HITOS 1 Y 2 TRABAJO FINAL – RTOS 1 – CESE
R. OLIVA – v 11-04-2019

CONTENIDOS:

1. **Introducción**: El proyecto prevé completar una parte de la integración de FreeRTOS al Proyecto Final iniciado en 2018 *"Registrador industrial con soporte de placas periféricas"*, y según se indicó en el documento previo de Alcances, 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.

   Los Hitos indicados en la planilla de Trabajo Final RTOS1 son los siguientes:

   Hito 1: Definir las tareas que se van a ejecutar. ¿son periódicas? ¿son one shot? ¿quién las dispara? ¿otra tarea, hardware? ¿Necesitan información de terceras tareas? Establecer prioridades

   Hito 2: Definir semáforos a utilizar (binarios o contadores) e implementar la sincronización

   Hito 3: Definir colas de mensaje, espacios de memoria compartida y otros elementos de comunicación

   Hito 4: Implementar el uso de recursos de HW

2. **Hito 1, avances:** Se definió una posible distribución de tareas en el documento de alcances, a través de la siguiente diagrama (Figura 1)

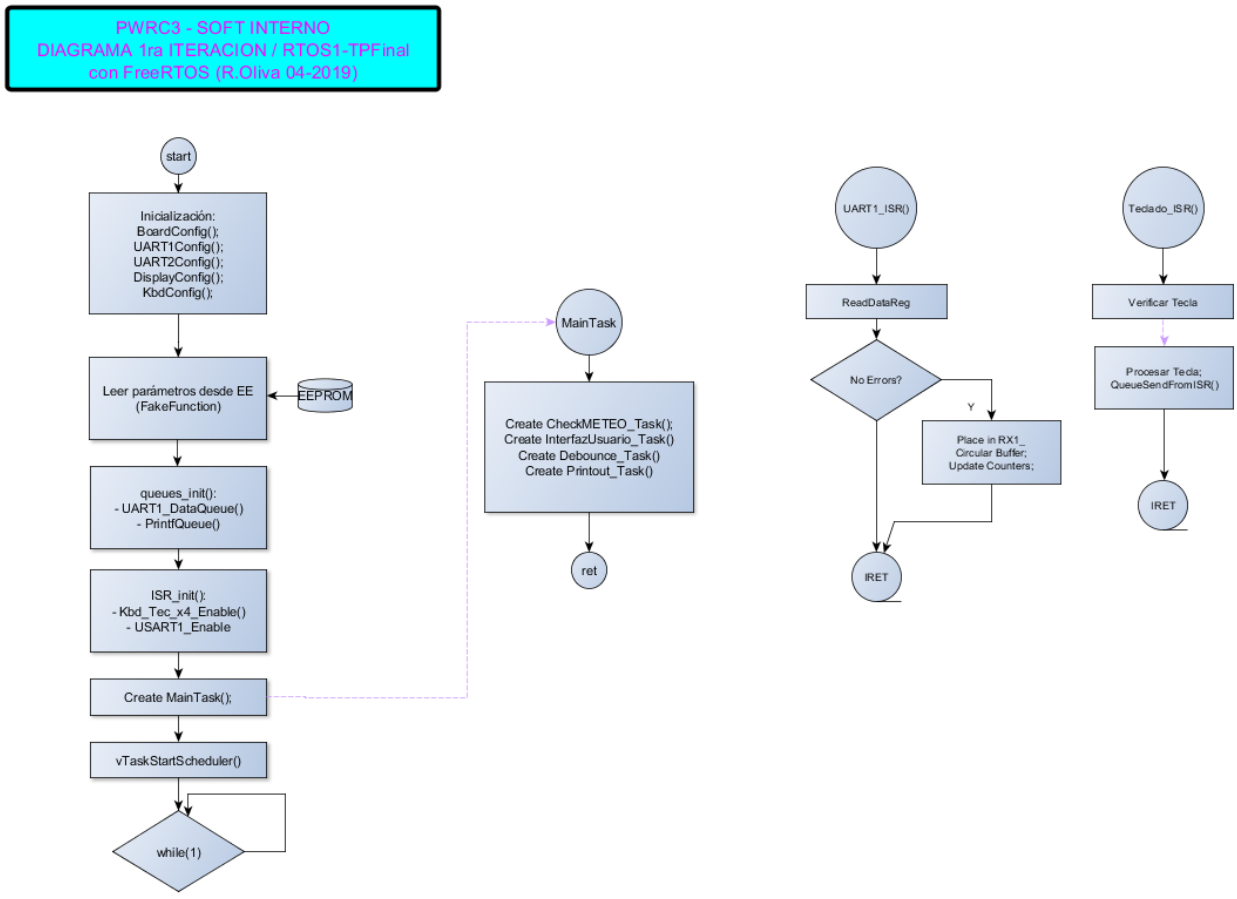


Figura 1 – Estructura general prevista del programa

En la implementación usual de estos equipos, el paquete de datos meteorológicos que se indicó (Figura 2) llega en forma autónoma aproximadamente a 1 Hz de frecuencia (1 paquete /segundo), y los equipos trabajan a 38400 baud, 8 bits, 1 bit de parada y sin paridad, con un simple checksum algebraico al final del paquete.

```
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 2 – Estructura del Paquete a ser recibido a una frecuecia de 1Hz.

CheckMETEO_Task() Puede considerarse una tarea periódica a 1Hz, pero no sincrónica ya que el tiempo de cómputo de la unidad METEO es variable, y a efectos del sistema está disparada por un "burst" de caracteres en UART1 a través de RS485. La ISR para UART1 todavía está por determinarse si será por polling o por DMA, hay varias alternativas posibles. Por de pronto, por ser Hardware no ensayado se requirieron pruebas de la parte UART1 ->RS485 (Figura 3), utilizando las sencillas funciones del la HAL de STM32.

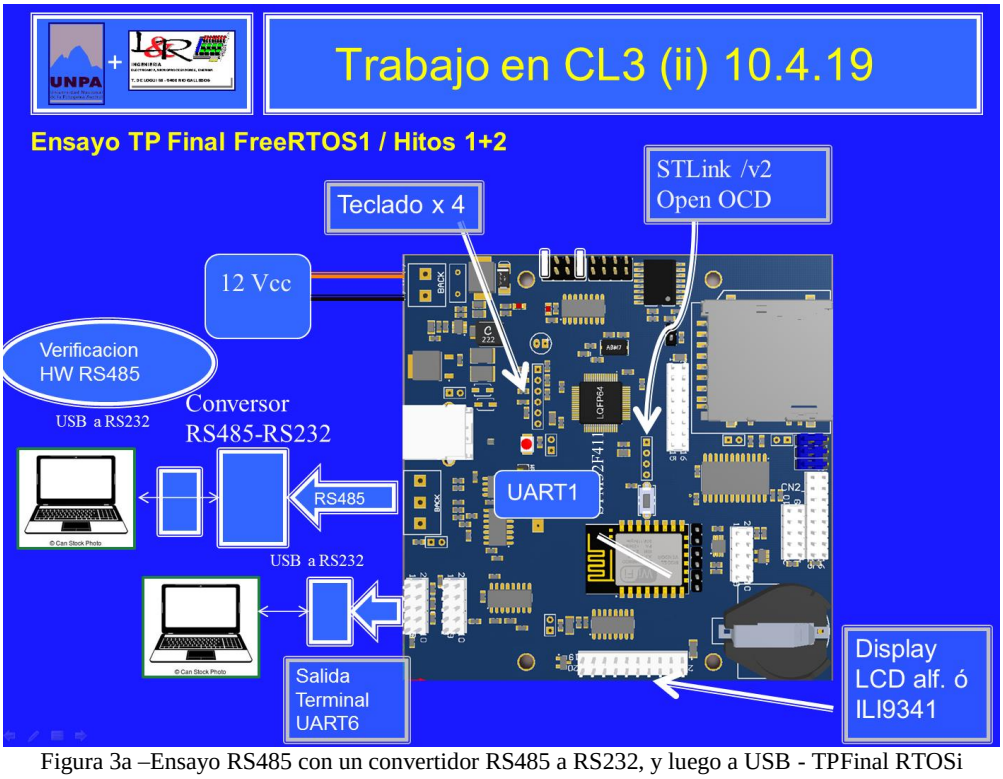


Figura 3a –Ensayo RS485 con un convertidor RS485 a RS232, y luego a USB - TPFinal RTOSi

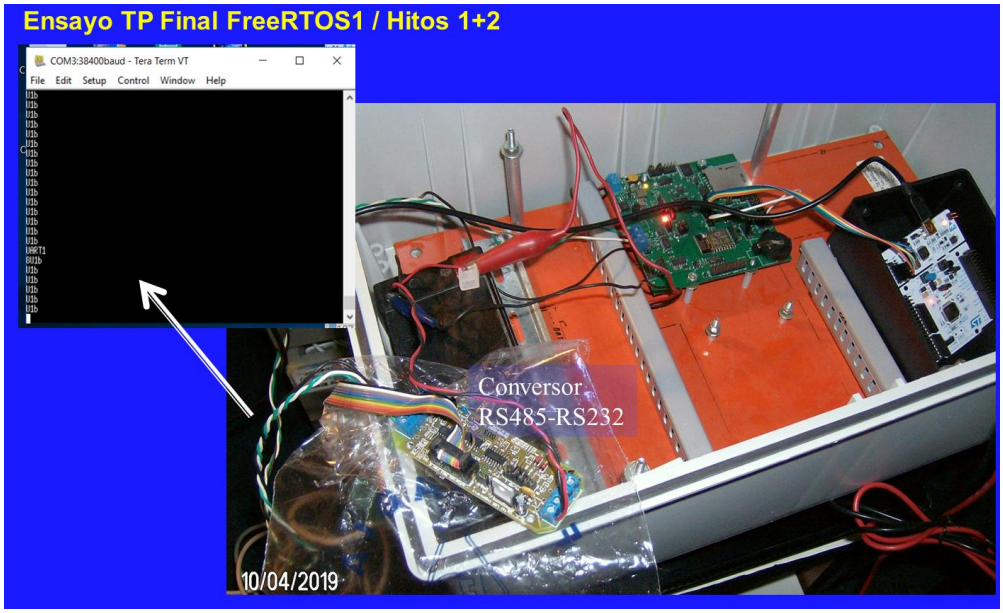


Figura 3b –Foto del Ensayo RS485 con un convertidor RS485 a RS232, y luego a USB

Lo que se ve, es la salida de la tarea muy sencilla de prueba:

```
07
08  void vTask1_handler(void *params)
09  {
10      uint8_t data1[40];
11      while(1){
12          sprintf((char *)data1,"U1b\n\r");
13          HAL_UART_Transmit(&huart1, data1, 10, 10);
14          vTaskDelay(500/portTICK_RATE_MS);
15      }
16  }
```

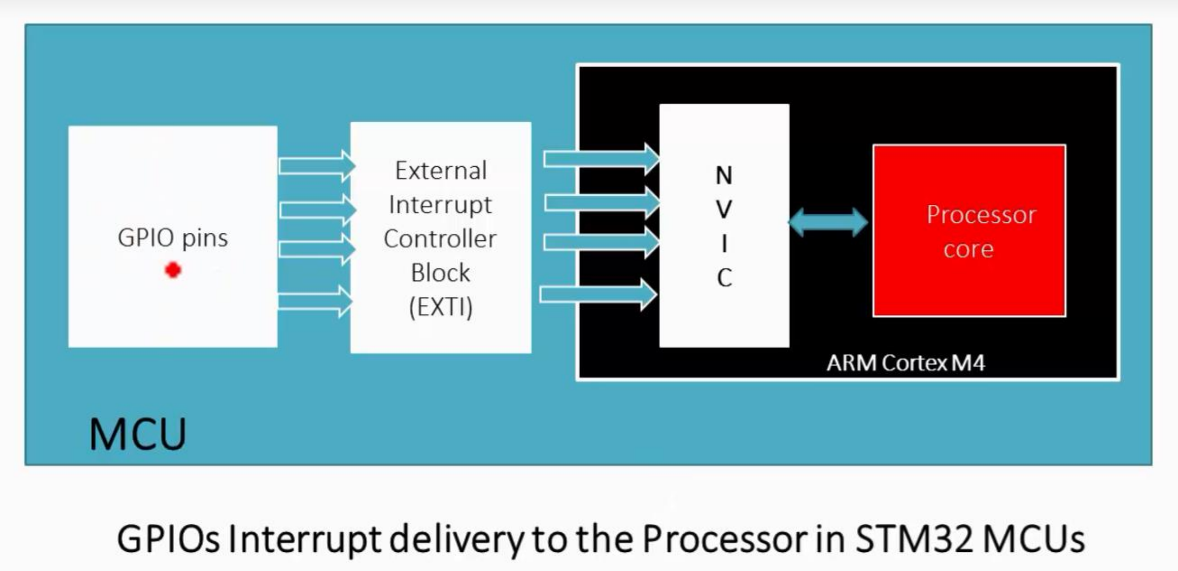
Dentro de un programa sencillo de ensayo configurado a través de la HAL. Se observa que utiliza funciones Wrapper CMSISv2 para el FreeRTOS v10.1.0, aunque acepta la notación nativa de FreeRTOS.

```c
int main(void)
{

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* Configure the system clock */
  SystemClock_Config();

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_I2C1_Init();
  MX_USART6_UART_Init();
  MX_USART1_UART_Init();

  uint8_t data[40];
  sprintf((char *)data,"CESE2\n\r");
  HAL_UART_Transmit(&huart6, data, 10, 10);
  HAL_Delay(100);
  // Added 10.4.2019 Transmit @38400 thru UART1
  sprintf((char *)data,"UART1\n\r");
  HAL_UART_Transmit(&huart1, data, 10, 10);
   HAL_Delay(100);
  /* Call init function for freertos objects (in freertos.c) */
  MX_FREERTOS_Init();
  // 3. Create 2 tasks..
  xTaskCreate(vTask1_handler,
                "Task-1",  // char
                configMINIMAL_STACK_SIZE,
                NULL,
                2,   // Max is 5 in FRTCFG.h..
                &xTaskHandle1);

  /* Start scheduler */
  osKernelStart();
```

Figura 4 –Ensayo sencillo con la HAL, que utiliza funciones Wrapper CMSISv2 para el FreeRTOS v10.1.0

Por otro lado, la tarea InterfazUsuario_Task() complementa la lectura de 4 switches convencionales con debounce, y la interacción con un Display y Terminal.

- ISR_Teclado: La implementación de los switches es con Pullup, por lo cual en STM32 hay que configurar los EXTI para que ingresen a la NVIC, algo que en la CIAA se realiza con la sAPI o indirectamente con LPC_Open.



GPIOs Interrupt delivery to the Processor in STM32 MCUs

Por ejemplo para inicializar un botón en el GPIO PC13, se configura como sigue:

```
175     //interrupt configuration for the button (PC13)
176     //1. system configuration for exti line (SYSCFG settings)
177     SYSCFG_EXTILineConfig(GPIOC,EXTI_PinSource13);
178
179     //2. EXTI line configuration 13,falling edge, interrup mode
180     EXTI_InitTypeDef exti_init;
181     exti_init.EXTI_Line = EXTI_Line13;
182     exti_init.EXTI_Mode = EXTI_Mode_Interrupt;
183     exti_init.EXTI_Trigger = EXTI_Trigger_Falling;
184     exti_init.EXTI_LineCmd = ENABLE;
185     EXTI_Init(&exti_init);
```

Y el NVIC de la siguiente manera, suponiendo una asignación de prioridad 5:

```
//3. NVIC settings (IRQ settings for the selected EXTI line(13)
NVIC_SetPriority(EXTI15_10_IRQn,5);
NVIC_EnableIRQ(EXTI15_10_IRQn);
```

Hito 2, avances: Se prevé la siguiente distribución. El uso de semáforos / Mutex estará para la protección de las estructuras de datos.

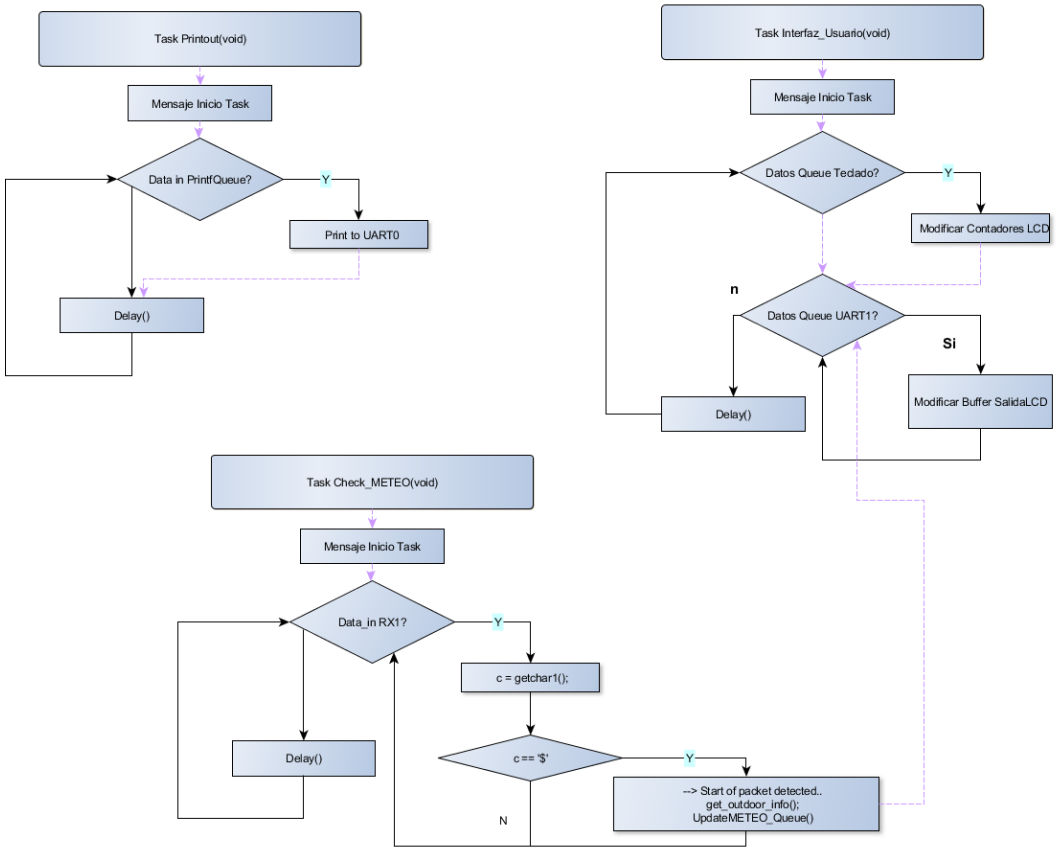


Figura 5– Posible distribución de Tareas

ANEXO

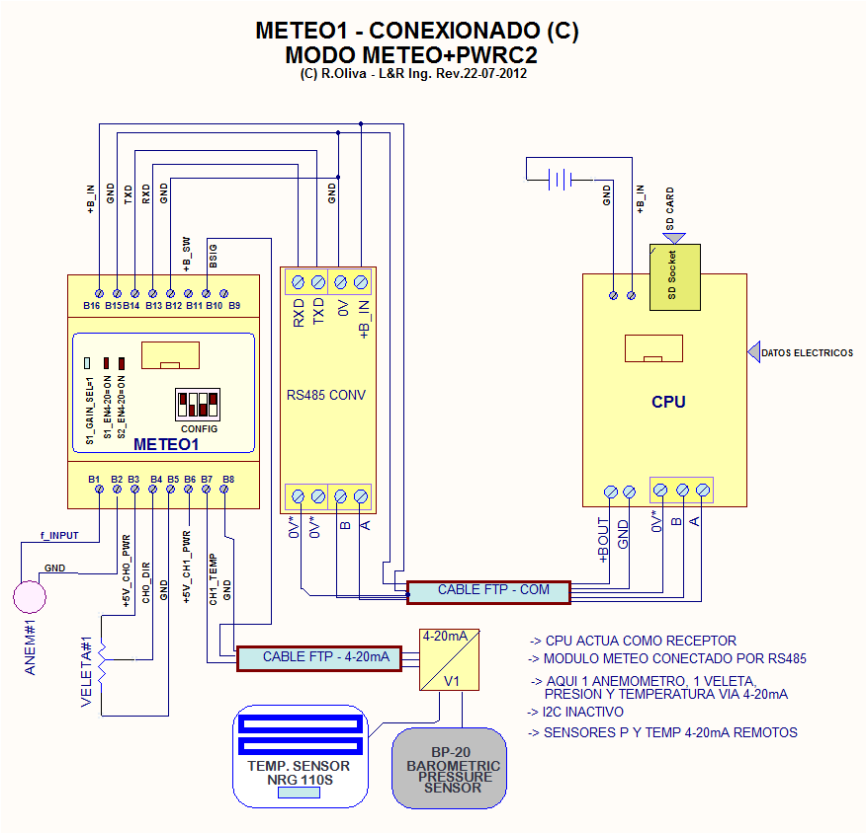


Figura A1 – METEO y módulo de conversión a RS485 – Conexionado a CPU

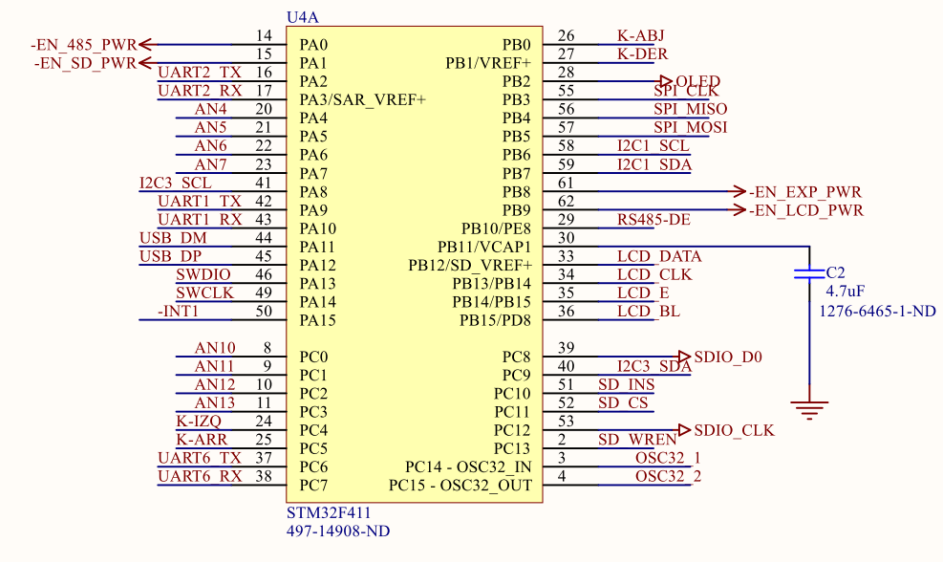


Figura A2 – Asignación de funciones en el STM32F411 de CL3