TRABAJO FINAL – RTOS 1 – CESE
R. OLIVA – Rev3 v 27-04-2019 12:07

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 a través de una UART1, y su envío como datos formateados a través de una cola a la tarea Interfaz_Usuario(). Se reemplazó el envío de los datos al LCD por una interfase de terminal de usuario a través de una UART6 del controlador.

2. **Avances implementación:** Se definió la siguiente distribución (Figura 1)

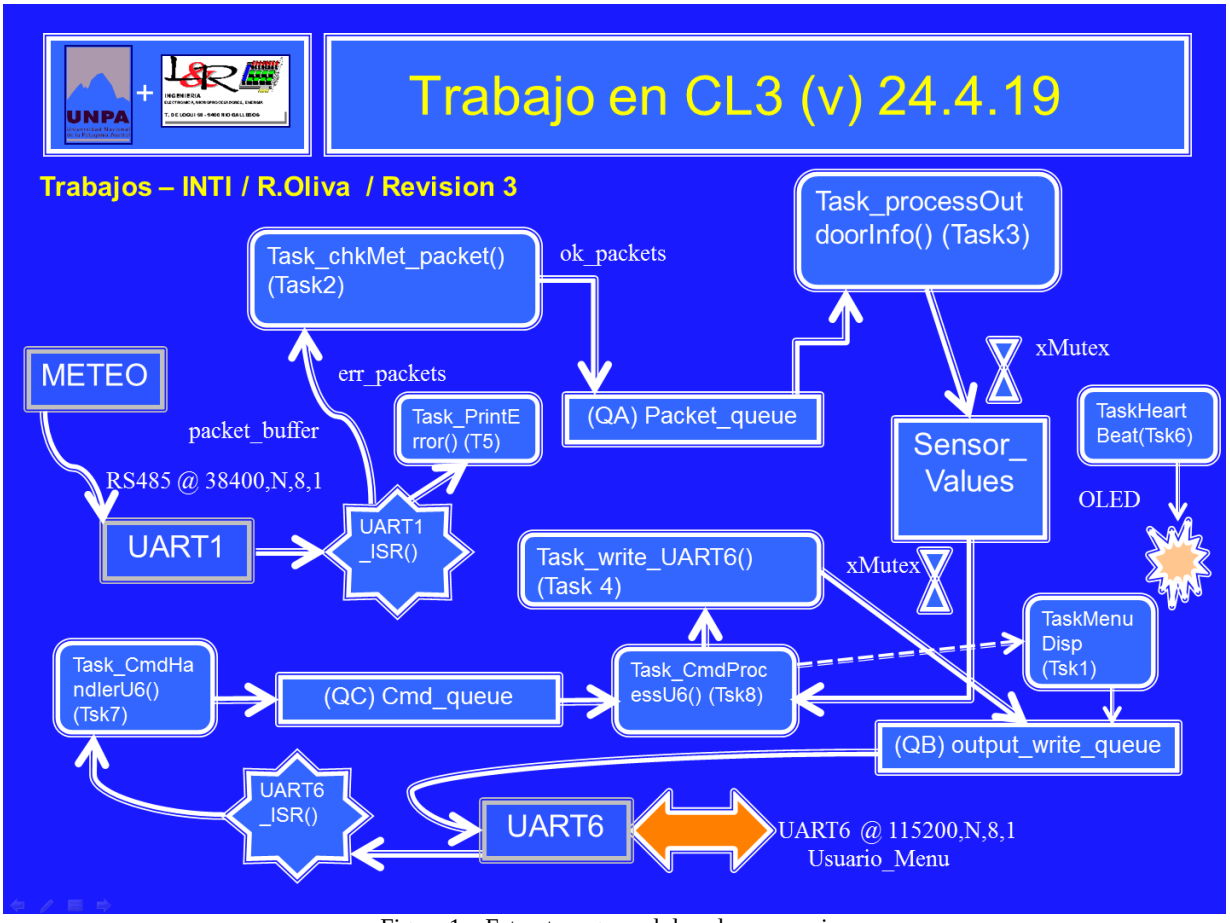

Figura 1 – Estructura general de colas y espacios

2.1 Componente UART6 Terminal de usuario

En la implementación actual se logró llevar adelante el menú de usuario ampliado como se muestra en la Figura 2. En este menú se muestra la salida con la Tarea 2 generando leyendo datos de la unidad METEO a través de la UART1

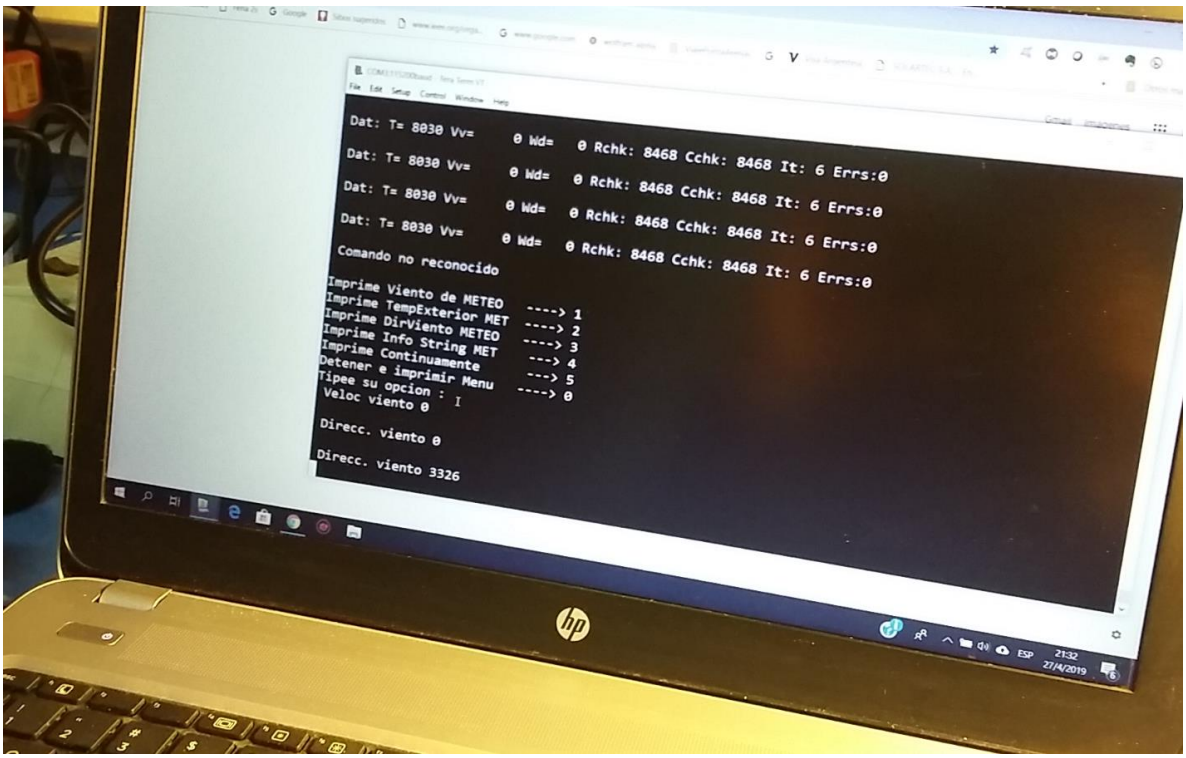

Figura 2 – Menu de usuario presentado - Trabajando con datos reales de muestra via UART1

2.2 Modificaciones realizadas respecto a la versión 25.04.2019

RESUMEN (únicamente en main.c, subida a / UART1_5_LLFRc en el repositorio Github)

1) Modificamos Task2 para recepción del Buffer leído desde la ISR UART1, en lugar del string de muestra.
2) Nueva ISR UART1, con reconocimiento de inicio '$' y fin '*' a través de una mini-FSM
3) Aumentamos de 2 a 3 la prioridad de la tarea Task2 para la Lectura del String generado por la ISR
4) La tarea de impresión Task5 de errores simplemente actualiza un contador de errores de recepción, que es leído en la opción 5 despues de Nº de ítems
5) Se realizó un nuevo video con una veleta NRG 200P conectada a la unidad METEO, se accionó para ir variando el valor de Wd como Angulo 0-360 x 10 (0 a 3600).

2.3 Modificaciones en el Task 2

La nueva rutina del Task 2, para recepción del buffer vía la ISR de USART1 se muestra en el listado siguiente (`DEMO_STRING_COPIED no definido`):

```
// Task2 vTask_Check_Meteo_packet - Verifica lo que llega
// de METEO por UART1, y lo pasa a la cola de paquetes ok
// En modo Demo, envía cada 1000 ms un paquete de muestra
// espera notificación de un paquete via UART1
// 27.4.2019 Sacamos modo Demo, solo String ISR_UART1
// Copiado y pasado al Packet queue, delay 0
//      UUU$   start identifier
//      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 from Anemometer / Thies.
//      vvv   was voltage, not used.
//      CRCC  is simple checksum (paso a uint16_t)
//      *QQQ  end identifier
//      Total length from $ is: 5+1+5+1+4+1+5+1+3+1+4= then '*' =15+7+4+5 =31
//      Total length form first U is 31+4 = 35
//      TestCStr[] = "UUU$29335.10156.2562.15100.125.dfbe*QQQ";
// 27.4.2019 nuevo ISR detecta $ y copia hasta *, reemplaza este por un /0 termination..
```

```
void vTask_Check_Meteo_packet(void *params)
{
//      typedef struct PACKET_OK
//      {
//        char packet_content[PACK_OK_LEN+2];
//      } PACKET_OK_t;
// PACKET_OK_t new_packet; // (as global)
//
#ifdef DEBUG_VERBOSE
   printmsg("\n\rTsk2");
#endif
//vTaskDelay(20);
        while(1)
        {
        // 27.4.2019 - Sacamos modo Demo, esperamos String de ISR_UART1
                // se utiliza xTskNotfromISR() allí.
                xTaskNotifyWait(0,0,NULL,portMAX_DELAY);

                // Cada 0.5 seg - enviaba string de prueba
                // vTaskDelay(pdMS_TO_TICKS(500));

                // 1. allocate space.. Not used
                // new_packet = (PACKET_OK_t*) pvPortMalloc(sizeof(PACKET_OK_t));

                taskENTER_CRITICAL();

                // packet_buffer es generado por la ISR hasta que encuentra el caracter '*' de terminación
                // nuevo packet 27.4.19 desde '$? hasta '*' con null termination
                strncpy(new_packet.packet_content, (char *)(packet_buffer),PACK_OK_LEN);

        #ifdef DEMO_STRING_COPIED
                // Copiamos uno de muestra (corregido 25.42019)
                // char okTestCStr[] = "UUU$29335.10156.2562.15100.125.dfbe";
                strncpy(new_packet.packet_content, okTestCStr,PACK_OK_LEN);
        #endif

                taskEXIT_CRITICAL();

                xQueueSend(packet_queue,&new_packet, portMAX_DELAY);
                // 24.4.2019 Delay
                vTaskDelay(pdMS_TO_TICKS(50));
        }

}
```

2.4 Modificaciones en la ISR de UART1

A fin de evitar errores en la recepción, que se daban con la configuración de la ISR anterior, se la modificó para agregar una mini-FSM de detección del carácter de inicio '$' y del de fin '*', con lo cual la tasa de errores se redujo a prácticamente 0 en condiciones de ensayo. Se eliminó la lectura del primer conjunto UUU$ y su envío al string de decodificación por ser redundante.

```
// USART1_ISR R.Oliva 18.4.2019 moved to main 24.4.2019
// 27.4.2019 - Added FSM for start '$' and end '*' of packet detection
// adds global variable packet_started = 0;
void USART1_IRQHandler(void)
{
        uint8_t data_byte;  // LL_ requiere 8 bit
        //a data byte is received from the user
        BaseType_t xHigherPriorityTaskWoken = pdFALSE;
        // char dbg_msg[20];

        if( LL_USART_IsActiveFlag_RXNE(USART1) && LL_USART_IsEnabledIT_RXNE(USART1))
        {
                data_byte = LL_USART_ReceiveData8(USART1);
        // (A0) 27.4.2019 - Add startofpacket detection
                if((data_byte == '$' ) && (packet_started == 0)){
                        packet_started = 1;
                    // Point (A)
                } else if (packet_started == 1){
                        // Point (B1)
                        packet_buffer[packet_len++] = data_byte;
```

```
                        if((data_byte == '*' ) && (packet_len < PACKET_MAX)){
                                {
                                //then packet is ok.. (E1)
                                packet_buffer[packet_len]= '\0';   // Null Terminate the string
                                packet_started = 0;
                                //notify the CheckMeteoHand task (to copy the buffer)
                                xTaskNotifyFromISR(xTaskHandleChkMeteo,0,eNoAction,&xHigherPriorityTaskWoken);
                                //reset the packet_len variable
                                packet_len = 0;
                                } // F1
                        if (packet_len > PACKET_MAX){
                                packet_len = 0;
                                xTaskNotifyFromISR(xTaskHandlePrintErr,0,eNoAction,&xHigherPriorityTaskWoken);
                            }
                        }   // End
                } // Point G1
            }
            // if the above freertos apis wake up any higher priority task, then yield the processor to the
            //higher priority task which is just woken up.
        if(xHigherPriorityTaskWoken)
            {
                        taskYIELD();
            }
}
```

2.5 Modificaciones en Task5 de reporte de error y aumento de prioridad en la creación de la Task2

La Task5 simplemente actualiza un contador de errores, protegido por el mismo Mutex del conjunto de datos globales. La Tash7 de menú de usuario agregó la impresión de ese valor como última columna “errs” (Figura 2). Al crear la Task2 se aumentó su prioridad de 2 a 3, para que al salir de la ISR de UART1 sea la primera en entrar, y copiar el buffer leído para su posterior decodificación en el Task3.
A continuación se muestra la Task 5 modificada:

```
// Task5 vTask_PrintError solo en caso de recepcion erronea UART1
// 27.4.2019 _Incrementa contador de errores global, no envía a imprimir
void vTask_PrintError(void *params)
{
        // char pData[] = "Error en Recepcion";
        printmsg("\n\rTsk5");
        vTaskDelay(20);
        while(1)
        {
                xTaskNotifyWait(0,0,NULL,portMAX_DELAY);


                // Protect Access, increment counter
        xSemaphoreTake(xMutex, portMAX_DELAY);
        g_Rx_Errors++;
        xSemaphoreGive(xMutex);
        // Counter can be later printed with Task UART6
                // xQueueSend(output_write_queue,&pData,portMAX_DELAY);

                vTaskDelay(pdMS_TO_TICKS(50));
        }

}
```

En la creación de las tareas, se observa que la Task2 tiene mayor prioridad que las otras:

```
        if((packet_queue != NULL) && (output_write_queue != NULL) && (command_queue != NULL) && (xMutex !=
NULL) && xSem1 != NULL )
        {
                // Create task-1 vTask_Display
                // TaskHandle_t xTaskHandleDisplay -  Task 1 Display Handle
                xTaskCreate(vTask_Display,"TASK_DISPLAY-1",500,NULL,1,&xTaskHandleDisplay);

                // Create task-2 vTask_Check_Meteo_packet
                // TaskHandle_t xTaskHandleChkMeteo - Task 2 CheckMeteo Handle
                // 27.4.2019 Give higher priority to copy UART1 stringbuffer right away..
                xTaskCreate(vTask_Check_Meteo_packet,"TASK_CHK_MET_2",500,NULL,3,&xTaskHandleChkMeteo);
```

```
        // Create task-3 vTask_Process_OutdoorInfo
        // TaskHandle_t xTaskHandleProcOutd - Task 3 ProcessOutdoorInfo Handle
        xTaskCreate(vTask_Process_OutdoorInfo,"TASK_PROCESS_OUTD_3",500,NULL,2,&xTaskHandleProcOutd);

        // Create task-4 vTask_Write_Uart6
        // TaskHandle_t xTaskHandleWrUart6  - Task 4 Write_Uart6 Handle
        xTaskCreate(vTask_Write_Uart6,"TASK4-UART-WRITE",500,NULL,2,&xTaskHandleWrUart6);

        // Create task-5 vTask_PrintError
        // TaskHandle_t xPrintError  - Task 5
        xTaskCreate(vTask_PrintError,"TASK5-PRINTERROR",500,NULL,2,&xTaskHandlePrintErr);

        // Create task-6 vTask_HeartBeat
        // TaskHandle_t HeartBeat  - Task 6
        xTaskCreate(vTask_HeartBeat,"TASK6-HrtBEAT",500,NULL,2,&xTaskHandleHeartBeat);

        //create task-7 U6 cmd handle
        xTaskCreate(vTask_uart6_cmd_handling,"TASK7-U6CMD-HANDLING",500,NULL,2,&xTaskHandleUart6CmdH);

        //create task-8 U6 cmd process 25.4.Subimos Stack
        xTaskCreate(vTask_uart6_cmd_processing,"TASK8-U6CMD-ROCESS",700,NULL,2,&xTaskHandleUart6CmdP);

    // start the scheduler
    vTaskStartScheduler();
}
```

3. Instalación de la veleta / sensor de dirección de viento

La operación de la unidad se verificó con una veleta potenciométrica tipo NRG 200P que se instaló en el canal correspondiente de la unidad METEO (que cuenta con un procesaror PSoC 1 tipo 29466, y un canal destinado a la veleta con un ADC de 13 bits, además de circuito de acondicionamiento y protección). En la Figura 3 se muestra dicha conexión.

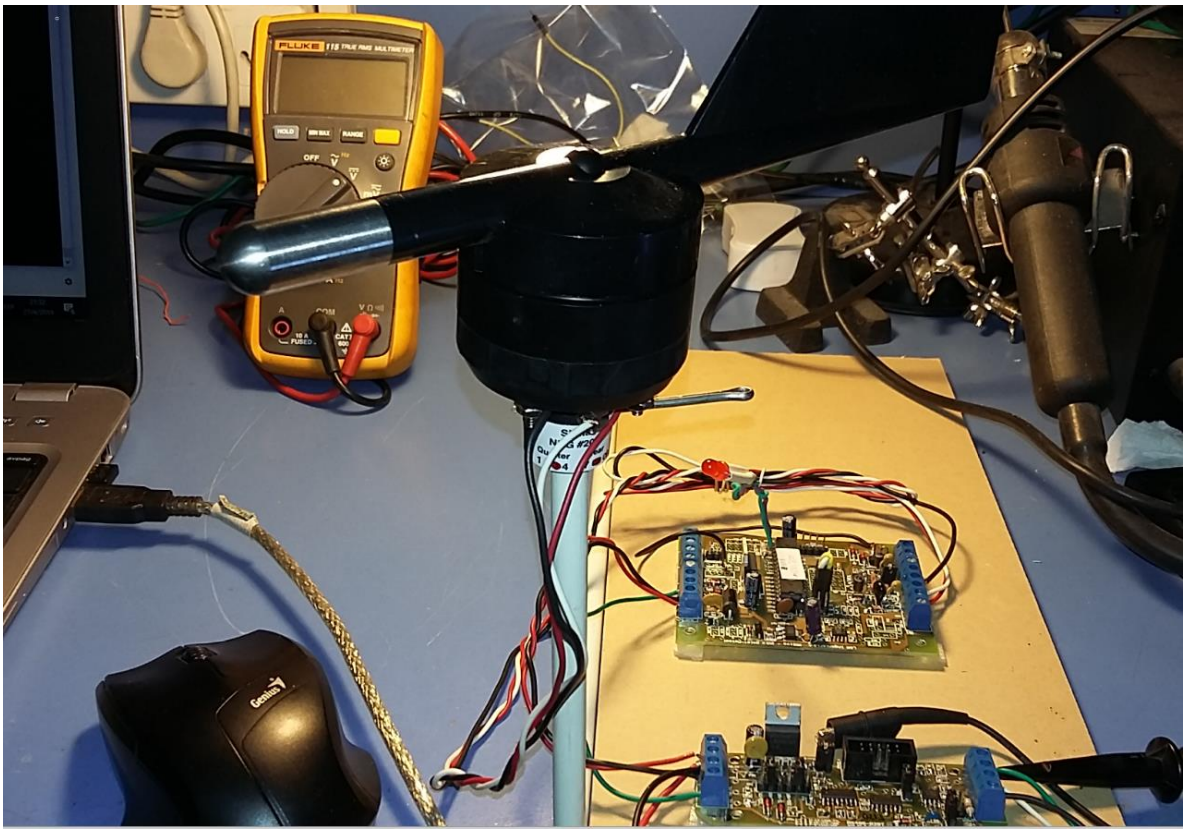

Figura 3 – Configuración del circuito METEO con la Veleta conectada

3.1 Salida de comando 3 / Direccion Viento

La salida del comando va variando con la dirección de viento. Se presenta como un entero (0-360 en grados x 10).

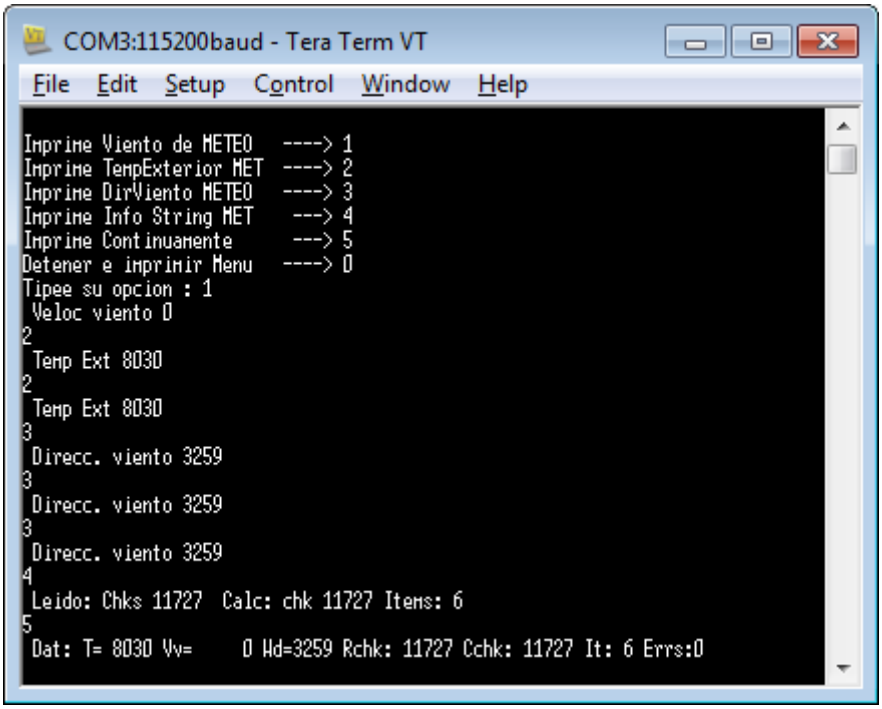

Figura 4 – Menú y resultados de distintos comandos, entre ellos el Comando 3 Dir Viento

3.2 Vista de la Placa en operación

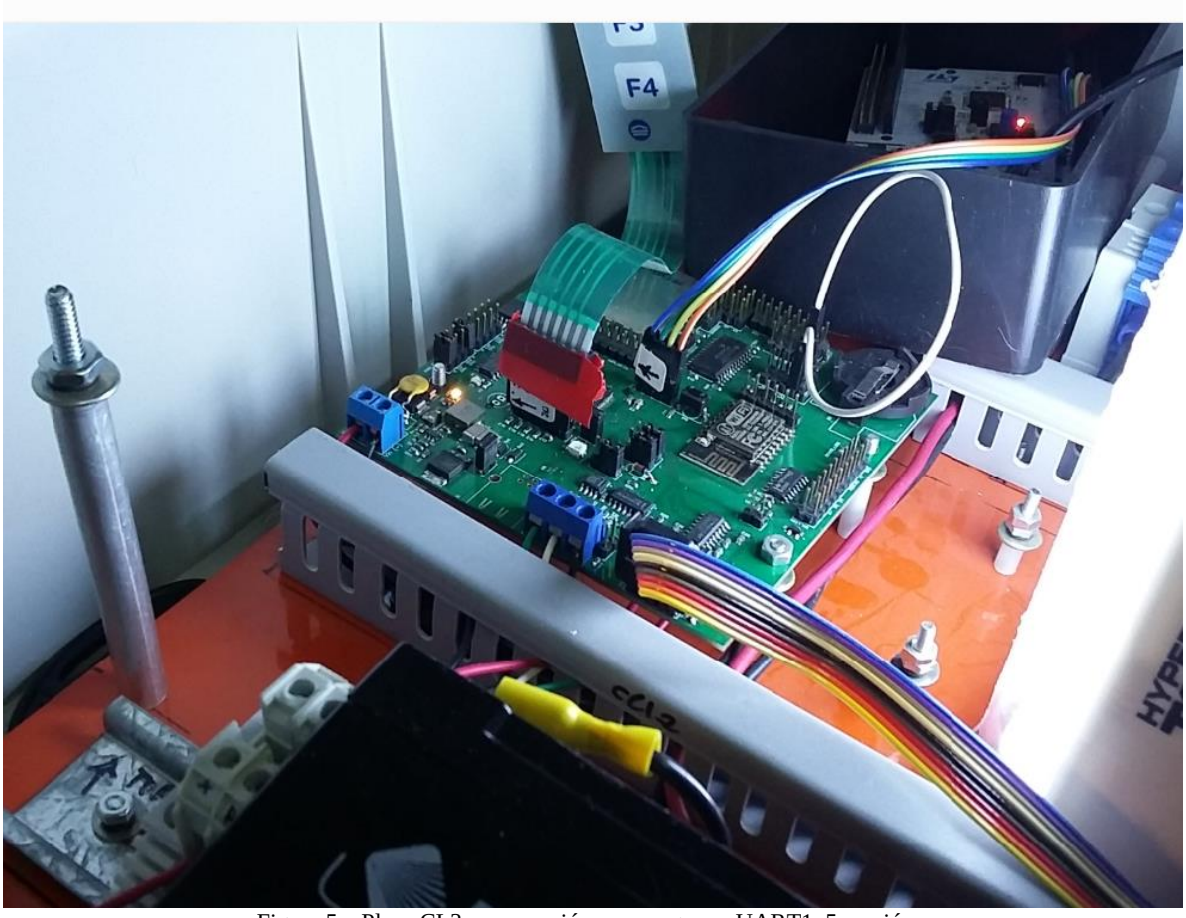

Figura 5 – Placa CL3 en operación con programa UART1_5 versión c

3.3 Video de operación con Veleta conectada:

https://www.youtube.com/watch?v=vWE3tUfOhIE&feature=youtu.be

ANEXO

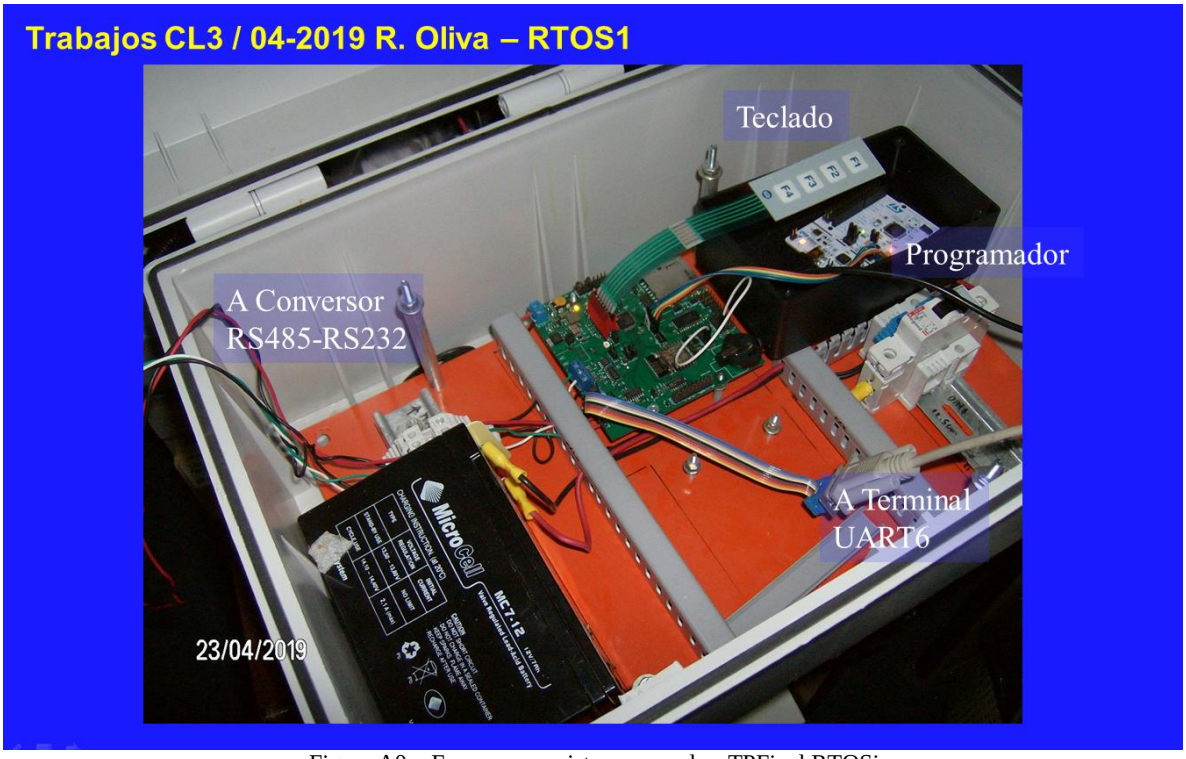

Figura A0 – Ensayos con sistema armado - TPFinal RTOSi

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

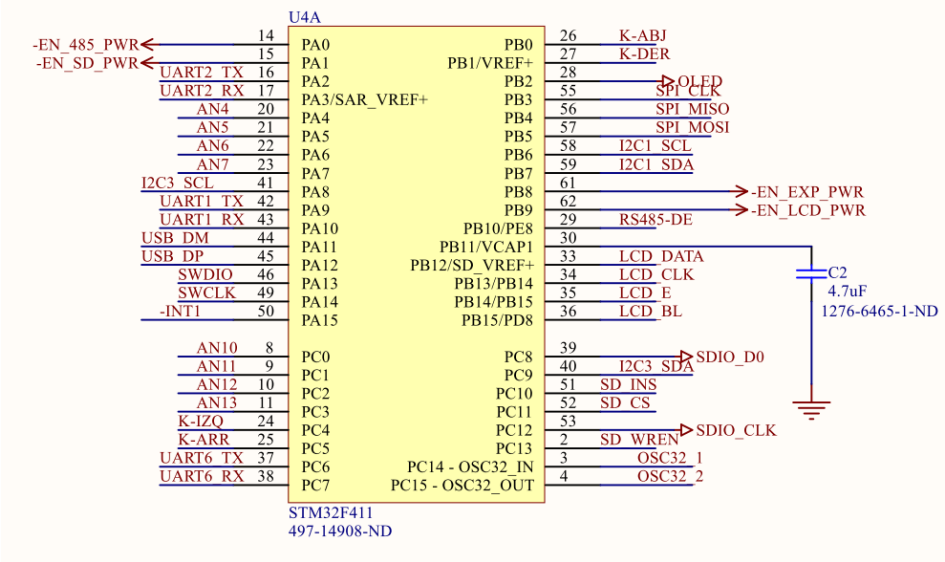

Figura A2 – Asignación de funciones en el STM32F411 de CL3

-0-