%02d/%04d-%02d:%02d:%02d",LPC_RTC->DOM,LPC_RTC->MONTH,LPC_RTC->YEAR,LPC_RTC->HOUR,LP
C_RTC->MIN,LPC_RTC->SEC);
//  if (COUNTERS->Segundos == 60)
//  {
//      __sumaMinReloj__();
//      COUNTERS->Segundos = 0;
//  }
}
/**-------------------------------------------------------------------------------------------------------------------//
//                                                                                                                   //
//                                                                                                                   //
//      @end       ENDFILE.                                                                                         //
//                                                                                                                   //
//-------------------------------------------------------------------------------------------------------------------**/
```

Timers.h

```
/**-------------------------------------------------------------------------------------------------------------//
//      @filename       Timers.h                                                                                //
//      @version        0.00                                                                                    //
//      @author         Alberto Palomo Alonso                                                                       //
//                                                                                                              //
//      @brief      Cabecera para configurar los timers.                                                        //
//                                                                                                              //
//      @category       Principal.                                                                          //
//                                                                                                              //
//      @map            @include                                                                                //
//                  @funcdef                                                                                //
//                  @end                                                                                    //
//                                                                                                          //
//                                                                                                          //
//-------------------------------------------------------------------------------------------------------------//
//                                                                                                          //
                                                                                                                //
//      @include        Estos son los archivos utilizados para los timers.                                          //
//                                                                                                          //
//-------------------------------------------------------------------------------------------------------------**/
#ifndef RTL
#define     RTL
#include    "RTL.h"
#endif
#ifndef LPC17XX
#define     LPC17XX
#include    "LPC17XX.H"
#endif
#ifndef SYSTEMSYMBOLS
#define     SYSTEMSYMBOLS
#include    "Systemsymbols.h"
#endif
#ifndef ANEMOMETRO
#define     ANEMOMETRO
#include    "Anemometro.h"
#endif
#ifndef LDR
#define     LDR
#include    "LDR.h"
#endif
#ifndef DAC
#define     DAC
#include    "DAC.h"
#endif
#ifndef PWM
#define     PWM
#include    "PWM.h"
#endif
#ifndef UFONO
#define     UFONO
#include    "uFono.h"
#endif
#ifndef ONEWIRE
#define     ONEWIRE
#include    "OneWire.h"
#endif
#ifndef I2C
#define     I2C
#include    "I2C.h"
#endif
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                          //
                                                                                                                //
//      @private        Estos son los símbolos correspondientes a los timers.                                       //
```

```
//                                                                                                                  //
//------------------------------------------------------------------------------------------------------------------**/
#define     FREQ_OVERFLOW_SYSTICK   10                      //  Frecuencia en Hz de overflow o fin de cuenta del SysTick. (Tsyst =
100ms)
#define     ENABLEBIT_SYST          0x1
#define     FCPUBIT_SYST        0x4
#define     ENABLEINTBIT_SYST       0x2
#define     MASCARA_CTRL_SYSTICK FCPUBIT_SYST | ENABLEBIT_SYST | ENABLEINTBIT_SYST
#define     SYSTICK_COUNTFLAG       0x1 << 16
//  Para cada timer.
#define     ACTIVAR_TIMER           0x1
#define     RESET_TIMER_TCR         0x2
#define     TIMER0_BIT          (0x1 << 1)
#define     TIMER1_BIT          (0x1 << 2)
#define     TIMER2_BIT          (0x1 << 22)
#define     TIMER3_BIT          (0x1 << 23)
#define     TIMER0_MCR_MASK         0x3 << (0*3)                //  Activo la interrupción y reseteo el contador.
#define     TIMER1_MCR_MASK         0x3 << (0*3)                //  No usado.
#define     TIMER2_MCR_MASK         0x3 << (0*3)
#define     TIMER3_MCR_MASK         0x1 << (0*3)

#define     MODO_ENTRADA            1
#define     MODO_SALIDA             0
/**----------------------------------------------------------------------------------------------------------------//
//                                                                                                                  //
                                                                                                                    //
//      @funcdef        Estas son las funciones correspondientes a los timers.                                      //
//                                                                                                                  //
//------------------------------------------------------------------------------------------------------------------**/
void __configuraSysTick__    (    void    );  //  TCP.
void __configuraTimer0__         (    void    );  //  Muestreo.
void __configuraTimer1__         (    void    );  //  ...
void __configuraTimer2__         (    void    );  //  Audio.
void __configuraTimer3__         (    void    );  //  ...
/**----------------------------------------------------------------------------------------------------------------//
//                                                                                                                  //
                                                                                                                    //
//      @end        ENDFILE.                                                                                    //
//                                                                                                                  //
//------------------------------------------------------------------------------------------------------------------**/
```

Timers.c

```
/**-------------------------------------------------------------------------------------------------------------//
//      @filename       Timers.c                                                                                //
//      @version        7.00                                                                                    //
//      @author         Alberto Palomo Alonso                                                                   //
//                                                                                                              //
//      @brief      Código que configura y programa los manejadores de los timers.                              //
//                                                                                                              //
//      @category       Principal.                                                                              //
//                                                                                                              //
//      @map            @include                                                                                //
//                  @variables                                                                                  //
//                  @funcion                                                                                    //
//                  @end                                                                                        //
//                                                                                                              //
//                                                                                                              //
//-------------------------------------------------------------------------------------------------------------//
//                                                                                                              //
                                                                                                                //
//      @include        Estos son los archivos utilizados para los timers.                                      //
//                                                                                                              //
//-------------------------------------------------------------------------------------------------------------**/
#ifndef TIMERS
#define     TIMERS
#include    "Timers.h"
#endif
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                              //
                                                                                                                //
//      @variables          Variables del fichero.                                                              //
//                                                                                                              //
//-------------------------------------------------------------------------------------------------------------**/
uint8_t                 TIM0_ticks      =   0;
uint8_t                 Timer2_MODO     =   MODO_SALIDA;
uint32_t                    CAP11_BUFF  =   0;
//  Contador.
uint16_t                    contadorLUZ =   0;
//  Externos.
extern      uint8_t         __brilloAuto;
extern uint8_t          __brilloFade;
extern uint8_t          YaPuedesMedir;
extern Counters_t   *   COUNTERS;
extern misDatos_t   *   DATOS;
extern actualizador_t * ACTUALIZADOR;
extern uint8_t      *   AUDIO;
extern uint8_t      *   CAPcont;
extern modificables_t   MODIFICABLES;
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                              //
                                                                                                                //
//      @function       __configuraSysTick__()                                                                  //
//                                                                                                              //
//      @brief      Configura el systick para desbordar cada 100 ms.                                            //
//                                                                                                              //
//-------------------------------------------------------------------------------------------------------------**/
void __configuraSysTick__()
{
    SysTick->LOAD = (SystemCoreClock / FREQ_OVERFLOW_SYSTICK) - 1; //  SysTick configurado a desbordar cada 100 ms para TcpNet.
    SysTick->CTRL = MASCARA_CTRL_SYSTICK;                          //  Fcpu como clock y no activo la interrupción del SysTickTimer.
    SysTick_Config( SystemCoreClock / FREQ_OVERFLOW_SYSTICK);
}
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                              //
                                                                                                                //
//      @function       __configuraTimer0__()                                                                   //
```

```
//                                                                                                  //
//      @brief      Configura el Timer0 para nterrumpir cada Ts0 segundos.                          //
//                                                                                                  //
//--------------------------------------------------------------------------------------------------**/
void __configuraTimer0__()
{
    LPC_SC->PCONP |=   0x1 << 22 | 0x1 << 23 | 1 << 16; //  Todos los timer.
    LPC_SC->PCONP  |= TIMER0_BIT;                   //  Activo el módulo del timer 0.
    LPC_TIM0->MCR  =  TIMER0_MCR_MASK;              //  Activo el ISR y reseteo TC.
    LPC_TIM0->PR=   0;                          //  Sin prescaler.
    LPC_TIM0->TCR   |= ACTIVAR_TIMER;               //  Activo el timer.
    LPC_TIM0->MR0  =  Ftick * Ts0 - 1;          //  Cargo para que interrumpa cada 0.5s.
    NVIC_SetPriority( TIMER0_IRQn ,   1   );        //  Para que el ADC interrumpa bien.
    NVIC_EnableIRQ( TIMER0_IRQn );
}
/**------------------------------------------------------------------------------------------------//
//                                                                                                  //
//                                                                                                  //
//      @function       __configuraTimer1__()                                                       //
//                                                                                                  //
//      @GOTO           ¡DEFINIDO EN EL ANEMOMETRO! (Anemometro.c)                                  //
//                                                                                                  //
//--------------------------------------------------------------------------------------------------**/
/**------------------------------------------------------------------------------------------------//
//                                                                                                  //
//                                                                                                  //
//      @function       __configuraTimer3__()                                                       //
//                                                                                                  //
//      @GOTO           ¡DEFINIDO EN EL ONEWIRE! (OneWire.c)                                        //
//                                                                                                  //
//--------------------------------------------------------------------------------------------------**/
/**------------------------------------------------------------------------------------------------//
//                                                                                                  //
//                                                                                                  //
//      @HANDLER        SysTick_Handler()                                                           //
//                                                                                                  //
//      @brief      Manejador de la interrupción del SysTick. Cada 100 ms se realizan acciones.     //
//                                                                                                  //
//--------------------------------------------------------------------------------------------------**/
void SysTick_Handler()
{

    timer_tick();
    if (contadorLUZ   <   FREQ_OVERFLOW_SYSTICK * (MODIFICABLES.TiempoBrillo))
    {
        contadorLUZ++;
    }
    else
    {
        if (__brilloFade)                           //  Si pasan 60s y el brillo automático está desactivado...
        {
            __brilloAuto = 0;
            __brilloFade = 0;
            modificaPulso(    PWM6 ,    MODO_CICLO ,    1   ,    none   ,    none   ,    none   );  //  Apago la pantalla.
        }
    }
}
/**------------------------------------------------------------------------------------------------//
//                                                                                                  //
//                                                                                                  //
//      @HANDLER        TIMER0_IRQHandler()                                                         //
//                                                                                                  //
//      @brief      Manejador de la interrupción del Timer0. Reanima el muestreo de los sensores.   //
//                                                                                                  //
//--------------------------------------------------------------------------------------------------**/
//  Bloque 1:   Apoyo del timer 1:
```

```
//                          Temperatura + Humedad + Vel.Viento.
static void _subAnemoTempe()
{
    LPC_TIM0->IR =  LPC_TIM0->IR;           // Borro interrupción.
    if(  !(TIM0_ticks % (uint8_t)CsCAP)  )      // Si toca muestrear captures...
    {
        LPC_TIM1->CCR  |= CCR_MASCARA_EN;       // Genera interrupción el CAP1.0, ojo que se mata así en el timer 1.
        LPC_TIM1->CCR  |= OW_CCR_MASCARA_EN;    // Genera interrupción el CAP1.1, ojo que se mata así en el timer 1.
        mideTemperatura();                      // Le digo a la placa que lanze la señal de request.
        medirBMP();                         // Leo el sensor de presión atmosférica.
        if ( !ACTUALIZADOR->AnemometroRev && YaPuedesMedir)     // Si el actualizador está a 0 (Es decir, no hay datos
capturados).
        {
            DATOS->VelViento = 0;               // No hay viento.
            ACTUALIZADOR->Anemometro = 1;       // Ya está medido, es 0 m/s.
        }
        ACTUALIZADOR->AnemometroRev = 0;        // Digo que ya he medido.
        ACTUALIZADOR->TempRev  =  1;
    }
}
// Bloque 2:  ADC burst:
//                      UVA + LDR.
static void     _subBurst()
{
    if(  !(TIM0_ticks % (uint8_t)CsADC)  )          // LDR + UVA van el BURST.
    {
        if(  ACTUALIZADOR->LDRrev && YaPuedesMedir  )   // Es bloqueable por el audio.
        {
            LPC_SC->PCONP           |= PCONP_ADC_ON; // Enciendo el ADC.
            ACTUALIZADOR->LDRrev    =  0;           // Aviso que no he medido aún.
            LPC_ADC->ADCR           &= ~ADC_START; // Ojito que es modo ráfaga, no hay start.
            LPC_ADC->ADCR           |= BRUST_PIN;   // Ráfaga.
        }
    }
}
// Actualizo el servo.
void __subServo( void  )
{
    if ( !MODIFICABLES.Var_medida )
    {
        if (DATOS->Temperatura >= MODIFICABLES.Max_servo_t)
        {
            modificaPulso  ( PWM2,  MODO_SERVO  ,  none  ,  180  ,  MINIMO_SERVO  ,  MAXIMO_SERVO );
            if (ACTUALIZADOR->Audiorev)
            {
                ACTUALIZADOR->Audiorev = 0;
                __configuraTono__();
                activarDac();
            }
        }
        if (DATOS->Temperatura <= MODIFICABLES.Min_servo_t)
        {
            modificaPulso  ( PWM2,  MODO_SERVO  ,  none  ,  0  ,  MINIMO_SERVO  ,  MAXIMO_SERVO );
            if (ACTUALIZADOR->Audiorev)
            {
                ACTUALIZADOR->Audiorev = 0;
                __configuraAudio__();
                activarDac();
            }
        }
        modificaPulso  ( PWM2,  MODO_SERVO  ,  none  ,  (180*(DATOS->Temperatura - MIN_TEMP)/(MAX_TEMP -
MIN_TEMP))  ,  MINIMO_SERVO  ,  MAXIMO_SERVO );
    }
    else
    {
        if (DATOS->Presion >= MODIFICABLES.Max_servo_p)
```

```
        {
            modificaPulso ( PWM2, MODO_SERVO , none , 180 , MINIMO_SERVO , MAXIMO_SERVO );
            if (ACTUALIZADOR->Audiorev)
            {
                ACTUALIZADOR->Audiorev = 0;
                __configuraTono__();
                activarDac();
            }
        }
        if (DATOS->Presion <= MODIFICABLES.Min_servo_p)
        {
            modificaPulso ( PWM2, MODO_SERVO , none , 0 , MINIMO_SERVO , MAXIMO_SERVO );
            if (ACTUALIZADOR->Audiorev)
            {
                ACTUALIZADOR->Audiorev = 0;
                __configuraAudio__();
                activarDac();
            }
        }
        modificaPulso ( PWM2, MODO_SERVO , none , (180*(DATOS->Presion - MIN_PRES)/(MAX_PRES - MIN_PRES))
    , MINIMO_SERVO , MAXIMO_SERVO );
    }
}
// Ahora sí, el handler: Ojo que aquí es donde actualizo el servo.
void TIMER0_IRQHandler( void )
{
    _subAnemoTempe();
    _subBurst();
    TIM0_ticks++;
    __subServo();
}
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
//      @HANDLER        TIMER1_IRQHandler()                                                                    //
//                                                                                                             //
//      @ref            Ir a Anemómetro.c (comparte con OneWire.c)                                             //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
void TIMER1_IRQHandler()
{
    uint8_t SWART   =   (uint8_t)(LPC_TIM1->IR);

    if  (SWART  &  CAP10_IR)
    {
        mideAnemometro();
    }
    if  (SWART  &  MR1_IR)
    {
        desactivarDAC();
    }
    LPC_TIM1->IR =  LPC_TIM1->IR;       // No pierdo nada en asegurarme que se cierra el timer.
}
/**-------------------------------------------------------------------------------------------------------------//
//                                                                                                             //
//      @HANDLER        TIMER2_IRQHandler()                                                                    //
//                                                                                                             //
//      @brief      N/A                                                                                        //
//                                                                                                             //
//-------------------------------------------------------------------------------------------------------------**/
void TIMER2_IRQHandler()
{
    // NO USADO.
}
/**-------------------------------------------------------------------------------------------------------------//
```

```
//                                                                    //
                                                                                        //
//      @HANDLER        TIMER3_IRQHandler()                                       //
//                                                                    //
//      @brief      Timer de apoyo para el monohilo.                           //
//                                                                    //
//----------------------------------------------------------------------------------------**/
//  USADO POR EL MONOHILO.
void    TIMER3_IRQHandler()
{
    //  NO USADO.
}
/**---------------------------------------------------------------------------------------//
//                                                                    //
                                                                                  //
//      @end        ENDFILE.                                                  //
//                                                                    //
//----------------------------------------------------------------------------------------**/
```