Sistemas Basados en Microprocesador

Departamento de Ingeniería Telemática y Electrónica
Universidad Politécnica de Madrid

2023-24

Integración y desarrollo de una aplicación:

Sistema de control de iluminacion ambiental

Alumno:
A: Lukas Gdanietz de Diego..
B:……..

Puesto Nº: x

1 OBJETIVOS DE LA PRÁCTICA

1.1 Resumen de los objetivos de la práctica realizada

Se deben enumerar los objetivos que ha alcanzado al realizar la práctica. Enumérelos de forma precisa y sencilla.

1.2 Acrónimos utilizados

Identifique los acrónimos usados en su documento.

UART	Universal Asynchronous Receiver Transmitter
SPI	Serial Peripheral Interface
I2C	Inter-Integrated Circuit
O.S. / OS	Operating System
HAL	Hardware Abstraction Layer
PWM	Pulse Width Modulation
CS	Chip Select
CMSIS	Cortex Microcontroller Software Interface Standard

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 x horas.

1.4 Bibliografía utilizada

[RD1] Hoja de catalogo xxxx, libro, manual de usuario, etc

[RD2] Serial Wire Viewer (SWV) for ARM Cortex-M3 Processors. ARM Real-Time Trace aids debugging. [LINK]

[RD3] Using Serial Wire Viewer (SWV) to Confirm CPU Speed [LINK]

[RD4] How to display printf output over SWO in Keil µVision [LINK]

[RD5] Serial Wire Viewer (SWD + SWO) - fast & native Debugging [LINK]

[RD6] Reference Manual STM32F429

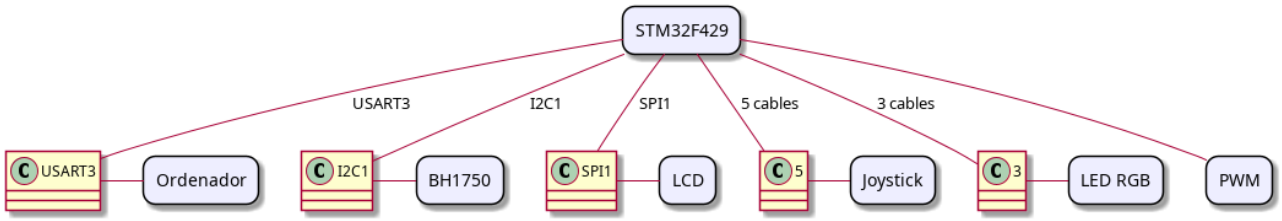
1.5Autoevaluación.

CODIGO	Objetivo	Adquisicion	Anotaciones
RA971	Manejar temporizadores hardware para gestionar la temporización y sincronización de una aplicación	✔️	
RA970	Establecer y gestionar las comunicaciones entre dos sistemas utilizando diferentes interfaces	✔️	
RA907	Desarrollo de aplicaciones en grupos de trabajo	✔️	
RA736	Interpretar las especificaciones de funcionamiento de un sistema basado en microcontrolador de mediana complejidad	✔️	
RA737	Escribir el código necesario para desarrollar una aplicación basada en microcontrolador de mediana complejidad	✔️	
RA730	Conectar un periférico a un microcontrolador utilizando interfaces basadas en protocolos estándar	✔️	
RA735	Analizar la arquitectura software y hardware de sistemas basados en microcontrolador de mediana complejidad	✔️	
RA733	Aprender a manejar cualquier periférico de mediana complejidad de un microcontrolador a partir de la documentación proporcionada por el fabricante	✔️	
RA968	Manejar instrumentación electrónica específica para el desarrollo de sistemas basados en microprocesador	✔️	
RA738	Elaborar el informe que justifica y describe la toma de decisiones adoptadas en el desarrollo de un proyecto y defenderlo oralmente con precisión y detalle	✔️	
RA734	Manejar entornos de CAD para la codificación, la compilación y la depuración de aplicaciones basadas en microcontrolado	✔️	

Anotacion personal: Trabajo como programador de dispositivos embeddidos en la industria automovilistica desde hace unos años.

2 RECURSOS UTILIZADOS DEL MICROCONTROLADOR

2.1Diagrama de bloques hardware del sistema.



2.2Cálculos realizados y justificación de la solución adoptada.

- USART: El STM32 se configura para que utilice el periférico USART3 Se configura el periférico para que funcione de forma asíncrona, con 8 bits de datos, 0 bits de paridad, 1 bit de stop, y una velocidad de transmisión de 9600 Baudios. Además, se activan tanto la recepción como la transmisión, con salida en los pines PD8 (TX) & PD9 (RX).
- I2C: El STM32 se configura para que utilice el periférico I2C1. El periférico se inicializa para que opere a 400Khz.
- Timer: El STM32 se configure para utilizar el TIMER 4. La funcionalidad del TIMER 4 será la de un PWM. Para obtener un resultado óptimo sin parpadeos visibles, se desea obtener una frecuencia de operación de 40 KHz. Para conseguir esta frecuencia, se establece un preescaler de 20. Con ello se consigue dividir la frecuencia de APB1, de 84Mhz a 4,2MHz. Se configura el registro de Periodo del timer 4 a 100. El motivo para configurarlo a 100, es que de esta forma se vuelve a dividir la frecuencia de 4.2Mhz a 42Khz. Por tanto, conseguimos una frecuencia fija de 42 Khz. Se activa la auto carga del registro de periodo, para que una vez finalizada la cuenta de 0 a 100 en el registro de periodo, vuelva a comenzar, y se establece inicialmente en el registro de comparación un valor de 50, para tener un duty cycle de 50%. Esto quiere decir que inicialmente, el 50% de λ estará el pulso en alto, y a la otra parte en bajo. Este valor del registro de comparación ira variando con una frecuencia de 2 segundos (más adelante) para conseguir el efecto dimming.
- SPI: El STM32 se configura para utilizar el periférico SPI1 Se configura para que funcione en modo master, en modo 3 (Lógica en 1, y polaridad en 1). El tamaño de palabra es 8, y el sentido es del bit más significativo al bit menos significativo. Además, se configura para que opere a 2 MHz.

3 SOFTWARE

3.1 Descripción de cada uno de los módulos del sistema

En primer lugar quiero expresar claramente mis ideas y opiniones sobre la estructuracion de un proyecto de esta indole. En un programa embedded hay (en mi opinion) 3 capas.

1. La primera capa es la capa de configuracion y capa base. Aqui es donde se configuran los perifericos del microcontrolador y se inicializa el sistema. En esta capa se crean ademas las funciones de acceso para la siguiente capa

2. La segunda capa es la capa de software de abstraccion. Esta capa provee de metodos y funciones que no dependen del microcontrolador ni de su hardware. Tan solo usan funciones y metodos que les ha provisto la anterior capa.

3. La ultima capa es la capa de usuario y de programa. Aqui es donde la logica del proyecto toma lugar

Tener esta distincion permite organizar el proyecto y da lugar a un software limpio y mantenible

3.1.1 Ficheros centrales

Los ficheros centrales son ficheros comunes, usualmente headers (.h) a los cuales todos los modulos del sistema tienen acceso, con el objetivo de establecer reglas y servicios comunes. Ademas de todas las librerias que STM y ARM proveen, he incluido yo mismo algunas. A continuacion se nombran estos ficheros y los servicios que proveen:

- **sbm_err.h:** Define una serie de codigos de error que pueden devolver las funciones de los modulos y el programa. Ademas define unos macros con los que se puede assertar y notificar a traves de stderr fallos que pueda estar sucediendo en el sistema
-

3.1.2 Biblioteca BH1750

Se ha creado una biblioteca que provee funciones de control y lectura del sensor BH1750. Esta biblioteca ha sido creado bajo la premisa "**Hardware Agnostic**". Esto nos permite utilizar la biblioteca en otros sistemas y/o programas que utilizan otro microcontrolador, un HAL distinto o ni siquieran utilizan una capa de abstraccion (Control directo del periferico I2C). Para conseguir este objetivo se utilizan funciones de acceso al periferico, que se pasan a un "manejador" (handle) que la biblioteca utilizara en cada uno de sus operaciones.

3.1.3 Modulo LUM

El modulo LUM hace uso de la biblioteca BH1750. Realiza la inicializacion del periferico I2C, provee las funciones de acceso al periferico I2C a la biblioteca, e inicializa un hilo que hara las medidas oportunas según el estado en el que se encuentra el sistema.

3.1.4 Modulo COM

El modulo COM es un modulo especifico del sistema cuya funcion es ocuparse de las comunicaciones con el Host. Para ello emplea el periferico USART a una velocidad de 9600 Baudios, 8 bits de datos, 1bit de stop y sin paridad. El modulo COM puede recibir y transmitir datagramas con una estructura predefinida y longuitud variable. Si el datagrama cumple los controles de validacion y no es descartada, se introduce en una cola de mensajes, donde otro hilo ejecuta la funcion que esta asociado con el ID del datagrama.

3.1.5 Modulo CLOCK

Mantiene la hora gracias a un timer del sistema operativo que llama un callback que incrementa la hora por 1 segundo. Además, incluye una función para cambiar la hora. La hora está disponible globalmente en todo el proyecto.

3.1.6 Modulo Joystick

El estado de los pines del joystick se comprueba y los eventos se envían al hilo principal mediante señales. El hilo del joystick (joystick_thread) espera las señales de interrupción y filtra los eventos del joystick, como pulsaciones largas y cortas. Estos eventos se colocan en una cola de mensajes (queueId) para su procesamiento posterior. La función init_joystick_proc inicializa el hilo del joystick y configura los pines GPIO necesarios para detectar las interrupciones.

3.1.7 Modulo led_control

El código configura y controla los LEDs RGB. La función conf_led_control inicializa los pines GPIO como salidas para los LEDs RGB. Se ocupa de encender y apagar los leds según los comandos que reciba por la cola de mensajes

3.1.8 Modulo Principal

El módulo principal gestiona la máquina de estados del dispositivo y coordina las tareas entre varios hilos, incluyendo "lcd", "com", "LUM", "RGB", "Joystick" y "clock". Se encarga de interpretar comandos recibidos por "joystick" y "COM" para actualizar el estado de la máquina de estados según la lógica interna. Además, controla el funcionamiento de otros hilos utilizando Flags, colas de mensajes y funciones específicas según el estado y los comandos recibidos. Internamente, gestiona un buffer circular para almacenar las medidas más recientes.

3.1.9 Modulo PWM

Configura el TIMER 1 para producir una señal PWM por el canal 1 (GPIO PE9). Recibe la señal de modulacion por un cola de trabajo desde el Hilo principal.

3.2Descripción global del funcionamiento de la aplicación.

En el programa hay principalmente 3 automatas. Estos automatas estan representados (malamente...) en los siguientes diagramas.

El primer y principal automata es el del programa principal. Tiene 4 estados.

- Estado REPOSO
- Estado MANUAL
- Estado Automatico
- Estado Programacion

A cada uno de los estados se puede acceder desde el anterior estado con el evento JOY_CENTER_SPULSE, que corresponde con un pulso corto en el Pulsador central del Joystick.

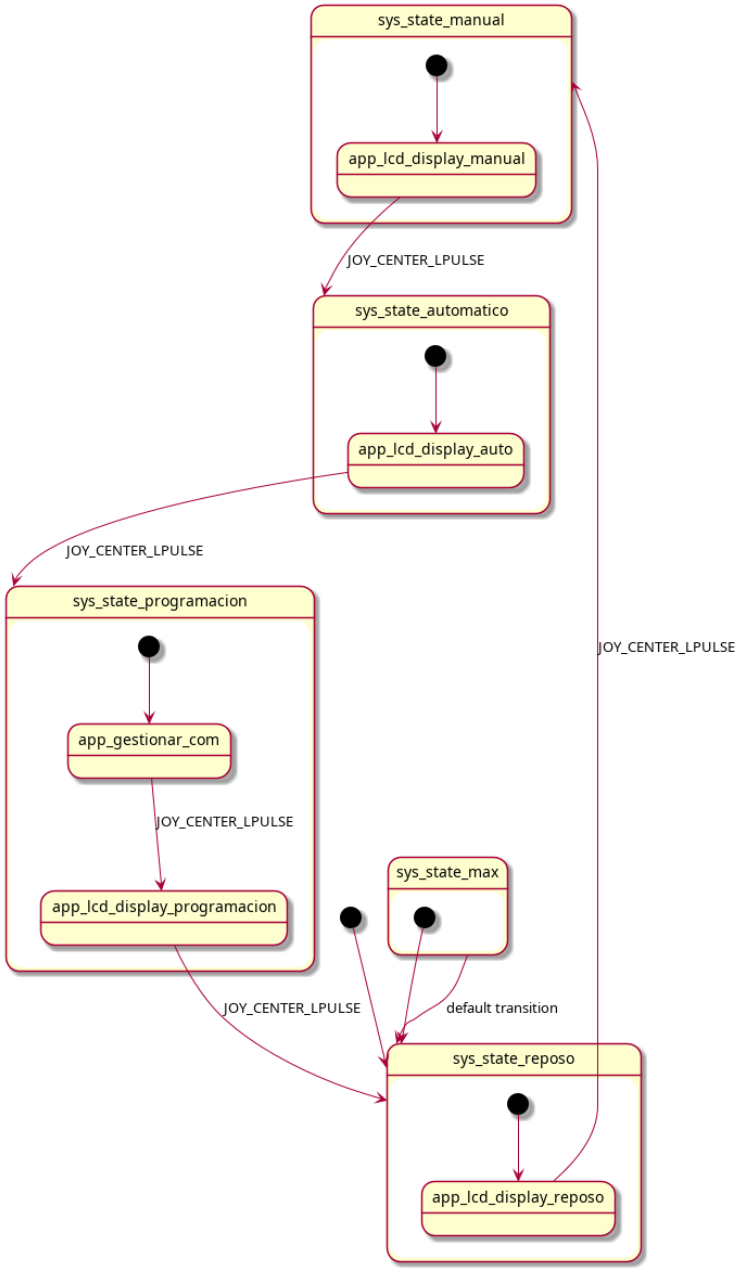


Figure 1: Maquina de estados Programa Principal

El siguiente Automata es la maquina de estados del modo programacion. Segun los eventos que reciba del modulo JOYSTICK, el Modo programacion actuara de una manera u otra. Mover el joystick a la derecha o a la izquierda permitira seleccionar, un elemento en el display, Mover el joystick arriba o abajo, permite incrementar o decrementar el valor. Estos cambios se producen con cambios cortos en el joystick. Para fijar los cambios, es necesario realizar un pulso corto en el Joystick central.

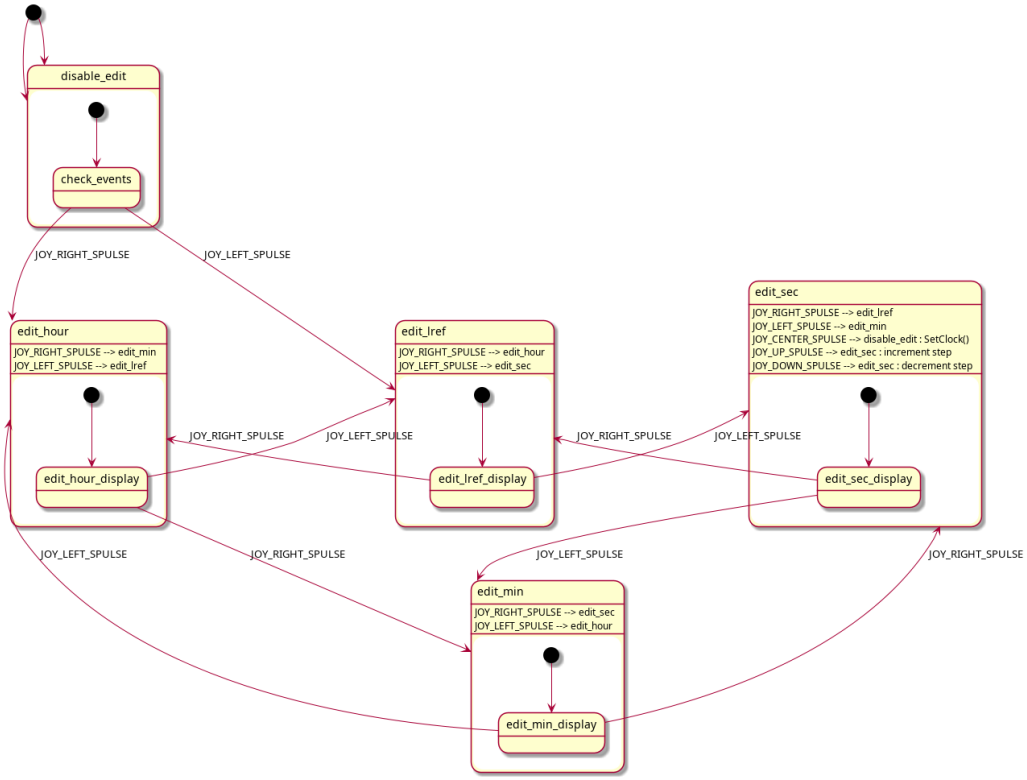


Figure 2: Maquina de estados de Modo Programacion

Por ultimo esta la maquina de estados del decodificador COM. Para evitar usar los recursos del sistema se de forma continuada se ha buscado una solucion, que trata de estar solamente activo cuando se detecta el Start Of Header (SOH).
Inicia en COM_DECODER_INIT, y solamente espera al indicador de inicio de trama SOH
Cuando lo detecta pasa a COM_DECODER_WAIT_HEADER e incrementa el buffer de recepcion a 2 bytes. En este estado:
Cuando se vuelve a llenar el buffer, se incrementa el buffer por el tamaño que dice la cabezera que tendra el datagrama. Y se pasa al estado COM_DECODER_WAIT_DATA
Una vez se llena el buffer por ultima vez, se comprueba que no se ha cometido ningun error y que el datagrama cierra con End of File (EOF).
Si en cualquiera de estos estados se produce un error, se vuelve al estado COM_DECODER_INIT

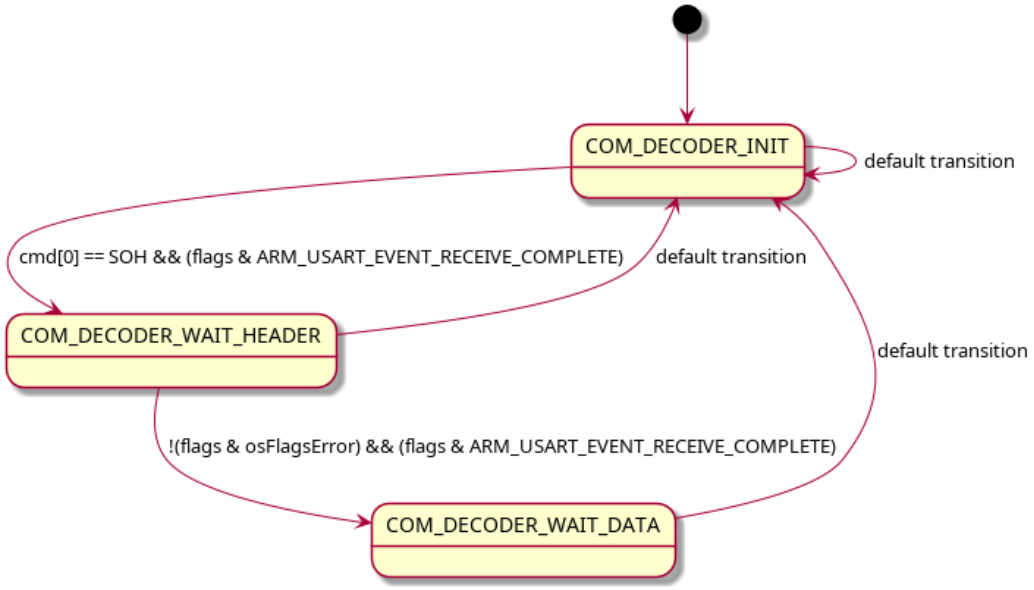


Figure 3: Maquina de estados Decodificador COM

3.3Descripción de las rutinas más significativas que ha implementado.

3.3.1 JOYSTICK

El modulo joystick configura los GPIO conectados al joystick para emitir Interrupciones al sistema, que notifican al hilo del joystick a traves de flags de hilo. El hilo al recibir el flag de la interrupcion, inicia un temporizador de sistema que leera 6 veces en total cada 75 ms cada uno de los pines y notificara al hilo principal del estado de los pines. Al finalizar la lectura multiple de los pines, se pasa el array leido por una mascara. Si todas las lecturas dan positivo, se mandara una notificacion al hilo principal de que ha sucedido una pulsacion larga en cierto pin. En cambio, si al filtrar el array, solo estan de 3 lecturas a 5 lecturas positivas (en sentido ascendente), se informara al hilo principal que solo se ha realizado una pulsacion corta. En cualquier otro caso no se notificara al hilo principal

3.3.2 CLOCK

Mantiene la hora en un struct formado por Hora, Minutos y Segundos. Un Timer de sistema hace de contador de segundos.

3.3.3 LCD

El Modulo LCD se ocupa de rellenar un buffer que representa el sistema grafico, de los caracteres y diagramas que se desean mostrar según el estado del sistema. Ademas se ocupa de la comunicación con el LCD

3.3.4 LUM

Configura el sensor BH1750 para que haga muestras continuas. Una vez cada segundo lee el registro del sensor y manda el valor por una cola de mensajes al hilo principal.

3.3.5 COM

Configura el periferico USART y se ocupa de aceptar y enviar datagramas con una estructura predefinida. Para ello hace uso de una maquina de estados. Esta seccion es critica, por tanto se sube la prioridad de hilo a la maxima.

4 DEPURACION Y TEST

4.1Pruebas realizadas.

Para todos los modulos se ha realizado un testbench donde se podia testear el funcionamiento del modulo mientras se desarrollaba. Para conseguir datos de los testbench se crearon MACROS y un sistema de Logs por SWO (Serial Wire Output) donde se podia depurar el codigo y ver cuando se producia un error. Sin embargo no he cumplido dos objetivos que deberian ser imprescimdibles para estas MACROS:

- Thread Safe:
  Cuando hay multiples hilos operando a la vez, y usan la depuracion serie, se pueden entremezclar los Logs, por tanto no es thread safe, e inutiliza el sentido de esta funcion.
- Deshabilitables por modulo
  La idea es que con los macros se pudiera discernir entre los diferentes tipos de salida y su proveniencia. Ademas debian ser deshabilitables por el fichero donde estuvieran situados

Cada uno de los modulos tiene un testbench en la carperta “test_apps”. Al ejecutar el susodicho testbench, se comproara la inicializacion correcta del modulo a testear y se podran observar en la consola o en el periferico su salida, tal como si estuviera en el programa final.

Idealmente se deberian hacer unity-tests, que establecen los requisitos iniciales y finales, comprueban que en la ejecucion en un sandbox del modulo, se siguen cumpliendo ls requisitos y permite la evolucion del modulo por features, y asegurando al desarrollador que cada vez que realiza un cambio en el modulo, este puede ser testeado antes de ser empujado a “produccion”, y que si hay un cambio que rompa con los requisitos, el programador sera notificado.