**TALLER PRACTICO UART**

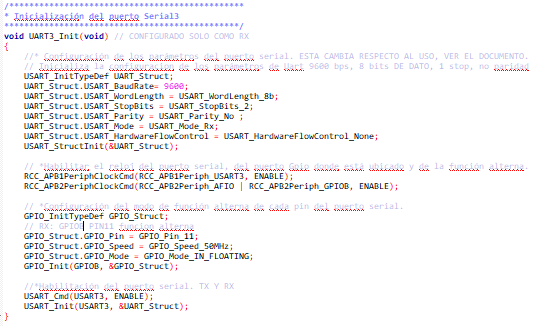
**CARLOS ANDRES AGUILERA HERRERA**

**2020-2**

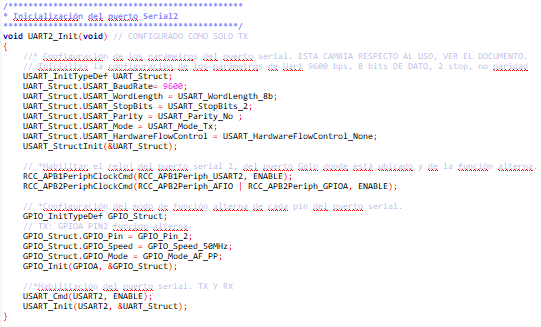
***1. CONFIGURACION DE LOS PUERTOS SERIALES.***

* Inicialice cada uno de los puertos seriales con una velocidad de 9.600 baudios, 8 bits de dato, sin paridad, 2 bits de parada y sin control de flujo. Al USART2 lo configura solo de transmisión, al USART3 solo de recepción y al USART1 de transmisión y recepción.

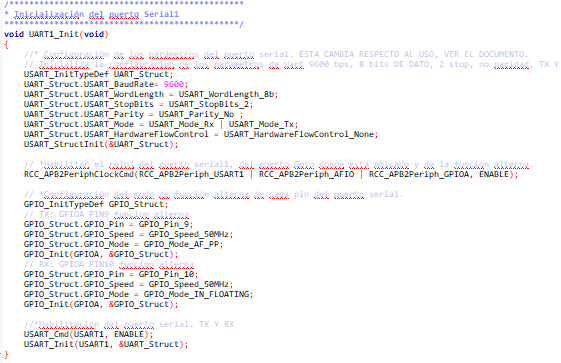
**CONFIGURACIÓN PUERTO SERIAL 3**



**CONFIGURACIÓN PUERTO SERIAL 2**

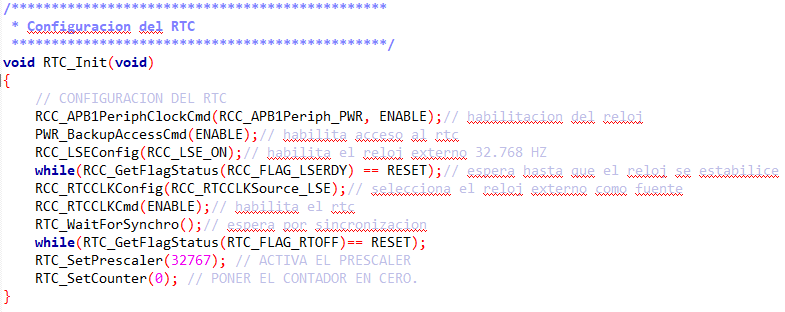


**CONFIGURACIÓN PUERTO SERIAL 1**

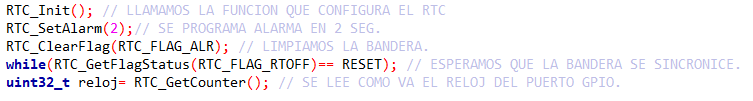


***2. Inicialice el contador RTC para que de una alarma cada dos segundos.***

**CONFIGURACION DEL RTC**



**CONFIGURACION DENTRO DEL MAIN ANTES DEL WHILE.**



***3. Dentro del main() crea una variable entera de 16 bits sin signo con el nombre de***

***“contador” y asignele el valor de cero.***



***4. Dentro del main() crea una variable entera de 16 bits sin signo con el nombre de***

***“direccion” y asignele el valor de cero. Esta variable tiene cero para indicar que el***

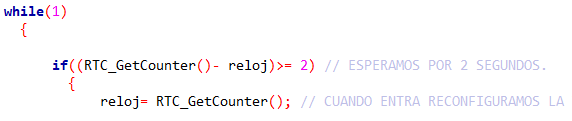
***conteo es ascendente y uno para indicar que el conteo es descendente.***



***5. Dentro del while() debe observar en qué momento se completó los dos segundos del***

***contador RTC. El programa no debe seguir hasta que se haya completado el tiempo de***

***dos segundos.***



**6. Envíe el valor de la variable “contador” por el puerto serial USART2.**



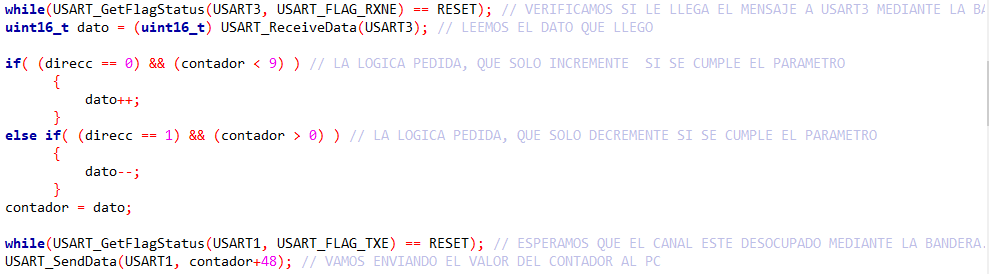
***7. Revise que haya llegado algún dato en el puerto serial USART3. Si llegó un dato, lea***

***este dato y lo incrementa en uno si la variable “direccion” es cero, o lo decrementa en***

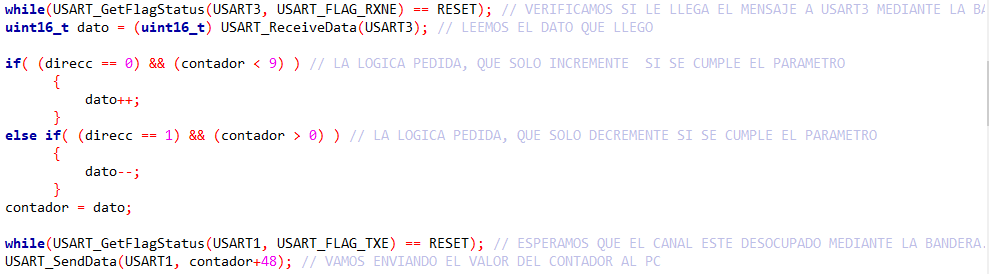
***uno si la variable “direccion” es uno. La variable “contador” no debe modificarse si la***

***variable “contador” es cero y la variable “direccion” es uno, ni tampoco si la variable***

***“contador” es nueve y la variable “direccion” es cero.***

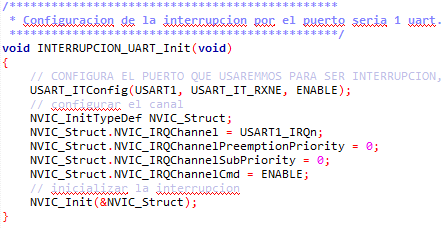


***8. Envíe el valor de la variable “contador” por el puerto serial USART1.***

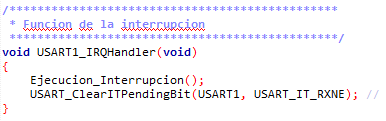


***9. En cualquier momento verifique si ha llegado algún dato en el puerto serial USART1. Si llega un dato, lo lee y si este dato es ‘>’, debe poner la variable “direccion” en cero. Si el dato que llegó es ‘<’, debe poner la variable “direccion” en uno.***

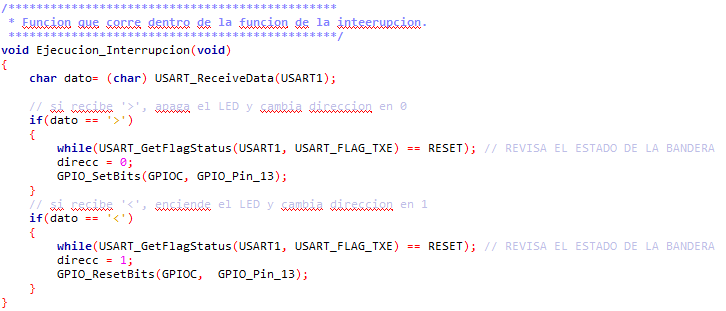
**CONFIGURAMOS EL PUERTO SERIAL 1 PARA SER USADO POR INTERRUPCION.**



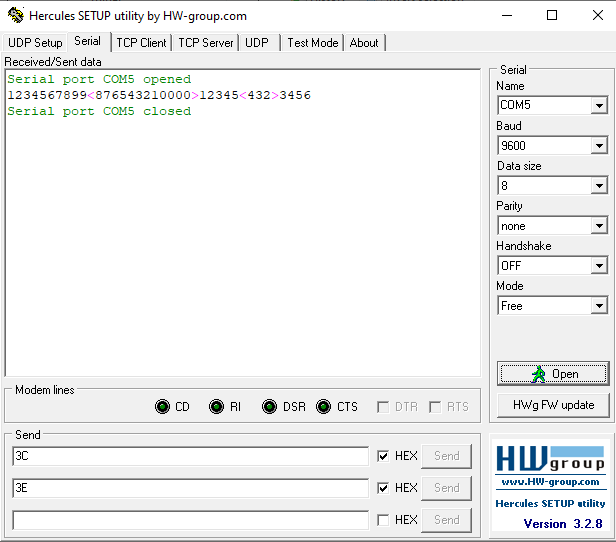
**ESTA ES LA FUNCION DE LA INTERRUPCION POR EL UART 1**



**ESTA ES LA FUNCION QUE SE CORRE DENTRO DE LA FUNCION DE INTERRUPCION, AQUÍ SE LEE EL DATO ENVIADO POR EL UART, Y DEPENDIENDO SI ES < O > CAMBIA EL ESTADO DE LA VARIABLE DIRECC Y ENCIENDE O APAGA EL LED.**



***RESULTADOS:* El video funcionando ya fue mostrado en clase.**



***A CONTINUACION DE ADJUNTA TODO EL CODIGO:***

#include "stm32f10x\_conf.h"

// COMUNICACION ENTRE LOS 3 UARTS, EN EL TALLER ESTA MAS EXPLICITO..

// SERIAL 1 (RX Y TX)

// SERIAL 2 (TX)

// SERIAL 3 (RX)

// ASCII https://datasagar.com/everything-about-ascii-in-c/

void LED\_Init(void);

void UART1\_Init(void);

void UART2\_Init(void);

void UART3\_Init(void);

void RTC\_Init(void);

void INTERRUPCION\_UART\_Init(void);

void Ejecucion\_Interrupcion(void);

void USART1\_IRQHandler(void);

uint16\_t direcc= 0; // Esta variable tiene cero para indicar que el conteo es ascendente y uno para indicar que el conteo es descendente.

int main(void)

{

LED\_Init(); // LLAMAMOS LA FUNCION QUE CONFIGURA EL PUERTO C CON EL LED

UART1\_Init(); // LLAMAMOS LA FUNCION QUE CONFIGURA LOS PUERTOS SERIALES.

UART2\_Init();

UART3\_Init();

INTERRUPCION\_UART\_Init(); // LLAMAMOS LA FUNCION QUE CONFIGURA LA INTERRUPCION POR UART.

RTC\_Init(); // LLAMAMOS LA FUNCION QUE CONFIGURA EL RTC

RTC\_SetAlarm(2);// SE PROGRAMA ALARMA EN 2 SEG.

RTC\_ClearFlag(RTC\_FLAG\_ALR); // LIMPIAMOS LA BANDERA.

while(RTC\_GetFlagStatus(RTC\_FLAG\_RTOFF)== RESET); // ESPERAMOS QUE LA BANDERA SE SINCRONICE.

uint32\_t reloj= RTC\_GetCounter(); // SE LEE COMO VA EL RELOJ DEL PUERTO GPIO.

uint16\_t contador= 0;// variables que se incrementa o decrementa.

while(1)

{

if((RTC\_GetCounter()- reloj)>= 2) // ESPERAMOS POR 2 SEGUNDOS.

{

reloj= RTC\_GetCounter(); // CUANDO ENTRA RECONFIGURAMOS LA VARIABLE RELOJ PARA ENTRAR SOLO CUANDO PASAN 2 SEG.

while(USART\_GetFlagStatus(USART2, USART\_FLAG\_TXE) == RESET); // ESPERAMOS QUE EL CANAL ESTE DESOCUPADO PARA ENVIAR.

USART\_SendData(USART2, contador); // ENVIAMOS EL VALOR DE CONTADOR DESDE USART2

while(USART\_GetFlagStatus(USART3, USART\_FLAG\_RXNE) == RESET); // VERIFICAMOS SI LE LLEGA EL MENSAJE A USART3 MEDIANTE LA BANDERA

uint16\_t dato = (uint16\_t) USART\_ReceiveData(USART3); // LEEMOS EL DATO QUE LLEGO

if( (direcc == 0) && (contador < 9) ) // LA LOGICA PEDIDA, QUE SOLO INCREMENTE SI SE CUMPLE EL PARAMETRO

{

dato++;

}

else if( (direcc == 1) && (contador > 0) ) // LA LOGICA PEDIDA, QUE SOLO DECREMENTE SI SE CUMPLE EL PARAMETRO

{

dato--;

}

contador = dato;

while(USART\_GetFlagStatus(USART1, USART\_FLAG\_TXE) == RESET); // ESPERAMOS QUE EL CANAL ESTE DESOCUPADO MEDIANTE LA BANDERA.

USART\_SendData(USART1, contador+48); // VAMOS ENVIANDO EL VALOR DEL CONTADOR AL PC

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Inicialización del puerto C, pin 13

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void LED\_Init(void)// ESTO ES HECHO PARA EL GPIO DEL LED PERO SE PUEDE HACER EL MISMO PROCEDIMIENTO PARA CUALQUIER GPIO.

{

GPIO\_InitTypeDef GPIOC\_Struct; // CREO LA ESTRUCTURA DEINICIALIZACION, TENIENDO EN CUENTA QUE ES EL PUERTO C

GPIOC\_Struct.GPIO\_Pin = GPIO\_Pin\_13; // SELECCIONO EL PIN A USAR DEL PUERTO C

GPIOC\_Struct.GPIO\_Speed = GPIO\_Speed\_2MHz; // SELECCIONO LA VELOCIDAD DEL PUERTO. PUEDEN SER 2M, 10M Y 50M.

GPIOC\_Struct.GPIO\_Mode = GPIO\_Mode\_Out\_PP; // SELECCIONO EL MODO DEL PUERTO, EN ESTE CASO ES SALIDA PUSH PULL.

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOC, ENABLE); // HABILIDO EL PUERTO C

GPIO\_Init(GPIOC, &GPIOC\_Struct); // INICIALIZO LA ESTRUCTURA.

GPIO\_ResetBits(GPIOC, GPIO\_Pin\_13);// PRENDO EL LED.

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Inicialización del puerto Serial1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void UART1\_Init(void)

{

//\* Configuración de los parámetros del puerto serial. ESTA CAMBIA RESPECTO AL USO, VER EL DOCUMENTO.

// Inicializa la configuracion de los parámetros de Uart 9600 bps, 8 bits DE DATO, 2 stop, no paridad, TX Y RX.

USART\_InitTypeDef UART\_Struct;

UART\_Struct.USART\_BaudRate= 9600;

UART\_Struct.USART\_WordLength = USART\_WordLength\_8b;

UART\_Struct.USART\_StopBits = USART\_StopBits\_2;

UART\_Struct.USART\_Parity = USART\_Parity\_No ;

UART\_Struct.USART\_Mode = USART\_Mode\_Rx | USART\_Mode\_Tx;

UART\_Struct.USART\_HardwareFlowControl = USART\_HardwareFlowControl\_None;

USART\_StructInit(&UART\_Struct);

// \*Habilitar el reloj del puerto serial1, del puerto Gpio donde está ubicado y de la función alterna.

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_USART1 | RCC\_APB2Periph\_AFIO | RCC\_APB2Periph\_GPIOA, ENABLE);

// \*Configuración del modo de función alterna de cada pin del puerto serial.

GPIO\_InitTypeDef GPIO\_Struct;

// TX: GPIOA PIN9 funcion alterna

GPIO\_Struct.GPIO\_Pin = GPIO\_Pin\_9;

GPIO\_Struct.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_Struct.GPIO\_Mode = GPIO\_Mode\_AF\_PP;

GPIO\_Init(GPIOA, &GPIO\_Struct);

// RX: GPIOA PIN10 funcion alterna

GPIO\_Struct.GPIO\_Pin = GPIO\_Pin\_10;

GPIO\_Struct.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_Struct.GPIO\_Mode = GPIO\_Mode\_IN\_FLOATING;

GPIO\_Init(GPIOA, &GPIO\_Struct);

//\*Habilitación del puerto serial. TX Y RX

USART\_Cmd(USART1, ENABLE);

USART\_Init(USART1, &UART\_Struct);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Inicialización del puerto Serial2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void UART2\_Init(void) // CONFIGURADO COMO SOLO TX

{

//\* Configuración de los parámetros del puerto serial. ESTA CAMBIA RESPECTO AL USO, VER EL DOCUMENTO.

// Inicializa la configuracion de los parámetros de Uart 9600 bps, 8 bits DE DATO, 2 stop, no paridad

USART\_InitTypeDef UART\_Struct;

UART\_Struct.USART\_BaudRate= 9600;

UART\_Struct.USART\_WordLength = USART\_WordLength\_8b;

UART\_Struct.USART\_StopBits = USART\_StopBits\_2;

UART\_Struct.USART\_Parity = USART\_Parity\_No ;

UART\_Struct.USART\_Mode = USART\_Mode\_Tx;

UART\_Struct.USART\_HardwareFlowControl = USART\_HardwareFlowControl\_None;

USART\_StructInit(&UART\_Struct);

// \*Habilitar el reloj del puerto serial 2, del puerto Gpio donde está ubicado y de la función alterna.

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_USART2, ENABLE);

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_AFIO | RCC\_APB2Periph\_GPIOA, ENABLE);

// \*Configuración del modo de función alterna de cada pin del puerto serial.

GPIO\_InitTypeDef GPIO\_Struct;

// TX: GPIOA PIN2 funcion alterna

GPIO\_Struct.GPIO\_Pin = GPIO\_Pin\_2;

GPIO\_Struct.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_Struct.GPIO\_Mode = GPIO\_Mode\_AF\_PP;

GPIO\_Init(GPIOA, &GPIO\_Struct);

//\*Habilitación del puerto serial. TX Y RX

USART\_Cmd(USART2, ENABLE);

USART\_Init(USART2, &UART\_Struct);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Inicialización del puerto Serial3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void UART3\_Init(void) // CONFIGURADO SOLO COMO RX

{

//\* Configuración de los parámetros del puerto serial. ESTA CAMBIA RESPECTO AL USO, VER EL DOCUMENTO.

// Inicializa la configuracion de los parámetros de Uart 9600 bps, 8 bits DE DATO, 1 stop, no paridad

USART\_InitTypeDef UART\_Struct;

UART\_Struct.USART\_BaudRate= 9600;

UART\_Struct.USART\_WordLength = USART\_WordLength\_8b;

UART\_Struct.USART\_StopBits = USART\_StopBits\_2;

UART\_Struct.USART\_Parity = USART\_Parity\_No ;

UART\_Struct.USART\_Mode = USART\_Mode\_Rx;

UART\_Struct.USART\_HardwareFlowControl = USART\_HardwareFlowControl\_None;

USART\_StructInit(&UART\_Struct);

// \*Habilitar el reloj del puerto serial, del puerto Gpio donde está ubicado y de la función alterna.

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_USART3, ENABLE);

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_AFIO | RCC\_APB2Periph\_GPIOB, ENABLE);

// \*Configuración del modo de función alterna de cada pin del puerto serial.

GPIO\_InitTypeDef GPIO\_Struct;

// RX: GPIOB PIN11 funcion alterna

GPIO\_Struct.GPIO\_Pin = GPIO\_Pin\_11;

GPIO\_Struct.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_Struct.GPIO\_Mode = GPIO\_Mode\_IN\_FLOATING;

GPIO\_Init(GPIOB, &GPIO\_Struct);

//\*Habilitación del puerto serial. TX Y RX

USART\_Cmd(USART3, ENABLE);

USART\_Init(USART3, &UART\_Struct);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Configuracion de la interrupcion por el puerto seria 1 uart.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void INTERRUPCION\_UART\_Init(void)

{

// CONFIGURA EL PUERTO QUE USAREMMOS PARA SER INTERRUPCION, EN ESTE CASO EL UART1 (GPIOA4)

USART\_ITConfig(USART1, USART\_IT\_RXNE, ENABLE);

// configurar el canal

NVIC\_InitTypeDef NVIC\_Struct;

NVIC\_Struct.NVIC\_IRQChannel = USART1\_IRQn;

NVIC\_Struct.NVIC\_IRQChannelPreemptionPriority = 0;

NVIC\_Struct.NVIC\_IRQChannelSubPriority = 0;

NVIC\_Struct.NVIC\_IRQChannelCmd = ENABLE;

// inicializar la interrupcion

NVIC\_Init(&NVIC\_Struct);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Funcion que corre dentro de la funcion de la inteerupcion.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Ejecucion\_Interrupcion(void)

{

char dato= (char) USART\_ReceiveData(USART1);

// si recibe '>', apaga el LED y cambia direccion en 0

if(dato == '>')

{

while(USART\_GetFlagStatus(USART1, USART\_FLAG\_TXE) == RESET); // REVISA EL ESTADO DE LA BANDERA

direcc = 0;

GPIO\_SetBits(GPIOC, GPIO\_Pin\_13);

}

// si recibe '<', enciende el LED y cambia direccion en 1

if(dato == '<')

{

while(USART\_GetFlagStatus(USART1, USART\_FLAG\_TXE) == RESET); // REVISA EL ESTADO DE LA BANDERA

direcc = 1;

GPIO\_ResetBits(GPIOC, GPIO\_Pin\_13);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Funcion de la interrupcion

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void USART1\_IRQHandler(void)

{

Ejecucion\_Interrupcion();

USART\_ClearITPendingBit(USART1, USART\_IT\_RXNE); // lo que esta arriba de el es lo que se ejecuta cuando entre la interrupcion.

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Configuracion del RTC

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void RTC\_Init(void)

{

// CONFIGURACION DEL RTC

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_PWR, ENABLE);// habilitacion del reloj

PWR\_BackupAccessCmd(ENABLE);// habilita acceso al rtc

RCC\_LSEConfig(RCC\_LSE\_ON);// habilita el reloj externo 32.768 HZ

while(RCC\_GetFlagStatus(RCC\_FLAG\_LSERDY) == RESET);// espera hasta que el reloj se estabilice

RCC\_RTCCLKConfig(RCC\_RTCCLKSource\_LSE);// selecciona el reloj externo como fuente

RCC\_RTCCLKCmd(ENABLE);// habilita el rtc

RTC\_WaitForSynchro();// espera por sincronizacion

while(RTC\_GetFlagStatus(RTC\_FLAG\_RTOFF)== RESET);

RTC\_SetPrescaler(32767); // ACTIVA EL PRESCALER

RTC\_SetCounter(0); // PONER EL CONTADOR EN CERO.

}