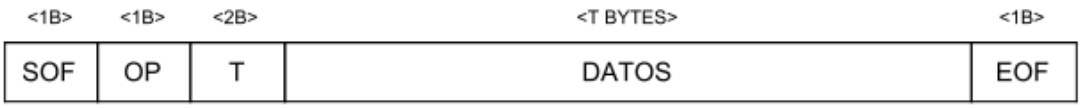
TRABAJO FINAL – RTOS 2 – CESE
GRUPO 1 v22-05-2019

INTEGRANTES:
Julian Bustamante Narvaez
Jacobo Salvador
Gustavo Paredes D.
Rafael Oliva

Introducción

PROBLEMA

Paquetes de datos:



Delimitación de paquete: 00 a 99

SOF: Carácter '{'
EOF: Carácter '}'.

Campos:

OP (Operación):
0: Convertir los caracteres recibidos a mayúsculas. (CMD/RTA)
1: Convertir los caracteres recibidos a minúsculas. (CMD/RTA)
2: Reportar stack disponible (RTA)
3: Reportar heap disponible. (RTA)

T (Tamaño): El tamaño del campo DATOS. "00" a "99" o sea máximo 99 bytes
DATOS: Texto a procesar. Deben ser caracteres ASCII legibles.

1. VARIANTES REVISADAS

UART –ISR + Task Ppal / Rafael

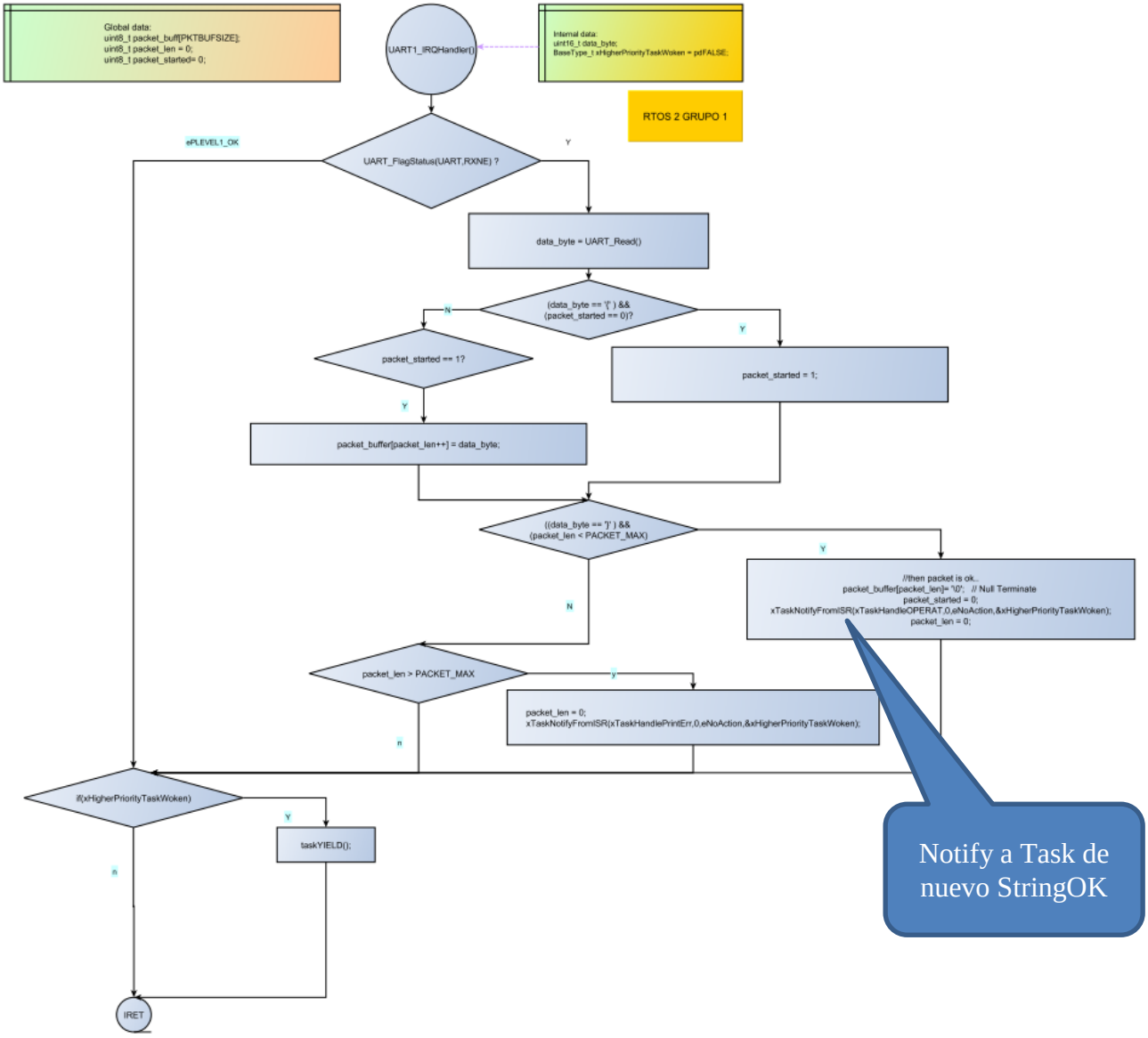
1.a Posible ISR de UART – Alt1 (PRESENTADA 18/5/19)

La ISR contiene una mini-FSM de detección del carácter de inicio '$' (ahora '{') y del de fin '*' (ahora ' }'),.cuando esto llega bien, usa un xNotify() al Task ppal. (en amarillo)

```
// 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 CheckHandleOPERAT task (to copy the buffer)
                                xTaskNotifyFromISR(xTaskHandleOPERAT,0,eNoAction,&xHigherPriorityTaskWoken);
                                //reset the packet_len variable
                                packet_len = 0;
                                } // F1
```

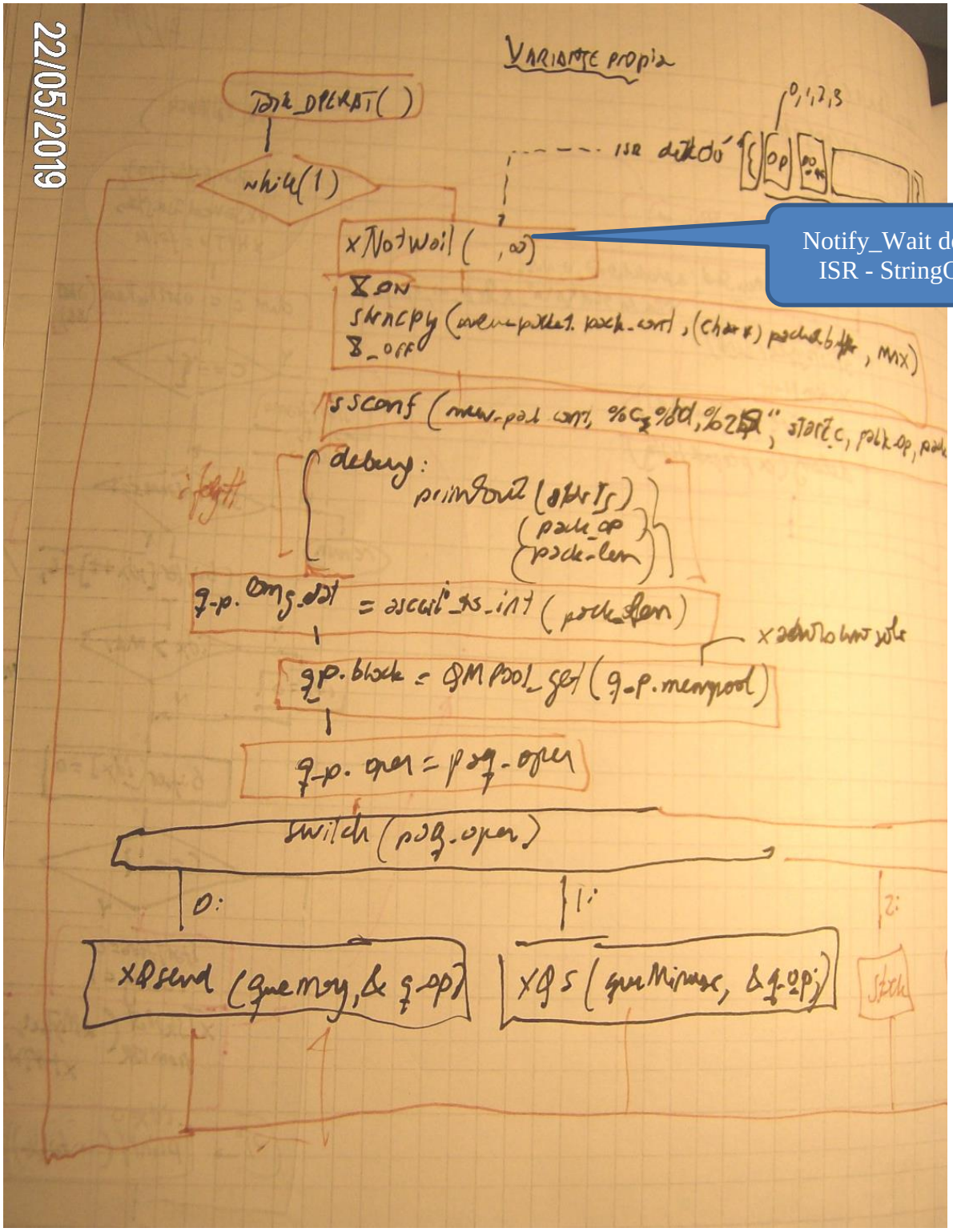
```
                        if (packet_len > PACKET_MAX){
                                packet_len = 0;
                                xTaskNotifyFromISR(xTaskHandlePrintErr,0,eNoAction,&xHigherPriorityTaskWoken);
                            }
                        }   // End
                    } // 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();
            }
}
```

1.b DIAGRAMA de ISR_ALT1)



1.c DIAGRAMA de Task_OPERAT() que copia el buffer de la ISR string a un string, decodifica los primeros '{', OP, Nº de bytes (00 a99), pide memoria y decide en base a OP lo que hay que hacer:

a. **Mayusculizar,**
b. **Minusculizar**
c. **Reportar Stack**
d. **Reporta Heap**



1.d Falta crear:

1.d.1 Task Mayusculizar , Minusculizar, Stack, Heap (todos reciben en su cola propia los strings de tamaño variable)

1.d.2 Task para enviar Strings al COM de salida

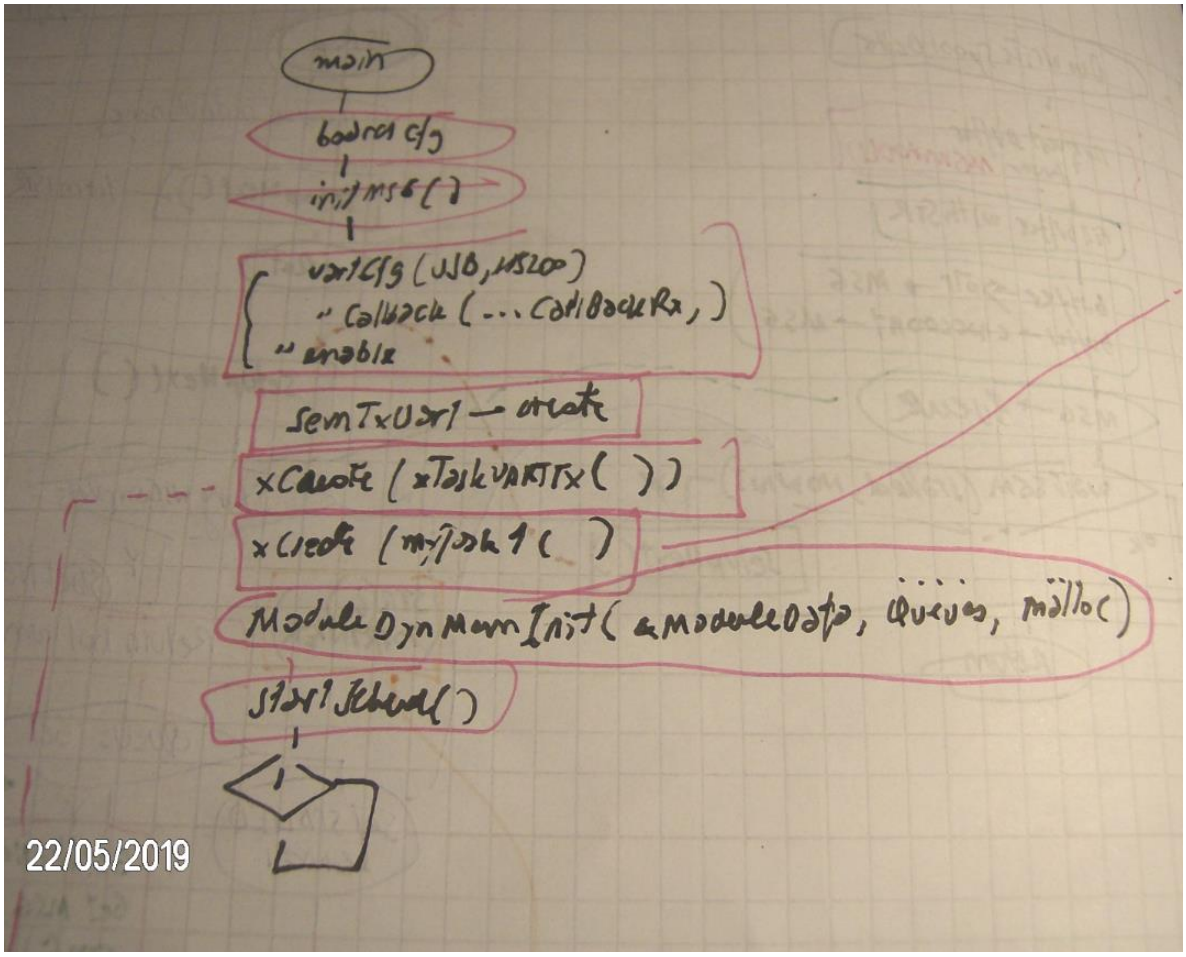
1.e Codigo tentativo de Task_Operat(), aquí con un llamado a una FuncionFSM_RX() , notificada desde rutina de ISR, pide la memoria y decodifica la OP requerida:

```
// AHORA nuevo ISR detecta '{'  y copia hasta '}', y llama a Funcion FSM_RX() que determina que hacer

void TaskHandleOPERAT (void *params)
{
//      typedef struct PACKET_OK
```

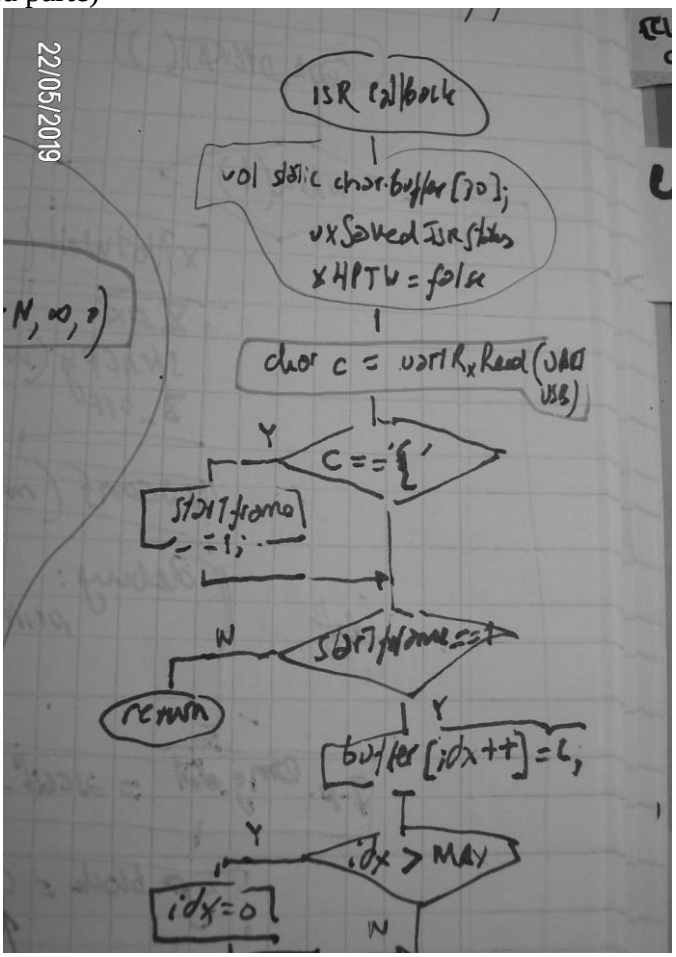
```
//      {
//        char packet_content[PACK_OK_LEN+2];
//      } PACKET_OK_t;
// PACKET_OK_t new_packet; // (as global)
//
//vTaskDelay(20);
        while(1)
        {
        // 27.4.2019 - Sacamos modo Demo, esperamos String de ISR_UART1
                // se utiliza xTskNotfromISR() allí.
                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);

                taskEXIT_CRITICAL();

                Funcion_FSM_RX();
                // 24.4.2019 Delay
                vTaskDelay(pdMS_TO_TICKS(50));
        }

}
```

2. VARIANTE 2

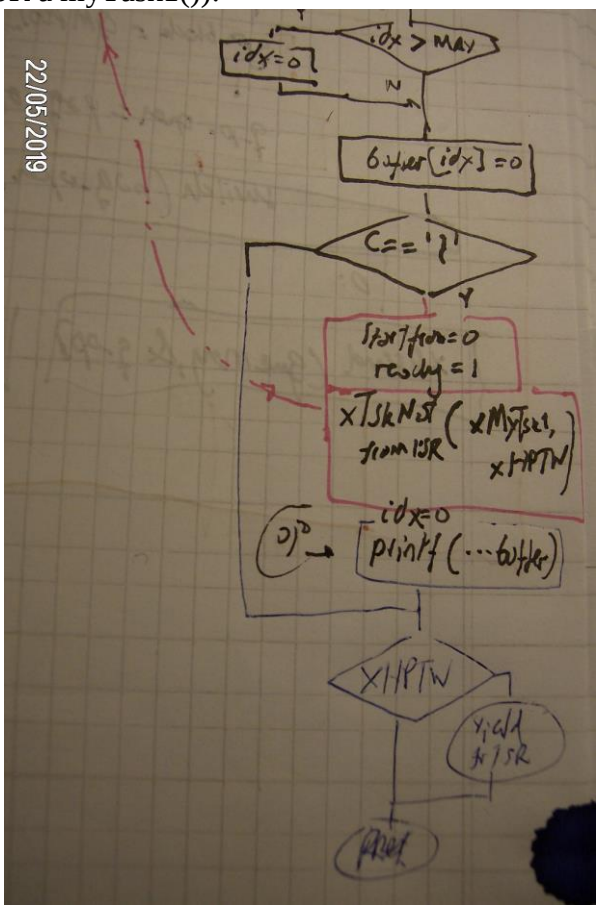
2.a Main() de Julian: (ya implementado en EDU-CIAA)



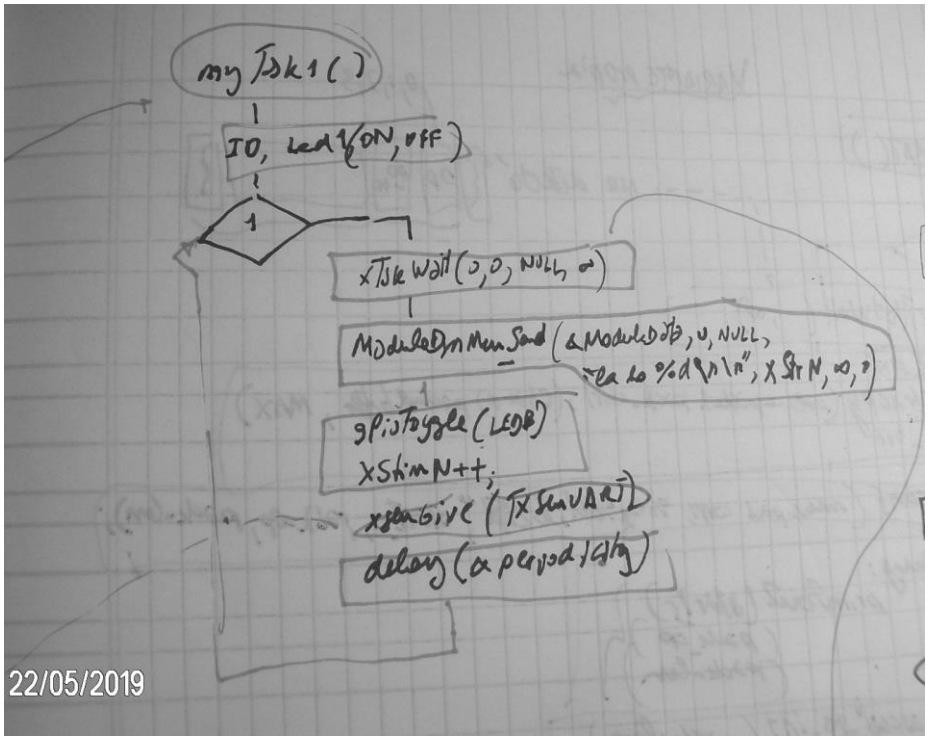
2.b ISR_Callback de Julian (1ra parte)



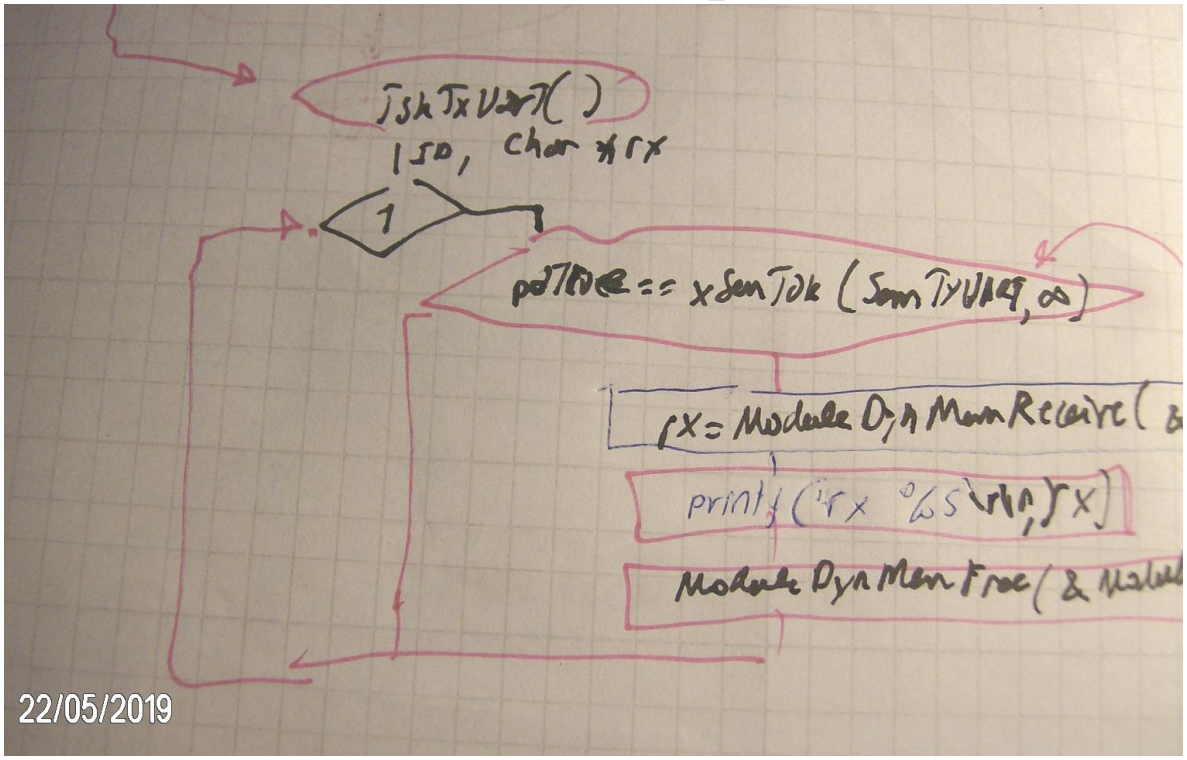
2da parte (c/ Notify from ISR a myTask1()):



2.c MyTask1() de Julian, recibe la notificación de la ISR, y entrega Semaforo para imprimir – Antes llama al ModuleDinMem_Send():



2.d Print Task de Julian / envía cola de salida a UART_USB



ANEXO

-0-