Sistemas Basados en Microprocesador

Integración y desarrollo de una aplicación: Controlador de sensor inercial – acelerómetro.

Alumno:
A: Antonio Doña
B: Patricia Alarcón

2024-25

Departamento de Ingeniería Telemática y Electrónica
Universidad Politécnica de Madrid

Índice del documento

1 OBJETIVOS DE LA PRÁCTICA

1.1 Resumen de los objetivos de la práctica realizada

Basándonos en CMSIS-Driver y CMSIS-RTOS2, se busca obtener un controlador capaz de realizar medidas de aceleración en 3 ejes con rango máximo {+-2g}, con frecuencia de 1Hz.
Dentro de estas medidas el controlador debe ser capaz de realizar 10 medidas y almacenarlas en un buffer para poder ser volcadas a un PC por medio de comunicación serie (símil básico de RS-232) bajo petición de este.
Otras funcionalidades que permite el controlador son la programación de parámetros como los umbrales de referencia de aceleración (los cuales ante un exceso producirán un código led en cada eje) y la programación de la hora.
Estas funciones son también ejecutables a través del joystick integrado, navegando con pulsaciones centrales sostenidas por 3 modos de operación: Reposo, Activo y Programación. En este último se podrán gestionar todas las variables tanto con el controlador como desde el PC.
Cada una de estas interacciones reporta al equipo PC una trama al ejecutar dicha interacción.

1.2 Acrónimos utilizados

Identifique los acrónimos usados en su documento.

UART	Universal Asynchronous Receiver Transmitter
I2C	Inter-Integrated Circuit
CMSIS-RTOS2	Cortex Microcontroller Software Interface Standard – Real-Time Operating System (ver.2)
SPI	Serial Peripheral Interface Bus
GPIO	General Purpose Input/Output
RGB	Red-Green-Blue
IRQ	Interrupt Request
USART	Universal Synchronous/Asynchronous Receiver Transmitter
LCD	Liquid Cristal Display
RS-232	**R**ecommended **S**tandard 232
FSM	Finite State Machine

1.3 Tiempo empleado en la realización de la práctica.

Debe realizar una descripción sencilla del tiempo que ha dedicado a la realización de las actividades relacionadas con la práctica.

[Tiempo empleado para realizer la práctica]: El tiempo total empleado ha sido de 36 horas.

Para la entrega 1 (que se compone por: joystick, hora) hemos empleado: 2 horas.
Para la entrega 2 (que se compone de acelerómetro, lcd, leds núcleo): hemos empleado: 10 horas.

Para la entrega 3 (que se compone del com-pc) hemos empleado: 6 horas.

Para el proyecto final, unificando todos los módulos, hemos tardado: 16 horas.
Para realizar la memoria, hemos tardado: 2 horas.

1.4 Bibliografía utilizada

Para realizar esta práctica a parte de haber utilizado los apuntes proporcionados por Moodle. También hemos dado uso de las siguientes bibliografías para realizar la práctica.

[RD1] "NUCLEO-F429ZI-USER MANUAL", STM.

[RD2] Esquemático mbed-Aplication-Board

[RD3] "STM32F429-REFERENCE-MANUAL", STM.

[RD4] "STM32F429-DATASHEET", STM.

[RD5] https://arm-software.github.io/CMSIS_6/latest/General/index.html

[RD6] Barr, Michael. "Embedded C Coding Standard", BARR Group, 2018

[RD7] Noviello, Carmine. "Mastering STM32 Release 2.0", 2022

1.5 Autoevaluación.

Tras revisar los objetivos de aprendizaje indicados en la guía de la asignatura, consideramos que sí cumplimos con los requisitos, debido a que, durante el desarrollo del proyecto, hemos podido implementar al código nuevos conocimientos, como es: el USART (en el com-pc) y el I2C (en el mpu6050).

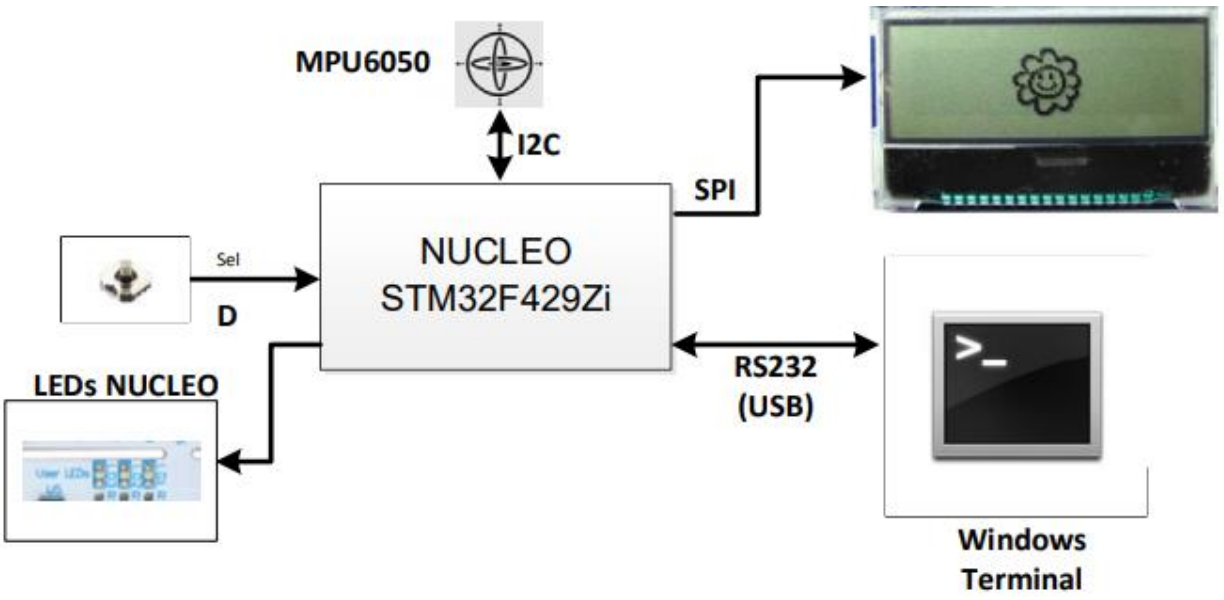
Su dificultad fue hacer una búsqueda exhaustiva en el documento de referencia, para poder hacer un uso correcto del I2C y en particular el uso del MPU-6050 y sus modos de operación. Pero al encontrar la correcta implementación del I2C y del USART, reforzó nuestras capacidades de análisis, concretamente en el uso de datasheets de comunicación entre dispositivos y mapas de registro.
Consideramos que las funcionalidades buscadas han dado resultado y por tanto se cumplieron los requisitos.

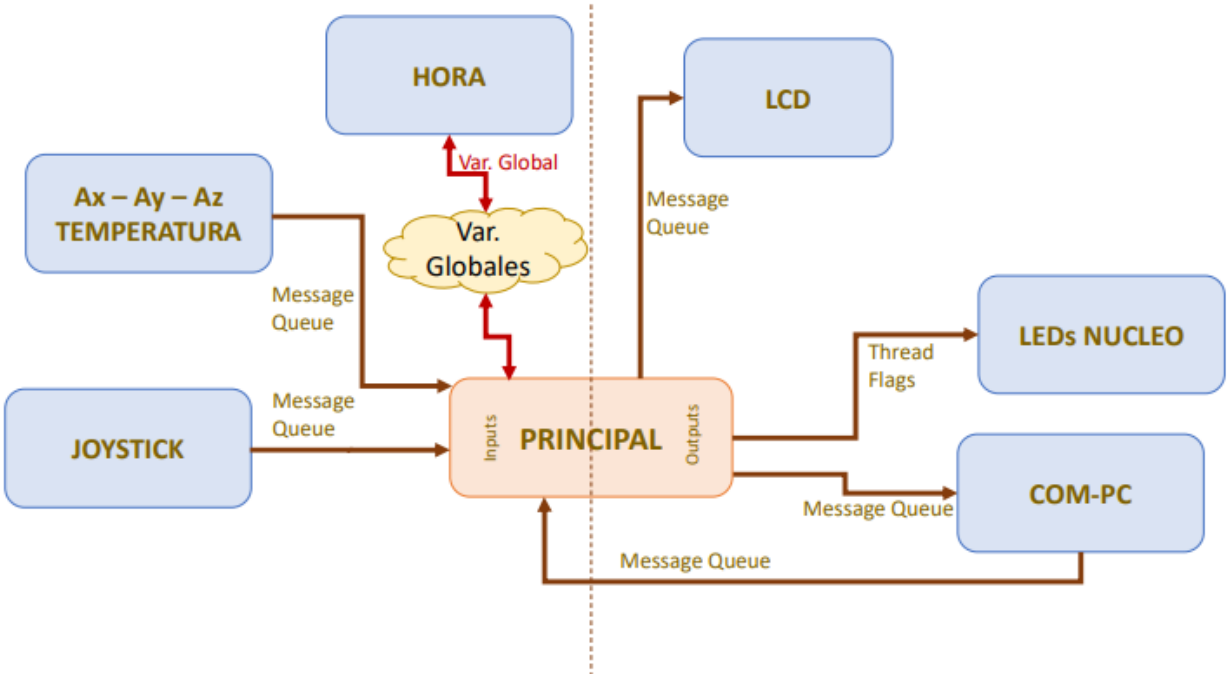
2 RECURSOS UTILIZADOS DEL MICROCONTROLADOR

2.1 Diagrama de bloques hardware del sistema.

Las conexiones deben ser de la siguiente manera obligatoriamente:

		Pin mbed app board	STM32F429	NUCLEO STM32F429Zi
VCC	+ 3.3V	DIP40	+ 3.3V	CN8.7 / +3.3V
GND	GND	DIP1	GND	CN8.11 / GND
LCD (SPI)	MOSI	DIP5	PA7	CN7.14 / D11
	SCK	DIP7	PA5	CN7.10 / D13
	CS	DIP11	PD14	CN7.16 / D10
	A0	DIP8	PF13	CN10.2 / D7
	RESET	DIP6	PA6	CN7.12 / D12
Joystick	Down	DIP12	PE12	CN10.26 / D39
	Left	DIP13	PE14	CN10.28 / D38
	Center	DIP14	PE15	CN10.30 / D37
	Up	DIP15	PB10	CN10.32 / D36
	Right	DIP16	PB11	CN10.34 / D35





2.2 Cálculos realizados y justificación de la solución adoptada.

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.

Gracias a CMSIS-RTOS2, una parte de cálculo ha sido solventada gracias al uso de timers en SW respecto a timers físicos como en entregas previas, pudiendo usar parámetros directamente en milisegundos, definidos a través de macros en la mayoría de casos. En concreto, para cada módulo:

- Clock: cada segundo fue sumado hasta llegar a 60 segundos, sumando uno a minutos y reiniciando a su vez los segundos. Una vez llegado a 60 minutos y 60 segundos, se suman las horas, reiniciando los otros a cero. La hora se reinicia por completo cuando llega a 24 horas, 60 minutos y 60 segundos.

- Joystick: para las pulsaciones largas inicia un timer, si dura un segundo pulsando el botón, se igualará a una pulsación larga. Al contrario, será una pulsación corta.

- Mpu: para los valores del eje, combinaremos los bytes en un valor de 16 bits con signo en rango {+-2g} (resultando en lecturas de 2^14 valores) y se dividirá por 16384, para conseguir el valor del eje. En cambio, si queremos obtener los grados, combinaremos los bytes en un valor de 16 bits con signo y con ese resultado para obtener la temperatura haremos la siguiente operación: (bytes/ 340.f) + 36.53f.

- Leds: dependiendo del flag que le llegue encenderá un led u otro o ninguno.

- Com: El formato de la comunicación RS232 será 115200 baudios, 8 bits de datos, 1 bit de stop y sin paridad. Al enviar la macro, la recibirá de byte a byte .

3 SOFTWARE

3.1 Descripción de cada uno de los módulos del sistema

En este punto debe explicar el funcionamiento de cada uno de los módulos que haya desarrollado.

Modelo Hora:

Se implementa un código con 3 contadores secuenciales para 3 variables {segundos, minutos, horas} codificando el funcionamiento de un reloj 24H.

- Cola de mensajes: no contiene colas de mensajes.
- Variables globales: horas, minutos y segundos.
- Threads: contiene el thread llamado clock_Thread. Se inicia el timer y pone todas las variables globales a 0.
- Timer: hemos creado un timer periodico de duración de 1 segundo. Se inicia en el Thread de hora. Cada vez que pasa un segundo, en el Callback clock sumamos los segundos. Una vez llegue a 60 segundos, se suma 1 minuto, reseteando a 0 los segundos. Y cuando lleguen estos a 60 segundos y 60 minutos, se suma una hora, reseteando los segundos y minutos a 0.

Modelo Joystick:

El código ejecuta una máquina de estados donde tras la interrupción externa generada en los pines conectados al joystick y contemplando los rebotes, se envía un mensaje joy_msg con el botón pulsado codificado numéricamente al soltar el joystick o bien al caducar el tiempo de pulsación larga, (tras 1s; joy_msg.longpress = 1) el primero en suceder.

- Cola de mensajes: contiene la información de qué botón se pulsó y si fue un pulso largo o corto.
- Variables globales: ninguna, la información se pasará por colas.
- Thread: La gestión del joystick se realizará por interrupciones, eliminando los posibles rebotes que éste origine, distinguiendo entre pulsaciones cortas y largas.
- Timer: timer con duración 50 ms para anular rebotes y timer de 1s para discernir pulsaciones cortas y largas.

Modelo LEDs:

El código espera eternamente algún flag del 0x07. En ningún momento se encenderán los tres leds a la vez, solo se encenderá uno. Para que se encienda el LED 1 le tendrá que llegar el flag 0x01. El LED 2 se encenderá si le llega un 0x02 y por último, el LED 3 se encenderá con el flag 0x04.

- Cola de mensajes: no contiene cola de mensajes.
- Variables globales: ninguna.
- Thread: La gestión de los LEDs se realizará por flags. Esperará eternamente hasta que le llegue algún flag y dependiendo del flag que le llegue, encenderá un led, otro o ninguno.
- Timer: no contiene timers.

Modelo LCD:

Módulo encargado de representación de información por el LCD conectado al bus SPI. Thread leyendo de cola de mensajes con la información que debe representarse en el LCD y la línea en que debe representarse

- Cola de mensajes: contiene cola de mensajes para sincronizarse. En la cola de mensajes le llega una estructura compuesta por la cadena que debe representar y en la línea que tiene que mostrarlo.
- Variables globales: ninguna.
- Thread: La gestión de los LEDs se realizará por flags. Esperará eternamente hasta que le llegue algún flag y dependiendo del flag que le llegue, encenderá un led, otro o ninguno.
- Timer: no contiene timers.

Modelo MPU-6050:

Módulo encargado de la toma de datos de acelerómetro en 3 ejes y la temperatura ambiente incorporados en el chip MPU6050 a través del protocolo I2C con cadencia de lecturas de 1 segundo.

- Cola de mensajes: Genera una cola de mensajes donde reportará los resultados de lectura convertidos al módulo principal.
- Variables globales: mpu6050_msg, read_buffer, conf_buffer; responsables de generar estructuras más versátiles para el tratamiento de datos de mensaje, de lectura y de escritura respectivamente.
- Thread: Tras verificar la correcta inicialización del HW y cola de mensajes realizará la espera al módulo principal, el cual en el modo Activo generará señales de lectura cada segundo. En caso de estar en cualquier otro modo, dicha flag no se activará y quedará a la espera sin generar nuevos mensajes.
- Timer: no contiene timers. El timer de lectura será gestionado desde el principal, facilitando la coordinación del código.

Modelo COM-PC:

Módulo encargado de la comunicación con el PC a través de la línea serie integrada en el USB.

- Cola de mensajes: Contiene dos colas de mensajes. Una llamada msg_comReceive, que guarda la macro enviada por Tera Term y se la envía a la principal, y la otra llamada msg_comSend, que recibe del principal lo que le tiene que enviar al Tera Term.
- Variables globales: Ninguna, la información se pasará por colas.
- Thread: Tenemos dos hilos. Una hilo llamado tid_com_receive recibe byte a byte la macro que fue enviada desde Tera Term y la guarda en una trama. Una vez captada toda la trama, se la envía al principal. El segundo hilo llamado tid_com_send recibe desde la cola la trama que tiene que enviar al Tera Term.
- Timer: no contiene timers.

Modelo Principal:

Módulo principal del sistema que se encarga de coordinar todos los demás. Es el módulo que decide las acciones a tomar en función del modo del sistema y de la información que reciba del resto de los módulos.

- Cola de mensajes: Toda la información, le llega por colas, y envía una cola al com-pc.
- Variables globales: horas, minutos y segundos vienen externas del módulo clock.
- Threads: Tiene un thread, en él, unifica todos los módulos y decide las acciones a tomar.
- Timer: Ponemos un pequeño timer, de esta manera no envía constantemente una cola al lcd y no se ve parpadeo.

3.2 Descripción global del funcionamiento de la aplicación. Descripción del autómata con el comportamiento del software (si procede)

El programa comenzará en el modo REPOSO. En este modo se presentará en el LCD el mensaje “SBM 2024” y la hora. Tras un reset, el reloj marcará las 00:00:00. En este modo sólo se atenderá a una pulsación larga del botón CENTER del joystick, que hará que el sistema pase a modo ACTIVO.

El modo ACTIVO estará leyendo la aceleración y la temperatura que entrega el sensor MPU6050 cada segundo. Los valores de la aceleración obtenidos en cada eje (Ax, Ay y Az) se compararán con los valores por defecto de referencia (Ax_r=Ay_r=Az_r=1.0). Si los valores obtenidos son mayores que los de referencia se iluminará un led de la tarjeta NUCLEO: eje x->LD1, eje y->LD2, eje z->LD3. Si los valores obtenidos son menores se deberán apagar los leds correspondientes. El sensor debe inicializarse para utilizar el rango ±2g.

Todas las referencias de los ejes comenzarán por 1.0. Más Adelante se podrán modificar esos valores de referencia para cada eje.

Se debe mostrar la temperatura (T) obtenida del sensor y el valor de la aceleración para los tres ejes.

El sistema debe mantener, cuando se encuentre en este modo, una tabla de 10 medidas almacenadas en un buffer circular. Cada entrada debe mantener en ASCII la siguiente información (hora, temperatura medida, y los valores de aceleración): HH:MM:SS--Tm:TT.Tº-Ax:n.n-Ay:n.n-Az:n.n

Desde el modo ACTIVO podrá pasarse en cualquier momento al modo PROGRAMACIÓN/DEPURACIÓN pulsando la posición CENTER del joystick.

En el modo PROGRAMACIÓN/DEPURACIÓN se programará la hora del sistema y los valores de referencia de la aceleración para los tres ejes. La modificación de estos valores se realizará utilizando el joystick (pulsaciones cortas), presentando los cambios realizados en el LCD.

Utilizando los gestos UP/DOWN/LEFT/RIGHT se modificarán los valores y para hacerlos efectivos se realizará una pulsación corta en el gesto CENTER. Se modificarán sucesivamente los parámetros hora, Ax_r, Ay_r y Az_r tras sucesivas pulsaciones del gesto CENTER.

Desde el modo PROGRAMACIÓN/DEPURACIÓN podrá pasarse en cualquier momento al modo REPOSO realizando una pulsación LARGA del gesto CENTER del joystick.

En este modo, el sistema además de responder a las peticiones del joystick, se controlará desde un PC a través de un canal serie de comunicaciones RS232. El formato de la comunicación RS232 será 115200 baudios, 8 bits de datos, 1 bit de stop y sin paridad. Es obligatorio que las comunicaciones serie se gestionen con CMSIS Driver (USART) utilizando la función de callback (interrupción/eventos).

Para el envío de estas tramas/comandos se recomienda el uso del programa Teraterm. Dentro de este programa se pueden utilizar macros para enviar tramas, ficheros de texto, en los que se definen cada uno de los bytes que componen dicha trama. En Moodle se encuentra un ejemplo de este tipo de ficheros.

Las tramas que no estén bien formadas, no respetando el criterio indicado en la tabla anterior, deberán ser ignoradas por el sistema.

El funcionamiento del software del sistema debe desarrollarse de manera modular, de forma que cada uno de los módulos que componen el mismo se encargue de la gestión de un aspecto determinado (tarea,

periférico, sensor, funcionalidad, etc). Además de los módulos específicos, deberá desarrollarse un módulo principal que se encargará de la gestión y sincronización de todos ellos. Cada uno de los módulos específicos debe ser autocontenido y tendrá únicamente interacción con el módulo principal, sin posibilidad de ninguna comunicación directa entre módulos específicos que no esté controlada por el módulo principal.

El software correspondiente a cada uno de los módulos debe estar recogido en un fichero que contenga el código (.c) y un fichero de cabecera (.h).

3.3 Descripción de las rutinas más significativas que ha implementado.

En este punto debe enumerar y describir la funcionalidad de las rutinas más importantes que ha implementado.

- I2C: rutina encargada de comunicarse con la placa para hallar el eje x, eje y, eje z y la temperatura. Se basa en una comunicación entre maestro-esclavo para que mediante los buffers de los bit mas siginificativos y menos significativos se halle la temperature y los ejes.

- I/O pulsadores: rutina que se activa cuando se realiza una pulsación corta o larga de cualquier botón. Se han evitado los rebotes. Dichos rebotes se resuelven con timer de 50 ms.

- Colas: implementación para la comunicación entre módulos. Se encargan de enviar información mediante mensajes.

- GPIO_WRITE_PIN: se encarga de encender o apargar pines con set y reset.

- OsDelay(): se encarga de poner un determinado tiempo, actúa como un timer.

- USART: es el encargado de enviar macros al principal, y en el principal se gestiona cual será su funcionamiento. Una vez analizado en el principal cual es la función de la macro recibida, el principal envía al com para que muestre en el tera term si hubo algún fallo.

- SPI: se encarga de enviar la cadena a mostrar en el LCD.

4 DEPURACION Y TEST

4.1 Pruebas realizadas.

Descripción del método de prueba utilizado para comprobar que las rutinas funcionan adecuadamente. Resultados de los tests. Indicación explicita de si el test es correcto o incorrecto. En este punto debe hacer especial hincapié en definir:

1. Cuál es el objetivo de la prueba, indicando los módulos implicados
2. Cuál es el proyecto de Keil que permite realizar la prueba
3. Cuáles son las condiciones de entrada que permiten ejecutar la prueba
4. Cuáles son los resultados que se esperan y cuáles son los realmente obtenidos.

Modelo Hora:

- Test: la prueba realizada es poniendo las horas: horas = 23, minutos = 59 y segundos = 55. De esta manera podremos ver a través de watches y el led1 como los segundos cambian correctamente y al llegar a horas = 23, minutos = 59 y segundos = 59, se inicia con todas las variables a 0.

Modelo Joystick:

- Test: Se evalúa la captura del mensaje en rx_msg de tipo MENSAJE_JOY. Se representa por los leds del STM la correcta adquisición del joystick.

Modelo LEDs:

- Test: Se evalúa que el flag que llegue, encienda el led correspondiente. Y le envío el flag 0x03 para comprobar que no hace nada y no encienda dos leds a la vez.

Modelo LCD:

- Test: El test envía al LCD una cadena de caracteres en la línea 1, que tiene que representar en el LCD. Luego envía otra cadena de caracteres para la línea 2, sin hacer ninguna modificación a la línea 1.

Modelo MPU-6050:

- Test: Se integra en la funcionalidad de test del LCD, la captura del mensaje de la cola del mpu6050 y se representa con un decimal las lecturas de los ejes y la temperatura de acuerdo al test previamente configurado

Modelo COM-PC:

- Test: Para probar que su funcionamiento es correcto y que recibe y envía correctamente, haremos que el hilo de recepción envíe su trama (una vez recibida correctamente) y se la envíe por la cola al hilo de enviar. De esta manera observaremos, que el módulo recibe y envía al Tera Term adecuadamente.

Modelo PRINCIPAL:

- Test: Comprobaremos si se coordinan todas las acciones. Comprobaremos si cambia de modo correctamente mediante la pulsación center. Si realiza correctamente las medidas del acelerómetro y la cuenta del reloj. En el modo programación verificaremos si cambia adecuadamente los valores de las medidas seleccionadas. Y por último, evaluaremos todas las posibles macros que puedes analizadas.