



NOMBRE	LEGAJO	CURSO	FECHA	HOJAS
Alejandro Galfrascoli	43029	6R3	09/08/2021	1 de 21

Sensor de Temperatura ST-1820

Introducción

Este dispositivo desarrollado en el presente informe es un medidor de temperaturas basado en sensores 1-Wire del tipo DS18B20^[1] capaz de medir hasta 4 sensores simultáneamente (aunque esta limitación está dada solo por configuración del firmware para adaptarla a las opciones del menú).

Tiene además la posibilidad de modificar la escala de medición, es decir, en °C y °F.

También es configurable la frecuencia de adquisición de los sensores (3, 5 o 10 segundos).

Para la visualización de las temperaturas se pueden elegir entre 2 modos de operación que serán descritos más adelante.

Descripción

Para este proyecto se utilizó la placa de desarrollo Blue Pill que cuenta con un microprocesador ARM Cortex-M3 STM32F103C8T6 de ST Microelectronics en el cual se ha embebido el Kernel uCOSII^[2] v2.9216.

Además de la placa de desarrollo se han utilizado los siguientes elementos:

- Pantalla OLED SSD1306 128x64 pixeles de 0.96" con comunicación I2C.
- Encoder rotativo, que es utilizado para controlar el dispositivo.
- Sensores de temperatura del tipo 1-Wire como por ej. los DS18B20, DS1820 o DS18S20.
- UART para poder realizar una conexión con la PC (para un menú de diagnóstico).

Un vistazo al Hardware

Se puede apreciar a continuación en la figura 1 las vistas de la placa terminada con todos los componentes ensamblados del sistema embebido.

Y en la figura 2 el circuito correspondiente.

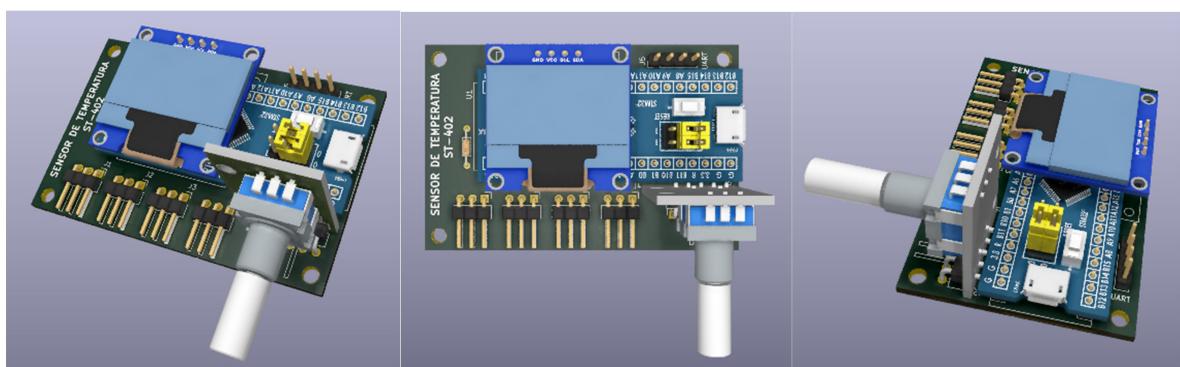


Figura 1: Vistas del renderizado de la placa.

NOMBRE	LEGAJO	CURSO	FECHA	HOJAS
Alejandro Galfrascoli	43029	6R3	09/08/2021	2 de 21

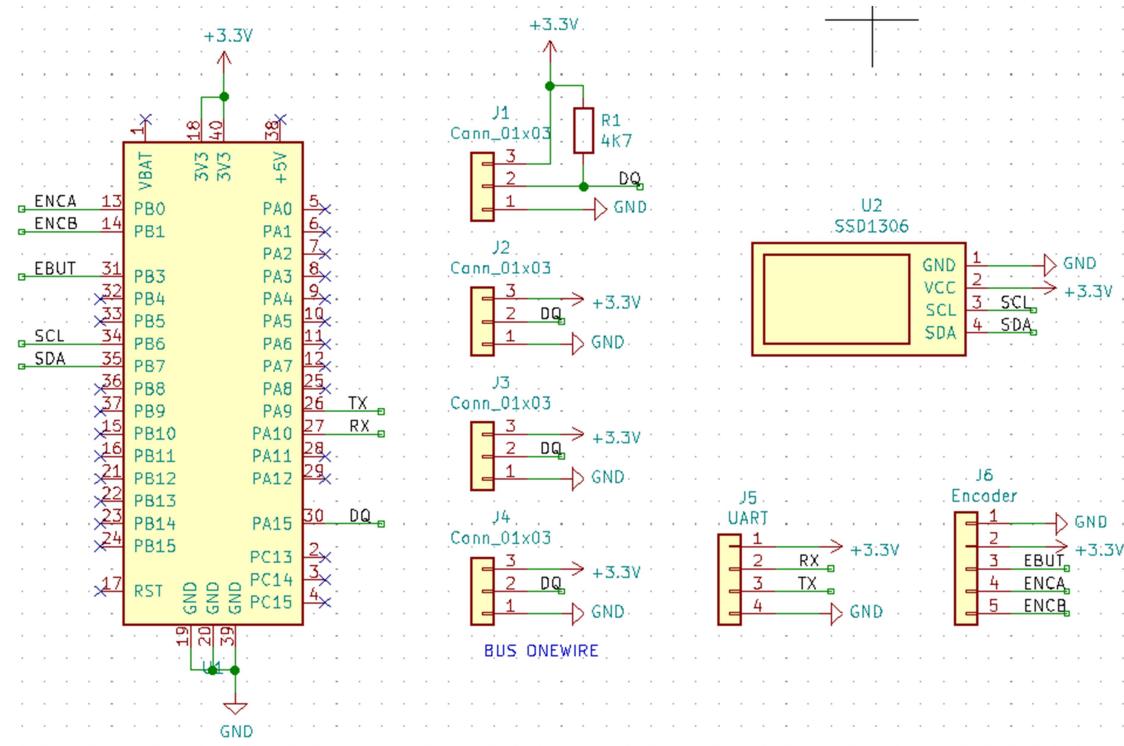


Figura 2: Esquemático

Descripción del Software

En lo que concierne al software, en esta aplicación se hace uso de del módulo Discreto de Entrada Salida DIO diseñado por Jean J. Labrosse^[3], el cual fue adaptado para poder funcionar con la plataforma de desarrollo elegida.

Para sincronizar las tareas se hace uso de eventos del tipo Flags, Semáforos binarios y Mailbox.

Por otra parte se han desarrollado las siguientes librerías y adaptadas para trabajar con el RTOS y que se describen a continuación:

- onewire:** es el driver para establecer el protocolo de comunicación a través del bus 1-wire, contiene las funciones de bajo nivel para el manejo de dispositivos 1-wire. En ella se hace uso del Timer 1 para poder generar los retardos necesarios para el protocolo de comunicación. Esta librería fue basada en la nota de aplicaciones Maxim 187^[4].
- sensor:** esta librería es para el manejo y lectura de los sensores, es una capa de software adicional entre la aplicación de usuario y el driver onewire del cual hace uso.
- ssd1306:** esta librería es el driver para el control del display OLED que a su utiliza como periférico de comunicaciones el bus I2C1 del microcontrolador que implementa un semáforo para compartir el recurso.



NOMBRE	LEGAJO	CURSO	FECHA	HOJAS
Alejandro Galfrascoli	43029	6R3	09/08/2021	3 de 21

- d) **gui:** Interfaz Gráfica de Usuario para la gestión del menú de opciones, contiene todas las funciones para necesarias para la presentación de datos en la pantalla, la misma hace uso de la librería ssd1306.
- e) **ftostr:** es una librería auxiliar, diseñada para convertir números enteros decimales, hexadecimales y de punto flotantes en un cadena de caracteres. También admite números negativos, para poder representar correctamente temperaturas bajo cero.
Este módulo se desarrolló para no hacer uso de las funciones estándar como sprintf(), svprintf() por ej. Debido al alto consumo de memoria que requieres.
- f) **COM:** Para la comunicación serial via UART la cual implementa el uso de semáforo para compartir el recurso.

Se intentará hacer una breve descripción de cómo está constituido el firmware desde el punto de vista de la aplicación de usuario y no enfocado en el Kernel en sí, tratando de explicar cómo interactúan entre si las tareas cuyo esquema es visualizado en el diagrama de la figura 2.

La aplicación consta de 11 tareas de las cuales las 3 primeras son del sistema operativo en tiempo real, por lo que no serán descritas:

- 1- **Idle task.**
- 2- **Stat.**
- 3- **Tmr.**
- 4- **Startup:** Es la tarea de inicialización del sistema, cumple varias funciones que se detallan a continuación:
Configura el clock del sistema, inicializa las estadísticas, crea a las restantes tareas con el llamado a `App_TaskCreate()`.
También se inicializan los módulos utilizados llamando a `App_InitModules()`.
Luego crea los eventos para la sincronización entre las tareas con `App_EventCreate()`.
A posterior suspende la task1, la cual solo reanuda su ejecución cuando es solicitado desde el menú de opciones.
Finalmente esta tarea se auto elimina, quedando solo 10 tareas en el planificador.
- 5- **DIOTask:** Esta tarea es creada por el modulo Discreto de Entrada Salida DIO para el control de los pines digitales de E/S.
- 6- **Task1:** Aquí se puede realizar una prueba de diagnóstico del hardware vía RS-232, ejecutando comandados desde una consola o aplicación tipo Hyperterminal. Esta tarea se ejecuta cada 500ms para verificar si hay teclas presionadas.
La misma se encuentra suspendida hasta ser habilitada desde el menú de opciones.
La Task1 se suspende a sí misma si es seleccionada la opción 'X'.
El carácter es recibido por la UART y las opciones disponibles son las siguientes:
'D' → **Select/OK:** simula una pulsación del encoder.
'W' → **UP:** simula un giro en sentido ascendente del encoder.
'S' → **DOWN:** simula un giro en sentido descendente del encoder.

NOMBRE	LEGAJO	CURSO	FECHA	HOJAS
Alejandro Galfrascoli	43029	6R3	09/08/2021	4 de 21

Cualquiera de las 3 opciones anteriores permite la navegación a través del menú como si fuera generado desde el mismo encoder rotativo.

‘P’ → **Test OLED**: hace una prueba de la pantalla OLED graficando figuras geométricas.

‘T’→ **Sensor Table**: muestra la tabla de sensores detectados, en ella se puede ver el índice de sensor en la tabla, el código de ROM del sensor (su address), y la temperatura medida en formato raw de 12 bits.

‘H’→ **Help**: despliega nuevamente el menú de opciones de test por uart.

‘X’→ **Exit**: Finaliza el test por uart, la tarea se suspende nuevamente.

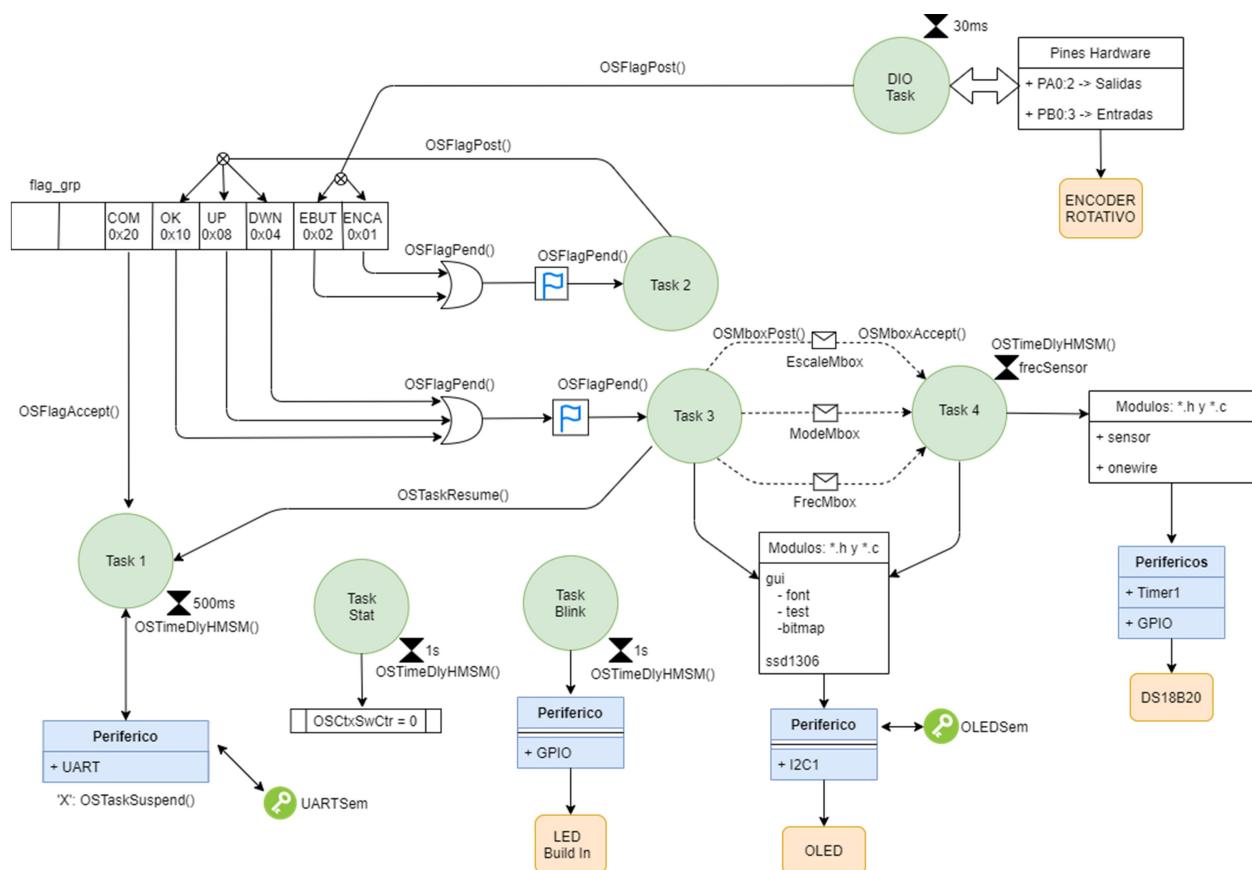


Figura 2: diagrama de la interacción entre tareas.

7- Task2: Esta es la tarea encargada de manejar el encoder rotativo. Se ejecuta cada vez que se activan cualquiera de los siguientes flags:

ENCA: detección de flanco descendente en pin Encoder A.

EBUT: detección de flanco descendente en pin Button.

Los flags ENCA y EBUT son seteados en **EdgeAFnct()** y **EdgeButFnct()** que son las funciones de callback en el modo detección de flancos del módulo DIO.



NOMBRE	LEGAJO	CURSO	FECHA	HOJAS
Alejandro Galfrascoli	43029	6R3	09/08/2021	5 de 21

8- Task3: Gestiona el menú de opciones y su presentación en la pantalla OLED.

Esta tarea se ejecuta cada vez que se activan cualquiera de los siguientes flags, los cuales son consumidos en la llamada a `OSFlagPend()`:

DOWN: giro en sentido antihorario del encoder.

UP: giro en sentido horario del encoder.

OK: pulsación del botón de encoder.

Esta tarea también envía los MailBox a la Task4 con la configuración de:

`EscaleSensor`: Escala de lectura en grados Celsius (1) o en grados Fahrenheit (2).

`ModeSensor`: Es el modo de visualización de los sensores en lcd (1) o (2) para visualizar los sensores 1 y 2, o el 3 y 4 respectivamente.

`FrecSensor`: Es la frecuencia de actualización de los sensores: (3), (5) o (10) segundos.

9- Task4: Es la tarea encargada de iniciar la petición de conversión a los sensores que se encuentran conectados en el bus y mostrar los valores de temperatura (Solo se visualizan los valores de temperatura si el menú se encuentra inactivo).

Esta tarea se ejecuta cada 3, 5, o 10 (se inicializa en 3s), dato almacenado en la variable `frecSensor`.

Se verifican 3 MailBox que contienen la configuración seleccionada en el menú de opciones:

- `EscaleMbox`: contiene la frecuencia de actualización de los sensores.
- `ModeMbox`: guarda el modo de visualización de los sensores.
- `FrecMbox`: es la frecuencia de actualización.

10- TaskStat: Esta tarea corre una vez por segundo y solamente pone a cero la variable `OSCtxSwCtr`, la cual lleva la cuenta de los cambios de contexto, por lo que, al reiniciarse cada un segundo, tenemos la cantidad de cambio de contexto por segundo. Que es uno de los datos que serán mostrados de las estadísticas del sistema.

11- TaskBlink: Esta tarea se ejecuta cada 1 segundo para cambiar el estado del led Built In de la placa, es el indicador de “sistema vivo”.

Como se mencionó antes se cuenta con un encoder rotativo para acceder al menú de opciones el cual dispone de las siguientes a saber.

`Root Menu`: menú principal, contiene las siguientes opciones figura 3a y 3b:

- 1) `Escala`: permite seleccionar la escala de la temperatura y cuenta con las siguientes opciones mostradas en la figura 3c:
 - a. `°C`: muestra en grados Celsius.
 - b. `°F`: muestra en grados Fahrenheit.
 - c. `<Back`: volver al nivel de menú principal.
- 2) `Visualizacion`: permite seleccionar entre los siguientes modos de visualización de las temperaturas que pueden verse en la figura 3d:
 - a. `T1 y T2`: Se muestra la Temp. de los sensores 1 y 2.
 - b. `T3 y T4`: Se muestra la Temp. de los sensores 3 y 4.
 - c. `<Back`: volver al nivel de menú principal.



NOMBRE	LEGAJO	CURSO	FECHA	HOJAS
Alejandro Galfrascoli	43029	6R3	09/08/2021	6 de 21

- 3) **Actualizacion:** permite seleccionar la frecuencia de adquisición de la temperatura y dispone de las siguientes opciones vistas en la figura 3e:
- 1 Segundo: Cada 1 seg.
 - 3 Segundos: Cada 3 seg.
 - 5 Segundos: Cada 5 seg.
 - 10 Segundos: Cada 10 seg.
 - <Back: volver al nivel de menú principal.
- 4) **Info. Sistema:** Muestra las estadísticas del sistema como se ve en la figura 3f, es una instantánea de los valores actuales de:
- Uso del CPU en %.
 - Número de tareas.
 - Cambios de contexto por segundo.
 - versión de uCO-II.
- 5) **Test por COM:** Habilita la comunicación vía RS-232 y accede a las opciones de diagnóstico de hardware ya descritas.



Figura 3a



Figura 3b



Figura 3c



Figura 3d

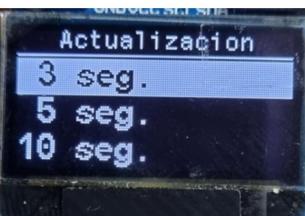


Figura 3e



Figura 3f

La figura 4 muestra el logotipo de inicio de la aplicación.



Figura 4



NOMBRE	LEGAJO	CURSO	FECHA	HOJAS
Alejandro Galfrascoli	43029	6R3	09/08/2021	7 de 21

A continuación, se muestran los módulos más relevantes del firmware.

app_cfg.h

```
/*
***** TASK PRIORITIES *****
*/
#define APP_CFG_STARTUP_TASK_PRIO      20u    /* Tarea para inicializar al resto */
#define APP_CFG_TASK1_PRIO             14u    /* Lectura del Teclado cada 500ms */
#define APP_CFG_TASK2_PRIO             5u     /* Manejo del Encoder Rotativo */
#define APP_CFG_TASK3_PRIO             7u     /* Gestion del menu de opciones */
#define APP_CFG_TASK4_PRIO             9u     /* GUI y lectura de sensores */
#define APP_CFG_TASK_STAT_PRIO         11u    /* estadisticas del sistema operativo */
#define APP_CFG_TASK_BLINK_PRIO        19u    /* Blink Led Built-In cada 1s */

#define OS_TASK_TMR_PRIO              (OS_LOWEST_PRIO - 2u)

/*
***** TASK STACK SIZES *****
*
*           Size of the task stacks (# of OS_STK entries)
*/
#define APP_CFG_STARTUP_TASK_STK_SIZE  280u   /* Stk medidos con OSTaskStkChk() */
#define APP_CFG_TASK1_STK_SIZE         160u   /* stk_data_chk.OSUsed: 51 */
#define APP_CFG_TASK2_STK_SIZE         80u    /* stk_data_chk.OSUsed: 160 */
#define APP_CFG_TASK3_STK_SIZE         256u   /* stk_data_chk.OSUsed: 47 */
#define APP_CFG_TASK4_STK_SIZE         196u   /* stk_data_chk.OSUsed: 156 */
#define APP_CFG_TASK_STAT_STK_SIZE    60u    /* stk_data_chk.OSUsed: 160 */
#define APP_CFG_TASK_BLINK_STK_SIZE   48u    /* stk_data_chk.OSUsed: 40 */

/* stk_data_chk.OSUsed: 38 */
```

tareas.h

```
/*
* ***** Este archivo contiene las definiciones de las tareas necesarias para la aplicacion de usuario *****
*
* Filename:    tareas.h
* Author:      Alejandro Galfrascoli
*/
#ifndef TAREAS_H
#define TAREAS_H

/*
***** CONFIGURATION CONSTANTS *****
*/
#define MENU_OUT_EN          1      // Habilitacion de salida del menu al seleccionar una opcion
#define MENU_OUT_AUTO         1000   // Tiempo de salida automatica del menu en milisegundos
#define LOGO_EN               1      // Habilitacion del logotipo de inicio
#endif
```



NOMBRE	LEGAJO	CURSO	FECHA	HOJAS
Alejandro Galfrascoli	43029	6R3	09/08/2021	8 de 21

```
*
*****
PROTOTYPES
*****
*/
void StartupTask (void *p_arg); /* Crea el resto de las tareas de App y eventos */
void Task1 (void *p_arg); /* Realiza el test de hardware por puerto serial */
void Task2 (void *p_arg); /* Gestiona el encoder rotativo */
void Task3 (void *p_arg); /* Gestiona el menu de opciones con gui */
void Task4 (void *p_arg); /* presentacion de datos de los sensores */

void TaskBlink (void *p_arg); /* Blink Built-in Led */
void TaskStat (void *p_arg); /* Estadisticas del sistema */

void App_TaskCreate (void); /* Creacion de las tareas de la App */
void App_EventCreate(void); /* Creacion de los eventos de la App */
void App_InitModules(void); /* Inicializacion de Modulos HAL */

void EdgeAFnct(void *arg); /* Callback para la deteccion de flancos en EncA */
void EdgeButFnct(void *arg); /* Callback para la deteccion de flancos en button*/
void menuCom(void); /* Despliega las opciones de menu por COM */
void TestOLED(void); /* Grafica animaciones del tipo geometricas */
void SensorTable(void); /* Despliega la tabla de sensores actual */
/*
*****
DEFINICION DE STACKS
*****
*/
OS_STK StartupTaskStk[APP_CFG_STARTUP_TASK_STK_SIZE];
OS_STK Task1Stk[APP_CFG_TASK1_STK_SIZE];
OS_STK Task2Stk[APP_CFG_TASK2_STK_SIZE];
OS_STK Task3Stk[APP_CFG_TASK3_STK_SIZE];
OS_STK Task4Stk[APP_CFG_TASK4_STK_SIZE];
OS_STK TaskBlinkStk[APP_CFG_TASK_BLINK_STK_SIZE];
OS_STK TaskStatStk[APP_CFG_TASK_STAT_STK_SIZE];
/*
*****
DEFINICIONES PARA EVENTOS
*****
*/
// FLAG para control de estados (del encoder y menu)
OS_FLAG_GRP *flag_grp; // Declaro puntero a estructura tipo Event Flag

/* MASCARAS PARA EL CONTROL DE EVENTO FLAG */
#define ENCA 0x01 // flanco descendente - Encoder señal A
#define E BUT 0x02 // flanco descendente - Encoder Button
#define DOWN 0x04 // accion DOWN
#define UP 0x08 // accion UP
#define OK 0x10 // accion OK (Enter/ok/select)
#define COM 0x20 // indica el acceso al test por COM

// Mail Box para enviar la escala de los sensores
OS_EVENT *EscaleMbox;

// Mail Box para enviar el modo de visualizacion de los sensores
OS_EVENT *ModeMbox;

// Mail Box para enviar la frecuencia de actualizacion de los sensores
OS_EVENT *FrecMbox;

/*
*****
*/
#endif /* TAREAS_H */
```



NOMBRE	LEGAJO	CURSO	FECHA	HOJAS
Alejandro Galfrascoli	43029	6R3	09/08/2021	9 de 21

```
/*
 * ****
 * Este archivo contiene las definiciones de las tareas necesarias para la aplicacion de usuario.
 *
 * Filename:    tareas.c
 * Author:      Alejandro Galfrascoli
 *
 * En las siguientes tareas se hace uso de Eventos tipo Flag , Mbox y Semaforo para la sincronizacion de
 * tareas y asegurar el acceso exclusivo a recursos del microcontrolador.
 * Se hace uso del modulo Discreto de Entrada Salida DIO.
 *
 * ucos-II:      v2.92.16
 * Asignacion de pines:
 *
 ****
 */
/*
 ****
 *          INCLUDE FILES
 ****
 */
#include "includes.h"
/*
 ****
 *          VARIABLES
 ****
 */
extern BOOLEAN   Led;                                // Para monitorear el estado del LED
extern INT8U     CtxPerSec;                          // Cambios de contexto por segundo
extern char      *ErrStr;                            // Str para debug de errores en eventos o funciones

extern GUI_MENU  guiMenu;                           // Estructuraa del menu
extern Ds18b20Sensor_t ds18b20[DS18B20_MAX_SENSORS]; // tabla de sensores
//-----
uint8_t mjeCom0[] =      "\r\n\n### Test por COM ###\r\n";
uint8_t mjeCom1[] =      "[W]-> Button Up\r\n";
uint8_t mjeCom2[] =      "[S]-> Button Down\r\n";
uint8_t mjeCom3[] =      "[D]-> Button OK\r\n";
uint8_t mjeCom4[] =      "[P]-> Test OLED\r\n";
uint8_t mjeCom5[] =      "[T]-> Sensor Table\r\n";
uint8_t mjeCom6[] =      "[H]-> Menu Options\r\n";
uint8_t mjeCom7[] =      "[X]-> Exit\r\n";

uint8_t mjeComSalida[] = "\r\nFinish Test COM..."; // 18 caracteres max
uint8_t mjeComST[] =     "\r\nSensorTable()"; // 14 caracteres max
uint8_t mjeComTO[] =     "\r\nTestOLED()"; // 14 caracteres max
uint8_t mjeComOK[] =     "\r\nOK"; // 14 caracteres max
uint8_t mjeComUP[] =     "\r\nUP"; // 14 caracteres max
uint8_t mjeComDW[] =     "\r\ndOWN"; // 14 caracteres max

// textos para los menus
static char itmBack[] = "< Back"; // 14 caracteres max

static char itmRoot[] = "Root menu"; // 18 caracteres max
static char itm_submenu1[] = "Escala"; // 14 caracteres max
static char itm_submenu2[] = "Visualizacion"; // 14 caracteres max
static char itm_submenu3[] = "Actualizacion"; // 14 caracteres max
static char itm_submenu4[] = "Inf. Sistema"; // 14 caracteres max
static char itm_submenu5[] = "Test por COM"; // 14 caracteres max

static char itmGC[] = "Celsius"; // 14 caracteres max
static char itmGF[] = "Fahrenheit"; // 14 caracteres max
static char itmT12[] = "T1 y T2"; // 14 caracteres max
static char itmT34[] = "T3 y T4"; // 14 caracteres max
```



NOMBRE	LEGAJO	CURSO	FECHA	HOJAS
Alejandro Galfrascoli	43029	6R3	09/08/2021	10 de 21

```
static char itm1s[] = " 1 seg.";                                // 14 caracteres max
static char itm3s[] = " 3 seg.";                                // 14 caracteres max
static char itm5s[] = " 5 seg.";                                // 14 caracteres max
static char itm10s[] = "10 seg.";                               // 14 caracteres max
//-----
//menús: el primer elemento es el título del menú y no cuenta para cnt
//-----
static char* mnuRoot[] = {itmRoot,
                         itm_submenu1,itm_submenu2,itm_submenu3,itm_submenu4,itm_submenu5,itmBack};
uint8_t cntRoot = 6;

static char* mnu_submenu1[] = {itm_submenu1,
                             itmGC,itmGF,itmBack};           // Escala
uint8_t cnt_submenu1 = 3;

static char* mnu_submenu2[] = {itm_submenu2,
                             itmT12,itmT34,itmBack};         // Visualizacion
uint8_t cnt_submenu2 = 3;

static char* mnu_submenu3[] = {itm_submenu3,
                             itm1s,itm3s,itm5s,itm10s,itmBack}; // Actualizacion
uint8_t cnt_submenu3 = 5;

char* mnu_submenu4[] = {itm_submenu4
                      };                           // Info. Sistema
uint8_t cnt_submenu4 = 0;

char* mnu_submenu5[] = {itm_submenu5
                      };                           // Test UART
uint8_t cnt_submenu5 = 0;

/*
*****
*          StartupTask()
*
* Description : The startup task. The uC/OS-II ticker should only be initialize once multitasking starts.
*
*           Inicializa estadisticas (Si estan habilitadas), Eventos, y ademas crea las restantes tareas
*           con el llamado a App_TaskCreate().
*           App_InitModules() inicializacion de Modulos.
*           App_EventCreate() Crea los eventos para las tareas.
*
* Argument(s) : p_arg      Argument passed to 'App_TaskStart()' by 'OSTaskCreate()'.
* Return(s)   : none.
* Caller(s)   : This is a task.
* Note(s)     : Esta tarea se elimina asi misma una vez que se ejecuta
*****
*/
void StartupTask (void *p_arg)
{
    CPU_INT32U cpu_clk;
    (void)p_arg;

    cpu_clk = HAL_RCC_GetHCLKFreq();                                /* Initialize and enable System Tick timer */
    OS_CPU_SysTickInitFreq(cpu_clk);

    #if (OS_TASK_STAT_EN > 0)
        OSStatInit();                                         /* Determine CPU capacity. */
    #endif
//*****
    App_EventCreate();                                         /* Create application events. */
    App_InitModules();                                         /* Inicializacion de Modulos HAL. */
    App_TaskCreate();                                          /* Create application tasks. */
//*****
    while (DEF_TRUE) {
        OSTaskSuspend(APP_CFG_TASK1_PRIO);                     /* Task 1 inicia suspendida, se habilita por menu */
        OSTaskDel(APP_CFG_STARTUP_TASK_PRIO);                  /* elimino al tarea de inicio una vez que se ejecutó */
    }
}
```



NOMBRE	LEGAJO	CURSO	FECHA	HOJAS
Alejandro Galfrascoli	43029	6R3	09/08/2021	11 de 21

```
}

/*
***** Task1()
*
* Description : Verificación de teclas presionadas. El carácter es recibido por la UART. las opciones
*                disponibles son las siguientes:
*                D      -> Select/OK
*                W      -> UP
*                S      -> DOWN
*                P      -> Test OLED
*                T      -> Sensor Table
*                H      -> Help
*                X      -> Exit
*
* Argument(s) : p_arg      Argument passed to 'Task1()' by 'OSTaskCreate()'.
* Return(s)   : none.
* Caller(s)   : This is a task.
* Note(s)    : Esta tarea se ejecuta cada 500ms para verificar si hay teclas presionadas.
*               Esta tarea se encuentra suspendida hasta ser habilitada desde el menu de opciones.
*               Tambien se suspende asi misma si es seleccionada la opcion 'X'.
*****
*/
void Task1 (void *p_arg) {
(void)p_arg;
INT8U err;                                // para devolucion de errores
INT8U D[1];                                // longitud de caracteres recibidos
OS_FLAGS flags_rdy;

OS_STK_DATA stk_data_chk;                  // para verificar el Stack
INT32U stk_size_chk;
INT8U err_chk;

while(DEF_TRUE) {

    flags_rdy = OSFlagAccept (flag_grp,(OS_FLAGS ) COM, OS_FLAG_WAIT_SET_ALL | OS_FLAG_CONSUME, &err);
    if((flags_rdy & COM) == (COM)) {
        menuCom();
        COMGets(D, 1);                         // limpia buffer
    }

    if(COMGets(D, 1) == HAL_OK) {
        D[1]=0;                                /* Pongo un "0" al final del vector */

        if(D[0]=='d' || D[0]=='D') {           /* Opcion Select/OK del Encoder */
            COMPuts(mjeComOK);
            OSFlagPost (flag_grp, (OS_FLAGS) OK, OS_FLAG_SET, &err);
                                         /* seteo el bit OK de flag group */
        }
        if(D[0]=='w' || D[0]=='W') {           /* Opcion UP del Encoder */
            COMPuts(mjeComUP);
            OSFlagPost (flag_grp, (OS_FLAGS) UP, OS_FLAG_SET, &err);
                                         /* seteo el bit UP de flag group */
        }
        if(D[0]=='s' || D[0]=='S') {           /* Opcion DOWN del Encoder */
            COMPuts(mjeComDW);
            OSFlagPost (flag_grp, (OS_FLAGS) DOWN, OS_FLAG_SET, &err);
                                         /* seteo el bit DOWN de flag group */
        }
        if(D[0]=='p' || D[0]=='P') {           /* Opcion Test OLED */
            COMPuts(mjeComTO);
            TestOLED();                         /* ejecuta prueba de la pantalla LCD */
            guiShowMenu();
        }
        if(D[0]=='t' || D[0]=='T') {           /* Opcion Sensor Table*/
            COMPuts(mjeComST);
        }
    }
}
```



NOMBRE	LEGAJO	CURSO	FECHA	HOJAS
Alejandro Galfrascoli	43029	6R3	09/08/2021	12 de 21

```
SensorTable();                                /* visualiza la tabla de sensores actual */
}
if(D[0]=='h' || D[0]=='H') {
    menuCom();                                /* Opcion Help */
    /* despliega el menu de opciones */
}
if(D[0]=='x' || D[0]=='X') {
    COMPuts(mjeComSalida);
    guiFinishMenu();
    OSTaskSuspend(APP_CFG_TASK1_PRIO);        /* Task1 se suspende asi misma, se
habilita desde el menu */
}

} // fin if(HAL_UART_Receive)

//-----
err_chk = OSTaskStkChk(APP_CFG_TASK1_PRIO, &stk_data_chk);      // verificacion del stack
if (err_chk == OS_ERR_NONE) {
    stk_size_chk = stk_data_chk.OSFree + stk_data_chk.OSUsed;
    stk_size_chk += 0;
}
//-----

OSTimeDlyHMSM(0u, 0u, 0u, 500u);
}

/*
*****
* Task2()
*
* Description : Esta tarea es la encargada de manejar el encoder rotativo
*
* Argument(s) : p_arg      Argument passed to 'Task2()' by 'OSTaskCreate()'.
* Return(s)   : none.
* Caller(s)   : This is a task.
* Note(s)     : Esta tarea se ejecuta cada vez que se activan cualquiera de los sig. flags:
*                 ENCA: deteccion de flanco descendente en pin Encoder A.
*                 EBUT: deteccion de flanco descendente en pin Button.
*
*                 Los flags ENCA y EBUT son seteados en EdgeAFnct() y EdgeButFnct() que son las funciones
*                 callback de deteccion de flancos del modulo DIO
*****
*/
void Task2 (void *p_arg)
{
    (void)p_arg;
    INT8U err;
    OS_FLAGS flags_rdy;

    DICfgMode(0, DI_MODE_EDGE_LOW_GOING);           // deteccion de flancos descendentes en EncA
    DICfgEdgeDetectFnct(0, EdgeAFnct, (void *)flag_grp); // EdgeAFnct: callback para señalizar los flag

    DICfgMode(1, DI_MODE_DIRECT);                   // pin de EncB

    DICfgMode(3, DI_MODE_EDGE_LOW_GOING);           // pin de Button
    DICfgEdgeDetectFnct(3, EdgeButFnct, (void *)flag_grp); // EdgeButFnct: callback para señalizar los flag
    DIClr(0);                                     // borro el num de transiciones

    while (DEF_TRUE)
    {
        flags_rdy = OSFlagPend (flag_grp,(OS_FLAGS ) ENCA | EBUT, OS_FLAG_WAIT_SET_ANY | OS_FLAG_CONSUME, 0,
&err);
        if ((flags_rdy & ENCA) == (ENCA)) {
            if(DIGet(1)) {
                OSFlagPost (flag_grp, (OS_FLAGS) UP, OS_FLAG_SET, &err);          /* seteo el bit UP de
flag group */
            }
        }
    }
}
```



NOMBRE	LEGAJO	CURSO	FECHA	HOJAS
Alejandro Galfrascoli	43029	6R3	09/08/2021	13 de 21

```
        else {
            OSFlagPost (flag_grp, (OS_FLAGS) DOWN, OS_FLAG_SET, &err); /* seteo el bit DOWN de
flag group */
        }

        DIClr(0); // borro el num de transiciones
    }

    if ((flags_rdy & EBUT) == (EBUT)) {
        OSFlagPost (flag_grp, (OS_FLAGS) OK, OS_FLAG_SET, &err);/* seteo el bit OK de flag group */
    }
//    OSTimeDlyHMSM(0u, 0u, 0u, 50u);
}

/*
*****
* Task3()
*
* Description : Gestiona la presentacion del menu en la pantalla OLED
*
* Argument(s) : p_arg      Argument passed to 'Task3()' by 'OSTaskCreate()'.
* Return(s)   : none.
* Caller(s)   : This is a task.
* Note(s)    : Esta tarea se ejecuta cada vez que se activan cualquiera de los sig. flags, los cuales
son consumidos en la llamada:
*               DOWN: giro en sentido horario del encoder.
*               UP:     giro en sentido antihorario del encoder.
*               OK:    pulsación del boton del encoder.
*
* Esta tarea tambien envia MailBox a la Task4() con la configuracion de:
* ModeSensor:      Es el modo de visualizacion de los sensores en lcd: 1, 2.
* FrecSensor:       Es la frecuencia de actualizacion de los sensores: 3, 5 o 10 segundos.
* EscalaSensor:    Escala de lectura Celsius (1) o Fahrenheit (2).
*****
*/
void Task3 (void *p_arg) {
(void)p_arg;
INT8U err;
OS_FLAGS flags_rdy;
uint8_t clickedItem; // indice del item cliqueado
uint8_t EscalaSensor = 1; // Escala en grados Celsius
uint8_t ModeSensor = 1; // Modo 1 de visualizacion
uint8_t FrecSensor = 3; // Frecuencia de inicio: 3 segundos

OS_STK_DATA stk_data_chk; // para verificar el Stack en CubeMonitor
INT32U stk_size_chk;
INT8U err_chk;

if(LOGO_EN) {
    guiBitmap(); // visualizar logotipo de inicio
}

while (DEF_TRUE) {

    flags_rdy = OSFlagPend (flag_grp,(OS_FLAGS ) DOWN | UP | OK, OS_FLAG_WAIT_SET_ANY | OS_FLAG_CONSUME, 0,
&err);
    if((flags_rdy & DOWN) == (DOWN)) /* por DOWN... */
        if(guiMenu.active) /* si estoy dentro del menu */
            guiPress(DOWN); /* informo a guiPress() un down */

    if((flags_rdy & UP) == (UP)) /* Por UP... */
        if(guiMenu.active) /* si estoy dentro del menu */
            guiPress(UP); /* informo a guiPress() un UP */

    if((flags_rdy & OK) == (OK)) {
        clickedItem = guiPress(OK);
        if(!guiMenu.active) { /* si presiono ok lo informo a guiPress(), devuelve item seleccionado */
            /* si estoy fuera del menu */
        }
    }
}
}
```



NOMBRE	LEGAJO	CURSO	FECHA	HOJAS
Alejandro Galfrascoli	43029	6R3	09/08/2021	14 de 21

```
        guiInitMenu(mnuRoot, cntRoot, 1);           /* lo inicio en nivel Root */
        clickedItem = 0;
    }

    if(clickedItem > 0) {
        if(guiMenu.CurrentMenu == mnuRoot)           /* ### Root Menu ### */
            switch(clickedItem) {
                case 1: guiInitMenu(mnu_submenu1, cnt_submenu1, 1);      /* cargo el submenu
correspondiente */
                    break;
                case 2: guiInitMenu(mnu_submenu2, cnt_submenu2, 1);
                    break;
                case 3: guiInitMenu(mnu_submenu3, cnt_submenu3, 1);
                    break;
                case 4: guiStat();                                /* mostrar estadisticas */
                    break;
                case 5: guiMessageBox("Iniciando UART");          /* arrancar tarea para uart */
OSFlagPost (flag_grp, (OS_FLAGS) COM, OS_FLAG_SET, &err);      /* seteo el bit
COM de flag group */
                    OSTaskResume(APP_CFG_TASK1_PRIO);           /* Reanudo la task1, para test Uart */
                    break;
                case 6: guiFinishMenu();                         // salir del menu
                    break;
            }
        else if(guiMenu.CurrentMenu == mnu_submenu1) {        /* ### Submenu 1: Escala ### */
            switch (clickedItem) {
                case 1: guiMessageBox("Grados: *C");
                    EscaleSensor = 1;
                    break;
                case 2: guiMessageBox("Grados: *F");
                    EscaleSensor = 2;
                    break;
                case 3: guiInitMenu(mnuRoot, cntRoot, 1);           //back
                    break;
            }
            err = OSMboxPost(EscaleMbox, (void *) &EscaleSensor); // Config de Escala a task4
            if(err != OS_ERR_NONE)
                ErrStr = "Err OSMboxPost: EscaleMbox " + err;

            if(MENU_OUT_EN & (clickedItem != 3)) {
                OSTimeDly(MENU_OUT_AUTO);
                guiFinishMenu();                            // salida del menu luego de seleccionar Escala
            }
        }
        else if(guiMenu.CurrentMenu == mnu_submenu2) {        /* ### Submenu 2: Visualizacion ### */
            switch (clickedItem) {
                case 1: guiMessageBox("T1 y T2");
                    ModeSensor = 1;
                    break;
                case 2: guiMessageBox("T3 y T4");
                    ModeSensor = 2;
                    break;
                case 3: guiInitMenu(mnuRoot, cntRoot, 2);           // back
                    break;
            }
            err = OSMboxPost(ModeMbox, (void *) &ModeSensor);      /* Config de Visualizacion a
task4 */
            if(err != OS_ERR_NONE)
                ErrStr = "Err OSMboxPost: ModeMbox " + err;

            if(MENU_OUT_EN & (clickedItem != 3)) {
                OSTimeDly(MENU_OUT_AUTO);
                guiFinishMenu();                            // salida del menu luego de seleccionar Modo
            }
        }
    }
}
```



NOMBRE	LEGAJO	CURSO	FECHA	HOJAS
Alejandro Galfrascoli	43029	6R3	09/08/2021	15 de 21

```
else if(guiMenu.CurrentMenu == mnuSubMenu3)           /* ### Submenu 3: Actualizacion ### */
    switch (clickedItem) {
        case 1: guiMessageBox("1 segundo");
                   FrecSensor = 1;
                   break;
        case 2: guiMessageBox("3 segundos");
                   FrecSensor = 3;
                   break;
        case 3: guiMessageBox("5 segundos");
                   FrecSensor = 5;
                   break;
        case 4: guiMessageBox("10 segundos");
                   FrecSensor = 10;
                   break;
        case 5: guiInitMenu(mnuRoot, cntRoot, 3);          // back
                   break;
    }
    err = OSMboxPost(FrecMbox, (void *) &FrecSensor);      /* enviar configuracion a task4
*/
    if(err != OS_ERR_NONE)
        ErrStr = "Err en OSMboxPost: " + err;

    if(MENU_OUT_EN & (clickedItem != 5)) {
        OSTimeDly(MENU_OUT_AUTO);
        guiFinishMenu();           // salida del menu luego de seleccionar Frecuencia
    }
//-----
    clickedItem = 0;
}      // Fin de if(clickedItem > 0)

//-----
err_chk = OSTaskStkChk(APP_CFG_TASK3_PRIO, &stk_data_chk);      // verificacion del stack
if (err_chk == OS_ERR_NONE) {
    stk_size_chk = stk_data_chk.OSFree + stk_data_chk.OSUsed;
    stk_size_chk += 0;
}
//-----

// OSTimeDlyHMSM(0u, 0u, 0u, 50u);
}

/*
*****
*                               Task4()
*
* Description : Verifica el 3 MailBox que contienen la configuracion seleccionada en el menu de opciones
* y ademeas realiza la lectura de los sensores 1-Wire.
*
*      EscaleMbox: contiene la frecuencia de actualizacion de los sensores.
*      ModeMbox: contiene el modo de visualizacion de los sensores.
*      FrecMbox: contiene la frecuencia de actualizacion.
*
* Argument(s) : p_arg      Argument passed to 'Task4()' by 'OSTaskCreate()'.
* Return(s)   : None.
* Caller(s)   : This is a task.
* Note(s)     : Esta tarea se ejecuta cada 'frecSensor' segundos (3, 5, o 10), Se inicializa en 3s.
*               Solo se visualizan los resultados de temperatura si el menu no se encuentra activo.
*****
*/
void Task4 (void *p_arg) {
    (void)p_arg;
    uint8_t escaleSensor = 1;
    uint8_t modeSensor = 1;
    uint8_t freqSensor = 3;
```



NOMBRE	LEGAJO	CURSO	FECHA	HOJAS
Alejandro Galfrascoli	43029	6R3	09/08/2021	16 de 21

```
void *msg;

OS_STK_DATA stk_data_chk; // para verificar el Stack en CubeMonitor
INT32U stk_size_chk;
INT8U err_chk;

while(DEF_TRUE)
{
    OSTimeDlyHMSM(0u, 0u, freqSensor, 0u);
    //-----
    msg = OSBoxAccept(EscaleMbox); // si hay un msg, es el nuevo valor de escala
    if(msg != (void *)0)
        escaleSensor = *(uint8_t *) msg;
    //-----
    msg = OSBoxAccept(ModeMbox); /* si hay un msg, es el nuevo valor de visualizacion */
    if(msg != (void *)0)
        modeSensor = *(uint8_t *) msg;
    //-----
    msg = OSBoxAccept(FreqMbox); /* si hay un msg, es el nuevo valor de actualizacion */
    if(msg != (void *)0)
        freqSensor = *(uint8_t *) msg;
    //-----

    DS18B20_Del(ds18b20); /* limpiar la tabla de sensores */
    DS18B20_Init(0); /* inicializar la tabla de sensores */
    DS18B20_StartAll(); /* lectura del ScrtchPad de todos los sensores */
    DS18B20_ReadAll();

    if(!guiMenu.active) /* si el menu no esta activo visualizar... */
        guiTemp(escaleSensor, modeSensor);

    //-
    err_chk = OSTaskStkChk(APP_CFG_TASK4_PRIO, &stk_data_chk); // verificacion del stack
    if (err_chk == OS_ERR_NONE) {
        stk_size_chk = stk_data_chk.OSFree + stk_data_chk.OSUsed;
        stk_size_chk += 0;
    }
    //-----
}

/*
*****
* TaskStat()
*
* Description : Estadisticas del Sistema Operativo.
* Argument(s) : p_arg      Argument passed to 'TaskStat()' by 'OSTaskCreate()'.
* Return(s)   : none.
* Caller(s)   : This is a task.
* Note(s)     : Esta tarea se ejecuta cada 1seg para resetear la variable OSCtxSwCtr, por lo que
*                 su ultimo valor indica la cantidad de cambios de contexto por segundo y se guarda en
*                 CtxPerSec para su uso en impresion de estadisticas.
*****
*/
void TaskStat (void *p_arg) {
    (void)p_arg;

    while (DEF_TRUE) {
        CtxPerSec = OSCtxSwCtr;
        OSCtxSwCtr = 0;
        OSTimeDlyHMSM(0u, 0u, 1u, 0u);
    }
}
/*
*****
* TaskBlink()
*
*
```



NOMBRE	LEGAJO	CURSO	FECHA	HOJAS
Alejandro Galfrascoli	43029	6R3	09/08/2021	17 de 21

```
* Description : Led indicador de sistema "Vivo".
* Argument(s) : p_arg      Argument passed to 'TaskBlink()' by 'OSTaskCreate()'.
* Return(s)   : none.
* Caller(s)   : This is a task.
* Note(s)     : Esta tarea se ejecuta cada 1s para invertir el estado del Built in LED.
*****
*/
void TaskBlink (void *p_arg) {
(void)p_arg;

while (DEF_TRUE) {
    HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13);
    Led = HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13);                                /* variable Led para monitoreo */

    OSTimeDlyHMSM(0u, 0u, 1u, 0u);
}

/*
*****
*          App_TaskCreate()
*
* Description : llamada en StartupTask() para la creación de las restantes tareas de la App.
* Argument(s) : None.
* Return(s)   : None.
* Caller(s)   :
* Note(s)     : none.
*****
*/
void App_TaskCreate (void)
{
CPU_INT08U      os_err;

os_err = OSTaskCreateExt( (void (*)(void *))Task1,
                         (void *) 0,
                         (OS_STK) &Task1Stk[APP_CFG_TASK1_STK_SIZE - 1],
                         (INT8U) APP_CFG_TASK1_PRIO,
                         (INT16U) APP_CFG_TASK1_PRIO,
                         (OS_STK) &Task1Stk[0],
                         (INT32U) APP_CFG_TASK1_STK_SIZE,
                         (void *) 0,
                         (INT16U) (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR));

#if (OS_TASK_NAME_EN > 0u)
    OSTaskNameSet( APP_CFG_TASK1_PRIO, (INT8U *)"Task1", &os_err);
#endif
//*****
os_err = OSTaskCreateExt( (void (*)(void *))Task2,
                         (void *) 0,
                         (OS_STK) &Task2Stk[APP_CFG_TASK2_STK_SIZE - 1],
                         (INT8U) APP_CFG_TASK2_PRIO,
                         (INT16U) APP_CFG_TASK2_PRIO,
                         (OS_STK) &Task2Stk[0],
                         (INT32U) APP_CFG_TASK2_STK_SIZE,
                         (void *) 0,
                         (INT16U) (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR));

#if (OS_TASK_NAME_EN > 0u)
    OSTaskNameSet( APP_CFG_TASK2_PRIO, (INT8U *)"Task2", &os_err);
#endif
//*****
os_err = OSTaskCreateExt( (void (*)(void *))Task3,
                         (void *) 0,
                         (OS_STK) &Task3Stk[APP_CFG_TASK3_STK_SIZE - 1],
                         (INT8U) APP_CFG_TASK3_PRIO,
                         (INT16U) APP_CFG_TASK3_PRIO,
                         (OS_STK) &Task3Stk[0],
                         (INT32U) APP_CFG_TASK3_STK_SIZE,
                         (void *) 0,
```



NOMBRE	LEGAJO	CURSO	FECHA	HOJAS
Alejandro Galfrascoli	43029	6R3	09/08/2021	18 de 21

```
(INT16U          ) (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR));  
#if (OS_TASK_NAME_EN > 0u)  
    OSTaskNameSet( APP_CFG_TASK3_PRIO, (INT8U *)"Task3", &os_err);  
#endif  
//*****  
os_err = OSTaskCreateExt( (void (*)(void *))Task4,  
                         (void          *) 0,  
                         (OS_STK        *) &Task4Stk[APP_CFG_TASK4_STK_SIZE - 1],  
                         (INT8U         ) APP_CFG_TASK4_PRIO,  
                         (INT16U        ) APP_CFG_TASK4_PRIO,  
                         (OS_STK        *) &Task4Stk[0],  
                         (INT32U        ) APP_CFG_TASK4_STK_SIZE,  
                         (void          *) 0,  
                         (INT16U        ) (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR));  
  
#if (OS_TASK_NAME_EN > 0u)  
    OSTaskNameSet( APP_CFG_TASK4_PRIO, (INT8U *)"Task4", &os_err);  
#endif  
//*****  
os_err = OSTaskCreateExt( (void (*)(void *))TaskBlink,  
                         (void          *) 0,  
                         (OS_STK        *) &TaskBlinkStk[APP_CFG_TASK_BLINK_STK_SIZE -  
1],  
                         (INT8U         ) APP_CFG_TASK_BLINK_PRIO,  
                         (INT16U        ) APP_CFG_TASK_BLINK_PRIO,  
                         (OS_STK        *) &TaskBlinkStk[0],  
                         (INT32U        ) APP_CFG_TASK_BLINK_STK_SIZE,  
                         (void          *) 0,  
                         (INT16U        ) (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR));  
  
#if (OS_TASK_NAME_EN > 0u)  
    OSTaskNameSet( APP_CFG_TASK_BLINK_PRIO, (INT8U *)"Task Blink", &os_err);  
#endif  
//*****  
os_err = OSTaskCreateExt( (void (*)(void *))TaskStat,  
                         (void          *) 0,  
                         (OS_STK        *) &TaskStatStk[APP_CFG_TASK_STAT_STK_SIZE - 1],  
                         (INT8U         ) APP_CFG_TASK_STAT_PRIO,  
                         (INT16U        ) APP_CFG_TASK_STAT_PRIO,  
                         (OS_STK        *) &TaskStatStk[0],  
                         (INT32U        ) APP_CFG_TASK_STAT_STK_SIZE,  
                         (void          *) 0,  
                         (INT16U        ) (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR));  
  
#if (OS_TASK_NAME_EN > 0u)  
    OSTaskNameSet( APP_CFG_TASK_STAT_PRIO, (INT8U *)"TaskStat", &os_err);  
#endif  
//*****  
}  
  
/*  
*****  
*          App_EventCreate()  
*  
* Description : llamada en StartupTask() para la creación de los Eventos en la App.  
* Argument(s) : None.  
* Return(s)   : None.  
* Caller(s)   :  
* Note(s)     : none.  
*****  
*/  
void App_EventCreate(void) {  
    INT8U err;  
  
    flag_grp = OSFlagCreate ((OS_FLAGS) 0, // el valor inicial del grupo de Flags de eventos  
                           (INT8U *) &err); // tipo de error  
  
    if(flag_grp == (void *)0)           /* verifico la creacion del Flag */  
        ErrStr = "Err en OSFlagCreate";  
  
    #if OS_FLAG_NAME_EN > 0u
```



NOMBRE	LEGAJO	CURSO	FECHA	HOJAS
Alejandro Galfrascoli	43029	6R3	09/08/2021	19 de 21

```
OSFlagNameSet (flag_grp, (INT8U *)"DIO Flag", &err);
#endif
//-----
EscaleMbox = OSMboxCreate((void *)NULL);           /* Mbox para modo de escala de los sensores */
if(EscaleMbox == (void *)0)                         /* verifico la creacion del Mbox */
    ErrStr = "Err en EscaleMbox Create";
//-----
ModeMbox = OSMboxCreate((void *)NULL);             /* Mbox para modo de Visualizacion de los sensores */
if(ModeMbox == (void *)0)
    ErrStr = "Err en ModeMbox Create";
//-----
FrecMbox = OSMboxCreate((void *)NULL);             /* Mbox para freq de actualizacion de los sensores */
if(FrecMbox == (void *)0)
    ErrStr = "Err en FrecMbox Create";
//-----

}

/*
*****
*          App_InitModules()
*
* Description : llamada en StartupTask() para la inicializacion de Modulos HAL.
* Argument(s) : None.
* Return(s)   : None.
* Caller(s)   :
* Note(s)     : none.
*****
*/
void App_InitModules(void) {

    #if MODULE_DIO
        DIOInit();      /* Inicializacion del modulo DIO */
    #endif
    COMInit();          /* Inicializacion del modulo de Comunicaciones UART */
    guiInit();          /* Inicializacion del modulo GUI con la pantalla OLED SSD1306 */
    DS18B20_Init(1);   /* InitFull = 1: Inicializacion del modulo de sensores 18B20 y bus 1-Wire */
}
/*
*****
*          EdgeAFnct()
*
* Description : Señala con ENCA cada vez que se detecta un flanco descendente en EncA.
* Argument(s) : El puntero al flag_grup.
* Return(s)   : None.
* Caller(s)   :
* Note(s)     : Es usada como callback en la deteccion de flancos de la DIOTask para poder señalar a la
*                  Task2 la ocurrencia de una transicion.
*****
*/
void EdgeAFnct(void *arg) {
    INT8U err;
    OSFlagPost (flag_grp, (OS_FLAGS) ENCA, OS_FLAG_SET, &err);      /* seteo el bit ENCA de flag group */
}
/*
*****
*          EdgeButFnct()
*
* Description : Señala con EBUT cada vez que se detecta un flanco descendente en boton del encoder.
* Argument(s) : El puntero al flag_grup.
* Return(s)   : None.
* Caller(s)   :
* Note(s)     : Es usada como callback en la deteccion de flancos de la DIOTask para poder señalar a la
*                  Task2 la ocurrencia de una transicion.
*****
*/
void EdgeButFnct(void *arg) {
    INT8U err;
```



NOMBRE	LEGAJO	CURSO	FECHA	HOJAS
Alejandro Galfrascoli	43029	6R3	09/08/2021	20 de 21

```
OSFlagPost (flag_grp, (OS_FLAGS) EBUT, OS_FLAG_SET, &err);      /* seteo el bit EBUT de flag group */
}
/*
*****
void menuCom(void) {
    COMPutS(mjeCom0);
    COMPutS(mjeCom1);
    COMPutS(mjeCom2);
    COMPutS(mjeCom3);
    COMPutS(mjeCom4);
    COMPutS(mjeCom5);
    COMPutS(mjeCom6);
    COMPutS(mjeCom7);
}
/*
*****
void TestOLED(void) {

#if DRAW_TRIANGLE_EN
    TestTriangles(1);
    OSTimeDly(800);
#endif
#if DRAW_CIRCLE_EN
    TestCircles(8,1);
    OSTimeDly(800);
    TestFilledCircles(8, 1);
    OSTimeDly(800);
#endif
    TestLines(1);
    OSTimeDly(500);
}
/*
*****
void SensorTable(void) {
    uint8_t i,j;
    char tmp[2];
    char ROM_tmp[17] = "";
    char p[3] = ". ";
    char c[7] = "temp: ";
    char t[5] = "";
    char cadena[35] = "\r\nSensores No Conectados";

    if (DS18B20_Quantity()) {
        for(i = 0; i < DS18B20_Quantity(); i++) {

            decToStr((char)i, tmp, 1);
            for(j = 0; j < 8; j++)
                hexToStr(ds18b20[i].Address[j], ROM_tmp + j*2, 2);

            memccpy(memccpy(cadena+2, tmp, '\0',3)-1, p, '\0',6);
            memccpy(memccpy(cadena+5, ROM_tmp, '\0',22)-1, p, '\0',24);
            decToStr(ds18b20[i].Temperature, t, 4);
            memccpy(memccpy(cadena+23, c, '\0',30)-1, t, '\0',35);
            COMPutS((uint8_t*)cadena);
        }
    }
    else
        COMPutS((uint8_t*)cadena);
}
/*
*****
```



NOMBRE	LEGAJO	CURSO	FECHA	HOJAS
Alejandro Galfrascoli	43029	6R3	09/08/2021	21 de 21

Referencias

- [1] <https://www.alldatasheet.es/datasheet-pdf/pdf/230838/DALLAS/DS18B20.html>
- [2] uCOS-II - The Real Time Kernel.pdf
- [3] Embedded Systems Building Blocks.pdf
- [4] Nota de Aplicaciones 187 de Maxim <https://www.maximintegrated.com/en/design/technical-documents/app-notes/1/187.html>

Conclusiones

Al momento de la codificación y pruebas se fue necesario el desarrollo de la librería **ftostr**, ya que se comprobó que con ellas el consumo de memoria flash se incrementaba entre un 30 y 35%, además de una notable reducción en el tiempo de ejecución que se comprobó al momento de debug.

Se Trabajó bastante debugueando el código no solo para depurarlo, sino también para analizar a fondo el funcionamiento del Kernel, tanto en su estructura, como variables, TCB y STK, todo ello fue de gran ayuda para lograr un código robusto.

Se pudo lograr una correcta sincronización entre las tareas utilizando los mecanismos que ofrece el propio RTOS, en este trabajo se hizo uso de Semaforos, MailBox y flags, pero tambié'n se hizo uso de las funciones como OS_ENTER_CRITICAL y OS_EXIT_CRITICAL para las secciones del código que así lo requieren.

Otro punto en que se trabajó mucho fue en comentarios documentando todo el código.

Se presentaron algunos problemas de desborde de stk, algunos de ellos cdificiles de encontrar, pero que la mayoría eran debido a un mal manejo de string por sobrepasar el tamaño declarado.

Otro de los problemas que llevo tiempo depurar era ocacionado por el modulo DIO.