Problema 1 (60 minutos – 4 puntos)

Se ha desarrollado un sistema de cerradura electrónica para la puerta de acceso a uno de los laboratorios de la universidad. Para esta aplicación se ha utilizado el microcontrolador de referencia del curso funcionando con una frecuencia de oscilación de 32MHz y alimentado entre 0 y 3V.

Como interfaz de usuario el sistema dispone de:

- Un dispositivo que se comunica con el microcontrolador mediante una comunicación serie asíncrona (DISPOSITIVO_SERIE). Este dispositivo está formado por una pantalla en la que pueden visualizarse los mensajes enviados desde el microcontrolador y un teclado con el que pueden introducirse los números de la clave de acceso. Esta clave de acceso consta de 4 dígitos.
- Un botón de cancelación (BOTON_CANCEL) que permite cancelar la introducción del código de acceso en cualquier momento. Si se pulsa este botón, es necesario volver a introducir la clave completa de nuevo.

Además, la puerta de acceso se encuentra conectada mediante un dispositivo intermedio (DISPOSITIVO_APERTURA) a un pin de salida digital del microcontrolador. Cuando este pin de salida tiene un "1" lógico la puerta se abre (con un "0" lógico en este pin de salida, la puerta se cierra).

El microcontrolador también tiene conectado a un pin de salida digital un dispositivo de seguridad de bloqueo de la puerta (DISPOSITIVO_BLOQUEO). Cuando el pin del microcontrolador al que se encuentra conectado el dispositivo de bloqueo se pone a un "1" lógico el dispositivo actúa y la puerta se bloquea. El bloqueo de la puerta se produce después de 3 intentos erróneos de introducir la clave de acceso. **El desbloqueo de la puerta se hace externamente al sistema que se presenta**, volviendo a poner en este pin de salida un "0" lógico.

Por último, se dispone de un ALTAVOZ conectado al microcontrolador que emite una alarma sonora para avisar que la puerta se encuentra bloqueada.

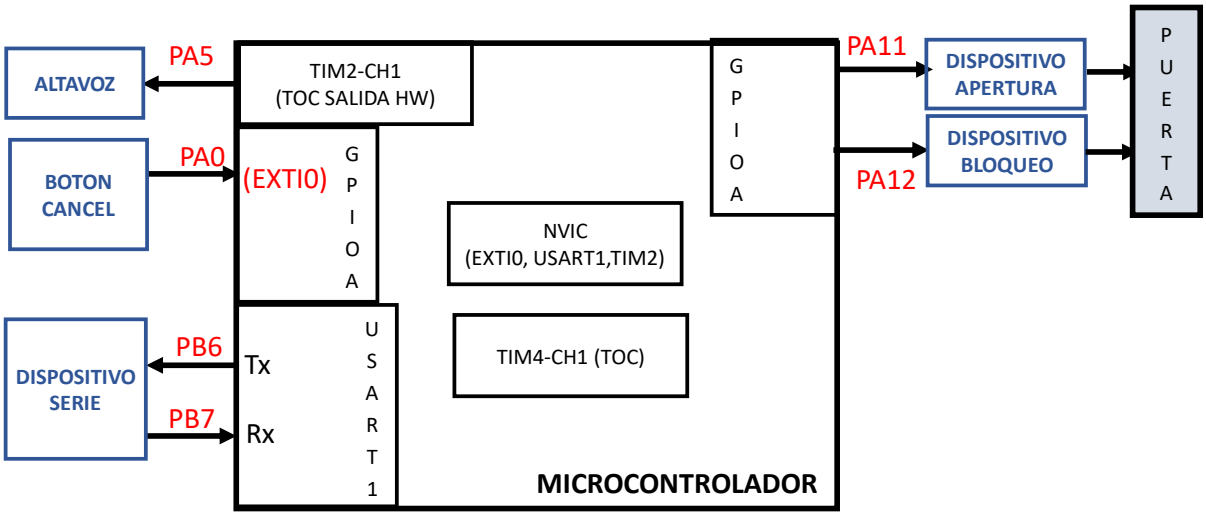
El esquema del sistema completo es el que se muestra en la figura 1:

Figura 1. Esquema del sistema de acceso con cerradura electrónica

El código desarrollado para la aplicación es el que se muestra al final de este enunciado (se recomienda ver y analizar el código antes de empezar a contestar a las preguntas que se plantean):

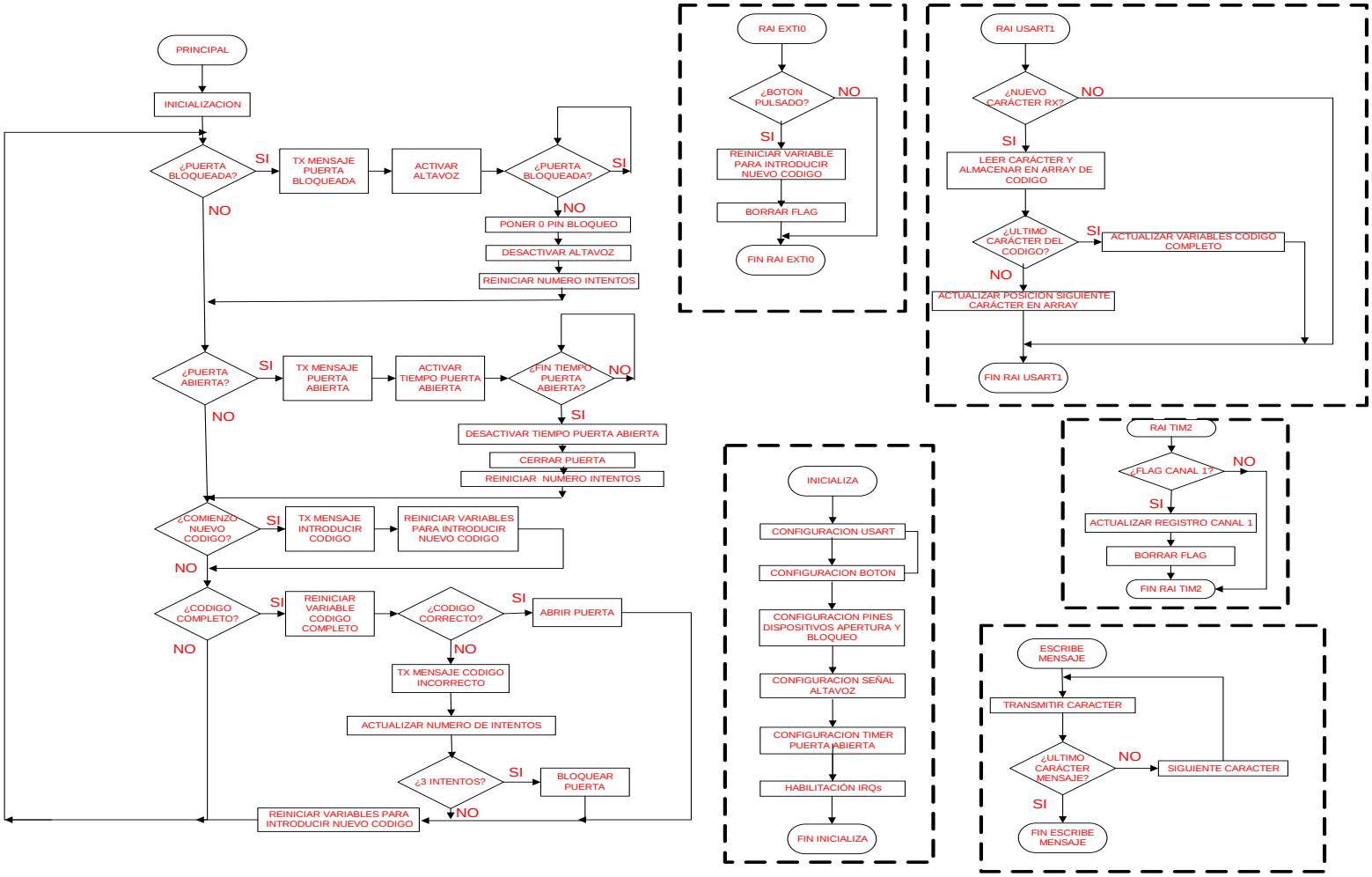
Se pide:

1. Represente el esquema del hardware (diagrama de bloques) para esta aplicación. Dibuje el microcontrolador como un bloque, indicando claramente todos los recursos del microcontrolador utilizados y qué pines del microcontrolador se conectan a cada una de las entradas o salidas de todos los elementos que componen el sistema (cada elemento debe estar claramente identificado en el diagrama con el mismo nombre con que se denota en la figura 1). Además, rellene la siguiente tabla. **(25%)**



PIN MICRO	HW CONECTADO	TIPO (E/S, DIGITAL/ ANALOGICO)	JUSTIFICACIÓN
PB6, PB7	DISPOSITIVO SERIE	TX/RX USART1	LÍNEAS 37-44 DEL CÓDIGO
PA0	BOTON CANCEL	ENTRADA DIGITAL (EXTI0)	LINEAS 46-51 DEL CODIGO
PA5	ALTAVOZ	SALIDA HW CH1-TIM2 MODO TOC	LINEAS 61-76 DEL CODIGO
PA11	DISPOSITIVO APERTURA	SALIDA DIGITAL	LINEAS 57-59 DEL CODIGO
PA12	DISPOSITIVO BLOQUEO	SALIDA DIGITAL	LINEAS 53-55 DEL CODIGO
ADEMÁS SE UTILIZA: - CH1-TIM4 EN MODO TOC SIN SALIDA HARDWARE (LINEAS 78-88 DEL CODIGO) - NVIC PARA EXTI0 E IRQ DE USART1 Y TIM2 (LINEAS 91-93 DEL CODIGO)			

2. Represente el diagrama de flujo del programa principal, función/es y rutina/s de atención a interrupción del código. **(30%)**



3. ¿Con qué recurso del microcontrolador se controla el tiempo que la puerta de acceso permanece abierta, y cuánto tiempo permanece la puerta abierta después de introducir el código de entrada correcto? Justifique su respuesta incluyendo todos los cálculos necesarios para obtenerla. **(15%)**

El tiempo que la puerta permanece abierta se controla con el canal 1 del TIM4 en modo TOC sin salida hardware. Este temporizador se configura en las líneas 78-88 del código. En esta configuración:

$$T_u = paso = T_{CLK} \cdot PSC = \frac{1}{32MHz} \cdot 32000 = 1ms$$

Como CCR1=10000 pasos, el temporizador (cuya cuenta se reinicia en el código cada vez que se abre puerta) cuenta 10s ⇒ Una vez introducido el código correcto la puerta permanece abierta 10s.

4. Indique la configuración completa de la comunicación serie utilizada para la introducción del código de acceso. Indique la/s modificación/es necesarias en el código para que los parámetros de esta comunicación serie pasen a ser 57600, 8, N, 1. **(15%)**

La conexión serie se configura en las líneas 41-43.

USART1->CR1 = 0x0000002C ⇒ Habilitación Tx, Rx; Rx por IRQ. Modo sobremuestro por 16 (OVER8=0), 8 bits de datos, sin bit de paridad
USART1->CR2 = 0x00000000 ⇒ 1 bit de parada
USART1->BRR = 0x00000115 ⇒ Div_Mantissa=0x11=17, Div_fraction=0x5=5 ⇒USARTDIV=17,5

$$BAUDRATE(bps) = \frac{f_{CLK}}{8x(2-OVER8)xUSARTDIV} = \frac{32MHz}{16x17,5} \cong 114,3kbps\,(NORMALIZADO: 115,2kbps)$$

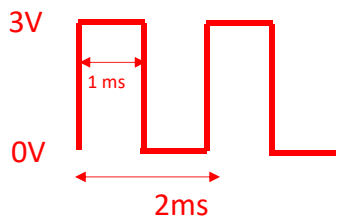
Por lo tanto, los parámetros de la comunicación serie configurada en el código son: 115200, 8, N, 1.

Para cambiar a la configuración solicitada (57600, 8, N, 1) manteniendo OVER8=0, solo habría que cambiar el registro BRR.

57600⇒USARTDIV=34,7⇒Div_Mantissa=34=0x22, Div_fraction=11=0xB
⇒ USART1->BRR = 0x0000022B

5. Represente dos períodos completos de la forma de onda de la señal que se aplica al altavoz (acotándola claramente en amplitud y tiempo). Indique la/s modificación/es necesarias en el código que esta señal sea una señal entre 0 y 3V con una frecuencia de 1Hz. **(15%).**

La señal que se aplica al altavoz se genera utilizando el canal 1 del TIM2 en modo TOC con salida HW toogle (configuración en líneas de código 61-76). En esta configuración: Tu=paso=1µs, Semiperíodo (registro CCR)=1000 pasos=1ms. Es una señal, por tanto de 500Hz.



Para cambiar la frecuencia a 1Hz, el semiperiodo debe ser 500ms, hay que cambiar:

- Preescala (con pasos de 1µs necesitaríamos 500000 pasos que no caben en un registro de 16bits). Cambiando por ejemplo la preescala a un valor para contar el mismo nº de pasos que en el código original en cada semiperiodo (1000 pasos) solo sería necesario modificar este registro: TIM2->PSC= 16000;

CÓDIGO PROGRAMA CERRADURA ELECTRÓNICA

```
#include "stm32l1xx.h"

void EscribeMensaje(unsigned char* mensaje, unsigned char longitud_mensaje);

unsigned char nueva_entrada_codigo=1;
unsigned char posicion_caracter_recibido=0;
unsigned char codigo_finalizado=0;
unsigned char codigo_apertura[4]={'1','2','3','4'};
unsigned char codigo_introducido[4]={0,0,0,0};

void EXTI0_IRQHandler(void) {
  if (EXTI->PR!=0){
        nueva_entrada_codigo=1;
        EXTI->PR =0x00001;
  }
}
void TIM2_IRQHandler(void) {
  if (TIM2->SR&0x0002!=0){
        TIM2->CCR1 += 1000;
        TIM2->SR = 0x0000;
  }
}

void USART1_IRQHandler(void) {
  if ((USART1->SR&0x0020)!=0){
        codigo_introducido[posicion_caracter_recibido]=USART1->DR;
        if (posicion_caracter_recibido==3){
              codigo_finalizado=1;
              posicion_caracter_recibido=0;
        }
        else  posicion_caracter_recibido++;
  }
}

void Inicializa(void){
  GPIOB->MODER &= ~(0x0000F000);
  GPIOB->MODER |= (0x0000A000);
  GPIOB->AFR[0] &= ~(0xFF000000);
  GPIOB->AFR[0] |= 0x77000000;
  USART1->CR1 = 0x0000002C;
  USART1->CR2 = 0x00000000;
  USART1->BRR = 0x00000115;
  USART1->CR1 |= 0x2000;

  GPIOA->MODER &= ~(1 << (0*2 +1));
  GPIOA->MODER &= ~(1 << (0*2));
  SYSCFG->EXTICR[0]&= ~(0x000F);
  EXTI->RTSR |= (0x00000001);
  EXTI->FTSR &= ~ (0x00000001);
  EXTI->IMR |= 0x00000001;

  GPIOA->MODER &= ~(1 << (12*2 +1));
  GPIOA->MODER |= (1 << (12*2));
  GPIOA->BSRR= (1<<12)<<16;

  GPIOA->MODER &= ~(1 << (11*2 +1));
  GPIOA->MODER |= (1 << (11*2));
  GPIOA->BSRR= (1<<11)<<16;

  GPIOA->MODER |=(1 << (5*2+1));
  GPIOA->MODER &= ~(1 << (5*2));
  GPIOA->AFR[0]&= ~(0x00F00000);
  GPIOA->AFR[0]|=(0x00100000);
  TIM2->CR1 = 0x0000;
  TIM2->CR2 = 0x0000;
  TIM2->SMCR = 0x0000;
  TIM2->PSC = 32;
  TIM2->CNT = 0;
  TIM2->ARR = 0xFFFF;
  TIM2->CCR1= 1000;
  TIM2->CCMR1 = 0x0030;
  TIM2->CCER = 0x0001;
```

```
74      TIM2->EGR |= 0x0001;
75      TIM2->SR = 0;
76      TIM2->DIER=0x0002;
77
78      TIM4->CR1 = 0x0000;
79      TIM4->CR2 = 0x0000;
80      TIM4->SMCR = 0x0000;
81      TIM4->PSC = 32000;
82      TIM4->ARR = 0xFFFF;
83      TIM4->CCR1= 10000;
84      TIM4->CCMR1 = 0x0000;
85      TIM4->CCER = 0x0000;
86      TIM4->EGR |= 0x0001;
87      TIM4->SR = 0;
88      TIM4->DIER=0x0000;
89
90      EXTI->PR=0x0001;
91      NVIC->ISER[0] |= (1 << 6);
92      NVIC->ISER[0] |= (1 << 28);
93      NVIC->ISER[1] |= (1 << (37-32));
94
95    }
96
97    void EscribeMensaje(unsigned char* mensaje, unsigned char longitud_mensaje){
98            unsigned char i=0;
99            for (i=0;i<longitud_mensaje;i++){
100                   while ((USART1->SR & 0x0080)== 0);
101                   USART1->DR = mensaje[i];
102           }
103   }
104
105   int main(void){
106   unsigned char intentos=0;
107   unsigned char mensaje_puerta_bloqueada[]="\nPUERTA BLOQUEADA\n";
108   unsigned char mensaje_puerta_abierta[]="\nPUERTA ABIERTA\n";
109   unsigned char mensaje_introducir_codigo[]="\nINTRODUZCA CODIGO\n";
110   unsigned char mensaje_codigo_incorrecto[]="\nCODIGO INCORRECTO\n";
111
112     Inicializa();
113     while (1) {
114           if ((GPIOA->IDR&0x1000)!=0){
115                   EscribeMensaje(mensaje_puerta_bloqueada, sizeof(mensaje_puerta_bloqueada));
116                   TIM2->CR1 |= 0x0001;
117                   while ((GPIOA->IDR&0x1000)!=0); ***//ESPERA DESBLOQUEO EXTERNO PUERTA***
118                   GPIOA->BSRR=(1<<12)<<16;
119                   TIM2->CR1 &= ~(0x0001);
120                   intentos=0;
121           }
122           if ((GPIOA->IDR&0x0800)!=0){
123                   EscribeMensaje(mensaje_puerta_abierta,sizeof(mensaje_puerta_abierta));
124                   TIM4->CNT=0;
125                   TIM4->CR1 |= 0x0001;
126                   while ((TIM4->SR&0x0002)==0);
127                   TIM4->SR = 0;
128                   TIM4->CR1 &= ~(0x0001);
129                   GPIOA->BSRR=(1<<11)<<16;
130                   intentos=0;
131           }
132            if(nueva_entrada_codigo==1){
133                   EscribeMensaje(mensaje_introducir_codigo,sizeof(mensaje_introducir_codigo));
134                   nueva_entrada_codigo=0;
135                   posicion_caracter_recibido=0;
136           }
137           if (codigo_finalizado==1){
138                   codigo_finalizado=0;
139                   if(codigo_introducido[0]==codigo_apertura[0]&codigo_introducido[1]==codigo_apertura[1]&codigo_introducido[2]==codigo_apertura[2]&
140                   codigo_introducido[3]==codigo_apertura[3])  GPIOA->BSRR= (1<<11);
141                   else{
142                           EscribeMensaje(mensaje_codigo_incorrecto,sizeof(mensaje_codigo_incorrecto));
143                           intentos++;
144                           if (intentos==3) GPIOA->BSRR= (1<<12);
145                   }
146                   nueva_entrada_codigo=1;
147           }
148     }
149   }
```

Problema 2 (90 minutos – 6 puntos)

Basándose en un microprocesador STM32L152RB con pclk de 32MHz (no conectado a la placa de desarrollo utilizada en el curso), se tiene que diseñar un sistema para controlar la temperatura y la humedad relativa del interior del habitáculo de un coche de gama media que cumpla con los siguientes **requisitos:**

- El microcontrolador debe tener conectado 2 sensores de temperatura que suministran una tensión entre 0 y 3,3V para indicar la temperatura, la cual variará, como más rápido, cada 5 segundos, siendo el valor máximo de temperatura de 60ºC y el valor mínimo de -10ºC. Los 2 sensores sirven para tomar la medida de temperatura en la parte delantera y trasera del habitáculo del vehículo, de manera que el programa debe calcular la media de los 2 sensores para tomar esa temperatura como temperatura real del habitáculo y a partir de ahí hacer lo siguiente (**se supone que el programa de control va a tener de alguna manera el valor de consigna deseado por el usuario**, no hay que preocuparse como lo consigue):
    - Si la temperatura media se aleja menos de 0,1 grado de la deseada por el conductor, el microcontrolador desconecta el sistema auxiliar que regula la temperatura del habitáculo.
    - Si la diferencia entre la temperatura media y la temperatura deseada es superior a 0,1 grados (tanto positiva, como negativa), el microcontrolador activa el sistema auxiliar que regula la temperatura del habitáculo para conseguir la temperatura deseada (enfriando o calentando en función de cada caso), el cual es controlado por una señal PWM de 100Hz proveniente del microcontrolador, de manera que el DC empieza al 10% y va subiendo de 10% en 10% hasta el 90% con cada 0,1º que se aleje de la temperatura deseada, es decir, si se aleja 0,1º el DC es 10%, si se aleja 0,2º, el DC es 20% y así hasta un máximo del 90% en el DC.
- El microcontrolador debe tener además conectado un sensor de humedad relativa en el centro del habitáculo del vehículo que cada minuto emite una señal eléctrica entre 3,3 y 0V, siendo el valor máximo una humedad relativa del 100ºC y el valor mínimo de 0ºC y hacer lo siguiente:
    - Si la humedad relativa está entre el 40% y el 70%, el microcontrolador desconecta el sistema auxiliar de compensación de humedad relativa.
    - Si la humedad relativa está por debajo del 40% o por encima del 70%, el sistema genera una señal digital que activa el sistema auxiliar de compensación de humedad relativa (‘0’ – sistema de compensación apagado; ‘1’ – sistema de compensación encendido).
- El sistema, además, recibe una señal de entrada digital externa que le indica, mediante flancos, que la plataforma donde está instalo el sistema se encuentra operativa (es decir, una especie de “señal de vida”). Dicha señal debería tener siempre menos de 10 segundos entre flancos para indicar que el sistema está operativo. Es necesario medir el tiempo que transcurre entra los flancos de esta señal, de forma que:
    - Si es menor de 10 segundos no se hace nada, pero si es mayor de 10 segundos, el sistema enciende un LED para indicar que algo raro ocurre y deshabilita el sistema de control de temperatura y humedad. Además, en ese caso, desactivará todas las señales de salida.
- El usuario dispone de un pulsador para reinicializar la funcionalidad descrita anteriormente en cualquier momento, desactivando además todas las señales de salida del sistema. (Si pulsador es ‘1’ = Sistema en funcionamiento normal; si es ‘0’ = Reinicializa la funcionalidad) y de un interruptor para encenderlo o apagarlo (‘0’ = Apagado / ‘1’ = Encendido).

Necesidades especiales de diseño:

- Por necesidades de optimización, todo se debe realizar con el puerto A del micro excepto la señal para el LED que debe ser del puerto B.
- Por necesidades de diseño, los 4 canales de cada temporizador se deben utilizar para los mismos modos del mismo, no se pueden mezclar modos en los canales. Y además no se puede utilizar el TIM1.
- La señal de vida se debe evaluar obligatoriamente utilizando alguno de los modos de funcionamiento de los temporizadores que no sea el modo TOC, ni tampoco se puede hacer con señales digitales normales y espera activa en programa.

Con estas especificaciones, optimizando los recursos y de forma razonada:

a) Dibuje el diagrama de bloques del sistema. (20%)

b) Indique si utilizará interrupciones, y en caso afirmativo, cuáles y para qué. (20%)

c) Indique qué periféricos utiliza y cómo los configuraría. No escriba el código, sino el valor que tendría que tener cada uno de los bits afectados de cada uno de los registros del periférico y por qué. (40%)

d) Dibuje el diagrama de flujo del programa de control y las posibles interrupciones. (20%)

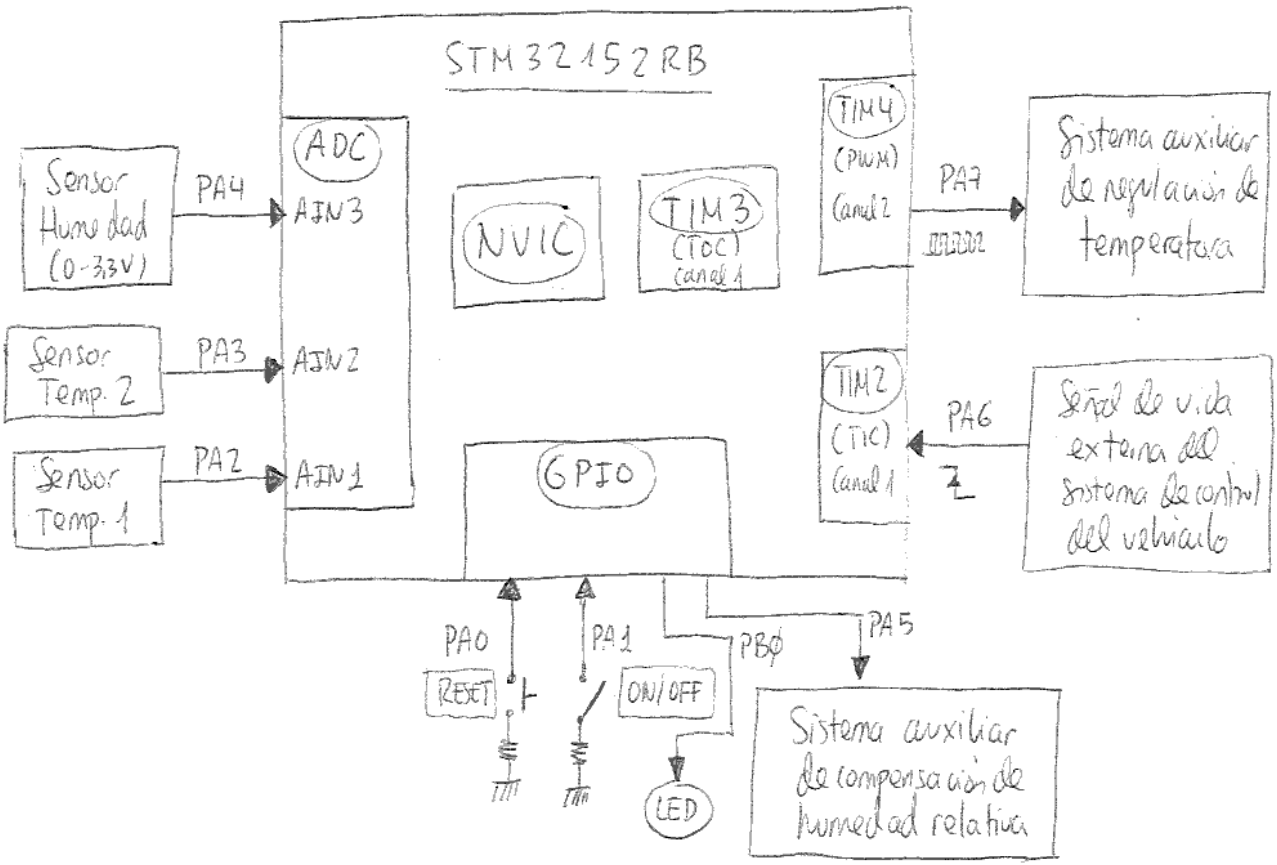
SOLUCIÓN:

a) Diagrama de bloques del sistema. (20%)

Un diagrama de bloques posible sería el siguiente. Consideraciones:

- Las señales de RESET y ON/OFF son simplemente señales de entrada digital del bloque GPIO
- Los 3 sensores analógicos necesarios para el sistema se conectan a 3 canales del bloque ADC
- El bloque NVIC se añade en diagrama de bloques porque se supone que alguna hará falta en el apartado b).
- Hay que contar segundos, generar una señal PWM y contar la señal de vida en modo TIC. Como se dice que no se pueden mezclar modos en los canales de cada temporizador, es necesario usar 3 diferentes. El canal 1 del TIM2 para contar los 10 segundos de la señal de vida en modo TIC, el canal 1 del TIM3 se utiliza para contar 5 segundos y un minuto en modo TOC y el canal 2 del TIM4 se utiliza para generar la señal PWM necesaria para el sistema auxiliar de control de temperatura, pero se pueden usar otros canales.

Con estas consideraciones y teniendo en cuenta que se pueden variar algunos de los pines del microcontrolador, no tienen por qué ser exactamente los propuestos, un posible diagrama de bloques sería el siguiente.



b) Indique si utilizará interrupciones, y en caso afirmativo, cuáles y para qué. (20%)

Los periféricos utilizados permiten la utilización de interrupciones, pero:

- La consulta de la señal de vida es cada 10 segundos y habría que usar el modo TIC del TIM 2 para hacerlo, que preferentemente exige el uso de interrupciones en el canal 1 del TIM 2.

- Los 5 segundos y el minuto de los sensores analógicos no son críticos en el tiempo pero son múltiplos entre sí por lo que con un sólo canal del TIM 3 sería suficiente para medir esos tiempos usando interrupciones. La interrupción debería saltar cada 5 segundos para trabajar con el sensor de temperatura y cada 12 saltos también trabajar con sensor de humedad relativa al detectar el minuto transcurrido.
- Para la señal PWM no sería necesario utilizar interrupciones en el TIM 4 porque la frecuencia es fija de 100Hz y el DC se puede cambiar en el programa principal en función de la señal media de las medidas de los sensores analógicos.

- El pulsador de RESET del PA0 hay que programarlo obligatoriamente por interrupción externa EXTI porque nunca se sabe cuándo podría aparecer y además se dice que debe actuar inmediatamente.
- Para el botón ON/OFF del PA1 no se dice nada de inmediatez y se puede elegir hacerlo por interrupciones o no. En la solución propuesta se hace por una entrada digital normal sin interrupciones.

c) Indique qué periféricos utiliza y cómo los configuraría. No escriba el código, sino el valor que tendría que tener cada uno de los bits afectados de cada uno de los registros del periférico y por qué. (40%)

Los distintos periféricos con su configuración son los siguientes:

- **GPIO:** Los pines PA0 (para el botón de RESET) y PA1 (para el botón de ON/OFF) como entradas digitales, y PA5 (para la salida digital del sistema auxiliar de compensación de humedad relativa) como salida digital. Los pines PA2 y PA3 como entradas analógicas de los canales 1 y 2 (para conectar los dos sensores de humedad de 0-3,3V) y el pin PA4 como entrada analógica del canal 3 (para conectar el sensor de humedad relativa de 0-3,3V). El pin PA6 se configura como función especial para la señal TIC asociada al TIM2.El pin PA7 se configura como función especial para la señal PWM asociada al TIM4.

```
GPIOA->MODER &= ~(1 << (0*2 +1));  // PA0 como entrada digital (00)
GPIOA->MODER &= ~(1 << (0*2 +1));

GPIOA->MODER &= ~(1 << (1*2 +1));  // PA1 como entrada digital (00)
GPIOA->MODER &= ~(1 << (1*2 +1));
```

```
GPIOA->MODER |= 0x000003F0;      // PA2, 3 y 4 como señales analógicas (11)

GPIOA->MODER &= ~(1 << (5*2 +1));          // PA5 como salida digital (01))

GPIOA->MODER |= (1 << (5*2));


GPIOA->MODER|=0x00000001 << (2*6 +1);          // PA6 como función especial (10)
GPIOA->MODER&=~(0x00000001 << (2*6));
GPIOA->AFR[0]|=(0x02 << (6*4));                // AFR[0] = 0x00010000000000 para que // el
                                               PA6 tenga la función especial 0001 = // TIM2

GPIOA->MODER|=0x00000001 << (2*7 +1);          // PA7 como función especial (10)
GPIOA->MODER&=~(0x00000001 << (2*7));
GPIOA->AFR[0]|=(0x02 << (7*4));                // AFR[0] = 0x02000000000000 para que // el
                                               PA7 tenga la función especial 0010 = // TIM4


GPIOB->MODER &= ~(1 << (0*2 +1));              // PB0 como salida digital
GPIOB->MODER |= (1 << (0*2));
```

ADC: Por sencillez se configuran para 12 bits, modo continuo, modo scan y sin interrupciones

```
ADC1->CR2 &= ~(0x00000001); // ADON = 0 (ADC apagado)

ADC1->CR1 = 0x00000100;      // Configuración del registro CR1
                             // OVRIE = 0 (sin la habilitación por interrupción)
                             // RES = 00 (resolución = 12 bits)
                                    // SCAN = 1 (modo scan para leer los 3 canales
                                    // de manera secuencial)
                                    // EOCIE = 0 (deshabilitada la interupción por EOC)
                             // OVRIE = 0 (deshabilitada la habilitación por
                                    // interrupción
ADC1->CR2 = 0x00000412;      // EOCS = 1 (activado bit EOC al acabar conversión)
                             // DELS = 000 (sin retardo de la conversión)
                             // CONT = 1 (conversión continua)
ADC1->SMPR1 = 0;                    // Sin sampling time (4 cycles)

ADC1->SMPR2 = 0;
```

```
ADC1->SMPR3 = 0;

ADC1->SQR1 = 0x00020000;  // 3 elementos  en la secuencia (0010 en los bits 23-20)
ADC1->SQR5 = 0x00000C41;        // Los elementos son los canales AIN1, AIN2 y AIN3 por // este
                                orden (0-00011-00010-00001)

ADC1->CR2 |= 0x00000001;        // ADON = 1 (ADC activado)
```

- **TIMER 2**: En modo TIC para contar lo 10 segundos de la señal de vida con el PA6 asociado a esa señal en su canal 1.. PCS = 32000 para contar 1000 pasos por segundo. En la INT se comprueban los 10 segundos de la señal de vida

```
TIM2->CR1 = 0x0000;         // ARPE = 0 -> No es PWM, es TIC; CEN = 0; Contador
                            // apagado
TIM2->CR2 = 0x0000;         // CCyIE = 0 -> No se provoca interrupción con el TIMER2
TIM2->SMCR = 0x0000;        // Siempre "0" en este curso

TIM2->PSC = 32000;          // Preescalado=32000 -> F_contador=32000000/32000 = 1000
                            // pasos/seg
TIM2->CNT = 0;        // Inicializo el valor del contador a cero
TIM2->ARR = 0xFFFF;         // Valor recomendado si no es PWM

TIM2->DIER = 0x0002;        // Se genera INT al terminar de contar -> CCyIE = 1

TIM2->CCMR1 = 0x0001;       // CCyS = 1 (TIC); OCyM = 000 y OCyPE = 0 (siempre en
                            // TIC)

TIM2->CCER = 0x0001;        // CCyNP:CCyP = 00 (activo a flanco de subida)
                      // CCyE = 1 (captura habilitada para TIC)

TIM2->EGR |= 0x0001;        // UG = 1 -> Se genera evento de actualización
TIM2->SR = 0;         // Limpio los flags del contador
```

```
TIM2->CR1 |= 0x0001;          // CEN = 1 -> Arranco el contador
```

- **TIMER 3:** En modo TOC para contar 5 segundos y un minuto con interrupciones y sin HW asociado. PCS = 32000 para contar 1000 pasos por segundo y CCR1 = 5000, para que la interrupción salte cada 5 segundos. Luego en la INT se comprueban los 5 segundos y el minuto

```
TIM3->CR1 = 0x0000;           // ARPE = 0 -> No es PWM, es TOC
                              // CEN = 0; Contador apagado
TIM3->CR2 = 0x0000;       // CCyIE = 1 -> No se provoca interrupción con el TIMER3
TIM3->SMCR = 0x0000;    // Siempre "0" en este curso

TIM3->PSC = 32000;            // Preescalado = 32000 -> Frecuencia del contador =
                              // 32000000/32000 = 1000 pasos por segundo
TIM3->CNT = 0;                // Inicializo el valor del contador a cero
TIM3->ARR = 0xFFFF;           // Valor recomendado = FFFF
TIM3->CCR1 = 5000;        // Registro donde se guarda el valor que marca la
                              // comparación existosa en TOC.
                              // Inicializo al valor que quiero llegar: 5000 pasos = 5 sg

TIM3->DIER = 0x0002;  // Se genera INT al terminar de contar -> CCyIE = 1
```

- **TIMER 4:** En modo PWM para generar por el PA7 una señal PWM de 100Hz y un DC de 5-90% sin ninguna interrupción asociada. Se configura inicialmente un DC del 10%

```
TIM4->CR1 = 0x0080;           // ARPE = 1 -> Es PWM
                              // CEN = 0; Contador apagado
TIM4->CR2 = 0x0000;           // CCyIE = 0 -> No se provoca interrupción con el
                              // TIMER4
TIM4->SMCR = 0x0000;    // Siempre "0" en este curso
TIM4->PSC = 32000;            // Preescalado = 32000 -> Frecuencia del contador =
        // 32000000/32000 = 1000 pasos por segundo
TIM4->CNT = 0;                // Inicializo el valor del contador a cero
TIM4->ARR = 9;                // Pongo una frecuencia PWM de 100 Hz, sólo cuento 10
        // pasos (de 0 a 9) y empiezo de nuevo
```

```
TIM4->CCR2 = 1;              // El Duty cycle se pone a 1 paso = 10% de 100Hz

TIM4->DIER = 0x0000;         // No se genera INT al terminar de contar -> CCyIE = 0

TIM4->CCMR1 = 0x6800; // CCyS = 0 (TOC, PWM)
                      // OCyM = 110 (PWM con el primer semiciclo a 1)
                      // OCyPE = 1 (con precarga)

TIM4->CCER = 0x0010;  // CCyP = 1 (siempre en PWM)
                      // CCyE = 0 (captura deshabilitada)

TIM4->CR1 |= 0x0001;         // CEN = 1 -> Arranco el contador
TIM4->EGR |= 0x0001;         // UG = 1 -> Se genera evento de actualización
TIM4->SR = 0;                // Limpio los flags del contador
```

- **EXTI para la EXTI0** asociada al PA0 para el botón de RESET

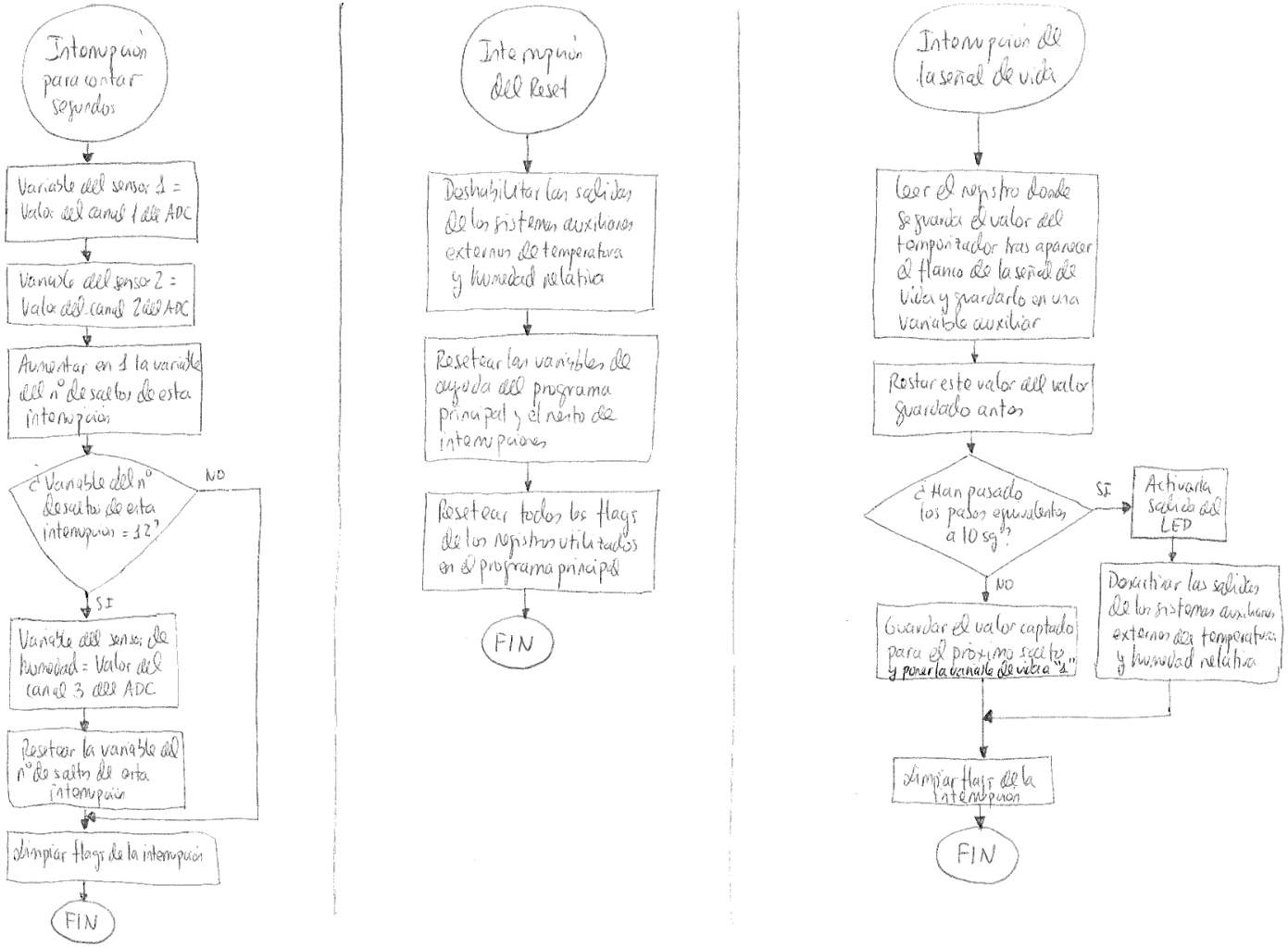
```
EXTI->FTSR |= 0x01;         // '1' habilita el evento por flanco de bajada en EXTI0
EXTI->RTSR &= ~(0x01);      // '0' inhabilita el evento por flanco de subida en EXTI0
SYSCFG->EXTICR[0] = 0;      // La EXTI0 la provoca el bit 0 del GPIOA (= PA0)
EXTI->IMR |= 0x01;          // Un '1' habilita la EXTI0, no la enmascara
```

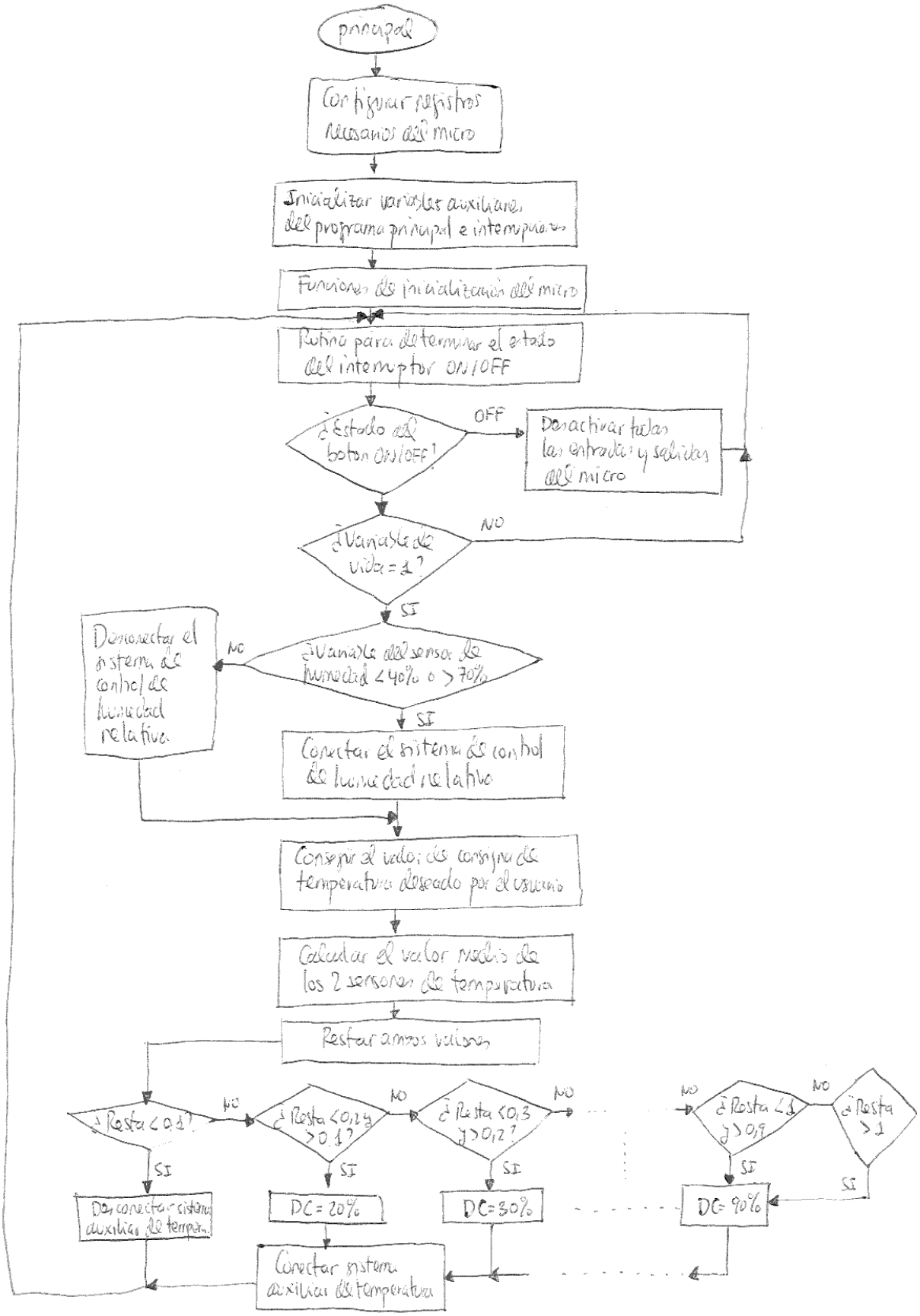
- **NVIC:** Se activan interrupciones para la EXTI0, el TIM2 y el TIM 4 (0,25 puntos).

```
NVIC->ISER[0] |= (1 << 6);   // Habilito la EXTI0 en el NVIC para la entrada de reset.
NVIC->ISER[0] |= (1 << 28); // Habilitación de la INT TIM2_IRQ en NVIC (pos. 28) para
                            // la señal de vida .
NVIC->ISER[0] |= (1 << 29); // Habilitación de la INT TIM3_IRQ en NVIC (pos. 29) para
                            // la señal PWM del sistema auxiliar de regulación de
                            // temperatura.
```

d) Dibuje el diagrama de flujo del programa de control y las posibles interrupciones. (20%)

Esta sería una posible solución según lo supuesto en los apartados anteriores.




principal
Configurar registros necesarios del micro
Inicializar variables auxiliares del programa principal e interrupciones
Funciones de inicialización del micro
Rutina para determinar el estado del interruptor ON/OFF
¿Estado del boton ON/OFF?
OFF
Desactivar todas las entradas y salidas del micro
¿Variable de vida = 1?
NO
SI
Desconectar el sistema de control de humedad relativa
NO
¿Variable del sensor de humedad < 40% o > 70%
SI
Conectar el sistema de control de humedad relativo
Consigna el valor de consigna de temperatura deseado por el usuario
Calcular el valor medio de los 2 sensores de temperatura
Restar ambos valores
¿Resta < 0,1?
NO
¿Resta < 0,2 y > 0,1?
NO
¿Resta < 0,3 y > 0,2?
NO
NO
¿Resta < 1 y > 0,9
NO
¿Resta > 1
SI
SI
SI
SI
SI
Desconectar sistema auxiliar de temper.
DC = 20%
DC = 30%
DC = 90%
Conectar sistema auxiliar de temperatura

1