Sistemas Basados en Microprocesador

Integración y desarrollo de una aplicación:
Reproductor MP3

Alumno:
Javier Garcia Rios.
……

2021-22

Departamento de Ingeniería Telemática y Electrónica
Universidad Politécnica de Madrid

Índice del documento

1 OBJETIVOS DE LA PRÁCTICA

1.1 Acrónimos utilizados

Identifique los acrónimos usados en su documento.

UART	Universal Asynchronous Receiver Transmitter
MP3	Moving Picture experts group audio layer 3
I2C	Inter-Integrated Circuit Inerface
CMD	Comando/Command
SPI	Serial Peripheral Interface
RTOS	Real Time Operating System
PC	Personal Computer
CMSIS	Common Microcontroller Software Interface Standard
LCD	Liquid Crystal Display
TID	Thread Identification
ADC	Analog to digital converter

1.2 Tiempo empleado en la realización de la práctica.

1ª ENTREGA: Aproximadamente 3h para hacer la primera entrega. Tuve algún problema a la hora de codificar el LCD, el cual solvente satisfactoriamente.
2ª ENTREGA: Aproximadamente 7h. módulos individuales funcionan perfectamente, solo que al incluirlos en el sistema completo, me han dado algún problema a lo largo de la practica, lo que ha hecho que esta cantidad de horas se incrementase más.
3ªENTREGA: Aproximadamente 6h. Encontré dificultades en entender como configurar bien la conexión I2C por lo que me llevo a tardar mas tiempo. Finalmente conseguí con la documentación necesaria, solventar este problema.
En total para esta aplicación abre dedicado aproximadamente alrededor de 72h aproximadamente en unir toda la aplicación y hacerla funcionar. Contando con que al principio plantee incorrectamente el sistema completo, pero al cambiarlo todo funcionaba ya como yo quería.

1.3 Bibliografía utilizada

[RD1] CMSIS-RTOS2, página web, https://www.keil.com/pack/doc/CMSIS/RTOS2/html/index.html

[RD2] StackOverflow, pagina web, https://meta.stackoverflow.com

[RD3] Youtube, página web, Videos varios

1.4 Autoevaluación.

Utilizando los códigos de las competencias como de los resultados académicos más importantes en mi opinión expuestos en la guía docente de la asignatura, expreso:

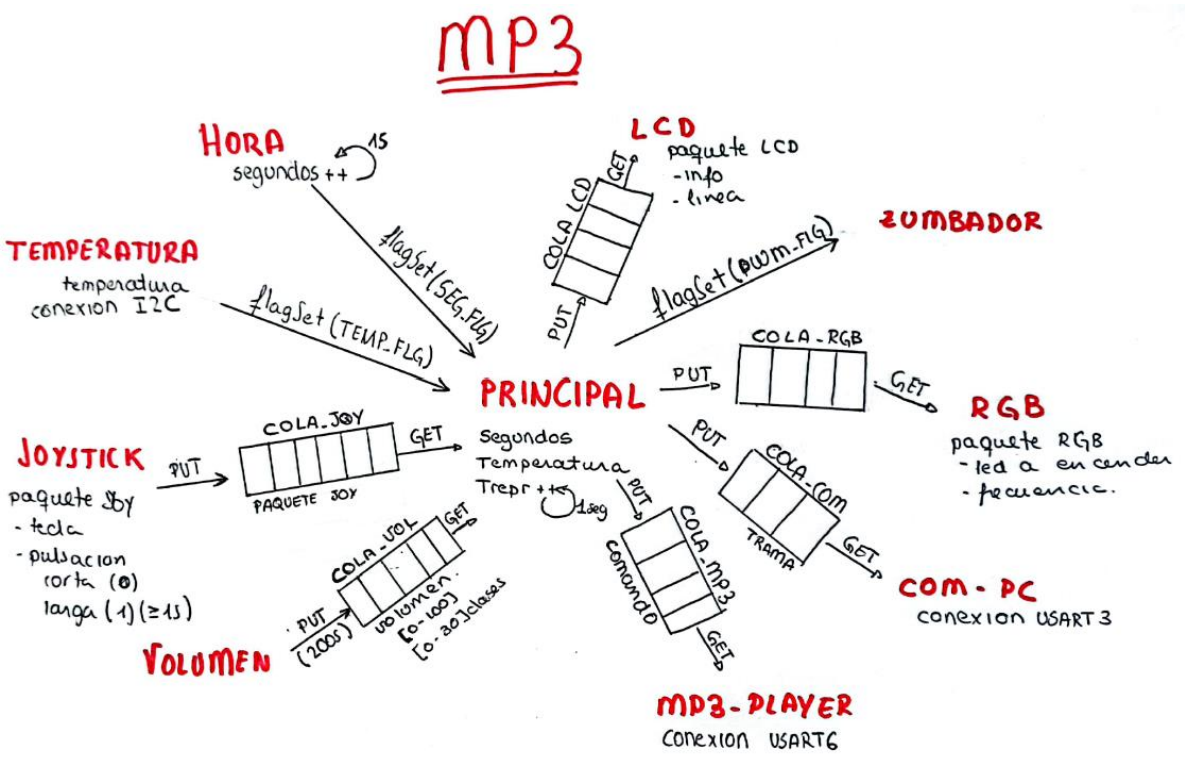
-Competencias adquiridas:

- [CE EC04] He podido crear mis propios proyectos utilizando el software y conocimientos obtenidos para programar.
- [CE EC07] Utilizando tanto el RTOS, el cual permite controlar de forma sencilla a nivel alto un sistema formado por varios Thread, sistema que me ha permitido crear un sistema formado por varios módulos. También he aprendido a mandar información utilizando el protocolo USART para poder enviar información al ordenador y viendo los datos enviados utilizando el programa TERA TERM.

-Resultados del aprendizaje:

- **[RA971]** Utilizando tanto Timers hardware, como Timers virtuales he conseguido temporizar señales para el buen funcionamiento de una aplicación y la sincronización del mismo.
- **[RA970]** Utilizando tanto la placa mbedAPPBoard, la placa NUCLEO F429ZI y el MP3 reproductor serie he podido crear una aplicación útil. He tenido algún problema a la hora de utilizar el MP3 ya que me daba algunos problemas y el datasheet completo con todos los comandos era difícil conseguirlo ya que el fabricante tiene una pagina web la cual está actualizándose y no he podido buscarlo, ni he podido conseguir a través de internet ninguno. De todas formas, la funcionalidad básica si que he podido programarla. También cabe decir, que los comandos no tienen una descripción que te sea fácil de entender lo que hace cada uno. **Al finalizar la práctica, no he podido conseguir que funcione el módulo de comunicación con el PC juntándolo con el resto del sistema. Al juntarlo con el sistema, lo que ocurre es que por alguna razon hace que el thread que controla el LCD termine y deje de enviar datos al LCD de la placa mbedAPPBoard. No he conseguido saber el porqué, ya que individualmente el bloque COM funciona perfectamente**. El modulo COM al final de practica es el que me ha dado mas problemas, los cuales ya se me salen de mi conocimiento que tengo, el cual me gustaría solucionar ya que tras depurar no he conseguido saber el porque.
- **[RA970]** Apoyándome en compañeros y mi profesor de laboratorio he podido sacar satisfactoriamente todos los problemas que he ido teniendo a lo largo del curso.
- **[RA736]** Finalmente he podido interpretar bien las especificaciones y a plantearme antes de programar todos los módulos, lo que me ha ayudad a tardar menos tiempo en hacer el programa.
- **[RA737]** He aprendido nuevas formas de codificar y de resolver los problemas de la aplicación, en especial, el pensar a nivel de bit.
- **[RA733]** Finalmente he podido distinguir donde buscar la información que necesitaba utilizando documentación que ofrece KEIL y CMSIS.

1.5 **Diagrama de bloques hardware del sistema.**



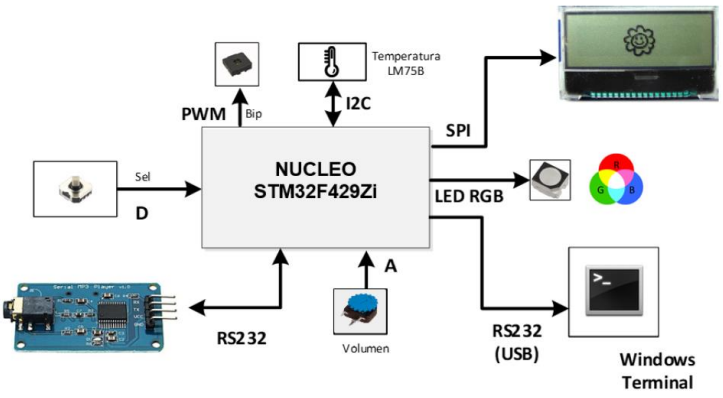


Ilustración 1: Conexiones entre dispositivos

1.6 Cálculos realizados, justificación de la solución adoptada, descripción de los módulos del sistema.

En este punto debe describir como ha configurado cada uno de los recursos del microcontrolador, los cálculos que haya realizado y los valores programados en los registros más significativos.

MODULOS:

CLOCK: Para configurar este modulo ha sido necesario un timer virtual de tipo periódico de 1 segundo el cual se utiliza para ir incrementando el valor de los segundos cada vez que se entra en el callback de dicho timer (tid_timer_1s). También he creado un thread para poder empezar el timer una vez iniciado el modulo desde el programa principal (main). Cabe destacar que cada vez que se incrementa la variable global segundos, se manda una señal al hilo principal para que haya sincronización a la hora de representar dicho valor en el LCD y que así no tenga el problema de que represente el valor todo el rato.

JOYSTICK: Para el joystick ha sido necesario configurar los pines establecidos en el guion de la práctica, con la activación de los correspondientes relojes. Estos pines se han configurado para que cada vez que haya una pulsación, creen una interrupción y así poder detectar que se ha pulsado y cuando no. Como ya conocemos como funciona un pulsador, detecte que los rebotes son un gran problema y lo he solucionado creando un timer que cada 50 ms se active, en cuanto detecte una interrupción en cualquier pulsador (arriba, abajo, izquierda, derecha, arriba, centro) y se reactive si dentro de los 50 ms se ha detectado otra interrupción. Cuando se ha conseguido terminar el timer de 50ms sin interrupciones, se inicia otro timer de 1s para detectar pulsaciones cortas y largas. La solución que yo he propuesto es mandar en todo momento que se trata de una pulsación corta y solamente cuando este timer de 1s termina (es decir, llega a su callback) se asignara a la variable pulsación el valor 1 para que posteriormente se pueda detectar si es corta o larga la pulsación. Cabe destacar de este módulo, que a la hora de detectar una interrupción, para implementar el modulo de PWM se envía una señal al thread de dicho modulo (tid_PWM), que posteriormente se explicara.
PWM: Este modulo encargado de hacer un bip cada vez que se pulsa un pulsador se ha configurado con el pin establecido por la guía de la práctica, utilizando así la función alternativa de dicho pin para poder utilizarlo con el modo GPIO_MODE_AF_PP y con la función alternativa GPIO_AF1_TIM2 la cual nos permite utilizar este pin para generar una señal cuadrada modelada por ancho de pulso y así poder "llevarla" al zumbador de la placa MBED APPBoard y generar el 'bip' requerido. Para generar dicha señal se ha creado un timer hardware utilizando el Timer 2 y configurando la estructura TIM_HandleTypeDef con un prescaler de 84, un periodo de 1000 cuentas y la estructura TIM_OC_InitTypeDef con el modo TIM_OCMODE_PW1 que nos permite crear dicha señal PWM junto con el valor del pulso, para que suene mas fuerte o menos fuerte el 'bip'. Dicho valor lo he configurado a 9 para que sea un leve 'bip'. Como ya se ha dicho anteriormente, este modulo a la hora de empezar el timer espera antes de todo a la señal que se manda desde el modulo del joystick cada vez que se pulsa un pulsador y es cuando se recibe cuando se genera dicha señal la cual dura 100ms. Al componer el paquete JOY hay que llenar una estructura con la tecla pulsada y si la pulsación ha sido corta (0) o larga (1). Los valores de cada tecla vienen dados por la siguiente tabla.

Gesto	Valor
Arriba	1
Derecha	2
Abajo	3
Izquierda	4
Centro	5

VOL: Modulo que se encarga del cambio de volumen del MP3. Este modulo se apoya en el módulo ya codificado ADC, dado a nosotros por los profesores, el cual devuelve un valor de voltaje devuelto por el potenciómetro de la placa MBED APP Board. Este modulo esta cada 200 ms recibiendo una señal la cual inicia

el proceso de verificar si el valor cogido anteriormente es igual al valor de ahora y si es distinto envía a la COLA_VOL el correspondiente valor.

MP3: Módulo que se encarga de recibir los comandos por la COLA_MP3 y enviarlos a través del protocolo USART al Serial MP3 Player. Se ha utilizado la USART6, correspondiente a los pines establecidos.

COM: Modulo que se encarga de enviar la trama requerida al PC utilizando la interfaz USART3.

LCD: Modulo que se encarga de encender aquellos bits del LCD correspondientes a l buffer que se enviar a dicho LCD utilizando el bus SPI1. podrá encontrar en la cabecera la función que se encarga de toda la inicialización del módulo, el cual inicializa el bus SPI, los pines correspondientes para la configuración y puesta en marcha del LCD (CS, A0, RST)

LED: Modulo que se encarga de recoger de la COLA_RGB el led a encender y la frecuencia a la que se quiere encender dicho led. Se ha configurado los pines de los leds para generar una señal de salida y así que llegue esa señal a los pines de los leds RGB de la placa de aplicaciones.

TEMP: Modulo encargado de consultar la temperatura utilizando el sensor LM75BD de la placa de aplicaciones. Se ha tenido que configurar el bus I2C para poder transmitir este valor a dicho modulo. Configurándolo de la siguiente forma:

```
I2Cdrv->Initialize (NULL);
I2Cdrv->PowerControl (ARM_POWER_FULL);
I2Cdrv->Control     (ARM_I2C_BUS_SPEED, ARM_I2C_BUS_SPEED_STANDARD);
I2Cdrv->Control     (ARM_I2C_BUS_CLEAR, 0);
```

En este proceso de intercambio se pregunta al esclavo con dirección 0x48 que devuelva el valor, el cual lo recibe en 2 bytes y quedándonos solamente con los 11 bits de mayor peso y multiplicándolo por 0,125 como dice el fabricante:

Table 8. Temp register

MSByte								LSByte							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	X	X	X	X	X

When reading register Temp, all 16 bits of the two data bytes (MSByte and LSByte) are provided to the bus and must be all collected by the controller to complete the bus operation. However, only the 11 most significant bits should be used, and the 5 least significant bits of the LSByte are zero and should be ignored. One of the ways to calculate the Temp value in °C from the 11-bit Temp data is:

1. If the Temp data MSByte bit D10 = 0, then the temperature is positive and Temp value (°C) = +(Temp data) × 0.125 °C.
2. If the Temp data MSByte bit D10 = 1, then the temperature is negative and Temp value (°C) = −(2's complement of Temp data) × 0.125 °C.

El valor de la temperatura se actualiza en el hilo principal cuando se detecta un cambio. Esta comunicación la he solucionado utilizando una señal que avisa al principal que se ha cambiado dicho valor.

PRINCIPAL: Modulo que se encarga de controlar todo el sistema. En este módulo, se crean las colas del lcd, rgb, teraterm y del mp3. Colas que servirán para comunicarse con los módulos anteriormente explicados. También tiene un thread que se ha creado para manejar los 2 modos correspondientes al REPOSO y a la REPRODUCCION. Entrando en thread, la primera secuencia que se hace es consultar las colas del JOYSTICK y del VOLUMEN cada 200 ms. Segundo se envía una secuencia de iniciación al mp3 a través de la cola MP3. Secuencia que se ejecuta al principio y solo al principio del programa y deja al MP3 en SLEEP MODE. Como el modo inicial es el reposo, se entra en modo = 0, momento en el que se consulta si se ha activado los flags correspondientes a los segundos y a la temperatura, si es así se actualiza la línea correspondiente del LCD. En este modo solamente se puede interactuar con él a través de una pulsación larga del pulsador del centro. Tras dicha activación, se activa el modo reproducción (modo = 1)y se envía una señal que se recoge

en el modo reproducción y se despierta al MP3 y se reproduce una canción de la 1 carpeta. Cabe destacar que cada vez que se envía un comando al MP3, se envía el mismo comando al tera term. Nada mas entrar en el modo reproducción, se inicializa un timer ya que se esta reproduciendo la canción, el cual sirve para contabilizar el tiempo que lleva la canción en reproducción. Después se espera a que las señales correspondientes del volumen, del cambio de modo, del cambio de careta y/o de canción y del tiempo de reproducción se activen y así representar la información en el LCD. En este modo se espera a la pulsación de cualquier tecla siguiendo la especificación de la practica:

- Right → Se selecciona la siguiente canción disponible en la carpeta actual.
- Left → Se selecciona la canción anterior disponible en la carpeta actual.
- Down → Se selecciona la siguiente carpeta.
- Up → Se selecciona la carpeta anterior.
- Center → Permite poner el reproductor en pausa o en reproducción.
- Center Larga → Se sale de este modo y se pasa al modo REPOSO.

Cabe destacar que cada vez que se cambia de canción o de carpeta se envia una señal al mismo thread, para que sepa que se ha cambiado de canción y/o carpeta. También cada vez que se cambia de canción se vuelve a activa el tiempo de reproducción (Trepr) y poniendo dicha variable a 0 para que se incie de nuevo. Solamente cuando se pone en pausa este timer se para y se reanuda cuando se pasa a reproducirse la canción. Cuando se desea volver al otro modo, el MP3 entra en modo sleep y se reanuda todo el procedimiento de ejecución.

2 DEPURACION Y TEST

2.1 **Pruebas realizadas.**

LCD: Representar en dicho LCD. Enviar desde un thread al LCD un mensaje de prueba para observar si se representa adecuadamente. También se ha probado con un valor que va cambiando con el tiempo para observar si se actualiza como nosotros queremos. Este test se ha conseguido lo que esperaba.

Zumbador: Este modulo no ha sido necesario la creación de un programa de test ya que es bastante independiente y solamente necesita recibir una señal de un modulo de fuera, señal que se recibe correctamente.

RGB: Este modulo se tuvo que hacer un mini proyecto para comprobar el funcionamiento de los leds, en el que se encendía y apagaba cada led síncronamente. Este test resulto correcto.

MP3: Test que mandaba una secuencia de comandos al mp3 el cual, había a veces que el test se completaba correctamente y otras no. Estos fallos yo creo que es porque no debí de entender correctamente la secuencia inicial que hay que hacer para activar el mp3. A la hora de implementarlo con el sistema completo, me ha dado algún que otro problema. Ya que había a veces que reseteaba todo y me funcionaba y lo volvía a hacer y ya no funcionaba. Consultando con varios compañeros, les ha ocurrido lo mismo, llegando a la conclusión de que el MP3 falla en algunas ocasiones, y el MP3 debe entrar en un estado desconocido y no trabaja como debería.

Temperatura: Ha sido necesario la creación de un test para aprender a utilizar el i2c y recibir información a través del I2C. Creando 1 hilos que reciba la información del sensor de temperatura para así comprobar con la configuración que aporta la pagina web de keil sobre el bus I2C ha sido correcto el funcionamiento.

Los demás módulos han sido comprobados con practicas anteriormente hechas en la asignatura.