



SOLUCIÓN

EJERCICIO 1 (2.0 puntos):

Se tiene una arquitectura Von Neumann de 32 bits, con una capacidad de direccionar únicamente palabras, hasta un máximo de 64MB. Dicha arquitectura contiene 16 registros internos propósito general y filosofía Load & Store. Con esta información, conteste a las siguientes preguntas:

1) (70%) Diseñe una codificación de los distintos tipos de instrucciones microprocesador, minimizando en lo posible el tamaño de la instrucción ajustando a números enteros de palabras. Los tipos de instrucciones que debe tener el microprocesador, son:

- 5 instrucciones de transferencia de datos entre memoria y registros internos, con direccionamiento absoluto.
- 5 instrucciones de transferencia de datos entre memoria y registros internos, con direccionamiento indexado.
- 14 instrucciones aritmético/lógicas de operar entre registros, indicando en la instrucción tanto los registros operandos, como el registro que almacena el resultado.
- 14 instrucciones aritmético/lógicas de operar entre registros, con uno de los operandos limitado a 16 bits y dado por direccionamiento inmediato.
- 5 instrucciones de control con direccionamiento inherente
- 7 saltos condicionales con direccionamiento relativo a contador de programa, siendo el desplazamiento relativo de más/menos 1MB

Al ser una arquitectura Von Neumann el bus de direcciones y de datos de la memoria de programa es el mismo que el de datos. Las palabras son de 32bits y sólo se direccionan palabras, por lo que el bus de direcciones es de 24 líneas (64MB son 16Mpalabras). Además tenemos 16 registros de propósito general, por lo que los registros se referencian con 4 bits. Por último, en el caso del desplazamiento relativo de 1MB, al ser palabras de 32bits, el desplazamiento se dará con 18 bits, pero con la necesidad de incluir 1 bit más para dar el bit de signo (en realidad, complemento a 2).

Con todo esto, la codificación de instrucciones podría ser:

	31																24	23																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																			
--	----	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	----	----	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

2) (30%) Indique una respuesta justificada a cada una de las siguientes preguntas:



a) Número mínimo de palabras que usa una instrucción

Tal y como se puede ver del apartado anterior, la instrucción más pequeña se puede codificar en 1 byte, pero como la arquitectura es a 32 bits, la instrucción más pequeña ocupará una única palabra.

b) Número máximo de palabras que usa una instrucción

También se ve, que la instrucción más larga también ocupa 1 palabra

c) Tamaño del Registro de Instrucción

El registro de instrucción tendrá que ocupar lo que la instrucción más grande, y por tanto deberían ser 32 bits.

d) Tamaño del Contador de Programa

El contador de programa utilizará el mismo número de bits que el bus de direcciones, es decir, 24 bits.

e) Tamaño de los Registros internos

Los registros internos son de propósito general, por lo que se utilizan para datos (32 bits) y para direcciones (24 bits). Por tanto los registros internos tendrán el máximo de los dos, es decir, 32 bits.



EJERCICIO 2 (4.0 puntos):

En las próximas hojas aparece el código de un programa a cargar en un sistema basado en un STM32L152RB, **el cual puede algún error de programación**. El programa corresponde a la implementación de un velocímetro de bicicleta, capaz de mostrar por la pantalla la información sobre la distancia recorrida, la velocidad instantánea y la temperatura. De dicho sistema se conocen las siguientes características:

- Tiene dispositivos conectados cuya señal de entrada al STM32L152RB, es un valor cualquiera entre 0 y 3,3V.
- Puede tener algún otro dispositivo más conectado.

Se pretende que el alumno analice dicho código y, a partir de ahí conteste **razonadamente** a las siguientes preguntas (*algunas de las justificaciones se pueden realizar indicando las líneas de código donde se encuentra la evidencia*). Se supone que las funciones "LCD_Mostrar" y "MuestraVelocímetro" están programadas en la librería "Biblioteca_SDM.h":

1. ¿Qué elementos (periféricos) del STM32L152RB se están utilizando? (10%)

Los elementos que se están utilizando son los siguientes:

- GPIO (líneas 79-86)
- Conversor Analógico/Digital (líneas 87-97)
- Temporizadores (Timer 4) (líneas 103-118)
- Interrupciones externas 0 (EXTI0) (líneas 98-102)
- Vector de interrupciones NVIC (líneas 14, 26, 106 y 118)
- Reloj en tiempo real (RTC) (línea 120)

2. Teniendo en cuenta que tiene una configuración de reloj que hace que el pclk de todos los periféricos vayan a 12MHz, indique la configuración de cada uno de los periféricos del microcontrolador cuando se encuentren en funcionamiento. Absténgase de simplemente decir el valor de cada registro de configuración; lo que se pide es la funcionalidad que se obtiene. Es imprescindible detallar la escala temporal (caso de que exista) que utilizan los periféricos. (20%)

La configuración de los distintos periféricos es:

- GPIO: En la línea 79 se configura el PA4 como entrada analógica (para el sensor de temperatura). En las líneas 80-82 se configura el PA0 como entrada digital (para el botón de RESET). En las líneas 83-86 se configura el PB6 como función especial (para el TIM4).
- ADC: En la línea 88 se configura a 8 bits, y en la línea 89 se configura con retardo a una velocidad más lenta. En las líneas 93-94 se configura un único canal que sea el



AIN4. Por lo que se hacen medidas continuas de un valor analógico externo (suponemos que es la temperatura).

- **TIMER 4:** En la línea 106 se configura el pre-escalado, para que el temporizador cuente en unidades de paso de 1ms. Además se activa la funcionalidad TIC, mediante el flanco de bajada en el canal 1 y pin PB6. En un determinado momento se utiliza la funcionalidad TOC sin HW asociado (canal 2), para medir 2 segundos desde que se pulsa el reset (se activa la EINT1). Todo evento (tanto el TIC como el TOC) provoca interrupción, y en la RAI correspondiente se mira qué servicio ha dado la interrupción y se procede en consecuencia. El temporizador funciona siempre en modo continuo, no reseteándose nunca, por lo que todas las medidas se hacen de forma relativa (en relación al valor anterior o el actual del CNT).
- **EXTI:** Se utiliza el EXTI0 por PA0 y se configura para que salte por flanco de bajada inicialmente y luego se cambia la polaridad, pasando a flanco de subida, o flanco de bajada, según interese (el cambio se hace en la RAI correspondiente). La configuración inicial se hace en las líneas 98-101.
- **NVIC:** Se activan interrupciones por dos canales de entrada: la EXTI0 (línea 102), y el TIM4 (línea 118). En una de las RAI del se desactiva el NVIC del EXTI0 y luego se vuelve a activar.
- **RTC:** Se configura sólo la hora en la línea 120 a las 16:30 horas, utilizando para ello la función `config_RTC` programada en las líneas 45-60.

3. De un significado a la variable “valor 3” del programa. (10%):

“Valor 3” es un flag que se activa cuando se recibe una pulsación por EXTI0, y se mantiene durante un tiempo de 2 segundos mínimo. Provoca el poner a cero la distancia y la velocidad.

4. El código presenta un error muy sutil relacionado con la función “hora_to_str” (esto no tiene porqué significar que el error esté dentro del programa de la función). Encuéntrelo y dé una solución al mismo. (10%)

El error sutil está en la línea 120 donde se llama a la función “hora_to_str” con la variable “cadena” (que se ha declarado en la línea 9 como entero). Esta variable debería ser declarada de tipo `unsigned char*` (como se ve en la línea 62).

5. Explique brevemente lo que sucede en las líneas 47-48 y dibuja cómo van evolucionando los bits de la variable “valor 1” según se van ejecutando las instrucciones (10%)



En estas líneas se configura la fecha y hora del RTC y se van haciendo operaciones para ir colocando cada valor en su sitio dentro de cada registro.

D = Bits de las decenas de minuto
U = Bits de las unidades de minuto
H = Bits de las decenas de hora
G = Bits de las unidades de hora

47 $\text{valor1} = ((((\text{minuto} / 10) \ll 4) + (\text{minuto} \% 10)) \ll 8)$; Se pasa a la variable “valor1” los 4 bits de las decenas de minuto y los 4 bits de las unidades de minuto para que queden colocados en las posiciones necesarias para el registro RTC->TR. Paso a paso.

- $\text{minuto} / 10$ -> Nos quedamos con el cociente de dividir los minutos por 10
- $(\text{minuto} / 10) \ll 4$ -> Se desplaza lo anterior 4 posiciones a la izquierda y “valor 1” queda así: XXXXXXXXXXXXXXXXXXXXXXXDDDDXXX
- $\text{minuto} \% 10$ -> Nos quedamos con el resto de dividir los minutos por 10 y valor 1 queda así: XXXXXXXXXXXXXXXXXXXXXXXUUUU
- $(((\text{minuto} / 10) \ll 4) + (\text{minuto} \% 10))$ -> Sumo decenas y unidades y “valor 1” queda así: XXXXXXXXXXXXXXXXXXXXXXXDDDUUUU
- $= ((((\text{minuto} / 10) \ll 4) + (\text{minuto} \% 10)) \ll 8)$ -> Se desplaza lo anterior 4 posiciones a la izquierda y “valor 1” queda así: XXXXXXXXXXXXXXXXXXXDDDUUUUXXXXXXXX

48 $\text{valor1} |= ((((\text{hora} / 10) \ll 4) + (\text{hora} \% 10)) \ll 16)$; Se pasa a la variable “valor1” los 4 bits de las decenas de hora y los 4 bits de las unidades de hora, junto con los que ya había de los minutos, para que queden colocados en las posiciones necesarias para el registro RTC->TR. Paso a paso

- $\text{hora} / 10$ -> Nos quedamos con el cociente de dividir las horas por 10
- $(\text{hora} / 10) \ll 4$ -> Se desplaza lo anterior 4 posiciones a la izquierda y “valor 1” queda así: XXXXXXXXXXXXXXXXXXXDDDUUUUHHHHXXX
- $\text{hora} \% 10$ -> Nos quedamos con el resto de dividir los minutos por 10 y valor 1 queda así: XXXXXXXXXXXXXXXXXXXDDDUUUUHHHHGGGG
- $(((\text{hora} / 10) \ll 4) + (\text{hora} \% 10))$ -> Sumo decenas y unidades y “valor 1” queda así: XXXXXXXXXXXXXXXXXXXDDDUUUUHHHHGGGG
- $|= ((((\text{hora} / 10) \ll 4) + (\text{hora} \% 10)) \ll 16)$ -> Se desplaza lo anterior 16 posiciones a la izquierda, se hace un AND con lo anterior y “valor 1” queda así: XXXXXXXHHHHGGGGDDDUUUUXXXXXXXX

Y finalmente queda en la variable “valor1” los bits de las unidades y decenas de horas y minutos como debe ser para pasárselo al registro RTC->TR en la línea 55.



6. Explique brevemente lo que sucede en las líneas 64-72 y dibuja cómo van evolucionando los bits de la variable “valor 2” según se van ejecutando las instrucciones (10%)

En estas líneas se convierten los bits de las decenas y unidades de minutos y segundos del registro RTC-TR de valor entero a cadena de caracteres para luego seguir trabajando con este formato en el programa y, por ejemplo, sacarlo por el LCD.

62 void hora_to_str (unsigned char *cadena) {
Declaración de la función

63 int valor2 = 0;
Se inicializa “valor2” a cero

64 valor2 = RTC->TR;
Se pasa el valor del registro RTC a la variable valor 2

65 valor2 = valor >> 8;
Se desplaza todo 8 bits a la derecha para eliminar ceros y “valor 2” queda así:
XXXXXXXXXXXXXXXXHHHHGGGGDDDDUUUU

66 *(cadena+4)=(valor2 & 0x0000000F)+'0';
Me quedo sólo con los 4 bits UUUU haciendo un AND con F y le sumo el número ASCII del carácter cero para quedarme con el valor ASCII del valor numérico a convertir a formato carácter, colocándolo en la posición 4 de la cadena de caracteres.

67 valor2 = valor2 >> 4;
Desplazo 4 bits para hacer ahora lo mismo con DDDD.

68 *(cadena+3)=(valor2 & 0x00000007)+'0';
Lo mismo que el 66 pero con DDDD pero lo coloco en la posición 3 de la cadena de caracteres

69 valor2 = valor2 >> 4;
Desplazo 4 bits para hacer ahora lo mismo con GGGG.

70 *(cadena+1)=(valor2 & 0x0000000F)+'0';
Lo mismo que el 66 pero con GGGG pero lo coloco en la posición 1 de la cadena de caracteres

71 valor2 = valor2 >> 4;
Desplazo 4 bits para hacer ahora lo mismo con HHHH.

72 *cadena=(valor2 & 0x00000003)+'0';



Lo mismo que el 66 pero con HHHH pero lo coloco en la posición 0 de la cadena de caracteres

Finalmente la cadena de caracteres "Cadena" queda con los siguientes valores:

Posición 0 -> HHHH

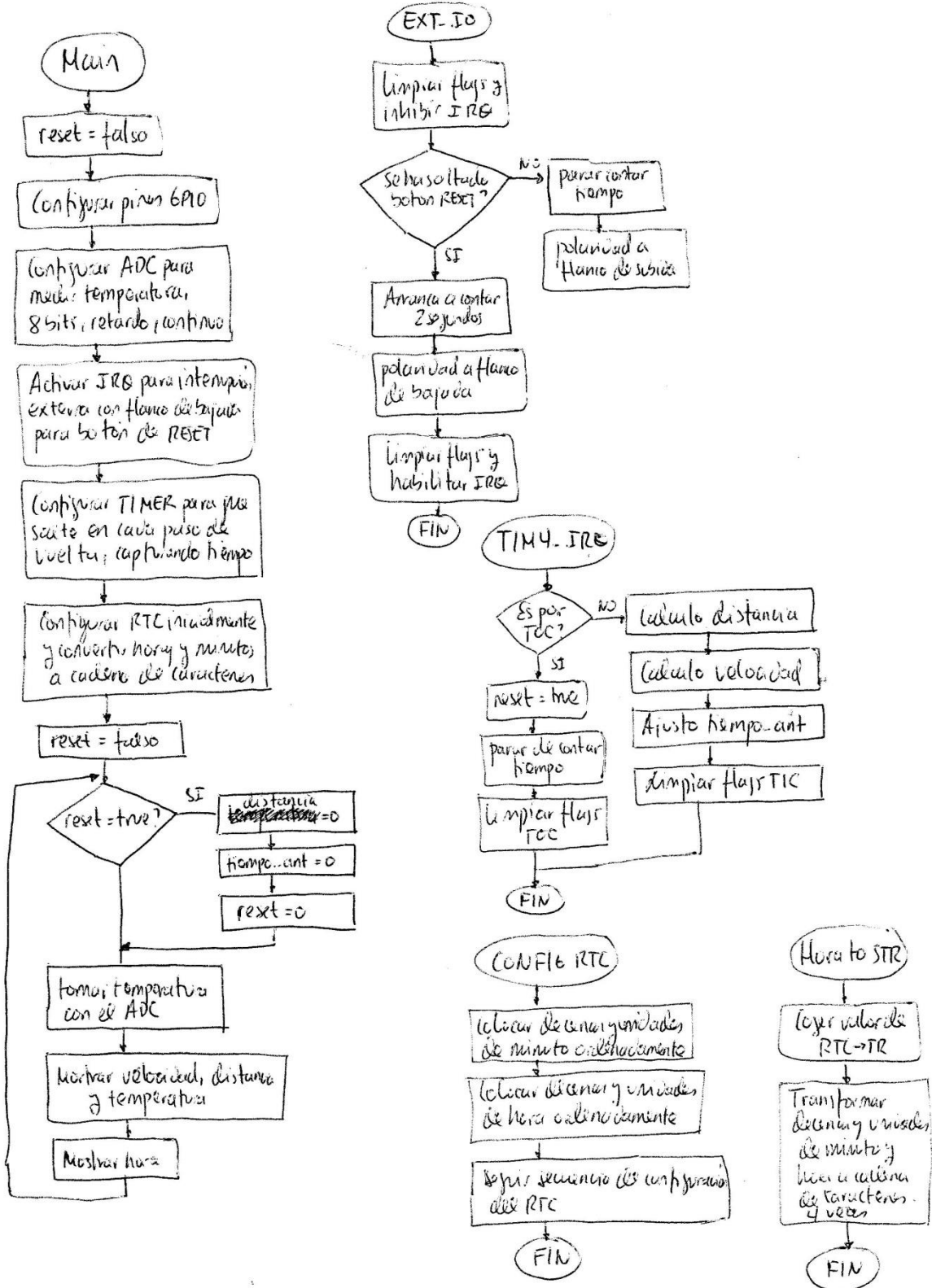
Posición 1 -> GGGG

Posición 2 -> Indefinido (por ejemplo para poner ":" fijamente después)

Posición 3 -> DDDD

Posición 4 -> UUUU

7. Realice el diagrama de flujo de todas y cada una de las funciones utilizadas, así como del programa principal. No haga siempre una transposición directa del código en un diagrama de flujo, sino represente la funcionalidad obtenida, mediante dicho diagrama de flujo. (20%).





8. ¿Qué configuraría adicionalmente en los registros de configuración de la USART del sistema para poder enviar cada 30 minutos por el PB6 y PB7 el valor medio de la velocidad a una memoria USB conectada al velocímetro y entre qué líneas las colocaría del programa actual (9600,8,N,1 y sin interrupciones). ¡¡¡No el programa de transmisión!!! (10%)

El código de la configuración de los registros sería el siguiente:

Habría que cambiar alternativamente en el código del programa main() la configuración del PB6 cuando se quisiese transmitir/recibir porque ya estaba configurado al principio para el TIMER 4, con los siguientes comandos:

```
// Configuración del USART

// PB7 y PB6 como AF - USART
GPIOB->MODER |= (0x01 << (2*7+1)); // (10 la posición de PB6 en MODER)
GPIOB->MODER &= ~(0x01 << (2*7));
GPIOB->MODER |= (0x01 << (2*6+1));
GPIOB->MODER &= ~(0x01 << (2*6)); // (10 la posición de PB7 en MODER)
GPIOB->AFR[0] = 0x77000000; // AF7 = 0111 -> La función AF es la función USART
// 0111 en la posición del PB6 y del PB7 en AFR[0])
```

El resto de comandos necesarios para transmitir con los requisitos del problema (9600,8,N,1 y sin interrupciones) podría colocar en las posiciones 118 y 120, por ejemplo.

```
// Comunicación 9600,8,N,1 con PCLK = 32MHz
USART1->CR1 = 0x00000008; // OVER8 = 0 -> Oversampling = 16 -> DIV_fraction[3:0] en el USARTx->BRR usa 4 bits
// UE = 0 -> Deshabilitación de la USART
// M = 0 -> Tamaño de palabra: 8 bits
// PCE = 0 -> Control de paridad deshabilitado
// TXEIE, TCIE, RXNEIE = 0 -> Sin IRQs en la comunicación
// TE = 1 -> Habilitación de la transmisión
// RE = 0 -> Deshabilitación de la recepción
USART1->CR2 = 0x00000000; // STOP = 00 -> Números de bit de parada = 1
USART1->BRR = 0x00000D05; // 9600 baudios -> USARTDIV = 208,3125 (mirar tabla de teoría para 32Mhz)
// Mantisa = 208 = 0xD0;
// Fracción = 0,31*16 = 5 = 0X05;

// Habilitación del USART
USART1->CR1 |= 0x01 << 13; // UE = 1 en el bit 13 del CR1
```



ANEXO

```
1  #include "stm32l1xx.h"
2  #include "Biblioteca_SDM.h"
3  unsigned char radio=23;
4  unsigned tiempo_ant, tiempo;
5  unsigned char valor3;
6  unsigned distancia;
7  unsigned temperatura;
8  unsigned velocidad, diámetro;
9  int hora, minuto, cadena;
10 int diámetro = 2 * 3142 * radio;
11
12 void EXTI0_IRQHandler (void) {
13     EXTI->PR = 0x01;
14     NVIC->ICER[0] |= (1 << 6);
15     if ((EXTI->RTSR & 0x01) == 0) {
16         TIM4->CCR2 = TIM4->CNT + 2000;
17         TIM4->CR1 |= 0x0001;
18         EXTI->RTSR |= 0x01;
19         EXTI->FTSR &= ~(0x01);
20     }
21     else {
22         TIM4->CR1 &= ~(0x0001);
23         EXTI->FTSR |= 0x01;
24         EXTI->RTSR &= ~(0x01);
25     }
26     NVIC->ISER[0] |= (1 << 6);
27 }
28
29 void TIM4_IRQHandler (void) {
30     if (TIM4->SR & 0x0004 == 1) {
31         valor3 = 1;
32         TIM4->CR1 &= ~(0x0001);
33         TIM4->SR = 0x0004;
34     }
35     else {
36         distancia += diámetro;
37         tiempo = TIM4->CCR1 - tiempo_ant;
38         if (tiempo < 0) tiempo += 0xFFFFFFFF;
39         velocidad = distancia / tiempo;
40         tiempo_ant = TIM4->CCR1;
41         TIM4->SR = 0x0002;
42     }
43 }
44
```



```
45 void config_RTC(int hora, int minuto) {
46     int valor1 = 0;
47     valor1 = (((minuto / 10) << 4) + (minuto % 10)) << 8;
48     valor1 |= (((hora / 10) << 4) + (hora % 10)) << 16;
49     RTC->WPR=0xCA;
50     RTC->WPR=0x53;
51     RTC->ISR |= (1<<7);
52     while ((RTC->ISR & (1<<6))==0);
53     RTC->PRER=255;
54     RTC->PRER|=127<<16;
55     RTC->TR = valor1;
56     RTC->CR = 0x00000000;
57     RTC->ISR &= ~(1<<7);
58     RTC->WPR=0;
59     while ((RTC->ISR & (1<<6))!=0);
60 }
61
62 void hora_to_str (unsigned char *cadena) {
63     int valor2 = 0;
64     valor2 = RTC->TR;
65     valor2 = valor2>> 8;
66     *(cadena+4)=(valor2 & 0x0000000F)+'0';
67     valor2 = valor2 >> 4;
68     *(cadena+3)=(valor2 & 0x00000007)+'0';
69     valor2 = valor2 >> 4;
70     *(cadena+1)=(valor2 & 0x0000000F)+'0';
71     valor2 = valor2 >> 4;
72     *cadena=(valor2 & 0x00000003)+'0';
73 }
74
75 int main (void) {
76     Init_SDM();
77     Init_LCD();
78     valor3 = 0;
79     GPIOA->MODER |= 0x00000300;
80     GPIOA->MODER &= ~(1 << (0*2 +1));
81     GPIOA->MODER &= ~(1 << (0*2));
82     GPIOA->PUPDR &= ~(11 << (0*2));
83     GPIOB->MODER |= 0x00000001 << (2*6 +1);
84     GPIOB->MODER &= ~(0x00000001 << (2*6));
85     GPIOB->AFR[0] &= ~(0x0F << (4*6));
86     GPIOB->AFR[0] |= 0x02 << (4*6);
87     ADC1->CR2 &= ~(0x00000001);
88     ADC1->CR1 = 0x02000000;
89     ADC1->CR2 = 0x00000472;
90     ADC1->SMPR1 = 0;
```



```
91  ADC1->SMPR2 = 0;
92  ADC1->SMPR3 = 0;
93  ADC1->SQR1 = 0x00000000;
94  ADC1->SQR5 = 0x00000004;
95  ADC1->CR2 |= 0x00000001;
96  while ((ADC1->SR&0x0040)==0);
97  ADC1->CR2 |= 0x40000000;
98  EXTI->FTSR |= 0x01;
99  EXTI->RTSR &= ~(0x01);
100 SYSCFG->EXTICR[0] = 0;
101 EXTI->IMR |= 0x01;
102 NVIC->ISER[0] |= (1 << 6);
103 TIM4->CR1 = 0x0000;
104 TIM4->CR2 = 0x0000;
105 TIM4->SMCR = 0x0000;
106 TIM4->PSC = 12000;
107 TIM4->CNT = 0;
108 TIM4->ARR = 0xFFFF;
109 TIM4->CCR2 = 0;
110 TIM4->DCR = 0;
111 TIM4->DIER = 0x0006;
112 TIM4->CCMR1 = 0x0001;
113 TIM4->CCMR2 = 0x0000;
114 TIM4->CCER = 0x0003;
115 TIM4->CR1 |= 0x0001;
116 TIM4->EGR |= 0x0001;
117 TIM4->SR = 0;
118 NVIC->ISER[0] |= (1 << 30);
119
120 config_RTC (16,30);
121 hora_to_str (cadena);
120 valor3 = 0;
121 while (1) {
122     if (valor3!=0) {
123         tiempo_ant = 0;
124         distancia = 0;
125         valor3 = 0;
126     }
127     while ((ADC1->SR & 0x0002)==0);
128     temperatura = (unsigned char) (ADC1->DR & 0x000000FF);
129     MuestraVelocimetro (velocidad, distancias, temperatura);
130     LCD_Texto (cadena);
131 }
132 }
```



EJERCICIO 3 (4.0 puntos):

Con un microcontrolador STM32L152RB alimentado con 3V y pclk a 32MHz, se quiere diseñar un sistema de control de la depuradora de una piscina. Es importante tener en cuenta que la piscina es de 50m³, y que el flujo de agua a través de la depuradora es de 10m³/h. Por simplicidad del ejercicio se va a suponer que las condiciones del agua son homogéneas en todo su volumen (es decir, sus características son idénticas en la propia piscina, que en el caudal que está pasando por la depuradora). La depuradora de la piscina consta de las siguientes características:

- Control de encendido y apagado de la depuradora, con posibilidad de programar las horas de funcionamiento y la hora de encendido, a través de una parámetros que se le pueden dar al microcontrolador.
- Posibilidad de encendido y apagado manual de la depuradora a través de un interruptor que proporciona Vcc para encenderla y GND para apagarla.
- Control de los niveles de cloro:
 - Se cuenta con un sensor que indica el nivel de cloro en el agua mediante una tensión entre 0 y Vcc. El nivel óptimo de cloro vendrá dado por el parámetro OBJETIVO_CLORO, y se considerará normal siempre que se encuentre en un margen del 10% arriba o abajo.
 - Se cuenta con un clorador salino que se activa mediante un tren de pulsos de periodo 1ms, el cual, cuanto mayor es el tamaño del pulso, más fuerza de cloración ejerce. El ancho del pulso nunca puede ser inferior 0,1ms ni mayor de 0,9ms. Se supone que si el pulso es de 0,5ms, se mantiene el nivel de cloro actual.
 - Si el cloro se desvía en más de un 50%, entonces se activará la alarma.
- Control del pH:
 - Se cuenta con un sensor de pH que da, mediante una tensión entre 1V y 2V, el valor de pH en decimales, entre 1,0pH y 12,0pH. El nivel óptimo de pH vendrá dado por el parámetro OBJETIVO_PH, y se considerará normal siempre que esté dentro de las 2 décimas anteriores o posteriores al nivel de OBJETIVO_PH.
 - Como el pH tiende siempre a subir, se dispone de un reductor de pH que se activa mediante una señal periódica de 1KHz. Esta señal se activará cuando haya que poner en marcha el reductor, manteniéndola a GND cuando no haya que actuar. Cuando se active, se podrá hacer a velocidad lenta (donde la señal periódica tendrá un ciclo de trabajo del 30%) o a velocidad alta (donde la señal periódica tendrá un ciclo de trabajo del 75%).
 - Si el pH baja de 5,5 o sube de 8,5, se activará la alarma.
- Control del nivel del agua:
 - Se detectará el nivel del agua a través de 3 sensores digitales (S1, S2 y S3) colocados en la pared de la piscina, en vertical, estando el S1 el más profundo y S3 el más cercano a la superficie. Cada uno de esos sensores, si está cubierto de agua (es decir, el nivel llega a la altura del sensor) da un 0 a su salida, mientras que si no está cubierto de agua, dará un 1.
 - Se podrán activar 2 llaves de paso de agua, mediante un par de salidas digitales. Si la salida digital es un 1, la llave se abre, dejando pasar el agua. Si es un 0, la llave se cierra, cortando el agua.
 - Si los tres sensores están cubiertos, las llaves deben estar cerradas. Si sólo S1 y S2 están cubiertos, se debe abrir una de las llaves de paso. Si tanto S2 como S3 se encuentran sin nivel de agua, entonces se deben abrir las dos llaves de paso. Si los tres sensores se encuentran sin agua, entonces, se lanzará una alarma.



- Comunicación vía serie (a 19200, 8, N, 1) con una consola central, la cual le podrá enviar mensajes de configuración (de cada uno de los valores objetivos, de la hora actual, y de la hora de inicio y duración de la activación de la depuradora), y a la que el microcontrolador le podrá mandar el mensaje de alarma, compuesto por la hora, y el valor de todos y cada uno de los sensores. Ese mensaje se repetirá cada minuto, en el caso de que la situación de alarma no se corrija.
 - Tendrá Vd. que definir las tramas de comunicaciones.
- En la situación de alarma, deberá hacer sonar un pitido de 800Hz, intermitente cada segundo, mediante una salida digital entre 0 y Vcc. Este tono debe mantenerse hasta que se salga de la situación de alarma.
 - Al mismo tiempo, y tal como se ha comentado anteriormente, deberá enviar cada minuto un mensaje a la consola central.

Con esta información, y aquellas aproximaciones o decisiones que Vd. considere necesario (siempre justificándolas) conteste a las siguientes preguntas:

- a) Realice el diagrama de bloques de la solución (20%)

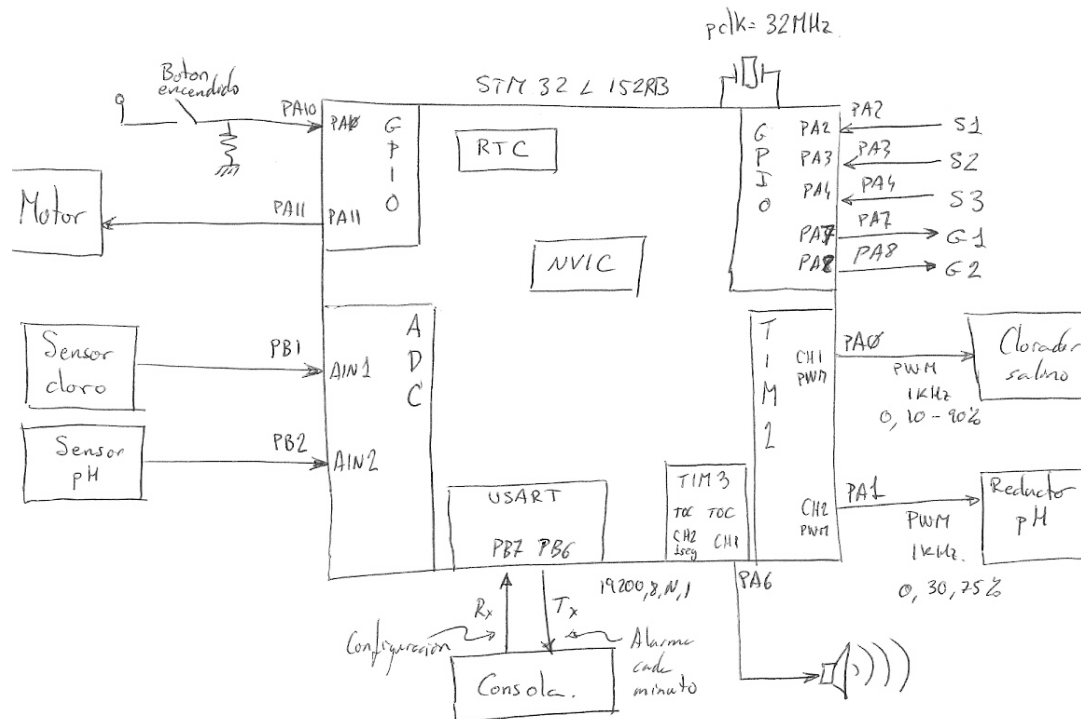
A la hora de plantear una solución hay que analizar determinadas condiciones del problema. Lo primero es que se trata de manejar la depuración de una piscina que, atendiendo a la velocidad de depuración y al volumen de la piscina, se trata de hacer una gestión de un mínimo de 5 horas. Esto, en otras palabras, indica que todo el proceso es enormemente lento para lo que es una velocidad razonable de un microcontrolador. Por lo tanto, casi toda la gestión se podrá hacer por esperas activas, en lugar de por interrupciones, salvo, por ejemplo, la recepción serie, que puede requerir interrupción para no perder un canal. Por lo tanto, atendiendo a cada uno de las partes comentadas, tenemos las siguientes decisiones:

- *Arranque del motor de la depuradora de forma manual y de forma temporizada.*
 - *El arranque será una señal digital de salida que se pondrá a 1 para que se ponga en marcha y a 0 para pararla. (GPIO)*
 - *El arranque manual se dará por una señal digital de entrada (GPIO)*
 - *El arranque automático se realizará internamente atendiendo a la hora que se mantenga con el RTC.*
- *Control de Cloro:*
 - *El sensor entrará a un canal del conversor ADC. Se pueden hacer muestras individuales, puesto que la variación será mucho más lenta que lo que tarda en hacerse la conversión.*
 - *El clorador se controlará con una señal PWM de 1KHz con DC variable entre 10% y 90%*
- *Control de pH:*



- El sensor entrará a otro canal del conversor ADC. Se pueden hacer muestras individuales, puesto que la variación será mucho más lenta que lo que tarda en hacerse la conversión.
- El reductor de pH se controlará con una señal PWM de 1KHz con DC de 0%, 30% y 75%.
- Control de agua:
 - Se sensorá el nivel con 3 entradas GPIO procedentes de S1, S2 y S3. No se necesita mirar el flanco, sino el nivel.
 - Los grifos (G1 y G2) se activarán mediante 2 salidas digitales del GPIO
- Alarma:
 - Se generará con un TOC con salida hardware que generará el tono de 800Hz. La salida hardware se activará o desactivará atendiendo a si el tono tiene que sonar o no. Será mucho más cómodo utilizar una IRQ para este temporizador y esta funcionalidad, que actualice el CCR en cada semiciclo.
 - Se utilizará otro TOC para que cada segundo desactive o active la salida hardware del tono cuando la alarma está activada.
- Conexión:
 - Se usará la USART para Tx y Rx atendiendo a los requisitos de 19200, 8, N, 1 y las tramas definidas en el apartado c).

Con todo esto queda el diagrama de bloques siguiente:





- b) Indique si utilizaría interrupciones, y en caso afirmativo, para qué y con qué funcionalidad. (10%)

Tal y como se ha ido comentando, no es necesario utilizar interrupciones salvo para la Rx de la USART (la Tx no va a tomar tanto tiempo como para impactar el funcionamiento del programa, y por tanto no se va a considerar por interrupciones), y la gestión cómoda del tono de 800Hz intermitente (es decir, IRQ en la actualización del toggle del tono, y en la activación y desactivación del tono cada segundo).

- c) Diseñe las tramas de comunicaciones. (10%)

Para la comunicación entre el control de la depuradora y la consola, la única trama necesaria sería la formada por la siguiente secuencia de bytes (1 byte por parámetro):

- Año
- Mes
- Día
- Hora
- Minuto
- Nivel de cloro
- Nivel de pH
- Estado del agua (0=ningún sensor cubierto, es decir, alarma; 3 = los tres sensores cubiertos)
- Y un byte de control de errores (un LRC)

Para la comunicación entre la consola y la depuradora, los mensajes serían:

- 'C' X, siendo X el valor de OBJETIVO_CLORO
- 'P' X, siendo X el valor de OBJETIVO_PH
- 'F' AA MM DD HH MM, para darle la fecha y hora actual a la depuradora
- 'I' HH MM DURH DURM, para darle la hora de arranque y la duración del funcionamiento

- d) Diga como configuraría los periféricos del microcontrolador utilizados, detallando el valor de los bits relevantes de los registros de control. (25%)

GPIO: PA2-PA4 como entrada digital; PA7-PA8 como salida digital; PA10 como entrada digital; PA11 como salida digital.

- *GPIOA->MODER &= ~(3 << (2*2));*



- $GPIOA \rightarrow MODER \&= \sim(3 \ll (3*2));$
- $GPIOA \rightarrow MODER \&= \sim(3 \ll (4*2));$
- $GPIOA \rightarrow MODER \&= \sim(3 \ll (10*2));$
- $GPIOB \rightarrow MODER \&= \sim(1 \ll (7*2 + 1));$
- $GPIOB \rightarrow MODER |= (1 \ll (7*2));$
- $GPIOB \rightarrow MODER \&= \sim(1 \ll (8*2 + 1));$
- $GPIOB \rightarrow MODER |= (1 \ll (8*2));$
- $GPIOB \rightarrow MODER \&= \sim(1 \ll (11*2 + 1));$
- $GPIOB \rightarrow MODER |= (1 \ll (11*2));$

ADC: PB1 y PB2 como AIN1 y AIN2. Ambos a 8 bits, sin retardo y con conversión simple.

- $GPIOB \rightarrow MODER |= \sim(3 \ll (1*2));$
- $GPIOB \rightarrow MODER |= \sim(3 \ll (2*2));$
- $ADC1 \rightarrow CR2 \&= \sim(0x00000001); // ADON=0$
- $ADC1 \rightarrow CR1 = 0x00000000;$
- $ADC1 \rightarrow CR2 = 0x00000400;$
- $ADC1 \rightarrow SMPR1 = 0;$
- $ADC1 \rightarrow SMPR2 = 0;$
- $ADC1 \rightarrow SMPR3 = 0;$
- $ADC1 \rightarrow SQR1 = 0x00000000; // 1 elemento solo$
- $ADC1 \rightarrow SQR5 = 0x00000001; // Canal AIN1 (se cambiará a AIN2 cuando toque pH)$
- $ADC1 \rightarrow CR2 |= 0x00000001; // ADON=1$
- Se deja listo para arrancar cuando se necesite

USART1: PB7 como Rx y PB6 como transmisión. A 19200, 8, N, 1 y con interrupciones por recepción.

- $GPIOB \rightarrow MODER |= (0x01 \ll (2*7+1));$
- $GPIOB \rightarrow MODER \&= \sim(0x01 \ll (2*7));$
- $GPIOB \rightarrow MODER |= (0x01 \ll (2*6+1));$
- $GPIOB \rightarrow MODER \&= \sim(0x01 \ll (2*6));$
- $GPIOB \rightarrow AFR[0] = 0x77000000;$
- $USART1 \rightarrow CR1 = 0x0000000C;$
- $USART1 \rightarrow CR2 = 0x00000000;$
- $USART1 \rightarrow BRR = 0x00000683; (104,1875 \text{ por tabla con } pclk \text{ 32MHz y } over8 = 0)$
- $USART1 \rightarrow CR1 |= 0x01 \ll 13;$
- $USART1 \rightarrow CR1 |= 0x01 \ll 5;$
- $NVIC \rightarrow ISER[1] |= (1 \ll (37-32));$



TIM3: Como TOC interno el canal 2 y como TOC con salida externa el canal 1 (conectado al altavoz) y función toggle. Ambos por interrupción inicialmente desactivada que se activará cuando salte la alarma.

- *GPIOA->MODER &= ~(1 << (6*2));*
- *GPIOA->MODER |= (1 << (6*2 +1));*
- *GPIOA->AFR |= (2 << (6*4));*
- *// Selección de reloj interno*
- *TIM3->CR1 = 0x0000;*
- *TIM3->CR2 = 0x0000;*
- *TIM3->SMCR = 0x0000;*
- *TIM3->PSC = 3199; // saltos en décimas de milisegundos*
- *// Valores en ARR y CCRx*
- *TIM3->CNT = 0;*
- *TIM3->ARR = 0xFFFF;*
- *TIM3->CCR1 = 6; // cada 0,6ms*
- *TIM3->CCR2 = 10000; // cada 1 segundo*
- *// Modo de salida*
- *TIM3->CCMR1 = 0x0030;*
- *TIM3->CCMR2 = 0x0000;*
- *TIM3->CCER = 0x0011;*
- *// Habilitación de contador*
- *// TIM3->CR1 |= 0x0001; (cuando sea necesario)*
- *TIM3->EGR |= 0x0001;*
- *// IRQ*
- *TIM3->DCR = 0;*
- *//TIM3->DIER = 0x0006; // irq activada en CH1 y CH2 (cuando sea necesario)*
- *TIM3->SR = 0; // Limpia flags*
- *// NVIC->ISER[0] |= (1 << 30); // se activa cuando sea necesario*

TIM2: Como PWM de 1ms de periodo, con el canal 1 variando a 0%, y ente 10 y 90%, y el canal 2 variando entre 0, 30 y 75%.

- *GPIOA->MODER|=0x00000001<<(2*0 +1);*
- *GPIOA->MODER&=~(0x00000001<<(2*0));*
- *GPIOA->AFR[0]|=(0x01<<(0*4));*
- *GPIOA->MODER|=0x00000001<<(2*1 +1);*
- *GPIOA->MODER&=~(0x00000001<<(2*1));*
- *GPIOA->AFR[0]|=(0x01<<(1*4));*
- *TIM2->CR1 = 0x0000;*



- `TIM2->CR2 = 0x0000;`
- `TIM2->SMCR = 0x0000;`
- `TIM2->PSC = 320;` // centésimas de milisegundo
- // Valores en ARR y CCRx
- `TIM2->CNT = 0;`
- `TIM2->ARR = 99;`
- `TIM2->CCR1 = 0;` // esto se cambiará al % que toque
- `TIM2->CCR2 = 0;` // esto se cambiará al % que toque
- // Sin IRQ
- `TIM2->DCR = 0;`
- `TIM2->DIER = 0x0000;`
- // Modo de salida
- `TIM2->CCMR1 = 0x6868;`
- `TIM2->CCMR2 = 0x0000;`
- `TIM2->CCER = 0x0011;`
- // Habilitación de contador
- `TIM2->EGR |= 0x0001;`
- `TIM2->CR1 |= 0x0001;`
- `TIM2->SR = 0;`

RTC: activado a una hora estándar y que se pueda modificar por comando recibido en la recepción serie.

- `RTC->WPR=0xCA;`
- `RTC->WPR=0x53;`
- `RTC->ISR |= (1<<7);`
- `while ((RTC->ISR & (1<<6))==0);`
- `RTC->PRER=255;`
- `RTC->PRER|=127<<16;`
- `RTC->TR = 0x00123456;`
- `RTC->DR = 0x00120314;`
- `RTC->CR = 0x00000000;`
- `RTC->ISR &= ~(1<<7);`
- `RTC->WPR=0;`

NVIC: Con activación por USART y, aunque inicialmente desactivado, por TIM3.

- *Configurado previamente en los apartados superiores.*



e) Realice el diagrama de flujo de la solución (35%)

