EPS

Asignatura: G0460028 Sistemas Empotrados

Curso: 2024/2025	Examen: Extraordinaria	Fecha: 01-07-25
Semestre: 1	Convocatoria: Extraordinaria	

Sistemas Empotrados

Duración: 2 horas
Materiales permitidos: Apuntes personales en papel
Instrucciones: El examen consta de dos partes – Teórica (4 puntos) y Práctica (6 puntos) – con un total de 10 puntos.

- Responda a las preguntas teóricas de manera clara y concisa.
- Se valorará la claridad, la justificación de las decisiones y el correcto uso de los conceptos aprendidos durante el curso.
- Para las preguntas prácticas,
    - El código debe ser pseudocódigo o C/C++ enfocado a la plataforma YD-ESP32 (basada en ESP32),
    - No exceder 10-20 líneas por ejercicio.
    - Puede omitirse cabecera de includes y configuración extensa, centrándose en el fragmento esencial que demuestre la competencia solicitada.

PARTE TEÓRICA (4 puntos)

1. **Arquitectura de memoria (1 pto):** La tarjeta YD-ESP32 se basa en el microcontrolador ESP32. Explique brevemente los tipos de memoria disponibles en el ESP32 (*IRAM, DRAM, Flash*) y su uso típico en una aplicación embebida. Además, ¿por qué a veces es preferible ubicar el código de las rutinas de interrupción en IRAM en lugar de Flash?

2. **Polling vs. interrupciones (1 pto):** Compare el uso de un bucle de polling, por ejemplo, usando `millis()` frente a las rutinas de servicio de interrupción (ISR). Mencione 2 ventajas y 2 desventajas de cada enfoque, y dé un ejemplo de cuándo escogería uno u otro en la YD-ESP32.

3. **Modo Deep Sleep (1 pto):** La YD-ESP32 suele usarse en aplicaciones de bajo consumo. Describa el modo *Deep Sleep* del ESP32 y cómo se puede despertar la placa de este modo (al menos dos fuentes de despertador). ¿Qué sucede con el estado de las variables normales del programa después de entrar a Deep Sleep?

4. **Generación de PWM (1 pto):** El ESP32 puede generar señales PWM (Pulse Width Modulation) mediante su hardware. Explique cómo se genera una señal PWM (indique qué son el *duty cycle* y la frecuencia de PWM) y cite una aplicación práctica en la que se usaría PWM en la plataforma YD-ESP32.

5. **I2C vs. SPI (1 pto):** La YD-ESP32 dispone de periféricos I2C y SPI libres. Supongamos que se debe conectar un sensor de temperatura digital de bajo costo con solo unos pocos bytes de datos. ¿Qué protocolo elegiría (I2C o SPI) y por qué? Menciona 1 ventaja de tu elección y 1 situación donde preferirías el otro protocolo.

PARTE PRÁCTICA (6 puntos)

Nota

- *Cada ejercicio práctico vale 1 punto.*
- *Se pide pseudocódigo o C/C++ con funciones orientadas a YD-ESP32.*
- *Se valorará la corrección, la síntesis y la coherencia del fragmento de código.*
- *No exceda las 20 líneas por ejercicio.*

1. **Interrupción por pulsador (1 pto):** La YD-ESP32 tiene un pulsador conectado al pin `GPIO0` (activo con **nivel bajo**). Se desea que cada vez que se pulse el botón se encienda un LED (en `GPIO2`) durante 2 segundos. A continuación, se muestra un fragmento de código incompleto. Indique qué código falta en la rutina de interrupción para lograr el objetivo, y explique por qué no es correcto usar `delay(2000)` dentro de la ISR

```
#define BUTTON_PIN 0
#define LED_PIN    2

void IRAM_ATTR handleButtonISR() {
    // … completa aquí …









}

void setup() {
  pinMode(LED_PIN, OUTPUT);
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  attachInterrupt(BUTTON_PIN, handleButtonISR, FALLING);
}
```

2. **PWM para atenuación (1 pto):** Usando la funcionalidad PWM del ESP32 (LED Control PWM, ledc), escriba un fragmento de código en pseudocódigo o C++ (máx. 12 líneas) que genere un **parpadeo suave** de un LED conectado al pin `GPIO15`. El LED debe aumentar gradualmente su brillo de apagado a encendido, luego disminuirlo, en un bucle continuo. Indique cómo se configuran canal, frecuencia y resolución en el ESP32

3. **Comunicación ADC (1 pto):** En la YD-ESP32 hay conectado un potenciómetro al pin `ADC1_CHANNEL_0 (GPIO36)`. Escriba un pseudocódigo breve (máx. 10 líneas) que lea el valor analógico, lo convierta a voltaje (suponiendo referencia de 3.3V), y encienda otro LED (`GPIO4`) si el voltaje supera 1.5V. Considere que `analogRead(pin)` devuelve un entero de 0 a 4095

4. **Blink no bloqueante (1 pto):** Sin usar `delay()`, escribe en C++ un bucle principal que haga parpadear un LED en `GPIO16` cada 500 ms usando `millis()`. No incluyas configuración de pin, solo la lógica de temporización.

5. **Profundizar en Deep Sleep (1 pto):** Se quiere ahorrar energía usando *Deep Sleep*. Proponga un pseudocódigo (máx. 15 líneas) que encienda un LED (`GPIO5`) durante 1 segundo y luego ponga el ESP32 en *Deep Sleep* durante 10 segundos. Además, indique cómo configuraría la fuente de despertador por tiempo.