UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CAMPUS DE GANDIA

GRADO EN TECNOLOGIAS INTERACTIVAS

Microprocesadores y Acondicionadores de señal **Examen Prueba 1 del 6 de noviembre de 2025**

Apellidos: Aula: Columna: Fila:

Nombre: DNI:

Conteste a los diferentes apartados en las casillas habilitadas para la respuesta. Se proporciona, junto con el examen, un anexo de funciones que pueden resultarle útiles para completar los apartados que requieran codificar.

EJERCICIO 1 (1.5 Puntos)

Se desea programar el dispositivo M5Stack para que cambie el color del fondo de la pantalla cada vez que el usuario presione el botón A.

Apartado 1.1) (0.5 puntos) Completa la línea de código señalada en los comentarios según la información del Anexo.

```
#include <M5Stack.h>

#define BOTON_A_PIN 39  // GPIO del botón A

volatile bool interrupcionDetectada = false;
bool colorAzul = true;

void IRAM_ATTR cambiarColor() {
  interrupcionDetectada = true;
}

void setup() {
  M5.begin();
  M5.Lcd.setTextSize(2);
  M5.Lcd.fillScreen(BLACK);
  M5.Lcd.setCursor(10, 10);
  M5.Lcd.println("Presiona Boton A para cambiar color");

  pinMode(BOTON_A_PIN, INPUT);

  // Configurar la interrupción en flanco descendente (cuando se presiona el botón A)
 attachInterrupt(digitalPinToInterrupt(BOTON_A_PIN), cambiarColor, FALLING);

}

void loop() {
  if (interrupcionDetectada) {
   noInterrupts();
      if (colorAzul) {
      M5.Lcd.fillScreen(BLUE);
      M5.Lcd.setTextColor(WHITE);
      M5.Lcd.setCursor(60, 100);
      M5.Lcd.println("COLOR AZUL ACTIVO");
    } else {
      M5.Lcd.fillScreen(GREEN);
      M5.Lcd.setTextColor(BLACK);
      M5.Lcd.setCursor(60, 100);
      M5.Lcd.println("COLOR VERDE ACTIVO");
    }

    colorAzul = !colorAzul;

    interrupcionDetectada = false;
    interrupts();

    delay(300);
  }
}
```

Apartado 1.2) (0.5 puntos) ¿Por qué en el código anterior se declara la variable global como "volatile"?

a) Para evitar que la variable se modifique desde dentro de la función loop().
b) Porque la variable cambia su valor dentro de una rutina de interrupción, y el compilador debe leer siempre su valor actual en memoria.
c) Para que la variable se almacene en la memoria flash y no en la RAM.
d) Para impedir que se use dentro de una función attachInterrupt().

Apartado 1.3) (0.5 puntos) En el código anterior, ¿qué problemas podrían ocurrir si se eliminan las líneas **noInterrupts(); y interrupts();**? Explica brevemente por qué son necesarias.

Si se eliminan, podrían producirse nuevas interrupciones mientras el programa aún está ejecutando el código asociado a la interrupción anterior. Esto puede provocar inestabilidad, valores incorrectos en variables compartidas o comportamientos impredecibles (por ejemplo, que el color cambie varias veces sin control).

Se usan para proteger una sección crítica, desactivando (noInterrupts()) y luego reactivando (interrupts()) las interrupciones.

EJERCICIO 2 (1.5 Puntos)

Se desea capturar la señal analógica de salida de un potenciómetro conectado al GPIO34 (canal 6) del ADC1 del ESP32. La tensión de referencia del sistema es 3.9 V, y el ADC1 se configura con una precisión de 12 bits.

Apartado 2.1) (1 punto) Configura el ADC1 utilizando la librería de esp-idf proporcionadas en el Anexo del ADC. Declara la variable lectura_adc con el valor obtenido del canal configurado.

```
// Configura ADC1 con resolución de 12 bits
adc1_config_width(ADC_WIDTH_BIT_12);

// Configura el canal 6 (GPIO34) con atenuación de 11 dB (rango 0-3.9 V)
adc1_config_channel_atten(ADC1_CHANNEL_6, ADC_ATTEN_DB_11);

// Lee el valor del ADC
int lectura_adc = adc1_get_raw(ADC1_CHANNEL_6);
```

Apartado 2.2) (0.5 puntos) Calcula el valor digital que leerá el ADC1 cuando la entrada analógica tenga una tensión de 2.5 V.

$Vd = Va \times (2^{12} - 1) / 3.9$

$Vd = 2.5 \times (4095) / 3.9 = 2.5 \times 1050 = 2625$

EJERCICIO 3 (1.5 Puntos)

Se requiere generar con el DAC1 (GPIO25) del ESP32 dos tensiones analógicas consecutivas de 1.2 V y 2.7 V, con una pausa de 500 ms entre ambas. La tensión máxima del DAC es 3.3 V.

Apartado 3.1) (0.5 puntos) Calcula los valores digitales que deben enviarse al DAC1 para obtener las tensiones indicadas.

Para Va1 = 1.2 V → Vd1 = (1.2 × 255) / 3.3 = 92.7 ≈ 93
Para Va2 = 2.7 V → Vd2 = (2.7 × 255) / 3.3 = 208.6 ≈ 209

Por tanto, los valores digitales a aplicar son 93 y 209.

Apartado 3.2) (1 punto) Escribe el código básico necesario (setup y loop) utilizando la librería esp-idf proporcionadas en el Anexo del DAC.

```
void setup() {
  //habilita el DAC que se va a usar
  dac_output_enable(DAC_CHANNEL_1); // GPIO25
}

void loop() {
  dac_output_voltage(DAC_CHANNEL_1, 93);   // valor digital de 1.2 V
  delay(500);
  dac_output_voltage(DAC_CHANNEL_1, 209);  // valor digital de 2.7 V
  delay(500);
}
```