Lab 1: Introducción al ESP32-C6

SEU-MEI (FIB) Laboratory Assignment

Mario Ventura Burgos

Jofre Coll Vila

Índice

1. Introducción..... 3
2. Solución planteada..... 4
    2.1. Caso 1..... 4
    2.2. Caso 2..... 5
    2.3. Caso 3..... 6
3. Referencias..... 8

1. Introducción

En este primer trabajo de laboratorio, se pretende familiarizarse con el IDE del SoC ESP32-C6 y trabajar conceptos de E/S digitales (GPIOs), timers o detección de flancos para encuesta periódica.

Para ello, se desea realizar una máquina sencilla de estados que, en función del estado detectado en una entrada digital GPIO0, modifique una salida digital GPIO4, donde habrá un LED.

Se debe implementar el código y un diagrama de estados para cada uno de los siguientes casos.

1. Por cada pulsación de la entrada digital, se modificará el estado del LED
2. Para cada pulsación de la entrada digital que sea suficientemente larga (pulsación durante > 0’5s), se modificará el estado del LED.
3. Una pulsación suficientemente larga (>0’5s) provoca una rápida intermitencia del LED durante 10 segundos. Si se vuelve a detectar una pulsación larga, la intermitencia se cancela.

Todos los algoritmos implementados se encuentran en el repositorio siguiente:

Embedded-Systems/LAB 1 - ESP32C6/ (privado, es necesario solicitar acceso)

2. Solución planteada

Para cada uno de los casos descritos en la introducción, se plantea la siguiente solución:

2.1. Caso 1

Para el primer caso planteado, en el que la pulsación del botón permuta el estado del LED, se ha planteado el código que puede verse en el archivo A.c (adjuntado en el zip de la entrega). Este código no se detalla en este informe con el objetivo de no extenderlo en exceso, pero a continuación se explica cuál es el planteamiento y qué hace este código.

En primer lugar, dado que la entrada y salida digital son, respectivamente, GPIO0 y GPIO4 (General Purpose Input Output 0 y 4), se declaran dos constantes con estos valores.

```
#define BTN_PIN 0  // Pin del botón
#define LED_PIN 4  // Pin del LED
```

Además, se establece el estado inicial del led como apagado y se crea una variable para almacenar el último estado del botón.

```
int led_on = 0;         // (0 = apagado, 1 = encendido)
int last_btn = 0;
```

En un bucle infinito (***while(1)***), se obtiene el estado del botón mediante la función ***gpio_get_level(BTN_PIN)*** y se establece el nuevo estado del LED mediante la permutación del estado actual de LED (***led_on = !led_on***). Esto se hace cada vez que se detecta que el usuario pulsa el botón y es visible inmediatamente en el LED del circuito.

```
while (1) {
      int btn_now = gpio_get_level(BTN_PIN);  // Leer estado botón

      // Si sueltas el botón (de 0 a 1), cambiar el LED
      if (btn_now == 1 && last_btn == 0) {
            led_on = !led_on;                 // Cambiar estado
            gpio_set_level(LED_PIN, led_on);
            printf("Botón pulsado -> LED %s\n", led_on ? "ON" : "OFF");
      }

      last_btn = btn_now;  // Guardar estado actual

      // Pausa de 50ms para evitar rebotes
      vTaskDelay(pdMS_TO_TICKS(50));
}
```

Mientras no se detenga la ejecución, se puede pulsar el botón tantas veces como se desee, encendiendo y apagando el LED con cada pulsación.

```
Botón pulsado -> LED ON
Botón pulsado -> LED OFF
Botón pulsado -> LED ON
Botón pulsado -> LED OFF
Botón pulsado -> LED ON
```

Para este primer caso, se presenta el siguiente diagrama de estados:

2.2. Caso 2

En este segundo caso, el código es prácticamente idéntico ya que el objetivo es el mismo, con la diferencia de que en este caso la permutación del estado del LED se realizará únicamente si la pulsación del botón dura más de 0,5s.

Por tanto, el único cambio a nivel de código se verá reflejado en el fragmento en que se tratan las pulsaciones del botón, ya que ahora es necesario detectar la duración de estas pulsaciones.

Para detectar la duración de las pulsaciones, el planteamiento es el siguiente:

1. Se obtiene la **hora exacta** (en microsegundos) en la que se detecta el **flanco de bajada**, es decir, el inicio de la pulsación. Esto se hace con la función `esp_timer_get_time()` [1].
2. Se obtiene la hora exacta del flanco de subida, es decir, el fin de la pulsación.
3. Se restan las horas (fin de pulsación - inicio de pulsación) y se obtiene la duración de esta.
4. Se comprueba si la duración obtenida supera los 500.000 microsegundos (0'5 segundos) y, en caso de ser así, se permuta el estado del LED.

```
// Detectar flanco de bajada (inicio de pulsación)
if (button_state == 0 && last_button_state == 1) {
      press_start_time = esp_timer_get_time(); // tiempo en µs
      button_pressed = 1;
}

if (press_duration > 500000 && last_button_state == 1) {
      led_state = !led_state
      gpio_set_level(LED_GPIO, led_state);
}
```

Para este segundo caso, el diagrama de estados es el mismo que en el apartado anterior, pero las flechas que indican las posibles acciones a realizar para provocar un cambio de estado ahora reflejan la condición establecida para la duración de la pulsación.

2.3. Caso 3

Por último, para el tercer caso expuesto en el enunciado, se pretende hacer lo mismo que en el caso anterior, pero en lugar de simplemente permutar el estado del LED, cada pulsación de duración superior a 0’5 segundos generará una intermitencia rápida en el LED durante 10 segundos. Pasados los 10 segundos, la LED se apaga automáticamente.

Además, si se vuelve a hacer una pulsación de más de 0’5 segundos antes de que transcurran los 10 segundos, la LED se apagará.

Para generar una intermitencia rápida durante 10 segundos, simplemente se actualiza el estado a ***STATE_BLINKING*** (definido en el propio algoritmo, al igual que el resto de estados) y se obtiene la hora (en microsegundos) en la que se inició la intermitencia. Siguiendo la lógica del apartado anterior para medir duraciones, se esperan 10 segundos y se vuelve a actualizar el estado a ***STATE_NORMAL***.

```
// Entrar en modo intermitencia
state = STATE_BLINKING;
blink_start_time = esp_timer_get_time();
last_toggle_time = blink_start_time;
printf("Pulsación larga -> INICIO intermitencia\n");
```

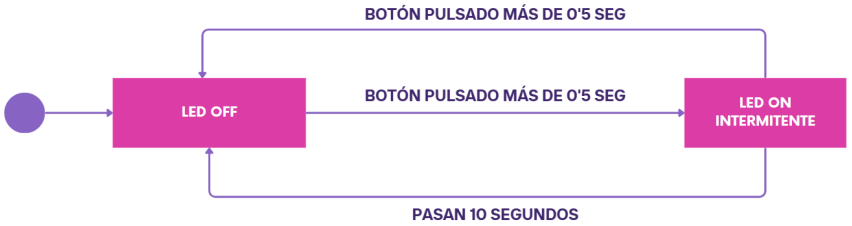
```
// Cancelar automáticamente tras 10s
if ((now - blink_start_time) > 10000000) { // 10s en microsegundos
      state = STATE_NORMAL;
      gpio_set_level(LED_GPIO, 0);
      printf("Fin de intermitencia automática\n");
}
```

Como el código está constantemente escuchando al botón, si se detecta una pulsación de más de 0’5 segundos mientras el LED está en estado ***STATE_BLINKING***, se apaga el LED automáticamente.

Para este caso, el diagrama de estados presentado es el que se muestra a continuación.

DIAGRAMA CASO 3

Cada pulsación conmuta el estado del LED generando intermitencias



En este caso, el conjunto de estados es distinto ya que cuando el LED está encendido es de forma intermitente. Una vez en este estado, se puede volver al estado de LED OFF con dos acciones diferentes:

1. Pulsando el botón durante más de 0’5 segundos.
2. Esperando a que pasen 10 segundos.

3. Referencias

[1] ESP Timer (High Resolution Timer)