ALCANCES TRABAJO FINAL – RTOS 1 – CESE
R. OLIVA – v 04-04-2019

CONTENIDOS:

1. **Introducción**: La idea general es integrar FreeRTOS al Proyecto que comencé a preparar en 2018 *"Registrador industrial con soporte de placas periféricas"*, que consiste de una placa registradora (CL3) que utiliza un STM32F4 para reemplazo de unas existentes (CL2) que usan AVR. La placa nueva (diseñada por encargo con otro ingeniero de UTN San Nicolás, en 2017), cumple el mismo formato y funciones ampliadas respecto a la anterior. Carecía de firmware por lo cual buena parte del proyecto era realizar funciones similares a la sAPI de Eric Pernia (que es mi director) implementadas en la CIAA. La aplicación principal en que se usan estas placas es un sitio de pruebas del INTI donde se mide una curva de potencia de aerogeneradores de baja potencia, de acuerdo a una normativa IEC. Se implementará una solución parcial bajo FreeRTOS para reemplazar la aplicación actual "lazo infinito" con varias FSMs e interrupciones que corre en un AVRMega1284P (Figura 1), que atiende:
   i. **Display alfanumérico,**
   ii. **Teclado de 4 teclas**,
   iii. **puerto RS485 COM1 para recibir mediciones externas de viento y ambiente,**
   iv. puerto I2C para mediciones de potencia en placa externa (Módulo M4/E), RTC y E/S
   v. **puerto COM0 que puede hacer de terminal convencional con usuario** en un modo, y terminal Modbus en otro, de acuerdo al estado de la prueba.
   vi. administración de seteos, estado de prueba y datos de calibración de cada canal en EEPROM
   vii. Estadística de 6 canales y cálculo de densidad de aire.
   viii. Almacenamiento en SD de estadísticas cada 1 minuto.

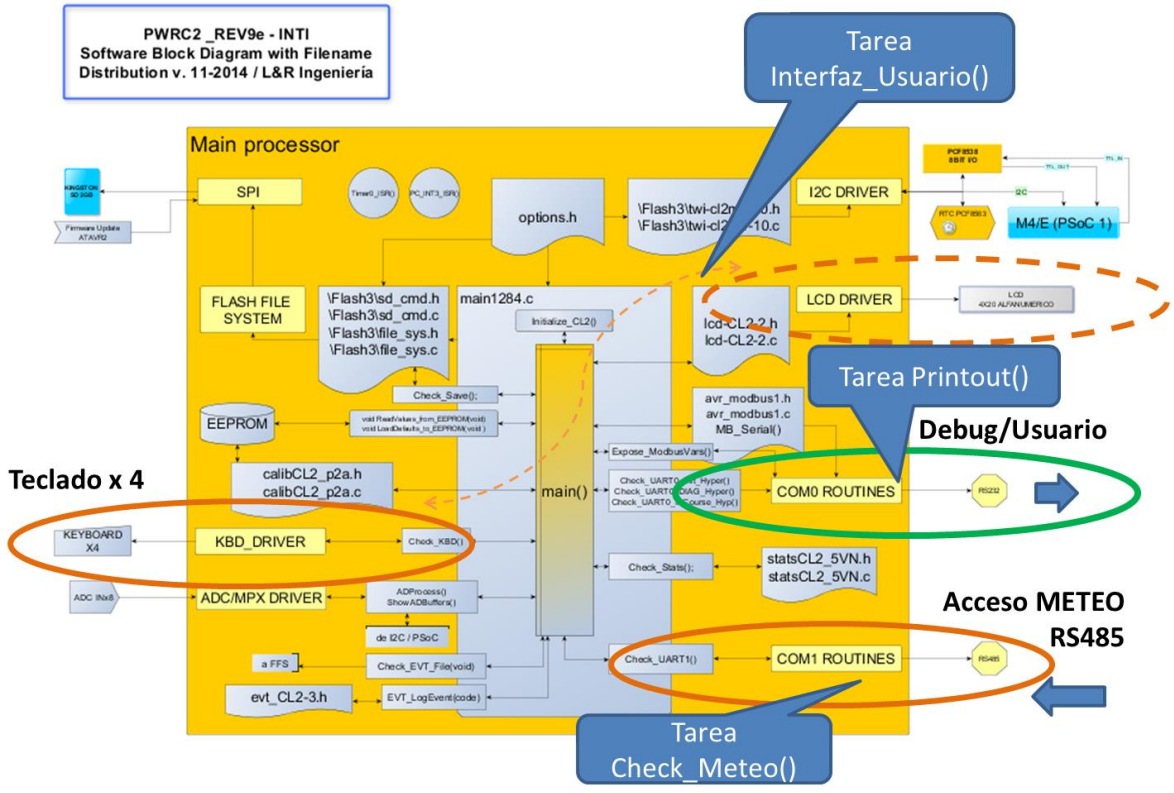

Figura 1 – Implementación Parcial de Sistema PWRC2-INTI con FreeRTOS

2. **Objetivo del Trabajo**: La implementación sobre la placa CL3 usando FreeRTOS pretende atender los requerimientos de los puntos (i),(ii) en una Tarea Interfaz_Usuario() [se ingresa punteado el LCD debido a que no es segura la implementación en el tiempo esperado], el punto (iii) de recepción de una cadena formateada de datos meteorológicos a través de un Puerto RS485 desde la estación exterior METEO, a través de una Tarea Check_Meteo(), y una implementación parcial del punto (v) de monitoreo/terminal, como salida únicamente a través de tarea Printout().

3. **Diagrama previsto Hardware:** El Sistema se implementará utilizando la UART1 para comunicación sobre RS485 con una unidad existente METEO (Figura 2a,b) y módulo de conversión, y UART2 ó UART6 para la comunicación de monitoreo con usuario (Figura 3a,b). Se prevé el uso del conector CN_LCD1 para conectar un display alfanumérico o SPI, que funcionará en conjunto con el teclado que habitualmente es un teclado de membrana de 4 teclas, implementado como GPIOs con Pullup externo (Figura 4). En la Figura 5 se ven las asignaciones de funciones en el CL3 utilizado.

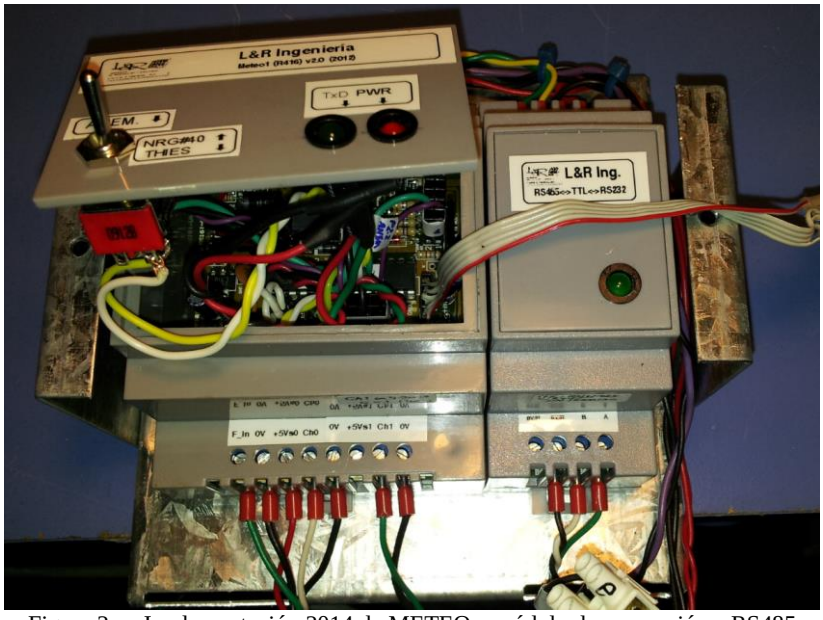

Figura 2a – Implementación 2014 de METEO y módulo de conversión a RS485

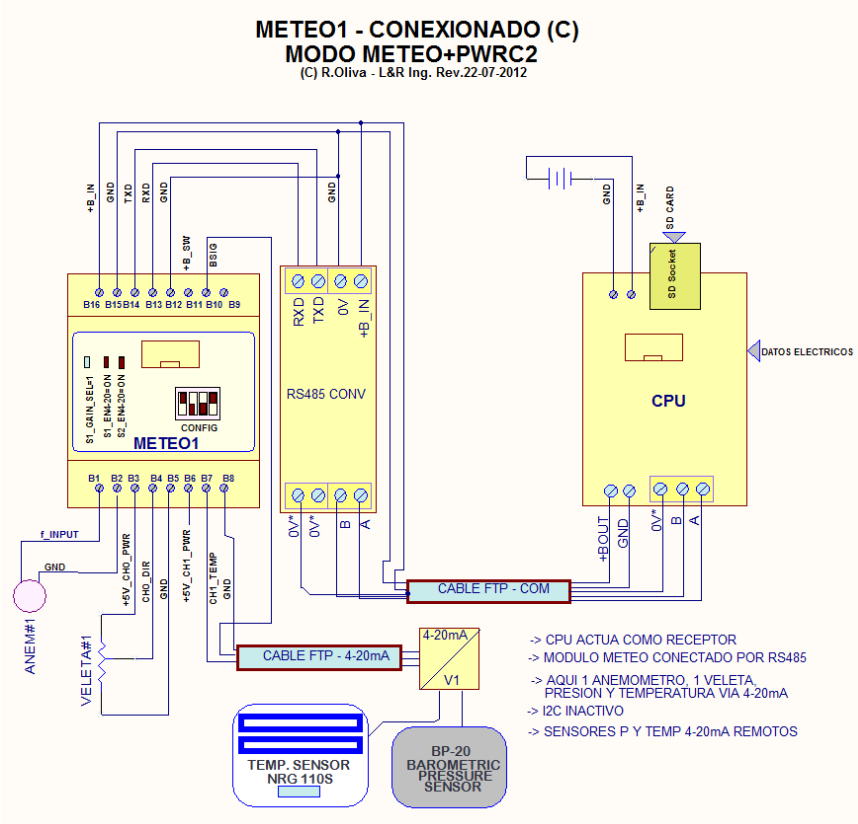

Figura 2b – METEO y módulo de conversión a RS485 – Conexionado a CPU

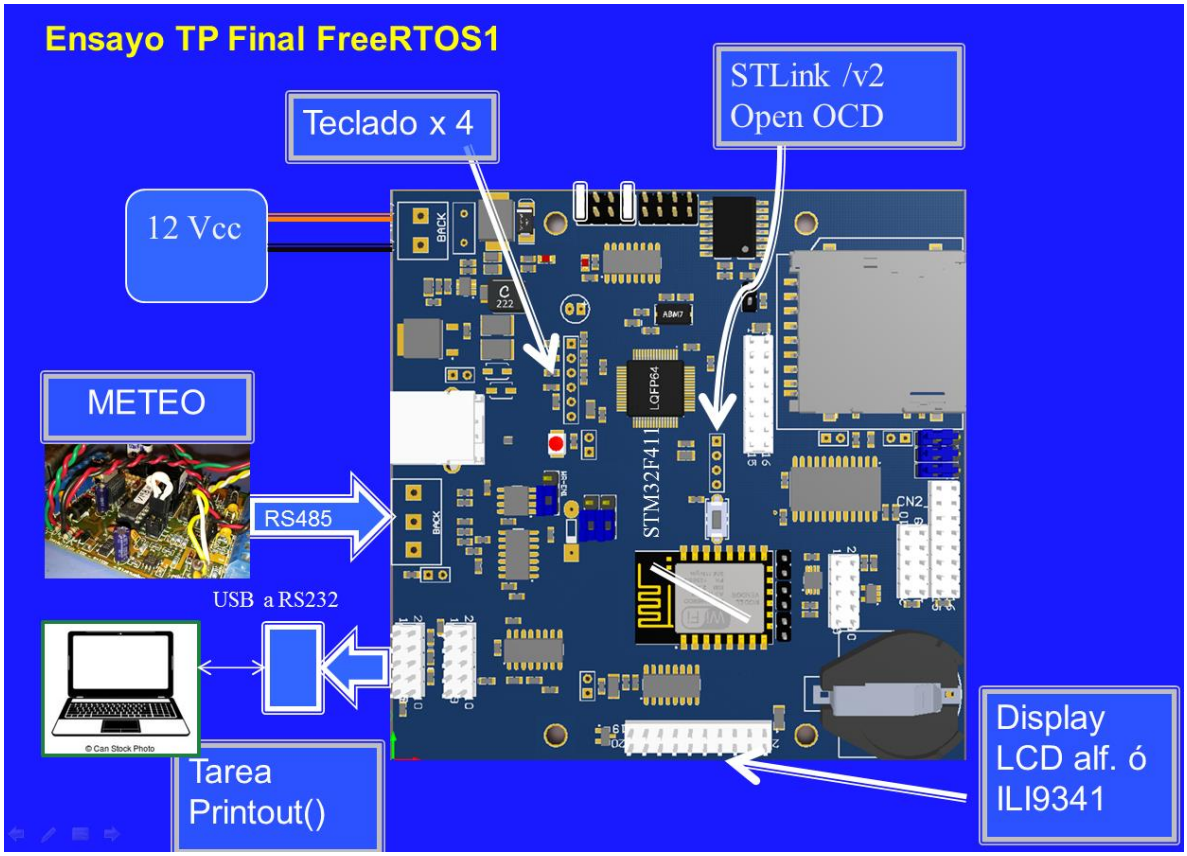

Figura 3a – Conexiones previstas Ensayo TPFinal RTOSi

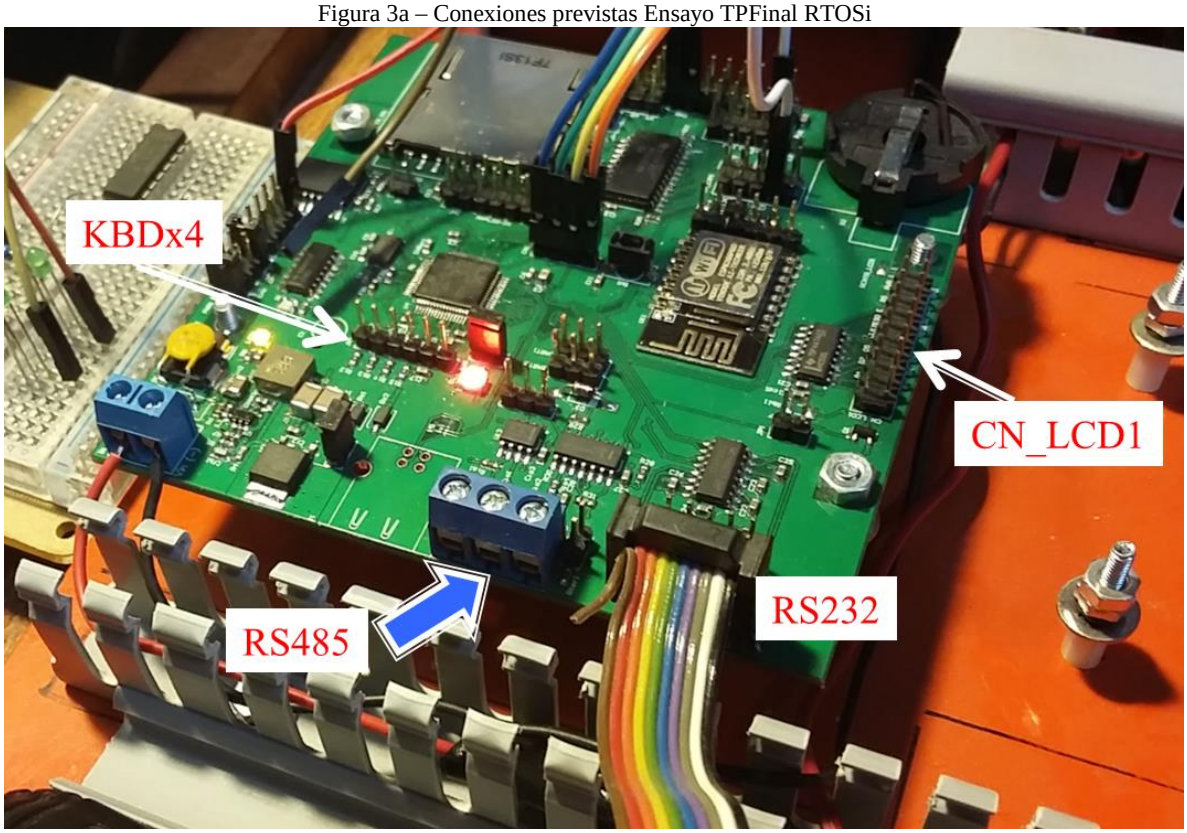

Figura 3b – Conexiones previstas Ensayo TPFinal RTOSi

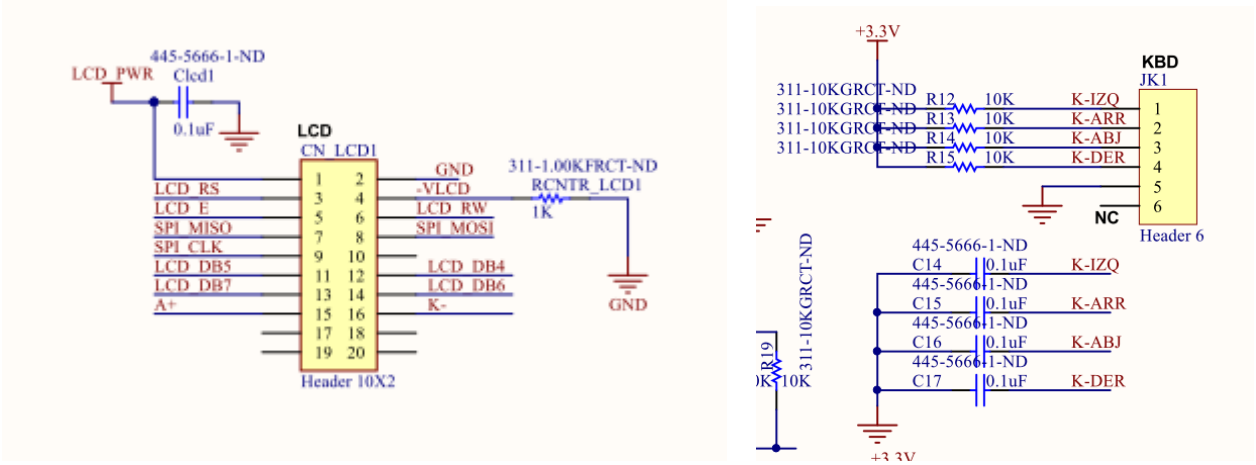

Figura 4 – (izq) Conector Dual para LCD alfanumérico convencional o LCD tipo ILI9341, utilizando uno de los pines GPIO como CS y (der) Conector de Teclado conectado a GPIOs del STM32F411

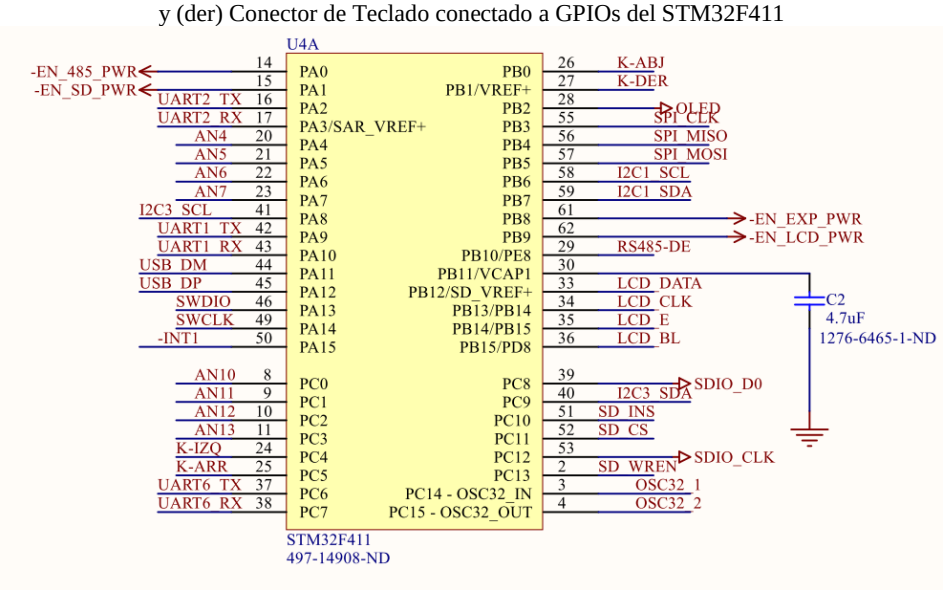

Figura 5 – Asignación de funciones en el STM32F411 de CL3

4. **Estructuras de Datos:** La parte más compleja de la propuesta es el reemplazo de la rutina Check_UART1() que se alimenta de una ISR (Figura 6) con buffer de 96 bytes, para trabajar a 38400 baud. La rutina Check_UART1() (Figura 7) de PWRC2 (que ejecuta varias veces en un lazo infinito, dentro del loop general del programa, Figura 8) por una tarea de FreeRTOS que se llamará Check_METEO(). La UART1 - ISR con Buffer deberá reemplazarse por un similar en STM32F4.

```
136 /*****************************************************************************
137 **
138 ** USART1 Receiver interrupt service routine
139 ** Buffer Size 256 not considered..30.1.18
140 *****************************************************************************/
141
142 interrupt [USART1_RXC] void usart1_rx_isr(void)
143 {
144 char status,data;
145 status=UCSR1A;
146 data=UDR1;
147 if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
148    {
149    rx_buffer1[rx_wr_index1++]=data;
150    // #if RX_BUFFER_SIZE1 == 256
151    // special case for receiver buffer size=256
152    // if (++rx_counter1 == 0)
153    //    {
154    // #else
155    if (rx_wr_index1 == RX_BUFFER_SIZE1) rx_wr_index1=0;
156    if (++rx_counter1 == RX_BUFFER_SIZE1)
157       {
158       rx_counter1=0;
159    //#endif
160       rx_buffer_overflow1=1;
161       }
162    }
163 }
```

Figura 5 – ISR Actual de UART1 en CL2, con ATMega1284P

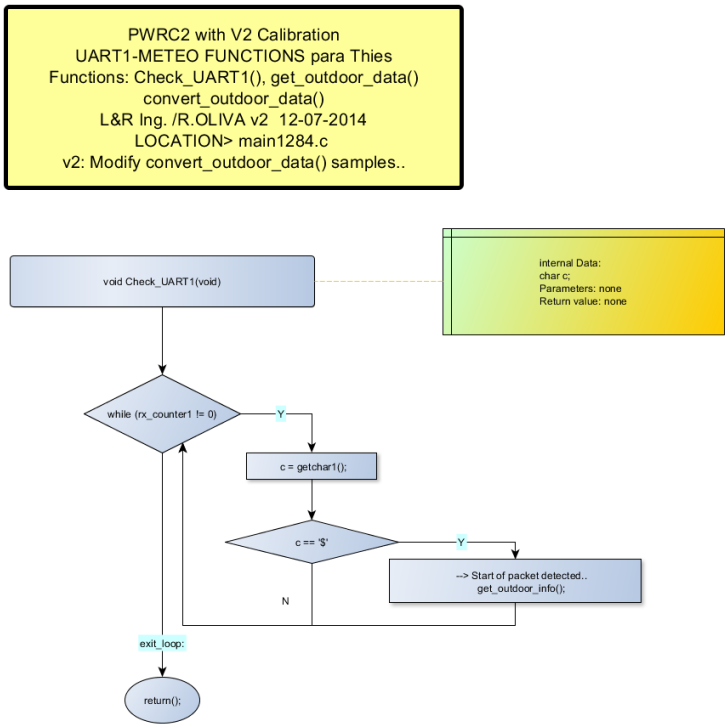

Figura 7 – Rutina Actual Check_UART1() a convertir en Task

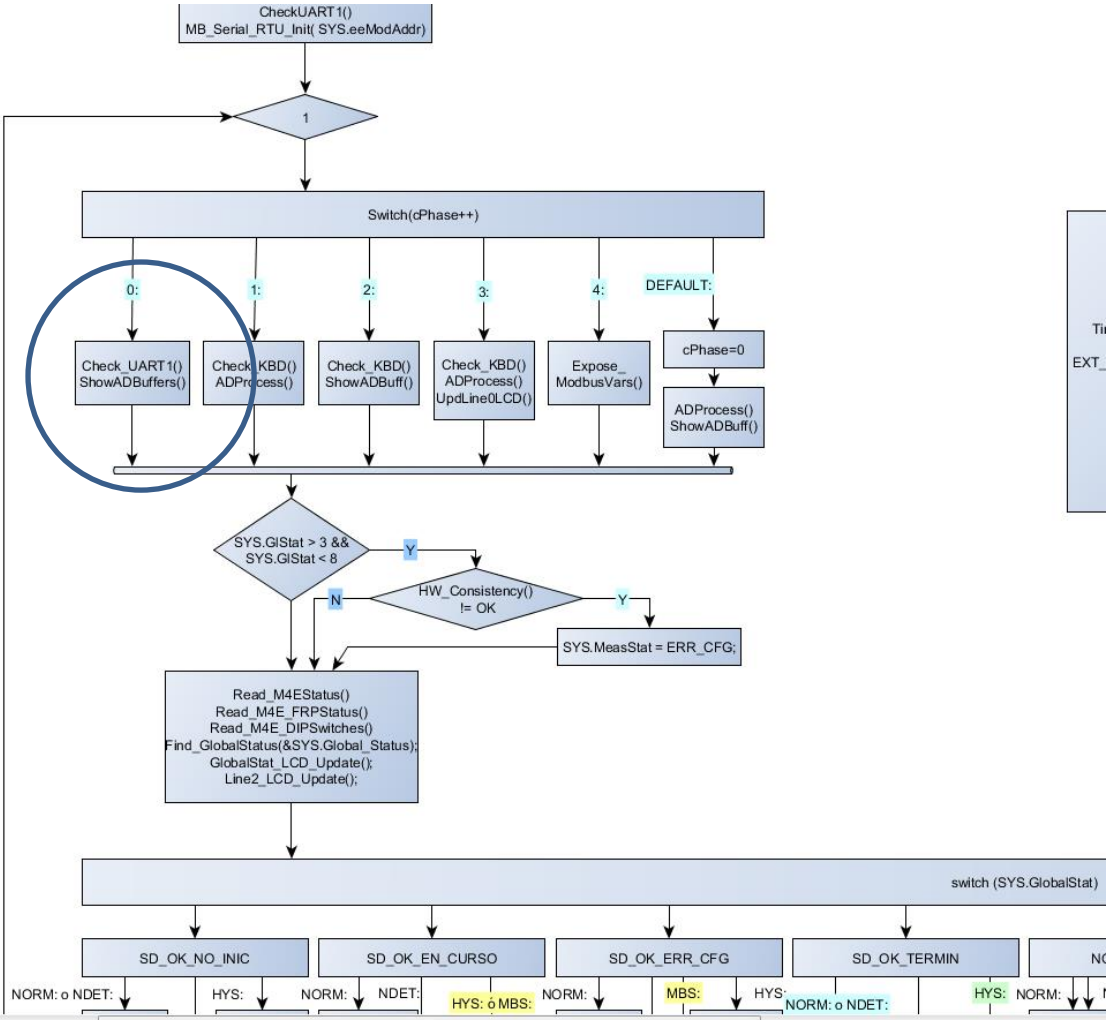

Figura 8 – Llamados a Check_UART1() en lazo infinito principal

```
METEO PACKET STRUCTURE FROM METEO v20 (R.Oliva)
 / Thies uses higher freq, send 'SSSSS'
instead of 'SSS' :
->  (Temperature is Txd in 100*K = 100*(T°C+273.15) =
                    100*(273.15+20.1)=29335):
char    TestCStr[] = "UUU$29335.10156.2562.15100.125.1095*QQQ";
So the sscanf() function is %5d for the first.. (total 39c)

For v20 Thies- 2014 :
Maintains same packet "UUU$ttttt.bbbbb.dddd.sssss.vvv.CRCC*QQQ" but:
    ttttt is 00000 08191 Raw Temperature ADC reading,
                    can be 0-5V( Direct sensor with G=2) or 1-5V (4-20mA)
    bbbbb is 00000 08191 Raw BaroPressure ADC reading,
                    can be 0-5V( Direct sensor with G=1) or 1-5V (4-20mA)
    dddd  is 0000 to 3600, WDIR*10 in UWORD
    sssss   is 00000 to 99999, 0 to 9999.9Hz (no scaling) from Thies Anemometer.
   (15100 reading would be 1510Hz, or Typical: V= 0.04597*1510+0.21=69.6m/s)
    vvv   was voltage, not used.
    CRCC  is simple checksum
```

Figura 9 – Estructura del Paquete recibido por Check_UART1()

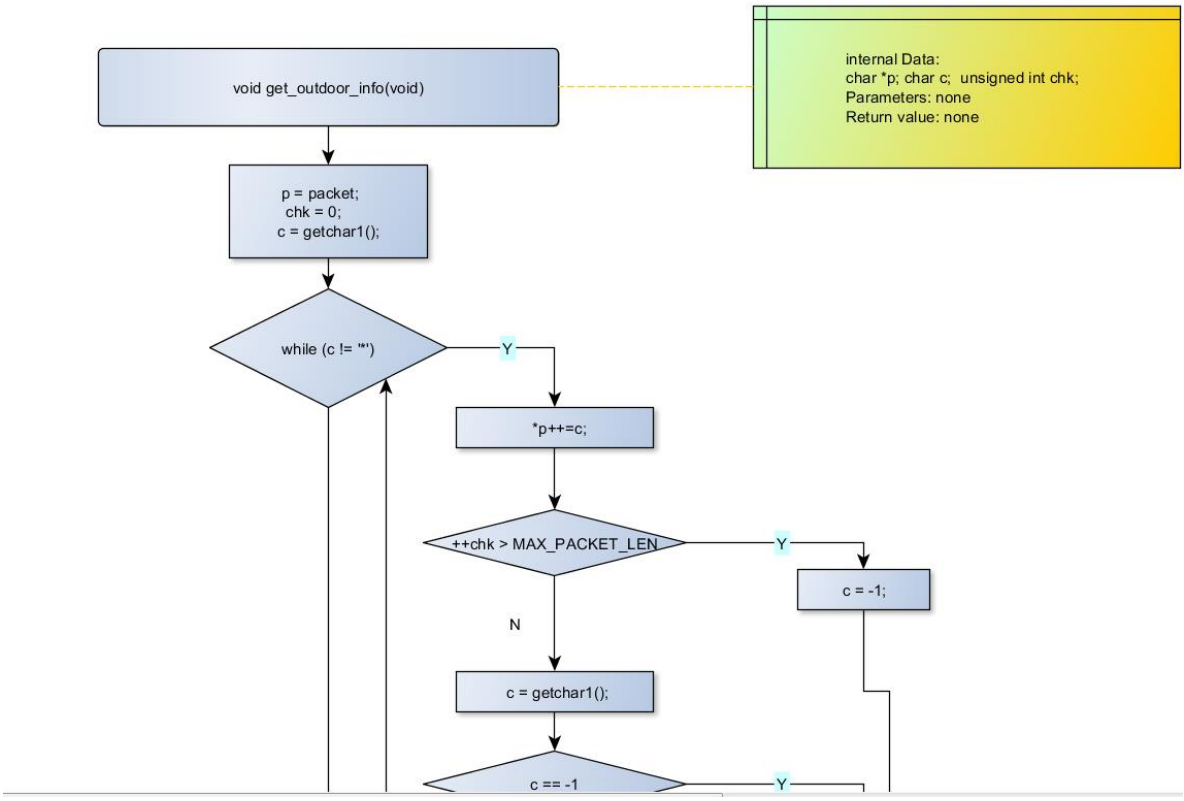

Figura 10 – Rutinas a implementar en tarea Check_METEO() – Inicio de get_outdoor_info() que decodifica el paquete

La estructura del paquete recibido se muestra en Figura 9, y en la Figura 10 se muestra la parte inicial de la rutina de decodificación get_outdoor_info() y luego la final en Figura 11, que llama a la rutina de decodificación convert_outdoor_data() (Figura 12) – Las estructuras globales FS. / FPS. deberían convertirse en una cola de FreeRTOS.

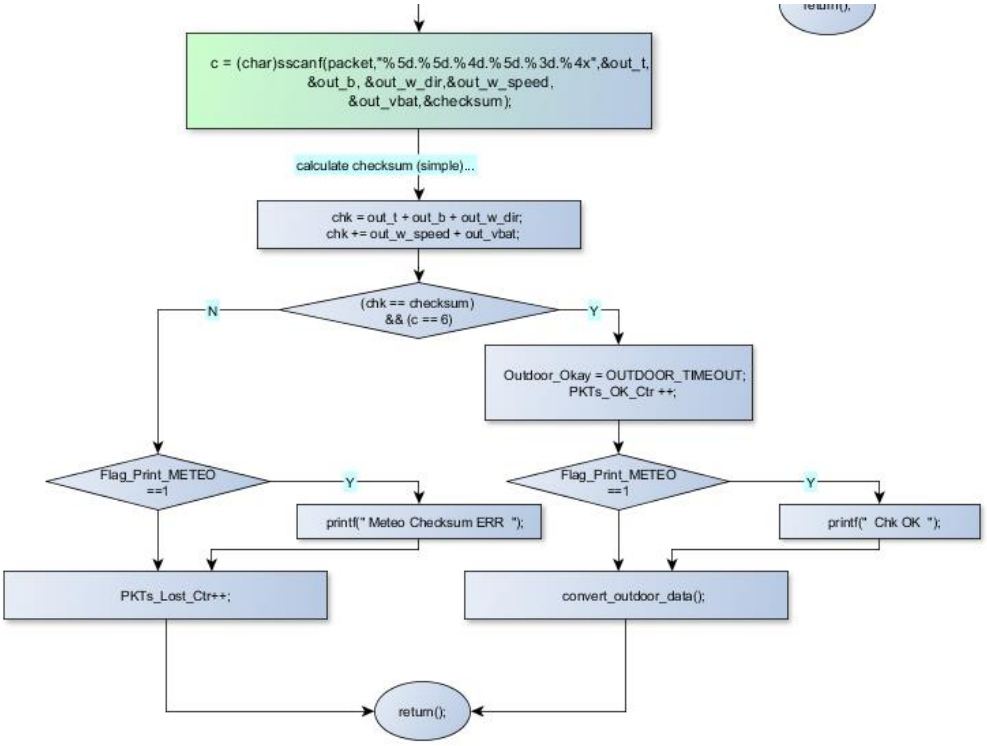

Figura 11 – Rutinas a implementar en tarea Check_METEO() – Final get_outdoor_info() que decodifica el paquete, y llama a convert_outdoor_data()

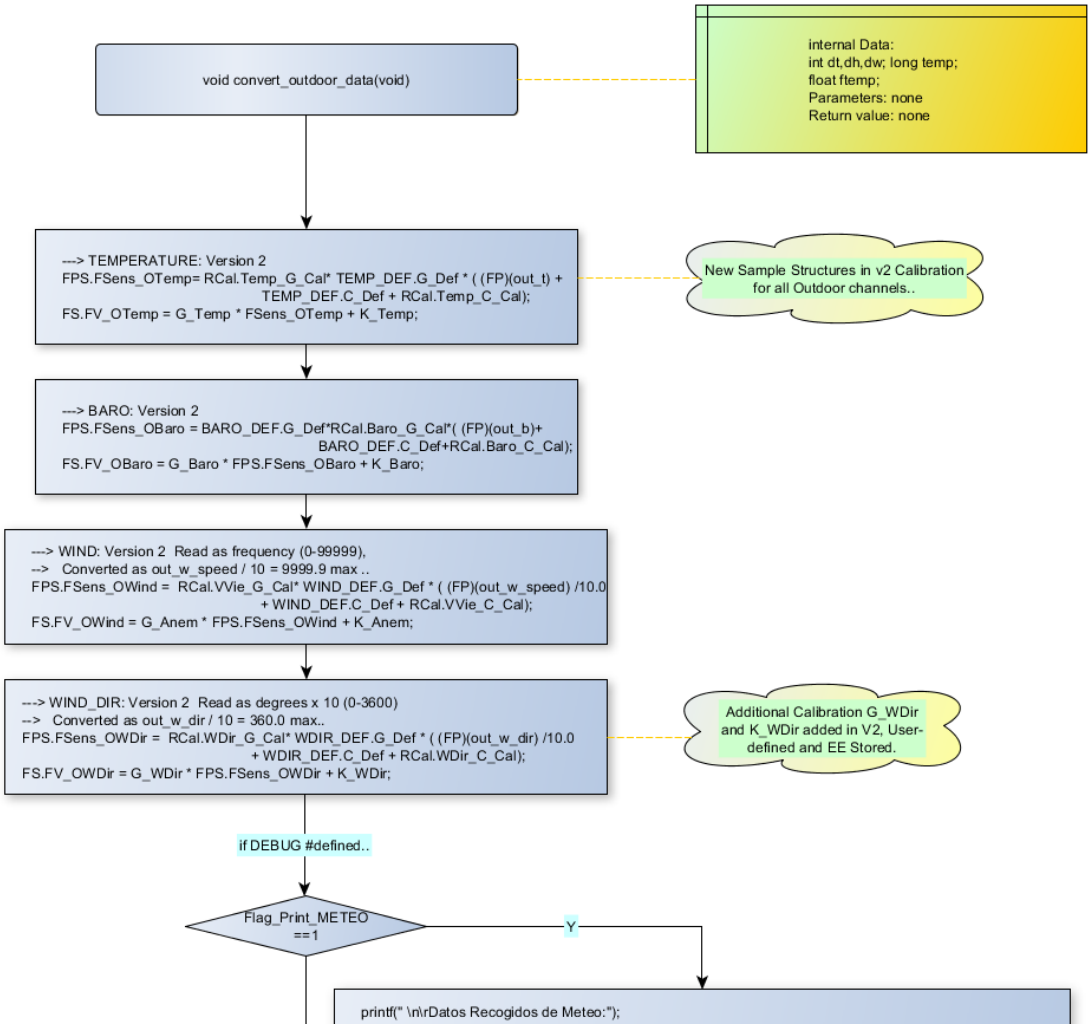

Figura 12 – Rutinas a implementar en tarea Check_METEO() – Parcial de convert_outdoor_data() – Las estructuras globales FS. / FPS. deberían convertirse en una cola de FreeRTOS

5. **Tareas previstas:** La distribución prevista del programa se muestra en la Figura 13.

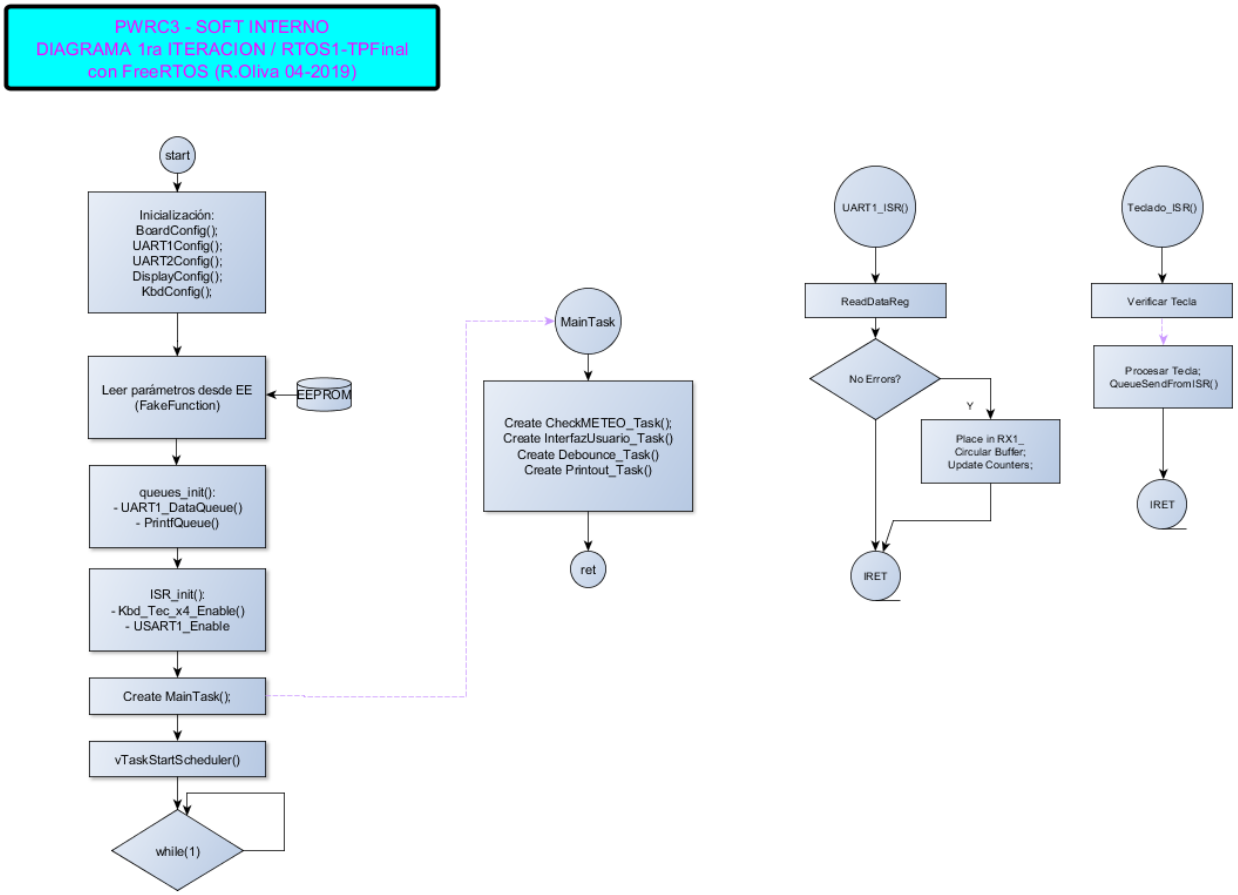

Figura 13 – Estructura general prevista del programa

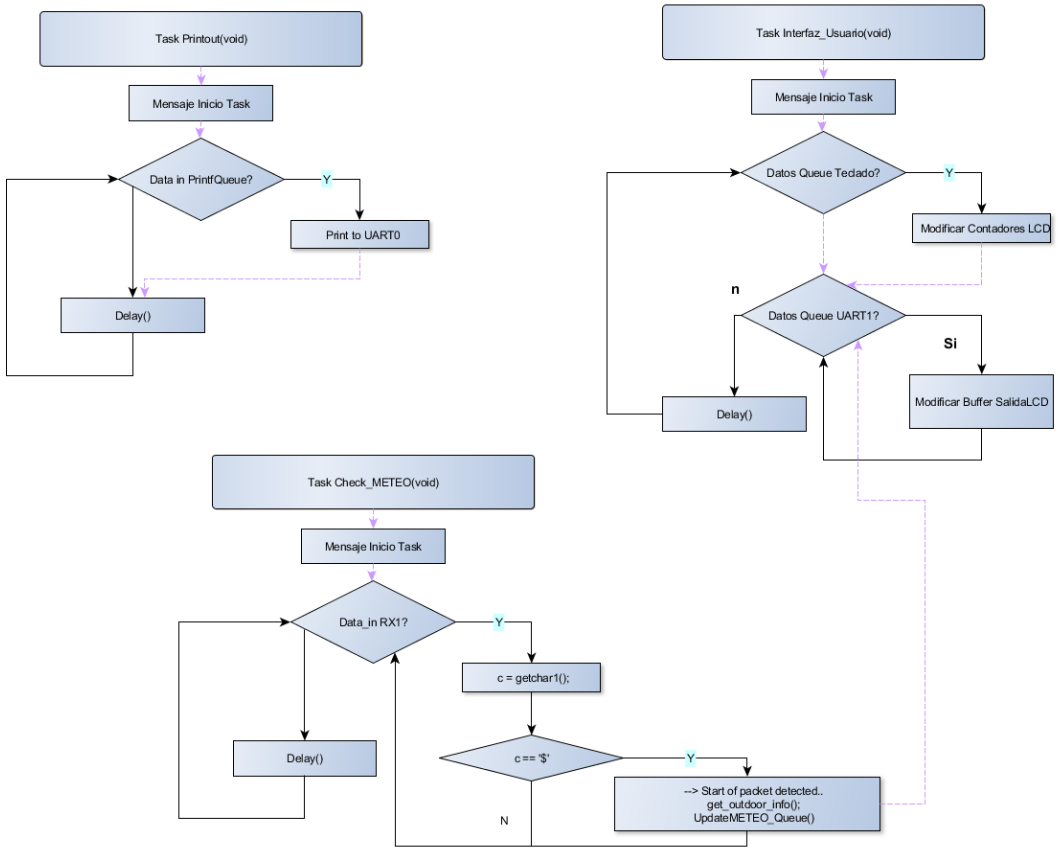

Figura 14 – Posible distribución de Tareas

6. **Alcances de funcionamiento:** Se prevé lograr una decodificación del paquete remitido por el módulo METEO vía RS485, y su envío como datos formateados a través de una cola a la tarea Interfaz_Usuario(). La misma las envía al LCD y (no mostrado) a la UART principal. El presionado de las Teclas (conectadas a un ISR y tarea Debounce no mostrada) modifica el Buffer que es mostrado en el LCD.

-0-