

**CONTENIDOS:**

1. Introducción
2. Avances logrados



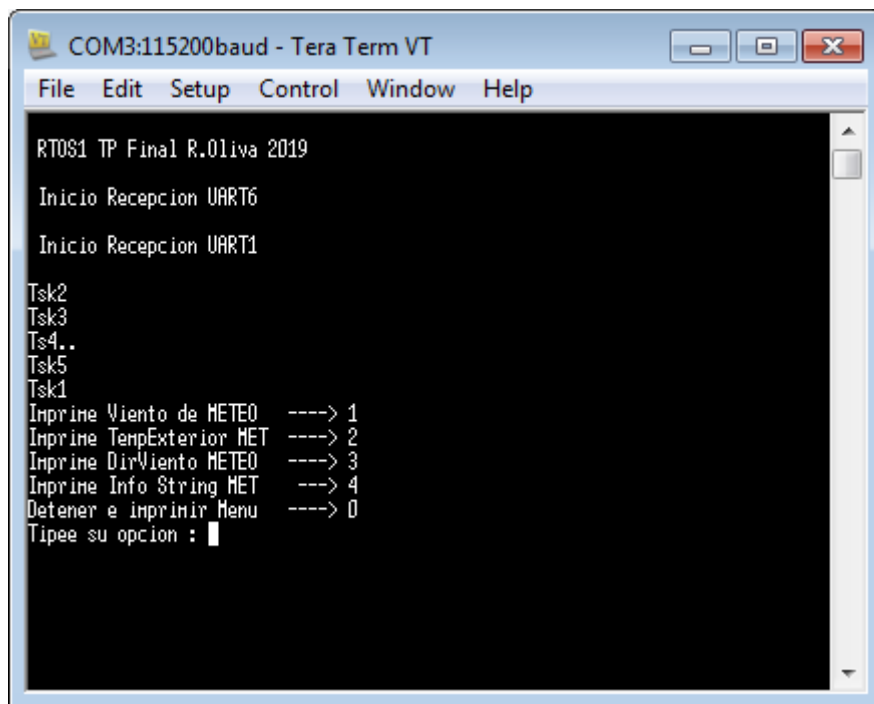


Figura 2 – Menu de usuario presentado - Trabajando con Paquete de Muestra

### 2.1.1 Salida de comando 1 – Intensidad de Viento

Corresponde al valor sssss del string fijo, o sea 15100 (Hz x 100, luego escalado de acuerdo a constantes de calibración)

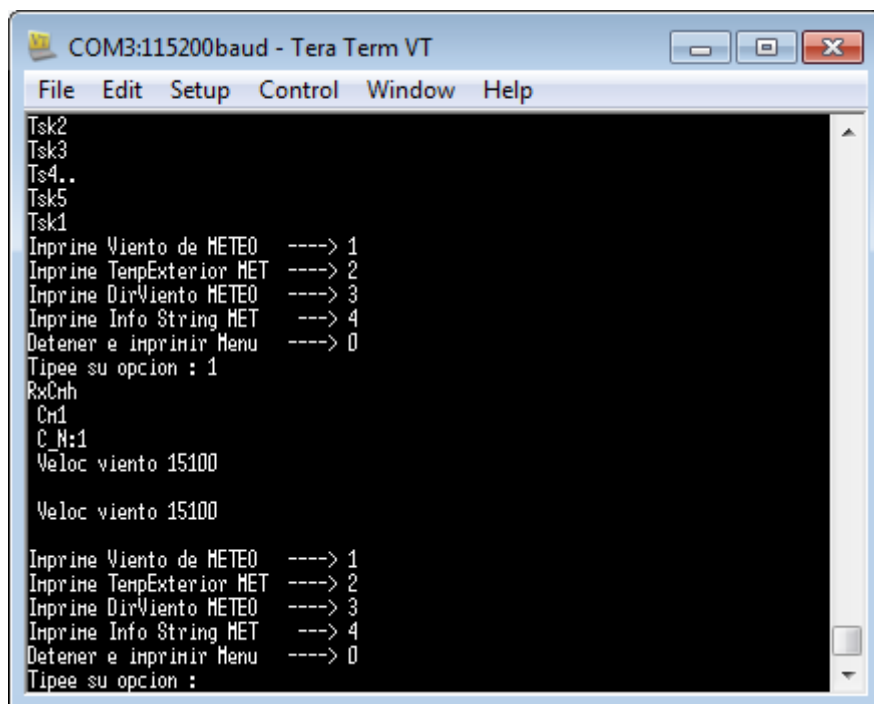
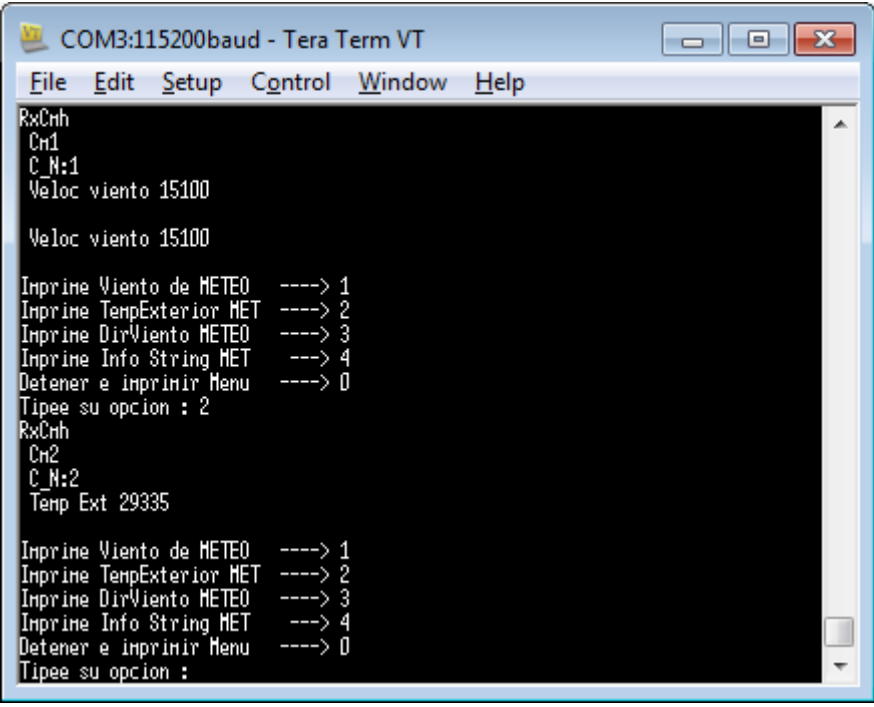


Figura 3 – Comando 1 intensidad de viento

### 2.1.2 Salida de comando 2 / Temp

Corresponde a la temperatura exterior, valor tttt (en Kelvin x 100) 29335 en el string fijo

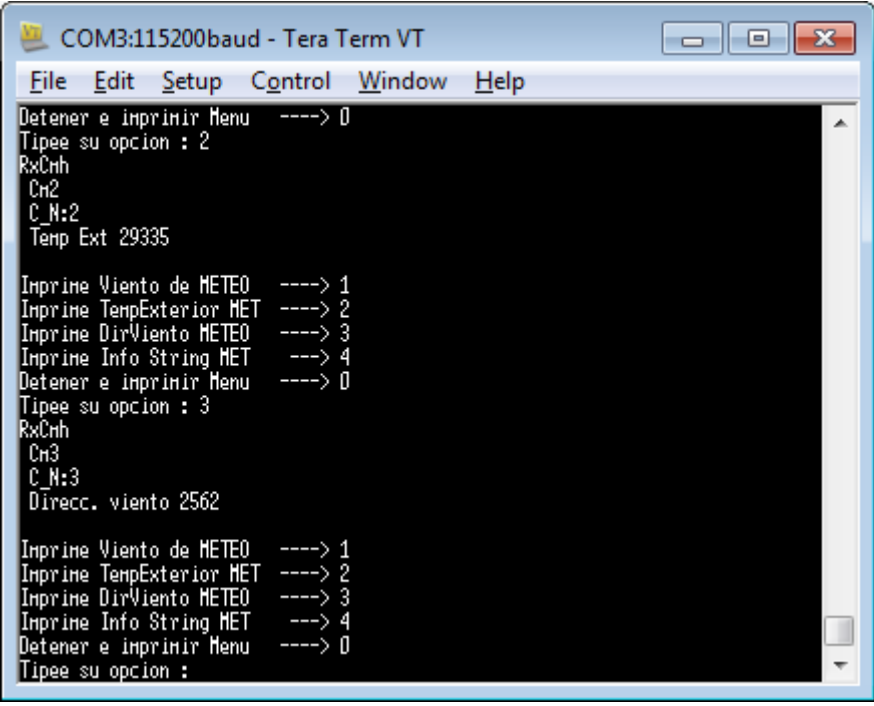


```
COM3:115200baud - Tera Term VT
File Edit Setup Control Window Help
RxChh
Cn1
C_N:1
Veloc viento 15100
Veloc viento 15100
Imprime Viento de METEO ----> 1
Imprime TempExterior MET ----> 2
Imprime DirViento METEO ----> 3
Imprime Info String MET ----> 4
Detener e imprimir Menu ----> 0
Tipee su opcion : 2
RxChh
Cn2
C_N:2
Temp Ext 29335
Imprime Viento de METEO ----> 1
Imprime TempExterior MET ----> 2
Imprime DirViento METEO ----> 3
Imprime Info String MET ----> 4
Detener e imprimir Menu ----> 0
Tipee su opcion :
```

Figura 3 – Comando 2 temperatura

### 2.1.3 Salida de comando 3 / Direccion Viento

Corresponde a la dirección de la veleta, valor dddd (0-360 en grados x 10) 2562 en el string fijo



```
COM3:115200baud - Tera Term VT
File Edit Setup Control Window Help
Detener e imprimir Menu ----> 0
Tipee su opcion : 2
RxChh
Cn2
C_N:2
Temp Ext 29335
Imprime Viento de METEO ----> 1
Imprime TempExterior MET ----> 2
Imprime DirViento METEO ----> 3
Imprime Info String MET ----> 4
Detener e imprimir Menu ----> 0
Tipee su opcion : 3
RxChh
Cn3
C_N:3
Direcc. viento 2562
Imprime Viento de METEO ----> 1
Imprime TempExterior MET ----> 2
Imprime DirViento METEO ----> 3
Imprime Info String MET ----> 4
Detener e imprimir Menu ----> 0
Tipee su opcion :
```

Figura 4 – Comando 3 Dir Viento

### 2.1.4 Salida de comando 4 / Info del String + checksum

Corresponde a los valores de:

```
char okTestCStr[] = "UUU$29335.10156.2562.15100.125.DFBE";
```

Checksum leído (chks = 57278 = 0xDFBE) , checsum calculado chk = 52278, e ítems leídos (7, incluyendo identificador inicial UUU\$)

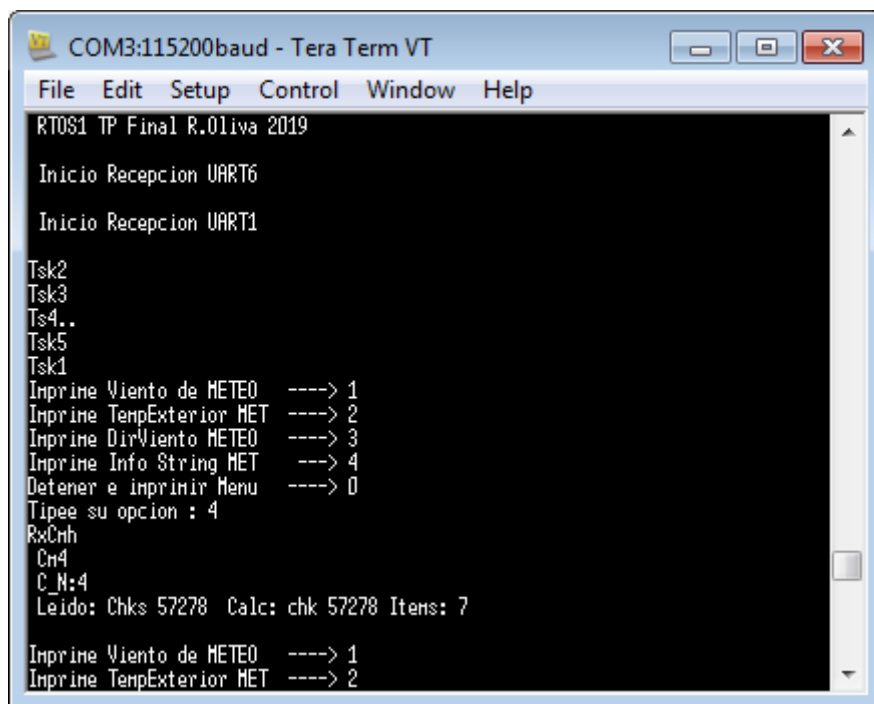


Figura 5 – Comando 4 Info String leído

### 2.1.5 Tarea Terminal UART6

Se muestra el procesamiento de los datos recibidos del usuario, en la Tarea 8-

```
// Task 8 void vTask_uart6_cmd_processing - ejecuta los comandos de la cola
// de comandos alimentada por el task anterior
// int16_t out_t = 0;
// int16_t out_b = 0;
// int16_t out_w_speed = 0;
// int16_t out_vbat = 0;
// int16_t out_w_dir = 0;
// uint16_t checksum = 0; // temporary variables
// Valores del String transmitido por METEO:
// v6 for use with METEO v20 uses packet "UUU$ttttt.bbbbb.dddd.sssss.vvv.CRCC*QQQ": (WSpeed is 5 chars long)
//   UUU$ 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
//   *QQQ end identifier
// Si se usa el de muestra:
// char okTestCStr[] = "UUU$29335.10156.2562.15100.125.1095";
// Imprime Viento de METEO ----> 1
// Imprime TempExterior MET ----> 2
// Imprime DirViento METEO ----> 3
// Imprime Info String MET ----> 4
// Detener e imprimir Menu ----> 0
// Tipee su opcion : ";
// Task 8 = Terminal UART6 proceso comandos

void vTask_uart6_cmd_processing(void *params)
{
    APP_CMD_t *new_cmd;
    char task_msg[70];
```

```

uint8_t local_command_sel = 0;
uint16_t u6chk = 0;
int16_t u6items = 0;
int16_t outtemp=0;
int16_t u6outbaro = 0;
int16_t u6outwindspeed = 0;
int16_t u6outwinddir = 0;
uint16_t u6checksum = 0;

while(1)
{
    xQueueReceive(command_queue, (void*)&new_cmd, portMAX_DELAY);

    #ifdef DEBUG_USART6
        sprintf(task_msg, "\r\n C_N:%d", new_cmd->COMMAND_NUM);
        printmsg(task_msg);
    #endif

    xSemaphoreTake(xMutex, portMAX_DELAY);
    outtemp = sensor_val.Vs_OTemp;
    u6outbaro = sensor_val.Vs_OBaro;
    u6outwindspeed = sensor_val.Vs_OWind;
    u6outwinddir = sensor_val.Vs_OWDir;
    u6checksum = g_checksum;
    u6chk = g_chk;
    u6items = g_items;
    xSemaphoreGive(xMutex);

    if(new_cmd->COMMAND_NUM == 1)
    {
        print_Uart6_messageCmd1(task_msg);
        print_Wind_Speed(task_msg, u6outwindspeed);
    }
    else if(new_cmd->COMMAND_NUM == 2)
    {
        print_Out_Temp(task_msg, outtemp);
    }
    else if(new_cmd->COMMAND_NUM == 3)
    {
        print_Wind_Dir(task_msg, u6outwinddir);
    }
    else if(new_cmd->COMMAND_NUM == 4 )
    {
        print_String_Items(task_msg, u6checksum, u6chk, u6items);
    }else
    {
        local_command_sel = 0;
        print_Uart6_error_message(task_msg);
    }

    // liberar memoria asignada a new_cmd
    vPortFree(new_cmd);
    vTaskDelay(pdMS_TO_TICKS(50));
    if (local_command_sel == 0){
        xTaskNotify(xTaskHandleDisplay, 0, eNoAction);
    }
}
}

```

## 2.2 Componente METEO / UART1

La parte de UART1 se ha ensayado parcialmente en la versión actual. 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 frecuencia de 1Hz.

## 2.2 UART1 Handler

Es el primer procesamiento de los datos que ingresan. En la versión de 8 bits se implementaba un buffer de 96 bytes, capaz de alojar 2 paquetes. En la presente versión se usan las rutinas LL de STM32, para chequear el flag “RX Not Empty” de la UART. Si hay datos, se leen y se copian a un buffer global packet\_buff, con un puntero packet\_len . La detección es por el último carácter ‘\*’ y la longitud adecuada, en cuyo caso se notifica a vTask\_CheckMeteo\_Packet() para que tome los datos y los copie, sinó se notifica al printError task. En ambos casos se resetea el puntero al inicio ( buffer\_len = 0).

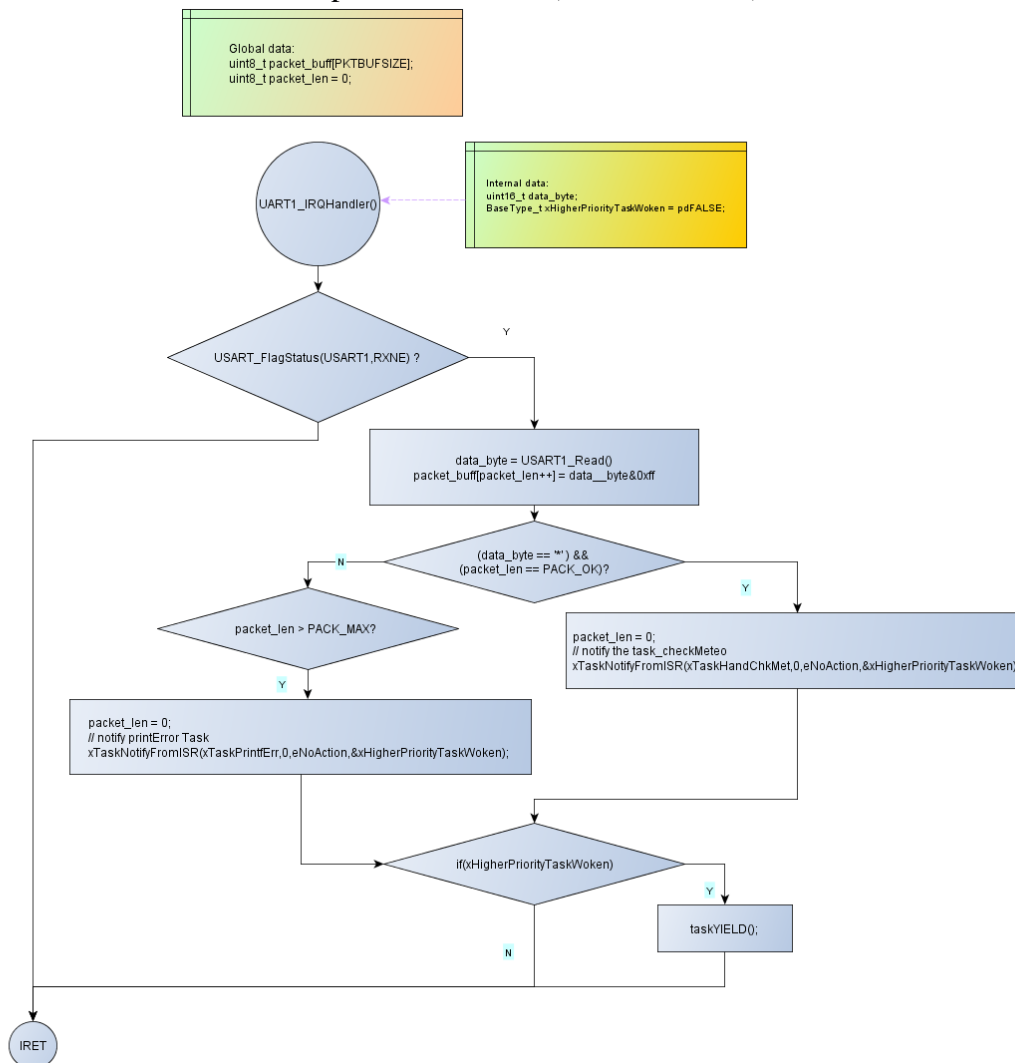


Figura 3 – Estructura del ISR UART1 Handler

## 2.2.1 Implementación del UART1\_Handler:

```
// USART1_ISR R.Oliva 18.4.2019
// Recibe datos de METEO a 38400 por USART1
// Global data:
// uint8_t packet_buffer[PKTBUFSIZE];
// uint8_t packet_len = 0;

// Packet:
// v6 for use with METEO v20 uses packet "UUU$ttttt.bbbbb.dddd.sssss.vvv.CRCC*QQQ": (WSpeed is 5 chars long)
//   UUU$ 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
//   *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$29335.10156.2562.15100.125.1095*QQQ";
//   Parsed originally (after $ detected) with
//   c = (char)sscanf(packet,"%5d.%5d.%4d.%5d.%3d.%4x",&out_t,
//   &out_b, &out_w_dir,&out_w_speed,
//   &out_vbat,&checksum);

void USART1_IRQHandler(void)
{
    /* USER CODE BEGIN USART1_IRQn 0 */
    uint16_t data_byte;
    BaseType_t xHigherPriorityTaskWoken = pdFALSE;

    if( USART_GetFlagStatus(USART1,USART_FLAG_RXNE) )
    {
        //a data byte is received from the user
        data_byte = USART_ReceiveData(USART1);

        packet_buffer[packet_len++] = (data_byte & 0xFF) ;

        if((data_byte == '*' ) && (packet_len == PACK_OK_LEN))
        {
            //then packet is ok..
            //reset the packet_len variable
            packet_len = 0;

            //notify the CheckMeteoHand task
            xTaskNotifyFromISR(xTaskHandChkMet,0,eNoAction,&xHigherPriorityTaskWoken);
        }
        else if(packet_len > PACK_MAX)
        {
            //then packet is corrupt..
            //reset the packet_len variable
            packet_len = 0;
            // notify printError Task
            xTaskNotifyFromISR(xTaskPrintfErr,0,eNoAction,&xHigherPriorityTaskWoken);
        }
        else{
            // do nothing
        }
    }

    // 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();
    }
}
/* USER CODE END USART1_IRQn 0 */
/* USER CODE BEGIN USART1_IRQn 1 */

/* USER CODE END USART1_IRQn 1 */
}
```



## 2.3 Task\_CheckMeteo()

El vTask\_CheckMeteo\_Packet() recibe los paquetes que están OK y copia el buffer de RX de la UART a un buffer "Packet\_OK", ahora pasado a memoria global. Este nuevo paquete se pone en la cola de "paquetes\_ok" para procesar por el vTask\_Process\_OutdoorInfo():

```
// 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, con uint8_t flag_demo_U1_sinCom = DEMO1_ON
// envía cada 1000 ms un paquete de muestra - sino en OFF
// espera notificación de un paquete via UART1
// 24.4.2019 Sacamos modo Demo, solo String ISR_UART1
// Copiado y pasado al Packet queue, delay 0
void vTask_Check_Meteo_packet(void *params)
{
    // uint8_t command_code=0;
    // typedef struct PACKET_OK
    // {
    //     uint8_t packet_content[PACK_OK_LEN+1];
    // } PACKET_OK_t;
    // PACKET_OK_t *new_packet;
#ifdef DEBUG_VERBOSE
    printmsg("\n\rTsk2");
#endif
    //vTaskDelay(20);
    while(1)
    {
        // Sacamos modo Demo, esperamos String de ISR_UART1
        // En vez de PortMaxDelay, que espere 1500 ms pdMS_TO_TICKS(1500));
        // xTaskNotifyWait(0,0,NULL,portMAX_DELAY);
        vTaskDelay(pdMS_TO_TICKS(500));

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

        //taskENTER_CRITICAL();
        // make a copy of the buffer_packet to new_packet
        // strncpy(new_packet->packet_content, packet_buffer,PACK_OK_LEN);
        strncpy(new_packet->packet_content, okTestCStr,PACK_OK_LEN);
        // Copiamos uno de muestra (corregido 25.42019)
        // char okTestCStr[] = "UUU$29335.10156.2562.15100.125.dfbe";
        // sprintf(new_packet->packet_content,"%s",okTestCStr);
        //taskEXIT_CRITICAL();

        // 24.4.2019 Delay
        xQueueSend(packet_queue,&new_packet, portMAX_DELAY);

        vTaskDelay(pdMS_TO_TICKS(50));
    }
}
```

## 2.4 vTask\_Process\_OutdoorInfo()

El vTask\_Process\_OutdoorInfo () recibe un paquete de la cola de los OK y separa los valores recibidos en las variables enteras int16\_t out\_t,out\_b, out\_w\_speed,out\_vbat, out\_w\_dir,checksum; (todavía no escaladas), y los almacena en una estructura protegida por un Mutex.

```
// Task3 vTask_Process_OutdoorInfo - el paquete
// de METEO por UART1 es procesado para sacar la info
// 24.4.2019
// Menu por USART 6
// Nuevo menu 24.4.2019
// char menu[]={
// Imprime Viento de METEO ----> 1
// Imprime TempExterior MET ----> 2
// Imprime DirViento METEO ----> 3
// Imprime Info S Completo MET---> 4
// nDetener e imprimir Menu ----> 0
// 24.4.19 Copiar datos a
// Control
// int16_t g_chk = 0;
// in t16_t g_items = 0;
```

```

// V_SENSOR_t sensor_val;
//typedef struct sensor_holder {
//    int16_t IS_Aero;        // Current in 0-5V - Sampled
//    int16_t Vs_Vbat;        // Bat. Voltage in 0-5V - Sampled
//    int16_t Vs_OWind;       // OutDoor Wind Freq [Hz] from METEO / COM1
//    int16_t Vs_OWDir;       // OutDoor WindDirection 0-360.0 [°] from METEO / COM1
//    int16_t Vs_OTemp;       // External Temp 0-5V from METEO+NOMAD2/COM1
//    int16_t Vs_OBaro;       // Barometric Pressure 0-5V from METEO+NOMAD2/COM1
// V_SENSOR_t;

void vTask_Process_OutdoorInfo(void *params)
{
    // PACKET_OK_t *new_packet;
    // char task_msg[50];
    char out_id[6];
    uint16_t chk = 0;
    int16_t items = 0;
    int16_t out_t = 0;
    int16_t out_b = 0;
    int16_t out_w_speed = 0;
    int16_t out_vbat = 0;
    int16_t out_w_dir = 0;
    uint16_t checksum = 0; // temporary variables 25.4.19 checksum unsigned
    //uint32_t toggle_duration = pdMS_TO_TICKS(500);
#ifdef DEBUG_VERBOSE
    printmsg("\n\rTsk3");
#endif

    while(1)
    {
        xQueueReceive(packet_queue, (void*)&new_packet, portMAX_DELAY);
        // printmsg("\n\r1");

        items = sscanf(new_packet.packet_content, "%4s%5hd.%5hd.%4hd.%5hd.%3hd.%4hx", &out_id, &out_t,
            &out_b, &out_w_dir, &out_w_speed, &out_vbat, &checksum);

        // print_items_message(items);
        // void print_items_message(char *task_msg, int16_t items)
        // sprintf("\n\r I %d, o_v %d", items, out_vbat);
        // printmsg(task_msg);

        chk = out_t + out_b + out_w_dir;
        chk += out_w_speed + out_vbat;

        // Protect Access
        xSemaphoreTake(xMutex, portMAX_DELAY);

        sensor_val.Vs_OTemp = out_t;
        sensor_val.Vs_OBaro = out_b;
        sensor_val.Vs_OWind = out_w_speed;
        sensor_val.Vs_OWDir = out_w_dir;
        g_checksum = checksum;
        g_chk = chk;
        g_items = items;

        xSemaphoreGive(xMutex);

        /* solucion anterior
        if((chk == checksum) && (items == 7))
        {
            print_Wind_Speed(task_msg, out_w_speed);
        }
        else
        {
            print_error_message(task_msg);
        }
        */
        // enviar a QB_output a Terminal, delay 0
        // xQueueSend(output_write_queue, &task_msg, portMAX_DELAY);

        // liberar memoria de new_packet - ya no usado
        // vPortFree(new_packet);
        vTaskDelay(pdMS_TO_TICKS(100));
    }
}

```

## Trabajos CL3 / 04-2019 R. Oliva – RTOS1

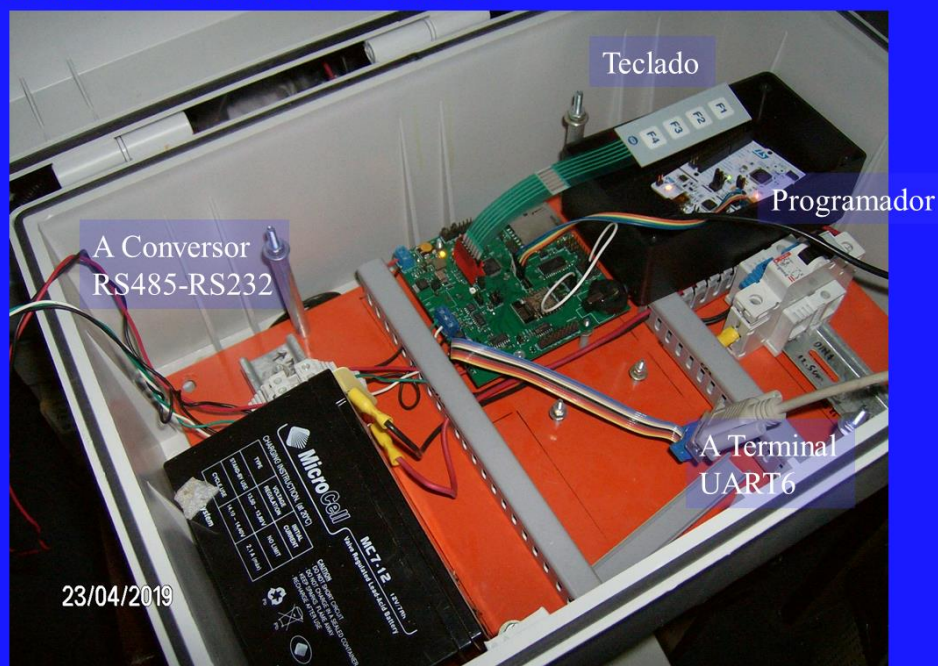


Figura A0 – Ensayos con sistema armado - TPFinal RTOSi

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

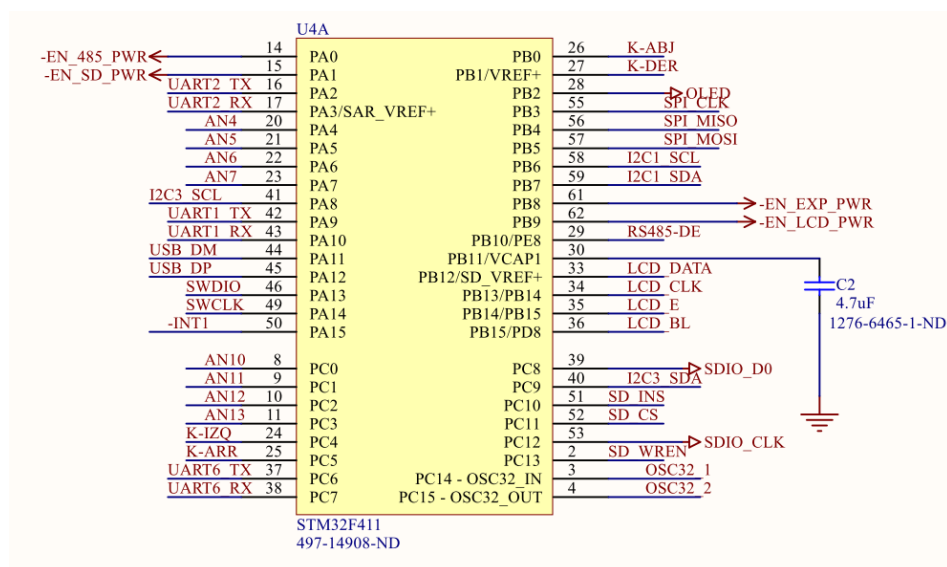


Figura A2 – Asignación de funciones en el STM32F411 de CL3

-0-